SEMINAR NASIONAL TEKNOKA

Vol. 6, 2021
ISSN No. 2502-8782

Analisis Kinerja Honeypot Dionaea Dan Cowrie Dalam Mendeteksi Serangan

Melia Mispriatin, Jaffaruddin Gusti Amri Ginting, & Bongga Arifwidodo
Institut Teknologi Telkom Purwokerto Telp 0821-641629 Fax (0281) 641630, Mobile 081 126 161 16
Website: http://ittelkom-pwt.ac.id/, E-mail: info@ittelkom-pwt.ac.id

Abstrak

Pada era digital, kerentanan sistem menjadi poin utama untuk masuknya dari berbagai serangan yang dapat membuat sistem dikendalikan pihak asing bahkan lumpuh. Maka, diperlukan penanganan untuk masalah tersebut. Honeypot merupakan sistem keamanan untuk memantau dan menganalisis setiap aktivitas serangan yang masuk ke sistem. Low interaction honeypot (dionaea) dan medium interaction honeypot (cowrie) dipilih untuk mengamankan server. Serangan yang dipilih adalah Port scanning, Bruteforce SSH dan DoS dengan 2 skenario yaitu tanpa honeypot dan dengan honeypot. Hasil penelitian menunjukan Dionaea dapat menangkap lebih banyak serangan dibanding Cowrie. Sedangkan dari segi interaksi antara server dengan penyerang, Cowrie lebih unggul. Hasil QoS serangan DoS, baik sebelum maupun setelah terpasang honeypot nilai throughput, packet loss dan delay masih dalam kategori sangat bagus dengan index 4 sedangkan nilai jitter setelah terpasang honeypot naik dari index 2 menjadi 3, kategori sedang menjadi baik. Sedangkan penggunaan CPU naik 6,66% menjadi 49% dan Memory naik 12,48% menjadi 37,6%.

***Kata Kunci**: Dionaea, Cowrie, CPU, QoS*

Abstract

In the digital era, system vulnerabilities are the central point for various entry attacks that can make the system controlled by foreign parties and even paralyzed. So, a solution for this problem is needed. Honeypot is a security system to monitor and analyze every attack activity that enters the system—low interaction honeypot (Dionaea) and medium interaction honeypot (Cowrie) to secure the server. The attacks chosen are Port scanning, Bruteforce SSH and DoS with two scenarios, without honeypot and honeypot. Results showed that Dionaea could catch more attacks than Cowrie. Meanwhile, in terms of interaction between server and attacker, Cowrie is superior. The QoS results of DoS attacks, both before and after honeypot was installed, throughput, packet loss and delay values were still in the excellent category with index 4. In contrast, jitter value after honeypot was installed increased from index 2 to 3, moderate category to good. While CPU usage decreased 6.66% to 49% and Memory increased 12.48% to 37.6%.

***Keyword**: Dionaea, Cowrie, CPU, QoS*

1 PENDAHULUAN

Semakin bertambahnya pengguna internet, semakin banyak pula gangguan yang mengancam komputer, terlebih lagi sistem pemerintah yang menegakkan kebijakan *work from home* (WFH) pada sebagian instansi, baik perkuliahan maupun perkantoran yang membuat keadaan tersebut semakin dimanfaatkan oleh para *hacker* untuk melakukan berbagai serangan. Berdasarkan Pusat Operasi keamanan Siber Nasional (Pusopskamsinas) Badan Siber dan Sandi Negara (BSSN) mencatat sebanyak tidak kurang dari 88 jt serangan siber telah terjadi sejak 1 Januari hingga 12 April 2020 [1]. Hal tersebut tentu mengkhawatirkan bagi banyak orang, terlebih berbagai ancaman yang menyerang *server* mulai dari *port scanning*, *bruteforce* dan *denial of service* dapat membuat *server* lumpuh sehingga tidak dapat melayani berbagai permintaan dari *client* [2].

Honeypot merupakan sistem yang sengaja di korbankan untuk menjadi sasaran penyerang dengan tujuan dapat mencacat aktivitas serangan yang masuk dalam *server* sehingga *server* dapat memperbaiki bagian – bagian yang terkena serangan [3].

Berdasarkan beberapa penelitian sebelumnya seperti penelitian [4] yang meneliti tentang

Dipresentasikan pada Tanggal 27 November 2021

perbandingan honeypot menurut tingkat interaksinya dengan mengambil satu *sample* pada tiap tingkatnya yaitu dionaea, cowrie dan suricata. Raspberry Pi sebagai pengganti komputer sedangkan ELK Stack untuk visualisasi *log* yang dihasilkan. Adapun hasil penelitian ini berupa *log file* honeypot, seberapa banyak serangan dapat terdeteksi dan monitoring performa CPU yang dihasilkan pada masing – masing honeypot yang terpasang. Kemudian penelitian [5] yang mengambil Honeypot Kippo dan Dionaea untuk mendeteksi berbagai serangan selama 55 hari. Serta penelitian [6] yang menganalisa jaringan nirkabel dari serangan DdoS menggunakan honeypot Dionaea.

Penelitian [4],[5], dan [6] menyatakan bahwa dengan dengan perbedaan jenis interaksi keduanya membuat keduanya saling melengkapi. Jenis interaksi dalam honeypot hanya menunjukan seberapa intens honeypot tersebut dapat berinteraksi dengan *attacker*. Sehingga semakin tinggi jenis interaksinya tidak menjamin bahwa honeypot tersebut akan lebih banyak dalam menangkap serangan. Maka dari itu, penulis ingin menggabungkan honeypot jenis interaksi rendah (dionaea) untuk mendapatkan lebih banyak serangan yang dapat dideteksi seperti pada penelitian sebelum – sebelumnya dan honeypot dengan interaksi sedang (cowrie) untuk lebih mengetahui bagaimana respon dari *server* ketika *attacker* berhasil masuk dan informasi apa yang disampaikan oleh honeypot. Adapun serangan yang menjadi bahan uji coba kedua honeypot adalah *port scanning*, *bruteforce* ssh dan *denial of service*.

