MENDETEKSI SERANGAN DDOS (*DISTRIBUTED DENIAL OF SERVICE*) MENGGUNAKAN DDOS *DEFLATE*

Oleh

ZULKIFLI IBRAHIM

T3115111

SKRIPSI

Untuk Memenuhi Salah Satu Syarat Ujian

Guna Memperoleh Gelar Sarjana

PROGRAM SARJANA
TEKNIK INFORMATIKA
UNIVERSITAS ICHSAN GORONTALO
GORONTALO
2020

MENDETEKSI SERANGAN DDOS (*DISTRIBUTED DENIAL OF SERVICE*) MENGGUNAKAN DDOS *DEFLATE*

Oleh

ZULKIFLI IBRAHIM

T3115111

SKRIPSI

Untuk Memenuhi Salah Satu Syarat Ujian

Guna Memperoleh Gelar Sarjana

PROGRAM SARJANA
TEKNIK INFORMATIKA
UNIVERSITAS ICHSAN GORONTALO
GORONTALO
2020

PENGESAHAN SKRIPSI

MENDETEKSI SERANGAN DDOS (*DISTRIBUTED DENIAL OF SERVICE*) MENGGUNAKAN DDOS *DEFLATE*

Oleh

ZULKIFLI IBRAHIM

T3115111

SKRIPSI

Untuk memenuhi salah satu syarat ujian
guna memperoleh gelar Sarjana
Program Studi Teknik Informatika,
ini telah disetujui oleh Tim Pembimbing

Gorontalo, 04 Desember 2020

Pembimbing I	Pembimbing II
Yasin Aril Mustofa, M.Kom	**Serwin M.Kom**
NIDN. 0926088503	NIDN. 0918078802

PERSETUJUAN SKRIPSI

MENDETEKSI SERANGAN DDOS (*DISTRIBUTED DENIAL OF SERVICE*) MENGGUNAKAN DDOS *DEFLATE*

Oleh

ZULKIFLI IBRAHIM

T3115111

Diperiksa oleh Panitia Ujian Strata Satu (S1)
Universitas Ichsan Gorontalo

Gorontalo, 09 Desember 2020

1. Pembimbinng I
   Yasin Aril Mustofa, M.Kom .....
2. Pembimbing II
   Serwin, M.Kom .....
3. Penguji I
   Budy Santoso, S.Kom, M.Eng .....
4. Penguji II
   Warid Yunus, M.Kom .....
5. Penguji III
   Rofiq Harun, M.Kom .....

Mengetahui

Dekan Fakultas Ilmu Komputer	**Ketua Program Studi**
Zohrahayaty, M.Kom	**Irvan Abraham Salihi, M.Kom**
NIDN. 0912117702	**NIDN, 0928028101**

PERNYATAAN SKRIPSI

Dengan ini saya menyatakan bahwa:

1. Karya tulis (Skripsi) saya ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik (Sarjana) baik di Universitas Ichsan Gorontalo maupun di perguruan tinggi lainnya.
2. Karya tulis (Skripsi) saya ini adalah murni gagasan, rumusan, dan penelitian saya sendiri, tanpa bantuan pihak lain, kecuali arahan dari Tim Pembimbing.
3. Dalam karya tulis (Skripsi) saya ini tidak terdapat karya atau pendapat yang telah dipublikasikan orang lain, kecuali secara tertulis dicantumkan sebagai acuan/sitasi dalam naskah dan dicantumkan pula dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila dikemudian hari terdapat penyimpangan dan ketidak benaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya tulis ini, serta sanksi lainnya sesuai dengan normanorma yang berlaku di Universitas Ichsan Gorontalo.

Gorontalo, 04 Desember 2020
Yang Membuat Pernyataan,

Zulkifli Ibrahim

ABSTRACT

Disributed Denial of Service or what we know as DDoS is a network problem that is currently growing rapidly dynamically. This DDoS attack can result in the server's inability to serve legitimate service requests. Therefore DDoS attacks are very detrimental and need to be given effective prevention. In this study, the authors will try how to detect DDoS attacks by giving messages in the form of sound in order to make it easier for managers to take what to do next to secure DDoS attacks. By combining the sound alert system design with DDoS Deflate so that in the event of blocking the ip sound alert functions to provide a warning message in the form of a beep sound on the server. From the experimental results, it is proven that the sound alert is able to provide a warning message in the form of a sound when an IP is blocked on the black list on DDoS Deflate.

Keyword : Server, Flooding, DDOS, Deflate, Ubuntu, WEB, Sound, Alert

ABSTRAK

Disributed Denial of Service atau yang kita kenal dengan DDoS merupakan permasalahan jaringan yang sampai saat ini terus berkembang pesat secara dinamis. Serangan DDoS ini dapat mengakibatkan ketidak mampuan server untuk melayani *service request* yang sah. Karena itu serangan DDoS sangat merugikan dan perlu diberikan pencegahanyang efektif. Pada penelitian ini penulis akan mencoba bagaimana agar serangan DDoS dapat diketahui dengan dan dapat memberikan pesan berupa bunyi agar memudahkan pengelolah untuk melakukan tidakan apa yang harus dilakukan selajutnya untuk mengamankan serangan DDoS. Dengan melakukan penggabungan rancangan sistem *sound alert* dengan DDoS *Deflate* agar pada saat terjadi pemblokiran ip *sound alert* berfungsi memberikan pesan peringatan berupa suara *beep* pada server. Dari hasil percobaan terbukti *sound alert* mampu memberikan pesan peringatan berupa suara pada saat adanya ip yang terblokir pada *black list* di DDoS *Deflate*.

Kata Kunci : Server, Flooding, DDOS, Deflate, Ubuntu, WEB, Sound, Alert

KATA PENGANTAR

Alhamdulillah, penulis dapat menyelesaikan usulan penelitian ini dengan judul: **"Mendeteksi Serangan DDOS (*Distributed Denial Of Service*) Menggunakan DDOS *Deflate*"**, untuk memenuhi salah satu syarat penyusunan Skripsi Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Ichsan Gorontalo.

Penulis menyadari sepenuhnya bahwa usulan penelitian ini tidak mungkin terwujud tanpa bantuan dan dorongan dari berbagai pihak, baik bantuan moril maupun materil. Untuk itu, dengan segala keikhlasan dan kerendahan hati, penulis mengucapkan banyak terima kasih dan penghargaan setinggi-tingginya kepada:

1. Muhammad Ichsan Gaffar, SE, M.Ak, selaku Ketua Yayasan Pengembangan Ilmu Pengetahun dan teknologi (YPIPT) Ichsan Gorontalo;
2. Bapak Dr. Abdul Gaffar La Tjokke, M.Si, selaku Rektor Universitas Ichsan Gorontalo;
3. Ibu Zohrahayaty, M.Kom, selaku Dekan Fakultas Ilmu Komputer Universitas Ichsan Gorontalo;
4. Bapak Sudirman S. Panna, M.Kom, selaku Wakil Dekan I Bidang Akademi Fakultas Ilmu Komputer Universitas Ichsan Gorontalo;
5. Ibu Irma Surya Kumala Idris, M.Kom, selaku Wakil Dekan II Bidang Administrasi Umum Fakultas Ilmu Komputer Universitas Ichsan Gorontalo;
6. Bapak Sudirman Malangi, M.Kom, selaku Wakil Dekan III Bidang Kemahasiswaan Fakultas Ilmu Komputer Universitas Ichsan Gorontalo;
7. Bapak Irvan Abraham Salihi, M.Kom, selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Ichsan Gorontalo;
8. Bapak Yasin Aril Mustofa, M.Kom, selaku Pembimbing I;
9. Bapak Serwin, M.Kom, selaku Pembimbing II;
10. Bapak dan Ibu Dosen Universitas Ichsan Gorontalo yang telah mendidik dan mengajarkan berbagai disiplin ilmu kepada penulis;
11. Kedua Orang Tua saya yang tercinta, atas segala kasih sayang, jerih payah dan doa restunya dalam membesarkan dan mendidik penulis;

12. Rekan-rekan seperjuangan yang telah banyak memberikan bantuan dan dukungan moril yang sangat besar kepada penulis;
13. Kepada semua pihak yang ikut membantu dalam penyelesaian proposal ini yang tak sempat penulis sebutkan satu-persatu.

Semoga Allah, SWT melimpahkan balasan atas jasa-jasa mereka kepada kami. Penulis menyadari sepenuhnya bahwa apa yang telah dicapai ini masih jauh dari kesempurnaan dan masih terdapat banyak kekurangan. Oleh karena itu, penulis sangat mengharapkan adanya kritik dan saran yang konstruktif. Akhirnya penulis berharap semoga hasil yang telah dicapai ini dapat bermanfaat bagi kita semua, Aaamiiin.

Gorontalo, 04 Desember 2020

Penulis

DAFTAR ISI

KATA PENGANTAR v
DAFTAR ISI vii
DAFTAR GAMBAR ix
DAFTAR TABLE ix
DAFTAR LAMPIRAN x
BAB I PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Identifikasi Masalah 4
1.3 Rumusan masalah 4
1.4 Tujuan penelitian 4
1.5 Manfaat penelitian 4
1.5.1 Manfaat Teoritis 4
1.5.2 Manfaat Praktis 4
BAB II LANDASAN TEORI 5
2.1 Tinjauan Studi 5
2.2 Tinjauan Pustaka 7
2.2.1 DoS (*Denial Of Service*) 7
2.2.2 DDOS (*Distributed Denial Of Service*) 8
2.2.3 DDOS Deflate 18
2.2.4 Sound Alert 20
2.3 Kerangka Berpikir 21
BAB III METODE PENELITIAN 22
3.1 Jenis, Metode, Subjek, Objek, Waktu, Dan Lokasi Penelitian 22

3.2 Pngumpulan Data 23
3.2.1 Data Primer 23
3.2.2 Data Sekunder 23
3.3 Analisis Sistem 23
3.4 Desain Sistem 23
3.5 Konstuksi Sistem 24
BAB IV HASIL PENELITIAN 25
4.1 Hasil Pengumpulan Data 25
4.2 Analisa Dan Perancangan Sistem 27
4.2.1 Analisa Sistem 27
4.2.2 Perancangan Sound Alert 28
4.2.3 Pengujian Serangan DDoS TCP Flooding 31
4.2.4 Pengujian Serangan DDoS HTTP Flooding 34
BAB V PEMBAHASAN 38
5.1 Pembahasan Model 38
5.2 Hasil Pengujian Model 39
5.2.1 DDOS Deflate Tanpa Menggunakan Sound Alert 39
5.2.2 DDOS Deflate Dengan Menggunakan Sound Alert 39
BAB IV PENUTUP 40
6.1 Kesimpulan 40
6.2 Saran 40
DAFTAR PUSTAKA 41
LAMPIRAN SCRIPT 43

DAFTAR GAMBAR

Gambar 2. 1 Contoh Alur Serangan DoS 7
Gambar 2. 2 Contoh Alur Serangan DDoS 9
Gambar 2. 3 Contoh Serangan POD 14
Gambar 2. 4 Contoh Grafik Serangan POD 14
Gambar 2. 5 Contoh Serangan Syn Flooding 15
Gambar 2. 6 Contoh Hasil Analisa Syn Flooding Dengan Wireshark 16
Gambar 2. 7 Contoh Grafik Analisa Serangan Menggunakan Wireshark 16
Gambar 2. 8 Contoh Metasploit 17
Gambar 2. 9 Contoh tampilan DDoS Deflate 18
Gambar 2. 10 Cara Kerja DDoS Deflate 19
Gambar 2. 11 Contoh Gelombang Suara 20
Gambar 3. 1 Desain Penelitian 22
Gambar 4. 1 Trafik Data DOS TCP 25
Gambar 4. 2 Trafic Data DOS HTTP 26
Gambar 4. 3 Kinerja CPU Pada Saat Terjadi Serangan DDoS 26
Gambar 4. 4 Diagram Alur Kerja sound Alert 28
Gambar 4. 5 Rancangan Program Sound Alert 28
Gambar 4. 6 Notifikasi Setelah Banning IP 29
Gambar 4. 7 Penggunaan Penjadwalan Crontab 30
Gambar 4. 8 Konfigurasi Cron Sound Alert 30
Gambar 4. 9 Konfigurasi Cron Sound Alert 31
Gambar 4. 10 Ujicoba TCP Flooding dengan LOIC 32
Gambar 4. 11 Hasil TCP Flooding Menggunakan LOIC 33
Gambar 4. 12 Black List IP TCP Flooding 33
Gambar 4. 13 LOIC HTTP Flooding 34
Gambar 4. 14 Hasil HTTP Flooding Menggunakan LOIC 35
Gambar 4. 15 Black List IP HTTP Flooding 35
Gambar 4. 16 Hasil UDP Flooding Menggunakan LOIC 36
Gambar 5. 1 Kinerja CPU Tanpa DDoS Deflate 38

DAFTAR TABLE

Tabel 4. 1 *Black List* DDoS *Deflate Without Sound Alert* 27
Tabel 4. 2 Program Perancanzgan *Sound Alert*..... 29
Tabel 4. 3 Ujicoba TCP Flooding 31
Tabel 4. 4 HTTP Flooding 34
Tabel 4. 5 Pengujian Serangan UDP Flooding 36
Tabel 5. 1 Hasil DDoS *Deflate* Tanpa *Sound Alert* 39
Tabel 5. 2 DDoS *Deflate* Yang Menggunakan *Sound Alert* 39

DAFTAR LAMPIRAN

Lampiran : 1 Lampiran Script Program 43
Lampiran : 2 Riwayat Hidup Peneliti..... 82

BAB I
PENDAHULUAN

1.1 Latar belakang

Kehadiran internet beberapa dekade yang lalu benar-benar telah merevolusi cara kerja dunia ini, informasi mengalir begitu cepatnya bahkan secara *real time*. Seiring dengan perkembangan internet kejahatan dalam dunia internet pun mulai bermunculan, menurut data yang dilansir dari laman *Nerwork Security Infrastucture Report* (NETSCOUT) salah satu serangan yang paling populer pada akhir tahun 2014 yakni serangan DDoS[1]. *Distributed Denial of Service* atau disingkat menjadi DDoS merupakan merupakan jenis serangan DoS (*Denial of Service)* yang di distribusikan menggunakan banyak *host* atau memanfaatkan *Trojan* untuk memaksa komputer yang terinfeksi untuk melakukan serangan DoS (*Computer Zombie*)[2].

Dalam hal ini, serangan DDoS tidak boleh di anggap remeh, teknik serangan ini kerap di gunakan *hacker* untuk melumpuhkan suatu komputer atau server. Seperti yang terjadi pada salah satu layanan *Sony Entertainment* mengalami kerusakan. Server *PlayStation Network* (PSN) di serang DDoS oleh kelompok *hacker* bernama *Lizard Squad* [3].

Table 1.1 DDOS Attack In South Asia 2015-2018

NO	NEGARA	PRESENTASE
1.	Indonesia	76%
2.	China	74%
3.	Thailand	71%
4.	South Korea	63%
5.	Vietnam	62%

Berdasarkan tabel di atas, Indonesia menduduki peringkat nomor 1 dari tahun 2015-2018 sebagai Negara yang paling banyak melakukan serangan DDoS, baik itu serangan untuk luar Indonesia maupun yang berada di dalam[4]. Hal ini sempat di alami KASKUS salah satu forum terbesar di Indonesia yang di serang anak komunitas *YogyaFree* pada tahun 16-17 mei 2008. Serangan ini mengakibatkan *thread* yang telah dibuat terpaksa dikunci (*Locked*) oleh *administrator* forum tersebut. Karena berlangsung cukup lama akhirnya *administrator* mematikan server. Dan membuat KASKUS mengganti server yang bisa meminimalisir serangan serupa[5]. DDoS sendiri merupakan teknik serangan dengan membanjiri server menggunakan paket data berkapasitas besar, serangan ini dilakukan secara terus menerus hingga sistem tidak dapat menampung data dan akhirnya rusak. Dibutuhkan keahlian khusus untuk mengetahui adanya serangan yang berbahaya ini.

Pendeteksian serangan *Distribute Denial of Services* (DDoS) bisa dengan melakukan pelacakan berapa banyak user yang mengunjungi *website* setiap hari, setiap jam, bahkan setiap menit. Dengan begitu, dapat mengerti berapa rata-rata tingkat *trafic* yang belebihi ambang batas yang di wajarkan serta memeriksa koneksi jaringan yang terhubung pada server.

Pada dasarnya sebagai manusia tidak mampu bekerja selama 24 jam penuh untuk mengamati kinerja server demi mengetahui adanya serangan DDOS. Maka dari itu untuk mengantisipasi hal tersebut perlu adanya sistem yang dapat mendeteksi serangan DDoS dengan memberikan pemberitahuan berupa notifikasi berupa *Sound Alert* sebagai penanda bahwa server sedang di serang. Tentunya dengan memanfaatkan DDoS *Deflate* untuk melakukan pendeteksian serangan DDoS serta menambahkan fungsi *Sound Alert* pada program sehingga dapat merespon adanya serangan DDoS pada server web.

DDoS *Deflate* merupakan *script bash shell* ringan dan mudah di gunakan yang berjalan pada sistem operasi berbasis Linux yang dirancang untuk membantu dalam proses pemblokiran serangan atau penolakan layannan menggunakan alamat IP dengan jumlah koneksi yang melebihi konfigurasi, Serta program ini bersifat *Open Sources*. Penggunaan DDoS *Deflate* pada server web mampu

mengenali serangan DDoS dan melakukan tindakan pemblokiran terhadap serangan yang di curigai. Pemblokiran terjadi karena adanya interaksi yang melebihi batas yang di inginkan, sehingga DDoS *Deflate* melakukan pemblokiran terhadap alamat ip tersebut. Namun DDoS *Deflate* tidak hanya melakukan pemblokiran terhadap suatu koneksi yang di curigai akan tetapi melakukan pemblokiran pada suatu koneksi biasa yang melebihi konfigurasi. Artinya DDoS *Deflate* tidak bisa membedakan antara penyerang atau *user* biasa yang melakukan interaksi secara berlebihan. Sehingga dilakukan pengecekan terhadap alamat IP yang terblokir untuk di lakukan konfigurasi, apa termasuk dalam *white list* atau *black list*. Maka dari itu penggunaan *Sound Alert* sangat di perlukan untuk mengirimkan pemberitahuan berupa bunyi bahwa ada pemblokiran otomatis terhadap koneksi yang di curigai oleh DDoS *Deflate*.

Dengan adanya bunyi peringatan (*Sound Alert*) itu sendiri diharapkan sangat membantu kepada pengelola server, *Sound Alert* bertujuan untuk meningkatkan kewaspadaan terhadap serangan DDoS, serta memudahkan pengelola untuk mengetahui bahwa adanya serangan DDoS dan selanjutnya melakukan tindakan seperti apa untuk mengamankan suatu server web. Dan juga dengan adanya *Sound Alert* bisa meminimalisir terjadinya kerusakan pada sistem, dengan terdeteksinya adanya serangan DDoS pihak pengelola bisa melakukan tindakan pengamanan data-data penting sebelum terjadinya kerusakan. Dengan demikian dengan memanfaatkan DDoS *Deflate* yang bisa mendeteksi serangan DDoS dan menggunakan fungsi *Sound Alert* diharapkan dapat meminimalisir serangan DDoS yang sangat berbahaya bagi server web.

Pada beberapa penelitian sebelumnya tentang mendeteksi serangan DDoS. Hasil pengujian hanya menampilkan skema analisa untuk mengetahui adanya serangan. Berdasarkan permasalahan diatas dapat dilakukan pengembangan penelitian agar DDoS *Deflate* mampu memberikan notifikasi berupa bunyi pada saat system melakukan pemblokiran ip. Dengan mengangkat judul **"Mendeteksi Serangan DDOS (*Distributed Denial of Service*) Menggunakan DDOS *Deflate*"** diharapkan penelitian ini dapat memberikan keuntungan dan meminimalisir serangan DDoS.

1.2 Identifikasi Masalah

Berdasarkan latar belakang dapat diidentifisi masalah:

1) Serangan DDoS sangat berbahaya bagi kinerja komputer dan bersifat merugikan.
2) Sulitnya mengetahui serangan DDoS.

1.3 Rumusan masalah

1) Bagaimana mencegah serangan DDoS dengan memanfaatkan DDoS *Deflate*.
2) Bagaimana mengetahui serangan DDoS pada server web.

1.4 Tujuan penelitian

1) Menguji coba serangan DDoS dan melakukan pencegahan dengan memanfaatkan DDoS *Deflate*.
2) Mampu memberikan pesan peringatan berupa bunyi kepada pengelolah bahwa adanya serangan DDoS pada server web.

1.5 Manfaat penelitian

1.5.1 Manfaat Teoritis

Memberikan ilmu pengetahuan bagi pengembang khususnya dalam bidang keamanan jaringan.

1.5.2 Manfaat Praktis

Sumbangan pemikiran, karya, untuk meminimalisir terjadinya serangan DDoS.

