PENGEMBANGAN PENCEGAHAN SERANGAN DISTRIBUTED DENIAL OF *SERVICE* (DDOS) PADA SUMBER DAYA JARINGAN DENGAN INTEGRASI *NETWORK BEHAVIOR ANALYSIS* DAN *CLIENT PUZZLE*

Septian Geges[1)], Waskitho Wibisono[2)]
[1, 2)]Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Jalan Raya ITS, Surabaya, Jawa Timur 60111, Indonesia
e-mail: septian.geges@gmail.com[1)], waskitho@cs.its.ac.id[2)]

ABSTRAK

Denial of Service (DoS) merupakan permasalahan keamanan jaringan yang sampai saat ini terus berkembang secara dinamis. Semakin tinggi kemampuan komputasi suatu komputer penyerang, serangan DoS yang dapat dihasilkan juga semakin membahayakan. Serangan ini dapat mengakibatkan ketidakmampuan server untuk melayani service request yang sah. Karena itu serangan DoS sangat merugikan dan perlu diberikan pencegahan yang efektif. Ancaman berikutnya yang juga sangat membahayakan adalah Distributed Denial of Service (DDoS), dimana serangan ini memanfaatkan sejumlah besar komputer untuk menjalankan serangan DoS kepada server, web service, atau sumber daya jaringan lain. Mengingat resiko besar yang diakibatkan serangan DDoS ini, banyak peneliti yang terdorong untuk merancang mekanisme pengamanan sumber daya jaringan.

Pada penelitian ini, penulis mengkhususkan pokok permasalahan pada pengamanan web service. Penulis mengemukakan sebuah mekanisme untuk mengamankan web service dengan cara melakukan filtrasi dan validasi permintaan yang diterima untuk mengakses sumber daya jaringan. Filtrasi dan validasi ini dilakukan dengan gabungan metode Network Behavior Analysis (NBA) dan Client Puzzle (CP). Metode NBA menjadi lapisan pertahanan pertama untuk mendeteksi apakah sedang terjadi serangan DDoS dengan mengukur tingkat kepadatan jaringan/Network density. Dari metode NBA, didapatkan IP Address yang perlu divalidasi dengan metode CP sebagai lapisan pertahanan kedua. Apabila suatu service request sudah berhasil melewati proses filtrasi dan validasi ini, maka service request ini baru akan dilayani.

Dari hasil percobaan, terbukti metode ini dapat mendeteksi serangan DDoS sekaligus menjamin bahwa service request yang sah mendapat pelayanan yang seharusnya sehingga server dapat melayani service request dengan baik.

***Kata Kunci**: DDoS, Client Puzzle, Network Behavior Analysis, Network density.*

ABSTRACT

Denial of Service (DoS) still a network security problems that continues to evolve dynamically. The higher the computing capabilities of a attackerhost, DoS attacks that can be produced is also more powerful and dangerous. These attacks can result in the inability of server to serve a legitimate service requests. Therefore, DoS attacks can be very harmful andeffective prevention should be given. The next network security threat that also very dangerous is Distributed Denial of Service (DDoS), in which the attacker takes advantage of a large number of computers to run a DoS attack againts a server, web service, or other network resources. Because of the huge risks caused by DDoS attacks, these have encouraged many researchers to design a mechanism for securing network resources.

In this research, the authors focus on the main issue that related to web service security. The authors propose a mechanism for securing web services by means of filtration and validation of the service requests aim for accessing the network resources. Filtration and validation are performed by the combined method of Network Behavior Analysis (NBA) and Client Puzzle (CP). NBA become the first layer of defense method which is going to detect whether the DDoS attack by measuring the level of network congestion / Network density. NBA method obtained the IP addresses that need to be validated by the CP method as a second layer of defense. When a service request has been made through the process of filtration and this validation, then the demand for this new service will be served.

From the experimental results, it is proved that this method can detect DDoS attacks while ensuring that legitimate service requests received appropriate services so that the server can serve requests with good service.

***Keywords**: DDoS, Client Puzzle, Network Behavior Analysis, Network density.*

I. PENDAHULUAN

TEKNOLOGI internet saat ini berkembang dengan pesat, begitu pula dengan jumlah penggunanya yang semakin banyak. Internet tidak lagi hanya digunakan sebagai sarana bertukar informasi, namun mulai digunakan untuk keperluan komersial, misalnya saja sebagai sarana transaksi pembayaran. Hal ini tentu

menyebabkan sejumlah besar data berharga semakin banyak beredar melalui jaringan internet. Namun, dari waktu ke waktu semakin banyak celah keamanan internet yang ditemukan dan disalahgunakan oleh para penjahat elektronik. Lebih spesifik lagi, motif yang melatarbelakangi penyalahgunaan internet belakangan ini sudah berbeda dengan motif tradisional (untuk menyerang *server* atau perangkat lain dalam jaringan), serangan yang dilakukan saat ini dimaksudkan untuk memperoleh keuntungan finansial [6]. Hal ini tentu menjadi ancaman baru yang membahayakan jutaan orang yang menggunakan internet dalam beraktivitas. Contoh serangan yang dapat dilakukan melalui internet antara lain, pencurian informasi pribadi oleh para penjahat elektronik yang dapat menyebabkan kerugian keuangan yang signifikan, internet digunakan untuk mengirim *spam mail*, hingga sebagai sarana meluncurkan serangan *Denial of Service* (DoS) dan *Distributed Denial of Service* (DDoS).

Sampai saat ini, serangan DoS dan DDoS masih belum memiliki metode pencegahan yang dapat diterapkan pada semua jenis DoS dan DDoS. Hal ini disebabkan karena manajemen dan serangan DoS/DDoS memiliki mekanisme yang bervariasi, para *hacker* juga terus mengembangkan metode serangan yang sudah ada, bahkan menggunakan metode baru untuk melakukan penyerangan. Saat ini, terdapat beberapa pendekatan untuk memerangi serangan DoS/DDoS. Perlindungan terhadap serangan DoS/DDoS yang dapat dilakukan dari sisi *server* adalah dengan menerapkan protokol yang mengatur penggunaan sumber daya *server* dengan tujuan untuk mengurangi eksploitasi sumber daya yang dimiliki *server*.

Dalam penelitian ini, penulis mengemukakan sebuah rancangan protokol untuk melakukan verifikasi *service request* kepada *web service*. Protokol ini memanfaatkan karakteristik utama serangan DDoS (*Network Behavior*) dan mengkombinasikannya dengan metode *Client Puzzle*. Proses verifikasi ini dapat dilakukan diluar *server* sehingga tidak mengurangi kinerja *server* untuk menyediakan layanan.

II. KAJIAN PUSTAKA

Serangan DoS biasanya melibatkan penyerang mengirimkan pesan untuk mengeksploitasi kerentanan tertentu yang mengarah kepada ketidakstabilan atau kelumpuhan sistem bisnis [7-8][16]. Penyerang juga dapat melakukan DoS dengan mengirim sejumlah besar pesan normal dengan cepat ke *node* tunggal, tujuannya adalah untuk menghabiskan sumber daya sistem sehingga menyebabkan kegagalan sistem bisnis. Serangan DDoS adalah serangan DoS yang memanfaatkan beberapa sumber daya serangan yang terdistribusi. Biasanya, para penyerang menggunakan sejumlah besar *bots* yang dikendalikan (komputer inang/*daemon*, juga disebut sebagai *zombie*) dan terdistribusi di beberapa lokasi yang berbeda untuk melancarkankan sejumlah besar serangan DoS terhadap target tunggal atau beberapa target. Seiring dengan perkembangan pesat dari *botnet* (jaringan *bot*) dalam beberapa tahun terakhir, skala lalu lintas serangan yang disebabkan oleh serangan DDoS telah meningkat, dengan targetnya tidak hanya *server* untuk keperluan bisnis, tetapi juga infrastruktur internet seperti *firewall*, *router* dan sistem DNS serta *bandwidth* jaringan [10].

A. Karakteristik DDoS

Bentuk serangan terdistribusi DDoS "banyak ke satu" yang membuat serangan ini lebih sulit untuk dicegah. Sebuah serangan DDoS terdiri dari empat elemen, seperti yang ditunjukkan pada Gambar 1. 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 [10]:

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 [11].

B. Metode Serangan DDoS

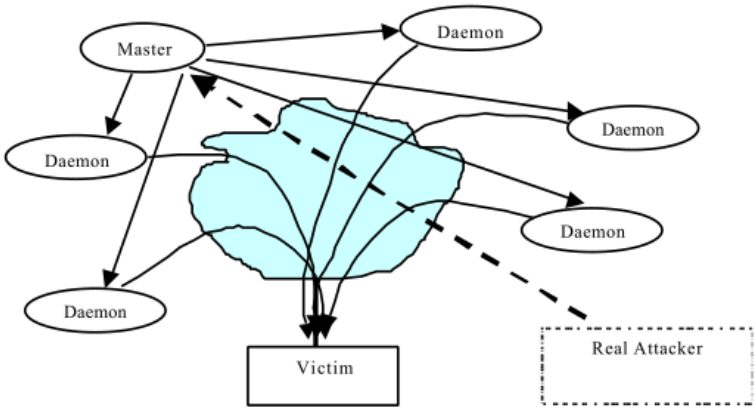
Secara umum, paket data yang beredar di jaringan menggunakan protokol TCP / IP untuk transmisinya. Paket ini sendiri tidak berbahaya, tetapi jika ada terlalu banyak paket yang abnormal, maka perangkat jaringan atau *server* akan mengalami kelebihan beban/*overload*. Kondisi ini dapat dengan cepat mengkonsumsi 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 data normal. 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:

- Serangan dengan basis *bandwidth*
  Serangan 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.
- 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.
- 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.