2 LANDASAN TEORI

Honeypot adalah sistem keamanan yang sengaja dibuat untuk menyelidiki penyerangan atau membuat *web server* untuk menjebak *attacker*[4]. Prinsip kerja dari honeypot sendiri adalah ketika terdapat komputer asing yang ingin berkomunikasi dengan *server* asli mencoba untuk menyusup maka honeypot akan menjadi benteng dari *server* asli yang menyerupainya sehingga *attacker* akan melihat honeypot sebagai *server* asli yang memang harus diserang.

Dionaea adalah salah satu jenis honeypot *low interaction* yang memiliki tujuan untuk mendapatkan *copy* dari malware, selain itu dapat mendeteksi beberapa serangan salah satunya *Denial of service*. Untuk melakukannya, dionaea menggunakan bahasa pemrograman yaitu phyton sebagai bahasa libemu. Setelah dionaea memperoleh lokasi berkas yang diinginkan penyerang dari *shellcode*, dionaea akan mencoba mengunduh *file* tersebut. Protokol yang digunakan untuk mengunduh *file* tersebut adalah tftp dan ftp yang diimplementasikan di python format fttp.py dan ftp.py agar terbaca oleh dionaea sendiri[7].

Cowrie adalah honeypot yang didesain untuk mengatasi masalah yang berhubungan dengan ssh/telnet. Seringkali cowrie digunakan untuk merekam aktivitas serangan *bruteforce* maupun *shell interaction*. Pada dasarnya honeypot ini merupakan evolusi dari Kippo. Namun karena pengumpulan data pada Kippo dirasa lebih sulit dan terlalu banyak *script* maka cowrie datang sebagai pembaharuan dari honeypot tersebut. Jika di lihat dari jenis interaksinya, cowrie termasuk dalam *medium interaction* [8].

Server Modern Honeypot Network (MHN) adalah *server* terpusat untuk pengelolaan dan pengumpulan data honeypot. MHN memungkinkan untuk menyebarkan sensor dengan cepat dan mengumpulkan data dengan segera, dapat dilihat dari antarmuka web yang rapi. *Script* Honeypot mencakup beberapa teknologi honeypot umum, termasuk Snort, Cowrie, Dionaea, dan glastopf [9].

Bruteforce adalah serangan yang berusaha untuk membobol *username* dan *password* dengan cara terus – menerus mencoba memasukannya sampai berhasil masuk. Dalam hal ini *attacker* menggunakan kamus sandi sebagai perantara untuk mendapatkan *password* yang ingin di carinya. *Attacker* akan mencoba *password* satu per satu untuk di otentikasi. Jika kamus sandi tersebut terdapat *password* yang benar maka *attacker* akan berhasil membobol *server* tersebut. Semakin rumit *password* maka akan semakin sulit juga *attacker* membobol *password* tersebut dan semakin banyak juga percobaan yang dilakukan oleh *attacker* [10].

Denial of service atau DoS merupakan salah satu serangan yang bertujuan untuk membuat komputer atau *server* yang menjadi target serangan menjadi *down* atau mati dikarenakan penyerang membanjiri komputer tersebut dengan pesan-pesan sampah dalam jumlah besar dan dilakukan secara terus menerus [11].

Port scanning adalah salah satu teknik yang digunakan oleh para penyerang untuk mencari celah sehingga mereka dapat masuk ke suatu layanan. *Port scanning* terdiri dari penyelidikan sejumlah jaringan untuk mencari *port* yang terbuka[12]. Prinsip kerja dari *port scanning* sebenarnya adalah memeriksa setiap *port* dari *port* 0 sampai dengan *port* 65535. Hal ini dilakukan hanya dengan mengirimkan permintaan ke setiap *port* dan meminta respon [13].

Pada suatu sistem akan selalu terjadi beberapa proses yang berjalan dalam satu waktu yang memerlukan sebuah CPU untuk menjalankannya.

Utilization CPU merupakan keadaan CPU saat menjalankan suatu sistem. Pada umumnya *Utilization* CPU dinyatakan dalam bentuk persen yaitu 0-100% [14].

Memory Utilization adalah rata – rata pemanfaatan *memory* yang digunakan pada waktu tertentu. Tingkat penggunaan *memory* yang tinggi dapat menyebabkan pembacaan *memory* melambat dan menurunkan kinerja perangkat dari waktu ke waktu karena *memory* tidak lagi tersedia [15].

Throughput adalah kecepatan (*rate*) transfer data efektif yang diukur dalam bps. *Throughput* di dapat dari jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama waktu tertentu di bagi oleh durasi interval waktu tersebut.

Tabel 1 *Standar Throughput*

Kategori	*Throughput* (bps)	Indeks
Sangat Bagus	100	4
Bagus	75	3
Sedang	50	2
Jelek	<25	1

Packet loss merupakan parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collision* dan *congestion* pada jaringan.

Tabel 2 *Standar Packet loss*

Kategori	*Packet loss*	Indeks
Sangat Bagus	0%	4
Bagus	3%	3
Sedang	15%	2
Jelek	25%	1

Delay merupakan waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses yang lama.

Tabel 3 *Standar Delay*

Kategori	*Delay* (ms)	Indeks
Sangat Bagus	<150	4
Bagus	<250	3
Sedang	<350	2
Jelek	<450	1

Jitter biasa disebut variasi *delay*, berhubungan erat dengan *latency*, yang menunjukkan banyaknya variasi *delay* pada transmisi data di jaringan.