BAB II
LANDASAN TEORI

2.1 Tinjauan Studi

Table 2.1 Tinjauan Studi

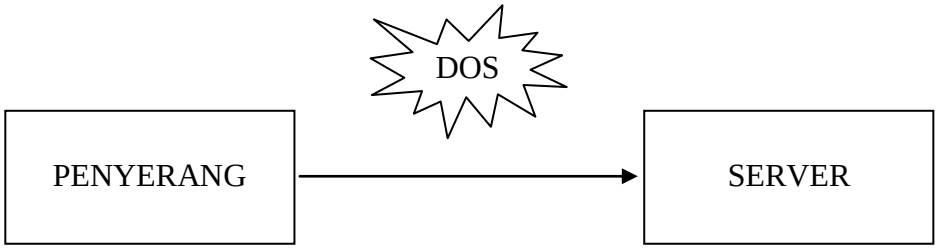
NO	PENELITI	JUDUL	TAHUN	METODE	HASIL
1.	Chirag Sheth Rajesh A Thakker[6]	Performance Evaluation and Comparison of Network Firewalls under DDoS Attack	2013		The maximum limits set are optimal for author's laboratory setup and has proved to be effective in mitigating laboratory generated DDoS in setup used.
2.	Jelena Mirkovic, Max Robinson, Peter Reiher, George Oikonomou [7]	Distributed Defense Against DDoS Attacks	2005	PKI	Each participant directly benefits from DefCOM operation. Victim networks receive protection from the attack, while source networks receive a guarantee that their traffic will be delivered during attacks

NO	PENELITI	JUDUL	TAHUN	METODE	HASIL
3.	Hijriyanto, Budi[8]	Perbandingan Penerapan Metode Pengamanan Web Server Menggunakan Mod Evasive Dan Ddos Deflate Terhadap Serangan Slow Post	2019		metode pengamanan yang baik dalam menangani serangan Slow Post. Baik disini dalam artian dimana metode tersebut dapat mengurangi atau menangkal serangan Slow Post yang di tujukan kepada web server.

2.2 Tinjauan Pustaka

2.2.1 DoS (*Denial Of Service*)

DoS merupakan jenis serangan terhadap sebuah komputer atau server didalam jaringan internet dengan cara menghabiskan sumber (*resurces*) yang dimiliki oleh komputer tersebut sampai komputer tersebut tidak dapat menjalankan fungsinya dengan benar sehingga secara tidak langsung mencegah pengguna lain untuk memperoleh akses layanan dari komputer yang diserang.



Gambar 2. 1 Contoh Alur Serangan DoS

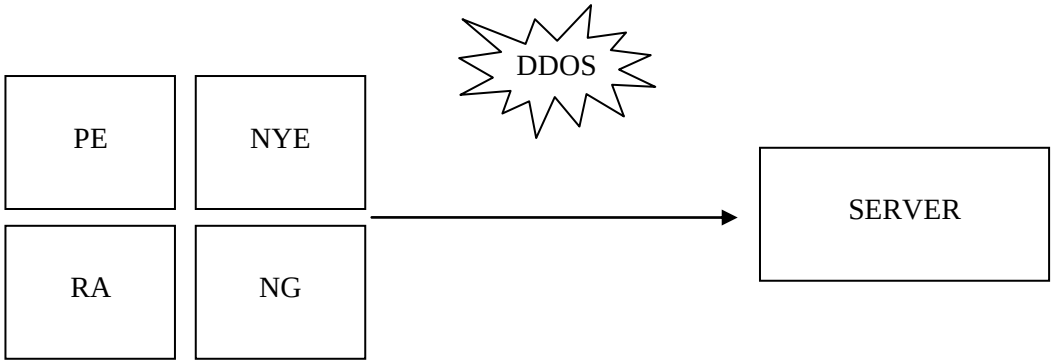
Serangan DoS pertama kali terjadi pada tahun 1974 oleh David Dennis – seorang siswa berusia 13 tahun di universitas *High School,* yang terletak di seberang jalan dari Laboratorium Penelitian Pendidikan Berbasis Komputer, *Computer Based Education Research* (CERL) di Universitas *Illinois Urbana-Champaign.* David baru-baru ini belajar tentang perintah baru yang dapat dijalankan pada terminal PLATO CERL. PLATO CERL adalah salah satu sistem pembelajaran yang terkomputas pertama dan merupakan pelopor banyak sistem komputas *multi-user* di masa depan. Disebut "external" atau "ext," perintah itu dimaksudkan untuk memungkinkan interaksi dengan perangkat eksternal yang terhubung ke terminal. Namun, ketika dijalankan pada terminal tanpa perangkat eksternal yang terpasang, hal itu akan menyebabkan terminal terkunci sehingga membutuhkan *restart* untuk menghidupkan kembali fungsi.

Karena penasaran untuk melihat seperti apa ruangan yang penuh dengan *user* jika dilakukan penguncian sekaligus. Ia menulis sebuah program yang akan mengirim perintah "ext" ke banyak terminal PLATO pada saat yang bersamaan. Dennis pergi ke CERL dan menguji programnya dan berhasil memaksa 31 *user* untuk melakukan *shutdown.* Akhirnya penerimaan perintah "ext" jarak jauh

dimatikan secara *default*. Selama pertengahan hingga akhir 1990-an, ketika *Internet Relay Chat* (IRC) pertama kali menjadi popular, beberapa *user* berusaha untuk mengontrol saluran obrolan *non-registered*, dimana *user administrator* akan kehilangan fungsi jika ia *logout.* Perilaku ini membuat peretas berusaha memaksa user didalam saluran untuk semua *logout,* sehingga mereka bisa masuk ke saluran itu sendiri dan mendapatkan hak *administrator* sebagai satu-satunya pengguna yang hadir. Pertempuran "*King Of The Hill*" ini di mana pengguna akan berusaha mengendalikan saluran IRC dan menahannya saat menghadapi serangan dari peretas lain, bertempur menggunakan DoS berbasis *bandwidth* yang sangat sederhana dan banjir obrolan IRC[2].

2.2.2 DDOS (*Distributed Denial Of Service*)

Sedangkan serangan DDoS merupakan serangan DoS yang didistribusikan menggunakan banyak komputer atau komputer yang dipaksa untuk menyerang (*Computer Zombie).* Serangan DDoS pertama kali muncul pada tahun 1999, tiga tahun setelah serangan DoS (*Denial of Service*) yang klasik muncul, dengan menggunakan serangan SYN *Flooding,* yang mengakibatkan beberapa server web di internet mengalami "*downtime".* Pada awal Februari 2000, sebuah serangan yang besar dilakukan sehingga beberapa situs *web* terkenal seperti Amazon, CNN, eBay, dan Yahoo! Mengalami "*downtime*" selama beberapa jam. Serangan yang lebih baru lagi pernah dilancarkan pada bulan Obtober 2002 ketika 9 dari 13 *root* DNS server diserang dengan menggunakan DDoS yang sangat besar yang disebut "Png *Flood*". Pada puncak serangan beberapa server tersebut pada tiap detiknya mendapatkan lebih dari 150.000 *request* paket *Internet Control Message Protocol* (ICMP). Untungnya, karena serangan hanya dilakukan selama setengah jam saja, lalulintas internet pun tidak terlalu terpengaruh dengan serangan tersebut[3].



Gambar 2. 2 Contoh Alur Serangan DDoS

Serangan DDoS kerap dilakukan melalui bebrapa metode. Masing cara kerja digunakan tergantung tujuan dari pelaku penyerangan (*hacker*) sendiri. Tidak seperti akibatnya yang menjadi suatu kerumitan yang sangat tinggi bagi para *administrator* jaringan dan server yang melakukan pernaikan server akibat dari serangan, teori dan praktik untuk melakukan serangan DDoS justru sederhana, yakmi sebagai berikut:

1. Menjalankan tool biasanya berupa program kecil yang secara otomatis akan memindai jaringan untuk menemukan host-host yang rentan (*vulneralbe*) yang terkoneksi ke internet. Setelah host yang rentan ditemukan, tool tersebut dapat mengintalasikan salah satu jenis *Trojan Hourse* yang disebut sebagai DDoS Trojan, yang akan mengakibatkan host tersebut menjadi zombie yang dapat di control secara jarak jauh (*remote*) oleh sebuah komputer master yang digunakan oleh si penyerang asli untuk melancarkan serangan.
2. Ketika si penyerang merasa telah mendapat jumlah host yang cukup (*zombie*) untuk melakukan menyerang, penyerang akan menggunakan komputer master untuk memberikan sinyal penyerang terhadap jaringan target atau host target. Serangan ini umumnya dilakukan dengan menggunakan bentuk *Syn Flood* atau skema serangan DoS yang sederhana, tetapi karena dilakukan oleh banyak host zombie, maka jumlah lalulintas jaringan di ciptakan oleh mereka adalah sangat besar sehingga “memakan habis” semua sumber daya *Tranmision Control Protocol* TCP.

Hampir semua *platform* komputer dapat dibajak sebagai sebuah *zombie* untuk melakukan serangan seperti ini. Sistem-sistem popular semacam *Solaris, Linux, Microsoft Windows* dan beberapa varian UNIX dapat menjadi zombie, jika

memang sistem tersebut atau aplikasi yang berjalan di atasnya memiliki kelemahan yang diekploitas oleh penyerang[9].

A. Karakteristik Serangan DDoS

Bentuk serangan terdistribusi DDoS "banyak ke satu" yang membuat serangan ini lebih sulit untuk dicegah. Sebuah serangan DDoS terdiri dari empat elemen. Empat komponen dari serangan DDoS antara lain penyerang, program kontrol utama, daemon serangan/bots, dan korban. Pertama, melibatkan korban, yaitu host target yang telah dipilih untuk menerima beban serangan. Kedua, melibatkan kehadiran agen serangan (daemon) yaitu program agent yang melakukan serangan secara langsung terhadap korban sasaran. Daemon biasanya ditempatkan di komputer inang/perantara. Instalasi daemon pada komputer inang mengharuskan penyerang untuk mendapatkan akses dan berhasil menyusup ke komputer yang menjadi inang daemon. Komponen ketiga dari serangan DDoS adalah program kontrol utama. Tugasnya adalah untuk mengkoordinasikan serangan. Akhirnya, ada penyerang yang menjadi aktor utama di balik serangan DDoS. Penyerang menginisiasikan serangan dengan menggunakan program kontrol utama di belakang layar. Berikut ini adalah langkah-langkah yang terjadi pada serangan terdistribusi

1) Penyerang mengirimkan perintah “eksekusi” yang berupa pesan ke program kontrol utama.
2) Program kontrol utama menerima pesan berupa perintah "eksekusi" dan kemudian menyebarkan perintah penyerangan untuk tiap daemon serangan yang berada di bawah kendalinya.
3) Begitu menerima perintah serangan, daemon serangan memulai serangan terhadap korban.

Meskipun tampaknya pelaku utama serangan DDoS hanya melancarkan aksinya dengan mengirim perintah eksekusi, namun sebenarnya dia benar-benar harus melakukan perencanaan demi serangan DDoS yang berhasil. Penyerang harus menyusup semua host komputer dan jaringan di mana para daemon harus terpasang. Penyerang harus mempelajari topologi jaringan target dan mencari

celah keamanan dan kecendrungan sistem yang dapat dimanfaatkan untuk melancarkan serangan.[10]

B. Metode Serangan DDoS

Secara umum, paket data yang beredar di jaringan menggunakan protokol TCP / IP untuk transmisinya. Paket ini sendiri tidak berbahaya, tetapi jika ada terlalu banyak paket yang abnormal, maka perangkat jaringan atau server akan mengalami kelebihan beban/overload. Kondisi ini dapat dengan cepat mengkonsumsi sumber daya sistem. Kasus lain adalah jika paket serangan memanfaatkan celah keamanan pada protokol tertentu (misalnya request layanan yang tidak lengkap atau penyalahgunaan formasi protokol). Tindakan ini juga dapat menyebabkan kegagalan perangkat jaringan atau server. Kedua pendekatan serangan ini sama-sama mengakibatkan DoS. Kedua pendekatan ini merupakan prinsip-prinsip dasar serangan DDoS. Alasan utama mengapa sulit untuk mencegah serangan DDoS adalah karena pada suatu jaringan, lalu lintas yang sah dan yang ilegal tercampur. Identifikasi akan menjadi semakin sulit, ketika paket data serangan terlihat seperti paket datanormal. Misalnya, dalam sistem Intrusion Detection System berbasis pencocokan pola signature khas, mungkin sulit untuk membedakan pesan ilegal dari pesan yang sah pada awal koneksi. Dalam banyak kasus, abnormalitas pada jaringan baru terlihat ketika serangan terjadi. Secara umum, serangan DDoS dapat dibagi ke dalam jenis berikut:

a) Serangan dengan basis bandwidthSerangan DDoS jenis ini mengirim pesan data sampah secara masal untuk menyebabkan overload, yang juga mengakibatkan berkurangnya bandwidth jaringan yang tersedia atau berkurangnya sumber daya perangkat jaringan. Seringkali router, server dan firewall yang diserang memiliki sumber daya yang terbatas. Serangan overload menyebabkan kegagalan perangkat jaringan untuk menangani akses yang normal, sehingga terjadi penurunan yang signifikan dalam kualitas layanan atau kelumpuhan total sistem (DoS). Dalam kedua kasus itu berarti pengguna tidak dapat mengakses sistem yang mereka butuhkan.

b) Serangan dengan basis lalu lintas jaringan Bentuk yang paling umum adalah serangan yang membanjiri lalu lintas jaringan. Serangan ini dilakukan dengan cara mengirimkan sejumlah besar paket TCP, paket UDP, paket ICPM yang tampaknya sah kepada host/server target. Beberapa serangan dengan basis ini juga dapat menghindari pemindaian sistem deteksi dengan teknologi kamuflase alamat asal. Permintaan yang sah pada akhirnya tidak terlayani karena begitu banyak paket serangan yang beredar di jaringan. Serangan ini juga dapat semakin merusak jika dikombinasikan dengan kegiatan ilegal lainnya, seperti eksploitasi menggunakan malware yang menyebabkan kebocoran informasi/pencurian data sensitif pada komputer target.
c) Serangan dengan basis aplikasi Serangan jenis ini biasanya mengirim pesan data pada tingkat layer aplikasi sesuai dengan fitur bisnis yang spesifik (menggunakan fungsi tampaknya legal dan operasional, seperti akses database), sehingga semakin berkurangnya sumber daya tertentu pada lapisan aplikasi (seperti jumlah pengguna dan koneksi aktif yang diperbolehkan) dan layanan sistem tidak lagi tersedia. Serangan seperti ini biasanya tidak dilancarkan dalam volume yang terlalu besar, serangan dengan lalu lintas tingkat rendah pun dapat menyebabkan gangguan serius pada sistem atau bahkan kelumpuhan kinerja sistem bisnis.[10]

C. Cirri-Ciri Server Web Yang Terkena DDoS

a) Bandwidth mengalami lalu lintas yang sangat padat secara drastis baik download atau upload. Terjadi tiba – tiba dan berlangsung secara terus menerus. Jika target adalah VPS, bisa jadi konsumsi bandwidth akan mencapai batas penggunaan sehingga VPS tidak bisa diakses.
b) Load CPU menjadi sangat tinggi padahal tidak ada proses yang dieksekusi yang mengakibatkan kinerja menjadi menurun sampai dengan website tidak bisa diakses.
c) Jika sistem Anda berada pada penyedia layanan VPS, terkadang ada yang menyediakan layanan informasi jika sewaktu-waktu terjadi aktivitas mencurigakan pada server. Anda juga bisa mengaturnya sendiri.[11]

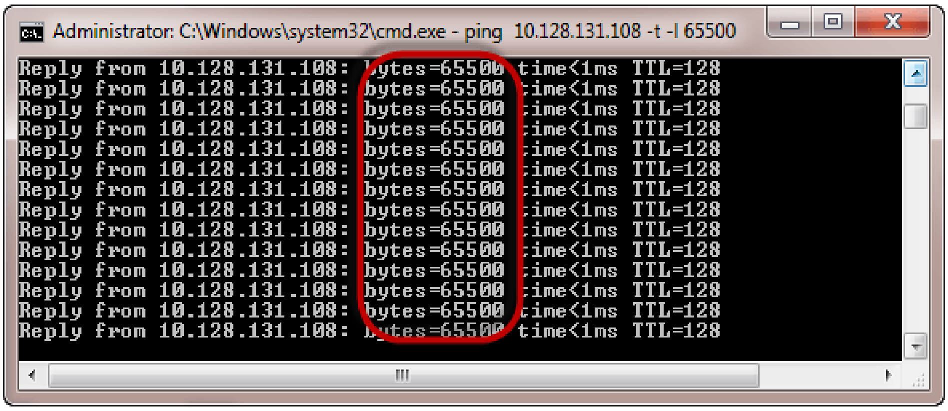
D. Pencegahan Serangan DDoS

1. Ada berbagai cara yang bisa Anda lakukan untuk mencegah serangan DDoS yang membuat sistem berhenti bekerja.
2. Memperbarui sistem operasi ke versi terbaru. Hal ini bertujuan untuk mengatasi menutupi bagian-bagian rentan yang bisa saja dijadikan pintu masuk akses ilegal.
3. Membatasi akses dari dan ke sistem sehingga bisa menyaring trafik data yang masuk dan keluar pada komputer atau server yang Anda gunakan.
4. Jika serangan yang terjadi menggunakan *Smurf*, Anda dapat mencoba mengatasinya dengan cara mematikan sementara *broadcast* address pada router. Bisa juga dengan melakukan penyaringan atau membatasi permintaan ICMP pada *firewall*.
5. Menggunakan perangkat lunak keamanan tambahan untuk sistem.[11]

Berikut beberapa metode umum serangan yang sering digunakan oleh penyerang untuk melakukan serangan DDoS:

1. *Ping Of Death* POD

Metode “Ping” biasanya digunakan untuk mengecek posisi dari sebuah alamat ip situs/website. Namun ditangan orang yang salah, “Ping” kadang digunakan untuk mengirimkan data berukuran hingga 65 kb secara massif. Situs yang memiliki server dengan *bandwidth* rendah tentu akan mengalami kendala pengaksesan. Namun cara ini sudah terbilang tak lagi efektif, mengingat perkembangan jaringan saat ini, situs-situs konvensional sekalipun sudah memiliki *bandwidth* besar. Berikut contoh serangan *Ping Of Death* POD menggunakan CDM pada sistem operasi *Windows*



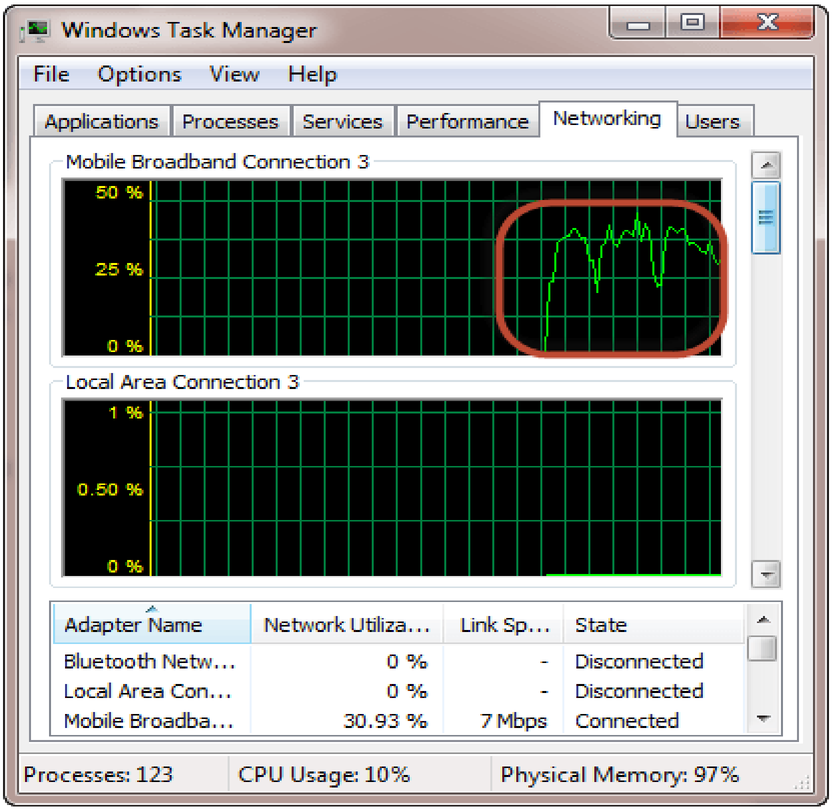
Gambar 2. 3 Contoh Serangan POD

“10.128.131.108” adalah alamat ip korban

“-t” berarti paket data harus dikirim sampai program di hentikan

“-l” menentukan muatan data yang akan di kirimkan ke korban

Berikut contoh dari efek serangan POD dengan melihat performa komputer menggunakan *Task Manager* yang ada pada sistem operasi *Windows*



Gambar 2. 4 Contoh Grafik Serangan POD

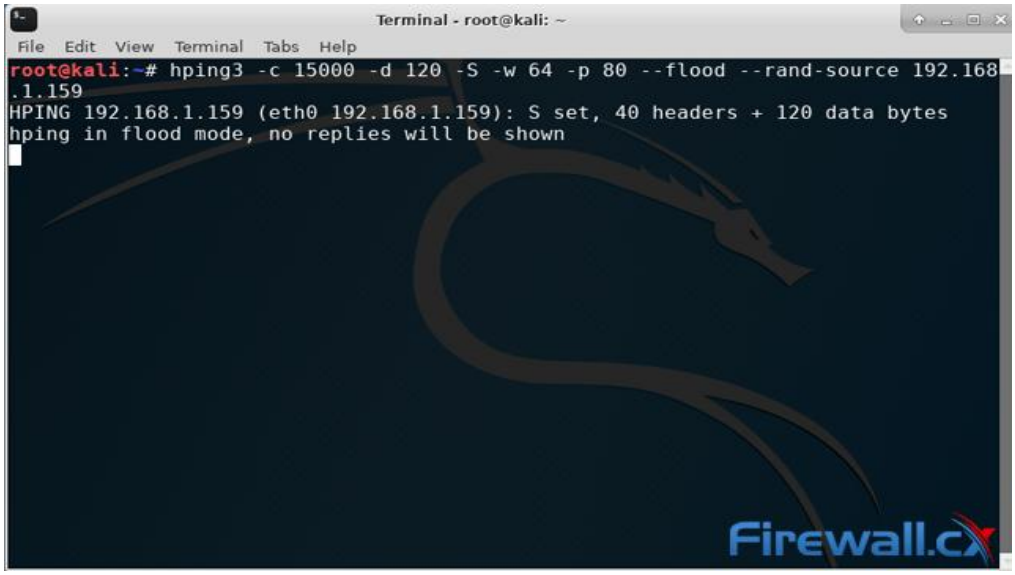
Kita bisa melihat dari grafik di atas terjadi lonjakan aktifitas jaringan, yang berarti serangan POD juga sangat berbahaya juka dilakukan dengan banyak komputer secara bersamaan[12].

