CS-1631
Computer Networks Lab Project

SECURE SHELL USING WEBSOCKETS

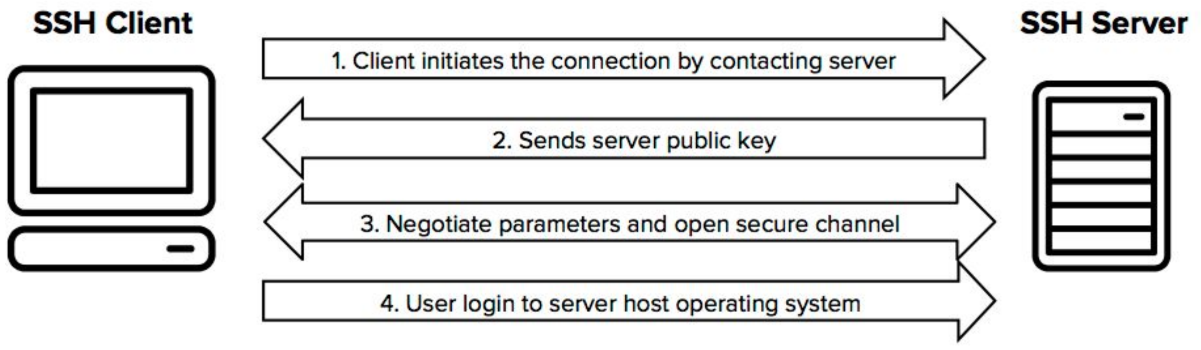
Submitted to

Dr. Rohit Verma
Manipal University Jaipur

Submitted By
Mridu Kejariwal 179301121
Kashish Madan 179202069

Introduction

The SSH protocol uses encryption to secure the connection between a client and a server. All user authentication, commands, output, and file transfers are encrypted to protect against attacks in the network.
Mock SSH using websockets with encryption.


SSH Client
SSH Server
1. Client initiates the connection by contacting server
2. Sends server public key
3. Negotiate parameters and open secure channel
4. User login to server host operating system

SSH works within a network through a client/server architecture. An SSH client is the program that runs SSH protocol from a specific device in order to access remote machines, automate data transfers, issue commands, and even manage network infrastructure. The client/server model means that the network system components being used to establish an SSH secure connection must be enabled for SSH. This can mean installing the proper software, or simply utilizing the SSH services program the computer has built in.

SSH, also known as Secure Shell, is a network protocol that provides administrators with a secure way to access remote servers. SSH establishes a cryptographically secured connection between two parties(client and server), authenticating each side to the other, and passing commands and output back and forth.

Aim

The purpose of our project is to implement a simple SSH using Web Sockets. The server is written in Python and the client in Javascript. User and password authentication is also implemented. The password is encrypted before sending and hashed before storing.

Methodology

Network Concepts Used

1. Web sockets
2. Socket programming
3. Symmetric key encryption
4. Web socket handshaking

Main Modules Developed

Server:

1. *feed* : To accept messages from the client and to perform TCP handshaking on first contact.
2. *sendMessage* : To send a message to the client
3. *validate_username* : To verify that the user is registered in the database
4. *validate_password* : To decrypt the password and to check if it matches with the hashed password stored in the database
5. *execute_command* : To execute the command sent by the client and prepare the output to be delivered.

Client:

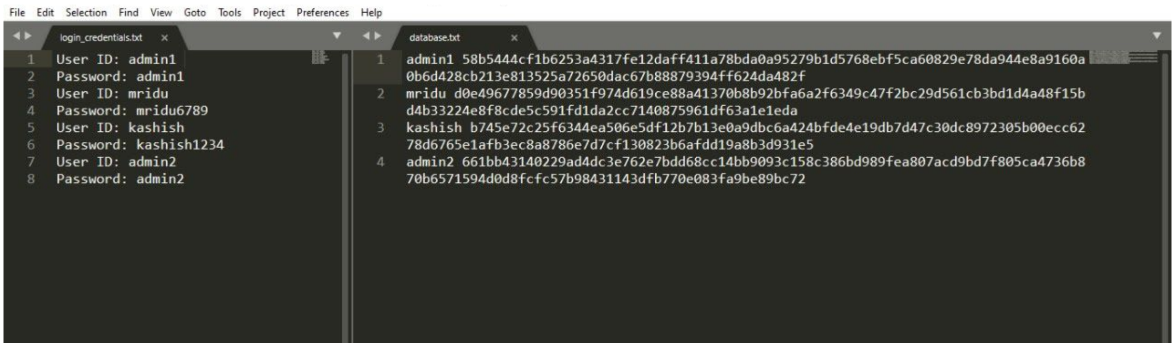
6. *onSubmit* : To send the command entered by the user to the server. Also ensures encryption of the password before sending.
7. *onMessage* : To output the message sent by the server, that is the result of the command sent.

