ANALISIS PERFORMANSI VIRTUALISASI CONTAINER MENGGUNAKAN DOCKER DIBAWAH SERANGAN NETWORKED DENIAL OF SERVICE

PERFORMANCE ANALYSIS OF CONTAINER VIRTUALIZATION USING DOCKER UNDER NETWORKED DENIAL OF SERVICE ATTACKS

Chrisna Fiddin[1], Dr. Rendy Munadi, Ir.,M.T. [2], Ratna Mayasari, S.T, M.T.[3]
[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
chrisnafc@gmail.com[1] , rendymunadi@telkomuniversity.ac.id[2] , ratnamayasari@telkomuniversity.ac.id[3]

Abstrak

Penggunaan teknologi virtualisasi dalam menyediakan infrastruktur komputasi meningkat cukup tinggi seiring berkembangnya dunia industri IT. Penerapan teknologi virtualisasi dan *container* merupakan bagian yang sangat penting dalam penerapan *cloud computing*, karena sangat berpengaruh pada efisiensi pengelolaan sumber daya infrastruktur *cloud computing.* Pada konsep virtualisasi, dibutuhkan sebuah sistem yang dapat menyediakan teknik virtualisasi yang mampu menyediakan sebuah sistem dengan peformansi mendekati atau sama dengan native. salah satu faktor yang memperngaruhi performansi sebuah teknik virtualisasi yaitu dari sisi keamanannya. Terdapat banyak jenis serangan terhadap sistem computer, yaitu salah satunya Denial of service, jenis serangan ini senantiasa berkembang dalam berbagai bentuk.

Pada tugas akhir ini akan dilakukan penelitian mengenai performansi *container* menggunakan platform Docker pada saat mengalami serangan DoS TCP SYN. Penelitian ini bertujuan untuk menganalisa kinerja mesin dari sisi overall performance, dan layanan web server, yang dijalankan di atas container pada saat keadaan normal dan mendapatkan serangan. Hasil pengujian akan dianalisa performansinya dengan membandingkan mesin native.

Dari hasil pengujian didapat bahwa serangan Denial of service memberikan dampak penurunan performansi dari sisi overall performance dan layanan web server, baik mesin native maupun menggunakan teknik virtualisasi container Docker. Pada serangan yang relatif ringan didapatkan hasil bahwa pada parameter request per second menyebabkan penurunan performansi sebesar 40,22% pada native dan 37,65% pada Docker, sedangkan pada parameter response time web server, serangan DoS menyebabkan peningkatan response time pada server native sebesar 40,80%, sedangkan pada Docker sebesar 38,38.

Kata kunci : Keamanan Jaringan, Virtualisasi, *Container.*

Abstract

The application of virtualization technology in providing computing infrastructure has increased considerably in line with the development of the IT industry. Implementation of virtualization and container technology is very influential on the efficiency of cloud computing infrastructure resource management. On the concept of virtualization, a system that can provide a virtualization technique that is capable of providing a system with performance close to or equal to the native is required. One of the factors that affect the performance of a virtualization technique is security. There are many types of attacks on computer systems, one of them is Denial of service, this type of attack is constantly evolving in various forms.

In this final project, a research will be conducted on container performance using Docker platform under DoS SYN flood attack. This study aims to analyze the performance of machines in terms of overall performance, and web services performance, which run on top of the container during normal state and under attack. The test results will be analyzed by comparing the performance native machine.

From the test results discovered that Denial of service attacks have the impact of performance degradation on overall performance and web services, both native machines and using Docker container virtualization techniques. In the relatively light attack, the results show that the request per second parameter resulted in a decrease in performance of 40,22% in native and 37,65% in the Docker, whereas in the response time parameter of the web server, DoS attacks resulted in increased response time on the native server by 40,80%, while at the Docker by 38,38%.

Keywords: Network security, Virtualization, Container

1. Pendahuluan

Teknologi virtualisasi pada komputasi sekarang menjadi topik yang cukup ramai dalam pembahasan riset teknologi *cloud*. Virtualisasi dan container merupakan bagian yang sangat penting dalam penerapan *cloud computing*, karena sangat berpengaruh pada efisiensi pengelolaan sumber daya infrastruktur *cloud computing*. Hasil survey dari sdxcentral yang dilakukan pada tahun 2015 menunjukan bahwa 94% responden yang berasal dari kalangan perusahaan telah mengadopsi teknologi container sejak 12 bulan terakhir[1]. Dalam implementasinya saat ini, beberapa teknologi virtualisasi masih memerlukan penelitian maupun percobaan lebih lanjut karena beberapa teknologinya masih tergolong baru. Salah satu aspek penelitian yang banyak dikaji adalah mengenai jalannya aplikasi di atas teknologi virtualisasi serta aspek keamanan jaringan. Di antara berbagai serangan DDOS, seranga flood TCP SYN adalah yang paling umum dan dikenal sebagai salah satu metode DoS yang paling kuat. Jenis serangan ini masih mendominasi serangan DDOS berdasarkan laporan Kaspersky tahun 2016 [2].

Sebelumnya telah dilakukan penelitian oleh Ryan Shea, Jiangchuan Liu dan dipublikasi pada *IEEE SYSTEMS JOURNAL* dengan judul "*Performance of Virtual Machines Under Networked Denial of Service Attacks: Experiments and Analysis*" , hasil dari parameter connection time diketahui bahwa performa hypervisor ketika mendapatkan serangan DoS terjadi penurunan yang cukup besar, persentase perbandingannya yaitu diatas 400%, sedangkan pada container Open Vz terdapat penurunan performa hanya sebesar 12%. Dari hasil penelitian tersebut disimpulkan bahwa, ada kemungkinan dengan menggunakan teknologi container serangan DoS tidak membuat performa mesin menurun secara signifikan. Karena pada konsep virtualisasi dibutuhkan sebuah sistem yang dapat menyediakan teknik virtualisasi dengan performa mendekati atau sama dengan native [7].

