ANALISA PROSES KILL BOTS DALAM MEMPERTAHANKAN DARI SERANGAN DISTRIBUTED DENIAL OF SERVICE DENGAN MENGGUNAKAN STRUKTUR DATA BLOOM FILTER

Ian Febrian Reza Mulia Yulianto
Institut Sains Terapan dan Teknologi Surabaya
Surabaya
Ian.yulianto@live.com

Yosi Kristian, S.Kom., M.Kom.
Institut Sains Terapan dan Teknologi Surabaya
Surabaya

***Abstrak* - Distributed Denial of Service atau DDoS adalah salah satu serangan yang mengancam sekuritas dari server. Serangan tersebut mempunyai jenis yang beraneka ragam, contohnya adalah Flash Crowd. Serangan Flash Crowd adalah serangan yang mengandalkan banyaknya agent untuk mengakses server yang ingin diserang dan tidak adanya eksploitasi dari segi protokol maupun dari segi port yang ada. Flash Crowd mempunyai perilaku hampir sama seperti pengguna pada umumnya, dengan demikian penanganannya harus berdasarkan perilaku dari pengguna yang sebenarnya. Berdasarkan masalah tersebut terciptalah Kill Bots, sistem pertahanan terhadap serangan Flash Crowd. Pada tugas akhir ini, Kill Bots akan dianalisa dan diimplementasi ulang pada Apache web server dengan rupa modul Apache.**

Kill Bots mempunyai dua buah status yang menandakan kondisi dari server saat itu, yaitu NORMAL dan SUSPECTED. Kill Bots memulai proses pembedaan antara pengguna yang sah dengan zombie dimulai pada kondisi server mencapai status SUSPECTED, di mana kondisi server pada saat itu mempunyai load server lebih dari 70%. Proses pembedaan atau autentikasi yang berjalan dengan menggunakan bantuan captcha dilakukan pada semua request yang baru masuk pada saat status SUSPECTED. Perilaku yang diharuskan untuk menjawab sebuah tes akan membedakan apakah itu pengguna yang sah atau tidak. Perilaku request yang berbau zombie, maka akan dimasukkan pada sebuah tabel zombie yang bersifat temporer. Ketika request yang telah dikenali sebagai zombie masuk, maka server akan langsung memberikan nilai kembalian bahwa request tersebut tidak dapat melanjutkan requestnya.

Pengujian yang dilakukan adalah pengujian performa antara tiga kondisi pertahanan dari server. Hasil dari pengujian performa akan disajikan dalam bentuk grafik garis di mana nilai yang ditunjukkan adalah nilai respond time dari server tanpa pertahanan apa pun, server dengan pertahanan mod_evasive, dan server dengan pertahanan Kil Bots.

Kata kunci – Apache; server; internet; security; modul

1. PENDAHULUAN

Seiring dengan perkembangan teknologi, banyak serangan DoS (Denial of Service) yang dilancarkan oleh para profesional dengan menggunakan Botnets pada ribuan mesin. Untuk menghindari deteksi, zombie (penyerang DoS) mulai meninggalkan teknik bandwidth flooding dan beralih ke teknik serangan yang mencontoh gerak gerik web browser dengan client yang sangat banyak dan mempunyai tujuan ke arah sumber yang lebih berharga pada layer yang mempunyai tingkat lebih tinggi seperti CPU, database dan disk bandwidth.

Serangan DoS yang dibantu dengan adanya virus worm, dapat menginfeksi sampai dengan 30.000 mesin baru setiap harinya. Virus worm ini membawa sebuah mesin yang nantinya setelah diinfeksikan ke korban, maka mesin tersebut akan melancarkan serangan DoS dan serangan yang lain seraca terus menerus. Ketika teknik SYN flood gagal dijalankan, maka mereka akan melancarkan teknik HTTP flood, mengunduh banyak gambar yang berukuran besar dari korban. Dalam contoh yang lain, zombie melancarkan permintaan melalui mesin pencari dengan jumlah yang sangat besar, sehingga membuat server rusak.

Pemberantasan serangan DoS ini adalah sebuah tantangan, karena permintaan yang dilakukan oleh zombie mempunyai ciri-ciri yang sama dengan pengguna yang sedang meminta data. Ada perbedaan yang cukup terlihat pada zombie ini, yaitu gerak geriknya. Permintaan yang mencurigakan datang dari berbagai belahan dunia, sehingga tidak dapat difilter dari awalan IP mereka. Disamping itu juga banyak site yang tidak menggunakan password atau autentikasi, jika pun ada, password cukup mudah dicuri. Untuk mencegah serangan, biasanya pengguna diberikan sebuah teka-teki, tetapi untuk memproses teka-teki ini membutuhkan proses yang cukup berat, sehingga kekurangan ini dapat digunakan untuk melakukan serangan yang lain. Serangan yang paling sulit untuk dideteksi adalah serangan yang bertujuan untuk mendapatkan data yang berasal dari layer yang lebih atas, seperti CPU, database, disk karena sistem operasi yang biasa tidak menyediakan monitoring sumber-sumber tersebut secara detil.

2. TEORI DASAR

Adapun teori-teori yang digunakan saat proses pembuatan adalah sebagai berikut,

2.1 Distributed Denial of Service

Pada tahun 2000, sebuah serangan DDoS dilancarkan pada website yang terkenal, yaitu Yahoo.com, mengakibatkan website tersebut tidak dapat diakses, atau sering disebut dengan istilah down, selama kurang lebih 120 menit atau dua jam lamanya. Selain itu, serangan yang memakan waktu yang cukup lama tersebut menyebabkan kehilangannya pendapatan pada bidang periklanan mereka. Setelah kejadian tersebut, banyak serangan-serangan mulai menyerang perusahaan yang bekerja dalam bidang menyediakan sebuah jasa untuk anti-spam.

Sebuah serangan *Distributed Denial of Service* atau yang biasa dikenal dengan singkatan DDoS, adalah serangan yang terkoordinasi oleh seorang penyerang yang ingin menjatuhkan sebuah service yang ada. Serangan DDoS biasanya mempunyai skala yang cukup luas, sehingga dampak yang dikenakan pada korban cukup signifikan. Terlepasnya besar skala yang dilancarkan, DDoS sendiri adalah sebuah serangan yang terjadi karena akumulasi serangan yang dinamakan Denial of Service atau yang sering disebut dengan DoS. Serangan DoS ini sendiri hanya memakai sebuah sumber, yaitu komputer sebagai agen yang diutus untuk menyerang sebuah target serangan.