Tabel 4 *Standar Jitter [16]*

Kategori	*Jitter* (ms)	Indeks
Sangat Bagus	0	4
Bagus	0 s/d 75	3
Sedang	75 s/d 125	2
Jelek	125 s/d 225	1

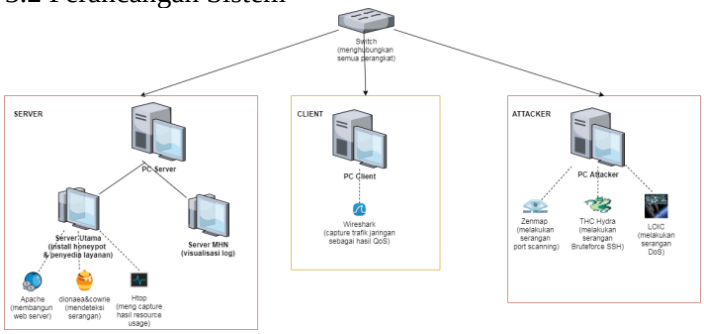
3 METODOLOGI PENELITIAN

Berikut adalah metode penelitian yang digunakan untuk membangun sistem honeypot:

3.1 Langkah Penelitian

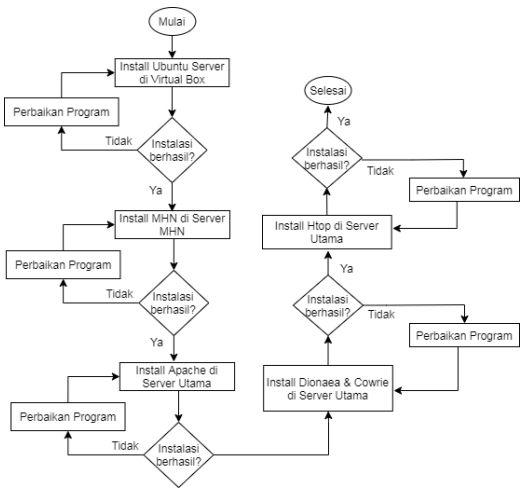
Tahap awal dalam melakukan penelitian analisis kinerja honeypot dalam mendeteksi serangan yaitu *study* literatur, kemudian perencanaan sistem, selanjutnya instalasi perangkat yang akan digunakan untuk melakukan simulasi. Tahap selanjutnya adalah melakukan pengujian terhadap sistem dan pengambilan data. Setelah data didapatkan, selanjutnya menganalisa hasil pengujian tersebut dan diambil kesimpulan serta saran yang membangun penelitian selanjutnya.

3.2 Perancangan Sistem



Gambar 1 *Topologi Jaringan*

Penelitian ini menggunakan 3 PC *real* dimana 1 PC digunakan untuk *Server* yang didalamnya terdapat 2 *virtual mechine* (*server* utama dan *server* MHN), PC 2 digunakan sebagai *client* yang mengakses web *server* dari *server* utama, PC 3 digunakan sebagai *attacker* yang didalamnya terdapat aplikasi LOIC (serangan DoS), Nmap (serangan *port scanning*) dan THC Hydra (serangan *bruteforce*) sebagai *tool* penyerang.



Gambar 2 *Konfigurasi Sistem*

Konfigurasi sistem digunakan untuk membuat sistem pada PC *Server*. Tahap pertama adalah dengan meng-*install server* MHN pada *virtual mechine* 1 terlebih dahulu dengan menggunakan *script* "*sudo ./install.sh*", kemudian menyelesaikan MHN *Configuration*. Tahap kedua yaitu meng-*install* Apache

pada *virtual mechine* 2 yang digunakan sebagai *server* utama dengan perintah "*sudo apt install apache2*".

Tahap selanjutnya meng-*install* dionaea dan cowrie di *virtual mechine* yang sama. Pada *dashboard* MHN bagian *deploy*, *link* sudah disediakan untuk meng-*install* honeypot tersebut. Tahap terakhir adalah meng-*install* htop yang digunakan untuk mengambil data *resource usage*.

Skenario pengujian sistem pada penelitian ini dengan menggunakan 2 skenario. Skenario pertama yaitu kondisi ketika *server* belum terpasang oleh honeypot. Skenario kedua ketika *server* sudah terpasang honeypot. Adapun *log file* dari honeypot akan tersimpan dalam *attack report* yang bisa diambil dari *server* MHN.

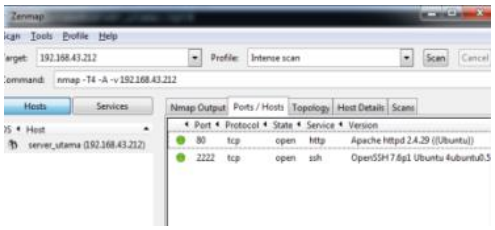
Alur pengujian skenario pertama dimulai dengan pada PC *Client* yang mengakses *web server* menggunakan url http://192.168.43.212 (ip *address server* utama) melalui *web browser* secara *idle*. Pada saat yang bersamaan, PC *attacker* melakukan serangannya mulai dari *port scanning* dengan menggunakan jaringan internet yang sama pada *client* dan *server* menuju PC *server*. Selanjutnya serangan *Bruteforces* SSH dan terakhir serangan DoS masing masing 30 kali percobaan setiap serangan. Setiap kali selesai melakukan percobaan, penulis akan mencatat hasil percobaan terlebih dahulu baik dari *tool* yang digunakan untuk menyerang maupun hasil wireshark pada sisi *client*. Alur pengujian skenario kedua sama halnya dengan alur pada skenario pertama. Hal yang membedakan antara skenario pertama dan kedua adalah adanya *server* MHN yang diaktifkan bersamaan dengan *server* utama. Setiap serangan yang masuk dari PC *attacker* otomatis akan dialihkan dari *server* asli menuju honeypot. Hasil *log file* dari tiap seranganpun dapat diambil melalui *dashboard server* MHN.