2. UDP *Flooding*

Metode serangan ini hampir mirip dengan dengan *Ping Of Death,* yakni mengirimkan paket data secara massif kepada komputer atau server target. Bedanya adalah, pola ini menggunakan protocol UDP sebagai kendaraan serangan. Akibatnya komputer atau server akan kesulitan menangani banjir data dalam jumlah besar. Biasanya komputer akan mengalami *hang.*

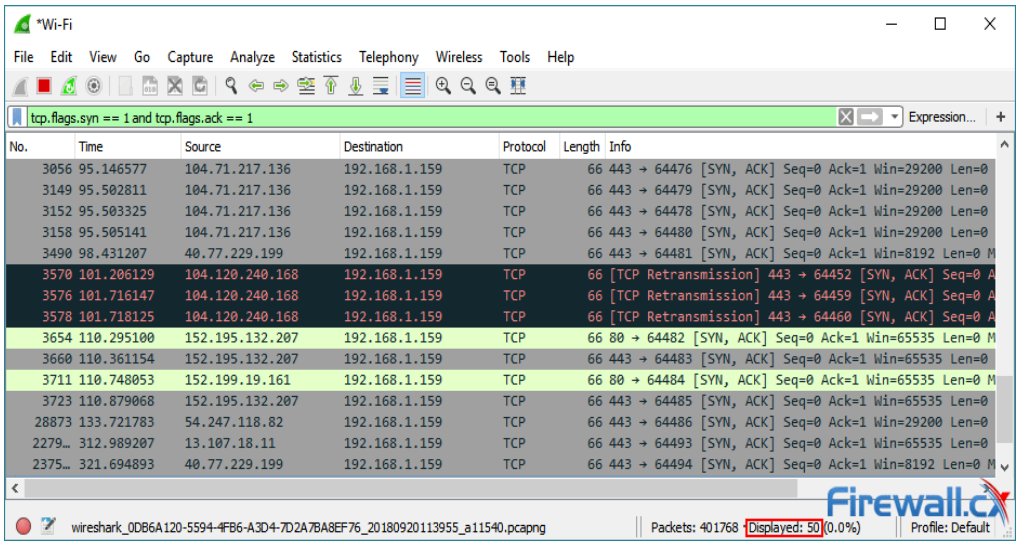
3. *Syn Flooding*

Serangan ini hanya bisa dilakukan saat dua komputer sudah melakukan sambungan komunikasi. Penyerang biasanya akan mengirimkan pesan "Syn ask" kepada komputer target. Lagi-lagi, pesan tersebut dikirimkan dengan jumlah besar. Hasilnya sama. Komputer atau server akan mengalami *hang* karena kelebihan beban. Contoh serangan Syn *Flood* dari System Operasi Kali Linux menggunakan tool hping3 dengan mengirimkan paket 15000 (-c 15000) per 120 data byte (-d 120) kepada target yang ingin di serang dengan ber alamat IP 192.168.1.158 pada port 80 (-p 80) sebagai berikut:

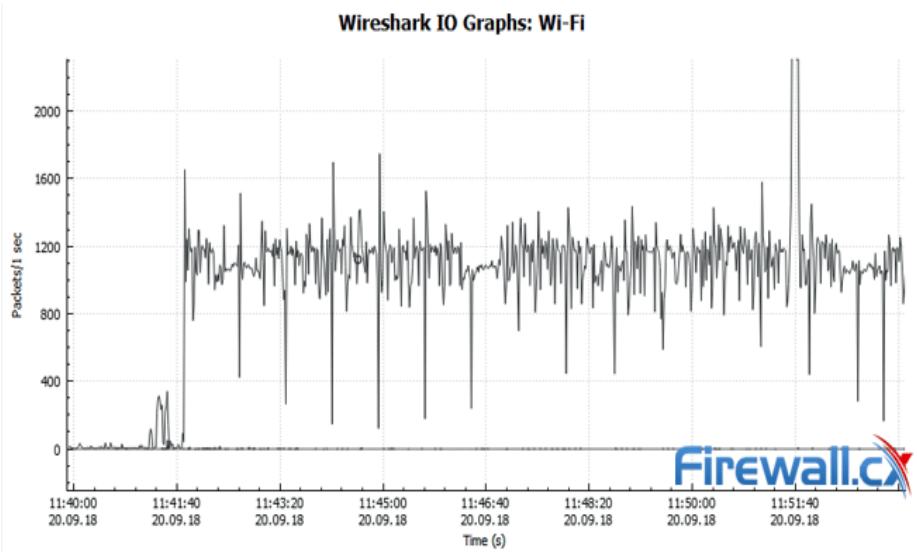


Gambar 2. 5 Contoh Serangan Syn Flooding

Berikut contoh hasil analisa serangan DDoS Syn *Flodding* dengan memanfaatkan analisa jaringan yang ada pada *Wireshark* .



Gambar 2. 6 Contoh Hasil Analisa Syn Flooding Dengan Wireshark



Gambar 2. 7 Contoh Grafik Analisa Serangan Menggunakan Wireshark

Dari hasil analisa di atas terlihat ada paket Syn volume tinggi dengan sedikit variasi waktu. Setiap paket Syn menunjukan berbagai sumber dari IP yang berbeda dengan port tujuan 80 (HTTP). Dan pada grafik *wireshark* menunjukan lonjakan besar dalam paket keseluruhan dari *near* 0 hingga 2400 paket per detik [12].

4. *Remote Control Attack*

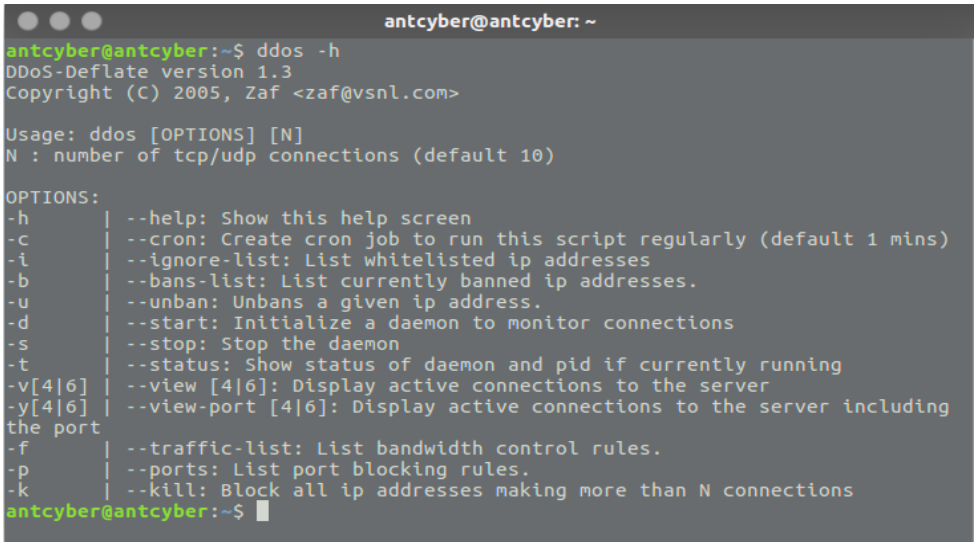
Metode ini terbilang rumit. Layaknya seorang pengendali boneka, pelaku menggunakan beberapa komputer dengan server bersar milik pihak lain untuk menyerang target. Besarnya server komputer yang digunakan untuk menyerang, tentu menimbulkan efek yang besar pula bagi komputer atau server target. Pelaku juga bisa bersembunyi di balik server dan tidak terlacak salah satu tools yang sering digunakan yakni *Metasploit* berikut contoh tampilan dari *Metasploit*.

Gambar 2. 8 Contoh Metasploit

Metasploit Framework merupakan *tool* yang sifatnya *opensources* untuk melakukan uji coba penetrasi serangan serta menyediakan ekploitasi berbagai aplikasi, sistem operasi, dan platform. *Metasploit* juga salah satu *tool* uji coba penetrasi yang paling umum digunakan dan terintegrasi dengan *Kali Linux.* Komponen utama kerangka Metasploit disebut *Module. Module* adalah potongan kode atau perangkat *stand alone* yang menyediakan fungsionalitas untuk *Metasploit.* Terdapat 6 *module*: *exploit*, *payloads*, *auxiliary*, *nops*, *posts*, dan *encoders*[13].

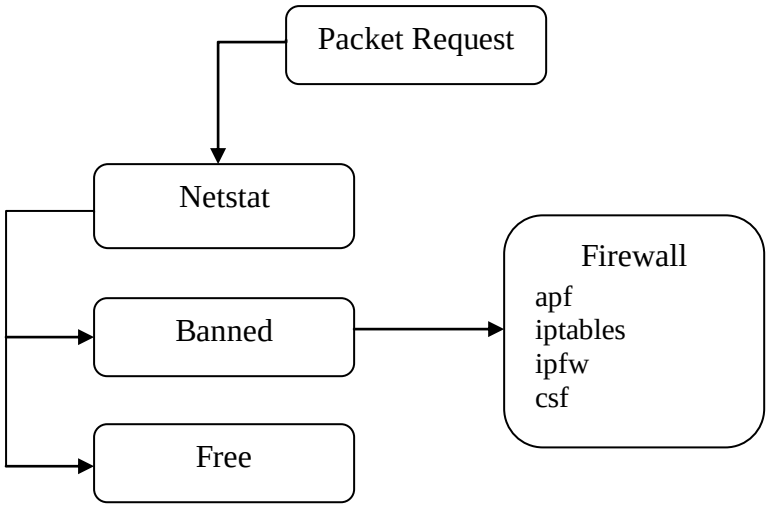
2.2.3 DDOS Deflate

(D)DoS *Deflate* adalah *script bash shell* ringan yang di rancang untuk membantu dalam memblokir serangan DoS pada server web. Itu tidak sepenuhnya melindungi terhadap serangan DDoS besar, tetapi sangat membantu. DDoS *deflate* juga dapat menjadi *firewall* tambahan pada Linux. Setiap kali terdeteksi jumlah koneksi dari satu *node* yang melebihi batas *pretest* tertentu yang di tentukan dalam file konfigurasi, *script* akan secara otomatis memblokir alamat ip tersebut melalui *iptables* atau APF sesuai dengan konfigurasi dan jika terdeteksi sebagai ip yang di curigai sebagai penyerang selanjutnya system akan secara otomatis melakukan pemblokiran ip tersebut dan akan dimasukan pada dafrta hitam (*Black Lis)t* dan apabila diketahui ip tersebu melakukan interaksi melebihi konfigurasi tapi sebagai interaksi yang tidak mencurigakan atau interaksi biasa selanjurnya pengelola dapat memasukan ip tersebut kedalam daftar putih (*White Lis)t* pada konfigurasi. Berikut tampilan awal dari DDoS *Deflate* pada terminal Linux dengan mengetikan perintah ddos -h:



Gambar 2. 9 Contoh tampilan DDoS Deflate

White list merupakan file konfigurasi pada DDoS *Deflate* yang dimana pengelola dapat memasukan ip yang disetujui atau ip yang di izinkan melakukan interaksi lebih pada komputer atau server sedangkan *Black List* merupakan kebalikannya[14]. Berikut cara kerja DDoS Deflate:



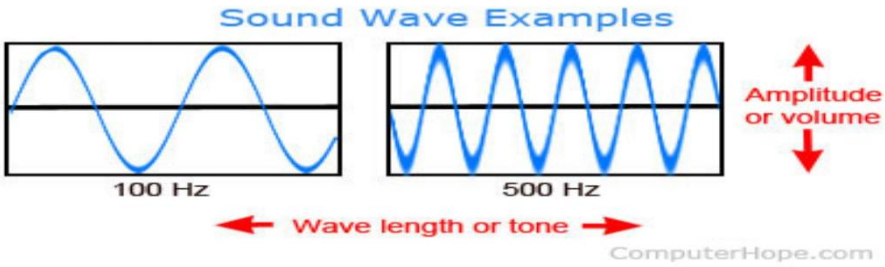
Gambar 2. 10 Cara Kerja DDoS Deflate

Berikut penjelasan dari cara kerja dari DDoS Deflate:

1) *Packet request* merupakan permintaan paket yang dikirimkan oleh penyerang atau seorang pengguna biasa dengan jumlah yang besar.
2) Netstat berfungsi untuk melakukan monitoring jaringan yang terhubung dengan koneksi *port*, antara lain, alamat ip yang terhubung, atau status koneksi apa yang digunakan. *Netstat* juga berfungsi untuk melakak sumber paket yang di dapat.
3) Selanjutnya pada *banned* yakni perintah yang di gunakan setelah terbacanya beberapa *request* paket dalam jumlah besar pada *netstat* dan selanjutnya akan dilakukan pelarangan ip tersebut.
4) *Firewall* merupakan *tool* yang di gunakan untuk melakukan pelarangan terhadap ip atau *port* yang di curigai pada konfigurasi DDoS *Deflate*.
5) Sedangkan *Free* yakni jika tidak terbaca paket dalam jumlah besar pada *netstat*, ip tersebut akan di izinkan untuk melakukan koneksi sampai jumlah paket yang di tentukan.

2.2.4 Sound Alert

Secara umum, suara *Sound* mengacu pada getaran yang di ambil oleh telinga manusia. Saat suara dihasilkan, ia meledak dalam gelombang yang bergetar dalam frekuensi yang di ukur dalam hertz (Hz).

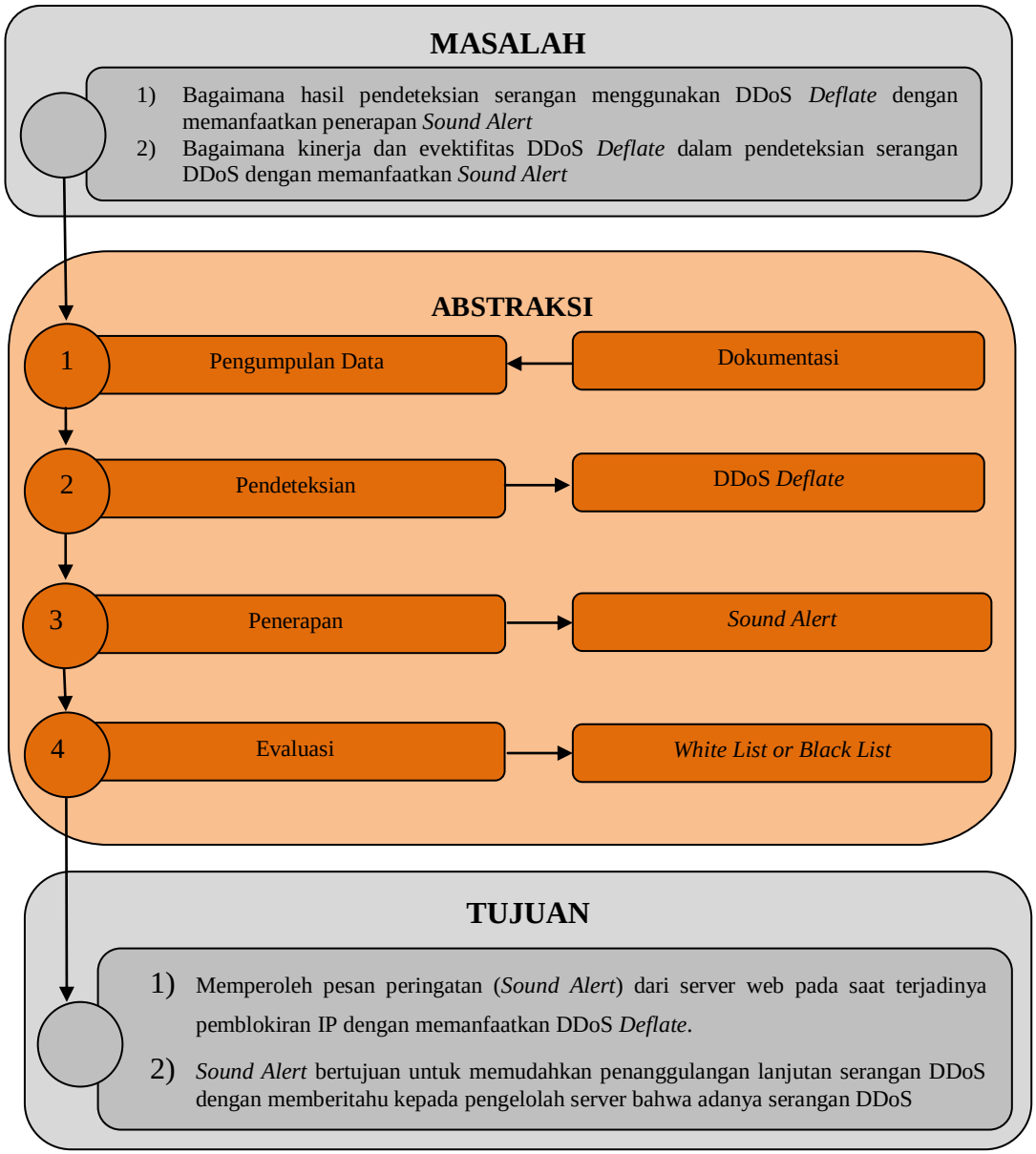


Gambar 2. 11 Contoh Gelombang Suara

Pada gambar di atas adalah dua gelombang suara, nada yang lebih rendah (gelombang yang lebih panjang) disebelah kiri dan nada yang lebih tinggi (gelombang yang lebih pendek) di sebelah kanan. *Sound Alert* adalah bunyi peringatan atau pemberitahuan ketika terjadinya suatu masalah atau memberikan tanda bahaya untuk disampaikan kepada pengelola agar dapat di antisipasi. *Sound Alert* memiliki kelebihan seperti dapat memberikan peringatan dini terhadap bahaya yang akan terjadi sehingga manusia dapat mengantisipasi dan meminimalisir korban jiwa maupun kerugian harta benda. Sedangkan kerugian yang dapat terjadi berupa menyebabkan reaksi positif dan negatif pada manusia. Orang yang mendengan bunyi peringatan dapat mengeluarkan reaksi panik sehingga apa yang dia kerjakan membuat dirinya kesusahan dan bingung[15].

Dalam kehidupan sehari-hari bunyi pesan peringatan atau *Sound Alert* sangat di butuhkan dalam kehidupan kita, contohnya penggunaan *Sound Alert* pada alaram untuk penanda waktu, *Sound Alert* juga digunakan sebagai penanda adanya kebakaran dan atau sebagai penanda adanya sunami. Sehingga peran *Sound Alert* diharapkan sangat dibutuhkan dalam melakukan pendekteksian serangan DDoS demi melakukan pengamanan pada server web yang terkena serangan DDoS.

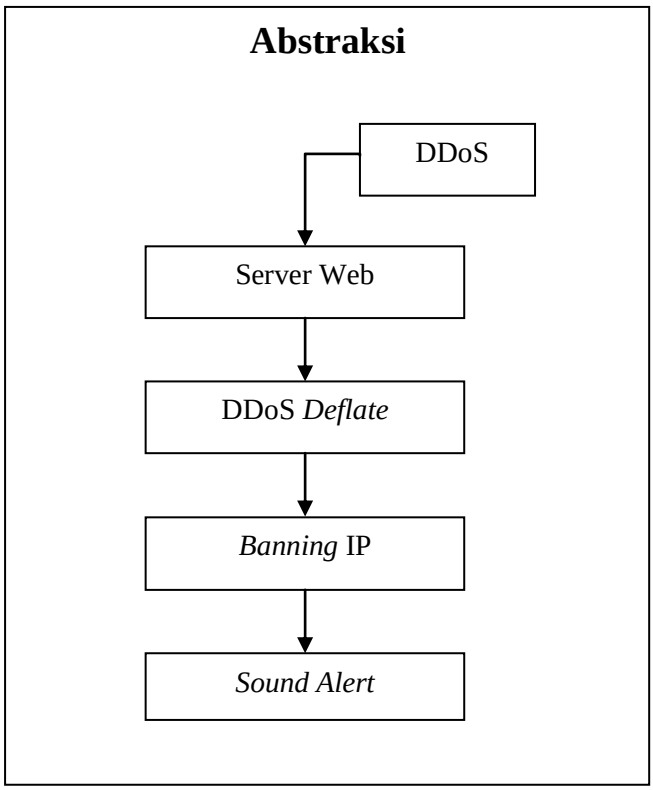
2.3 Kerangka Berpikir



BAB III
METODE PENELITIAN

3.1 Jenis, Metode, Subjek, Objek, Waktu, Dan Lokasi Penelitian

Dipandang dari tingkat penerapannya, maka penelitian ini merupakan penelitian terapan, dipandang dari perlakuan terhadap data, maka penelitian ini merupakan penelitian konfirmatori. Penelitian ini menggunakan metode penelitian *experimen.* Dengan demikian jenis penelitian ini adalah penelitian *eksperimental*. Subjek penelitian ini adalah optimalisasi pada DDoS *Deflate* pada Server web.



Gambar 3. 1 Desain Penelitian

3.2 Pngumpulan Data

3.2.1 Data Primer

Pengumpulan data dengan cara melakukan eksperimen langsung menggunakan dua buah komputer yaiu antara server dan client, dengan mengujicoba seragan ddos pada webserver pada server dan melakukan analisa serangan menggunakan
DDoS *Deflate*.