Pada penelitian yang lainnya mengenai analisis performansi antara virtual machine dan container pada penelitian yang telah dilakukan oleh Rabindra K. Barik, Rakesh K. Lenka, K. Rahul Rao, Devam Ghose dalam paper yang berjudul "Performance Analysis of Virtual Machines and Containers in Cloud Computing" [5] mendapatkan kesimpulan bahwa performansi virtualisasi container menggunakan Docker sedikit lebih unggul jika dibandingkan dengan teknik virtualisasi lainnya.

Berdasar latar belakang yang sudah dijabarkan dan kesimpulan yang didapat dari penelitian terkait sebelumnya, maka akan dilakukan penelitian pada performansi teknik virtualisasi container, dengan tujuan untuk menganalisa dan mengetahui dampak yang ditimbulkan oleh serangan Denial of Service (DoS) SYN attack terhadap overall performance dan layanan web yang dijalankan menggunakan teknik virtualisasi container dengan menggunakan platform Docker.

2. DASAR TEORI

2.1 VIRTUALISASI CONTAINER

Virtualisasi adalah teknik membuat versi abstrak atau virtual dari suatu sumber daya, sehingga pada satu sumber daya fisik dapat dijalankan atau disimpan beberapa sumber daya virtual sekaligus. Pada saaat ini semua jenis virtualisasi dapat digolongkan menjadi 3 buah kategori yaitu paravirtulization virtual machine, hardware virtual machine dan container virtualization. Konsep dasar dari container adalah sebuah sistem yang memungkinkan untuk membuat layanan dan sumber daya yang berbeda dengan cara berbagi di dalam sebuah operating sistem dan berjalan menggunakan *libraries*, *drivers* dan *binaries* yang berbeda Konsep ini mengurangi jumlah sumber daya yang terbuang selama komputasi dikarenakan container hanya menjalankan dan menyediakan kebutuhan *binary* atau *library* yang sesuai dengan aplikasi atau layanan yang dijalankan [18].

2.2 CONTAINER DOCKER

Docker adalah salah satu platform aplikasi yang dibangun berdasarkan teknologi container. Docker merupakan sebuah aplikasi *platform* virtualisasi container yang bersifat open source, yang menyediakan *platform* terbuka untuk *developer* maupun *sysadmin* untuk dapat membangun, mengemas, dan menjalankan aplikasi dimanapun sebagai sebuah *container* yang ringan. Docker pertama kali dikembangkan oleh Solomon Hykes sebagai *project* internal di dotCloud bersama dengan beberapa koleganya yaitu Andrea Luzzardi dan Francois-Xavier Bourlet[14]. Platform Docker ini pertama kali rilis secara open source pada tahun 2013 silam [13].

2.3 DOCKER ENGINE

Docker *engine* merupakan sebuah aplikasi *client-server* yang memiliki 3 komponen utama sebagai berikut :

- Sebuah server dengan jenis program *long-running* disebut proses daemon.
- Sebuah REST API yang menentukan *interface* yang dapat digunakan program untuk berbicara dengan daemon dan menginstruksikan apa yang harus dilakukan.
- Sebuah *client* antarmuka baris perintah *Command Line Interface.*

2.4 DASAR JARINGAN DOCKER

Container menggunakan kemampuan partisi Linux yang disebut Cgroups dan namespaces. Proses Container diatur oleh Docker engine sehingga proses didalam container dapat dipetakan ke dalam jaringan, sistem penyimpanan dan namespaces lainnya. Di sisi jaringan, namespace memiliki hubungan jaringan sendiri dengan interface jaringan host, tabel *routing*, soket dan aturan IPTABLE. Arsitektur jaringan Docker dibangun di atas satu set antarmuka yang disebut Container Networking Model (CNM).

2.5 DENIAL OF SERVICE TCP SYN FLOOD

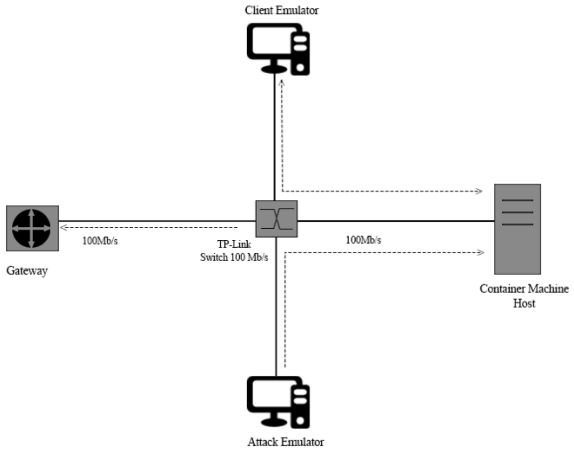
Denial Of Service (DoS) *Attack* adalah jenis serangan terhadap sebuah komputer atau server di dalam jaringan internet yang sistem kerjanya yaitu dengan cara menghabiskan sumber daya (*resource*) yang dimiliki oleh sebuah komputer sampai komputer tersebut tidak dapat menjalankan fungsinya dengan benar, sehingga secara tidak langsung mencegah pengguna lain untuk memperoleh akses layanan dari komputer yang diserang tersebut. Serangan SYN flooding adalah jenis serangan yang cukup membuat masalah didalam aplikasi yang menggunakan jaringan protocol TCP. Dalam kasus serangan SYN, klien tidak membalas respons paket SYN-ACK ke server untuk membuka koneksi antara klien dan server. Karena itu server dipenuhi dengan koneksi dalam keadaan setengah terbuka yang tersimpan di memori, dan sampai time out server mencoba mengirim paket SYN-ACK baru ke klien [16]. Salah satu jenis variasi serangan DoS TCP yaitu dengan menggunakan *Spoofing Attack*, dengan serangan jenis ini serangan akan menjadi lebih compleks dan cukup susah untu dideteksi karena alamat pengirimnya dibuat random dan bukan dari alamat pengirim asli, dan juga pengirim paket seranga tidak perlu untuk membalas paket SYN-ACKs yang direspon oleh target. Ilustrasi serangan ini dapat dilihat di Gambar 2.8. [3]