4 HASIL DAN PEMBAHASAN

Berikut hasil dari pengujian dan pembahasan dari penelitian ini.

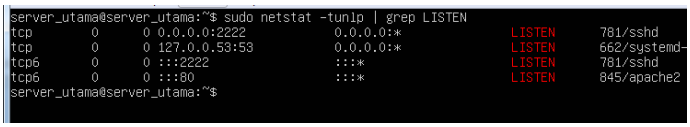
4.1 Pengujian *Port scanning*

Pengujian pertama adalah *server* tanpa honeypot. Pengujian *port scanning* digunakan untuk melihat *port* – *port* yang terbuka pada *server* yang sudah di targetkan. Pengujian skenario pertama (tanpa honeypot) dengan serangan *port scanning* menggunakan *tool* Nmap ip *address* yang menjadi target serangan adalah 192.168.43.212 dengan *username* '*server_utama'*. Hasil dari Nmap yang berbentuk tabel dengan kolom pertama menunjukan *port* apa saja yang ter-*scanning*, kolom kedua menunjukan jenis *protocol* dari *port* pada kolom sebelumnya. Kolom selanjutnya yaitu *state* menunjukan status *port* yang terbuka (*open*). Kolom keempat yaitu *service*, menunjukan nama *port* tersebut. Kemudian, dengan adanya dua baris yang tercantum pada hasil Nmap maka dapat diketahui bahwa terdapat dua *port* yang terbuka yaitu *port* 80 (http) dan *port* 2222 (ssh) dimana keduanya termasuk dalam *protocol* tcp.



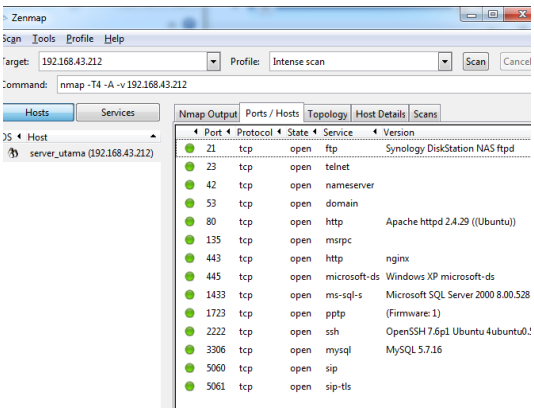
Gambar 3 *Pengujian Port scanning Pada Skenario Pertama*

Perintah '*sudo netstat –tunlp | grep LISTEN'* pada halaman *virtual mechine server* utama yang berfungsi untuk menampilkan berbagai *port* yang terdapat di sistem. Sesuai Gambar 3, Gambar 4 juga menampilkan hasil yang tidak jauh berbeda. Pada kolom pertama merupakan jenis *protocol* dari *port* yang tertera pada tabel. Kolom selanjutnya menunjukan jenis *port*-nya yang terdapat pada *system*. Kemudian keterangan status *port* tersebut (LISTEN). Baris pertama berjenis *protocol* tcp dengan pada *port* terbuka di sistem yaitu *port* 2222 (ssh) dan baris kedua *port* 80 (http). Adanya kesamaan antara keduanya menunjukan bahwa hasil dari *port scanning* pada skenario pertama berhasil dengan baik sesuai dengan tujuan aslinya yaitu *server* utama.



Gambar 4 *Hasil Port Pada Server Pada Skenario Pertama*

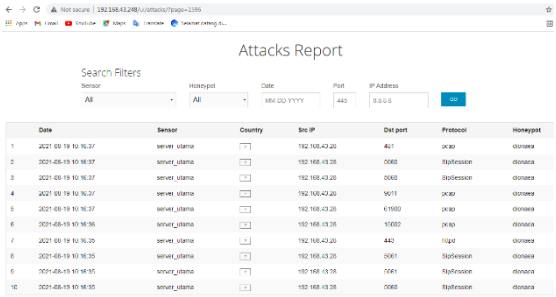
Perintah yang sama terjadi pada *virtual mechine server* utama dengan ip target 192.168.43.212 dan pemilihan *intense scan* menghasilkan informasi nama *host* yaitu *'server_utama (192.168.43.212)'* pada bagian kiri kolom hasil *scanning*. Sedangkan hasil *scanning* dapat dilihat pada sebelah kanan tepatnya pada *tab port/hosts* dimana dari hasil tersebut diketahui bahwa *protocol* yang terbaca atau yang berstatus *(state) open*, semuanya berjenis *protocol* TCP.



Gambar 5 *Pengujian Port scanning Pada Skenario Kedua*

Adapun *port* yang tercantum antara lain *port* 21 (ftp), *port* 23 (telnet), *port* 42 (nameserver), *port* 53 (domain), *port* 80(http), *port* 135 (msrpc), *port* 443(http), *port* 445 (Microsoft-ds), *port* 1433 (ms-sql-s), *port* 1723 (pptp), *port* 2222 (ssh), *port* 5060 (sip), dan *port* 5061 (sip-tls). Beberapa *port* tersebut sengaja dibuat oleh honeypot untuk mengelabui *attacker* seolah – olah semua *port* yang terbuka milik *server* utama yang bebas untuk diserang.

Adanya hasil tersebut membuktikan bahwa serangan *port scanning* dengan skenario kedua menggunakan *tool* Nmap telah berhasil dilakukan. Hal ini karena hasil tampilan dari *tool* Nmap dan *server* sama.