3.2.2 Data Sekunder

Data Sekunder yaitu Data diperoleh dengan cara mengumpulkan data atau keterangan melalui berbagai macam referensi seperti hasil penelitian terdahulu, buku teks, jurnal yang terkait dari internet yang berhubungan dengan serangan DDoS pada web server dan bagaimana cara menaggulanginaya.

3.3 Analisis Sistem

Sebelum Analisa serangan ddos menggunakan DDoS Deflate pada server web, peneliti melakukan beberapa tahap yang akan dilakukan oleh peneliti diantaranya :

1. Melakukan konfigurasi pada DDoS Deflate agar dapat menyesuaikan pada server web dan sealnjutnya melakukan desain script yang mampu membaca adanya ip pada *black list.*
2. Melakukan tes perbandingan antara server web dengan DDoS Deflate tanpa Sound alert dan server web dengan DDoS Deflate mengguakan Sound Alert

3.4 Desain Sistem

Desain sistem melakukan pendekatan pada pendeteksian kebakaran tapi lebih kepada bagai mana webserver dapat memberikan pesan berupa suara jika seragan DDoS terdeteksi pada web server, dengan spesifikasi sebagai berikut:
Spesifikasi *hardwere* dan *software* server web adalah:

1. Sistem Operasi : Ubuntu 16.04
2. Prosesor Dengan Kecepatan 2,6GHz
3. Memori : 4 GB
4. Harddisk space 1 TB

5. RAM: 4 GB
6. DDoS Deflate Tool
7. Speeker

3.5 Konstuksi Sistem

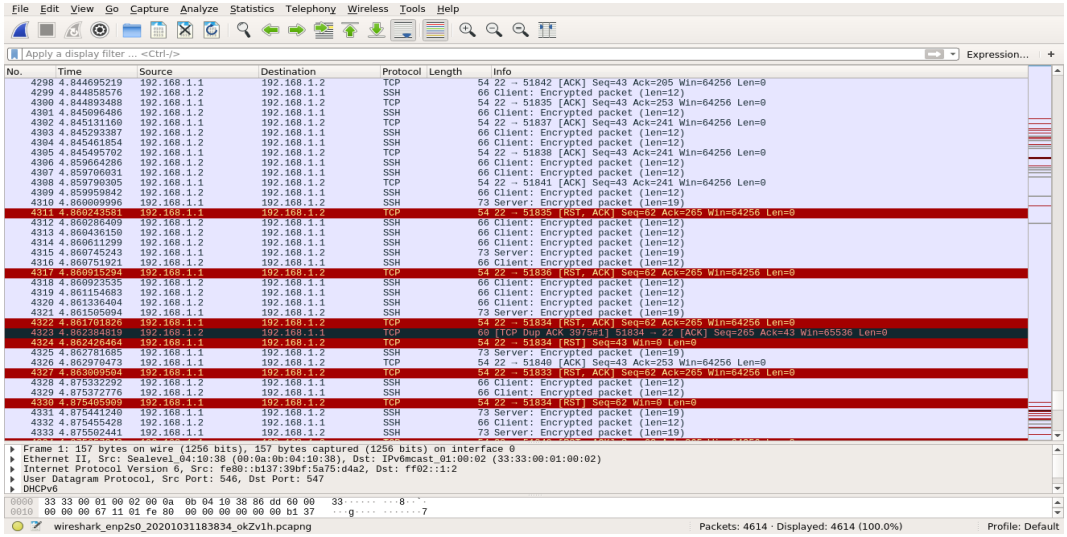
Pada tahap ini dilakukan perancangan pada server web dengan DDoS *Deflate* dengan menambahkan *Sound Alert* agar mampu memberikan pesan peringatan berupa suara para saat adanya ip yang terblokir pada *black list*. untuk menguji, kinerja sistem menggunakan simulasi langsung dengan menyerang langsung pada server web, pada tahap ini juga penulis akan melakukan analisa kinerja dari DDoS *Deflate* termasuk melakukan pemeriksaan tahap yang akan dilakukan jika terjadinya serangan server web.

BAB IV

HASIL PENELITIAN

4.1 Hasil Pengumpulan Data

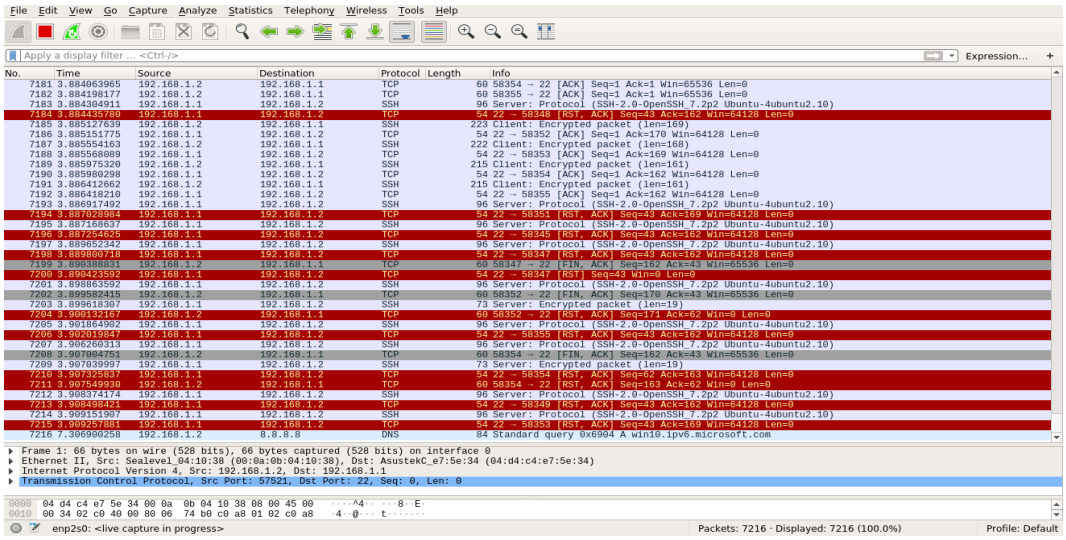
Berikut adalah *traffic* dari serangan DDoS pada server web Ubuntu 16.04 sebelum mengunakan DDoS *Deflate* dan *sound alert* dengan menggunakan metode serangan TCP Flooding. Hasil pemantauan serangan DDoS degan menggunakan *wireshark:*



Gambar 4. 1 Trafik Data DOS TCP

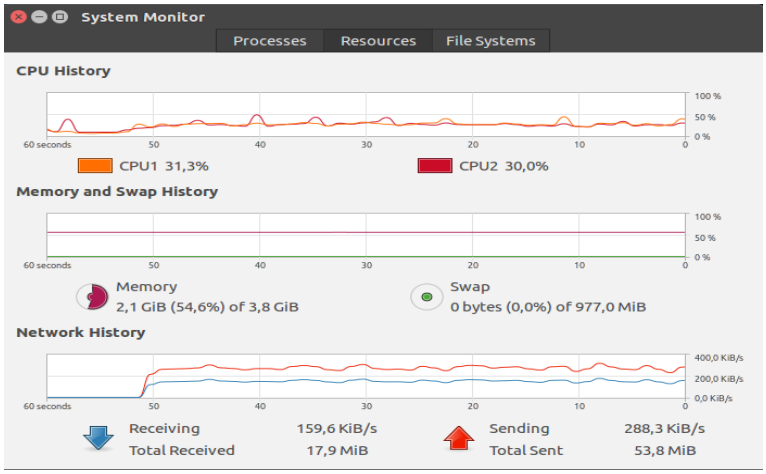
Dari informasi gambar di atas menjelaskan adanya request paket secara berulang yang di tandai dengan besarnya data yang di *reques*. Hal ini dapat membanjiri sirkulasi data dan dapat membuat server kewalahan dalam mengolah data sehingga kita perlukan DDoS *Deflate* untuk mengatur sirkulasi atau smemantau koneksi yang terbuhung keserver. Berikut Kinerja computer pada saat terjadi serangan DDoS pada server web. Pada uji coba selanjutnya menggunakan metode HTTP Flooding yang akan di ujicobakan pada server.

Berikut adalah *traffic* dari serangan DDoS pada server web Ubuntu 16.04 sebelum mengunakan DDoS *Deflate* dan *sound alert* dengan menggunakan metode serangan HTTP *Flooding*. Hasil pemantauan serangan DDoS degan menggunakan *wireshark* sebagai berikut*:*



Gambar 4. 2 Trafic Data DOS HTTP

Dari informasi gambar di atas, sama halnya dengan TCP *Flooding*, HTTP *Flooding* menjelaskan adanya request paket secara berulang yang di tandai dengan besarnya data yang di *reques*. Hal ini dapat membanjiri sirkulasi data dan dapat membuat server kewalahan dalam mengolah data sehingga kita perlukan DDoS *Deflate* untuk mengatur sirkulasi atau smemantau koneksi yang terbuhung keserver. Berikut Kinerja computer pada saat terjadi serangan DDoS pada server web. Namun, serangan DDoS seperti ini tanpa adanya pengamanan pada server mampu membuat suatu server rusak, sehingga di perlukan pengamanan pada sirkulasi jaringan. Berikut diagram kinerja komputer pada saat terkena serangan:



Gambar 4. 3 Kinerja CPU Pada Saat Terjadi Serangan DDoS

Pada gambar di atas juga di jelaskan kinerja komputer mengalami kenaikan pada saat terkena serangan DDoS dari 30,0% sampai dengan 31,3% dengan penggunaan memori 54,6% dan juga mengalami kenaikan pada koneksi jaringan.

4.2 Analisa Dan Perancangan Sistem

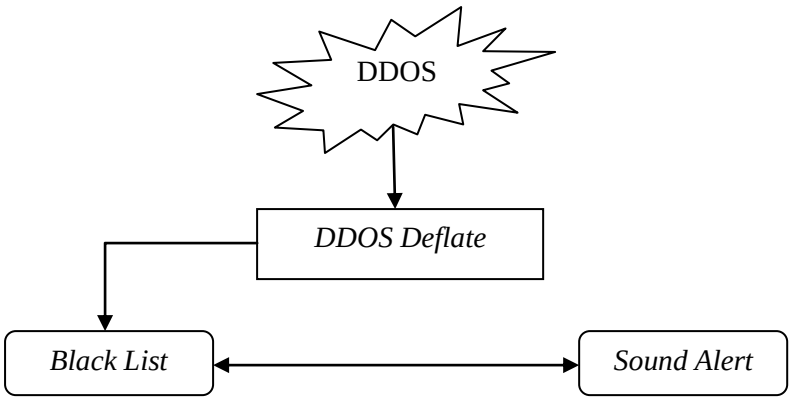
4.2.1 Analisa Sistem

Pada tahap ini penulis akan coba memanfaatkan DDoS *Deflate* untuk melakukan analisa serangan DDoS yang masuk ke server web, adanya serangan DDoS ditandai dengan adanya ip yang terblokir pada *list* pemblokiran. Contohnya dengan *output* pada *black list* 00:09 192.168.1.2 10. Berikut akan di jelaskan fungsinya pada tabel di bawah ini:

Tabel 4. 1 *Black List* DDoS *Deflate Without Sound Alert*

00:09	Lama waktu pemblokiran
192.168.1.2	IP
10	Banyaknya *Request* koneksi

Pada dasarnya DDoS *Deflate* hanya mampu menurunkan serangan DDoS dalam jumlah kecil tapi sangat membantu proses pengamanan sebuah server web. Dalam hal ini membuat pengamanan pada server tidak lebih maksimal karena masih melakukan pemekriksaan secara manual apakah adanya serangan. Dengan begitu DDoS *Deflate* memerlukan sebuah perigatan pendeteksian dengan memanfaatkan *Sound Alert*, sehingga memudahkan pengelolah mendapatkan pesan peringatan dari server bahwa ada ip yang sedang terblokir, ini memudahkan seorang pengelolah untuk melakukan tindakan apa yang dilakukan demi meningkatkan ke amanan server. Berikut alur kerja *Sound Alert* dalam memberitahu adanya serangan pada server.



Gambar 4. 4 Diagram Alur Kerja sound Alert

4.2.2 Perancangan Sound Alert

Untuk melakukan perancangan sistem *Sound Alert* pada DDOS *Deflate* di perlukan pembuatan program *bash* baru yang berfungsi untuk menjalankan dan membaca adakah ip yang terblokir pada *black list* secara berulang selama setiap satu menit, di butuhkan berupa *sound beep* berekstensi *mp3* sebagai bunyi yang akan di jalankan jika program membaca ada ip yang terblokir pada *black list*, dan program *Sound Alert* berfungsi membaca adanya ip yang terblokir baris demi baris. Peulis akan membuat programnya dengan nama *file* sound_alert.sh dengan rancangan program sebagai berikut:

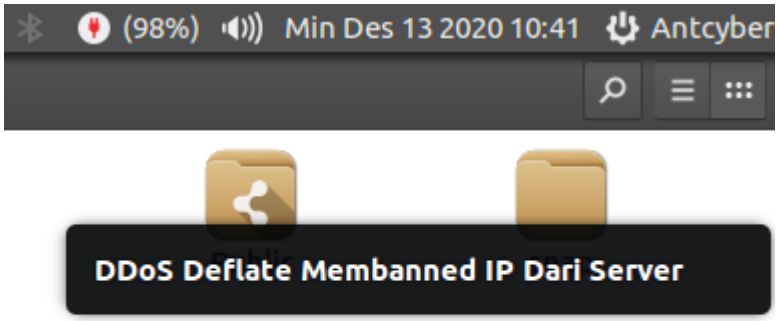
Gambar 4. 5 Rancangan Program Sound Alert

Program ini berjalan pada latar belakang dengan memanfaatkan *cron job* sehingga program mampu melakukan pengecekan setiap satu menit apakah ada ip yang terblokir pada *black list*. Penjelasan pada program akan di jelaskan dalam table berikut:

Tabel 4. 2 Program Perancanzgan *Sound Alert*

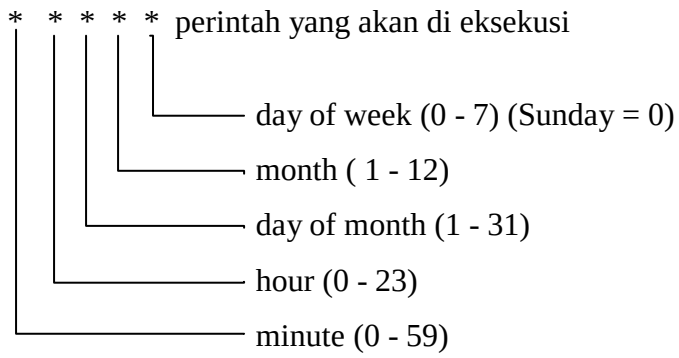
sound_alert.sh	
#!/bin/sh	Fungsi
BANS_IP_LIST=”/var/lib/DDOS/bans.list”	Lokasi black list
while read line; do	Membaca ip yang ada di black list
echo $line && mpv alert.mp3	Jika ada ip yang ter blokir maka akan dilakukan pemutaran alert menggunakan mpv
&& notify-send “DDoS Deflate Mebanned IP Dari Server”	Notifikasi berupa pesan yang muncul pada desktop
done < $BANS_IP_LIST	selesai pengecekan
Exit	Keluar

Berikut tampilan notifikasi yang dihasilkan setelah melakukan *banning* ip addres pada DDoS *Deflate.*



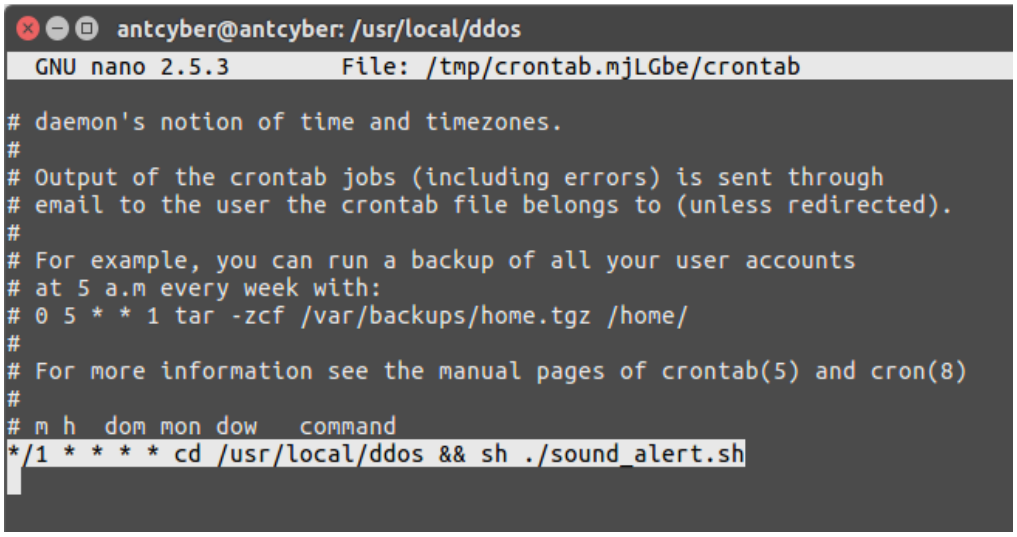
Gambar 4. 6 Notifikasi Setelah Banning IP

Selanjutnya agar program bisa berjalan secara otomatis di latar belakang maka perlu melakukan konfigurasi dan menambahkan perintah pada *crontab* agar di jalankan dalam satu menit setiap saat, dengan perintah */1 * * * * cd /usr/local/ddos && sh ./sound_alert.sh, berikut penjelasan dari penggunaan crontab pada Ubuntu 16.04:



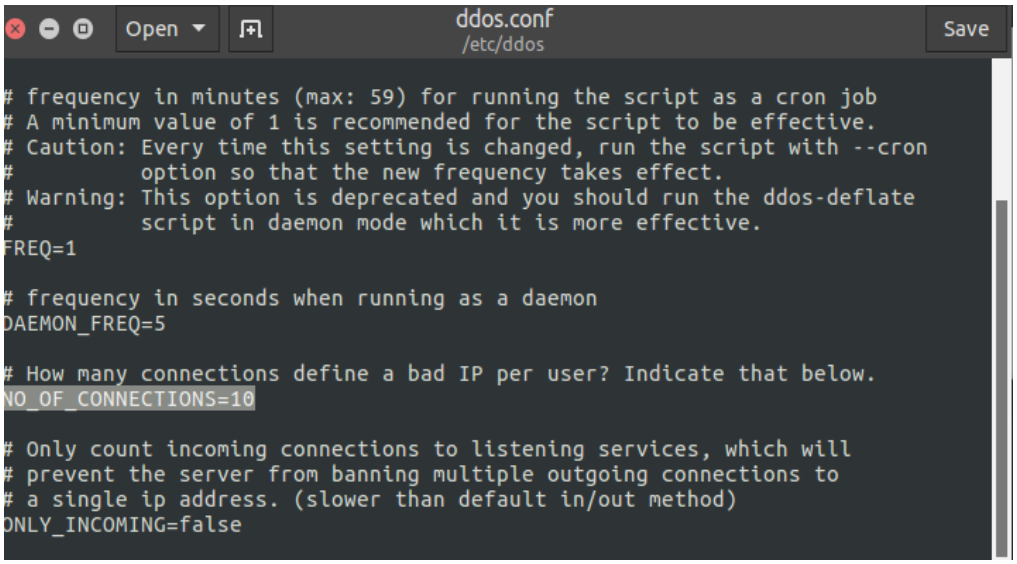
Gambar 4. 7 Penggunaan Penjadwalan Crontab

Untuk menjalankan perintah crontab agar programnya berjalan dalam satu menit setiap saat yaitu dengan menggunakan tanda */1 pada awal contohnya */ * * * * * sehingga crontab dapat mengeksekusi file dalam satu menit setiap saat. Untuk lebih jelasnya bisa di lihat pada gambar konfigurasi berikut:



Gambar 4. 8 Konfigurasi Cron Sound Alert

Pada tahap selanjutnya penulis melakukan konfigurasi pada DDoS *Deflate* dengan membatasi jumlah *request* menjadi 10 koneksi agar lebih *sensitive* pada serangan sehingga memudahkan pembacaan serangan DDoS. Konfigurasi dilakukan pada *file* DDoS *Deflate* dengan perintah *gedit /etc/ddos/ddos.conf*, berikut tampilan dari *file* konfigurasi:



Gambar 4. 9 Konfigurasi Cron Sound Alert

Default pada NO_OF_CONNECTOINS adalah 150, dengan mengubahnyamenjadi 10 kita memfokuskan sensifitas pemblokiran ip jika terjadi serangan DDoS, sehingga lebih mudah untuk melakukan percobaan serangan pada server.