Outputs & Result

A Secure Shell was implemented using websockets with the server and client written in Python and JavaScript respectively.
The user and password are encrypted in database.txt. The password is encrypted and hashed before storing.

For example:

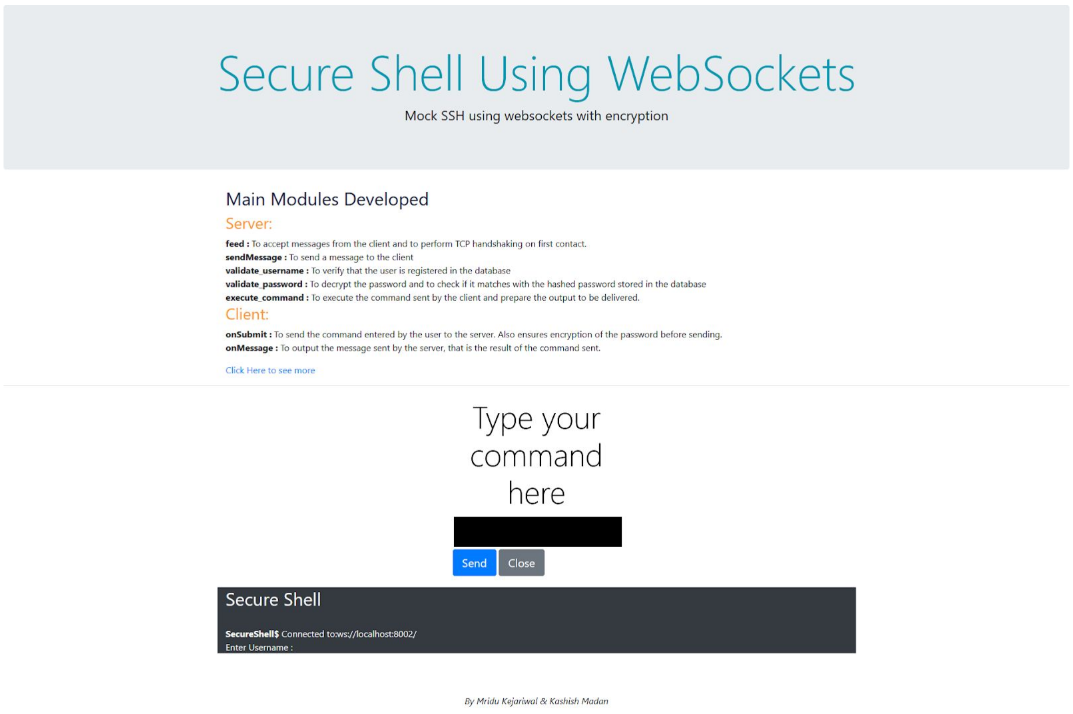
The actual UserID and Passwords are as shown in file login_credentials.txt and the passwords accessed by the server.py are encrypted and stored in database.txt


File Edit Selection Find View Goto Tools Project Preferences Help
login_credentials.txt
database.txt

```
User ID: admin1
Password: admin1
User ID: mridu
Password: mridu6789
User ID: kashish
Password: kashish1234
User ID: admin2
Password: admin2
```

```
admin1 58b5444cf1b6253a4317fe12daff411a78bda0a95279b1d5768ebf5ca60829e78da944e8a9160a0b6d428cb213e813525a72650dac67b88879394ff624da482f
mridu d0e49677859d90351f974d619ce88a41370b8b92bfa6a2f6349c47f2bc29d561cb3bd1d4a48f15bd4b33224e8f8cde5c591fd1da2cc7140875961df63a1e1eda
kashish b745e72c25f6344ea506e5df12b7b13e0a9dbc6a424bfde4e19db7d47c30dc8972305b00ecc6278d6765e1afb3ec8a8786e7d7cf130823b6afdd19a8b3d931e5
admin2 661bb43140229ad4dc3e762e7bdd68cc14bb9093c158c386bd989fea807acd9bd7f805ca4736b870b6571594d0d8fcfc57b98431143dfb770e083fa9be89bc72
```

Client.html



Secure Shell Using WebSockets

Mock SSH using websockets with encryption

Main Modules Developed

Server:

feed : To accept messages from the client and to perform TCP handshaking on first contact.
sendMessage : To send a message to the client
validate_username : To verify that the user is registered in the database
validate_password : To decrypt the password and to check if it matches with the hashed password stored in the database
execute_command : To execute the command sent by the client and prepare the output to be delivered.