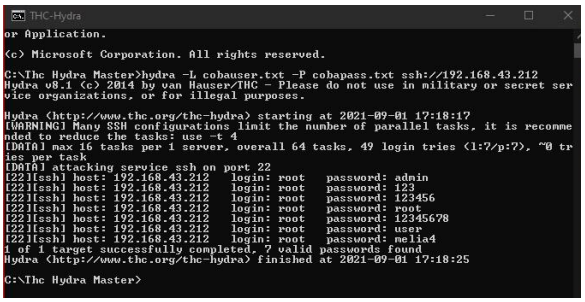
Gambar 6 *Tampilan Attack Report Serangan Port scanning*

Berdasarkan Gambar 6 yang terbagi menjadi beberapa kolom, maka dapat diketahui informasi antara lain seperti pada kolom pertama (*date*) yaitu tanggal dan waktu *attacker* melakukan serangan, kemudian pada kolom kedua adalah jenis sensor yang diserang, kolom selanjutnya IP *Address attacker*, kemudian dilanjut dengan informasi *port* berapakah yang menjadi tujuan serangan serta jenis *protocol*-nya, kolom terakhir memberikan informasi tentang jenis honeypot yang menangkap serangan tersebut.

Terdapat beberapa contoh *port* yang dapat tercatat di *attack report* seperti *port* 443 (httpd) dan *port* 5061 (SipSession) yang diserang oleh *attacker* dengan IP *Address* 192.168.43.28 pada tanggal 19 Agustus 2021 jam 10:16 dengan jenis honeypot yang dapat menangkap serangan tersebut adalah dionaea. Sedangkan cowrie hanya dapat menangkap serangan yang menyerang *port* 22 (ssh). Dari hasil tersebut maka tingkat efektifitas dionaea dalam mendeteksi serangan *port scanning* adalah 92,8% sedangkan cowrie 7,2% dimana rincian tersebut dihitung berdasarkan jumlah *port* yang masuk pada MHN dibandingkan dengan jumlah *port* yang tercantum pada Nmap.

4.2 Pengujian *Bruteforce* SSH

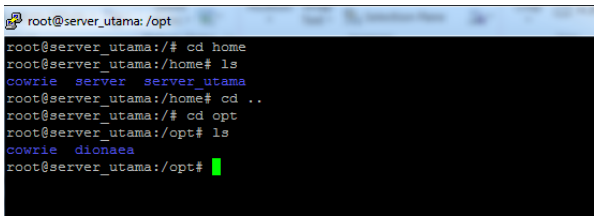
Pengujian kedua yaitu pengujian menggunakan serangan *Bruteforce* SSH sebagai bahan uji pada masing – masing skenario.



Gambar 7 *Tampilan THC Hydra Saat Penyerangan Skenario Kedua*

Pengujian kedua adalah serangan *Bruteforce* SSH menggunakan THC Hydra seperti pada Gambar 7. *Script* atau perintah yang digunakan untuk melakukan penyerangan adalah '*hydra –L cobauser.txt –P cobapass.txt ssh://192.168.43.212*'. Sesuai dengan perintahnya bahwa penyerang sedang mencoba melakukan serangan *Bruteforce* SSH dengan menggunakan *file 'cobauser.txt'*, dimana *file* tersebut merupakan data *list password* dan *username* dari *attacker* yang sengaja dibuat untuk melakukan serangan *Bruteforce* SSH.

Hasil THC Hydra menampilkan *''[22][ssh] host:192.168.43.212 login:root password:admin''* dimana dari pernyataan tersebut dapat didapatkan informasi seperti *host* yang menyatakan ip *address* target, *login* yang menyatakan *username* dan *password* yang berisi *password* yang mungkin cocok. Berdasarkan hasil yang tertera pada layar THC Hydra bahwa ada 1 login dan 7 kemungkinan *password* yang cocok. Adapun 1 login tersebut adalah root dan 7 *password* tersebut antara lain seperti admin, 123, 123456, root, 12345678, user, dan melia4.

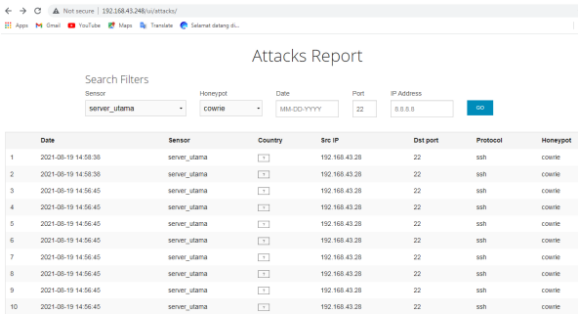


Gambar 8 *Tampilan SSH Client Sah*

Pada dasarnya *client* yang sah akan menggunakan *username* dan *password* yang sudah diketahui bersama antara *server* dan *client* tersebut, dan dalam hal ini *client* yang sah menggunakan '*server*_utama' (root@server_utama) sebagai *username* dan 'melia4' sebagai *password*. Pengujian pertama untuk memastikan bahwa yang diakses adalah *system* asli dengan mengetikan beberapa perintah, perintah pertama yaitu *'cd home'* menggunakan akses *root*. Hasilnya terdapat 3 folder yaitu cowrie, server dan server_utama. Perintah selanjutnya adalah *'cd ..'* untuk kembali ke menu sebelumnya. Perintah ketiga *'cd opt'* untuk membuka *folder* opt dilanjutkan perintah *'ls'*. Hasil yang muncul terdapat 2 folder yaitu cowrie dan dionaea. Hal tersebut menunjukan bahwa *client* tersebut sah.