C. Strategi menghadapi DoS dan DDoS yang telah dilakukan

Berikut ini adalah mekanisme pertahanan terhadap DDoS yang telah diusulkan oleh para peneliti, dan dikategorikan berdasarkan dimana mekanisme pertahanan ini dapat diterapkan.



Gambar. 1. Ilustrasi serangan DDoS

1. Pertahanan dari Sisi Jaringan Global
   Pertahanan dari sisi jaringan global bertujuan untuk mencegah *host* dalam jaringan dijadikan *bot* untuk melancarkan serangan DoS/DDoS. Contoh mekanisme pertahanan yang menggunakan pendekatan ini adalah D-WARD [13] dan BotGAD [3-5] D-WARD memantau lalu lintas dua arah antara alamat internal dan alamat dari internet. Statistik lalu lintas aktif disimpan dalam tabel *hash* koneksi jaringan dan dibandingkan dengan model standar dari lalu lintas normal, dan lalu lintas paket yang tidak mematuhi arus diberi tanda dibatasi. Tingkat pembatasan lalu lintas paket diberlakukan secara dinamis, disesuaikan dengan perubahan perilaku lalu lintas. Hal ini diberlakukan untuk memfasilitasi pemulihan sistem yang cepat yang mengakomodasi kemungkinan kesalahan klasifikasi lalu lintas sah yang pada awalnya dicurgai sebagai bagian dari serangan. BotGAD melakukan pemantauan lalu lintas jaringan dan mendeteksi kecendrungan perilaku *bot* (*group activity*) pada 2 periode waktu secara berurutan. Jika terjadi kesamaan aktivitas *bot* pada 2 periode waktu ini, dipastikan bahwa jaringan sudah terinfeksi *bot* atau menjadi bagian dari jaringan *bot* (*botnet*).
2. Pertahanan dari Sisi Target / Penyedia Layanan
   Pertahanan dari sisi target/penyedia layanan bertujuan untuk mencegah infrastruktur jaringan/penyedia layanan kehabisan sumber daya, sehingga masih dapat memberikan layanan kepada *client* yang *legitimate*. Contoh mekanisme pertahanan yang menggunakan pendekatan ini adalah *Client Puzzles* [9] dan *Hop-count filtering* [19]. *Client Puzzle* bekerja dengan cara melakukan validasi terhadap *client* yang meminta layanan dari *server*. Metode ini mengharuskan *client* untuk mengerjakan *puzzle* untuk mendapatkan hak akses kepada layanan yang disediakan oleh *server*. *Puzzle* dapat berupa perhitungan matematis, kriptografi dan persoalan lain yang dalam penyelesaiannya memerlukan konsumsi sumber daya (misal CPU dan memori) pada *client*. *Hop-count filtering* melakukan validasi paket dengan menghitung *hop-count* paket dan membandingkannya dengan *hop-count database* pada saat jaringan dalam keadaan normal (tidak terjadi serangan). Wang et al. mengajukan penyaringan *hop-count* didasarkan pada pengamatan secara acak bahwa sebagian besar paket palsu tidak membawa/menyimpan nilai *hop-count* yang konsisten dengan alamat palsu yang digunakan. Dengan demikian, nilai *time-to-live* (TTL) dalam paket dapat digunakan untuk memutuskan apakah sebuah paket adalah paket palsu. Sebuah *router* menurunkan nilai TTL alamat IP yang telah dilalui dari sebuah paket sebelum meneruskan paket ke *next-hop*. Nilai TTL akhir ketika sebuah paket mencapai tujuannya adalah nilai awal TTL dikurangi dengan jumlah *hop* yang telah dilalui (yaitu *hop-count*).

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.

D. Network Behavior

Ketika serangan DDoS ataupun DoS dilancarkan ke suatu *server*, maka akan terlihat perilaku *bot* yang secara signifikan mempengaruhi jaringan dan terjadi pada waktu yang hampir bersamaan [3-5] Perilaku ini disebut dengan *network behavior*. Contoh yang sangat jelas dari *network behavior* ini adalah meningkatnya paket *service request* yang ditujukan pada sebuah layanan jaringan/*server* tertentu. Di bawah komando penyerang, *bot* secara serentak melakukan *service request* ke sebuah *server* dengan tujuan merebut semua sumber daya *server* sehingga *server* tidak dapat melayani *service request* yang sah. Pengamatan *network behavior* akan sangat bermanfaat untuk mencegah serangan DDoS/DoS di tingkat *application layer*. *Network behavior* yang terjadi ini dapat dinyatakan dalam *network density* ketika *bot* menyerang *server*.

Network density digunakan untuk menyatakan bagian dari koneksi potensial yang merupakan koneksi aktif seperti dinyatakan dalam (1). Koneksi potensial/*possible connection* adalah koneksi yang mungkin terjadi antara dua "*node*", terlepas apakah koneksi tersebut ada atau tidak. Sebaliknya, koneksi aktif/*occuring connection* adalah koneksi yang benar-benar ada dan menghubungkan antara dua "*node*" [17]. Ilustrasi perhitungan *network density* dapat dilihat pada Gambar 2.

$$Network\ Density = \frac{Occuring\ Connection}{Possible\ Connection} \quad (1)$$

Occuring connection adalah akses yang sedang terjadi terhadap satu tujuan tertentu. Jika berkaitan dengan akses *web service*, jumlah *occuring connection* dapat dilihat pada jumlah alamat IP yang melakukan akses terhadap satu *web service* tertentu pada interval waktu tertentu. *Possible connection* berkaitan dengan akses *web service*

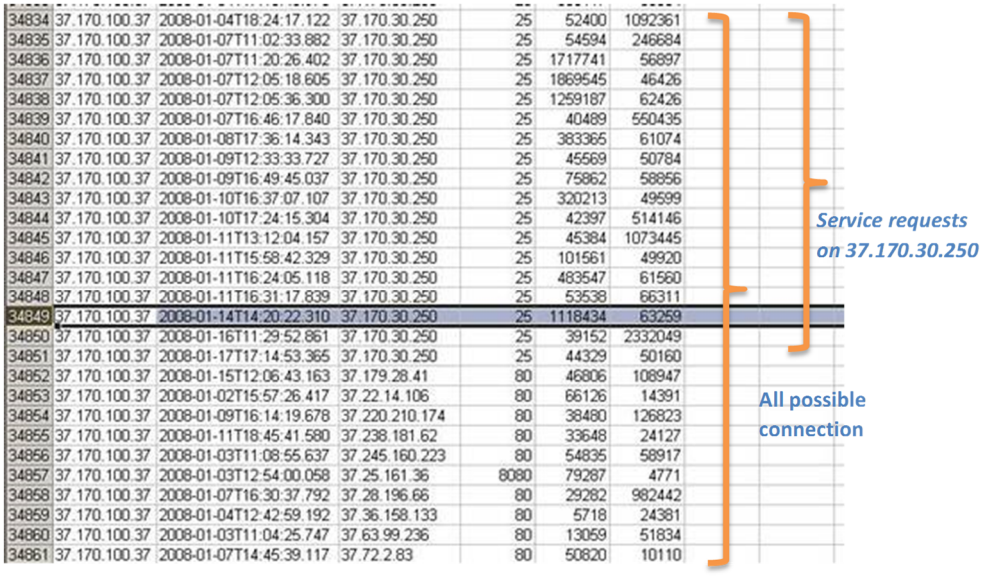
dapat dilihat pada seluruh permintaan layanan yang masuk ke seluruh *web service* yang tersedia dalam interval waktu tertentu. Dengan kata lain, *possible connection* merupakan jumlah seluruh *service request* yang tercatat pada interval waktu tertentu. Dengan mengetahui *possible connection* yang ada dan *occuring connection* terhadap *web service* tertentu, dapat diketahui tingkat kepadatan jaringannya.

