Port Scanner in C

Liam Powell

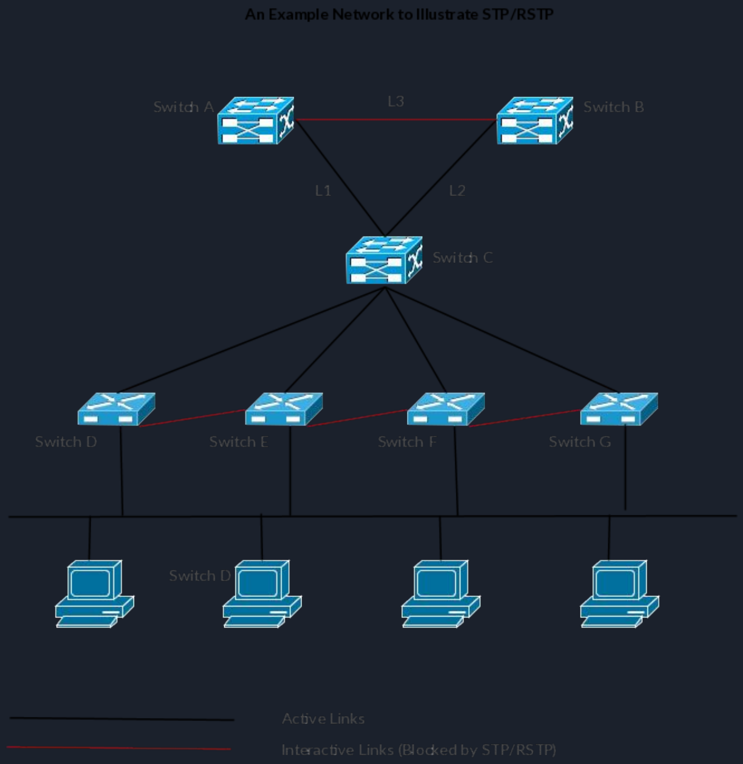
ICS 265-01 C Programming

Overview

- Scan Types
    - Full scan
    - Select Scan
    - Default Scan
- Examples
    - Output
    - Report info
- Function breakdown
    - Scan functions
    - Socket functions
- Challenges

Scan Type Overview

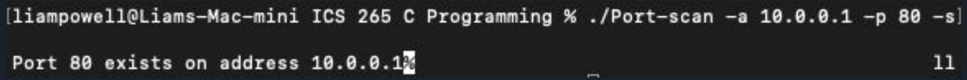
- Full Scan
  - Ports covered
  - Report Generation
- Select Scan
  - Simple address/port scan
- Default Scan
  - Scanning methods
  - Challenges



https://www.cleanpng.com/png-network-topology-computer-network-diagram-spanning-3878177/preview.html

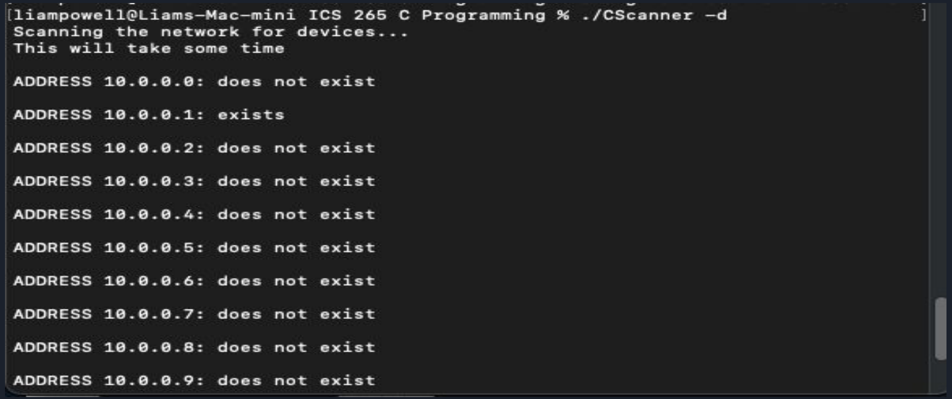
Example Output & Report

- Select Scan



```
[liampowell@Liams-Mac-mini ICS 265 C Programming % ./Port-scan -a 10.0.0.1 -p 80 -s]

Port 80 exists on address 10.0.0.1%                                                11
```