Gambar 9 *Tampilan SSH Attacker Pada Honeypot*

Gambar 9 merupakan tampilan saat *attacker* berhasil *login* dengan bantuan THC Hydra. Untuk tahu lebih jauh bagaimana respon dari honeypot, maka selanjutnya penulis berusaha masuk dengan mode *root* serta *password* 'admin' atau sesuai dengan hasil THC Hydra sebelumnya. Perbedaan pertama yang dapat dilihat dari akses yang sebelumnya sudah dilakukan adalah nama *username* yang muncul bukanlah '*server_utama*' (root@server_utama) melainkan '*server*' (root@server). Perintah yang akan diberikan pada *system* sama dengan sebelumnya, namun hasil akses *system* ini berbeda dengan akses *system* sebelumnya. Perbedaan kedua, saat *attacker* membuka *folder 'home'*, sistem hanya menampilkan kata *'richard'*, dimana *folder* ini sama sekali tidak ada dalam *server* asli. Hal tersebut juga sama halnya ketika membuka *folder 'opt'* tetapi sistem tidak menampilkan isinya. Hal tersebut mengartikan bahwa adanya keterbatasan *attacker* ketika sudah masuk honeypot, dimana *attacker* hanya bisa *login* tetapi tidak dapat merubah *file*/dokumen yang ada didalam sistem tersebut

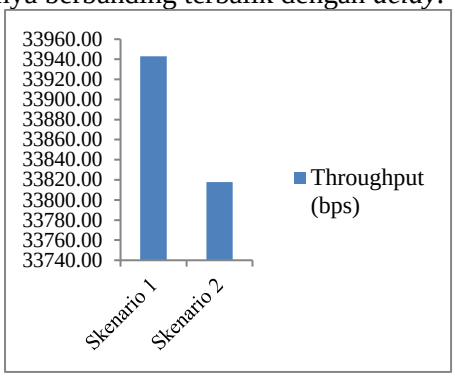


Gambar 10 *Attack report Serangan Bruteforce SSH*

Pada dasarnya, serangan *bruteforce* SSH adalah serangan yang mencoba untuk masuk ke *server* dengan perkiraan *username* dan *password* yang benar secara terus menerus. Dalam penelitian ini terdapat top *password* dan *top user*. *Top user* ditempati dengan nama *username* '*server*_utama' sedangkan *top password* nya adalah 'melia4'. Dari hasil tersebut maka tingkat efektifitas dionaea dalam mendeteksi serangan *bruteforce* adalah 0% sedangkan cowrie 100% dimana rincian tersebut dihitung berdasarkan jumlah *port* yang masuk pada MHN dibandingkan dengan jumlah *port* yang tercantum pada Nmap.

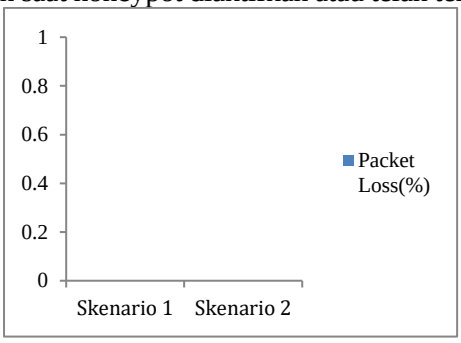
4.3 Pengujian *Denial of service*

Hasil pengujian *denial of service* terdiri dari QoS dan penggunaan CPU serta *Memory*. Pada skenario pertama menghasilkan nilai rata – rata *throughput* sebanyak 33942,94 bit/s sedangkan pada skenario kedua menghasilkan nilai rata – rata *throughput* 33817,86 bit/s. Hasil *throughput* dari kedua skenario terdapat anomali dimana nilai *throughput* seharusnya berbanding terbalik dengan *delay*.



Gambar 11 *Grafik Throughput Serangan DoS*

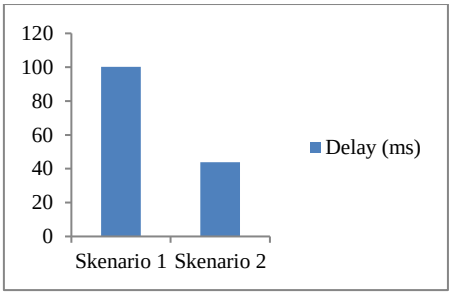
Pada dasarnya serangan yang masuk lewat honeypot mempunyai jalan sendiri karena serangan tersebut bertujuan ke *server* honeypot sehingga jalan yang dilaluinya pun berbeda sehingga *throughput* pada honeypot seharusnya tinggi. Hal tersebut dapat terjadi karena beberapa faktor seperti *software* yang *error*. Adapun penurunan *throughput* menunjukan bahwa *server* mengalami penurunan konektivitas jaringan pada saat honeypot terpasang. Hal ini dapat terjadi ketika *server* sudah sibuk menerima paket TCP yang dikirimkan oleh *attacker* dan aktivitas *client* yang mengakses web *server*. Selain aktivitas *attacker* dan *client*, aktivitas *server* utama pada kondisi kedua (honeypot aktif) yang mengharuskan halaman *dashboard server* MHN terbuka membuat layanan menjadi padat dan pada akhirnya koneksi jaringan menurun saat honeypot diaktifkan atau telah terpasang.



Gambar 12 *Grafik Packet Loss Serangan DoS*

Kondisi pertama (tanpa honeypot) nilai *packet loss* yang dihasilkan adalah 0% begitupun dengan nilai *packet loss* pada kondisi kedua (dengan honeypot). Nilai 0% pada *packet loss* menunjukan bahwa tidak ada paket data yang hilang saat proses pengiriman data berlangsung sehingga masuk pada kategori sangat bagus dengan nilai indeks 4.