E. Client Puzzle

Salah satu pendekatan untuk menanggulangi DDoS dari sisi penyedia layanan/target serangan DDoS adalah penerapan protokol *Client Puzzle* [12][20]. Protokol *Client Puzzle* ini bertujuan untuk bertahan dari serangan yang mengurangi kemampuan *server* melayani *service request* pada tahap awal, yaitu koneksi. Ide yang mendasari protokol ini sangat sederhana, ketika tidak ada tanda-tanda serangan, maka *server* menerima permintaan koneksi seperti biasa, tanpa membeda-bedakan tiap *service request*. Namun ketika mengalami serangan (jumlah *service request* ke *server* tiba-tiba meningkat), *server* akan mengirimkan *puzzle* ke masing-masing *client* yang meminta layanan [15]. Pada prinsipnya, *client* harus selalu menyediakan sumber dayanya untuk menjalankan protokol pengujian *Client Puzzle* sebelum akhirnya mengalokasikan sumber daya yang *server* miliki untuk melayani permintaan *client*. Dengan pendekatan ini, akses layanan dapat diberikan dengan tepat sasaran kepada *client* yang *legitimate*. Meskipun terdapat upaya serangan, sumber daya penyerang akan terkuras dan memerlukan upaya dan waktu lebih untuk dapat melumpuhkan *server*. Bentuk *puzzle* yang digunakan beraneka ragam, biasanya berupa *cryptographic puzzle*. Metode yang digunakan sebagai mekanisme pembuatan dan penyelesaian *puzzle* juga beraneka ragam, diantaranya dengan memanfaatkan sifat *hash fuction*, juga metode enkripsi dan deskripsi[1-2][9]. Metode *Client Puzzle* dapat dilakukan sebagai langkah pertahanan terhadap DDoS di tingkat *application layer*.

Berikut ini adalah deskripsi bagaimana proses *Client Puzzle* berjalan. Pertama-tama *client* mengirimkan pesan R1 kepada *server* beserta pesan lain berisi permintaan *puzzle*, dengan tidak memperhatikan apakah *server* tersebut mendistribusikan *puzzle* atau tidak. Apabila *server* tidak sedang diserang, *server* akan memberikan sinyal balasan bahwa tidak ada *puzzle* yang didistribusikan. Pada saat membalas pesan ini, pesan R1 diberikan hak untuk mendapat layanan dari *server*. Pesan R1 kemudian disimpan pada sebuah *register*. Hak mendapat layanan dari *server* ini diberlakukan selama beberapa waktu. Setelah melewati batas waktu tertentu, maka pesan R1 tidak akan dilayani lagi oleh *server*. *Client* memberi respon dalam rentang waktu ini. *Client* dapat dengan leluasa meminta layanan R1 dieksekusi beberapa kali, selama masih dalam rentang waktu hak akses [2]. Gambar 3a merupakan gambaran umum protokol *Client Puzzle* pada kondisi normal.

Server dikatakan sedang menerima serangan ketika memori penyimpanan mulai penuh, dimana koneksi yang terhubung dengan *server* melebihi batas maksimal koneksi pada *server* tersebut. Pada kasus ini, ketika *client* meminta persetujuan untuk mengeksekusi layanan R1, maka *client* akan menerima *Client Puzzle* P dari *server*. Agar *server* memperbolehkan eksekusi layanan R1, *client* harus mengirimkan solusi *puzzle* P yang benar dalam waktu T1. Ketika hal tersebut sudah dilakukan, maka *server* menyediakan satu slot untuk layanan R1 pada penyimpanan hak akses. Ini berarti *client* dapat mengakses layanan R1. Gambar 3b merupakan gambaran umum



34834	37.170.100.37	2008-01-04T18:24:17.122	37.170.30.250	25	52400	1092361
34835	37.170.100.37	2008-01-07T11:02:33.882	37.170.30.250	25	54594	246684
34836	37.170.100.37	2008-01-07T11:20:26.402	37.170.30.250	25	1717741	56897
34837	37.170.100.37	2008-01-07T12:05:18.605	37.170.30.250	25	1869545	46426
34838	37.170.100.37	2008-01-07T12:05:36.300	37.170.30.250	25	1259187	62426
34839	37.170.100.37	2008-01-07T16:46:17.840	37.170.30.250	25	40489	550435
34840	37.170.100.37	2008-01-08T17:36:14.343	37.170.30.250	25	383365	61074
34841	37.170.100.37	2008-01-09T12:33:33.727	37.170.30.250	25	45569	50784
34842	37.170.100.37	2008-01-09T16:49:45.037	37.170.30.250	25	75862	58856
34843	37.170.100.37	2008-01-10T16:37:07.107	37.170.30.250	25	320213	49599
34844	37.170.100.37	2008-01-10T17:24:15.304	37.170.30.250	25	42397	514146
34845	37.170.100.37	2008-01-11T13:12:04.157	37.170.30.250	25	45384	1073445
34846	37.170.100.37	2008-01-11T15:58:42.329	37.170.30.250	25	101561	49920
34847	37.170.100.37	2008-01-11T16:24:05.118	37.170.30.250	25	483547	61560
34848	37.170.100.37	2008-01-11T16:31:17.839	37.170.30.250	25	53538	66311
34849	37.170.100.37	2008-01-14T14:20:22.310	37.170.30.250	25	1118434	63259
34850	37.170.100.37	2008-01-16T11:29:52.861	37.170.30.250	25	39152	2332049
34851	37.170.100.37	2008-01-17T17:14:53.365	37.170.30.250	25	44329	50160
34852	37.170.100.37	2008-01-15T12:06:43.163	37.179.28.41	80	46806	108947
34853	37.170.100.37	2008-01-02T15:57:26.417	37.22.14.106	80	66126	14391
34854	37.170.100.37	2008-01-09T16:14:19.678	37.220.210.174	80	38480	126823
34855	37.170.100.37	2008-01-11T18:45:41.580	37.238.181.62	80	33648	24127
34856	37.170.100.37	2008-01-03T11:08:55.637	37.245.160.223	80	54835	58917
34857	37.170.100.37	2008-01-03T12:54:00.058	37.25.161.36	8080	79287	4771
34858	37.170.100.37	2008-01-07T16:30:37.792	37.28.196.66	80	29282	982442
34859	37.170.100.37	2008-01-04T12:42:59.192	37.36.158.133	80	5718	24381
34860	37.170.100.37	2008-01-03T11:04:25.747	37.63.99.236	80	13059	51834
34861	37.170.100.37	2008-01-07T14:45:39.117	37.72.2.83	80	50820	10110

Gambar. 2. Ilustrasi kepadatan jaringan dalam sebuah *service request log*

protokol Client Puzzle pada kondisi jaringan padat.

III. METODOLOGI

Bab ini memaparkan tentang langkah-langkah yang dilakukan pada penelitian yang dilakukan..

A. Perumusan Masalah

Masalah utama yang menjadi sorotan dalam penelitian ini adalah peluang terjadinya serangan DDoS melalui *service request* dalam jumlah besar sehingga dapat melumpuhkan kinerja *web service* [14][18]. Untuk menangani masalah ini, penulis mengemukakan sebuah metode pengamanan *web service* dari sisi penyedia layanan. Pendekatan ini dilakukan dengan melakukan filtrasi dan validasi *service request* menggunakan *Network Behavior Analysis* dan *Client Puzzle* sehingga layanan yang dilayani oleh *web service* adalah *service request* yang sah.

Dari sini, permasalahan berkembang ke karakreristik jaringan DDoS apa yang yang dapat dijadikan parameter serangan, mekanisme *Client Puzzle* yang dijalankan untuk validasi, serta bagaimana melakukan integrasi *Network Behavior Analysis* dengan *Client Puzzle* sehingga dapat menjaga kemampuan sistem melayani permintaan yang sah.

B. Studi Literatur

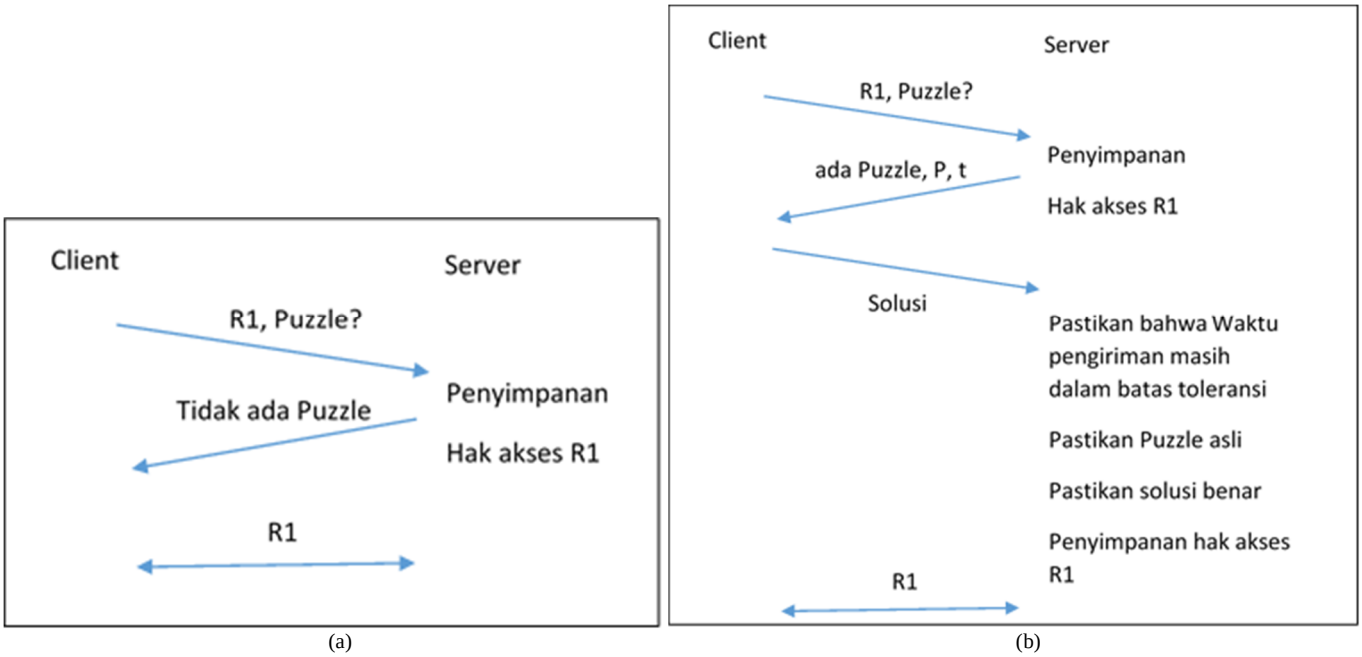
Setelah mengetahui permasalahan yang dihadapi, langkah selanjutnya adalah melakukan pengkajian materi yang berkaitan dengan topik penelitian yang diambil. Pada penelitian ini, referensi yang digunakan adalah jurnal-jurnal yang berkaitan dengan DoS, DDoS, dan *Client Puzzle*. Selain itu, *short paper* yang termuat pada prosiding-prosiding, artikel ilmiah yang terkait dan situs-situs penyedia informasi yang terkait. Dari studi literatur yang telah dilakukan maka diperoleh informasi yang berkaitan dengan penelitian yang dilakukan ini, seperti berikut:

a. Tipe serangan DoS dan DDoS, karakteristik dari serangan tersebut dan bagaimana serangan DoS dan DDoS dilancarkan.
b. Penelitian-penelitian terkait dengan pertahanan terhadap serangan DoS dan DDoS.
c. Teknik pengamanan *web service* yang sudah ada, terutama yang berkaitan dengan pengamanan dari sisi korban/*server* penyedia layanan.
d. Protokol dan arsitektur yang mendukung pengamanan *web service*, terutama protokol *Client Puzzle*.

Dari studi literatur, dapat disimpulkan juga mengenai kondisi pada saat serangan terjadi:

a. DoS: Kalau ada *request* dari alamat IP yang sama untuk fungsi yang sama dalam jumlah besar dalam rentang waktu yang hampir bersamaan
b. DDoS: Kalau ada *request* dari sejumlah alamat IP untuk fungsi yang sama dalam jumlah besar dalam rentang waktu tertentu (memperlihatkan aktifitas grup)

Kedua hasil observasi diatas dapat dijadikan sebagai indikator terjadinya serangan DoS/DDoS.



Gambar. 3. Protokol *Client Puzzle* ketika diakses oleh *client* yang berhak (a) dan ketika melakukan validasi *client* (b)

C. Perancangan Protokol

Protokol yang dirancang dalam penelitian ini adalah protokol *Network Behavior Analysis* (yang memanfaatkan karakteristik serangan DDoS dalam tingkat kepadatan jaringan/*network density*) dan protokol *Client Puzzle* (yang memanfaatkan operasi *string* untuk melakukan validasi *client*). Protokol kombinasi ini nantinya akan berjalan pada *application layer*. Gambaran Umum Sistem secara umum dapat kita lihat pada Gambar 4.

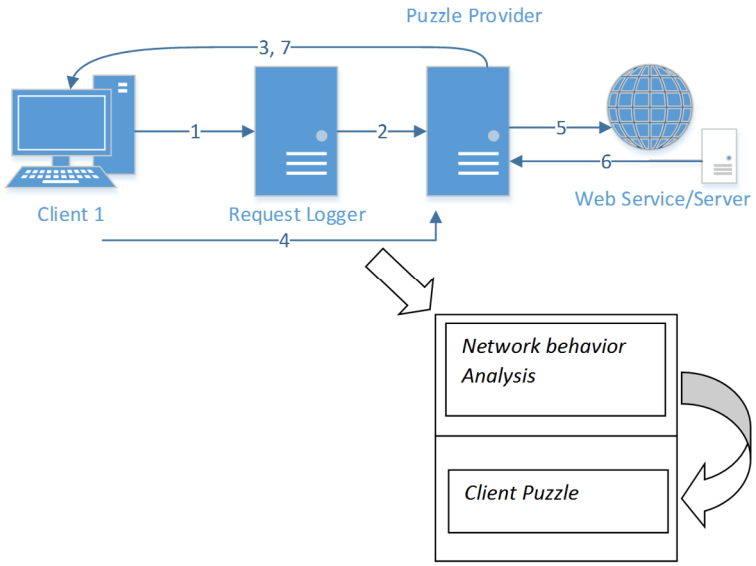
Entitas yang terlibat dalam sistem antara lain *Client*, *Attacker/Bot*, *Request Handler*/*Puzzle Provider*, *Server*/*Service provider*. Proses deteksi DoS/DDoS terjadi di *Request Handler*/*Puzzle Provider* dan secara teknis terpisah dari *server web service*. Hal ini menjadi suatu keuntungan karena proses deteksi serangan tidak mengganggu kinerja *server* untuk melayani permintaan yang valid/*legitimate*. Dalam menjalankan fungsinya menyaring *service request*, *request handler* pertama-tama menjalankan network analysis behavior untuk menganalisa lalu lintas jaringan. Jika terjadi abnormalitas pada jaringan, maka *request handler* akan menjalankan sistem *Client Puzzle* untuk melakukan validasi *service request* lebih lanjut.

1. Protokol *Network Behavior Analysis*

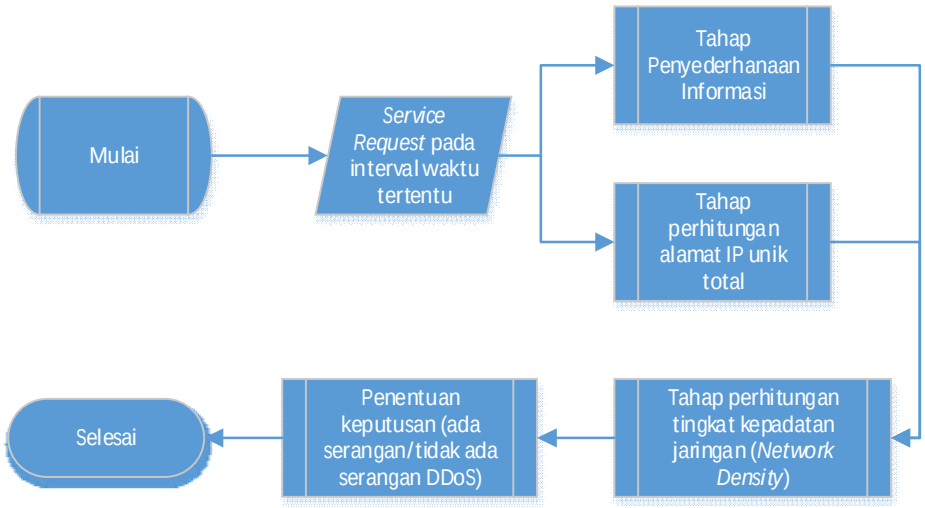
   Protokol *Network Behavior Analysis* adalah protokol yang digunakan untuk mendeteksi jika terjadi kemungkinan serangan DDoS secara *realtime*. Tujuan dari protokol ini adalah untuk mendapatkan karakteristik lalu lintas *service request* pada jaringan yang nantinya akan digunakan sebagai parameter penentu/*threshold* untuk memutuskan apakah terdapat serangan DDoS atau tidak. Protokol *Network Behavior Analysis* ini akan berjalan terus-menerus untuk mendata setiap permintaan yang masuk dan akan menjalankan fungsi analisisnya dalam interval waktu tertentu yang ditentukan. Alur dari protokol ini dapat dilihat pada Gambar 5.

   Protokol ini diawali dengan pengiriman *service request*. Permintaan layanan ini akan diterima oleh *request logger* untuk kemudian disimpan ke dalam sebuah *log service request* ketika sudah memenuhi interval waktu tertentu. Format *log* yang digunakan dalam proses ini mengikuti aturan berikut:
   req<spasi>*requestedService*<spasi>*param1*<spasi>*param2* ...

   Log ini kemudian akan dianalisa untuk mengetahui kemungkinan adanya serangan DDoS yang dibuat tiap satuan waktu yang ditentukan. Informasi dari *log* ini kemudian disederhanakan menjadi daftar nama-nama layanan yang diminta beserta alamat IP mana saja yang meminta layanan tersebut. Langkah berikutnya adalah menghitung tingkat kepadatan jaringan/*network density* untuk masing-masing layanan yang dituju. Setelah didapatkan tingkat kepadatan masing-masing layanan dan layanan yang sedang terserang DDoS, semua alamat IP yang mengakses layanan tersebut dicatat untuk akhirnya divalidasi oleh metode *Client Puzzle*. Perhitungan tingkat kepadatan jaringan dilakukan dengan konsep *network density* yang telah dijelaskan pada bab 2.