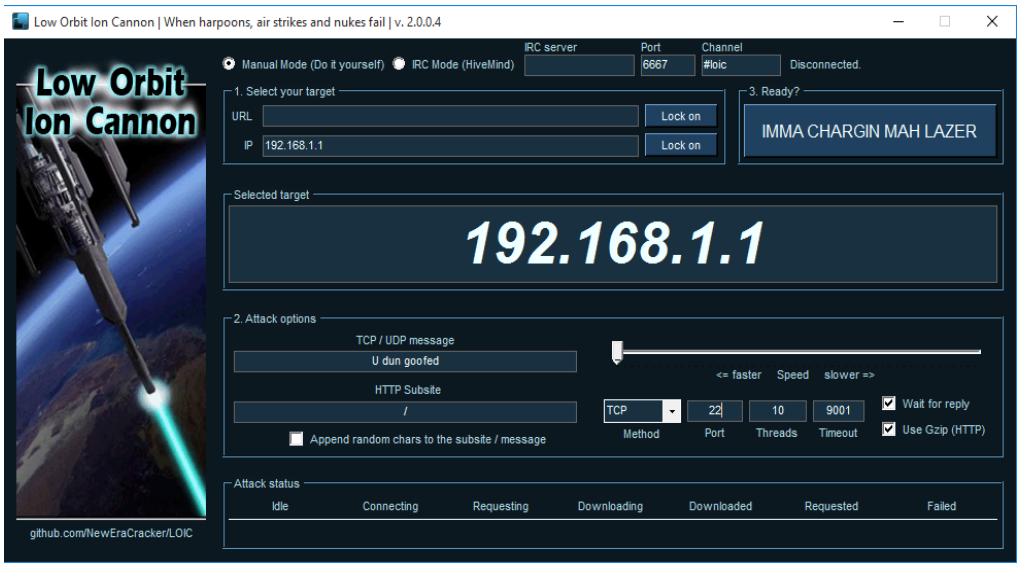
4.2.3 Pengujian Serangan DDoS TCP Flooding

Pengujian dilakukan dengan menggunakan sistem operasi windows 10 dengan ip 192.168.1.2 dan sistem operasi Ubuntu 16.04 dengan ip 192.168.1.1 sebagai server target serangan. Serangan DDoS dilakukan dengan metode SYN *flooding* dengan menggunakan LOIC (*Low Orbit Ion Cannon*). Untuk keterangan perintah sebagai berikut.

Tabel 4. 3 Ujicoba TCP Flooding

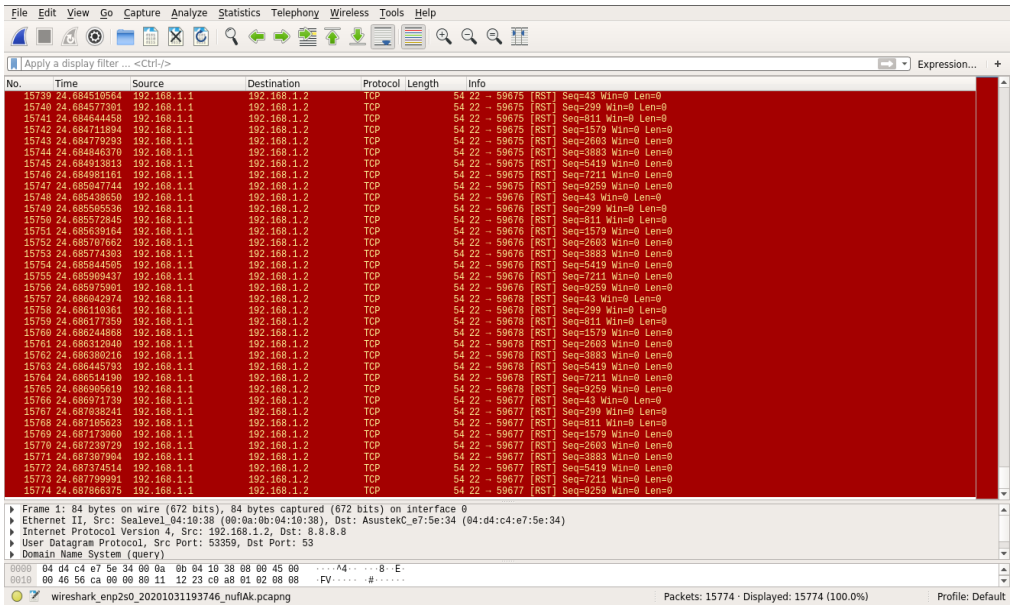
TCP FLOODING	
Selected Target IP	192.168.1.1
Method	TCP
Port	22
Threads	10
Timeoout	9001

Serangan ini masih terbilang cukup ringan dan tidak berdampak kerusakan pada server akan tetapi jika di lakukan dengan banyak computer penyerang mungkin akan berdampak buruk bagi server.



Gambar 4. 10 Ujicoba TCP Flooding dengan LOIC

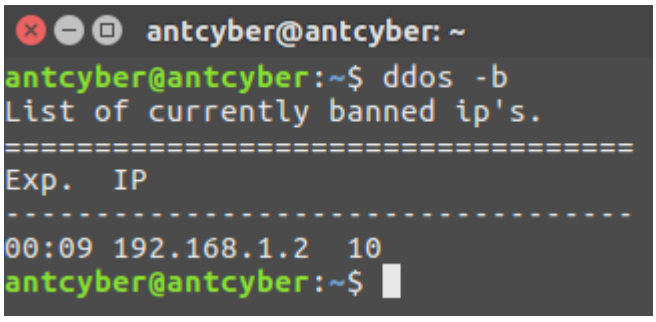
Selanjutnya tinggal menunggu respon dari server tujuan, jika serangan DDoS terdeteksi DDoS *Deflate* akan secara otomatis menganalisa koneksi atau sirkulasi paket yang di kirimkan, jika itu melebihi yang sudah kita konfigurasi sebelumnya maka akan terblokir dan akan di masukan ke *black list* pada DDoS *Deflate*. Berikut tampilan pemblokiran ip yang bisa di lihat menggunakan *wireshark* di bawah ini.



Gambar 4. 11 Hasil TCP Flooding Menggunakan LOIC

Dari hasil pengujian serangan *flooding* pada *protocol* TCP terlihat adanya pemblokiran pada ip 192.168.1.2 yang melakukan *request* berlebih pada server di tandai dengan baris merah pada *wireshark* yang artiya *invalid* atau tidak bisa melukan koneksi kembali, pemblokiran ip tersebut bisa dilihat pada gambar hasil *capture* TCP *flooding* LOIC menggunakan *wireshark*.

Untuk melakukan pengecekan ip yang di blokir pada *black list*, kita dapat melihatnya hasil pemblokiran di black list DDoS *Deflate* dengan mengetikan perintah ddos –b pada terminal. Pemblokiran ip tersebut akan berlangsung selama 600 detik sesuai dengan konfigurasi pada DDoS *Deflate*. Berikut penampakan dari *black list* DDoS *Deflate:*



Gambar 4. 12 Black List IP TCP Flooding

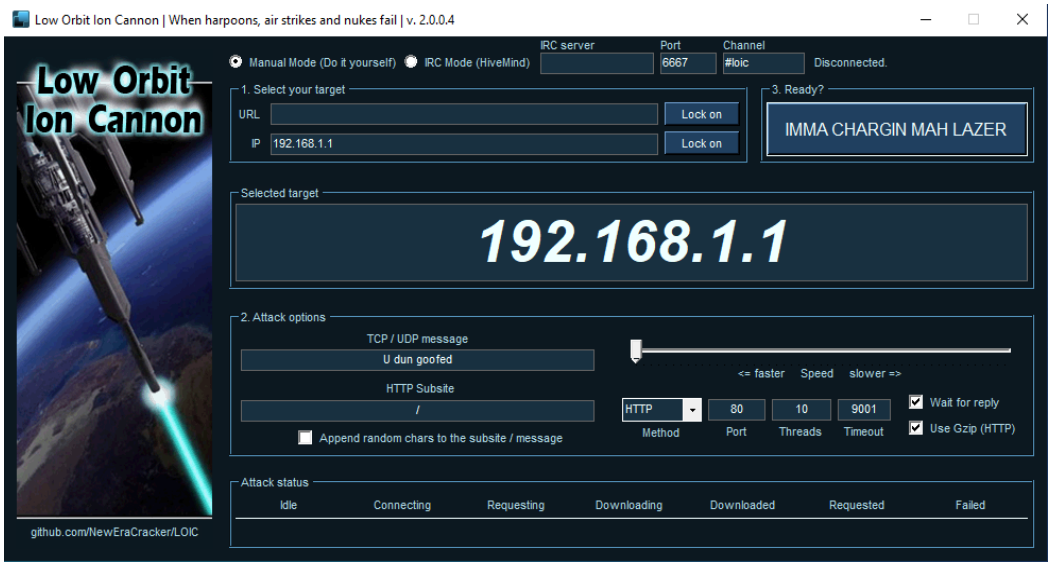
4.2.4 Pengujian Serangan DDoS HTTP Flooding

Pengujian dilakukan dengan menggunakan sistem operasi windows 10 dengan ip 192.168.1.2 dan sistem operasi Ubuntu 16.04 dengan ip 192.168.1.1 sebagai server target serangan. Serangan DDoS dilakukan dengan metode HTTP perintah sebagai berikut:

Tabel 4. 4 HTTP Flooding

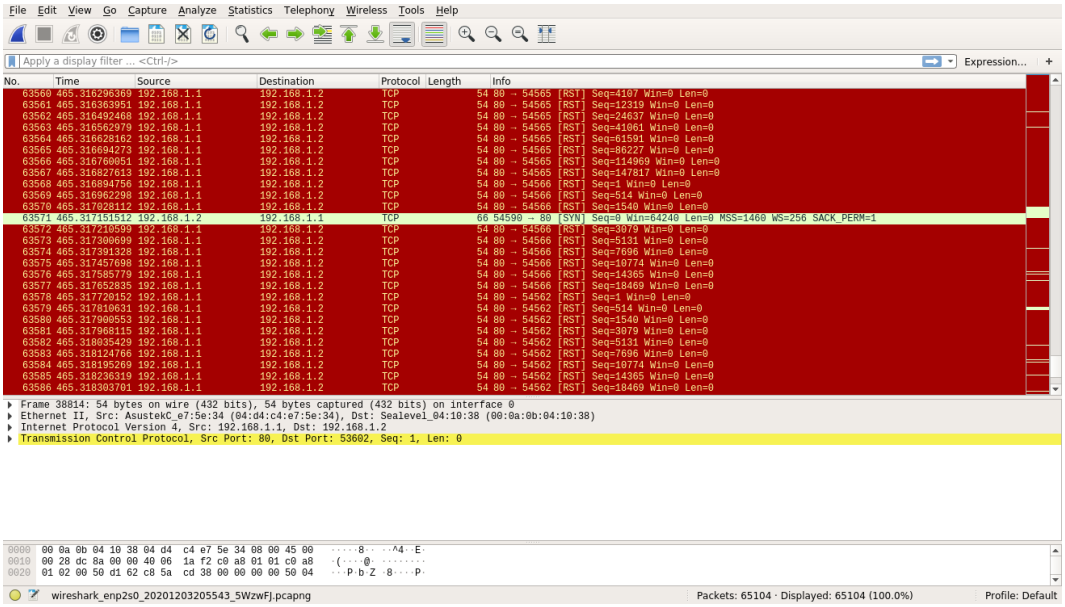
HTTP FLOODING	
Selected Target IP	192.168.1.1
Method	HTTP
Port	80
Threads	10
Timeoout	9001

Sama halnya dengan TCP *flooding*, ujicoba serangan HTTP *flooding* yang di gunakan tidak merubah banyak pada ujicoba seragan, hanya saja penulis mencoba merubah *port* untuk dilakukan ujicoba serangan yakni pada *port* 80, lebih lengkapnya pada gambar berikut:



Gambar 4. 13 LOIC HTTP Flooding

Pada tahap ini terlihat DDoS *Deflate* melakukan pemblokiran pada ip 192.168.1.2 terlihat pada *wireshark* dengan di tandai dengan baris merah pada *wireshark* yang artiya *invalid* atau tidak bisa melukan koneksi kembali.



Gambar 4. 14 Hasil HTTP Flooding Menggunakan LOIC

Samahalnya dengan ujicoba serangan DDoS TCP *flooding* , ip serangan HTTP flooding yang terdeteksi juga akan berada pada black list sebagai berikut:

Gambar 4. 15 Black List IP HTTP Flooding

Dari hasil ujicoba dua serangan TCP *flooding* dan HTTP *flooding* terlihat server mampu mengenali dan respon yang baik terhadap serangan dua serangan tersebut.

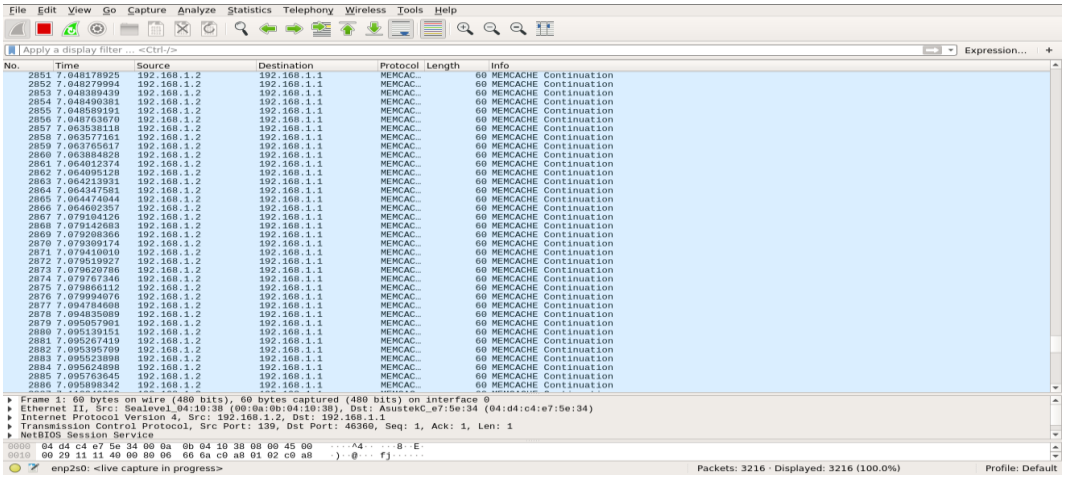
4.2.5 Pengujian serangan UDP FLooding

Pengujian dilakukan dengan menggunakan sistem operasi windows 10 dengan ip 192.168.1.2 dan sistem operasi Ubuntu 16.04 dengan ip 192.168.1.1 sebagai server target serangan. Serangan DDoS dilakukan dengan metode SYN *flooding* dengan menggunakan LOIC (*Low Orbit Ion Cannon*). Untuk keterangan perintah sebagai berikut.:

Tabel 4. 5 Pengujian Serangan UDP Flooding

UDP FLOODING	
Selected Target IP	192.168.1.1
Method	UDP
Port	11211
Threads	10
Timeoout	9001

Ujicoba serangan ini di khususkan untuk membandingkan antara serangan TCP Flooding dan HTTP *Flooding* demi melakukan pemantauan kinerja komputer, serangan ini merupakan serangan yang cukup handal untuk membuat sirkulasi jarinan menjadi sesak. Berikut *capture* serangan dari UDP *Flooding.*



Gambar 4. 16 Hasil UDP Flooding Menggunakan LOIC

Pada serangan ini DDoS Deflate tidak bisa melakukan pendeteksian atau pemblokiran ip serangan karena DDoS Deflate di khususkan untuk melakukan pendeteksian paket pada server web. Sehingga perlu dilakukan pengembangan lebih lanjut. Dengan begitu DDoS Deflate masih bisa dilakukan pengembangan sehingga kedepannya bisa untuk melakukan *scanning* untuk protocol ICMP dan beberapa protocol lainnya.

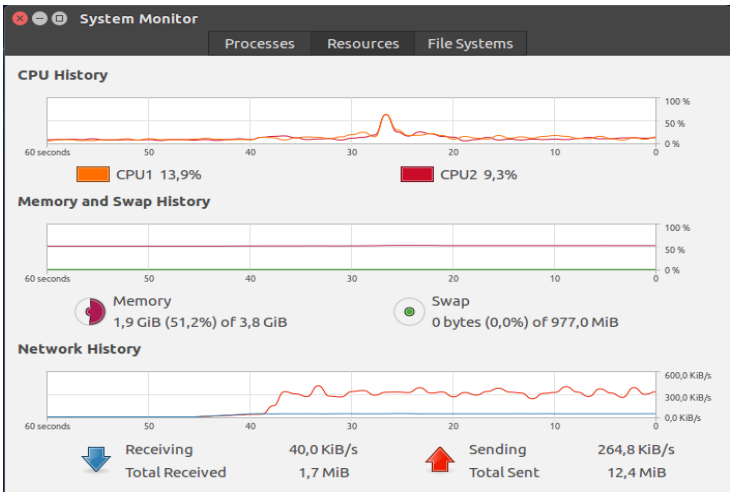
DDoS Deflate itu sendiri masih banyak kekurangan dalam pengamanan pada serangan DDoS, akan tetapi dengan melakukan bebrapa pegembangan seiranya mampu memberikan hasil yang maksimal dalam melakukan pendeteksian dengan memanfaatkan *firewall* untuk melakukan pendeteksian dan pengamanan server web.

BAB V

PEMBAHASAN

5.1 Pembahasan Model

Pada percobaan serangan TCP *flooding* pada server yang belum mengguakan DDoS *Deflate* terlihat adanya aktifitas data dalam keadaan sibuk dengan melayani 10-20 permintaan *request* data dengan cara membanjiri permintaan pada post TCP 22, hal ini membuat kinerja CPU mengalami kenaikan dari 9,3% sampai 13,9% dengan kenaikan pengiriman data 264,8 KiB/s dan menerima data 40,0 KiB/s berikut gambar monitoring CPU sebagai berikut:



Gambar 5. 1 Kinerja CPU Tanpa DDoS Deflate

Sedangkan Server dengan DDoS Deflate terlihat normal dengan kinerja CPU 4,0% sampai 5,0%, dengan kecepatan pengiriman data 106 bytes/s dan menerima data 199 bytes/s. Dengan adanya DDoS *Deflate* membantu memblokir ip yang mengganggu sirkulasi data. Dapat dilihat serangan tersebut mampu membuat server mengalami kenaikan, dari performa dan sirkulasi jaringan mengalami kenaikan drastis. Dengan adanya analisa kinerja komputer sekiranya bisa mengetahui bahayanya serangan DDoS yang di lancarkan pada server web, dan ini sangat merugikan bagi pengelolah server.

5.2 Hasil Pengujian Model

5.2.1 DDOS Deflate Tanpa Menggunakan Sound Alert

Tabel 5. 1 Hasil DDoS *Deflate* Tanpa *Sound Alert*

JENIS SERANGAN	NETWORK		CPU	RESPOND
	Sending	Receive		
TCP Flooding	65,1 Kib/s	292,5 KiB/s	20,0 %	No
HTTP Flooding	159,6 KiB/s	288,3 KiB/s	31,3 %	No

Dari tabel hasil percobaan DDoS *Deflate* tanpa *sound alert* menunjukan tidak adanya respon dari server jika terjadinya serangan DDoS sehingga pengelolah tidak megetahui terjadinya serangan, penyebabnya pengelolah tidak dapat melakukan pengamanan lebih lanjut pada server, maka dari itu pengujian dilanjutkan dengan menggunakan *sound alert* yang berfungsi pada saat ada ip yang terblokir pada DDoS *black list script sound alert* mampu memberikan pesan berupa suara *beep*.

5.2.2 DDOS Deflate Dengan Menggunakan Sound Alert

Tabel 5. 2 DDoS *Deflate* Yang Menggunakan *Sound Alert*

JENIS SERANGAN	NETWORK		CPU	RESPOND
	Sending	Receive		
TCP Flooding	63,9 Kib/s	263,1 KiB/s	21,4 %	Yes
HTTP Flooding	141,8 KiB/s	232,2 KiB/s	30,3 %	Yes

Dari table hasil percobaan DDoS *Deflate* yang menggunakan *Sound Alert* menunjukan adanya respon jika terjadi serangan pada server web ditandai dengan bunyi berupa beep yang sudah dikonfigurasi sebelumnya, sehingga memudahkan pengelolah untuk menyelidiki serangan atau megetahui apa yang di inginkan peyerang selanjutnya. Pemanfaatan *Sound Alert* pada DDoS *Deflate* juga berfungsi untuk mengetahui adanya koneksi mencurigakan pada server web.

BAB IV
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil penelitian yang di lakukan pada server menggunakan DDoS *Deflate* dengan fungsi *Sound Alert* seperti yang sudah di uraikan sebelumnya, maka dapat di tarik berupa kesimpulan bahwa:

1. Dari hasil analisa kinerja DDoS *Deflate* pada server web yang di terapkan mampu melakukan pemblokiran terhadap serangan dan memasukannya ke dalam *black list* jika terdeteksi melebihi konfigurasi.
2. Dengan pemanfaatan *Sound Alert* sekiranya dapat membantu melakukan pemberitahuan adanya serangan pada server, shingga sangat mudah bagi pengelolah untuk melakukan pengamanan lebih pada server web. Contoh mengamanan yang dilakukan melakukan penutupan port dan membatasi user yang melakukan koneksi pada server. Secara umum DDoS *Deflate* di gunakan untuk serangan DoS karena DDoS *Deflate* sangat sensitif dengan serangan kecil sehingga memudahkan mendeteksi serangan DDoS.

6.2 Saran

Setelah melakukan penelitian dan perancangan sistem *Sound Alert* pada DDoS *Deflate,* ada beberapa saran yang perlu di perhatikan untuk mencapai tujuan yang di harapkan. Yakni sebagai berikut:

1. Penggunaan *Sound Alert* ini dapat di kembangkan dengan melakukan *filtering* pada bunyi berdasarkan serangan DDoS, dan dapat di kembangkan mampu melakukan klasifikasi serangan berdasarkan bunyi.
2. Untuk *Sound Alert* dapat mengklasifikasi serangan berdasarkan bunyi, yakni dapat dilakukan beberapa kombinasi atau penambahan *scrip*t pada DDoS *Deflate* dan mengkombinasikan beberapa *tool* seperti *DDoSmon,* ini bertujuan pengelolah dapat melakukan monitoring serta mendapatkan pesan peringatan secara otomatis. DDoS Deflate masih mempunyai kekurangan yakni belum di tambahkan pendeteksian pada *protocol* ICMP sehingga tidak bisa melakukan pemblokiran IP yang berjalan pada *protocol* ICMP.

DAFTAR PUSTAKA

[1] NetscoutTM, "Nerwork Security Infrastucture Report," *Nerwork Security Infrastucture Report*, 2018. https://www.netscout.com/report/.