2.2 Apache Web Server

Apache adalah sebuah web server yang telah mendunia dan mempunyai catatan reputasi yang tidak dapat dianggap sebelah mata. Web server Apache ini bersifat *open source*, di mana para pemakai, sampai para pengembang yang ingin mengembangkan Apache yang mereka anggap kurang menjadi lebih baik.

Apache web server mempunyai kemampuan untuk mendukung berbagai macam fitur. Fitur-fitur tersebut diimplementasikan ke dalam sebuah modul yang telah melalui tahap kompilasi yang bertujuan untuk memperluas fungsionalitas inti Apache sendiri. Perluasan fungsi ini mulai dari pemrograman yang berbasis server atau server-side programming sampai dengan skema autentikasi sebuah request yang masuk. Apache medukung pemgromgraman yang berbasis server dengan menggunakan beberapa bahasa, yaitu Perl, Phyton, TCL dan PHP. TCL adalah sebuah bahasa pemrograman yang dikembangkan oleh John Ousterhout pada tahun 1990. TCL atau yang sering disebut dengan *tickle* sering digunakan untuk *Rapid Application Development* (RAD).

Modul-modul terkenal berbasis autentikasi yang ada pada Apache adalah mod_access, mod_auth, mod_digest dan mod_auth_digest. Selain modul-modul berbasis autentikasi ada pula modul-modul yang mendukung jalannya Apache web server, yaitu module *Secure Socket Layer* (mod_ssl), *Proxy* (mod_proxy), penggantian ulang URL (mod_rewrite), *Log* (mod_log_config) dan modul filter (mod_include dan mo_ext_filter). Metode kompresi yang digunakan pada Apache menggunakan module mod_gzip yang diimplementasikan untuk mengurangi ukuran dari halaman web site yang akan disediakan untuk pengguna. *Virtual Host* adalah salah satu fitur yang disediakan oleh Apache. Virtual Host adalah suatu metode di mana sebuah web server dapat mempunyai lebih dari satu website, contohnya adanya website www.example.com, testing.edu, www.nothing-happen.org.

Modul Apache adalah sesuatu yang dapat ditemukan dengan mudah di internet. Pengaturan Apache dapat bervariasi tergantung dengan kebutuhan dari penyedia dan pengguna itu sendiri. Selain itu, bagi para pengembang yang ingin mengembangkan modul-modulnya, Apache memberikan layanan penuh secara gratis karena modul-modul Apache bersifat *open-source*. Dengan kemudahan dan bersifat cuma-cuma, perkembangan modul-modul Apache naik secara signifikan. Perkembangan yang signifikan pada Apache akan membuat tantangan bisnis di dunia web server menjadi memanas dan membuat penyedia web server yang bersifat komersil harus selalu melangkah lebih jauh agar tidak kalah dengan Apache.

Meski Apache dirancang dengan tujuan untuk menjadi web server yang paling cepat, Apache mempunyai performa yang hampir serupa dengan web server dengan performa yang tinggi lainnya. Apache menyediakan fitur MPMs yang akan mendukung Apache untuk menjadi web server yang paling cepat. Multi-Processing Modules adalah kepanjangan dari singkatan MPMs yang mana membuat Apache dapan menjalankan *Process-Based, Hybrid Process and Thread* atau bahkan *Event-Hybrid* ke arah yang lebih baik dan menyamakan dengan permintaan setiap infrastruktur. Dengan demikian, pemilihan atas MPM dan pengaturan konfigurasi adalah sesuatu yang harus diperiksa dan dirancangkan dengan baik dan benar. Pemilihan tersebut akan berpengaruh pada beberapa aspek, yaitu pengurangan hambatan atau latensi dan menaikan angka *throughput* atau keberhasilan paket data sampai pada tujuan, melayani sebuah request yang masuk dan waktu yang dibutuhkan untuk memproses request yang masuk.

Pada Apache 2.2 muncul beberapa kelemahan yang ada pada bagian proses sebuah halaman website yang bersifat statik. Kelemahan ini dikerjakan dengan maksimal oleh web server yang bersifat open-source, yaitu *Nginx*. Tidak sama dengan pemrosesan halaman yang bersifat dinamik. Apache dapat menangani kategori ini dengan kecepata yang luar biasa. Dengan adanya kelemahan Apache tersebut, *Apache Foundation*, sebuah perkumpulan dengan beranggotakan para pengembang modul-modul Apache, mencetuskan ide multi-thread versi yang baru, yaitu menggabungkan beberapa proses dengan beberapa thread pada setiap prosesnya. Arsitektur multi-thread yang baru ini diimplementasikan dan dikokohkan pada Apache 2.4 di mana metode tersebut telah menambah performa dan mengurangi waktu yang dibutuhkan untuk memproses request-request yang masuk dan juga peningkatan performa dengan terobosan pada multi-thread mempunyai performa

yang lebih baik dari pada web server yang bersifat event-based.

2.3 Modul Evasive

Ketahanan dari sebuah serangan adalah suatu hal yang dibutuhkan sebuah server untuk bertahan dari kejahatan-kejahatan di dunia internet. Kejahatan-kejahatan tersebut tidak selalu terlihat hebat atau besar, tetapi kejahatan yang kecil pun dapat merepotkan kinerja server yang diserang, dengan demikian, proses kerja server akan berkurang dan tidak dapat melayani para clientnya secara maksimal dikarenakan adanya serangan yang bersarang di dalamnya.

