

Computer Networks
RA1911030010014
Experiment - 9
Remote Command Execution Using UDP

Aim: To create a Remote Command Execution Using UDP.

Server Code:

```
/**
 * RA1911030010014 - RCE Server
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <netinet/in.h>
#include <string.h>
#include <sys/stat.h>
#include <arpa/inet.h>
#include <unistd.h>

#define MAX 1000
#define PORT 8014

int main()
{
    int serverDescriptor = socket(AF_INET, SOCK_DGRAM, 0);
    int size;
    char buffer[MAX], message[] = "Command Successfully executed !";
    struct sockaddr_in clientAddress, serverAddress;
    socklen_t clientLength = sizeof(clientAddress);
    bzero(&serverAddress, sizeof(serverAddress));
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
    serverAddress.sin_port = htons(PORT);
    bind(serverDescriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));
    while (1)
    {
        bzero(buffer, sizeof(buffer));
        recvfrom(serverDescriptor, buffer, sizeof(buffer), 0, (struct sockaddr
*)&clientAddress, &clientLength);
        system(buffer);
        printf("Command Executed ... %s ", buffer);
        sendto(serverDescriptor, message, sizeof(message), 0, (struct sockaddr
*)&clientAddress, clientLength);
    }
    close(serverDescriptor);
    return 0;
}
```

Client Code:

```
/**
 * RA1911030010014 - RCE Client
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <unistd.h>
#include <netdb.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>

#define MAX 1000
#define PORT 8014

int main()
{
    int serverDescriptor = socket(AF_INET, SOCK_DGRAM, 0);
    char buffer[MAX], message[MAX];
    struct sockaddr_in cliaddr, serverAddress;
    socklen_t serverLength = sizeof(serverAddress);
    bzero(&serverAddress, sizeof(serverAddress));
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_addr.s_addr = inet_addr(INADDR_LOOPBACK);
    serverAddress.sin_port = htons(PORT);
    bind(serverDescriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));
    while (1)
    {
        printf("\nCOMMAND FOR EXECUTION ... ");
        fgets(buffer, sizeof(buffer), stdin);
        sendto(serverDescriptor, buffer, sizeof(buffer), 0, (struct sockaddr *)&serverAddress,
serverLength);
        printf("\nData Sent !");
        recvfrom(serverDescriptor, message, sizeof(message), 0, (struct sockaddr
*)&serverAddress, &serverLength);
        printf("UDP SERVER : %s", message);
    }
    return 0;
}
```

Output:

client.c

```
1 /**
2  * RA1911030010014 - RCE Client
3  */
4
5 #include <sys/types.h>
6 #include <sys/socket.h>
7 #include <stdio.h>
8 #include <unistd.h>
9 #include <netdb.h>
10 #include <netinet/in.h>
11 #include <string.h>
12 #include <arpa/inet.h>
13
14 #define MAX 1000
15 #define PORT 8014
16
17 int main()
18 {
19     int serverDescriptor = socket(AF_INET, SOCK_DGRAM, 0);
20     char buffer[MAX], message[MAX];
21     struct sockaddr_in cliaddr, serverAddress;
22     socklen_t serverLength = sizeof(serverAddress);
23     bzero(&serverAddress, sizeof(serverAddress));
24     serverAddress.sin_family = AF_INET;
25     serverAddress.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
26     serverAddress.sin_port = htons(PORT);
27     bind(serverDescriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));
28     while (1)
29     {
30         printf("\nCOMMAND FOR EXECUTION ... ");
31         fgets(buffer, sizeof(buffer), stdin);
32         sendto(serverDescriptor, buffer, sizeof(buffer), 0, (struct sockaddr *)&serverAddress,
33             sizeof(serverAddress));
34         printf("\nData Sent !");
35         recvfrom(serverDescriptor, message, sizeof(message), 0, (struct sockaddr *)&serverAddress,
36             sizeof(serverAddress));
37         printf("UDP SERVER : %s", message);
38     }
39     return 0;
40 }
```

38:2 C and C++ Spaces: 4

server.c

```
1 /**
2  * RA1911030010014 - RCE Server
3  */
4
5 #include <sys/types.h>
6 #include <sys/socket.h>
7 #include <stdio.h>
8 #include <stdlib.h>
9 #include <netdb.h>
10 #include <netinet/in.h>
11 #include <string.h>
12 #include <sys/stat.h>
13 #include <arpa/inet.h>
14 #include <unistd.h>
15
16 #define MAX 1000
17 #define PORT 8014
18
19 int main()
20 {
21     int serverDescriptor = socket(AF_INET, SOCK_DGRAM, 0);
22     int size;
23     char buffer[MAX], message[] = "Command Successfully executed !";
24     struct sockaddr_in clientAddress, serverAddress;
25     socklen_t clientLength = sizeof(clientAddress);
26     bzero(&serverAddress, sizeof(serverAddress));
27     serverAddress.sin_family = AF_INET;
28     serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
29     serverAddress.sin_port = htons(PORT);
30     bind(serverDescriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));
31     printf("Server is running on Port %d...\n", PORT);
32     while (1)
33     {
34         bzero(buffer, sizeof(buffer));
35         recvfrom(serverDescriptor, buffer, sizeof(buffer), 0, (struct sockaddr *)&clientAddress,
36             sizeof(clientAddress));
37         system(buffer);
38         printf("Command Executed ... %s ", buffer);
39         sendto(serverDescriptor, message, sizeof(message), 0, (struct sockaddr *)&clientAddress,
40             sizeof(clientAddress));
41     }
42     close(serverDescriptor);
43     return 0;
44 }
```

42:2 C and C++ Spaces: 4

client.c

```
extern in_addr_t inet_addr (const char *__cp) __THROW;
...
RA1911030010014:~/environment/Sep_18/RA1911030010014 $ gcc -o client client.c
RA1911030010014:~/environment/Sep_18/RA1911030010014 $ clear
RA1911030010014:~/environment/Sep_18/RA1911030010014 $ gcc -o client client.c
RA1911030010014:~/environment/Sep_18/RA1911030010014 $ ./client
```

COMMAND FOR EXECUTION ... ls
Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ... ls -alh
Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ...

server.c

```
RA1911030010014:~/environment $ cd Sep_18/RA1911030010014
RA1911030010014:~/environment/Sep_18/RA1911030010014 $ gcc -o server server.c
RA1911030010014:~/environment/Sep_18/RA1911030010014 $ clear
RA1911030010014:~/environment/Sep_18/RA1911030010014 $ gcc -o server server.c
RA1911030010014:~/environment/Sep_18/RA1911030010014 $ ./server
Server is running on Port 8014...
client client.c server server.c
Command Executed ... ls
total 40K
drwxrwxr-x 2 ubuntu ubuntu 4.0K Sep 19 15:31 .
drwxrwxr-x 19 ubuntu ubuntu 4.0K Sep 19 14:27 ..
-rwxrwxr-x 1 ubuntu ubuntu 8.5K Sep 19 15:31 client
-rw-rw-r-- 1 ubuntu ubuntu 1.2K Sep 19 15:30 client.c
-rwxrwxr-x 1 ubuntu ubuntu 8.5K Sep 19 15:31 server
-rw-rw-r-- 1 ubuntu ubuntu 1.3K Sep 19 15:31 server.c
Command Executed ... ls -alh
```

Result:

The required code for the Remote Command Execution Using UDP was written in the AWS Cloud9 environment and successfully compiled.