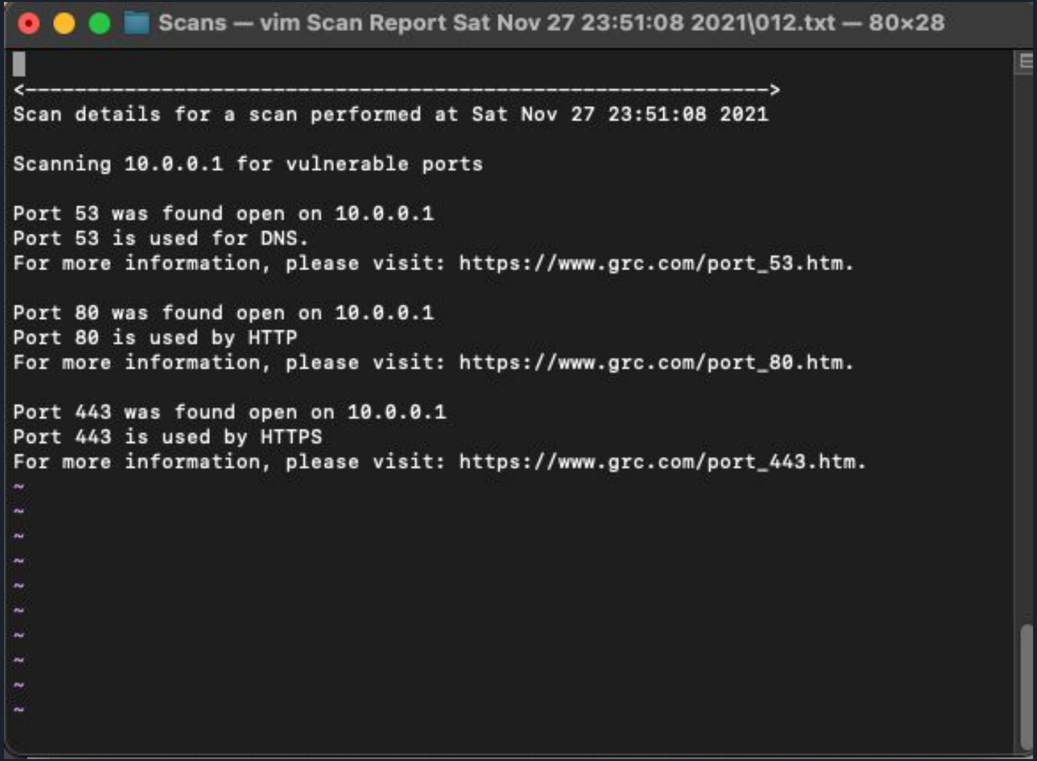
- Default Scan



```
[liampowell@Liams-Mac-mini ICS 265 C Programming % ./CScanner -d                   ]
Scanning the network for devices...
This will take some time

ADDRESS 10.0.0.0: does not exist

ADDRESS 10.0.0.1: exists

ADDRESS 10.0.0.2: does not exist

ADDRESS 10.0.0.3: does not exist

ADDRESS 10.0.0.4: does not exist

ADDRESS 10.0.0.5: does not exist

ADDRESS 10.0.0.6: does not exist

ADDRESS 10.0.0.7: does not exist

ADDRESS 10.0.0.8: does not exist

ADDRESS 10.0.0.9: does not exist
```

Example Output & Report cont.