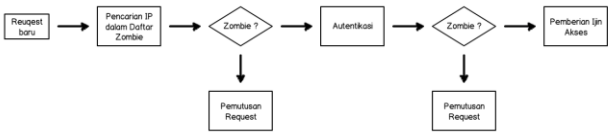
Mod_evasive adalah salah satu modul pertahanan yang dikembangkan pada Apache web server. Mod_evasive mempunyai kegunaan sebagai sebuah pertahanan untuk mengelak dari serangan HTTP DoS atau DDoS, atau serangan yang menyerang HTTP atau web yang dilakukan dengan menggunakan meotde *brute force*. Brute force adalah sebuah metode penyerangan di mana penyerang akan mencoba semua cara secara satu per satu. Mod_evasive melakukan proses deteksinya dengan pembuatan sebuah hash table yang bersifat internal di mana isi dari hash table tersebut adalah alamat ip dan URI (Uniform Resouce Identifier). URI adalah sebuah kesatuan dari URL (Uniform Resource Locator) dan URN (Uniform Resource Name). Penyimpanan alamat ip berguna untuk proses penolakan request yang dilakukan oleh sebuah alamat ip dengan sifat sebagai berikut,

- Melakukan request secara terus menerus pada sebuah halaman yang sama dalam waktu satu detik.
- Membangun lebih dari 50 request secara bersamaan pada child yang sama dalam kurun waktu satu detik.
- Melakukan request ketika tercantum pada daftar alamat ip yang tidak diijinkan masuk.

Metode ini telah bekerja secara baik pada serangan single-server script dan juga serangan yang dilakukan secara distribusi, tetapi seperti alat bantu untuk mengelak lainnya, metode ini hanya dapat berfungsi ke arah pengaturan bandwitdh dan konsumsi prosesor. Contoh konsumsi bandwitdh prosesor adalah ketika menerima, memproses, dan merespon balik kepada request yang tidak valid. Dengan demikian, pemakaian mod_evasive ini dan juga mengatur firewall dan router yang sesuai dengan data dari mod_evasive sendiri adalah sebuah tindakan yang baik.

3. DESAIN SISTEM KILL BOTS

Kill Bots mempunyai sistem yang hampir sama dengan sistem load balancer. Kalkulasi utama yang dilakukan adalah kalkulasi tentang load server. Pengertian dari load server itu sendiri adalah persentase dari berapa banyak service atau daemon (sinonim kata dari service yang sering digunakan dalam sistem operasi Linux) yang sedang bekerja. Load server adalah hal yang berbeda dengan load dari CPU atau yang sering disebut dengan CPU *usage*. Terkadang banyak yang mengartikan sama dengan hal tersebut.



Gambar 1
Flow Diagram Kill Bots

Sistem yang dipakai oleh Kill Bots adalah sebuah sistem yang cukup sederhana sehingga jika dilihat secara sepintas maka akan mudah dimengerti. Kesederhanaan dari Kill Bots mempunyai keunggulan tertentu, yaitu proses yang dipakai menjadi minimum. Proses yang minimum dan cepat itulah inti dari pengembangan web server untuk bekerja lebih efisien dan cepat dalam menangani request-request yang masuk.

Gambar 1 menggambarkan tentang alur kerja dari sistem Kill Bots dengan permulaan proses adalah dengan pencarian pada daftar zombie di mana pencarian didasarkan oleh alamat IP dari request yang masuk. Daftar zombie merupakan gabungan dari dua buah komponen, yaitu filter yang digunakan untuk mengurangi proses pencarian yang langsung diproses dalam sebuah tempat penyimpanan, dalam tugas akhir ini filter yang akan digunakan adalah *Bloom Filter*, dan tempat penyimpanan itu sendiri yang dalam tugas akhir ini tempat penyimpanan yang dipakai adalah hash table yang telah disediakan oleh Apache. Alamat IP yang masuk dan akan dicari dalam daftar zombie tidak dengan begitu saja dicari dalam daftar zombie, tetapi sebelumnya, proses *hashing* akan dilakukan pada alamat IP tersebut dan kemudian dilakukan prosedur pencarian dalam daftar zombie tersebut. Proses hashing tersebut menandakan bahwa data-data yang ada pada daftar zombie tersebut ada data hasil proses hashing yang mulanya adalah sebuah alamat IP. Jika alamat IP yang masuk ternyata ada dalam daftar zombie, maka server akan mengirimkan sebuah sinyal di mana request tersebut tidak dapat diteruskan ke tujuan request tersebut atau dalam bahasa singkatnya, request tersebut akan dihentikan untuk mendapatkan tujuan semulanya. Penghentian request tersebut tidak semudah memotoh sebuah tali. Dalam lifecycle dari Apache, tidak adanya prosedur yang dapat memotong request yang masuk, tetapi Apache mempunyai prosedurnya sendiri, yaitu mengirimkan sinyal kepada setiap fasenya untuk tidak memproses request tersebut lebih lanjut dan terakhir pada bagian content handler, sinyal tersebut akan mengembalikan sebuah nilai pada sumber request sebuah kode status html, contohnya kode status 4xx. Kode status 4xx adalah kode status mempunyai arti client error, di mana kesalahan ada pada client. Pada tugas akhir ini kode staus untuk menghentikan request agar tidak diproses lebih lanjut adalah 403, yaitu *Forbidden*.

Pada proses selanjutnya, yaitu akan dilakukan proses autentikasi terhadap request yang masuk tersebut. Autentikasi adalah nama sebuah proses yang mana di dalam proses tersebut terdapat beberapa proses kecil yang bertujuan untuk mengidentifikasi apakah request tersebut

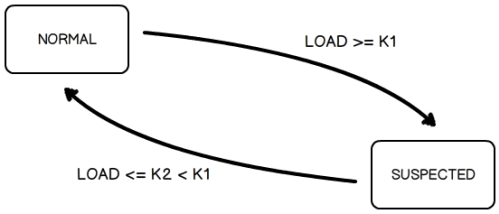
layak untuk mendapatkan ijin untuk melakukan prosesnya secara normal atau tidak. Jika tidak, sama seperti sebelumnya, akan dilakukan prosedur untuk menutup request tersebut dengan tanpa melakukan proses lebih lanjut pada request tersebut.

3.1 Autentikasi

Sistem Kill Bots menggunakan proses autentikasi setiap request yang masuk ketika sebuah server dalam status siaga, atau dalam sistem ini dinamakan mode *SUSPECTED*. Dalam Kill Bots sendiri terdapat dua buah mode yang menandakan status dari server dan masing-masing mempunyai proses yang berbeda. Seperti yang telah dikatakan salah satunya, mode tersebut adalah SUSPECTED dan NORMAL. Proses pembagian status server ini dimasukkan pada sebuah bagian sendiri, yaitu bagian *Stages*. Stages ini adalah proses yang dapat dikatakan paling penting dalam autentikasi serangan DDoS ini.