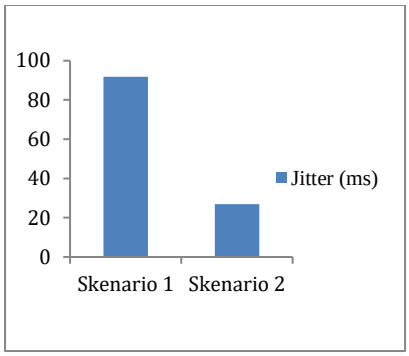
Packet loss dapat terjadi ketika *server* dalam keadaan sibuk membalas paket data yang dikirimkan oleh *attacker* sebelumnya. Sehingga pada saat *client* mengakses web *server*, terjadi kemacetan data dan *server* yang seharusnya bekerja melayani *client* berubah menjadi melayani permintaan *attacker*. Banyaknya layanan data yang dikirimkan oleh *attacker* dipengaruhi oleh *thread* yang telah diatur melalui *tool* LOIC sebelumnya. Semakin banyak *thread* yang dimasukan maka akan semakin banyak pula paket data yang dikirmkan kepada *server*. Kecepatan pengiriman juga dapat mempengaruhi banyaknya data yang masuk. Semakin cepat paket data dikirimkan maka data yang masuk juga akan semakin banyak.



Gambar 13 *Grafik Delay Serangan DoS*

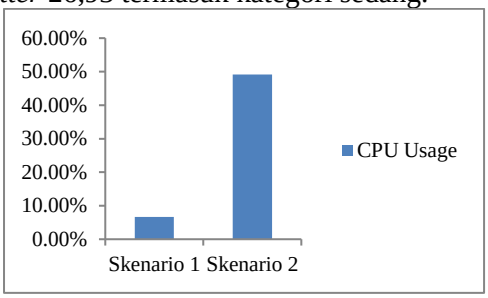
Kondisi pertama (tanpa honeypot) mendapatkan hasil dengan nilai *delay* sebesar 100,31 ms dan termasuk pada kategori sangat bagus dengan indeks 4. Kondisi kedua mendapatkan hasil dengan nilai *delay* sebesar 43,90 ms dan termasuk pada kategori sangat bangus dengan indeks 4. Terjadi penurunan *delay* dari kondisi pertama ke kondisi kedua. Hal ini menunjukan honeypot mempengaruhi waktu pengiriman paket. Pada dasarnya serangan DoS berorientasi untuk membanjiri layanan data dengan memakan bandwidth yang tersedia dalam sistem.

Pada skenario 1 (tanpa honeypot), paket data antara *client* dengan *attacker* bergabung menggunakan satu jalur yang sama untuk mencapai *server* dimana permintaan dari *attacker* lebih banyak dari permintaan *client* sehingga permintaan *client* yang sah menjadi terhambat. Sedangkan pada skenario 2 (dengan honeypot) dimana honeypot membuat *server* tiruan sendiri. Cara kerja honeypot yang membelokan serangan yang masuk dari *server* utama ke sistem honeypot sehingga jalur yang digunakan antara *client* dengan *attacker* berbeda. Artinya serangan DoS yang sebelumnya memenuhi *server* menjadi teralihkan ke jalur honeypot dan membuat *server* utama hanya melayani permintaan dari *client* saja. Sehingga *delay* pada skenario 1 (tanpa honeypot) lebih tinggi dibandingkan skenario 2 (dengan honeypot).



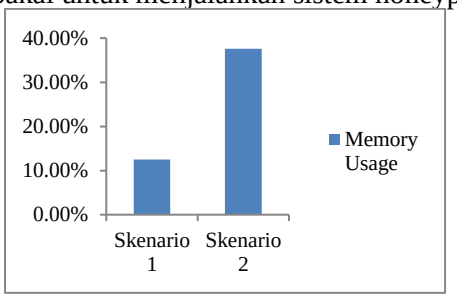
Gambar 14 *Grafik Jitter Serangan DoS*

Jitter digunakan untuk menentukan variasi *delay*. Jika nilai *delay* semakin tinggi maka nilai *jitter* juga akan semakin tinggi pula. Kondisi pertama didapatkan nilai *jitter* sebanyak 91,76 ms dan termasuk pada kategori bagus sedangkan pada kondisi kedua nilai *jitter* 26,93 termasuk kategori sedang.



Gambar 15 *Grafik CPU Usage Serangan DoS*

Penggunaan CPU saat terjadi serangan DoS pada kondisi pertama (tanpa honeypoy) mencapai 6,66%. Penggunaan CPU ini lebih rendah dibandingkan saat terjadi serangan DoS pada saat kondisi *server* terpasang honeypot atau kondisi kedua dimana grafik menunjukan angka 49%. Pengaruh honeypot pada penggunaan CPU terjadi karena didalam *server* utama terpasang honeypot dimana hal tersebut tentu akan menambah penggunaan CPU mengingat processor yang digunakan oleh *server* utama sama dengan honeypot. Jadi *processor* yang sebelumnya tidak terpakai sepenuhnya, ketika honeypot dijalankan akan dipakai untuk menjalankan sistem honeypot.



Gambar 16 *Grafik Memory Usage Serangan DoS*

Penggunaan *Memory* pada saat pengujian serangan DoS dengan kondisi pertama (tanpa honeypot) mencapai 12,48%. Penggunaan *memory* tersebut lebih rendah dibandingkan saat kondisi kedua (dengan honeypot). Pada saat *server* sudah terpasang honeypot rata – rata penggunaan *Memory* mencapai

37,6%. Hal tersebut menunjukan bahwa honeypot berpengaruh terhadap penggunaan *memory* pada *server*. *Memory* yang naik disebabkan karena honeypot dipasang pada *server* yang sama sehingga *memory* yang sebelumnya hanya digunakan untuk *server* utama saja, karena bertambahnya honeypot maka dibutuhkan pula ruang untuk membangun sistem tersebut. Hal tersebut sejalan dengan penggunaan CPU pada *server*. Semakin tinggi penggunaan CPU maka *Memory* yang digunakan juga akan semakin banyak.