[2] RadwareTM, "History of DDoS Attacks," *History of DDoS Attacks*, 2017. https://security.radware.com/DDoS-knowledge-center/DDoS-chronicles/DDoS-attacks-history/.

[3] Gito Yudha Pratomo, "DDoS yang Selalu Jadi Senjata Peretas," *DDoS yang Selalu Jadi Senjata Peretas*, 2014. https://cnnindonesia.com/teknologi/20140825162846-192-1758/DDoS-yang-selalu-jadi-senjata-peretas.

[4] Elad Natanson, "Indonesia The New Tiger of Southeast Asia," *Indonesia: The New Tiger Of Southeast Asia*, 2019. https://www.forbes.com/sites/eladnatanson/2019/05/14/indonesia-the-new-tiger-of-southeast-asia/.

[5] Widodo Groho Triatmojo, "Sejarah Tumbangnya Kaskus Ketika Di Serang DDOS Oleh YogyaFree Pada 2008," *Sejarah Tumbangnya Kaskus Ketika Di Serang DDOS Oleh YogyaFree Pada 2008*, 2007. https://www.widodogroho.com/2017/05/sejarah-tumbangnya-kaskus-ketika-di.html.

[6] C. Sheth and R. Thakker, "Performance Evaluation and Comparison of Network Firewalls under DDoS Attack," *Int. J. Comput. Netw. Inf. Secur.*, vol. 5, no. 12, pp. 60–67, 2013, doi: 10.5815/ijcnis.2013.12.08.

[7] J. Mirkovic, M. Robinson, P. Reiher, and G. Oikonomou, "Distributed Defense Against DDoS Attacks," *Defense*, vol. 51, pp. 1–12, 2005, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.2909&rep=rep1&type=pdf.

[8] "Perbandingan Penerapan Metode Pengamanan Web Server Menggunakan Mod Evasive Dan Ddos Deflate Terhadap Serangan Slow Post," 2019, [Online]. Available: http://repository.teknokrat.ac.id/id/eprint/1995.

[9] N. Febrianto, “Macam-macam Serangan DDoS Dan Cara Mengantisipasinya,” *Macam-macam Serangan DDoS Dan Cara Mengantisipasinya*, 2019. https://www.tagar.id/macammacam-serangan-DDoS-dan-cara-mengantisipasinya.

[10] W. W. Septian Geges, “PENGEMBANGAN PENCEGAHAN SERANGAN DISTRIBUTED DENIAL OF SERVICE (DDOS) PADA SUMBER DAYA JARINGAN DENGAN INTEGRASI NETWORK BEHAVIOR ANALYSIS DAN CLIENT PUZZLE,” *Pengemb. Pencegah. SERANGAN Distrib. DENIAL Serv. PADA SUMBER DAYA Jar. DENGAN Integr. Netw. Behav. Anal. DAN CLIENT PUZZLE*.

[11] Y. K, “Pengertian DDOS dan Bagaimana Menanggulanginya,” *Pengertian DDOS dan Bagaimana Menanggulanginya*, 2018. https://www.niagahoster.co.id/blog/ddos-adalah/.

[12] Guru99, “DoS (Denial of Service) Attack Tutorial: Ping of Death, DDOS,” *DoS (Denial of Service) Attack Tutorial: Ping of Death, DDOS*. https://www.guru99.com/ultimate-guide-to-dos-attacks.html.

[13] N. Handy, “Kali Linux & Metasploit: Getting Started with Pen Testing,” *Kali Linux & Metasploit: Getting Started with Pen Testing*, 2018. https://medium.com/cyberdefendersprogram/kali-linux-metasploit-getting-started-with-pen-testing-89d28944097b.

[14] Jgmdev, “DDOS DEFLATE,” *Denial of Service Attack*, 2005. https://github.com/jgmdev/ddos-deflate.

[15] C. Hope, “Sound,” *What Is Sound?*, 2018. https://www.computerhope.com/jargon/s/sound.htm.