Client:

onSubmit : To send the command entered by the user to the server. Also ensures encryption of the password before sending.
onMessage : To output the message sent by the server, that is the result of the command sent.

Click Here to see more

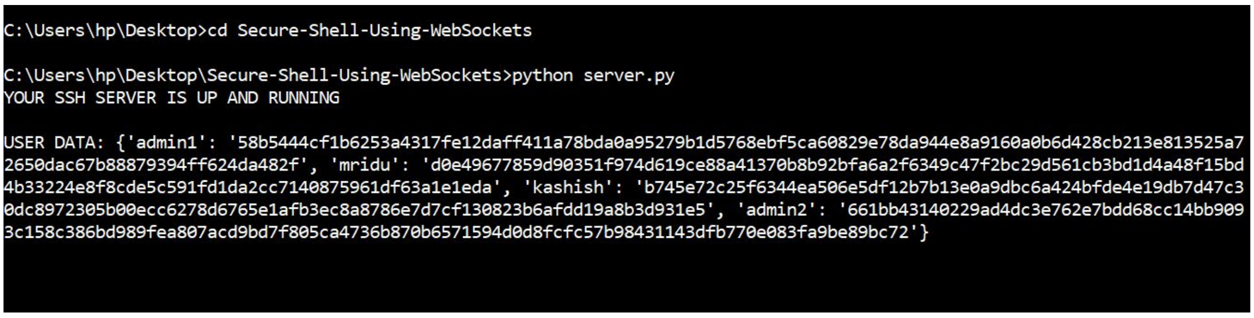
Type your command here

Secure Shell

```
SecureShell$ Connected to:ws://localhost:8002/
Enter Username :
```