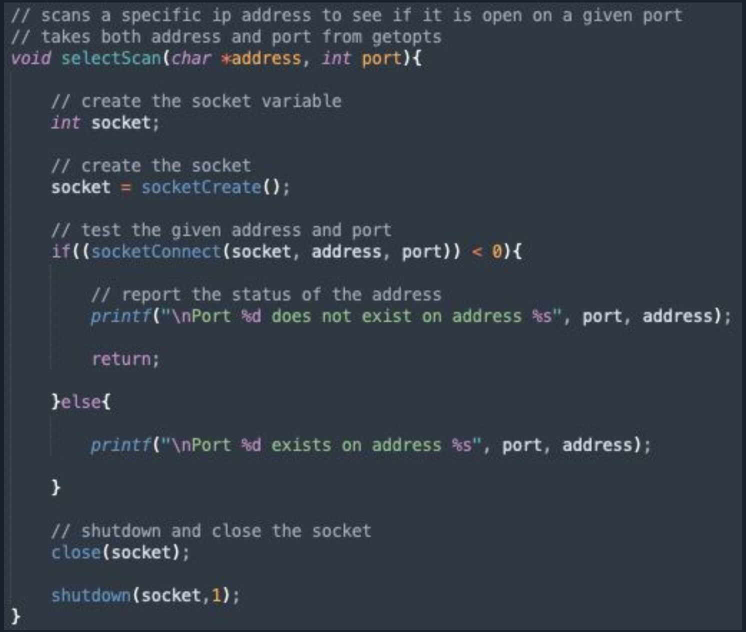
- Example Report


Scans — vim Scan Report Sat Nov 27 23:51:08 2021\012.txt — 80×28

```
<------------------------------------------------------------->
Scan details for a scan performed at Sat Nov 27 23:51:08 2021

Scanning 10.0.0.1 for vulnerable ports

Port 53 was found open on 10.0.0.1
Port 53 is used for DNS.
For more information, please visit: https://www.grc.com/port_53.htm.

Port 80 was found open on 10.0.0.1
Port 80 is used by HTTP
For more information, please visit: https://www.grc.com/port_80.htm.

Port 443 was found open on 10.0.0.1
Port 443 is used by HTTPS
For more information, please visit: https://www.grc.com/port_443.htm.
~
~
~
~
~
~
~
~
~
~
```