Lampiran : 1 Lampiran Script Program

LAMPIRAN SCRIPT

Script DDoS Deflate

```
#!/bin/sh
CONF_PATH="/etc/ddos"
CONF_PATH="${CONF_PATH}/"

# Other variables
BANS_IP_LIST="/var/lib/ddos/bans.list"
BANS_BW_IP_LIST="/var/lib/ddos/bans.bw.list"
BANS_CLOUDFLARE_IP_LIST="/var/lib/ddos/bans.cf.list"
CLOUDFLARE_PCAP="/var/lib/ddos/tcpdump.pcap"
SERVER_IP_LIST=$(ifconfig | \
   grep -E "inet6? " | \
   sed "s/addr: /addr:/g" | \
   awk '{print $2}' | \
   sed -E "s/addr://g" | \
   sed -E "s/\\/[0-9]+//g"
)
SERVER_IP4_LIST=$(ifconfig | \
   grep -E "inet " | \
   sed "s/addr: /addr:/g" | \
   awk '{print $2}' | \
   sed -E "s/addr://g" | \
   sed -E "s/\\/[0-9]+//g" | \
   grep -v "127.0.0.1"
)
SERVER_IP6_LIST=$(ifconfig | \
   grep -E "inet6 " | \
   sed "s/addr: /addr:/g" | \
   awk '{print $2}' | \
   sed -E "s/addr://g" | \
   sed -E "s/\\/[0-9]+//g" | \
   grep -v "::1"
)

SS_MISSING=false

if ! command -v ss>/dev/null; then
   SS_MISSING=true
fi

load_conf()
{
```

```
   CONF="${CONF_PATH}ddos.conf"
   if [ -f "$CONF" ] && [ -n "$CONF" ]; then
      . $CONF
   else
      ddos_head
      echo "\$CONF not found."
      exit 1
   fi
}

ddos_head()
{
   echo "DDoS-Deflate version 1.3"
   echo "Copyright (C) 2005, Zaf <zaf@vsnl.com>"
   echo
}

showhelp()
{
   ddos_head
   echo 'Usage: ddos [OPTIONS] [N]'
   echo 'N : number of tcp/udp connections (default
"'$NO_OF_CONNECTIONS'")'
   echo
   echo 'OPTIONS:'
   echo '-h      | --help: Show this help screen'
   echo '-c      | --cron: Create cron job to run this script regularly (default 1 mins)'
   echo '-i      | --ignore-list: List whitelisted ip addresses'
   echo '-b      | --bans-list: List currently banned ip addresses.'
   echo '-u      | --unban: Unbans a given ip address.'
   echo '-d      | --start: Initialize a daemon to monitor connections'
   echo '-s      | --stop: Stop the daemon'
   echo '-t      | --status: Show status of daemon and pid if currently running'
   echo '-v[4|6] | --view [4|6]: Display active connections to the server'
   echo '-y[4|6] | --view-port [4|6]: Display active connections to the server
including the port'
   echo '-f      | --traffic-list: List bandwidth control rules.'
   echo '-p      | --ports: List port blocking rules.'
   echo '-k      | --kill: Block all ip addresses making more than N connections'
}

# Check if super user is executing the
# script and exit with message if not.
su_required()
{
   user_id=$(id -u)
```

```
    if [ "$user_id" != "0" ]; then
        echo "You need super user priviliges for this."
        exit
    fi
}

log_msg()
{
    if [ ! -e /var/log/ddos.log ]; then
        touch /var/log/ddos.log
        chmod 0640 /var/log/ddos.log
    fi

    echo "$(date +'[%Y-%m-%d %T]') $1" >> /var/log/ddos.log
}

# Gets a list of ip address to ignore with hostnames on the
# ignore.host.list resolved to ip numbers
# param1 can be set to 1 to also include the bans list
ignore_list()
{
    for the_host in $(grep -v "#" "${CONF_PATH}${IGNORE_HOST_LIST}");
do
        host_ip=$(nslookup "$the_host" | tail -n +3 | grep "Address" | awk '{print
$2}')

        # In case an ip is given instead of hostname
        # in the ignore.hosts.list file
        if [ "$host_ip" = "" ]; then
            echo "$the_host"
        else
            for ips in $host_ip; do
                echo "$ips"
            done
        fi
    done

    grep -v "#" "${CONF_PATH}${IGNORE_IP_LIST}"

    if [ "$1" = "1" ]; then
        cut -d" " -f2 "${BANS_IP_LIST}"

        if $ENABLE_CLOUDFLARE; then
            cut -d" " -f2 "${BANS_CLOUDFLARE_IP_LIST}"
        fi
```

```
    fi
}

# Bans a given ip using autodetected firewall or
# ip6tables for ipv6 connections.
# param1 The ip address to block
ban_ip()
{
    if ! echo "$1" | grep ":">/dev/null; then
        if [ "$FIREWALL" = "apf" ]; then
            $APF -d "$1"
        elif [ "$FIREWALL" = "csf" ]; then
            $CSF -d "$1"
        elif [ "$FIREWALL" = "ipfw" ]; then
            rule_number=$(ipfw list | tail -1 | awk '/deny/{print $1}')
            next_number=$((rule_number + 1))
            $IPF -q add "$next_number" deny all from "$1" to any
        elif [ "$FIREWALL" = "iptables" ]; then
            $IPT -I INPUT -s "$1" -j DROP
        fi
    else
        if [ "$FIREWALL" = "ipfw" ]; then
            rule_number=$(ipfw list | tail -1 | awk '/deny/{print $1}')
            next_number=$((rule_number + 1))
            $IPF -q add "$next_number" deny all from "$1" to any
        else
            $IPT6 -I INPUT -s "$1" -j DROP
        fi
    fi
}

# Unbans an ip.
# param1 The ip address
# param2 Optional amount of connections the unbanned ip did.
unban_ip()
{
    if [ "$1" = "" ]; then
        return 1
    fi

    if ! echo "$1" | grep ":">/dev/null; then
        if [ "$FIREWALL" = "apf" ]; then
            $APF -u "$1"
        elif [ "$FIREWALL" = "csf" ]; then
            $CSF -dr "$1"
        elif [ "$FIREWALL" = "ipfw" ]; then
```

```
            rule_number=$($IPF list | awk "/$1/{print $1}")
            $IPF -q delete "$rule_number"
        elif [ "$FIREWALL" = "iptables" ]; then
            $IPT -D INPUT -s "$1" -j DROP
        fi
    else
        if [ "$FIREWALL" = "ipfw" ]; then
            rule_number=$($IPF list | awk "/$1/{print $1}")
            $IPF -q delete "$rule_number"
        else
            $IPT6 -D INPUT -s "$1" -j DROP
        fi
    fi

    if [ "$2" != "" ]; then
        log_msg "unbanned $1 that opened $2 connections"
    else
        log_msg "unbanned $1"
    fi

    grep -v "$1" "${BANS_IP_LIST}" > "${BANS_IP_LIST}.tmp"
    rm "${BANS_IP_LIST}"
    mv "${BANS_IP_LIST}.tmp" "${BANS_IP_LIST}"

    return 0
}

# Unbans ip's after the amount of time given on BAN_PERIOD
unban_ip_list()
{
    current_time=$(date +"%s")

    while read line; do
        if [ "$line" = "" ]; then
            continue
        fi

        ban_time=$(echo "$line" | cut -d" " -f1)
        ip=$(echo "$line" | cut -d" " -f2)
        connections=$(echo "$line" | cut -d" " -f3)

        if [ "$current_time" -gt "$ban_time" ]; then
            unban_ip "$ip" "$connections"
        fi
    done < $BANS_IP_LIST
}
```

```
# Bans a given ip using iptables or ip6tables for ipv6 connections.
# TODO: implement rules for other firewalls, eg: freebsd
# param1 The ip address to block
ban_ip_cloudflare()
{
   if ! echo "$1" | grep ":">/dev/null; then
      $IPT -I INPUT -m string --algo bm --string "CF-Connecting-IP: $1" -j
DROP
   else
      $IPT6 -I INPUT -m string --algo bm --string "CF-Connecting-IP: $1" -j
DROP
   fi
}

# Unbans an ip using iptables or ip6tables for ipv6 connections.
# TODO: implement rules for other firewalls, eg: freebsd
# param1 The ip address
# param2 Optional amount of connections the unbanned ip did.
unban_ip_cloudflare()
{
   if [ "$1" = "" ]; then
      return 1
   fi

   if ! echo "$1" | grep ":">/dev/null; then
      $IPT -D INPUT -m string --algo bm --string "CF-Connecting-IP: $1" -j
DROP
   else
      $IPT6 -D INPUT -m string --algo bm --string "CF-Connecting-IP: $1" -j
DROP
   fi

   if [ "$2" != "" ]; then
      log_msg "unbanned CF $1 that opened $2 connections"
   else
      log_msg "unbanned CF $1"
   fi

   grep -v "$1" "${BANS_CLOUDFLARE_IP_LIST}" >
"${BANS_CLOUDFLARE_IP_LIST}.tmp"
   rm "${BANS_CLOUDFLARE_IP_LIST}"
   mv "${BANS_CLOUDFLARE_IP_LIST}.tmp"
"${BANS_CLOUDFLARE_IP_LIST}"

   return 0
```

```
}

# Unbans ip's after the amount of time given on BAN_PERIOD
unban_ip_list_cloudflare()
{
   current_time=$(date +"%s")

   while read line; do
      if [ "$line" = "" ]; then
         continue
      fi

      ban_time=$(echo "$line" | cut -d" " -f1)
      ip=$(echo "$line" | cut -d" " -f2)
      connections=$(echo "$line" | cut -d" " -f3)

      if [ "$current_time" -gt "$ban_time" ]; then
         unban_ip_cloudflare "$ip" "$connections"
      fi
   done < $BANS_CLOUDFLARE_IP_LIST
}

add_to_cron()
{
   su_required

   echo "Warning: this feature is deprecated and ddos-deflate should" \
        "be run on daemon mode instead."

   if [ "$FREQ" -gt 59 ]; then
      FREQ=1
   fi

   # since this string contains * it is needed to double quote the
   # variable when using it or the * will be evaluated by the shell
   cron_task="*/$FREQ * * * * root $SBINDIR/ddos -k > /dev/null 2>&1"

   if [ "$FIREWALL" = "ipfw" ]; then
      cron_file=/etc/crontab
      sed -i " '/ddos/d' "$cron_file"
      echo "$cron_task" >> "$cron_file"
   else
      rm -f "$CRON"
      echo "$cron_task" > "$CRON"
      chmod 644 "$CRON"
   fi
```

```
   log_msg "added cron job"
}

# Get table of ip connections from ss or netstat. Makes netstat output
# similar to that of ss in order for functions that use this to assume
# ss output format.
# param1 optional ipversion can be 4 o 6
# param2 If not empty means listening services should be returned.
get_connections()
{
   # Find all connections
   if [ "$2" = "" ]; then
      if ! $SS_MISSING; then
         ss -ntu"$1" \
            state $(echo "$CONN_STATES" | sed 's/:/ state /g') | \
            # fixing dependency on '-H' switch which is unavailable in some
versions of ss
            tail -n +2 | \
            # Fix possible ss bug
            sed -E "s/(tcp|udp)/\\1 /g"
      else
         netstat -ntu"$1" | tail -n+3 | grep -E "$CONN_STATES_NS" | \
            # Add [] brackets and prepend dummy column to match ss output
            awk '{
               if($1 == "tcp6" || $1 == "udp6") {
                  gsub(/:[0-9]+$/, "]&", $4)
                  gsub(/:[0-9]+$/, "]&", $5)

                  print "col " $1 " " $2 " " $3 " [" $4 " [" $5;
               } else {
                  print "col " $1 " " $2 " " $3 " " $4 " " $5;
               }
            }'
      fi
   # Find listening services
   else
      if ! $SS_MISSING; then
         # state unconnected used to also include udp services
         ss -ntu"$1" state listening state unconnected | \
            tail -n +2 | \
            # Fix possible ss bug and convert *:### to [::]:###
            sed -E "s/(tcp|udp)/\\1 /g; s/ *:([0-9]+) / [::]:\\1 /g"
      else
         netstat -ntul"$1" | tail -n+3 | \
            # Add [] brackets and prepend dummy column to match ss output
```

```
            awk '{
               if($1 == "tcp6" || $1 == "udp6") {
                  gsub(/:[0-9]+$/, "]&", $4)
                  gsub(/:([0-9]+|\*)$/, "]&", $5)

                  print "col " $1 " " $2 " " $3 " [" $4 " [" $5;
               } else {
                  print "col " $1 " " $2 " " $3 " " $4 " " $5;
               }
            }'
      fi
   fi
}

# Count incoming and outgoing connections and stores those that exceed
# max allowed on a given file.
# param1 File used to store the list of ip addresses as: conn_count ip
ban_incoming_and_outgoing()
{
   whitelist=$(ignore_list "1")

   # Find all connections
   get_connections | \
      # Extract the client ip
      awk '{print $6}' | \
      # Strip port and [ ] brackets
      sed -E "s/\\[//g; s/\\]//g; s/:[0-9]+$//g" | \
      # Only leave non whitelisted, we add ::1 to ensure -v works for ipv6
      grepcidr -v -e "$SERVER_IP_LIST $whitelist ::1" 2>/dev/null | \
      # Sort addresses for uniq to work correctly
      sort | \
      # Group same occurrences of ip and prepend amount of occurences found
      uniq -c | \
      # sort by number of connections
      sort -nr | \
      # Only store connections that exceed max allowed
      awk "{ if (\$1 >= $NO_OF_CONNECTIONS) print; }" > \
      "$1"
}

# Count incoming connections only and stores those that exceed
# max allowed on a given file.
# param1 File used to store the list of ip addresses as: conn_count ip
ban_only_incoming()
{
   whitelist=$(ignore_list "1")
```

```
ALL_LISTENING=$(mktemp "$TMP_PREFIX".XXXXXXXX)
ALL_LISTENING_FULL=$(mktemp "$TMP_PREFIX".XXXXXXXX)
ALL_CONNS=$(mktemp "$TMP_PREFIX".XXXXXXXX)
ALL_SERVER_IP=$(mktemp "$TMP_PREFIX".XXXXXXXX)
ALL_SERVER_IP6=$(mktemp "$TMP_PREFIX".XXXXXXXX)

# Find all connections
get_connections | \
   # Extract both local and foreign address:port
   awk '{print $5" "$6;}' > \
   "$ALL_CONNS"

# Find listening connections
get_connections " " "l" | \
   # Only keep local address:port
   awk '{print $5}' > \
   "$ALL_LISTENING"

# Also append all server addresses when address is 0.0.0.0 or [::]
echo "$SERVER_IP4_LIST" > "$ALL_SERVER_IP"
echo "$SERVER_IP6_LIST" > "$ALL_SERVER_IP6"

awk '
FNR == 1 { ++fIndex }
fIndex == 1{ip_list[$1];next}
fIndex == 2{ip6_list[$1];next}
{
   ip_pos = index($0, "0.0.0.0");
   ip6_pos = index($0, "[::]");
   if (ip_pos != 0) {
      port_pos = index($0, ":");
      print $0;
      for (ip in ip_list){
         print ip substr($0, port_pos);
      }
   } else if (ip6_pos != 0) {
      port_pos = index($0, "]:");
      print $0;
      for (ip in ip6_list){
         print "[" ip substr($0, port_pos);
      }
   } else {
      print $0;
   }
}
```

```
   ' "$ALL_SERVER_IP" "$ALL_SERVER_IP6" "$ALL_LISTENING" >
"$ALL_LISTENING_FULL"

   # Only keep connections which are connected to local listening service
   awk 'NR==FNR{a[$1];next} $1 in a {print $2}' "$ALL_LISTENING_FULL"
"$ALL_CONNS" | \
      # Strip port and [ ] brackets
      sed -E "s/\\[//g; s/\\]//g; s/:[0-9]+$//g" | \
      # Only leave non whitelisted, we add ::1 to ensure -v works
      grepcidr -v -e "$SERVER_IP_LIST $whitelist ::1" 2>/dev/null | \
      # Sort addresses for uniq to work correctly
      sort | \
      # Group same occurrences of ip and prepend amount of occurences found
      uniq -c | \
      # Numerical sort in reverse order
      sort -nr | \
      # Only store connections that exceed max allowed
      awk "{ if (\$1 >= $NO_OF_CONNECTIONS) print; }" > \
      "$1"

   # remove temp files
   rm "$ALL_LISTENING" "$ALL_LISTENING_FULL" "$ALL_CONNS" \
      "$ALL_SERVER_IP" "$ALL_SERVER_IP6"
}

# Count incoming connections on specific ports and stores those
# that exceed max allowed on a given file.
# param1 File used to store the list of ip addresses as:
#       <conn_count> <ip> <port> <ban_time>
ban_by_port()
{
   whitelist=$(ignore_list "1")

   ip_all_list=$(get_connections)
   ip_port_list=$(mktemp "$TMP_PREFIX".XXXXXXXX)
   ip_only_list=$(mktemp "$TMP_PREFIX".XXXXXXXX)

   # Save all connections with port
   echo "$ip_all_list" | \
      # Extract client ip and server port client is connected to
      awk '{
         # Port of server the client connected to
         match($5, /:[0-9]+$/);
         # Strip port from client ip
         gsub(/:[0-9]+$/, "", $6);
         # Print client ip and server port
```

```
            print $6 " " substr($5, RSTART+1, RLENGTH)
        }' | \
        # Strip ipv6 brackets [ ]
        sed -E "s/\\[//g; s/\\]//g;" | \
        # Only leave non whitelisted, we add ::1 to ensure -v works for ipv6
        grepcidr -v -e "$SERVER_IP_LIST $whitelist ::1" 2>/dev/null | \
        # Sort addresses for uniq to work correctly
        sort | \
        # Group same occurrences of ip port and prepend amount of occurences
found
        uniq -c | \
        # sort by number of connections
        sort -nr > \
        "$ip_port_list"

    # Save all connections without port
    echo "$ip_all_list" | \
        # Extract the client ip
        awk '{print $6}' | \
        # Strip port and [ ] brackets
        sed -E "s/\\[//g; s/\\]//g; s/:[0-9]+$//g" | \
        # Only leave non whitelisted, we add ::1 to ensure -v works for ipv6
        grepcidr -v -e "$SERVER_IP_LIST $whitelist ::1" 2>/dev/null | \
        # Sort addresses for uniq to work correctly
        sort | \
        # Group same occurrences of ip and prepend amount of occurences found
        uniq -c | \
        # sort by number of connections
        sort -nr > \
        "$ip_only_list"

    unset ip_all_list

    # Analyze all connections by port
    for port in $(echo "$PORT_CONNECTIONS" | xargs); do
        number=$(echo "$port" | cut -d":" -f1)
        max_conn=$(echo "$port" | cut -d":" -f2)
        ban_time=$(echo "$port" | cut -d":" -f3)

        # Handle port ranges eg: 2000-2010
        if echo "$number" | grep "-">/dev/null; then
            number_start=$(echo "$number" | cut -d"-" -f1)
            number_end=$(echo "$number" | cut -d"-" -f2)

            awk -v pstart="$number_start" -v pend="$number_end" \
                -v mconn="$max_conn" -v btime="$ban_time" \
```

```
            -v mgconn="$NO_OF_CONNECTIONS" '
            FNR == 1 { ++fIndex }
            fIndex == 1 {
               # Match only ports in range
               if ($3 >= pstart && $3 <= pend) {
                  # store amount of connections for the ip
                  ip_port_list[$2]+=$1;
               }
               next
            }
            fIndex == 2 {
               # store global amount of connections per ip
               ip_all_list[$2]=$1;
               next
            }
            END {
               # Print all ip addresses that should be ban
               for (ip in ip_port_list) {
                  if(ip_port_list[ip] >= mconn || (ip_all_list[ip] > ip_port_list[ip] &&
ip_all_list[ip] >= mgconn)) {
                     print ip_port_list[ip] " " ip " " pstart "-" pend " " btime;
                  }
               }
            }' "$ip_port_list" "$ip_only_list" >> "$1"
      # Handle specific ports eg: 80
      else
         awk -v portn="$number" -v mconn="$max_conn" \
            -v btime="$ban_time" -v mgconn="$NO_OF_CONNECTIONS" '
            FNR == 1 { ++fIndex }
            fIndex == 1 {
               # Match only exact port
               if ($3 == portn) {
                  # store amount of connections for the ip
                  ip_port_list[$2]=$1;
               }
               next
            }
            fIndex == 2 {
               # store global amount of connections per ip
               ip_all_list[$2]=$1;
               next
            }
            END {
               # Print all ip addresses that should be ban
               for (ip in ip_port_list) {
```

```
                    if(ip_port_list[ip] >= mconn || (ip_all_list[ip] > ip_port_list[ip] &&
ip_all_list[ip] >= mgconn)) {
                        print ip_port_list[ip] " " ip " " portn " " btime;
                    }
                }
            }' "$ip_port_list" "$ip_only_list" >> "$1"
        fi
    done

    rm "$ip_port_list" "$ip_only_list"
}

# Add list of current cloudflare ip's to a given file or print it
# if no file is given.
ban_cloudflare()
{
    whitelist=$(ignore_list "1")

    if [ "$1" != "" ]; then
        tcpdump -r "$CLOUDFLARE_PCAP" -n -A 2>/dev/null | \
            # filter tcpdump data to the CF needed header
            grep "CF-Connecting-IP:" | \
            # Extract ip
            cut -d' ' -f2 | \
            # Only leave non whitelisted, we add ::1 to ensure -v works for ipv6
            grepcidr -v -e "$SERVER_IP_LIST $whitelist ::1" 2>/dev/null | \
            # Sort addresses for uniq to work correctly
            sort | \
            # Group same occurrences of ip and prepend amount of occurences found
            uniq -c | \
            # sort by number of connections and save to file
            sort -nr | \
            # Only store connections that exceed max allowed
            awk "{ if (\$1 >= $NO_OF_CONNECTIONS) print; }" > "$1"
    else
        tcpdump -r "$CLOUDFLARE_PCAP" -n -A 2>/dev/null | \
            # filter tcpdump data to the CF needed header
            grep "CF-Connecting-IP:" | \
            # Extract ip
            cut -d' ' -f2 | \
            # Only leave non whitelisted, we add ::1 to ensure -v works for ipv6
            grepcidr -v -e "$SERVER_IP_LIST $whitelist ::1" 2>/dev/null | \
             # Sort addresses for uniq to work correctly
            sort | \
            # Group same occurrences of ip and prepend amount of occurences found
            uniq -c | \
```

```
        # sort by number of connections
        sort -nr
    fi
}

# Check active cloudflare connections and ban if neccessary.
check_connections_cloudflare()
{
    su_required

    TMP_PREFIX='/tmp/ddos'
    TMP_FILE="mktemp $TMP_PREFIX.XXXXXXXX"
    BAD_IP_LIST=$($TMP_FILE)

    ban_cloudflare "$BAD_IP_LIST"

    FOUND=$(cat "$BAD_IP_LIST")

    if [ "$FOUND" = "" ]; then
        rm -f "$BAD_IP_LIST"

        if [ "$KILL" -eq 1 ]; then
            echo "No connections exceeding max allowed."
        fi

        return 0
    fi

    if [ "$KILL" -eq 1 ]; then
        echo "List of connections that exceed max allowed"
        echo "==========================================="
        cat "$BAD_IP_LIST"
    fi

    BANNED_IP_MAIL=$($TMP_FILE)
    BANNED_IP_LIST=$($TMP_FILE)

    echo "Banned the following ip addresses on $(date)" > "$BANNED_IP_MAIL"
    echo >> "$BANNED_IP_MAIL"

    IP_BAN_NOW=0

    FIELDS_COUNT=$(head -n1 "$BAD_IP_LIST" | xargs | sed "s/ /\\n/g" | wc -l)

    while read line; do
        BAN_TOTAL="$BAN_PERIOD"
```

```
        CURR_LINE_CONN=$(echo "$line" | cut -d" " -f1)
        CURR_LINE_IP=$(echo "$line" | cut -d" " -f2)

        IP_BAN_NOW=1

        echo "${CURR_LINE_IP} with $CURR_LINE_CONN connections" >>
"$BANNED_IP_MAIL"
        echo "${CURR_LINE_IP}" >> "$BANNED_IP_LIST"

        current_time=$(date +"%s")
        echo "$((current_time+BAN_TOTAL)) ${CURR_LINE_IP}
${CURR_LINE_CONN}" >> \
            "${BANS_CLOUDFLARE_IP_LIST}"

        ban_ip_cloudflare "$CURR_LINE_IP"

        log_msg "banned ${CURR_LINE_IP} with $CURR_LINE_CONN
connections for ban period $BAN_TOTAL"
    done < "$BAD_IP_LIST"

    if [ "$IP_BAN_NOW" -eq 1 ]; then
        if [ -n "$EMAIL_TO" ]; then
            dt=$(date)
            hn=$(hostname)
            cat "$BANNED_IP_MAIL" | mail -s "[$hn] Cloudflare IP addresses
banned on $dt" $EMAIL_TO
        fi

        if [ "$KILL" -eq 1 ]; then
            echo "============================================"
            echo "Banned CloudFlare IP addresses:"
            echo "============================================"
            cat "$BANNED_IP_LIST"
        fi
    fi

    rm -f "$TMP_PREFIX".*
}

# Check active connections and ban if neccessary.
check_connections()
{
    su_required

    TMP_PREFIX='/tmp/ddos'
```

```
TMP_FILE="mktemp $TMP_PREFIX.XXXXXXXX"
BAD_IP_LIST=$($TMP_FILE)

if $ENABLE_PORTS; then
   ban_by_port "$BAD_IP_LIST"
elif $ONLY_INCOMING; then
   ban_only_incoming "$BAD_IP_LIST"
else
   ban_incoming_and_outgoing "$BAD_IP_LIST"
fi

FOUND=$(cat "$BAD_IP_LIST")

if [ "$FOUND" = "" ]; then
   rm -f "$BAD_IP_LIST"

   if [ "$KILL" -eq 1 ]; then
      echo "No connections exceeding max allowed."
   fi

   return 0
fi

if [ "$KILL" -eq 1 ]; then
   echo "List of connections that exceed max allowed"
   echo "========================================="
   cat "$BAD_IP_LIST"
fi

BANNED_IP_MAIL=$($TMP_FILE)
BANNED_IP_LIST=$($TMP_FILE)

echo "Banned the following ip addresses on $(date)" > "$BANNED_IP_MAIL"
echo >> "$BANNED_IP_MAIL"

IP_BAN_NOW=0

FIELDS_COUNT=$(head -n1 "$BAD_IP_LIST" | xargs | sed "s/ /\\n/g" | wc -l)

while read line; do
   BAN_TOTAL="$BAN_PERIOD"

   CURR_LINE_CONN=$(echo "$line" | cut -d" " -f1)
   CURR_LINE_IP=$(echo "$line" | cut -d" " -f2)

   if [ "$FIELDS_COUNT" -gt 2 ]; then
```

```
            CURR_LINE_PORT=$(echo "$line" | cut -d" " -f3)
            BAN_TOTAL=$(echo "$line" | cut -d" " -f4)
        else
            CURR_LINE_PORT=""
        fi

        if [ "$CURR_LINE_PORT" != "" ]; then
            CURR_PORT=":$CURR_LINE_PORT"
        fi

        IP_BAN_NOW=1

        echo "${CURR_LINE_IP}${CURR_PORT} with $CURR_LINE_CONN
connections" >> "$BANNED_IP_MAIL"
        echo "${CURR_LINE_IP}${CURR_PORT}" >> "$BANNED_IP_LIST"

        current_time=$(date +"%s")
        echo "$((current_time+BAN_TOTAL)) ${CURR_LINE_IP}
${CURR_LINE_CONN} ${CURR_LINE_PORT}" >> \
            "${BANS_IP_LIST}"

        # execute tcpkill for 60 seconds
        timeout -k 60 -s 9 60 \
            tcpkill -9 host "$CURR_LINE_IP" > /dev/null 2>&1 &

        ban_ip "$CURR_LINE_IP"

        log_msg "banned ${CURR_LINE_IP}${CURR_PORT} with
$CURR_LINE_CONN connections for ban period $BAN_TOTAL"
    done < "$BAD_IP_LIST"

    if [ "$IP_BAN_NOW" -eq 1 ]; then
        if [ -n "$EMAIL_TO" ]; then
            dt=$(date)
            hn=$(hostname)
            cat "$BANNED_IP_MAIL" | mail -s "[$hn] IP addresses banned on $dt"
$EMAIL_TO
        fi

        if [ "$KILL" -eq 1 ]; then
            echo "========================================="
            echo "Banned IP addresses:"
            echo "========================================="
            cat "$BANNED_IP_LIST"
        fi
    fi
```

```
   rm -f "$TMP_PREFIX".*
}

# Check if a connection is exceeding the BANDWIDTH_CONTROL_LIMIT
# and applies a banning rule accordingly.
check_connections_bw()
{
   if ! $BANDWIDTH_CONTROL; then
      return
   fi

   whitelist=$(ignore_list "1")

   TMP_PREFIX='/tmp/ddos'
   TMP_FILE="mktemp $TMP_PREFIX.XXXXXXXX"

   BANNED_IP_MAIL=$($TMP_FILE)
   BANNED_IP_LIST=$($TMP_FILE)

   echo "Dropped transfer limit to following ip addresses on $(date)" >
"$BANNED_IP_MAIL"
   echo >> "$BANNED_IP_MAIL"

   ip_drop_rate=0

   onlyincoming_bw=""

   if $BANDWIDTH_ONLY_INCOMING; then
      onlyincoming_bw="1"
   fi

   iftop -nNt -s3 2>/dev/null | tail -n+4 | head -n-9 | \
      # Check for clients that reach the BANDWIDTH_CONTROL_LIMIT
      awk -v ratelimit="$BANDWIDTH_CONTROL_LIMIT" \
         -v onlyincoming="$onlyincoming_bw" '
         BEGIN {
            if(index(ratelimit, "mbit") > 0) {
               gsub(/mbit$/, "", ratelimit);
               # convert to kbit
               ratelimit*=1000;
            } else if(index(ratelimit, "kbit") > 0) {
               gsub(/kbit$/, "", ratelimit);
               ratelimit+=0; # force numeric cast
            }
         }
```

```
        {
          # Handle incoming then outgoing
          if(index($0, "=>") > 0) {
            # store outgoing bandwidth
            if(!onlyincoming) {
              bandwidth[0]=$4;
            }

            # skip to incoming line
            getline;

            # store client ip and incoming bandwidth
            client_ip=$1;
            bandwidth[1]=$3;

            bandwidth_total=0;
            for(i in bandwidth) {
              if(index(bandwidth[i], "Gb") > 0) {
                gsub(/Gb$/, bandwidth[i]);
                bandwidth_total+=bandwidth[i]*1000*1000;
              } else if(index(bandwidth[i], "Mb") > 0) {
                gsub(/Mb$/, bandwidth[i]);
                bandwidth_total+=bandwidth[i]*1000;
              } else if(index(bandwidth[i], "Kb") > 0) {
                gsub(/Kb$/, bandwidth[i]);
                bandwidth_total+=bandwidth[i];
              }
            }

            if(bandwidth_total >= ratelimit) {
              print client_ip " " bandwidth_total "Kb"
            }
          }
        }
      ' | \
      # Only leave non whitelisted, we add ::1 to ensure -v works for ipv6
      grepcidr -v -e "$whitelist ::1" 2>/dev/null | \
      # Only leave ip's that aren't already limited
      grepcidr -v -e "$(cut -d" " -f2 "${BANS_BW_IP_LIST}") 127.0.0.1 ::1" \
      > "$BAD_IP_LIST"

    while read line; do
      if [ "$line" = "" ]; then
        continue
      fi
```

```
        ip_drop_rate=1

        ip=$(echo "$line" | cut -d" " -f1)
        bandwidth=$(echo "$line" | cut -d" " -f2)

        prio_tc=$((prio_tc+1))

        drop_rate_ip "$ip" "$prio_tc"

        echo "$ip using $bandwidth/s of bandwidth" >> "$BANNED_IP_MAIL"

        current_time=$(date +"%s")
        echo "$((current_time+BANDWIDTH_DROP_PERIOD)) $ip $bandwidth
$prio_tc" >> \
            "${BANS_BW_IP_LIST}"

        if [ "$prio_tc" -gt 10000 ]; then
            prio_tc=0
        fi

        log_msg "drop transfer rate for $ip wich used ${bandwidth}/s for ban period
$BANDWIDTH_DROP_PERIOD"
    done < "$BAD_IP_LIST"

    if [ "$ip_drop_rate" -eq 1 ]; then
        if [ -n "$EMAIL_TO" ]; then
            dt=$(date)
            hn=$(hostname)
            cat "$BANNED_IP_MAIL" | mail -s "[$hn] IP addresses transfer rate limit
on $dt" $EMAIL_TO
        fi
    fi

    rm -f "$TMP_PREFIX".*
}

# Drops the transfer rate for a given ip.
# param1 The ip address ratelimit.
# param2 The tc priority
drop_rate_ip()
{
    ifindex=0
    for interface in $BANDWIDTH_INTERFACES; do
        # egress control
        $TC filter add dev "$interface" protocol ip parent 1:0 \
            prio "$2" u32 match ip dst "$1"/32 flowid 1:1
```

```
        $TC filter add dev "$interface" protocol ip parent 1:0 \
           prio "$2" u32 match ip src "$1"/32 flowid 1:2

        # ingress control
        $TC filter add dev "ifb${ifindex}" protocol ip parent 1:0 \
           prio "$2" u32 match ip dst "$1"/32 flowid 1:1
        $TC filter add dev "ifb${ifindex}" protocol ip parent 1:0 \
           prio "$2" u32 match ip src "$1"/32 flowid 1:2

        ifindex=$((ifindex+1))
    done
}

# Removes rate limit on ip.
# param1 The ip address
# param2 Optional amount of bandwidth the ip used.
# param3 The assigned tc rule # for the ip.
undrop_rate_ip()
{
    if [ "$1" = "" ] || [ "$3" = "" ]; then
        return 1
    fi

    ifindex=0
    for interface in $BANDWIDTH_INTERFACES; do
        tc filter del dev "$interface" prio "$3" 2>/dev/null
        tc filter del dev "ifb${ifindex}" prio "$3" 2>/dev/null
        ifindex=$((ifindex+1))
    done

    if [ "$2" != "" ]; then
        log_msg "undrop transfer rate for $1 that used $2/s of bandwidth"
    else
        log_msg "undrop transfer rate for $1"
    fi

    grep -v "$1" "${BANS_BW_IP_LIST}" > "${BANS_BW_IP_LIST}.tmp"
    rm "${BANS_BW_IP_LIST}"
    mv "${BANS_BW_IP_LIST}.tmp" "${BANS_BW_IP_LIST}"

    return 0
}

# Remove rate rule on ip's after the amount of time given on
# BANDWIDTH_DROP_PERIOD
undrop_rate_list()
```

```
{
   if ! $BANDWIDTH_CONTROL; then
      return
   fi

   current_time=$(date +"%s")

   while read line; do
      if [ "$line" = "" ]; then
         continue
      fi

      ban_time=$(echo "$line" | cut -d" " -f1)
      ip=$(echo "$line" | cut -d" " -f2)
      bandwidth=$(echo "$line" | cut -d" " -f3)
      tcrule=$(echo "$line" | cut -d" " -f4)

      if [ "$current_time" -gt "$ban_time" ]; then
         undrop_rate_ip "$ip" "$bandwidth" "$tcrule"
      fi
   done < "$BANS_BW_IP_LIST"
}

ip_to_hex()
{
   printf '%02x' "$(echo "$1" | sed "s/\\./ /g")"
}

# Active connections to server.
view_connections()
{
   ip6_show=false
   ip4_show=false

   if [ "$1" = "6" ]; then
      ip6_show=true
   elif [ "$1" = "4" ]; then
      ip4_show=true
   else
      ip6_show=true
      ip4_show=true
   fi

   whitelist=$(ignore_list "1")

   # Find all ipv4 connections
```

```
    if $ip4_show; then
        get_connections "4" | \
            # Extract only the fifth column
            awk '{print $6}' | \
            # Strip port
            cut -d":" -f1 | \
            # Sort addresses for uniq to work correctly
            sort | \
            # Only leave non whitelisted
            grepcidr -v -e "$SERVER_IP_LIST $whitelist" 2>/dev/null | \
            # Group same occurrences of ip and prepend amount of occurences found
            uniq -c | \
            # Numerical sort in reverse order
            sort -nr
    fi

    # Find all ipv6 connections
    if $ip6_show; then
        get_connections "6" | \
            # Extract only the fifth column
            awk '{print $6}' | \
            # Strip port and leading [
            sed -E "s/]:[0-9]+//g" | sed "s/\\[//g" | \
            # Sort addresses for uniq to work correctly
            sort | \
            # Only leave non whitelisted, we add ::1 to ensure -v works
            grepcidr -v -e "$SERVER_IP_LIST $whitelist ::1" 2>/dev/null | \
            # Group same occurrences of ip and prepend amount of occurences found
            uniq -c | \
            # Numerical sort in reverse order
            sort -nr
    fi

    if $ENABLE_CLOUDFLARE; then
        echo
        echo "List of CloudFlare ip's."
        echo "================================="
        ban_cloudflare
    fi
}

# Active connections to server including port.
view_connections_port()
{
    ip6_show=false
    ip4_show=false
```

```
    if [ "$1" = "6" ]; then
        ip6_show=true
    elif [ "$1" = "4" ]; then
        ip4_show=true
    else
        ip6_show=true
        ip4_show=true
    fi

    whitelist=$(ignore_list "1")

    # Find all ipv4 connections
    if $ip4_show; then
        get_connections "4" | \
            # Extract only the fifth column
            awk '{print $6}' | \
            # Sort addresses for uniq to work correctly
            sort | \
            # Only leave non whitelisted
            grepcidr -v -e "$SERVER_IP_LIST $whitelist" 2>/dev/null | \
            # Group same occurrences of ip and prepend amount of occurences found
            uniq -c | \
            # Numerical sort in reverse order
            sort -nr
    fi

    # Find all ipv6 connections
    if $ip6_show; then
        get_connections "6" | \
            # Extract only the fifth column
            awk '{print $6}' | \
            # Strip leading [ and ending ]
            sed -E "s/(\\[|\\])//g" | \
            # Sort addresses for uniq to work correctly
            sort | \
            # Only leave non whitelisted, we add ::1 to ensure -v works
            grepcidr -v -e "$SERVER_IP_LIST $whitelist ::1" 2>/dev/null | \
            # Group same occurrences of ip and prepend amount of occurences found
            uniq -c | \
            # Numerical sort in reverse order
            sort -nr
    fi
}

view_bans()
```

```
{
   echo "List of currently banned ip's."
   echo "================================="

   if [ -e "$BANS_IP_LIST" ]; then
      printf "% -5s %s\\n" "Exp." "IP"
      echo '---------------------------------'
      while read line; do
         time=$(echo "$line" | cut -d" " -f1)
         ip=$(echo "$line" | cut -d" " -f2)
         conns=$(echo "$line" | cut -d" " -f3)
         port=$(echo "$line" | cut -d" " -f4)

         current_time=$(date +"%s")
         hours=$(echo $(((time-current_time)/60/60)))
         minutes=$(echo $(((time-current_time)/60%60)))

         echo "$(printf "%02d" "$hours"):$(printf "%02d" "$minutes") $ip $port
$conns"
      done < "$BANS_IP_LIST"
   fi

   if $ENABLE_CLOUDFLARE; then
      echo
      echo "List of banned CloudFlare ip's."
      echo "================================="
      if [ -e "$BANS_CLOUDFLARE_IP_LIST" ]; then
         printf "% -5s %s\\n" "Exp." "IP"
         echo '---------------------------------'
         while read line; do
            time=$(echo "$line" | cut -d" " -f1)
            ip=$(echo "$line" | cut -d" " -f2)
            conns=$(echo "$line" | cut -d" " -f3)

            current_time=$(date +"%s")
            hours=$(echo $(((time-current_time)/60/60)))
            minutes=$(echo $(((time-current_time)/60%60)))

            echo "$(printf "%02d" "$hours"):$(printf "%02d" "$minutes") $ip
$conns"
         done < "$BANS_CLOUDFLARE_IP_LIST"
      fi
   fi
}

view_drops()
```

```
{
   echo "List of current traffic controls."
   echo
"=============================================================
=="

   if [ -e "$BANS_BW_IP_LIST" ]; then
      printf "% -5s % -30s % -7s\\n" "Exp." "IP" "Prio."
      echo '-------------------------------------------------------------'
      while read line; do
         time=$(echo "$line" | cut -d" " -f1)
         ip=$(echo "$line" | cut -d" " -f2)
         bw=$(echo "$line" | cut -d" " -f3)
         prio=$(echo "$line" | cut -d" " -f4)

         current_time=$(date +"%s")
         hours=$(echo $(((time-current_time)/60/60)))
         minutes=$(echo $(((time-current_time)/60%60)))

         echo "$(printf "%02d" "$hours"):$(printf "%02d" "$minutes")" \
            "$(printf "% -30s" $ip)" \
            "$(printf "% -7s" $prio)" \
            "$bw"
      done < "$BANS_BW_IP_LIST"
   fi
}

# Executed as a cleanup function when the daemon is stopped
on_daemon_exit()
{
   stop_bandwidth_control

   pkill -9 tcpdump

   if [ -e /var/run/ddos.pid ]; then
      rm -f /var/run/ddos.pid
   fi

   exit 0
}

# Return the current process id of the daemon or 0 if not running
daemon_pid()
{
   if [ -e "/var/run/ddos.pid" ]; then
      echo $(cat /var/run/ddos.pid)
```

```
        return
    fi

    echo "0"
}

# Check if daemon is running.
# Outputs 1 if running 0 if not.
daemon_running()
{
    if [ -e /var/run/ddos.pid ]; then
        running_pid=$(pgrep ddos)

        if [ "$running_pid" != "" ]; then
            current_pid=$(daemon_pid)

            for pid_num in $running_pid; do
                if [ "$current_pid" = "$pid_num" ]; then
                    echo "1"
                    return
                fi
            done
        fi
    fi

    echo "0"
}

start_daemon()
{
    su_required

    if [ "$(daemon_running)" = "1" ]; then
        echo "ddos daemon is already running..."
        exit 0
    fi

    echo "starting ddos daemon..."

    if [ ! -e "$BANS_IP_LIST" ]; then
        touch "${BANS_IP_LIST}"
    fi

    if [ ! -e "$BANS_BW_IP_LIST" ]; then
        touch "${BANS_BW_IP_LIST}"
```

```
    fi

    if [ ! -e "$BANS_CLOUDFLARE_IP_LIST" ]; then
        touch "${BANS_CLOUDFLARE_IP_LIST}"
    fi

    nohup "$0" -l > /dev/null 2>&1 &

    log_msg "daemon started"
}

stop_daemon()
{
    su_required

    if [ "$(daemon_running)" = "0" ]; then
        echo "ddos daemon is not running..."
        exit 0
    fi

    echo "stopping ddos daemon..."

    kill "$(daemon_pid)"

    while [ -e "/var/run/ddos.pid" ]; do
        continue
    done

    log_msg "daemon stopped"
}

daemon_loop()
{
    su_required

    if [ "$(daemon_running)" = "1" ]; then
        exit 0
    fi

    echo "$$" > "/var/run/ddos.pid"

    trap 'on_daemon_exit' INT
    trap 'on_daemon_exit' QUIT
    trap 'on_daemon_exit' TERM
    trap 'on_daemon_exit' EXIT
```

```
    echo "Detecting firewall..."
    detect_firewall

    start_bandwidth_control

    # Start tcpdump sniffing.
    if $ENABLE_CLOUDFLARE; then
        # kill any previous opened tcpdump session.
        pkill -9 tcpdump
        tcpdump -w "$CLOUDFLARE_PCAP" -G
"$((DAEMON_FREQ+DAEMON_FREQ))" &
    fi

    if $ENABLE_PORTS; then
        echo "Ban by port rules selected. Port rules: [$PORT_CONNECTIONS]"
    elif $ONLY_INCOMING; then
        echo "Ban only incoming connections that exceed
$NO_OF_CONNECTIONS"
    else
        echo "Ban in/out connections that combined exceed
$NO_OF_CONNECTIONS"
    fi

    # run unban and undrop lists after 10 seconds of initialization
    ban_check_timer=$(date +"%s")
    ban_check_timer=$((ban_check_timer+10))

    echo "Monitoring connections!"
    while true; do
        check_connections
        check_connections_bw

        if $ENABLE_CLOUDFLARE; then
            check_connections_cloudflare
        fi

        # unban or undrop expired ip's every 1 minute
        current_loop_time=$(date +"%s")
        if [ "$current_loop_time" -gt "$ban_check_timer" ]; then
            unban_ip_list
            undrop_rate_list

            if $ENABLE_CLOUDFLARE; then
                unban_ip_list_cloudflare
            fi
```

```
            ban_check_timer=$(date +"%s")
            ban_check_timer=$((ban_check_timer+60))
        fi

        sleep "$DAEMON_FREQ"
    done
}

daemon_status()
{
    current_pid=$(daemon_pid)

    if [ "$(daemon_running)" = "1" ]; then
        echo "ddos status: running with pid $current_pid"
    else
        echo "ddos status: not running"
    fi
}

detect_firewall()
{
    if [ "$FIREWALL" = "auto" ] || [ "$FIREWALL" = "" ]; then
        apf_where=$(whereis apf);
        csf_where=$(whereis csf);
        ipf_where=$(whereis ipfw);
        ipt_where=$(whereis iptables);

        if [ -e "$APF" ]; then
            FIREWALL="apf"
        elif [ -e "$CSF" ]; then
            FIREWALL="csf"
        elif [ -e "$IPF" ]; then
            FIREWALL="ipfw"
        elif [ -e "$IPT" ]; then
            FIREWALL="iptables"
        elif [ "$apf_where" != "apf:" ]; then
            FIREWALL="apf"
            APF="apf"
        elif [ "$csf_where" != "csf:" ]; then
            FIREWALL="csf"
            CSF="csf"
        elif [ "$ipf_where" != "ipfw:" ]; then
            FIREWALL="ipfw"
            IPF="ipfw"
        elif [ "$ipt_where" != "iptables:" ]; then
            FIREWALL="iptables"
```

```
            IPT="iptables"
        else
            echo "error: No valid firewall found."
            log_msg "error: no valid firewall found"
            exit 1
        fi
    fi
}

# Detects the available network interfaces with internet connection and
# stores them on BANDWIDTH_INTERFACES.
detect_interfaces()
{
    BANDWIDTH_INTERFACES=""

    interfaces=$(ifconfig | \
        grep -E "^[0-9a-zA-Z]+:" | \
        awk '{gsub(/(^lo)?:$/, "", $1); print $1}' | \
        xargs
    )

    for interface in $interfaces; do
        if ping -I "$interface" -c1 8.8.8.8>/dev/null; then
            BANDWIDTH_INTERFACES="$BANDWIDTH_INTERFACES
$interface"
        fi
    done
}

start_bandwidth_control()
{
    if $BANDWIDTH_CONTROL; then
        echo "Initializing bandwidth control..."

        echo "Detecting network interfaces..."
        detect_interfaces

        # Load right amount of needed virtual interfaces
        if_count="$(echo "$BANDWIDTH_INTERFACES" | wc -w)"
        rmmod ifb 2>/dev/null
        modprobe ifb numifbs="$if_count"

        # Activate all ifb network interfaces.
        for index in $(seq 0 $((if_count-1)) | xargs); do
            ifconfig "ifb${index}" up
        done
```

```
        #percent_25="$(echo $((.25*BANDWIDTH_DROP_RATE)) | sed
"s/\\.//g")"
        #percent_80="$(echo $((.80*BANDWIDTH_DROP_RATE)) | sed
"s/\\.//g")"

        ifindex=0
        for interface in $BANDWIDTH_INTERFACES; do
            # Redirect ingress to ifb# in order for traffic shaping to
            # properly work.
            $TC qdisc add dev "$interface" handle ffff: ingress
            $TC filter add dev "$interface" parent ffff: \
                protocol ip u32 match u32 0 0 action mirred \
                egress redirect dev "ifb${ifindex}"

            # ingress rules
            $TC qdisc add dev "ifb${ifindex}" root \
                handle 1: htb default 30
            $TC class add dev "ifb${ifindex}" parent 1: \
                classid 1:1 htb rate "$BANDWIDTH_DROP_RATE"
            $TC class add dev "ifb${ifindex}" parent 1: \
                classid 1:2 htb rate "$BANDWIDTH_DROP_RATE"
            $TC qdisc add dev "ifb${ifindex}" parent 1:1 fq_codel ecn
            $TC qdisc add dev "ifb${ifindex}" parent 1:2 fq_codel ecn

            # egress rules
            $TC qdisc add dev "$interface" root \
                handle 1: htb default 30
            $TC class add dev "$interface" parent 1: \
                classid 1:1 htb rate "$BANDWIDTH_DROP_RATE"
            $TC class add dev "$interface" parent 1: \
                classid 1:2 htb rate "$BANDWIDTH_DROP_RATE"
            $TC qdisc add dev "$interface" parent 1:1 fq_codel noecn
            $TC qdisc add dev "$interface" parent 1:2 fq_codel noecn

            ifindex=$((ifindex+1))
        done

        # we force DAEMON_FREQ to zero because iftop takes 3 seconds
        # to calculate correct bandwidth usage.
        DAEMON_FREQ=0
    fi
}

stop_bandwidth_control()
{
```

```
    if $BANDWIDTH_CONTROL; then
        ifcount=0
        for interface in $BANDWIDTH_INTERFACES; do
            echo "Disabling bandwidth control on: $interface..."

            $TC qdisc del dev "$interface" root 2>/dev/null
            $TC qdisc del dev "$interface" ingress 2>/dev/null

            $TC qdisc del dev "ifb${ifcount}" root 2>/dev/null
            $TC qdisc del dev "ifb${ifcount}" ingress 2>/dev/null

            ifcount=$((ifcount+1))
        done
    fi

    rmmod ifb 2>/dev/null
}

view_ports()
{
    printf "Port blocking is: "

    if $ENABLE_PORTS; then
        printf "enabled\\n"
    else
        printf "disabled\\n"
    fi

    printf -- '-%.0s' $(seq 48); echo ""
    printf "% -15s % -15s % -15s\\n" "Port" "Max-Conn" "Ban-Time"
    printf -- '-%.0s' $(seq 48); echo ""
    for port in $(echo "$PORT_CONNECTIONS" | xargs); do
        number=$(echo "$port" | cut -d":" -f1)
        max_conn=$(echo "$port" | cut -d":" -f2)
        ban_time=$(echo "$port" | cut -d":" -f3)
        printf "% -15s % -15s % -15s\\n" "$number" "$max_conn" "$ban_time"
    done
}

# Set Default settings
PROGDIR="/usr/local/ddos"
SBINDIR="/usr/local/sbin"
PROG="$PROGDIR/ddos.sh"
IGNORE_IP_LIST="ignore.ip.list"
IGNORE_HOST_LIST="ignore.host.list"
CRON="/etc/cron.d/ddos"
```

```
APF="/usr/sbin/apf"
CSF="/usr/sbin/csf"
IPF="/sbin/ipfw"
IPT="/sbin/iptables"
IPT6="/sbin/ip6tables"
TC="/sbin/tc"
FREQ=1
DAEMON_FREQ=5
NO_OF_CONNECTIONS=150
ENABLE_PORTS=false
PORT_CONNECTIONS="80:150:600 443:150:600"
FIREWALL="auto"
EMAIL_TO="root"
BAN_PERIOD=600
CONN_STATES="connected"
CONN_STATES_NS="ESTABLISHED|SYN_SENT|SYN_RECV|FIN_WAIT1|
FIN_WAIT2|TIME_WAIT|CLOSE_WAIT|LAST_ACK|CLOSING"
ONLY_INCOMING=false
ENABLE_CLOUDFLARE=false
BANDWIDTH_CONTROL=false
BANDWIDTH_CONTROL_LIMIT="1896kbit"
BANDWIDTH_DROP_RATE="512kbit"
BANDWIDTH_DROP_PERIOD=600
BANDWIDTH_ONLY_INCOMING=true

# Load custom settings
load_conf

# Overwrite old configuration values
if echo "$CONN_STATES" | grep "|">/dev/null; then
   CONN_STATES="connected"
fi

KILL=0

while [ "$1" ]; do
   case $1 in
      '-h' | '--help' | '?' )
         showhelp
         exit
         ;;
      '--cron' | '-c' )
         add_to_cron
         exit
         ;;
      '--ignore-list' | '-i' )
```

```
    echo "List of currently whitelisted ip's."
    echo "==================================="
    ignore_list
    exit
    ;;
'--bans-list' | '-b' )
    view_bans
    exit
    ;;
'--traffic-list' | '-f' )
    view_drops
    exit
    ;;
'--unban' | '-u' )
    su_required
    shift
    detect_firewall

    if ! unban_ip "$1"; then
        echo "Please specify a valid ip address."
    elif $ENABLE_CLOUDFLARE; then
        unban_ip_cloudflare "$1"
    fi
    exit
    ;;
'--start' | '-d' )
    start_daemon
    exit
    ;;
'--stop' | '-s' )
    stop_daemon
    exit
    ;;
'--status' | '-t' )
    daemon_status
    exit
    ;;
'--loop' | '-l' )
    # start daemon loop, used internally by --start | -s
    daemon_loop
    exit
    ;;
'--view' | '-v' )
    shift
    view_connections "$1"
    exit
```

```
            ;;
        '--view-port' | '-y' )
            shift
            view_connections_port "$1"
            exit
            ;;
        '-v6' | '-6v' )
            shift
            view_connections 6
            exit
            ;;
        '-y6' | '-6y' )
            shift
            view_connections_port 6
            exit
            ;;
        '-v4' | '-4v' )
            shift
            view_connections 4
            exit
            ;;
        '-y4' | '-4y' )
            shift
            view_connections_port 4
            exit
            ;;
        '--ports' | '-p' )
            view_ports
            exit
            ;;
        '--kill' | '-k' )
            su_required
            KILL=1
            ;;
        *[0-9]* )
            NO_OF_CONNECTIONS=$1
            ;;
        * )
            showhelp
            exit
            ;;
    esac

    shift
done
```

```
if [ $KILL -eq 1 ]; then
   detect_firewall
   check_connections

   if $ENABLE_CLOUDFLARE; then
      check_connections_cloudflare
   fi
else
   showhelp
fi

exit 0
```