Gambar. 4. Arsitektur metode yang diusulkan



Gambar. 5. Diagram alir protokol *Network Behavior Analysis*

2. Protokol *Client Puzzle*

Protokol *Client Puzzle* merupakan protokol yang digunakan untuk melakukan validasi *service request*, apakah layanan yang datang termasuk serangan DDoS atau merupakan *service request* yang sah. Protokol ini melibatkan tiga elemen, antara lain *client*, *puzzle provider* dan *web service*. Tujuan dari protokol ini adalah membuat *puzzle* yang akan dikerjakan oleh *client*, juga untuk memvalidasi jawaban dari *client*. Gambar 6 menunjukkan mekanisme *Client Puzzle* yang diusulkan. Protokol ini menggunakan daftar alamat IP yang dicurigai sebagai serangan DDoS oleh protokol *Network Behavior Analysis*. Dari informasi ini, *puzzle provider* akan menghubungi *client* untuk melakukan validasi. Secara singkat, alur *puzzle* dapat diurutkan sebagai berikut:

1. *Puzzle provider* mendapatkan alamat IP *client* yang dicurigai.
2. Pemilihan soal *puzzle* dan partial *rule* dari sisi *puzzle provider* secara random.
3. Pengiriman soal *puzzle* dan partial *rule* dari *puzzle provider* ke *client*.
4. *Client* menerima soal *puzzle* dan partial *rule* dari *puzzle provider*.
5. Pemilihan partial *rule* secara random di *client*.
6. Penggabungan kedua partial *rule* untuk dijadikan pedoman pengerjaan *puzzle*.
7. Pengerjaan *puzzle* oleh *client* dengan petunjuk *rule*.
8. Pengiriman *puzzle* dan jawaban ke *puzzle provider*.
9. Validasi kebenaran jawaban *client* (dengan table lookup) oleh *puzzle provider*.
8. *Puzzle provider* memberi akses atau menolak akses *client* ke *web service*.

Ketika jawaban benar, maka *puzzle provider* akan meneruskan *service request* dari *client* ke *web service* dan mengirimkan jawaban ke *client*.

Berikut ini adalah beberapa hal yang perlu diketahui berkaitan dengan rancangan protokol *Client Puzzle* yang diusulkan:

a. Soal *puzzle*

Soal *puzzle* berupa suatu kumpulan karakter dalam bentuk *string*. Sebagai contoh, soal dapat berupa bilangan biner 01101 atau angka 73425 atau huruf abcde ataupun kombinasi dari ketiganya. *Puzzle provider* memiliki beberapa alternatif soal yang tersimpan pada tabel soal. Ketika ada *service request* dari *client*, *puzzle provider* akan mengirimkan soal *puzzle* yang dipilih secara acak. Dalam implementasi yang dilakukan, soal ditentukan berupa bilangan biner.

b. *Rule* pengerjaan *puzzle*

Rule berupa aturan penyusunan/pengerjaan *puzzle* sehingga didapatkan hasil yang sesuai. *Rule* yang digunakan untuk pengerjaan *puzzle* merupakan hasil kombinasi/penggabungan dari partial *rule* yang berasal dari *server* dan partial *rule* dari *client*. Dengan sifatnya yang merupakan kombinasi, maka akan menambah tingkat keunikan jawaban yang diperoleh dari *rule* tersebut.

c.

Baik *rule* dari *puzzle provider* maupun dari *client* memiliki struktur yang sama, yaitu gabungan dari huruf dan angka. Huruf menentukan operasi pengerjaan seperti apa yang akan dilakukan pada *puzzle*, sedangkan angka menentukan jumlah operasi "huruf" yang dilakukan. Operasi pengerjaan *puzzle* ini ada dua jenis, yaitu operasi *circular shift left* dan operasi *circular shift right*.

Sebagai contoh, *rule* pengerjaan sebuah *puzzle* dapat didefinisikan seperti “q2” (yang berarti pada soal *puzzle* diberlakukan operasi *circular shift left* sebanyak dua kali) atau “p1” (yang berarti pada soal *puzzle* diberlakukan operasi *circular shift right* sebanyak satu kali). Ketika mekanisme *Client Puzzle* dijalankan, baik *puzzle provider* maupun *client* akan menentukan *partial key* yang dipilih secara acak untuk kemudian digabungkan dan digunakan untuk pengerjaan *puzzle*.

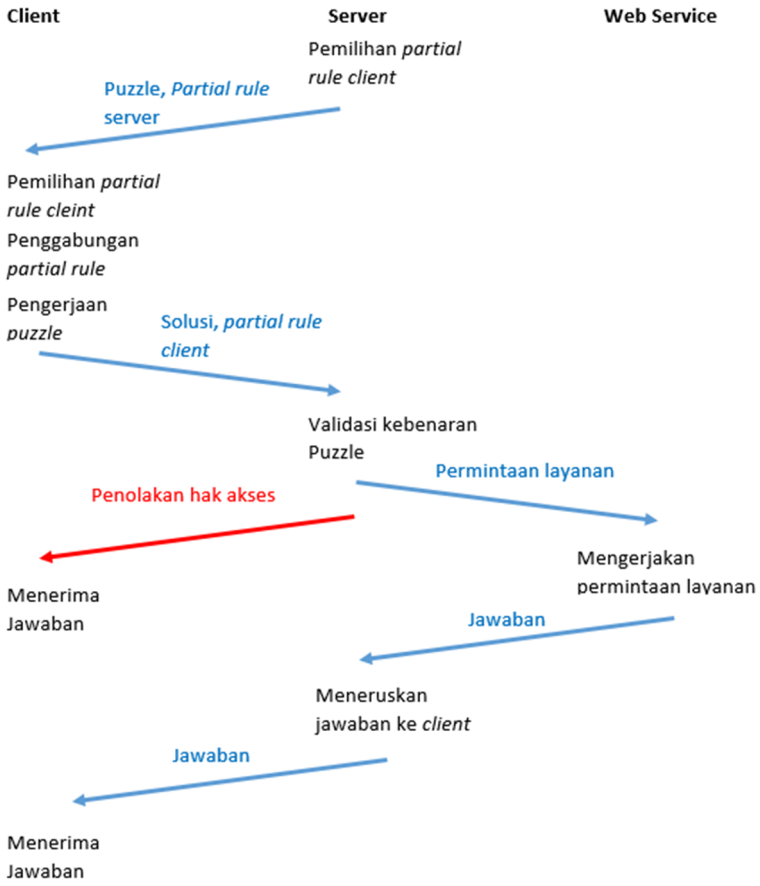
d. Proses pengerjaan *puzzle*

Pengerjaan *puzzle* dilakukan dengan cara memodifikasi urutan *string puzzle* sesuai dengan *rule* yang ada (operasi *circular shift left* atau *circular shift right*). Proses ini terus dilakukan hingga *rule* terpakai seluruhnya. Berikut ini akan diberikan contoh pengerjaan *puzzle*:

Misalkan *puzzle* yang harus diselesaikan adalah “01001”, partial *rule* yang terpilih secara acak dari *puzzle provider* adalah “p3” dan partial *rule* yang terpilih secara acak dari *client* adalah “q1”. Berdasarkan *rule* pengerjaan *puzzle*, maka akan dilakukan operasi *circular shift right* sebanyak tiga kali dan *clircular shift left* sebanyak satu kali. Untuk operasi pertama didapatkan hasil “00101”, kemudian dilakukan operasi kedua untuk memperoleh hasil akhir, yaitu “01010”.

e. Proses validasi *puzzle*

Puzzle provider memiliki jawaban untuk tiap kombinasi yang tersimpan dalam tabel sehingga untuk



Gambar. 6. Mekanisme *Client Puzzle* yang diusulkan

melakukan validasi jawaban *puzzle* dari *client*, dapat dilakukukan dengan cepat melalui *table lookup*.

D. Perancangan DDoS

Untuk dapat melakukan uji coba sistem, serangan DDoS perlu dimodelkan terlebih dahulu. Pemodelan ini dilakukan berdasarkan karakteristik serangan DDoS yang diamati oleh Choi dan rekan-rekan: “Ketika serangan DDoS dilancarkan ke suatu *server*, maka akan terlihat perilaku *daemon*/komputer inang yang secara signifikan mempengaruhi jaringan dan terjadi pada waktu yang hampir bersamaan” [3-5]. Jenis serangan DDoS yang dilakukan adalah jenis *request flooding*, dimana penyerang membanjiri jaringan dengan banyak *request* terhadap sebuah *web service*, sehingga *web service* tidak dapat melayani permintaan dari *client* yang membutuhkan

layanan tersebut.

Pemodelan serangan DDoS ini dilakukan dengan bantuan aplikasi pengujian Apache Jmeter. *Web service* akan diuji dengan serangan DDoS yang dimodelkan dari 254 *host* yang melakukan permintaan layanan secara bersamaan. *Request* layanan akan dilakukan secara terus menerus hingga ditemukan kegagalan fungsi pada *web service*. Pola serangan layanan seperti inilah yang nantinya akan digunakan sebagai kriteria DDoS untuk pengujian fungsionalitas sistem.