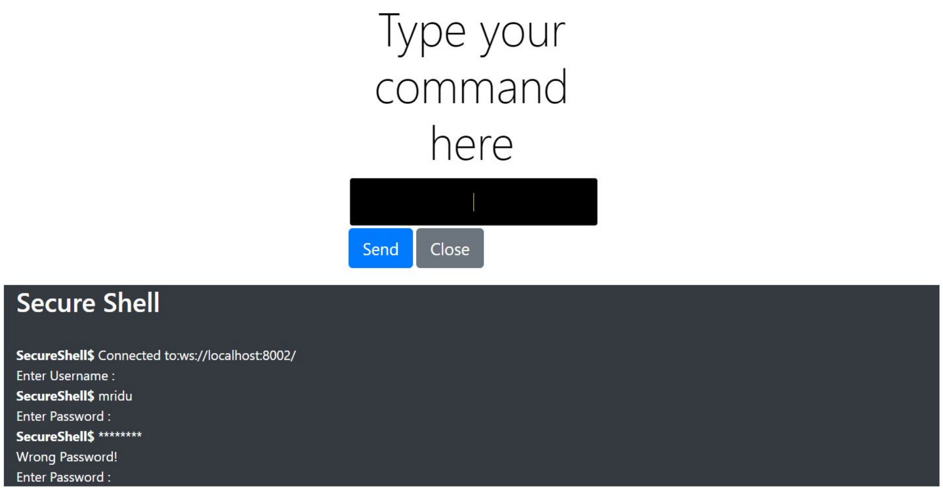
By Mridu Kejariwal & Kashish Madan

Server.py



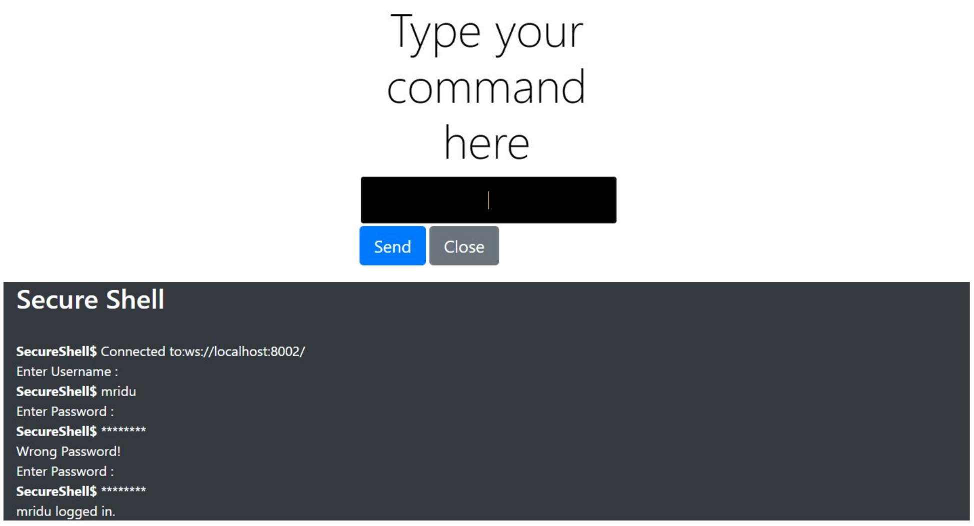
```
C:\Users\hp\Desktop>cd Secure-Shell-Using-WebSockets

C:\Users\hp\Desktop\Secure-Shell-Using-WebSockets>python server.py
YOUR SSH SERVER IS UP AND RUNNING

USER DATA: {'admin1': '58b5444cf1b6253a4317fe12daff411a78bda0a95279b1d5768ebf5ca60829e78da944e8a9160a0b6d428cb213e813525a7
2650dac67b88879394ff624da482f', 'mridu': 'd0e49677859d90351f974d619ce88a41370b8b92bfa6a2f6349c47f2bc29d561cb3bd1d4a48f15bd
4b33224e8f8cde5c591fd1da2cc7140875961df63a1e1eda', 'kashish': 'b745e72c25f6344ea506e5df12b7b13e0a9dbc6a424bfde4e19db7d47c3
0dc8972305b00ecc6278d6765e1afb3ec8a8786e7d7cf130823b6afdd19a8b3d931e5', 'admin2': '661bb43140229ad4dc3e762e7bdd68cc14bb909
3c158c386bd989fea807acd9bd7f805ca4736b870b6571594d0d8fcfc57b98431143dfb770e083fa9be89bc72'}
```

Wrong password verification



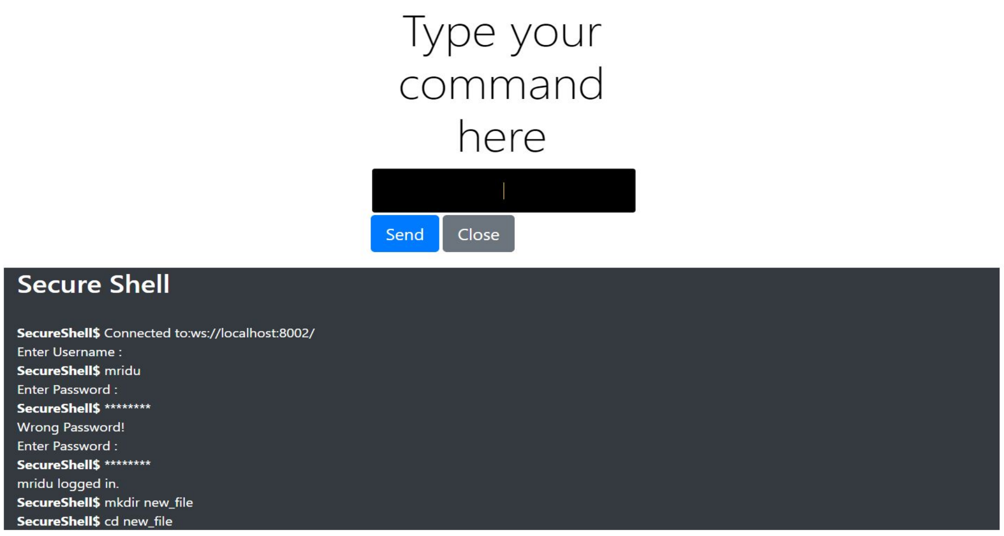
```
SecureShell$ Connected to:ws://localhost:8002/
Enter Username :
SecureShell$ mridu
Enter Password :
SecureShell$ ********
Wrong Password!
Enter Password :
```