2.6 PERLINDUNGAN SERANGAN SYN FLOOD DENGAN SYN COOKIES

Dengan mengaktifkan fungsi Cookie SYN dalam kernel Linux maka sistem akan mengabaikan serangan SYN flood yang masuk, dengan keamanan ini memungkinkan server untuk menghentikan koneksi menggantung (setengah terbuka) yang masuk pada saat telah mencapai batas pada buffer. Sementara pada saat pembangunan koneksi dihentikan, server akan mengirimkan paket SYN-ACK cookies ke klien, dan jika menerima respons ACK dari klien, server akan merekonstruksi koneksi setengah terbuka yang sebelumnya telah diterima, dan memungkinkan komunikasi dengan klien.

3. DESAIN MODEL SISTEM

Gambar 3.1 merupakan model secara keseluruhan dari sistem yang akan digunakan untuk melakukan pengujian.



Gambar 3.1 Infrastruktur sistem

Fokus pengujian pada penelitian ini yaitu hanya pada bagian mesin yang menjalankan virtualisasi *container*, didalam mesin tersebut nantinya akan dijalankan service ubuntu *server native* dan dengan *Container*, kemudian diuji performanya dari sisi *overall performance* dan layanan *web server* ketika normal dan mendapatkan serangan dari mesin *attack emulator*. Fungsi dari attacker dalam sistem ini yaitu untuk melakukan generate paket serangan SYN, diasumsikan trafik paket yang dihasilkan sebesar 100Mb/s. Gateway berfungsi untuk melakukan drop paket serangan yang dihasilkan oleh mesin yang diuji, serta sebagai gerbang akses ke internet untuk melakukan request installasi paket Docker image.

4. HASIL DAN PEMBAHASAN

Pada bab ini dibahas analisis dari hasil pengujian dengan skenario dan parameter yang telah ditentukan, dengan mengimplementasikan teknik virtualisasi *container* menggunakan platform *Docker* yang dijalankan diatas Ubuntu *server* 14.0. Penelitian ini melakukan simulasi serangan DoS guna melihat performansi mesin dan layanan yang dijalankan diatasnya. Sebagai pembanding akan digunakan data dari hasil pengujian performansi native, dari *server* yang menggunakan sistem operasi Ubuntu *server* 14 tanpa menggunakan virtualisasi.

.

4.1. Pengujian Overall Performance

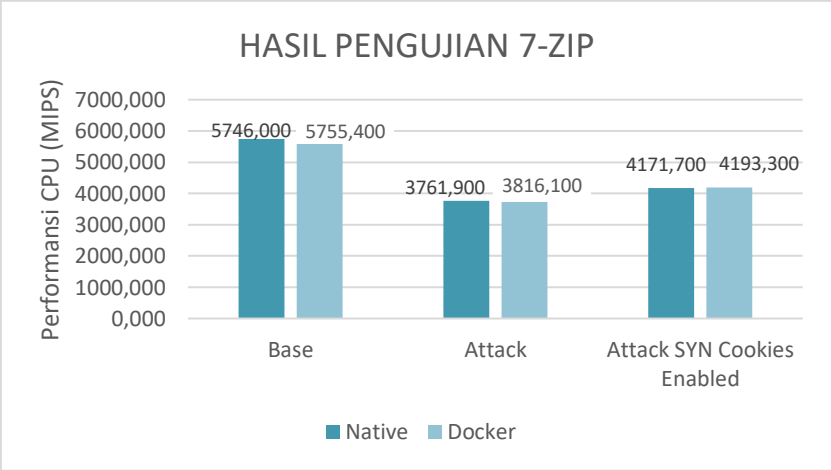
Pada pengujian ini ditampilkan hasil dari pengujian CPU, RAM, IOzone write, dan IOzone read. Skor *overall performance native* diasumsikan dengan nilai maksimal. Selanjutnya semua nilai hasil pengukuran akan dinormalisasi menggunakan perhitungan nilai normalisasi min-max. *Normalisasi min-max* adalah teknik sederhana yang mana teknik ini secara khusus membuat rentang data baru dengan batas yang telah ditentukan sebelumnya [11], sehingga dengan adanya normalisasi data dapat mempermudah dalam melakukan perbandingan pada parameter ini. Pada perhitungan ini rentang batas didefinisikan dari 0-1, maka digunakan rumus perhitungan *normalisasi min-max* pada persamaan (4.1).

$$A' = \frac{\text{A – Nilai Minimum A}}{\text{Nilai Maksimum A – Nilai minimum A}} \quad * \text{(D-C)+C} \qquad (4.1)$$

Dimana, A' berisi data Normalisasi Min-Max dengan rentang batas yang ditentukan adalah [C, D].

- **CPU**

  Hasil pengukuran performansi CPU menggunakan 7-Zip berupa satuan MIPS, pengukuran ini bertujuan untuk mendapatkan hasil berapa besar kemampuan CPU dalam menjalankan instruksi dalam satu detiknya.

