DDoS Attacks Detection in Cloud Computing

Muhammad Daffa Alfaridzi, 18217013, Sistem Teknologi Informasi

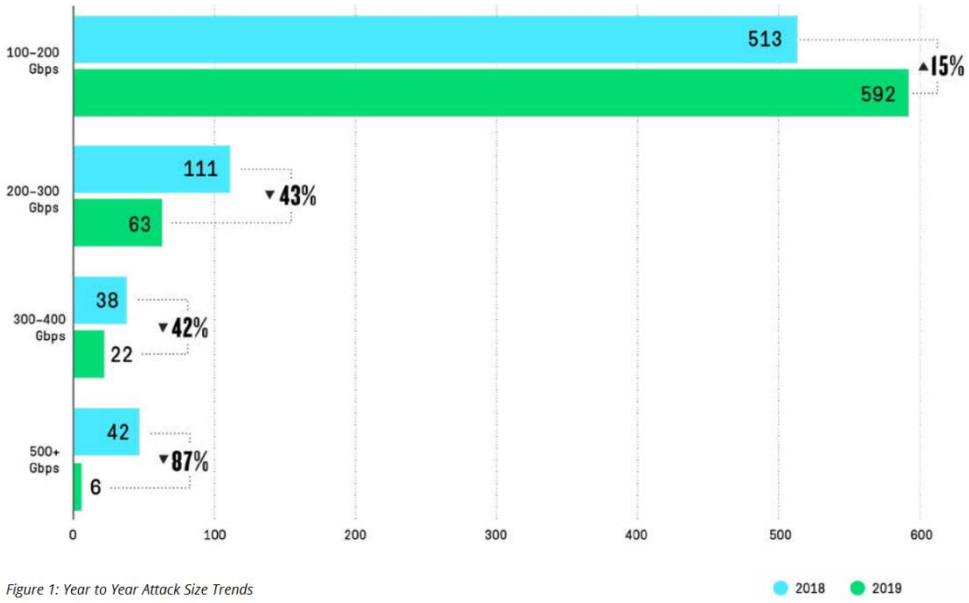
Makalah Keamanan Informasi

***Abstract*—Saat ini, Penggunaan *cloud computing* menjadi semakin populer zaman sekarang, baik untuk media penyimpanan, jaringan, atau pun komputasi sesuai dengan namanya. Sumber daya tersebut dapat disediakan oleh *private, public,* ataupun *hybrid* cloud. Namun, karena sifatnya yang terdistribusi, ia dapat dengan mudah dieksploitasi oleh serangan *Distributed Denial of Service* (DDoS). Dalam serangan DDoS, *user* dicegah menggunakan *cloud resources*. Setiap penyedia layanan *cloud* memiliki caranya masing-masing dalam melakukan pendeteksian *attack*. Dalam tulisan ini, berbagai mekanisme deteksi serangan DDoS pada *cloud* akan ditinjau. Teknik deteksi yang dibahas adalah *victim-end detection approach* dan *Protocol-specific Multi-threaded Network Intrusion Detection System* (PM-NIDS).**

Keywords*—makalah, keamanan informasi, *cloud computing, DDoS Attack, Detection

I. PENDAHULUAN

Berdasarkan sebuah survei mengenai *cloud computing* pada tahun 2019, dari 786 *enterprises* yang menjadi responden, 94 persen (≈ 739 *enterprises*) diantaranya menggunakan *cloud* [1]. Seiring dengan jumlah pengguna yang tinggi, begitu pun dengan jumlah *attacks* yang ada. Berdasarkan laporan yang diberikan oleh NetScout pada Februari 2020, mereka menemukan pada paruh kedua 2019, serangan DDoS dilakukan sebanyak 8,4 juta kali, sekitar 23 ribu kali dalam satu hari atau 16 kali setiap menit. Jumlah serangan tersebut meningkat sebanyak 16 persen dari jumlah global pada paruh kedua 2018 [2]. *Cloud service* sendiri dituliskan NetScout menerima sebanyak 1,176 juta serangan (14 persen) dari total serangan DDoS yang dilancarkan [2]. Meski jumlah serangan mengalami peningkatan, namun tren jumlah serangan DDoS berukuran melebihi 200Gbps mengalami penurunan sebanyak 54 persen (Gambar 1) [2].



Gambar 1 Tren ukuran serangan DDoS paruh kedua 2018 dan 2019 [2]

Serangan pada infrastruktur layanan publik meningkat cukup drastis, dilaporkan 51 persen dari penyedia layanan pada paruh kedua 2019, meningkat apabila dibandingkan dengan paruh kedua 2018 yang 'hanya' 38 persen [2]. Selain itu, serangan DDoS diyakini menjadi *top concern* yang akan dihadapi pada tahun 2020 [2]. Berbicara mengenai ukuran serangan, serangan terbesar pada paruh kedua 2019 berukuran 622Gbps [2]. Hingga saat ini, serangan DDoS terbesar terjadi pada tahun 2018, ketika Github, yang menjadi *code repository* paling terkenal di dunia [3], harus di-*take off* karena adanya serangan DDoS sebesar 1,3Tbps [4].