Function Breakdown
Scan Functions

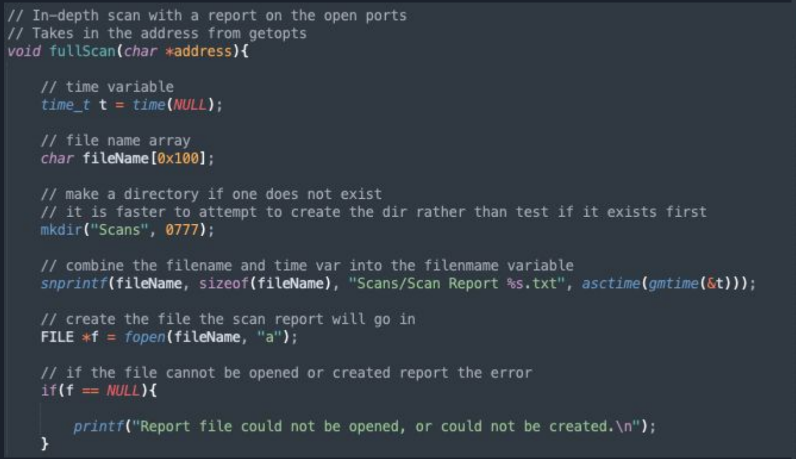
Select Scan function



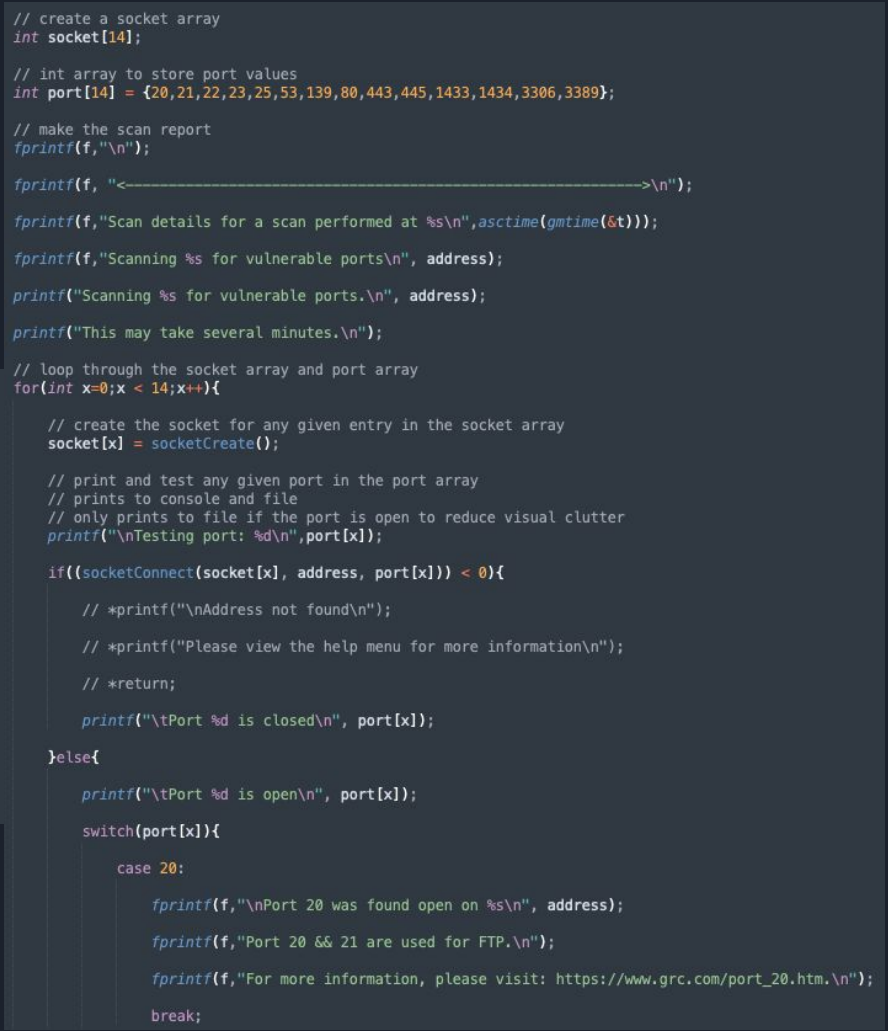
```c
// scans a specific ip address to see if it is open on a given port
// takes both address and port from getopts
void selectScan(char *address, int port){

    // create the socket variable
    int socket;

    // create the socket
    socket = socketCreate();

    // test the given address and port
    if((socketConnect(socket, address, port)) < 0){

        // report the status of the address
        printf("\nPort %d does not exist on address %s", port, address);

        return;

    }else{

        printf("\nPort %d exists on address %s", port, address);

    }

    // shutdown and close the socket
    close(socket);

    shutdown(socket,1);
}
```

Function Breakdown
Scan Functions

Full Scan function



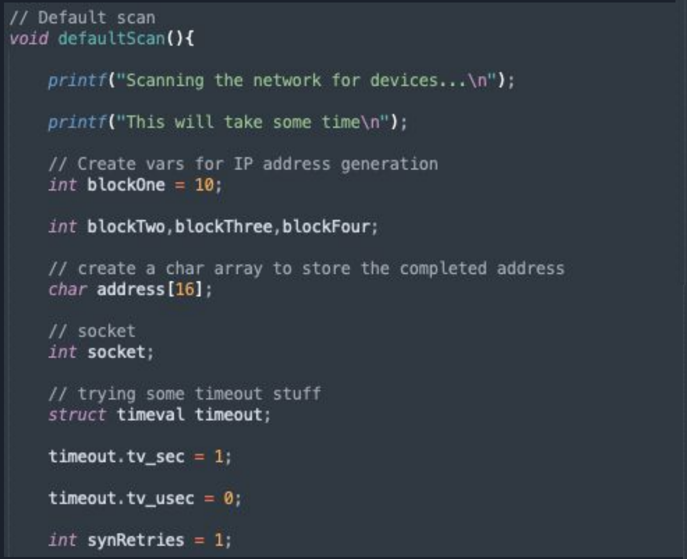
```c
// In-depth scan with a report on the open ports
// Takes in the address from getopts
void fullScan(char *address){

    // time variable
    time_t t = time(NULL);

    // file name array
    char fileName[0x100];

    // make a directory if one does not exist
    // it is faster to attempt to create the dir rather than test if it exists first
    mkdir("Scans", 0777);

    // combine the filename and time var into the filenmame variable
    snprintf(fileName, sizeof(fileName), "Scans/Scan Report %s.txt", asctime(gmtime(&t)));

    // create the file the scan report will go in
    FILE *f = fopen(fileName, "a");

    // if the file cannot be opened or created report the error
    if(f == NULL){

        printf("Report file could not be opened, or could not be created.\n");
    }
```