Successful Authentication



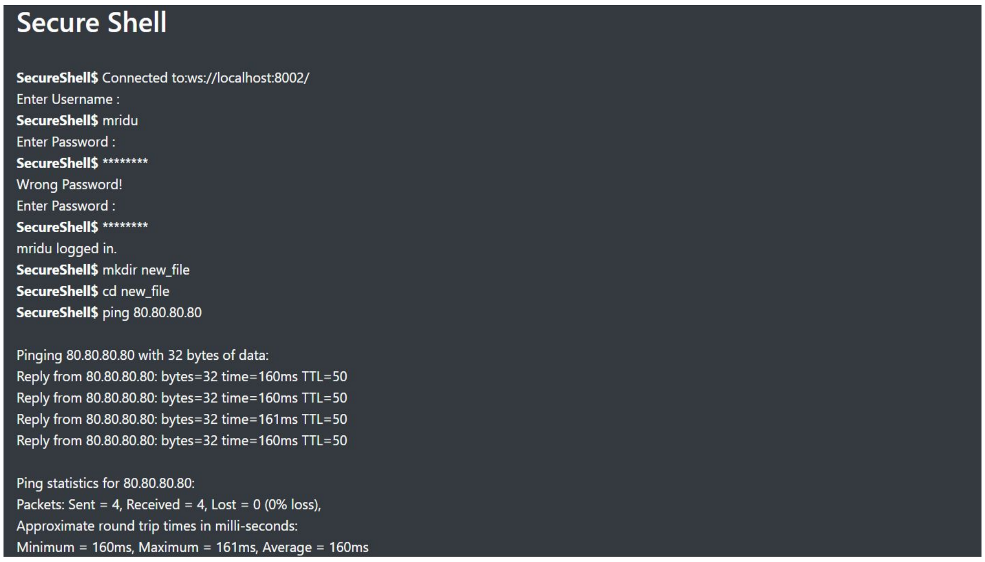
```
SecureShell$ Connected to:ws://localhost:8002/
Enter Username :
SecureShell$ mridu
Enter Password :
SecureShell$ ********
Wrong Password!
Enter Password :
SecureShell$ ********
mridu logged in.
```

mkdir new_file--->cd new_file



```
SecureShell$ Connected to:ws://localhost:8002/
Enter Username :
SecureShell$ mridu
Enter Password :
SecureShell$ ********
Wrong Password!
Enter Password :
SecureShell$ ********
mridu logged in.
SecureShell$ mkdir new_file
SecureShell$ cd new_file
```