Berdasarkan fakta-fakta di atas, kemampuan untuk mendeteksi serangan DDoS menjadi begitu kritis terhadap keberjalanan layanan *cloud*. Sudah banyak mekanisme yang diajukan dan eksis untuk melakukan deteksi. Pada makalah ini, penulis akan memaparkan tiga mekanisme; *victim-end detection approach, Protocol-specific Multi-threaded Network Intrusion Detection System (PM-NIDS),* dan *fuzzy logic-based detection*.

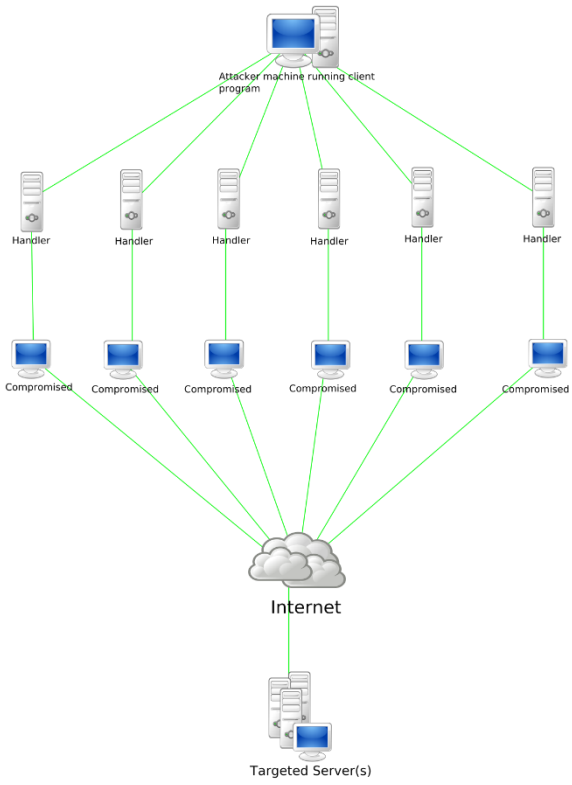
II. DASAR TEORI

Sebelum membahas lebih jauh mengenai mekanisme pendeteksian serangan DDoS, kita perlu memahami apa itu DDoS. Maka dari itu, penulis akan menjelaskan DDoS secara singkat.

A. *DDoS*

Distributed Denial-of-Service Attack atau serangan DDoS adalah sebuah serangan yang bertujuan untuk mendistrupsi *traffic* dari sebuah *server*, layanan, atau jaringan, dengan memberikan sasaran atau infrastruktur yang ada disekitarnya banjiran *traffic* internet [5]. Serangan DDoS umumnya dijalankan secara efektif dengan mengambil alih sejumlah sistem komputer dan mesin lainnya (seperti perangkat IoT) [5]. Pengambilalihan dilakukan dengan menginjeksi perangkat yang dibutuhkan untuk melakukan serangan dengan *malware*, yang mengubah perangkat menjadi *bot* [5]. Setelahnya, *attacker* (pelaku serangan DDoS) akan menginisialisasi sebuah *remote control* terhadap semua perangkat yang sudah jadi *bot*, bernama *botnet* [5].

Setelah sebuah *botnet* berhasil dibentuk, *attacker* akan memberikan sebuah alamat IP untuk dijadikan sasaran serangan oleh botnet dan membuat seluruh *bot* yang berada pada kendali *botnet* merespon dengan memberikan *requests* kepada IP tersebut (Gambar 2) [5]. Hal tersebut berpotensi membuat kapasitas *server* sasaran menjadi *overflow* yang berujung pada *denial-of-service*. Karena seluruh *bot* adalah perangkat internet yang *legitimate*, membedakan *attack traffic* dengan *normal traffic* menjadi sulit [5].