Script Konfigurasi Ddos Deflate

```
# Paths of the script and other files
PROGDIR="/usr/local/ddos"
SBINDIR="/usr/local/sbin"
PROG="$PROGDIR/ddos.sh"
IGNORE_IP_LIST="ignore.ip.list"
IGNORE_HOST_LIST="ignore.host.list"
CRON="/etc/cron.d/ddos"
# Make sure your APF version is atleast 0.96
APF="/usr/sbin/apf"
CSF="/usr/sbin/csf"
IPF="/sbin/ipfw"
IPT="/sbin/iptables"
IPT6="/sbin/ip6tables"
TC="/sbin/tc"
FREQ=1
DAEMON_FREQ=5
NO_OF_CONNECTIONS=10
ONLY_INCOMING=false
ENABLE_CLOUDFLARE=false
ONLY_INCOMING
ENABLE_PORTS=false
PORT_CONNECTIONS="80:150:600 443:150:600 20-21:150:600"
FIREWALL="apf"
EMAIL_TO="root"
BAN_PERIOD=600
CONN_STATES="connected"
CONN_STATES_NS="ESTABLISHED|SYN_SENT|SYN_RECV|FIN_WAIT1|
FIN_WAIT2|TIME_WAIT|CLOSE_WAIT|LAST_ACK|CLOSING"
```

```
BANDWIDTH_CONTROL=false
BANDWIDTH_CONTROL_LIMIT="1896kbit"
BANDWIDTH_CONTROL_LIMIT
BANDWIDTH_DROP_RATE="512kbit"
BANDWIDTH_DROP_PERIOD=600
BANDWIDTH_ONLY_INCOMING=true
```

Script Sound Alert

```
#!/bin/sh
BANS_IP_LIST="/var/lib/ddos/bans.list"
while read line; do
        echo $line && mpv ddos_beep.mp3
done < $BANS_IP_LIST
exit
```

Script Srontab

```
*/1 * * * * cd /usr/local/ddos && sh ./sound_alert.sh
```

Lampiran : 2 Riwayat Hidup Peneliti

RIWAYAT HIDUP PENELITI

Nama	: Zulkifli Ibrahim
Tempat Tanggal Lahir	: Gorontalo, 26 November 1997
Pekerjaan	: Mahasiswa
Email	: zulkifliibrahimc@gmail.com

Riwayat Pendidikan Yang Pernah Ditempuh:

1. SDN 2 Bulotalangi, Kec. Tapa. Kab. Bone Bolango, Provinsi Gorontalo, tahun 2004 sampai 2009.
2. SMP Negeri 5 Tapa, Kab.Bone Bolango, Provinsi Gorontalo, tahun 2010 sampai 2012.
3. SMK Negeri 4 Kota Gorontalo, Provinsi Gorontalo, Program Studi Teknik Komputer Dan Jaringan, tahun 2013 sampai 2015.
4. Tahun 2015 Lanjut-S1 di Universitas Ichsan Gorontalo, Fakultas Ilmu Komputer, Jurusan Teknik Informatika.

Riwayat Pekerjaan:

A. Pernah bekerja di Kantor Urusan Agama (KUA) sebagai *maintenance hardware* dan *network*

Kegiatan Yang Pernah Diikuti:

A. Pernah Mengikuti Fasilitas Bimbingan Teknis dan Sertifikasi Standar Kompetensi Kerja Nasional Indonesia tahun 2015

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS ICHSAN
(UNISAN) GORONTALO
SURAT KEPUTUSAN MENDIKNAS RI NOMOR 84/D/O/2001
Jl. Achmad Nadjamuddin No. 17 Telp (0435) 829975 Fax (0435) 829976 Gorontalo

SURAT REKOMENDASI BEBAS PLAGIASI

No. 0744/UNISAN-G/S-BP/XII/2020

Yang bertanda tangan di bawah ini:

Nama	:	Sunarto Taliki, M.Kom
NIDN	:	0906058301
Unit Kerja	:	Pustikom, Universitas Ichsan Gorontalo

Dengan ini Menyatakan bahwa :

Nama Mahasisw	:	ZULKIFLI IBRAHIM
NIM	:	T3115111
Program Studi	:	Teknik Informatika (S1)
Fakultas	:	Fakultas Ilmu Komputer
Judul Skripsi	:	Mendeteksi Serangan DDOS (Distributed Denial Of Service) Menggunakan DDOS Deflate

Sesuai dengan hasil pengecekan tingkat kemiripan skripsi melalui aplikasi Turnitin untuk judul skripsi di atas diperoleh hasil Similarity sebesar 13%, berdasarkan SK Rektor No. 237/UNISAN-G/SK/IX/2019 tentang Panduan Pencegahan dan Penanggulangan Plagiarisme, bahwa batas kemiripan skripsi maksimal 35% dan sesuai dengan Surat Pernyataan dari kedua Pembimbing yang bersangkutan menyatakan bahwa isi softcopy skripsi yang diolah di Turnitin SAMA ISINYA dengan Skripsi Aslinya serta format penulisannya sudah sesuai dengan Buku Panduan Penulisan Skripsi, untuk itu skripsi tersebut di atas dinyatakan BEBAS PLAGIASI dan layak untuk diujiankan.

Demikian surat rekomendasi ini dibuat untuk digunakan sebagaimana mestinya.

Gorontalo, 09 Desember 2020
Tim Verifikasi,

Sunarto Taliki, M.Kom
NIDN. 0906058301

Tembusan :
1. Dekan
2. Ketua Program Studi
3. Pembimbing I dan Pembimbing II
4. Yang bersangkutan
5. Arsip

turnitin

SKRIPSI_1_T3115111_ZULKIFLI IBRAHIM.docx
Dec 8, 2020
6571 words / 40650 characters

T3115111 ZULKIFLI IBRAHIM

MENDETEKSI SERANGAN DDOS (DISTRIBUTED DENIAL OF SER...

Sources Overview

13%

OVERALL SIMILARITY

1	achart403.blogspot.com INTERNET	3%
2	www.tagar.id INTERNET	3%
3	es.scribd.com INTERNET	2%
4	id.m.wikipedia.org INTERNET	2%
5	id.scribd.com INTERNET	<1%
6	id.123dok.com INTERNET	<1%
7	www.scribd.com INTERNET	<1%
8	www.defectink.com INTERNET	<1%
9	www.neliti.com INTERNET	<1%
10	beritaradar.com INTERNET	<1%
11	docplayer.net INTERNET	<1%
12	nothinux.id INTERNET	<1%

Excluded search repositories:

- Submitted Works

Excluded from Similarity Report:

- Small Matches (less than 25 words).

Excluded sources:

- None