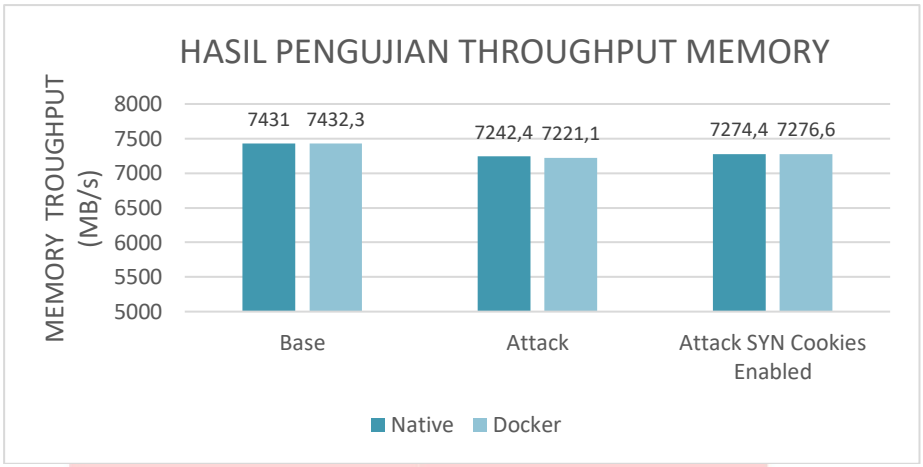


Gambar 4.1 Grafik Performansi

Hasil pengujian 7-Zip dapat dilihat di Tabel 4.1, jika nilai MIPS semakin tinggi maka semakin bagus performansinya. Terlihat selisih penurunan performansi pada skenario *SYN cookies* diaktifkan maupun di matikan antara *Docker* dan Native tidak lebih dari 5%. Penurunan performa terbesar terjadi pada skenario attack, karena pada skenario ini keamanan SYN cookies tidak diaktifkan, sehingga menyebabkan *server* berusaha memproses semua paket SYN dan dari proses tersebut berdampak pada penggunaan CPU. Sedangkan pada skenario *SYN cookies* diaktifkan nilai MIPS juga terdapat penurunan sebesar 27,40% untuk native dan 27,14% untuk *Docker* jika dibandingkan nilai base, penurunan ini disebabkan karena CPU bekerja dalam proses pembuatan paket *SYN-ACK cookies* dengan menggunakan algoritma *SYN cookies*, tetapi dengan aktifnya *SYN cookies* sistem akan mengabaikan *serangan SYN flood* yang masuk ketika telah mencapai batas buffer, sehingga penurunan performansi pada saat keamanan *SYN cookies* diaktifkan tidak sebesar ketika tidak diaktifkan.

- **MEMORY**

  Hasil pengujian memori ditunjukkan pada Tabel V . Dibawah serangan TCP DoS, semua setup menunjukan penurunan kinerja sekitar 2% baik SYN cookies diaktifkan dan dimatikan.

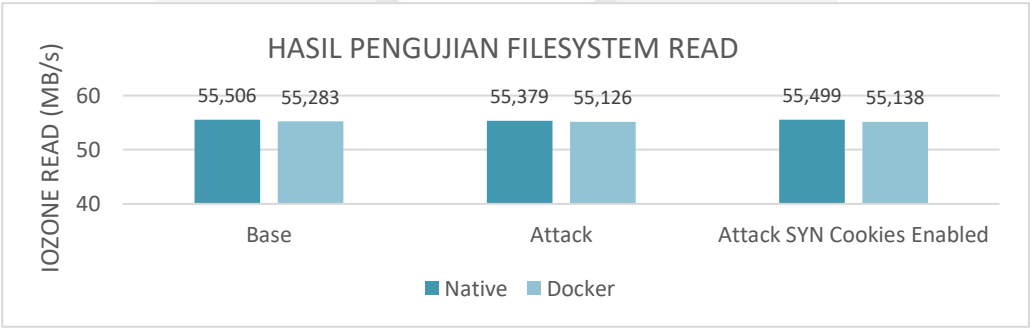


Gambar 4.2 Grafik Performansi RAM

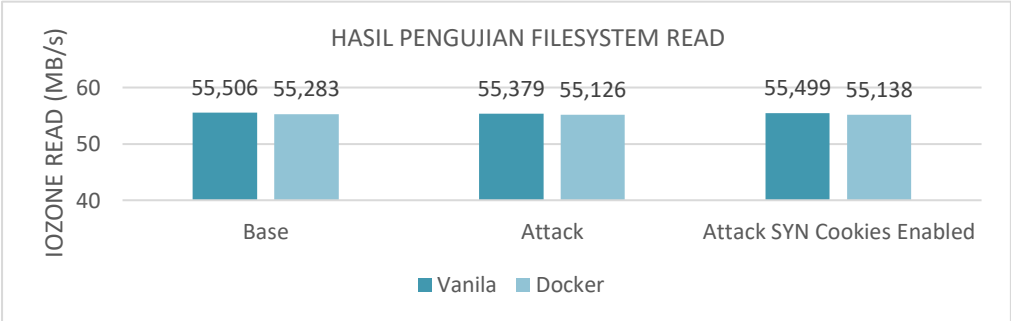
Dari hasil pengujian tersebut jika semakin besar nilainya maka semakin bagus performansinya. Terlihat terdapat penurunan performansi di sistem Native dan *Docker*, pada skenario serangan tanpa keamanan *SYN cookies*, besar persentase penurunannya yaitu 2.54% pada Native dan 1.95% pada *Docker*, penurunan ini diakibatkan oleh serangan *SYN* yang dianggap oleh *server* ingin mencoba membangun koneksi, kemudian *server* membalas dengan mengirim *SYN-ACK*, serta mengalokasikan resource *memory* untuk menunggu datangnya balasan *ACK* dari *client* yang melakukan *request*. Pada saat skenario serangan dengan *SYN cookies* diaktifkan juga terlihat terdapat penurunan performansi, tetapi penurunan performansi pada skenario ini tidak sebesar ketika tidak menggunakan keamanan *SYN cookies*, ini diakibatkan oleh aktifnya fungsi algoritma *SYN cookies* dalam kernel Linux sehingga sistem akan mengabaikan *serangan SYN flood* yang masuk, dengan keamanan ini memungkinkan *server* untuk menghentikan koneksi menggantung saat telah mencapai batas pada *buffer*, oleh sebab itu alokasi *memory* untuk koneksi tidak sebanyak pada saat skenario tidak menggunakan keamanan *SYN cookies*

- **Iozone Read dan Iozone Write**

Hasil pengujian terakhir untuk parameter overall performance adalah untuk uji Iozone read dan write, diberikan pada Gambar 4.3 dan Gambar 4.4.



