EXP NO: 8	**Develop a program to create reverse shell using TCP sockets**

Aim

Demonstrate the networking and I/O concepts behind reverse shells while preventing misuse: a server sends *whitelisted* commands to an authenticated client; the client executes only allowed commands and returns output.

Procedure

1. Server listens for TCP connection and authenticates client with a shared secret.
2. Server shows a menu of allowed commands (keys) and sends chosen key as JSON to client.
3. Client receives request, verifies key against its own whitelist, runs the mapped command with subprocess.run(shell=False), and returns JSON with returncode, stdout, and stderr.
4. Server prints returned output. Repeat until exit.

Code —

server.py (controller)

```
#!/usr/bin/env python3
# server.py — send whitelisted commands to a connected client (lab use only)

import socket, json

HOST = '127.0.0.1'   # lab/localhost
PORT = 9001
SHARED_SECRET = "lab-secret-123"

WHITELIST = {"uptime": ["uptime"], "hostname": ["hostname"], "date": ["date",
"+%Y-%m-%d %H:%M:%S"]}

def handle_client(conn, addr):
    print(f'Client connected: {addr}')
    try:
        raw = conn.recv(1024)
        auth = json.loads(raw.decode())
        if auth.get("secret") != SHARED_SECRET:
            conn.sendall(json.dumps({"status":"error","msg":"auth failed"}).encode());
return
```

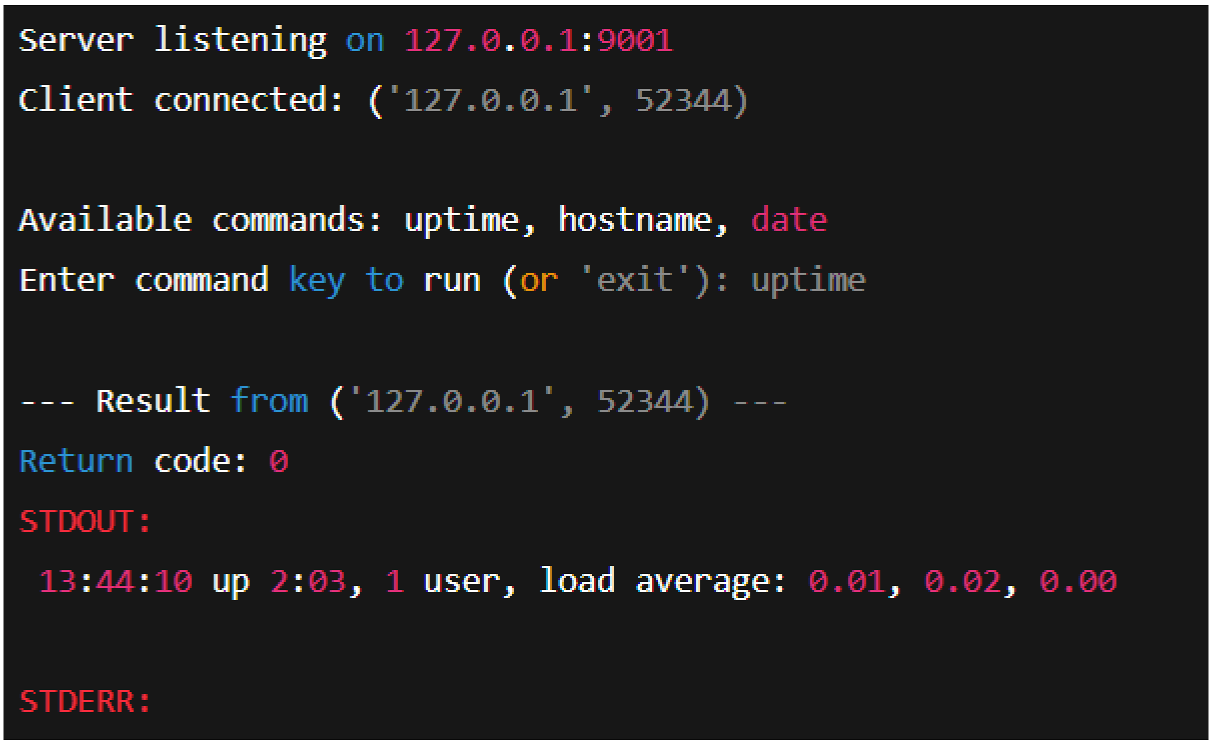
```
        conn.sendall(json.dumps({"status":"ok","msg":"auth success"}).encode())

        while True:
            print("\nAvailable commands:", ', '.join(WHITELIST.keys()))
            choice = input("Enter command key to run (or 'exit'): ").strip()
            if choice == "exit": break
            if choice not in WHITELIST:
                print("Invalid choice."); continue
            conn.sendall(json.dumps({"cmd_key": choice}).encode())
            resp_raw = conn.recv(65536)
            resp = json.loads(resp_raw.decode())
            print(f"\n--- Result from {addr} ---\nReturn code:
{resp.get('returncode')}\nSTDOUT:\n{resp.get('stdout')}\nSTDERR:\n{resp.get('stder
r')}\n")
    except Exception as e:
        print("Connection error:", e)
    finally:
        conn.close(); print("Connection closed.")

def main():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT)); s.listen(1)
        print(f"Server listening on {HOST}:{PORT}")
        conn, addr = s.accept(); handle_client(conn, addr)

if __name__ == "__main__": main()
```

Code — client.py (agent)

```
#!/usr/bin/env python3
# client.py — connect to server, authenticate, execute only whitelisted commands

import socket, json, subprocess

HOST = '127.0.0.1'
PORT = 9001
SHARED_SECRET = "lab-secret-123"

WHITELIST = {"uptime": ["uptime"], "hostname": ["hostname"], "date": ["date",
"+%Y-%m-%d %H:%M:%S"]}

def run_command(cmd_args):
    try:
        res = subprocess.run(cmd_args, capture_output=True, text=True, timeout=10)
        return {"returncode": res.returncode, "stdout": res.stdout, "stderr": res.stderr}
    except Exception as e:
        return {"returncode": -1, "stdout": "", "stderr": str(e)}

def main():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        s.sendall(json.dumps({"secret": SHARED_SECRET}).encode())
        auth_resp = json.loads(s.recv(1024).decode())
        if auth_resp.get("status") != "ok":
            print("Auth failed:", auth_resp); return
        print("Authenticated. Waiting for commands...")
        while True:
            data = s.recv(4096)
            if not data: break
            req = json.loads(data.decode()); key = req.get("cmd_key")
            if key not in WHITELIST:
                resp = {"returncode": -2, "stdout": "", "stderr": "command not allowed"}
            else:
                resp = run_command(WHITELIST[key])
            s.sendall(json.dumps(resp).encode())

if __name__ == "__main__": main()
```

Output:

Sample input / output (server terminal)



```
Server listening on 127.0.0.1:9001
Client connected: ('127.0.0.1', 52344)

Available commands: uptime, hostname, date
Enter command key to run (or 'exit'): uptime

--- Result from ('127.0.0.1', 52344) ---
Return code: 0
STDOUT:
 13:44:10 up 2:03, 1 user, load average: 0.01, 0.02, 0.00

STDERR:
```

Result:

Developed a program to create reverse shell using TCP sockets