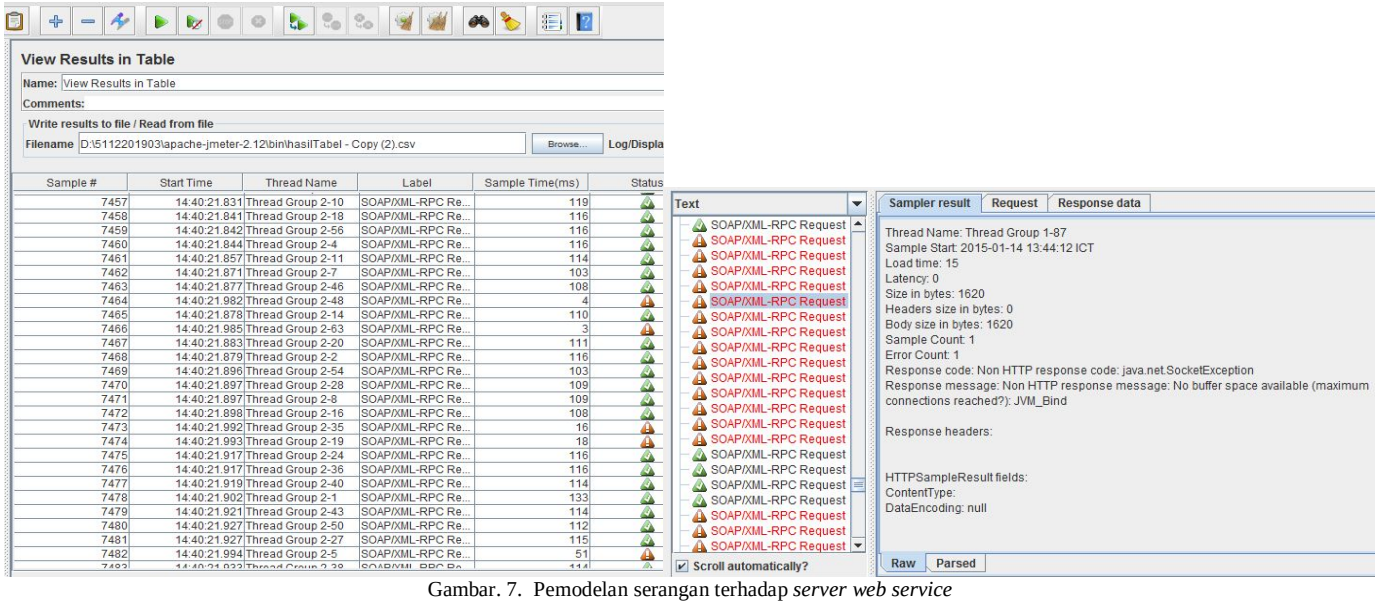
Dari Gambar 7, dapat dilihat bahwa *server* mulai mengalami kegagalan dalam melayani permintaan layanan pada *request* ke 7464 dan kegagalan yang dialami adalah *No buffer space available (maximum connection reached*). Hal ini menunjukkan bahwa serangan DDoS telah berhasil menguras sumber daya *server*. Perlu diketahui bahwa batas maksimal *request* layanan yang dapat ditangani *server* ini bisa berbeda-beda, tergantung kemampuan komputasi *server*. Untuk keperluan pengujian pada penelitian ini, akan dibuat traffic DDoS yang melebihi 7464 *request* DDoS yang menyesuaikan spesifikasi komputer yang digunakan dalam penelitian ini. Ada 10 *web service* yang berjalan pada sisi *server*, dengan 1 *web service* yang menjadi sasaran serangan. Jumlah *web service request* total pada *log* adalah adalah 9660. Berikut ini adalah spesifikasi pemodelan *log*:

- http://10.151.14.2:8080/tc-ws/1.ncc.if.its.ac.id diakses sebanyak 81 kali
- http://10.151.14.2:8080/tc-ws/2.ncc.if.its.ac.id diakses sebanyak 88 kali
- http://10.151.14.2:8080/tc-ws/3.ncc.if.its.ac.id diakses sebanyak 27 kali
- http://10.151.14.2:8080/tc-ws/4.ncc.if.its.ac.id diakses sebanyak 241 kali
- http://10.151.14.2:8080/tc-ws/5.ncc.if.its.ac.id diakses sebanyak 123 kali
- http://10.151.14.2:8080/tc-ws/6.ncc.if.its.ac.id diakses sebanyak 32 kali
- http://10.151.14.2:8080/tc-ws/7.ncc.if.its.ac.id diakses sebanyak 111 kali
- http://10.151.14.2:8080/tc-ws/8.ncc.if.its.ac.id diakses sebanyak 100 kali
- http://10.151.14.2:8080/tc-ws/9.ncc.if.its.ac.id diakses sebanyak 94 kali
- http://10.151.14.2:8080/tc-ws/10.ncc.if.its.ac.id (target serangan DDoS) diakses sebanyak 8763 kali

E. Mekanisme Pengujian

Untuk menjamin kemampuan sistem dalam mendeteksi DDoS dan menguji performanya, maka pengujian dibagi menjadi dua fokus utama, yaitu uji fungsionalitas dan uji performa sistem. Untuk melakukan uji coba fungsionalitas sistem, pengujian dilakukan secara menyeluruh yaitu mencakup *Network Behavior Analysis* dan dan terus dilanjutkan dengan *Client Puzzle*. Dengan perlakuan ini, maka dapat terbukti bahwa sistem yang dibangun dapat menyelesaikan permasalahan deteksi DDoS dengan baik.

Untuk uji coba performa sistem, pengujian dilakukan secara terpisah sehingga bisa didapatkan performa yang akurat dari masing-masing komponen sistem deteksi DDoS. Pengujian terpisah yang dimaksudkan adalah performa mekanisme *Network Behavior Analysis* diuji tersendiri, kemudian performa mekanisme *Client Puzzle*



Gambar. 7. Pemodelan serangan terhadap *server web service*

pun diuji tersendiri. Hal ini dilakukan dengan harapan bahwa performa sistem dapat terukur dengan lebih akurat. Penjelasan lebih lanjut mengenai masing-masing pengujian akan dijelaskan pada bagian berikut.

1. Pengujian fungsionalitas

Pengujian fungsionalitas sistem dimaksudkan untuk mengetahui apakah sistem dapat menjalankan fungsinya dengan baik. Pada percobaan ini, akan dilancarkan serangan DDoS ke satu *web service* tertentu, kemudian *client* yang sah mencoba juga mengakses *web service* pada waktu yang hampir bersamaan pada saat terjadi serangan. Parameter yang dilihat adalah

a. Sistem dapat berjalan dengan baik (dapat mendeteksi terjadinya serangan DDoS).
b. Sistem dapat melayani *service request* yang sah walaupun sedang mengalami serangan DDoS.

2. Pengujian performa

Untuk mengukur performa sistem, dilakukan pendekatan pengukuran nilai *recall* dan *precision* dari metode *Network Behavior Analysis*. *Recall* adalah persentase kondisi kenyataan positif yang terprediksi sebagai kondisi positif juga. *Recall* menyatakan tingkat sensitifitas sistem. *Precision* adalah persentase prediksi positif yang benar. Untuk dapat mengetahui nilai dari *recall* dan *precision*, terlebih dahulu didefinisikan *confusion matrix* sebagai dasar perhitungannya. *Confusion matrix* merupakan sarana untuk mengelompokkan hasil pengujian dari tiap skenario untuk kemudian dianalisis kecendrungannya. Berikut ini penjelasan dari masing-masing kelas pada *confusion matrix*:

- *True Positive* (TP): kondisi kenyataan terdapat serangan DDoS terprediksi sebagai kondisi terdapat serangan DDoS pula. Hal ini berarti prediksi tepat.
- *False Positive* (FP): kondisi kenyataan tidak ada serangan DDoS terprediksi sebagai kondisi ada serangan DDoS. Hal ini berarti prediksi tidak tepat.
- *True Negative* (TN): kondisi kenyataan tidak ada serangan DDoS terprediksi sebagai kondisi tidak ada serangan DDoS. Hal ini berarti prediksi tepat.
- *False Negative* (FN): kondisi kenyataan ada serangan DDoS terprediksi sebagai kondisi tidak ada serangan DDoS. Hal ini berarti prediksi tidak dapat memprediksi kenyataan dengan tepat.

Nantinya hasil percobaan dari beberapa studi kasus akan dikelompokkan dalam empat kelas ini. Selanjutnya dari keempat kelas ini, dapat dilakukan pengujian *recall* dari sistem deteksi DDoS yang dikembangkan.