3.2 Stages

Proses stages ini mengembalikan dua buah status yang mana status ini akan menandakan dari performa server. Kedua status tersebut adalah SUSPECTED dan NORMAL. Sesuai dengan namanya, NORMAL adalah status server yang menandakan bahwa server berada pada performa yang baik di mana server dapat melayani request-request yang datang secara normal dan tidak ada kendala apa pun. Sebaliknya, pada status server SUSPECTED menandakan bahwa server mempunyai kondisi yang sangat sibuk sehingga untuk kedepannya mungkin akan ada kesalahan proses atau koneksi yang terbentuk. Setelah server menandakan bahwa sekarang adalah berstatus SUSPECTED, maka akan diikuti serangkaian proses untuk mengambil alih request-request yang datang.



Gambar 2
Transisi Status Server

Proses pembedaan status server di sini menggunakan sebuah fungsi yang mana menggunakan kalkulasi sederhana. Kalkulasi yang digunakan berdasarkan oleh load dari server. Load server ini berbeda dengan CPU usage yang ada dalam komputer. Kadang, beberapa sumber mengatakan bahwa load server sama dengan load CPU atau CPU usage, dalam kasus ini, hal tersebut cukup berbeda. Load server sendiri adalah kumpulan service atau daemon yang tersisa atau kosong dalam waktu tersebut.

Contohnya, jika sebuah server mempunyai jumlah maksimal daemon untuk melayani request-request yang datang sebesar 256, pada saat kalkulasi sedang dilakukan, terdapat 100 request yang masuk dan sedang diproses, dengan demikian menyisakan 156 daemon dalam keadaan *idle ready*. Load server yang sekarang didapatkan dari kalkulasi pembagian antara daemon yang sedang melakukan proses terhadap request yang datang dengan jumlah maksimal daemon yang dapat menangani request dalam server, dengan demikian didapatkan nilai 0.39, dan load server merupakan nilai presentase, maka nilai tersebut dikalikan 100, sehingga didapatkan nilai 39% untuk load server.

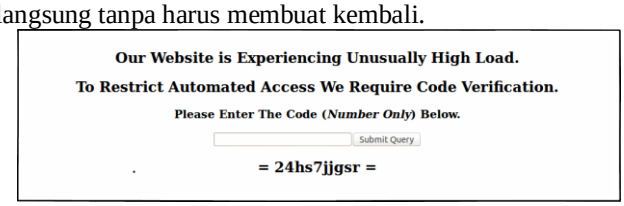
Pada awal mula server dijalankan, server akan berstatus NORMAL dengan tingkat load yang sangat rendah atau dapat dikatakan hampir nol karena pada kondisi tersebut server baru saja dijalankan. Setelah beberapa lama, dengan meningkatnya load dari server sehingga load dapat melebihi batas yang telah diberikan, yaitu K1, maka status server akan terganti dengan status SUSPECTED.

Untuk kembali ke status NORMAL, load harus mempunyai nilai kurang dari K2, di mana K2 bernilai lebih rendah dari pada K1. Gambar 2 menggambarkan transisi dari status server. Dalam tugas akhir ini, penilaian variabel K1 dan K2 adalah berdasar percobaan dan logika dasar. Nilai untuk variabel K1 adalah sebesar 70% dan untuk K2 adalah sebesar 50%.

Jika penyerang mempunyai ide untuk mengganti status server ini terus menerus, dari NORMAL ke SUSPECTED dan sebaliknya, proses tersebut tidaklah menguntungkan sama sekali, karena penentuan status tersebut memakan sumber daya yang sangat kecil sekali, hanya ada pengambilan load server saat ini dan akhirnya ditunjuk status mana yang cocok untuk kondisi load yang telah didapat.

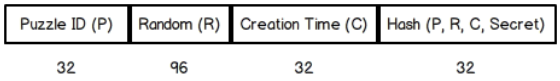
3.2.1 Stage SUSPECTED: CAPTCHA-Based Authentication

Setiap request yang masuk ke dalam server ketika status server adalah SUSPECTED, akan diberikan sebuah *graphical test* atau *captcha* di mana pengguna harus menjawab pertanyaan dari tes tersebut untuk melanjutkan proses untuk mendapatkan data dari server secara normal kembali. Pemberian captcha ini melepaskan pengguna dari perlakuan yang langsung ke arah data dari server, sehingga dalam hal ini server belum melemparkan data apa pun yang ada. Captcha itu sendiri adalah hasil pembuatan dari Kill Bots. Pembuatan captcha ini berlangsung ketika server sedang dijalankan untuk pertama kali atau nama lain dari proses ini adalah start-up, maka ketika start-up server telah selesai, captcha akan sudah siap untuk diakses secara langsung tanpa harus membuat kembali.



Gambar 3
Contoh Captcha Kill Bots

Jika sebuah pengguna yang sah sedang mencoba untuk mengakses server, di mana server sedang berada pada status SUSPECTED, kemudian pengguna tersebut dapat menjawab captcha yang telah diberikan, akan tidak baik jika pengguna yang telah menjawab dengan benar sebuah tes yang diberikan dan setelah itu pengguna tersebut mencoba mengakses server untuk kedua kalinya, dan diminta untuk menjawab sebuah captcha kembali. Kill Bots menggunakan sistem *one test per session*. Sistem tersebut memberikan kelonggaran pada pengguna yang telah dapat menjawab tes yang diberikan kepada mereka dengan memberi sebuah token yang berupa sebuah cookie yang menandakan bahwa pengguna tersebut dapat mengakses server tanpa harus diberikan tes sebelumnya. Perijinan tersebut mempunyai batas waktu yang telah ditentukan. Dalam tugas akhir ini perijinan tersebut bernilai 30 menit.