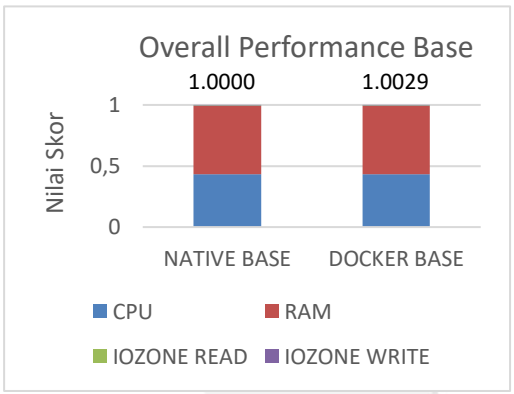
Gambar 4.3 Grafik Performansi Write Iozone



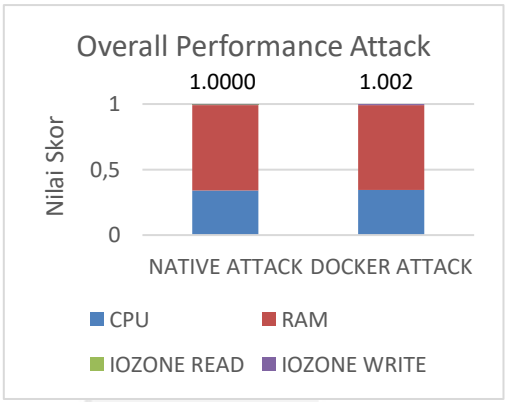
Gambar 4.4 Grafik Performansi Iozone Read

Untuk pengujian *Iozone read* dan *write* , semakin besar hasilnya maka semakin bagus performansinya. Hasil pengujian pada parameter ini tidak terlalu terdampak performansinya oleh serangan DoS, ini dikarenakan serangan DoS yang digunakan menggunakan paket berukuran sangat kecil, hanya terdapat header paket dan *payload* tidak disi data sama sekali, sehingga pada sisi read dan write data tidak terdapat penurunan yang besar pada semua skenario dibawah serangan DoS, adanya penurunan yang relatif kecil diakibatkan pada saat sistem mendapatkan serangan, sistem IO bekerja dalam melayani packet serangan yang masuk dan juga melakukan pembuatan paket ACK balasan yang juga melibatkan IO read dan write.

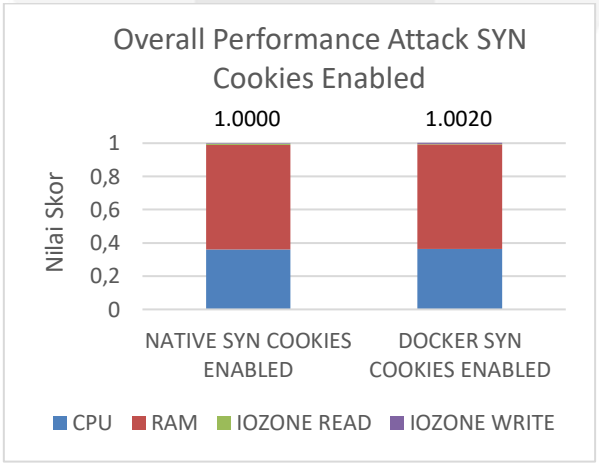
Pada saat *SYN cookies* diaktifkan penurunan performansi didapat lebih kecil jika dibandingkan tidak diaktifkan, ini karena pada saat *SYN cookies* diaktifkan *server* membatasi untuk membangun koneksi yang menggantung (SYN_RECV) sehingga dalam pembacaan paket yang masuk akan dibatasi, sehingga didapat performa sedikit lebih tinggi jika dibandingkan dengan *SYN cookeis* dimatikan. Selisih performansi antara docker pada saat diserang maupun kondisi normal tidak terdapat perbedaan yang jauh, terlihat di semua skenario selisih performansi tidak ada yang melebihi 1%. Hasil dari parameter pengujian ini didapat standar deviasi yang cukup kecil, pada semua skenario nilainya tidak ada yang melebihi 1%, ini menandakan bahwa performa write file sistem cukup stabil, tidak ada penurunan ataupun peningkatan yang signifikan pada setiap pengujian.



Gambar 4.5 Grafik Overall Performance Base



Gambar 4.6 Grafik Overall Performance Attack



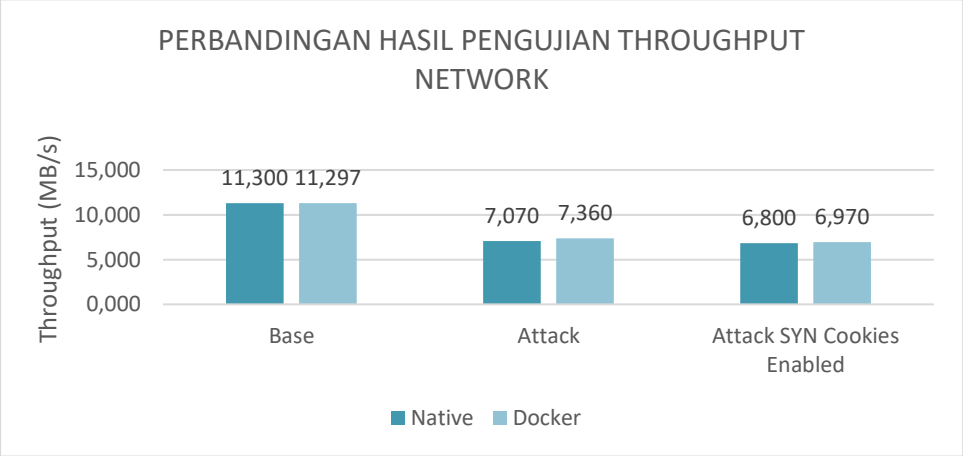
Gambar 4.7 Grafik Overall Performance Attack Cookies