Recall/*Sensitivity* (R) merupakan merupakan persentasi pengaruh dari parameter yang ada dalam suatu pengambilan keputusan. Perhitungannya dapat dilihat dalam (2)

$$R = \frac{TP}{(TP+FN)} \quad (2)$$

Presisi (P) merupakan persentasi dari kenyataan benar yang terprediksi benar pula. Perhitungannya dapat dilihat dalam (3)

$$P = \frac{TP}{(TP+FP)} \quad (3)$$

Pengujian akan dilakukan dengan menjalankan 5 *request traffic log* untuk kemudian diukur *Recall*/*Sensitivity* dari sistem deteksi serangan DDoS ini. Dari uji coba ini diharapkan menghasilkan deteksi yang tepat, yaitu menemukan *web service* yang mengalami serangan dan juga alamat IP yang melakukan serangan, untuk berikutnya di validasi oleh mekanisme *Client Puzzle*. Nilai *threshold network density* akan dimodifikasi untuk melihat pengaruhnya dalam penentuan deteksi serangan.

IV. HASIL PENGUJIAN

Pengujian sistem dilakukan dengan dua pendekatan yang disesuaikan dengan tujuan evaluasi sistem. Untuk melakukan uji coba fungsionalitas sistem, pengujian dilakukan secara menyeluruh yaitu mencakup *Network Behavior Analysis* dan dan terus dilanjutkan dengan *Client Puzzle* (Hasil dari metode *Network Behavior Analysis* menjadi masukkan bagi metode *Client Puzzle*). Dengan perlakuan ini, maka dapat terbukti bahwa sistem yang dibangun dapat menyelesaikan permasalahan DDoS dengan baik. Untuk uji coba performa sistem, pengujian dilakukan secara terpisah sehingga bisa didapatkan performa yang akurat dari masing-masing komponen sistem deteksi DDoS.

1. Hasil uji coba fungsionalitas sistem

Mekanisme deteksi DDoS diuji dengan melihat melihat kemampuan sistem untuk menyelesaikan mekanisme *Network Behavior Analysis* dan *Client Puzzle* dengan baik. Proses pengujuan dimulai dengan menjalankan program *request logger* yang bertugas menerima permintaan layanan, kemudian menjalankan *puzzle provider* yang bertugas memberikan *puzzle*. Begitu *puzzle provider* sudah berjalan, baru kemudian kita jalankan *client* sebagai *service request*er yang meminta layanan. Gambar 8 menunjukkan hasil output dari proses eksekusi protokol deteksi DDoS. Pada pengujian fungsionalitas ini, dijalankan serangan DDoS.

Serangan DDoS dilancarkan dengan melakukan *request* secara terus-menerus ke satu *service* yaitu penjumlahan. Ketika serangan berlangsung, *client* yang *legitimate* pun mengakses *service* penjumlahan ini. Dari percobaan yang dilakukan, terbukti bahwa walaupun sedang mengalami serangan DDoS, layanan penjumlahan masih dapat diberikan kepada *client* yang *legitimate*.

2. Hasil uji coba performa sistem

Pada bagian ini, akan diuji performa sistem dalam mendeteksi serangan DDoS. Tujuan awal sistem (*Network Behavior Analysis*) adalah mendeteksi nama *web service* yang diserang DDoS dan mengidentifikasi alamat IP mana saja yang terlibat dalam penyerangan DDoS. Untuk dapat menilai performa dalam melakukan fungsionalitasnya, digunakan pendekatan *recall* dan *precision*. Pada pengujian performa ini, digunakan data uji *log* yang merepresentasikan 9660 *service request*, 10 *web service*, dan pola serangan DDoS (terdapat satu *web service* yang diakses secara serentak dan dalam jumlah permintaan layanan yang banyak). Nilai *threshold* kepadatan yang diujikan berada pada rentang interval 0 <= *threshold* <=1 dengan ketelitian dua angka di belakang koma.

Dari hasil *Network Behavior Analysis*, didapatkan tingkat kepadatan untuk masing-masing *web service*. Tingkat kepadatannya dapat dilihat sebagai berikut:

- 1.ncc.if.its.ac.id memiliki tingkat kepadatan 0,002795031
- 2.ncc.if.its.ac.id memiliki tingkat kepadatan 0,003312629
- 3.ncc.if.its.ac.id memiliki tingkat kepadatan 0,008385093
- 4.ncc.if.its.ac.id memiliki tingkat kepadatan 0,009109731
- 5.ncc.if.its.ac.id memiliki tingkat kepadatan 0,009730849
- 6.ncc.if.its.ac.id memiliki tingkat kepadatan 0,010351967
- 7.ncc.if.its.ac.id memiliki tingkat kepadatan 0,011490683
- 8.ncc.if.its.ac.id memiliki tingkat kepadatan 0,012732919
- 9.ncc.if.its.ac.id memiliki tingkat kepadatan 0,02494824
- 10.ncc.if.its.ac.id (target serangan DDoS) memiliki tingkat kepadatan 0,907142857

Tingkat kepadatan yang dimiliki masing-masing *web service* kemudian dibandingkan dengan nilai *threshold network density* untuk menentukan apakah terjadi serangan DDoS atau tidak. Pada Gambar 9 terlihat hasil deteksi DDoS oleh sistem dengan berbagai nilai *threshold network density*. Gambar 9 merupakan hasil dari percobaan yang bertujuan untuk melihat pengaruh nilai *threshold network density* terhadap ketepatan deteksi DDoS oleh sistem. Dari Gambar 9 dapat diketahui bahwa sistem dapat mendeteksi dengan baik perilaku aktivitas grup *botnet* walaupun dengan variasi *threshold*. Pada nilai *threshold* 0.1 hingga 0.9, serangan DDoS dapat terdeteksi (digambarkan dengan garis biru *true positive*). Pada nilai *threshold* tinggi (>0.91), sistem tidak mendeteksi serangan DDoS karena pada saat deteksi, *request log* tidak terisi oleh *request* DDoS saja (ada juga *request* normal). Nilai *threshold* 1 berarti seluruh isi *request log* terisi seluruhnya oleh *request* DDoS. Pada Nilai *threshold* yang rendah, tingkat terjadinya kondisi *false positive* sangat tinggi. (*web service* yang beroperasi normal dianggap mengalami serangan DDoS) Nilai *true positive*, *false negative*, sensitivitas dan presisi adalah seperti berikut:

Nilai TP = 91

Gambar. 8. Hasil pengujian fungsionalitas sistem

Nilai FP = 14

Nilai FN = 10

$$\text{Sensitifitas/}Recall = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP)+False\ Negative\ (FN)} \times 100\%$$

$$= \frac{91}{(91+10)} \times 100\% = 90.09\ \%$$

$$\text{Presisi/}Precision = \frac{True\ Positive\ (TP)}{True\ Positive(TP)+False\ Positive(FP)} \times 100\%$$

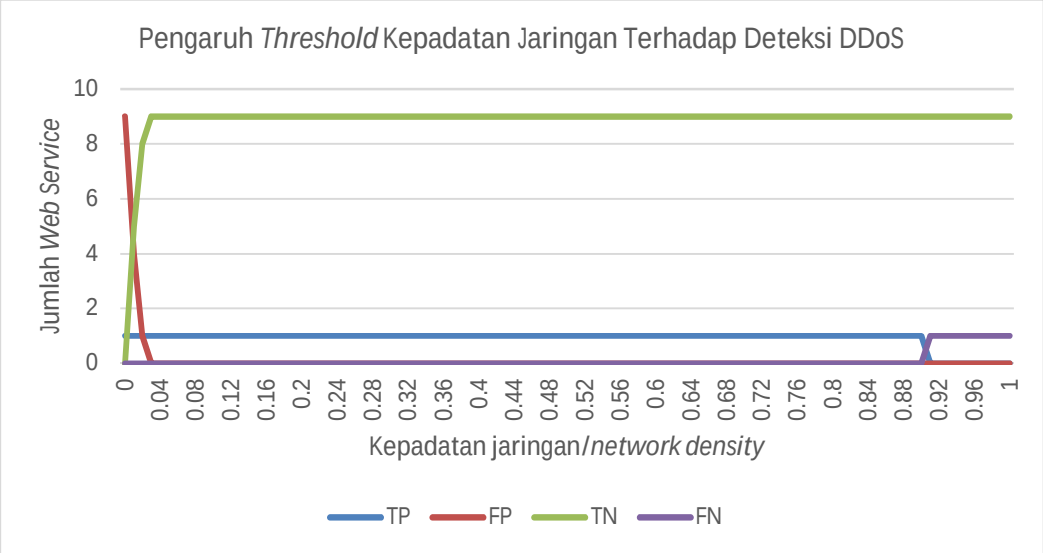
$$= \frac{91}{(91+14)} \times 100\% = 86.67\ \%$$

Pengujian berikutnya yang dilakukan untuk melihat pengaruh sistem *Network Behavior Analysis* terhadap validasi *client puzzle* adalah penambahan jumlah *request web service*. Hasil percobaan dapat dilihat pada Gambar 10. Terlihat bahwa pada awalnya metode yang diusulkan (*Network Behavior Analysis* + *Client Puzzle*) memerlukan waktu yang lebih lama jika dibandingkan dengan pengujian secara langsung dengan *Client Puzzle* saja. Hal ini disebabkan karena metode *Network Behavior Analysis* memerlukan waktu untuk melakukan *logging* permintaan layanan *web service* serta melakukan analisa dan penyederhanaan *log*. Namun pada saat jumlah *request* layanan diatas 20000, terlihat bahwa metode yang diusulkan memerlukan waktu yang lebih singkat untuk menyelesaikan validasi alamat IP. Terlihat juga ada Gambar 10 bahwa waktu yang diperlukan oleh mekanisme *client puzzle* untuk menyelesaikan validasi alamat IP berbanding lurus dengan jumlah *request* yang ada di *log*. Hal ini dikarenakan seluruh *request* yang ada pada *log* diikutsertakan dalam proses validasi. Sedangkan pada mekanisme yang diusulkan, peningkatan waktu yang terjadi tidak signifikan. Hal ini dikarenakan metode *Network Behavior Analysis* telah melakukan penyederhanaan *log* dan menentukan alamat IP mana saja yang harus diuji dengan metode *Client Puzzle*.