Gambar 4
Segmen Token

Berbicara tentang perijinan, berarti menandakan bahwa adanya pengamanan data agar data tersebut tidak dapat dibuat ulang dari bagian pengguna. Token yang mana menandakan perijinan tersebut terbagi menjadi empat bagian, seperti yang telah digambarkan pada gambar 3, yaitu segmen token. Terlihat bahwa segmen pertama pada token adalah *puzzle id*. Puzzle id ini merepresentasikan puzzle yang diberikan kepada pengguna untuk dijawab. Sebelah kanan dari puzzle id adalah sebuah rangkaian karakter *random*. Rangkaian karakter random pada Kill Bots ini berupa karakter angka. Selanjutnya, pada kolom ketiga terdapat *creation time* yang menandakan kapan token ini dibuat. Kolom terakhir, yaitu kolom yang memerlukan ketiga segmen sebelumnya dan sebuah *secret key* yang ada pada server. Kolom ini akan membungkus keempat komponen tersebut dengan menggunakan fungsi RS hash. RS hash akan mengembalikan nilai berupa rangkaian angka dan digabungkan dengan ketiga komponen sebelumnya dan terbentuklah sebuah token yang berfungsi menjadi sebuah perijinan agar pengguna yang dapat menjawab tes yang diberikan tidak mendapat tes kembali.

Pengguna yang mengakses server dan dalam akses tersebut ada token yang dimaksudkan, maka server akan memproses token tersebut terlebih dahulu sebelum ada proses yang lebih lanjut ke dalam server. Proses yang pertama kali dilakukan adalah memeriksa apakah hash yang ada pada token tersebut sama dengan hash yang dikalkukasi dengan berdasarkan komponen-komponen dalam token tersebut dan komponen secret key yang ada dalam server. Setelah cocok, maka pemeriksaan berlanjut pada creation time. Pada proses ini terdapat dua buah alur. Sebelum berlanjut kepada penjelasan alur, perlu diketahui bahwa token diselipkan pada captcha yang diberikan kepada pengguna. Alur pertama, yaitu ketika pengguna menjawab sebuah captcha. Pengguna diberikan waktu sebanyak empat menit untuk menjawab captcha yang telah diberikan. Jika pengguna menjawab lebih dari waktu yang diberikan, meski pun jawaban yang diberikan tepat, pengguna tersebut akan diberikan captcha dengan isi yang berbeda. Berlanjut pada alur yang kedua, yang mana alur kedua ini telah dijelaskan sebelumnya. Alur kedua ini adalah ketika pengguna telah menjawab benar tes yang diberikan. Pada tahap ini token tersebut tersimpan pada cookie pengguna. Pengguna yang telah menjawab benar tes, berhak mengakses secara bebas server tersebut tanpa harus diberikan tes ulang. Kebebasan ini mempunyai batas waktu tertentu, yaitu 30 menit. Terlepas dari waktu tersebut, pengguna akan diberikan tes ulang ketika mengakses server untuk lebih lanjut.

3.2.2 Stage SUSPTECTED: Pengguna yang Tidak Menjawab CAPTCHA

Mekanisme yang menitik beratkan pada captcha, mempunyai dua buah kerugian. Pertama, penyerang akan membuat server untuk terus memberikan captcha agar server selalu dalam keadaan sibuk. Kedua dan juga poin yang cukup penting adalah bagi para pengguna yang tidak dapat menjawab captcha yang diberikan, maka pengguna tersebut tidak dapat mengakses server sama sekali.

Poin-poin kerugian tersebut dapat diselesaikan dengan mengidentidikasi perilaku dari kedua belah pihak ini. Pada pihak pengguna, terdapat dua poin yang sering mereka lakukan pada kondisi mereka tidak dapat menjawab captcha tersebut. Poin pertama, pengguna tersebut akan menjawab captcha yang diberikan setelah beberapa kali me-reload halaman web yang berisi tes tersebut. Pada poin kedua, pengguna menyerah dalam menjawab tes yang diberikan dan meninggalkan tes tersebut dalam kurun waktu tertentu. Perilaku ini terjadi dikarenakan pengguna tersebut telah menjoba untuk menjawab tes yang diberikan beberapa kali tetapi jawaban yang diberikan tersebut adalah salah. Pada perilaku yang dilakukan oleh penyerang juga mempunyai dua buah poin yang dapat menandakan bahwa perilaku tersebut adalah perilaku dari penyerang. Pertama, penyerang mungkin akan mengikuti perilaku dari pengguna pada poin pertama dan meninggalkan tes tersebut setelah beberapa kali mencoba, dalam poin ini, tidak adanya perilaku yang mencurigakan, karena ketika penyerang meninggalkan tes, maka penyerang tersebut menandakan bahwa serangan telah dihentikan. Poin kedua, penyerang akan mengirimkan request terus menerus, meski pun mereka tidak dapat menjawab tes yang diberikan. Berdasarkan poin kedua yang ada pada pihak penyerang, membuat penyerang sangat terlihat bahwa mereka mempunyai keinginan untuk menyerang server.

Sebuah batasan diberikan dengan dasar dari poin-poin yang telah didapatkan dari kedua belah pihak. Batasan tersebut memerlukan bantuan dari bloom filter dan hash table untuk menanganinya. Kesalahan dalam proses menjawab sebuah tes akan dicatat kedalam hash table secara terus menerus. Pemakaian bloom filter dalam kondisi ini adalah untuk mempercepat proses dan juga meminimalisasikan proses yang harus dikerjakan. Setiap

request yang masuk, berdasarkan alamat IP mereka akan diberikan sejumlah kesempatan atas kesalahan menjawab. Ketika alamat IP yang mengirim request melebihi batas kesalahan yang diberikan, maka request untuk alamat IP tersebut akan dihentikan, dan tidak adanya tes yang diberikan. Hash table dan bloom filter akan dikosongkan ketika server masuk kepada status NORMAL, di mana server dapat bekerja melayani request-request yang masuk dengan maksimal.