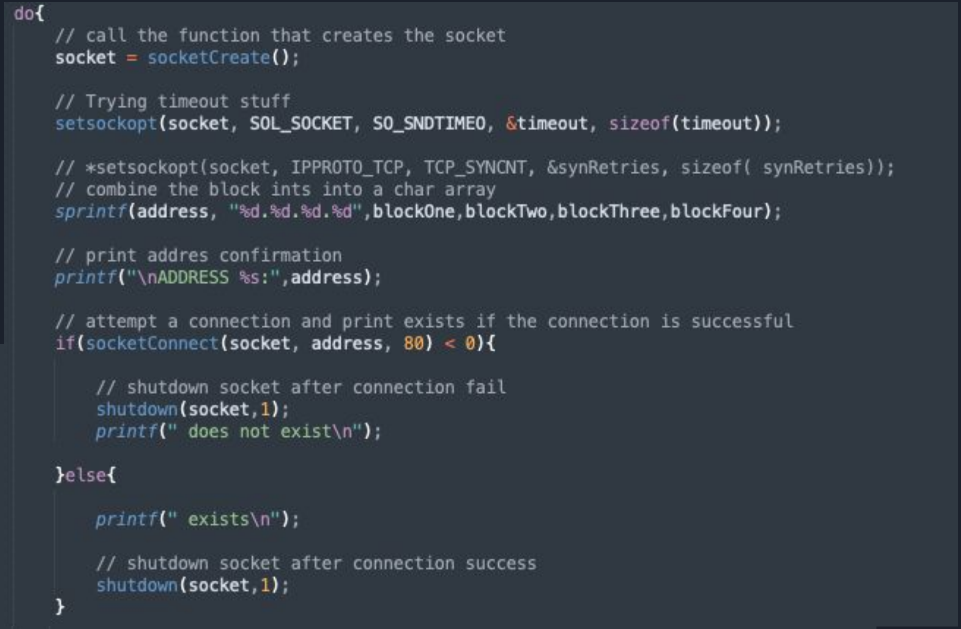
```c
// create a socket array
int socket[14];

// int array to store port values
int port[14] = {20,21,22,23,25,53,139,80,443,445,1433,1434,3306,3389};

// make the scan report
fprintf(f,"\n");

fprintf(f, "<------------------------------------------------------------------------->\n");

fprintf(f,"Scan details for a scan performed at %s\n",asctime(gmtime(&t)));

fprintf(f,"Scanning %s for vulnerable ports\n", address);

printf("Scanning %s for vulnerable ports.\n", address);

printf("This may take several minutes.\n");

// loop through the socket array and port array
for(int x=0;x < 14;x++){

    // create the socket for any given entry in the socket array
    socket[x] = socketCreate();

    // print and test any given port in the port array
    // prints to console and file
    // only prints to file if the port is open to reduce visual clutter
    printf("\nTesting port: %d\n",port[x]);

    if((socketConnect(socket[x], address, port[x])) < 0){

        // *printf("\nAddress not found\n");

        // *printf("Please view the help menu for more information\n");

        // *return;

        printf("\tPort %d is closed\n", port[x]);

    }else{

        printf("\tPort %d is open\n", port[x]);

        switch(port[x]){

            case 20:

                fprintf(f,"\nPort 20 was found open on %s\n", address);

                fprintf(f,"Port 20 && 21 are used for FTP.\n");

                fprintf(f,"For more information, please visit: https://www.grc.com/port_20.htm.\n");

                break;
```