Dari seluruh skenario untuk metrik *overall performance* tidak ada perbedan nilai performansi yang signifikan diantara *Docker* dan Native, semua skor performansi *Docker* hasilnya mendekati satu (pendekatan dua angka di belakang koma) maka dari hasil tersebut membuktikan bahwa dengan menggunakan teknik virtualisasi *container* menggunakan *Docker* perfomansinya yang sangat mendekati mesin native, baik pada saat keadaan normal maupun mendapatkan serangan TCP *SYN flood*. Pada container Docker didapat performansi sedikit lebih unggul, ini sebab virtualisasi *container* mengurangi jumlah sumber daya yang terbuang selama komputasi dengan hanya menjalankan dan menyediakan kebutuhan *binary* atau *library* yang sesuai dengan aplikasi atau

layanan yang dijalankan.mendekati mesin native, baik pada saat keadaan normal maupun mendapatkan serangan TCP SYN flood.

4.2. PENGUJIAN NETWORK

Hasil Pengujian Iperf dapat dilihat di Table 4.8. Pengukuran Iperf bertujuan untuk mengukur seberapa besar kecepatan transfer maksimum diantara dua host.



Gambar 4.8 Grafik Performansi Network

Pada pengukuran performansi base hampir tidak ada perbedaan performa, selisih diantara Native dan *Docker* hanya sebesar 0,003MB/s, semakin besar nilai throughput maka akan semakin bagus. Overhead pada sisi jaringan docker didapat sangat kecil sekali, diduga ini disebabkan oleh penggunaan *driver Macvlan* sebagai mode jaringan *container*, dengan *driver* Macvlan *container* dapat langsung terhubung dengan mesin client yang bertindak sebagai *user* pada layer 2, tanpa harus melalui jaringan mesin Ubuntu host, dan implementasi *driver* Macvlan sangatlah ringan di linux, *driver* tersebut hanya menerapkan pemisahan antara jaringan *container* dan koneksi fisik jaringan host [10].

Pengujian skenario serangan tanpa keamanan *SYN cookies* terdapat penurunan performa pada sistem Native sebesar 37,38% dan *Docker* 36,83%, pada sistem *Docker* penurunan performansi sedikit lebih kecil nilainya jika dibandingkan dengan Native, diduga dengan penggunaan *driver* Macvlan pada jaringan *Docker*,

Pada skenario *SYN cookies* diaktifkan terlihat pada kedua sistem terdapat penurunan performansi pada *Docker* sebesar 38.22 dan Native 39.79, selisih penurunan performa kedua sistem sangatlah kecil yaitu hanya sebesar 1,57%, pada skenario ini tetap ada penurunan performansi walaupun keamanan *SYN cookies* diaktifkan, dikarenakan pada saat pengujian jalur pengiriman paket data untuk pengujian terbagi oleh paket yang dibuat oleh algoritma *SYN cookies* , yaitu dengan paket balasan SYN-ACK cookies untuk meresponse setiap paket SYN yang dihasilkan oleh serangan DoS [20], oleh sebab itu bandwidth akan terbagi oleh paket cookies SYN-ACK yang dikirim oleh *server* dan paket serangan yang dikirim oleh *attacker*, sehingga pada saat *SYN cookies* diaktifkan terjadi penurunan *throughput* sedikit lebih besar jika dibandingkan dengan serangan tanpa *SYN cookies*.

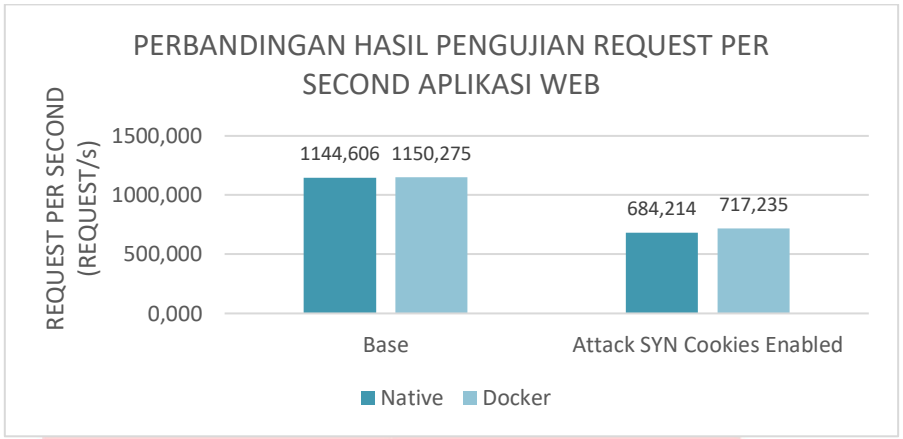
4.3. PENGUJIAN APLIKASI WEB

Dari beberapa hasil pengujian sebelumnya sudah jelas bahwa performa *container* dan *Docker* performanya kurang lebih sama, baik pada kondisi normal maupun mendapatkan serangan. Selanjutnya untuk akan dilakukan pengujian dengan skenario yang lebih kompleks, untuk mengukur seberapa besar pengaruh serangan terhadap sistem layanan *web server*. Pengujian dan simulasi pada sisi client digunakan aplikasi ab-apache.