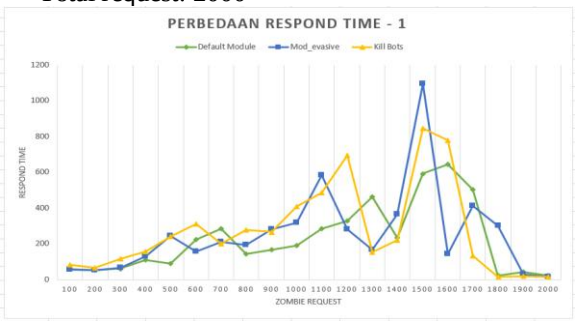
4. UJI COBA PERFORMA PADA SERANGAN FLASH CROWD

Terdapat tiga buah kondisi pengujian dalam uji coba performa ini. Kondisi pertama server dengan default module, yang dimaksud dengan default module adalah tidak adanya penambahan modul lainnya pada server tersebut, server akan memuat modul-modul standar dari server itu sendiri. Kondisi kedua adalah kondisi di mana server akan diberi modul tambahan sebagai penganangan serangan DDoS, modul tambahan yang dipakai adalah mod_evasive. Terakhir adalah kondisi di mana server akan diberi modul yang telah diimplementasikan sistem Kill Bots di dalamnya. Terdapat empat skenario yang akan dilakukan terhadap ketiga kondisi tersebut. Setiap scenario akan dilakukan, server akan dijalankan ulang agar mendapat performa yang sama setiap skenarionya. Dalam pengujian ini, hasil kembalian yang akan diperhitungkan adalah respond time dari server ketika sebuah request masuk ketika gelombang serangan sedang dilancarkan kepada server.

4.1 Skenario 1

Dalam skenario pengujian performa server ini akan digunakannya sepuluh unit komputer sebagai agent atau worker aktif di mana kesepuluh komputer ini akan melakukan request terus menerus kepada server dengan parameter jumlah thread per komputer dan berapa kali komputer-komputer tersebut melakukan request per threadnya sesuai dengan skenario yang akan dijalankan. Pada skenario pertama ini, parameter-parameter yang akan disiapkan adalah sebagai berikut,

- Jumlah komputer: 10
- Thread per komputer: 20
- Request per thread: 10
- Total request: 2000



Gambar 5
Hasil Uji Coba Skenario 1

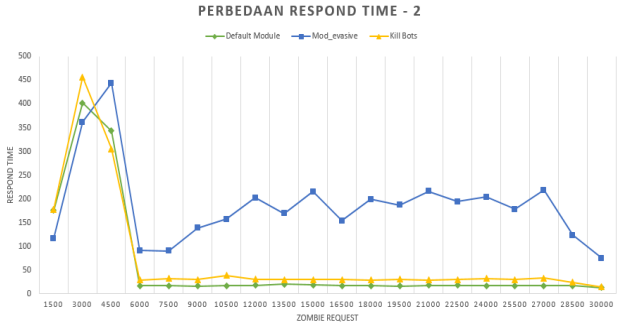
Pada gambar 5 dapat dilihat hasil pengujian performa dengan menggunakan skenario pertama. Pengujian dengan menggunakan skenario pertama ini tidaklah mendapatkan hasil yang memperlihatkan bahwa antara default module, mod_evasive, dan kill bots mempunyai perbedaan yang besar. Skenario pertama dirancang agar request yang dilancarkan bersifat lingkungan server yang normal. Meski hasil pengujian performa dari Kill Bots terlihat mempunyai respond time yang lebih tinggi dari default module, tetapi jarak perbedaan tersebut tidaklah tinggi dan cukup mendekati respond time dari default module. Begitu juga dengan server yang diberikan modul mod_evasive sebagai modul tambahan.

Berdasarkan data parameter yang diberikan, dengan delay per request yang dikirimkan adalah sebesar satu detik, maka akan didapatkan data rate request per dtiknya adalah 200 request. Kill Bots memiliki respond time yang lebih tinggi karena adanya proses autentikasi yang berjalan di dalamnya, sehingga menaikan waktu yang diperlukan untuk mengembalikan respon dari request yang diterima.

4.2 Skenario 2

Setelah melakukan pengujian berdasarkan lingkugan server yang cukup normal, kemudian pengujian berlanjut dengan menambahkan angka pada parameter request per threadnya. Dengan penambahan angka pada parameter request per thread ini, diharapkan adanya hasil yang cukup terlihat dari perbandingan ketiga kondisi tersebut. Berikut rincian dari parameter-parameter yang diberikan pada skenarion kedua,

- Jumlah komputer: 10
- Thread per komputer: 20
- Request per thread: 150
- Total request: 30000



Gambar 6
Hasil Uji Coba Skenario 2

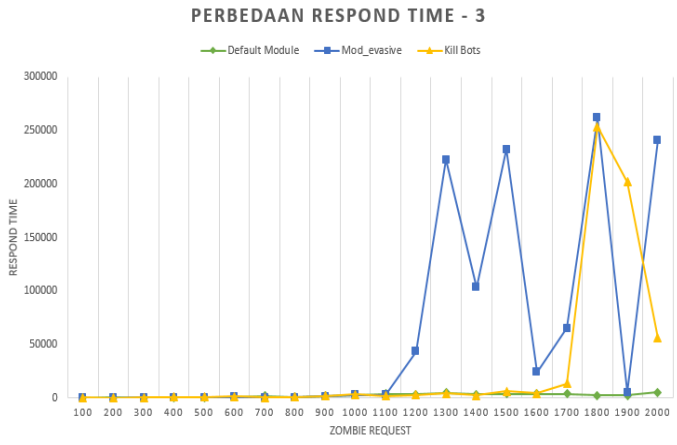
Hasil yang cukup menakjubkan terlihat pada gambar 6 yang telah disajikan. Harapan untuk mendapatkan hasil yang memperlihatkan perbedaan dari ketiga kondisi ini pun sedikit tercapai. Hasil skenario kedua ini memperlihatkan bahwa dalam rate request yang sama seperti skenario pertama, tetapi dengan total request yang berbeda, mempunyai dampak yang cukup terlihat pada kondisi server diberi modul tambahan berupa mod_evasive. Dalam skenario ini tampak sangat jelas mod_evasive memberikan

respond time yang sangat jauh berbeda dengan kondisi default module. Berbeda dengan kondisi di mana server diberi modul tambahan berupa implementasi dari Kill Bots. Sistem Kill Bots dapat menjaga jarak performanya dengan server yang tidak diberikan modul tambahan apa pun. Seperti yang sebelumnya telah dijelaskan, alasan kenapa Kill Bots mempunyai respond time di atas default module adalah adanya tambahan proses transisi status pada server di mana proses yang berlangsung adalah pengambilan data load server saat itu juga. Perbedaan sekecil itu tidaklah terasa, karena perbedaannya tidak lebih dari 50 milisecond, dengan demikian, performa dari Kill Bots dapat dikatakan sama dengan performa server tanpa adanya modul tambahan pada rate request 200 request per detik. Berdasarkan hasil pengujian performa diantara ketiga kondisi tersebut, dan dengan perbedaan yang sangat terlihat, kra-kira hampir 100milisecond, mod_evasive telah mulai mengundurkan posisinya dari kedua kondisi yang masih bersaing antar satu sama lain.