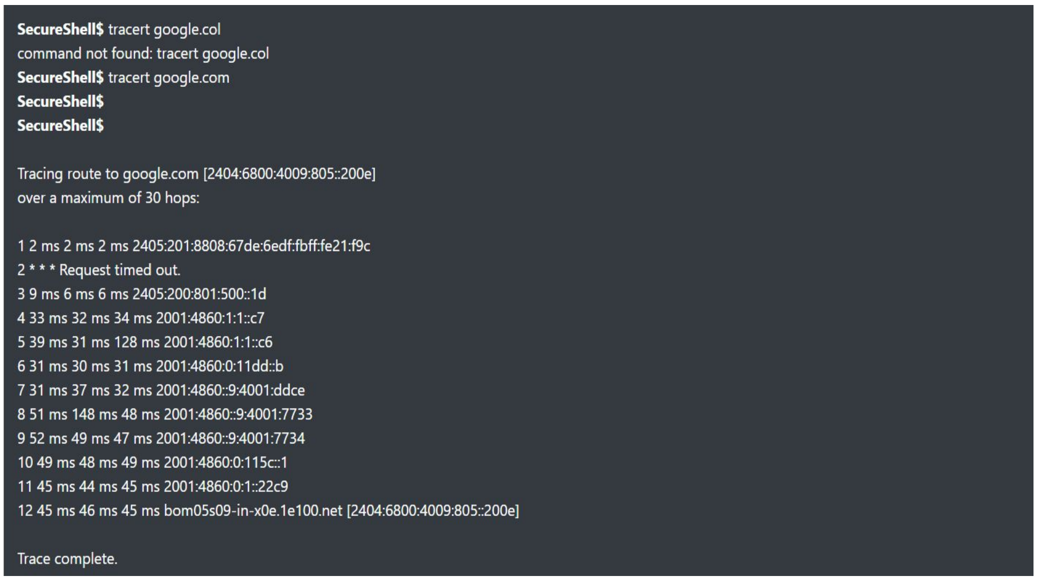
ping 80.80.80.80



Secure Shell

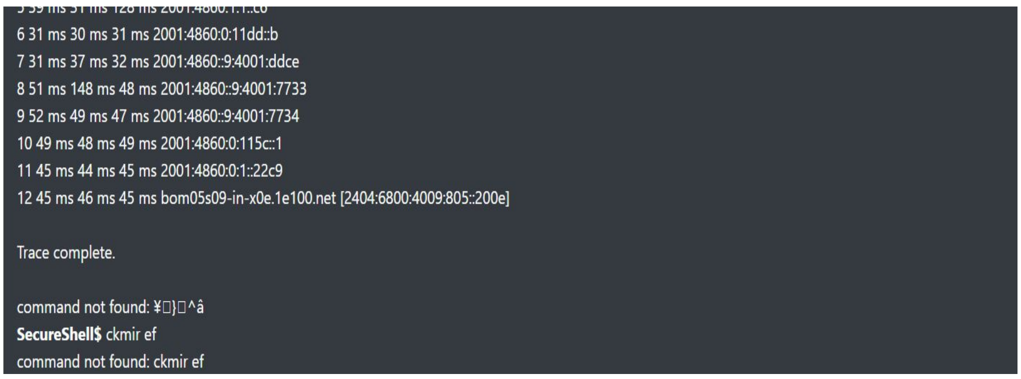
```
SecureShell$ Connected to:ws://localhost:8002/
Enter Username :
SecureShell$ mridu
Enter Password :
SecureShell$ ********
Wrong Password!
Enter Password :
SecureShell$ ********
mridu logged in.
SecureShell$ mkdir new_file
SecureShell$ cd new_file
SecureShell$ ping 80.80.80.80

Pinging 80.80.80.80 with 32 bytes of data:
Reply from 80.80.80.80: bytes=32 time=160ms TTL=50
Reply from 80.80.80.80: bytes=32 time=160ms TTL=50
Reply from 80.80.80.80: bytes=32 time=161ms TTL=50
Reply from 80.80.80.80: bytes=32 time=160ms TTL=50

Ping statistics for 80.80.80.80:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 160ms, Maximum = 161ms, Average = 160ms
```

tracert google.com



```
SecureShell$ tracert google.col
command not found: tracert google.col
SecureShell$ tracert google.com
SecureShell$
SecureShell$

Tracing route to google.com [2404:6800:4009:805::200e]
over a maximum of 30 hops:

1 2 ms 2 ms 2 ms 2405:201:8808:67de:6edf:fbff:fe21:f9c
2 * * * Request timed out.
3 9 ms 6 ms 6 ms 2405:200:801:500::1d
4 33 ms 32 ms 34 ms 2001:4860:1:1::c7
5 39 ms 31 ms 128 ms 2001:4860:1:1::c6
6 31 ms 30 ms 31 ms 2001:4860:0:11dd::b
7 31 ms 37 ms 32 ms 2001:4860::9:4001:ddce
8 51 ms 148 ms 48 ms 2001:4860::9:4001:7733
9 52 ms 49 ms 47 ms 2001:4860::9:4001:7734
10 49 ms 48 ms 49 ms 2001:4860:0:115c::1
11 45 ms 44 ms 45 ms 2001:4860:0:1::22c9
12 45 ms 46 ms 45 ms bom05s09-in-x0e.1e100.net [2404:6800:4009:805::200e]

Trace complete.
```

command not found



```
6 31 ms 30 ms 31 ms 2001:4860:0:11dd::b
7 31 ms 37 ms 32 ms 2001:4860::9:4001:ddce
8 51 ms 148 ms 48 ms 2001:4860::9:4001:7733
9 52 ms 49 ms 47 ms 2001:4860::9:4001:7734
10 49 ms 48 ms 49 ms 2001:4860:0:115c::1
11 45 ms 44 ms 45 ms 2001:4860:0:1::22c9
12 45 ms 46 ms 45 ms bom05s09-in-x0e.1e100.net [2404:6800:4009:805::200e]

Trace complete.

command not found: ¥▯}▯^â
SecureShell$ ckmir ef
command not found: ckmir ef
```

<u>Conclusion</u>

Secure Shell, or SSH, is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command line, login, and remote command execution, but any network service can be secured with SSH.

To conclude, SSH has following functions

- secure remote access to SSH-enabled network systems or devices for users, as well as automated processes;
- secure and interactive file transfer sessions;
- automated and secure file transfers;
- secure issuance of commands on remote devices or systems; and
- secure management of network infrastructure components.

This knowledge can be applied in any real-time application where communication with the server is critical.

SSH helped establish a cryptographically secured connection between the Client and the Server by authenticating each side.