1. Use `argv[1]` in argument as the port to which `spoofcheck.cpp` should bind itself.
2. Instantiate a TCP socket.
3. Go into an infinite loop (`while (true) ...`) where:
    1. Accept a new connection from a client through **`accept()`.**
    2. Spawn a child process through **`fork()`.** The parent closes this connection and goes back to the top of the loop, whereas the child continues checking the integrity of this connection.
    3. Retrieve the client's IP address and port of this connection through **`getpeername()`.**
    4. Retrieve the client's `hostent` data structure through **`gethostbyaddr()`.**
    5. Retrieve the client's official name, aliases, and registered IP addresses from the **`hostent`.**
    6. Decide whether this client is a honest or a spoofing client by matching its IP address retrieved from `getpeername()` and the list of addresses retrieved via `gethostbyaddr()`. (In other words, if you confirm that the client's IP address of this connection matches one of the addresses listed in `hostent`, you can trust this client.)
    7. Terminate this child process.

CSS 432 Networking

Program 4: Domain Name Service

Raghu Tirumala

Documentation of Spoofcheck.cpp:

The spoofcheck program checks the integrity of a client connection to a server by looking for IP address spoofing. The program takes one argument, the port for the server to bind to. Spoofcheck refers a client's IP address to a DNS server that retrieves a client's official hostname, aliases, and registered IP addresses.

The program behaves as follows:

1. Use argv[1] in argument as the port to which spoofcheck.cpp should bind itself.
2. Instantiate a TCP socket.
3. Go into an infinite loop (while (true) ...) where:
    1. Accept a new connection from a client through **accept()**.
    2. Spawn a child process through **fork()**. The parent closes this connection and goes back to the top of the loop, whereas the child continues checking the integrity of this connection.
    3. Retrieve the client's IP address and port of this connection through **getpeername()**.
    4. Retrieve the client's hostent data structure through **gethostbyaddr()**.

5. Retrieve the client's official name, aliases, and registered IP addresses from the **hostent**.
6. Decide whether this client is a honest or a spoofing client by matching its IP address retrieved from getpeername() and the list of addresses retrieved via gethostbyaddr().
7. Terminate this child process.

Execution Output:
This is the output running SpoofCheck while verifying the integrity of three different client's. One of these clients being a windows machine.

```
raghut2@uw1-320-04:~/Documents/CSSProjects/CSS432/Program4$ ./spoofcheck 51020
client addr = 172.21.198.85 port = 48852
official hostname: uw1-320-05.uwb.edu
alias: uw1-320-05
ip address: 172.21.198.85 ... hit!
an honest client

client addr = 172.21.198.83 port = 50048
official hostname: uw1-320-03.uwb.edu
alias: uw1-320-03
ip address: 172.21.198.83 ... hit!
an honest client

client addr = 10.156.7.123 port = 41592
official hostname: D-10-156-7-123.dhcp4.washington.edu
alias: none
ip address: 10.156.7.123 ... hit!
an honest client
```

Discussions

1. Server Initiated TCP Disconnection

The client in SpoofCheck merely initiates a connection with a server. After this the server handles the rest of the work. Since the client is (probalbly) not going to initiate the disconnection with the spoofcheck server, the server must terminate the connection. Since the server does not expect the client to disconnect there could be problems if the client somehow initiates the disconnection.

2. Client connecting through NAT

   Yes, the server can verify this client's integrity since devices operating with a NAT usually aren't able to be distinguished from other devices.

3. Client using dynamic address from DHCP server

   Since DHCP is usually tied to a DNS server, this guarantees there are proper DNS entries for clients using DHCP so that spoofcheck can properly function.