Function Breakdown
Scan Functions

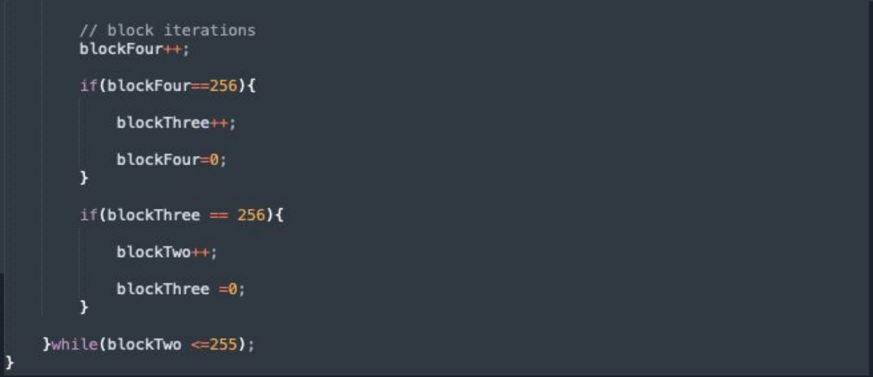
Default Scan function



```c
// Default scan
void defaultScan(){

    printf("Scanning the network for devices...\n");

    printf("This will take some time\n");

    // Create vars for IP address generation
    int blockOne = 10;

    int blockTwo,blockThree,blockFour;

    // create a char array to store the completed address
    char address[16];

    // socket
    int socket;

    // trying some timeout stuff
    struct timeval timeout;

    timeout.tv_sec = 1;

    timeout.tv_usec = 0;

    int synRetries = 1;
```



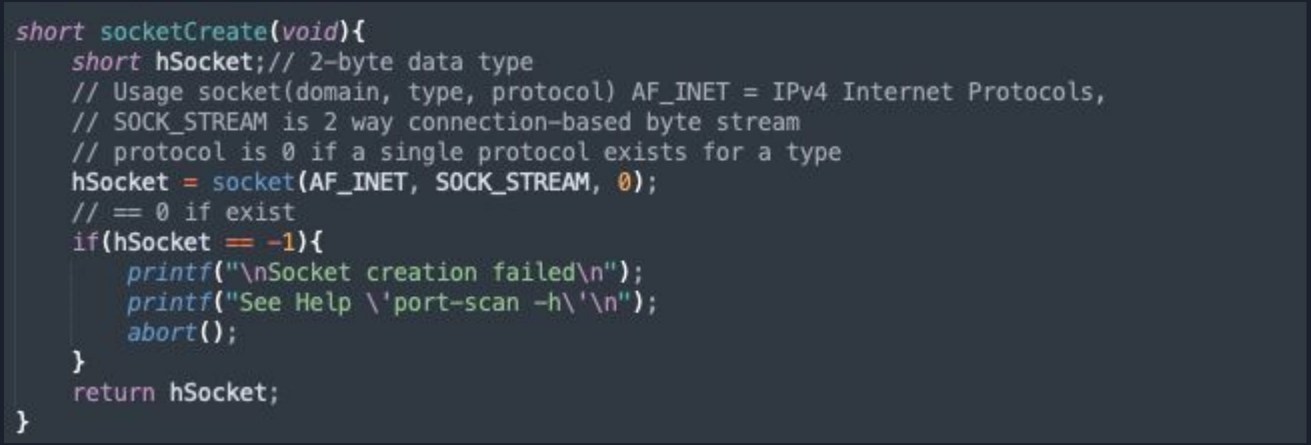
```c
do{
    // call the function that creates the socket
    socket = socketCreate();

    // Trying timeout stuff
    setsockopt(socket, SOL_SOCKET, SO_SNDTIMEO, &timeout, sizeof(timeout));

    // *setsockopt(socket, IPPROTO_TCP, TCP_SYNCNT, &synRetries, sizeof( synRetries));
    // combine the block ints into a char array
    sprintf(address, "%d.%d.%d.%d",blockOne,blockTwo,blockThree,blockFour);

    // print addres confirmation
    printf("\nADDRESS %s:",address);

    // attempt a connection and print exists if the connection is successful
    if(socketConnect(socket, address, 80) < 0){

        // shutdown socket after connection fail
        shutdown(socket,1);
        printf(" does not exist\n");

    }else{

        printf(" exists\n");

        // shutdown socket after connection success
        shutdown(socket,1);
    }
```



```c
        // block iterations
        blockFour++;

        if(blockFour==256){

            blockThree++;

            blockFour=0;
        }

        if(blockThree == 256){

            blockTwo++;

            blockThree =0;
        }

    }while(blockTwo <=255);
}
```