4.3 Skenario 3

Pada skenario ketiga ini ada sedikit perbedaan dari dua skenario sebelumnya, di mana dua skenario sebelumnya memainkan angka request per threadnya, dalam skenario ketiga ini, parameter yang akan dimainkan adalah thread per komputernya. Pada sebelumnya, dengan perbedaan request per threadnya, rate request dari skenarionya tidak berubah sama sekali. Berbeda dengan skenario ini, dengan angka yang diubah adalah thread per computernya dan request per threadnya dikurangi sehingga menjadi cukup kecil. Dengan demikian diharapkan dengan berbedanya rate request yang diberikan, hasil yang didapat pun akan beragam, sehingga dapat memberikan pengetahuan tentang keunggulan dari sistem Kill Bots. Berikut adalah rincian parameter yang diberikan pada skenario ketiga ini,

- Jumlah komputer: 10
- Thread per komputer: 100
- Request per thread: 2
- Total request: 2000



Gambar 7
Hasil Uji Coba Skenario 3

Hasil uji coba dengan skenario ketia ini memberikan kejutan yang cukup besar. Dapat dilihat pada gambar 7, kondisi default module berlenggang dengan sangat mudahnya dengan nilai respond time dari uji coba performanya hampir mendekati nilai nol dibadingkan dengan dengan kondisi-kondisi yang lain. Pada awal skenario sampai dengan angka 1700 request, Kill Bots masih dapat mengikuti performa dari default module. Kejadian yang cukup mengherankan terjadi pada angka lebih dari 1700 request. Tiba-tiba performa dari Kill Bots melambung tinggi sampai dengan angka 250000. Untuk kondisi mod_evasive mulai terlihat kekurangannya pada angka awal, yaitu angka 1100 request.

Dengan hasil pada skenario ketiga ini, mod_evasive menurunkan kualitasnya sebagai modul penanganan serangan DDoS untuk Apache web server. Sampai dengan akhir pengerjaan tugas akhir ini, belum diketahuinya penyebab tentang melambungnya respond time pada saat di mana akhir skenario tercapai. Dengan hasil yang terlihat pada skenario ketiga ini, dapat dinilai bahwa Kill Bots mempunyai kualitas yang sedikit dibawah ddefault modul pada skenario dengan rate request 1000 request per detik.

4.4 Skenario 4

Skenario keempat ini adalah skenario yang diambil dengan menggunakan pergantian thread per komputernya. Dengan mengganti thread per komputer, maka rate requestnya pun akan berbeda. Berbeda dengan skenario pertama dan skenario yang kedua, di mana perbedaan keduanya hanya pada berapa kali thread itu mengakses server bukan perbedaan berdasarkan rate request.

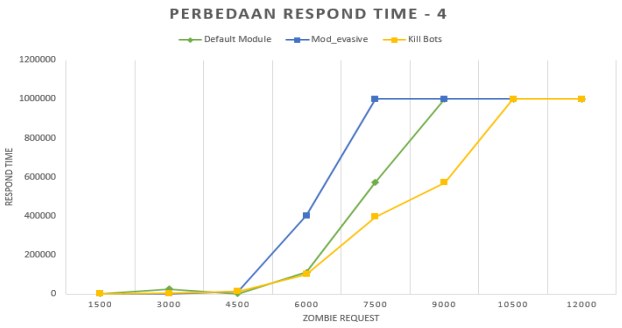
Dalam skenario keempat ini, rate request yang akan diujikan mempunyai perbedaan yang sangat jauh bila dibandingkan denan skenario sebelumnya, yaitu skenario ketiga. Perbadingannya hampir 15 kali lebih tinggi dari pada rate request dari skenario ketiga. Jika sebelumnya rate requestnya bernilai 1000 request per detik, pada skenario keempat ini akan dinaikkan nilai rate requestnya sampai dengan angka 15000 request per detik.

Pada skenario keempat ini parameter yang digunakan akan dirancang berdasarkan lingkungan dari serangan DDoS Flash Corwd. Serangan Flash Crowd sendiri memberikan kondisi lingkungan yang sangat berat kepada server sehingga server diharapkan dapat memberikan hasil server tidak berkutik atau tidak merespon request yang masuk dari mana pun dan isi request apa pun. Dengan jumlah rate request per detiknya yang mempunyai nilai yang sangat tinggi, karena sangat tingginya rate request yang dilakukan, server akan menemui keadaan di mana server tidak dapat merespon request apa pun.

Ketika kondisi server tidak dapat merespon apa pun, data yang diambil seketika itu juga dihentikan, sehingga dapat dipetakan perbandingan antar kondisi yang diuji performanya. Berikut adalah rincian parameter yang digunakan dalam skenario keempat ini,

- Jumlah komputer: 10
- Thread per komputer: 1500

- Request per thread: 2
- Total request: 30000



Gambar 7.6
Hasil Uji Coba Skenario 4

Hasil uji coba skenario keempat yang harusnya mempunyai data sampai dengan 30000 request dipotong sampai dengan nilai 12000, karena pada angka lebih dari 12000 server telah menunjukkan bahwa server tidak dapat diakses atau server sedang tidak dapat merespon request-request yang masuk. Pada sebelumnya, server yang diberikan modul dengan sistem Kill Bots di dalamnya menunjukan performa yang kurang dari default modul, pada skenario keempat ini Kill Bots mulai menunjukkan keunggulannya. Nilai 100000 pada bagian respond time menunjukan bahwa server tidak dapat diakses.