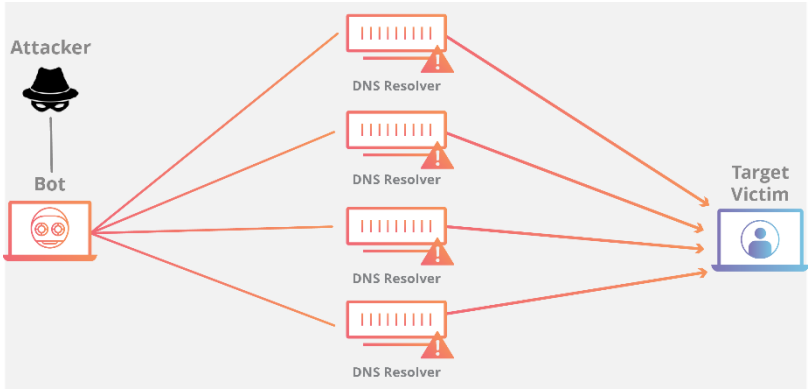


Gambar 2 Ilustrasi serangan DDoS [5]

Serangan DDoS dapat dibagi kedalam tiga jenis [5][6]:

1) *Volume-Based Attacks*

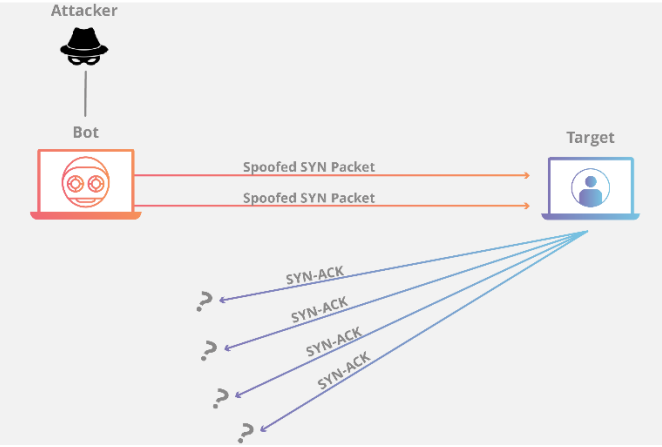
   Tujuan dari serangan ini adalah untuk memenuhi *bandwidth* di antara sasaran serangan dan internet yang digunakan dengan menciptakan *internet traffic* yang masif [5]. Satuan ukur serangan ini adalah *bits per second* (bps) [6]. Contoh dari *volume-based attacks* adalah *DNS Amplification*, yaitu serangan dengan mengirim request ke sebuah *DNS server* yang terbuka dengan menggunakan *spoofed IP address* (Mengganti IP *sender* pada *packets header* dengan IP dari sasaran [7]), sehingga sasaran serangan menerima *requests responds* dalam jumlah yang besar dari *server* (Gambar 3) [5].



Gambar 3 Ilustrasi *DNS Amplification* [5]

2) *Protocol Attacks*

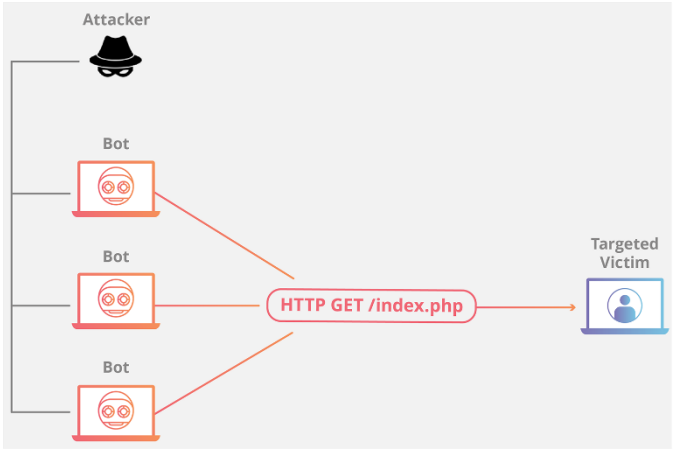
   Tujuan dari serangan ini adalah untuk menimbulkan gangguan pada *server* dengan menggunakan seluruh kapasitas *state table* yang tersedia dari *web server* atau *resources* seperti *firewalls* dan *load balancers* [5]. Satuan ukuran dari serangan ini adalah *packets per second* (pps) [6]. Contoh dari serangan ini adalah *SYN Flood*, yaitu serangan yang mengeksploitasi *three-way handshake* dari rangkaian koneksi TCP dengan mengirimkan beberapa *SYN requests* melalui sebuah *spoofed IP address* kepada target atau dengan tidak merespon SYN-ACK yang dikirimkan target atas *SYN requests* yang dikirimkan (Gambar 4) [6].



Gambar 4 Ilustrasi *SYN Floods* [5]

3) *Application Layer Attacks*

Tujuan dari serangan ini adalah untuk membuat *web server* mengalami *crash* [6] dengan menyerang *layer* dimana *web pages* ada dan mengirimkan *HTTP requests* [5]. Sebuah *HTTP requests* berpotensi menjadi cukup berat untuk dieksekusi karena adanya kemungkinan untuk melakukan *loading* terhadap beberapa *files* sekaligus dan me-*run database queries*. Contoh dari *application layer attacks* adalah *HTTP Flood* (Gambar 5) [5].