Pada pengujian layanan aplikasi *web*, skenario yang dapat dijalankan hanya skenario serangan pada saat *SYN cookies* diaktifkan, karena sebelumnya telah dilakukan pengujian tanpa menggunakan keamanan *SYN cookies*, semua sistem yang diuji mendapatkan penurunan *throughput* menjadi nol pada pengujian layanan *web server*. Hal ini dapat terjadi karena tanpa keamanan *SYN cookies*, antrian koneksi TCP dengan cepat diisi oleh paket SYN serangan DoS, sehingga tidak ada ruang untuk membangun *request* koneksi yang telah dikirimkan oleh client. Ruang buffer koneksi yang disediakan oleh *server* dapat mengalami penumpukan, dan *server* pun menjadi tidak dapat membaca permintaan koneksi baru yang datang, hingga paket SYN yang sebelumnya mengalami gagal eksekusi atau sering dikenal dengan istilah *Time Out*.

a. Request Per Second

Pengujian pertama dimulai dengan evaluasi throughput maksimum masing-masing sistem. Jumlah throughput layanan web server diekspreksikan dengan jumlah maksimum request yang dapat dilayani oleh web server dalam tiap detik.

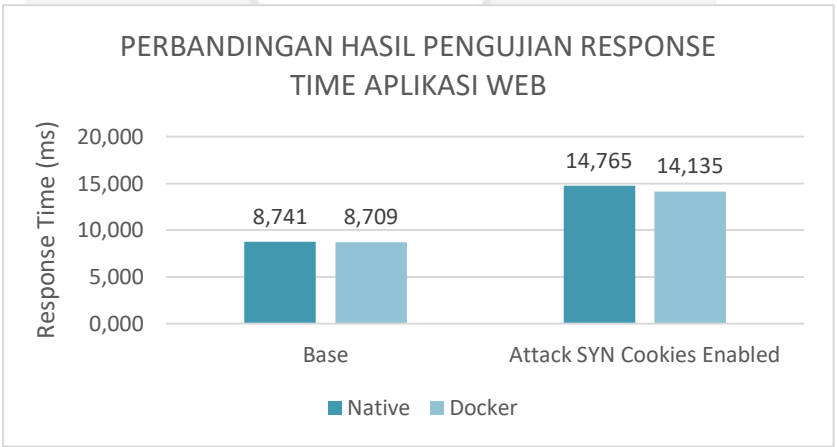


Gambar 4.9 Grafik Request Per Second

Pada pengukuran parameter ini semakin besar nilai request maka akan semakin bagus. Dapat dilihat pada Gambar 4.9, performansi Native dan *Docker* tidak memiliki perbedaan yang signifikan, pada saat mendapatkan serangan performansi Native dan *Docker* hanya memiliki selisih penurunan sebesar 2,57%. Performa sistem *Docker* pada saat skenario serangan DoS sedikit lebih unggul, ini karena berbanding lurus dengan hasil pengujian *throughput* jaringan sebelumnya. Penurunan performansi pada saat mendapatkan serangan DoS diakibatkan oleh trafik data yang terbagi oleh paket serangan SYN yang masuk kedalam *server*, dan juga adanya SYN-ACK cookies yang dikirimkan oleh *server* membuat jalur pengiriman data dan kapasitas jaringan menurun karena trafik data terbagi, sehingga mengakibatkan jumlah *request per second* mengalami penurunan sebesar 40,22% pada Native dan 37,65% pada *Docker*.

b. Response Time

Selanjutnya, akan diuji rata-rata waktu yang dibutuhkan untuk membangun koneksi HTTP ke server web. Pengukuran menggunakan aplikasi ab-apache, dari pengujian akan diukur waktu yang dibutuhkan dalam menjalankan tiap request ke server, dan akan dihitung nilai rata-rata dari semua request yang telah berhasil dijalankan.



Gambar 4.10 Grafik Response Time

Pada pengujian ini semakin kecil nilai *response time* maka akan semakin bagus. Hasil pengujian *response time* dapat dilihat pada Gambar 4.10. Pengujian performansi base kedua sistem terlihat hampir tidak ada perbedaan performansi, selisih performansi dari kedua sistem hanya sebesar 0,04ms. Pada saat mendapat serangan terdapat kenaikan *response time* yang cukup tinggi yaitu pada Native sebesar 40,80% dan pada *Docker* sebesar 38,38%, kenaikan *response time* ini disebabkan karena banyaknya paket serangan

SYN yang masuk dan harus diproses oleh *server*, kemudian komputer *server* juga harus menangani proses request layanan dari client bersamaan dengan proses penanganan paket SYN yang dikirim oleh penyerang, sehingga terjadilah kenaikan waktu *response time*. Tetapi dari perbandingan kedua sistem tidak terdapat perbedaan yang cukup jauh, selisih *response time* dari kedua sistem pada saat mendapatkan serangan hanya sebesar 0,62ms, dan hasil pengujian ini berbanding terbalik terhadap pengujian performansi CPU, semakin besar nilai CPU maka nilai *response time* akan semakin kecil.