Kondisi pertama yang menunjukan server tidak dapat merespon request adalah mod_evasive pada nilai request 7500, dilanjutkan dengan default module pada angka 9000 request, dan yang terakhir adalah Kill Bots pada angka 10500 request. Pada gambar 8 juga dapat dilihat bahwa kenaikan respond time dari Kill Bots tidak secara langsung menanjak ke angka 100000, tetapi bertahap sehingga lebih banyak request yang dapat dilayani sebelum server sampai pada batasnya, sehingga tidak dapat merespon request-request yang masuk. Pada rate request yang sangat tinggi ini, Kill Bots menunjukkan kemampuannya dalam menangani serangan DDoS Flash Crowd.

5. KESIMPULAN

Dari proses pengerjaan Tugas Akhir ini diperoleh beberapa kesimpulan yang akan dijelaskan sebagai berikut,

1. Apache merupakan web server yang berdasarkan keintegritasan dari modul-modulnya membuat pengembangan modul-modulnya lebih mudah untuk dipelajari. Arti dari kemudahan pengembangan modul-modulnya adalah tertatanya proses-proses apa saja yang dapat dikembangkan, sehingga ketika melakukan pengembangan modul Apache, tidak sulit untuk mencari proses atau tahapan apa yang harus dikembangkan.
2. Keintegritasan dari modul-modul Apache adalah berkat kerja dari proses MPM atau Multi Processing Modules dari Apache. MPM mengelompokkan proses-proses pada tahap yang sama dan menjalankannya pada secara bersamaan sesuai dengan prioritas-prioritas yang telah diberikan pada modul tersebut, dan dengan proses tersebut membuat proses kerja Apache berjalan lebih cepat.
3. MPM juga mempunyai kekurangan atas proses kerjanya. Kekurangan dari MPM adalah pelepasan pool yang tidak dipakai secara berkala, sehingga membuat pembentukan tempat penyimpanan yang bersifat dinamik cukup sukar.
4. Pada serangan Distributed Denial of Service dengan tipe Flash Crowd, adanya perilaku yang dapat membedakan apakah itu zombie atau pengguna yang sah. Perbedaan dari kedua pihak tersebut dapat dilihat pada perilaku mereka terhadap tes yang diberikan kepada mereka.
5. Jika adanya rencana untuk membuat sebuah penyimpanan data yang bersifat statik, maka alokasi dari data tersebut harus diletakkan pada configuration pool, di mana untuk mengakses configuration pool dapat dilakukan ketika server pertama kali dijalankan atau sedang melakukan proses start-up.
6. Sebelum mulai berkecimpung dalam Apache Module Development, ada baiknya untuk mengaji tentang perilaku Apache Module lifecycle dan MPM terlebih dahulu, karena kedua komponen tersebut adalah sesuatu yang sangat penting dalam mengembangkan sebuah modul untuk Apache web server.
7. Bloom filter dapat mengikuti kecepatan proses server, dengan demikian menandakan bahwa proses pencarian kata kunci dalam Bloom filter mempunyai kecepatan yang sangat cepat.
8. Kill Bots mempunyai kemampuan untuk melayani jumlah request yang lebih banyak dari pada server dengan tanpa modul pertahanan dan server yang diberi pertahanan mod_evasive.
9. Performa Kill Bots dalam lingkungan rate request yang rendah, terkadang terjadi anomali yang membuat respond time dari server meningkat cukup tinggi. Kenaikan respond time hanya berlangsung sekali saja, setelah kenaikan tersebut, respond time server menurun kembali.

6. SARAN

Tidak semua proses dalam pembuatan Tugas Akhir berjalan sesuai keinginan pembuat. Masih ada beberapa kekurangan yang terdapat dalam pembuatan Tugas Akhir ini. Berikut adalah saran yang dapat dipakai untuk pengembangan lebih lanjut.

1. Adanya penelitian lebih lanjut tentang Bloom filter yang dapat menyimpan data tidak hanya keynya saja, tetapi juga dapat menyimpan value dari key tersebut. Bloom filter yang dapat melakukan hal tersebut adalah Invertible Bloom filter.
2. Untuk menaikkan kualitas proses autentikasi Kill Bots, sebuah pemeriksaan berdasar performa server sebelumnya dapat diimplementasikan. Proses pemeriksaan tersebut dapat disebut dengan proses admission control, di mana acuan performa server dapat diambil dengan metode microbench di mana pengukuran performa server dapat diukur sampai nilai yang cukup kecil, yaitu sapai satuan microsecond.

REFERENSI

[1] Bloom, *B. Space/time Trade-offs in Hash Coding with Allowable Errors*. Communications of the ACM, 13 (7). 422-426.

[2] CERT. *Incident Note IN-2004-01 W32/Novarg.A Virus*, 2004.

[3] J. Leyden. *East European Gangs in Online Protection Racket*, 2003. www.theregister.co.uk/2003/11/12/east european gangs in online/.

[4] K. Poulsen. *FBI Busts Alleged DDoS Ma_a*, 2004. http://www.securityfocus.com/news/9411.

[5] http://billmill.org/bloomfilter-tutorial/.

[6] Goel dan P. Gupta. *Small Subset Queries and Bloom Filters Using Ternary Associative Memories, with Applications*. ACM SIGMETRICS, 2010.

[7] http://en.wikipedia.org/wiki/Bloom_filter.

[8] Dan Gookin. *C for Dummies 2nd Edition*. 2004.

[9] Specht., Stephen M. dan Lee., Ruby B. *Distributed Denial of Service: Taxonomies of Attack, Tools, and Countermeasure*. Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems, 2004 International Workshop on Security in Parallel and Distributed Systems, pp. 543-550, September 2004

[10] Kandula., Srikanth. Katabi., Dina. Jacob., Matthias. Berger., Arthur. *Botz-4-Sale: Surviving Organized DDoS Attack That Mimic Flash Crowds*. 2005.

[11] Jung., Jaeyeon. Krishnamurthy., Balachander. Rabinovich., Michael. *Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites*. Mei 2002.

[12] Kew., Nick. *The Apache Modules Book*. Januari 2007.