5 SIMPULAN

1. Tingkat keberhasilan honeypot dalam mengidentifikasi serangan yaitu dionaea sebanyak 92,8% untuk serangan *port scanning* dan 0% untuk serangan *Bruteforce* SSH. Sedangkan cowrie sebanyak 7,2% untuk serangan *port scanning* dan 100% untuk serangan *Bruteforce* SSH.
2. Kinerja *server* sebelum dipasangi honeypot dilihat dari nilai QoS nya yaitu *throughput* dikategorikan sangat bagus karna nilai indeksnya 4, *packet loss* dikategorikan sangat bagus karena tidak ada paket data yang hilang, *delay* dikategorikan sangat bagus dengan nilai indeks 4, dan *jitter* dikategorikan sedang dengan nilai indeks 2. Sedangkan jika dilihat dari nilai *resource usage*-nya, nilai CPU *usage* 6,66% dan *memory usage* 12,48%.
3. Kinerja *server* setelah dipasangi honeypot dilihat dari nilai QoS nya yaitu lebih baik dimana *throughput* dikategorikan sangat bagus dengan indeks 4, akan tetapi terdapat anomali karena nilainya menurun, *delay* menurun dan dikategorikan sangat bagus dengan nilai indeks 4, *jitter* juga menurun dimana kategorinya bagus dengan indeks 3. Sedangkan nilai CPU *usage* naik menjadi 12,48% dan *memory usage* menjadi 37,6%.

KEPUSTAKAAN

[1] "Riset: 64% Penduduk Indonesia Sudah Pakai Internet," *KumparanTECH*, 2020. https://kumparan.com/kumparantech/riset-64-penduduk-indonesia-sudah-pakai-internet-1ssUCDbKILp (diakses Jul 10, 2020).

[2] "Rekap Serangan Siber (Januari – April 2020)," *Security Advisory BSSN*. https://bssn.go.id/rekap-serangan-siber-januari-april-2020/ (diakses Jan 20, 2021).

[3] Satrio Wardanu, "Disusun Oleh : Disusun Oleh :," *Pelaks. Pekerj. Galian Divers. Tunn. Dengan Metod. Blasting Pada Proy. Pembang. Bendungan Leuwikeris Paket 3, Kabupaten Ciamis Dan Kabupaten Tasikmalaya Jawa Barat*, hal. 1–147, 2018.

[4] I. A. Romadhan, S. Syaifudin, dan D. R. Akbi, "Implementasi Multiple Honeypot pada Raspberry Pi dan Visualisasi Log Honeypot Menggunakan ELK Stack," *J. Repos.*, vol. 2, no. 4, hal. 475, 2020, doi: 10.22219/repositor.v2i4.114.

[5] S. Z. Melese dan P. S. Avadhani, "Honeypot System for Attacks on SSH Protocol," *Int. J. Comput. Netw. Inf. Secur.*, vol. 8, no. 9, hal. 19–26, 2016, doi: 10.5815/ijcnis.2016.09.03.

[6] Khairunnisa dan Sutarti, "Perancangan Dan Analisis Keamanan Jaringan Nirkabel Dari Serangan Ddos (Distributed Denial of Service) Berbasis Honeypot," *J. PROSISKO*, vol. 4, no. 2, hal. 8, 2017.

[7] F. B. Alfisyah, "IMPLEMENTASI MONITORING AUTONOMOUS SPREADING MALWARE DI ITS-NET DENGAN DIONAEA DAN CUCKOO," hal. 1–27, 2015.

[8] D. Oktavianto, P. Tamba, dan S. Wibowo, "Cyber Defense Bulletin Fourth Edition," *Cyber Defense Community*, hal. 128, 2018.

[9] Do Son, "Do Son." https://securityonline.info/modern-honey-network/.

[10] S. Pavitra, "Popular *Tool*s for Brute-force Attacks [Updated for 2019]," 2019. https://resources.infosecinstitute.com/popular-*tool*s-for-brute-force-attacks/#gref (diakses Jun 29, 2020).

[11] N. I. Prasetya, "MEREDUKSI SERANGAN DENIAL OF SERVICES TERDISTRIBUSI PADA LINUX VIRTUAL *SERVER* MENGGUNAKAN HONEYPOT," vol. XI, 2016.

[12] R. A. Habsoro, N. R. Rosyid, dan H. N. Isnianto, "Implementasi Honeypot Untuk Mengungkap Pola Port Scaninng Attack Dalam Jaringan," *Teknol. Jar.*, no. September, hal. 2015, 2015.

[13] Anonymous, "HTG Menjelaskan: Apa itu Port Scanning?," *If-Koubou*. https://id.if-koubou.com/articles/how-to/htg-explains-what-is-port-scanning.html (diakses Jul 11, 2020).

[14] N. Al-munawar dan A. Sediyono, "Karakteristik Konsumsi Daya Komputer Dengan Perubahan Tingkat Serangan Distributed Denial of Service (Ddos)," *Semin. Nas. Cendekiawan Ke 3*, hal. 141–147, 2017.

[15] "*Memory* Utilization," *Broadcom*. https://techdocs.broadcom.com/us/en/ca-enterprise-software/it-operations-management/performance-management/3-5/using/performance-metrics/*memory*-utilization.html (diakses Feb 08, 2021).

[16] ETSI, "Telecommunications and Internet Protocol Harmonization Over Networks

(TIPHON); General aspects of Quality of Service (QoS)," *Etsi Tr 101 329 V2.1.1*, vol. 1, hal. 1–37, 1999, [Daring]. Tersedia pada: http://www.etsi.org/deliver/etsi_tr/101300_101399/101329/02.01.01_60/tr_101329v020101p.pdf.

ISSN No. 2502-8782