Function Breakdown
Socket Functions

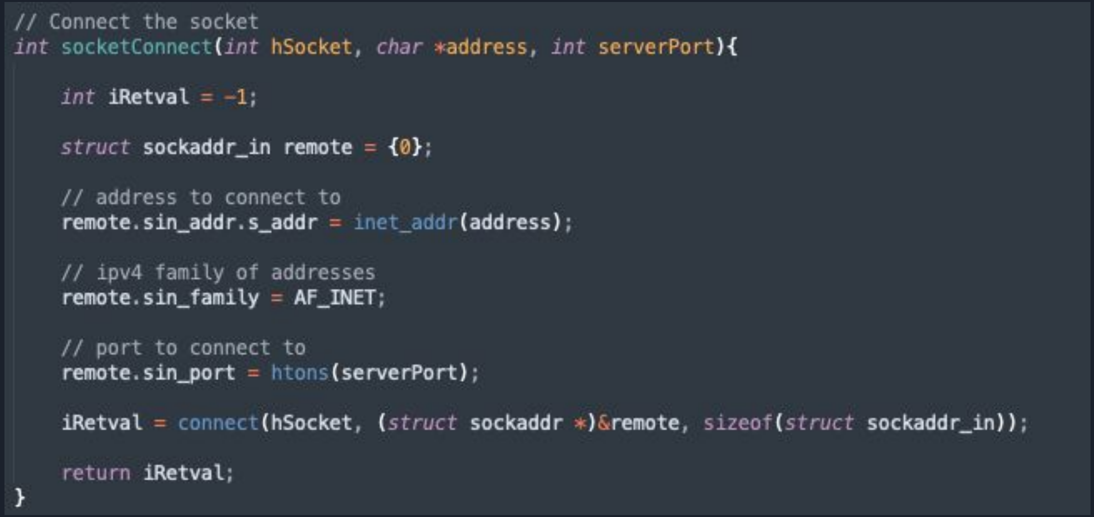
Socket Creation



```
short socketCreate(void){
    short hSocket;// 2-byte data type
    // Usage socket(domain, type, protocol) AF_INET = IPv4 Internet Protocols,
    // SOCK_STREAM is 2 way connection-based byte stream
    // protocol is 0 if a single protocol exists for a type
    hSocket = socket(AF_INET, SOCK_STREAM, 0);
    // == 0 if exist
    if(hSocket == -1){
        printf("\nSocket creation failed\n");
        printf("See Help \'port-scan -h\'\n");
        abort();
    }
    return hSocket;
}
```

Function Breakdown
Socket Functions

Socket Connection



```c
// Connect the socket
int socketConnect(int hSocket, char *address, int serverPort){

    int iRetval = -1;

    struct sockaddr_in remote = {0};

    // address to connect to
    remote.sin_addr.s_addr = inet_addr(address);

    // ipv4 family of addresses
    remote.sin_family = AF_INET;

    // port to connect to
    remote.sin_port = htons(serverPort);

    iRetval = connect(hSocket, (struct sockaddr *)&remote, sizeof(struct sockaddr_in));

    return iRetval;
}
```

Challenges

- Sockets
  - Learning about sockets
  - Connection speeds and timeouts
- Creating the default scan
  - Socket arrays
  - Ensuring proper socket handling
- Reports
  - Deciding limits for the full scan
- Potential improvements
  - External tool integration
  - User defined address ranges