V. KESIMPULAN DAN SARAN

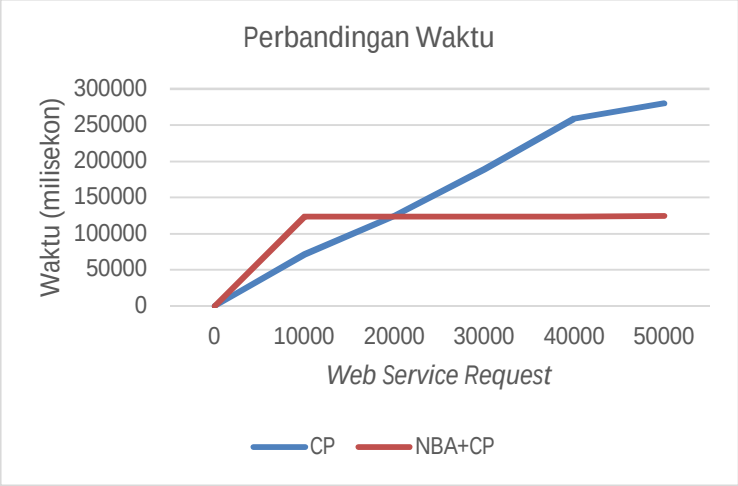
A. Kesimpulan

Adapun kesimpulan yang diambil berdasarkan dari hasil penelitian yang telah dilakukan dan analisa metode yang diusulkan adalah sebagai berikut:



Gambar. 9. Hasil deteksi *Network Behavior Analysis* dengan beberapa nilai *threshold* kepadatan jaringan/*network density*

1. Deteksi serangan DDoS dapat dilakukan dengan menerapkan metode pengamanan proaktif terhadap serangan yang ditujukan kepada sumber daya jaringan. Hal ini dibuktikan dari keberhasilan mekanisme yang diusulkan untuk mendeteksi serangan DDoS.



Gambar. 10. Ilustrasi serangan DDoS

2. Berdasarkan hasil analisa uji coba, tingkat presisi sistem untuk mendeteksi serangan DDoS sangat baik, mencapai tingkat 86.67% yang berarti sistem dapat mendeteksi dengan tepat *web service* yang diserang, namun mekanisme ini juga memiliki tingkat sensitifitas yang tinggi, yaitu 90.09% yang mendeskripsikan bahwa kemampuan sistem untuk mendeteksi serangan DDoS sangat tergantung pada parameter/*tresholding* yang ditentukan (dalam hal ini adalah tingkat kepadatan jaringan maksimal untuk identifikasi serangan DDoS). Namun hal ini tidak menjadi masalah yang besar karena perbedaan tingkat kepadatan jaringan serangan DDoS dan tingkat kepadatan jaringan *service request* kepada *web service* normal sangat jauh berbeda (tingkat kepadatan jaringan serangan DDoS berkisar antara 0.8 - 0.9, sedangan tingkat kepadatan jaringan *service request* yang normal berkisar antara 0.05 hingga 0.02).
3. Mekanisme *Network Behavior Analysis* menggunakan *network density* (kepadatan jaringan) dalam implementasinya dapat diintegrasikan dengan mekanisme *Client Puzzle*. Pengaruh *Network Behavior Analysis* terhadap sistem validasi secara keseluruhan antara lain:
   - Proses *Network Behavior Analysis* menyebabkan *request web service* yang harus divalidasi oleh mekanisme *Client Puzzle* menjadi lebih sedikit (akibat proses reduksi analitis dari *Network Behavior Analysis*).
   - Dengan berkurangnya jumlah *request web service* yang harus dianalisis, otomatis waktu untuk melakukan validasi secara keseluruhan menjadi lebih singkat.

B. Saran

Saran-saran yang dapat digunakan untuk melanjutkan pengembangan metode pengamanan ini terhadap serangan DDoS antara lain:

1. *Network Behavior Analysis* dapat diperlengkapi lagi dengan metode-metode alternatif untuk mengurangi dependensi total terhadap nilai *threshold network density*.
2. Pengembangan juga dapat dilakukan pada penerapan mekanisme *Client Puzzle* yang lain yang lebih efisien dan cepat.

DAFTAR PUSTAKA

[1] Abliz, Mehmud, and Taieb Znati. "A Guided Tour Puzzle For Denial Of Service Prevention". *2009 Annual Computer Security Applications Conference* (2009): n. pag.
[2] Aura, Tuomas, Pekka Nikander, and Jussipekka Leiwo. "DOS-Resistant Authentication with Client Puzzles". *Lecture Notes in Computer Science* (2001): 170-177.
[3] Choi, Hyunsang et al. "Botnet Detection By Monitoring Group Activities In DNS Traffic". *7th IEEE International Conference on Computer and Information Technology* (CIT 2007) (2007): n. pag.
[4] Choi, Hyunsang, Heejo Lee, and Hyogon Kim. "BotGAD". *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE* - COMSWARE "09 (2009): n. pag.
[5] Choi, Hyunsang, and Heejo Lee. "Identifying Botnets By Capturing Group Activities In DNS Traffic". *Computer Networks 56.1* (2012): 20-33.
[6] Falkenberg, Andreas et al. "A New Approach Towards DOS Penetration Testing On Web Services". *2013 IEEE 20th International Conference on Web Services* (2013): n. pag.
[7] Gu, Qijun, and Peng Liu. "Denial Of Service Attacks". *Handbook of Computer Networks* (2007): 454-468.
[8] Imperva,. Denial Of Service Attacks: A Comprehensive Guide To Trends, Techniques, And Technologies. Redwood City: Imperva, 2012. Print. ADC Monthly Web Attacks Analysis.
[9] Juels, Ari, and John Brainard. "Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks". *Proceedings of NDSS "99 (Networks and Distributed Security Systems)* (1999): 151-165. Print.

[10] Koopman, P. et al. “Comparing Operating Systems Using Robustness Benchmarks”. *Proceedings of SRDS”97: 16th IEEE Symposium on Reliable Distributed Systems* (1997): n. pag.
[11] Lau, F. et al. “Distributed Denial Of Service Attacks”. *SMC 2000 Conference Proceedings. 2000 IEEE International Conference on Systems, Man and Cybernetics. “Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions”* (Cat. No.00CH37166) (2000): n. pag.
[12] Laurens, Vicky, Abdulmotaleb El-Saddik, and Amiya Nayak. "Requirements for Client Puzzles to Defeat the Denial of Service and the Distributed Denial of Service Attacks." *Int. Arab J. Inf. Technol. 3.4* (2006): 326-333.
[13] Mirkovic, J., and P. Reiher. “D-WARD: A Source-End Defense Against Flooding Denial-Of-Service Attacks”. *IEEE Trans. Dependable and Secure Comput. 2.3 (2005)*: 216-232.
[14] Oliveira, Rui Andre, Nuno Laranjeiro, and Marco Vieira. “Experimental Evaluation Of Web Service Frameworks In The Presence Of Security Attacks”. *2012 IEEE Ninth International Conference on Services Computing* (2012): n. pag.
[15] Suriadi, Suriadi et al. “Defending Web Services Against Denial Of Service Attacks Using Client Puzzles”. *2011 IEEE International Conference on Web Services* (2011): n. pag.
[16] Specht, Stephen M., and Ruby B. Lee. "Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures." *ISCA PDCS*. 2004.
[17] The Vital Edge by Gideon Rosenblatt,. “What Is Network Density - And How Do You Calculate It?”. N.p., 2013. Tersedia: http://www.the-vital-edge.com/what-is-network-density/
[18] Vieira, Marco, Nuno Laranjeiro, and Henrique Madeira. “Assessing Robustness Of Web-Services Infrastructures”. *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN”07)* (2007): n. pag.
[19] Wang, Haining, Cheng Jin, and Kang G. Shin. “Defense Against Spoofed IP Traffic Using Hop-Count Filtering”. *IEEE/ACM Trans. Networking 15.1* (2007): 40-53.
[20] Waters, Brent, et al. "New client puzzle outsourcing techniques for DoS resistance." *11th ACM Conference on Computer and Communications Security. ACM*, 2004.