5. Kesimpulan dan Saran

A. Kesimpulan

Dari hasil pengujian dan analisis performansi sistem native dan Docker dibawah serangan DoS SYN flood didapatkan bahwa serangan mengakibatkan penurunan performansi disemua sistem tersebut, dengan perbedaan performansi berdasarkan metrik *overall performance*, *network* dan layanan web server, yang mana didapatkan kesimpulan sebagai berikut :

1. Ditinjau dari perbandingan pengukuran overall performance Ubuntu menggunakan *container Docker* performansinya tidak berbeda jauh dengan ubuntu native (pendekatan dua angka di belakang koma). Docker sedikit lebih unggul dikarenakan container mengurangi jumlah sumber daya yang terbuang selama komputasi dengan hanya menjalankan dan menyediakan kebutuhan *binary* atau *library* yang sesuai dengan aplikasi atau layanan yang dijalankan, oleh karena itu proses didalam container hampir tidak ada *overhead.*
2. Secara umum dampak dari setiap serangan yang dilancarkan terhadap overall performance tidak cukup besar. Dikarenakan kapasitas jaringan dalam hal ini lebar bandwidth yang terbatas serta kecepatan pemrosesan baik pada *node server* maupun klien yang cukup cepat.
3. Berdasarkan hasil pengukuran pada parameter *network* menunjukkan bahwa *throughput* pada *Docker* sedikit lebih unggul performansinya pada skenario serangan *SYN cookies*, ini diduga karena penggunaan *driver Macvlan* pada *container* Ubuntu membuat performa jaringan container sedikit lebih bagus, sehingga akan memberikan *throughput* yang lebih besar.
4. Serangan DoS *SYN flood* berpengaruh terhadap jalannya layanan web server, karena pada saat keamanan *SYN cookies* tidak diaktifkan layanan web tidak dapat diakses sama sekali akibat adanya serangan yang masuk.
5. Virtualisasi berbasis *container* dapat menjadi solusi yang lebih baik untuk sistem yang terpapar serangan DoS.

B. Saran

Penelitian selanjutnya disarankan untuk menguji jenis serangan yang berbeda dan juga menggunakan metode keamanan yang berbeda. Penggunaan firewall dapat menjadi solusi terbaik, karena perangkat tersebut dapat memfilter paket yang akan masuk kedalam server, sehingga server tidak akan mengalami penurunan performansi dalam melakukan penanganan serangan. Hal tersebut bertujuan untuk mengetahui dan mendapatkan kinerja sistem yang lebih baik sehingga didapatkan kualitas layanan yang lebih baik pula.

Daftar Pustaka

[1] [Online]. Available : https://www.sdxcentral.com/articles/news/survey-container-adoption-is-skyrocketing/2015/06/.[Accessed 5 April 2017]

[2] [Online]. Available : https://securelist.com/analysis/quarterly-malware-reports/74550/kaspersky-ddos-intelligence-report-for-q1-2016/.[Accessed 10 Januari 2018]

[3] Wesley M. Eddy.Defenses Against TCP SYN Flooding Attacks - The Internet Protocol Journal - Volume 9, Number 4

[4] Deshane, T., Shepherd, Z., Matthews, J., Ben-Yehuda, M., Shah, A. and Rao, B., 2008. Quantitative comparison of Xen and KVM. Xen Summit, Boston, MA, USA, pp.1-2.

[5] Rabindra K. Barik, Rakesh K. Lenka, K. Rahul Rao, and Devam Ghose, “Performance Analysis of Virtual Machines and Containers in Cloud Computing”, in International Conference on Computing, Communication and Automation (ICCCA2016)

[6] Casalicchio, Emiliano & Perciballi, Vanessa. (2017). Measuring Docker Performance: What a Mess!!!. . 10.1145/3053600.3053605.Conference: ACM ICPE ’17 Companion

[7] Ryan Shea, “Performance of Virtual Machines Under Networked Denial of Service Attacks: Experiments and Analysis”,in 2013 IEEE SYESTEM JOUNAL, VOL 7. NO. 2

[8] TCP SYN Flooding and IP Spoofing Attacks, 1996 Advisory, Software Engineering Institute, Carnegie-Mellon University

[9] W. Richard Stevens. TCP/IP Illustrated, Volume 1: The Protocols.

[10] Fabrizio Soppelsa, Chanwit Kaewkasi, “Native Docker Clustering with Swarm”.

[11] PATRO, S GOPAL & Kumar Sahu, Kishore. (2015). Normalization: A Preprocessing Stage. IARJSET.10.17148/IARJSET.2015.2305.

[12] Stallings (2005). Operating Systems, Internals and Design Principles. Pearson: Prentice Hall. p. 6.

[13] [Online]. Available : *Avram, Abel (2013-03-27). "Docker: Automated and Consistent Software Deployments". InfoQ.*[Accessed 5 April 2017]

[14] [Online]. Available : "One home for all your apps". *dotcloud.com. Archived from* the original *on 2014-05-17.*[Accessed 5 April 2017]

[15] [Online]. Available : *https://docs.Docker.com/engine/Docker-overview/*.[Accessed 5 April 2017]

[16] Bogdanoski, Mitko & Shuminoski, Tomislav & Risteski, Aleksandar. (2013). Analysis of the SYN flood DoS attack. International Journal of Computer Network and Information Security. 5. 1-11. 10.5815/ijcnis.2013.08.01.

[17] [Online]. Available : https://success.docker.com/article/Docker_Reference_Architecture-_Designing_Scalable,_Portable_Docker_Container_Networks/.[Accessed 5 April 2017]

[18] Rajdeep Dua, A Reddy Raja, and Dharmesh Kakadia, “Virtualization vs Containerization to support PaaS”, in 2014 IEEE International Conference on Cloud Engineering

[19] Hung, Ling-Hong & Kristiyanto, Daniel & Bong Lee, Sung & Yeung, Ka Yee. (2016). GUIdock: Using Docker Containers with a Common Graphics User Interface to Address the Reproducibility of Research. PloS one. 11. e0152686. 10.1371/journal.pone.0152686.

[20] Wesley M. Eddy, “Defenses Against TCP SYN Flooding Attacks” , in 2006 The Internet Protocol Journal - Volume 9, Number 4 ,

[21] STREAM: Sustainable Memory Bandwidth in High Performance Computers
John D. McCalpin, Ph.D.john@mccalpin.com*"Dr. Bandwidth"*