Gambar 5 Ilustrasi *HTTP Flood* [5]

III. RELATED WORK

Terdapat beberapa jurnal yang juga membahas mekanisme pendeteksian serangan DDoS di Cloud:

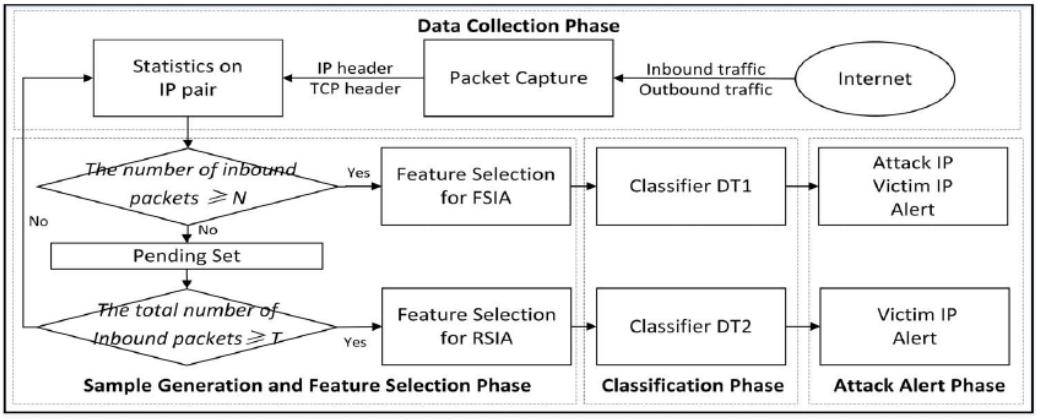
1) Sang *et al*. [8] menjabarkan pendeteksian serangan DDoS dengan menggunakan *traffic matrix* yang sudah dioptimasi
2) Thapngam *et al*. [9] menjabarkan pendeteksian serangan DDoS dengan menggunakan analisis *traffic pattern*
3) Xiao *et al.* [10] menjabarkan pendeteksian serangan DDoS pada sebuah *data center* dengan menggunakan analisis korelasi
4) Chaitanya *et al*. [11] menjabarkan sebuah *SDN Framework* bernama FlowTrApp yang menggunakan *traffic flow rate* dan *traffic flow duration* untuk mendeteksi serangan DoS
5) Anteneh *et al*. [12] menjabarkan sebuah model *hybrid statistical* untuk mendeteksi serangan DDoS dengan menggunakan matriks kovariansi dan entropi yang akan menemukan pola serangan dan mengelompokkannya ke dalam sebuah serangan DDoS.
6) Habib *et al*. [13] menjabarkan pendeteksian serangan DDoS menggunakan Snort dan mitigasi serangan dengan *IP Filtering.*

IV. MEKANISME DETEKSI SERANGAN DDOS

Pada bagian ini, penulis akan membahas mekanisme *victim-end detection approach, Remote Triggered Black Hole (RTBH),* dan *Protocol-specific Multi-threaded Network Intrusion Detection System (PM-NIDS)* secara menyeluruh.

A. *Victim-end detection approach* [14]

Victim-end detection approach adalah *TCP-based DDoS attacks detection system* yang menggunakan *decision tree technique* untuk mendapatkan *attack detection rate* hingga lebih dari 99% dan *false alarm rate* yang lebih rendah dari 1%. Sistem ini bermaksud untuk mampu mendeteksi serangan DDoS yang masuk dalam kelompok *fixed source IP attacks* (FSIA) ataupun *random source IP attacks* (RSIA) secara *real-time* dengan mengekstraksi *important features* setiap detiknya dari *TCP traffic flows*, baik yang *inbound* ataupun *outbound*. Secara keseluruhan, arsitektur dari sistem ini dipaparkan oleh Gambar 6.



Gambar 6 Arsitektur dari sistem *victim-end detection approach* [14]

Victim-end detection approach terdiri dari empat fase utama:

1. *Data collection phase*

   Pada fase ini, sistem akan memanfaatkan *packet sniffer* untuk mendapatkan setiap *packet* yang ada di *TCP traffic flows*. Setelah memisahkan *TCP/IP header* dari *packets* yang didapat, sistem akan membagi *packets* tersebut sesuai dengan pasangan *IP address*-nya (*IP address* dari *local host* (*local IP*) dan *IP address* dari *remote host* yang berkomunikasi dengan *local host* (*remote* IP)) dan menghitung jumlah dari *inbound packets* dari setiap *IP pair* setiap detiknya.

2. *Sample Generation and Feature Selection Phase*

   Terdapat *sample generation* dan *feature selection* yang dibedakan untuk *fixed source IP address* (FSIA) dan *random source IP address* (RSIA):

   1) *Sample Generation*

      Hal yang menjadi dasar *detection system* yang baik adalah pendeteksian *traffic flows* yang abnormal. Untuk itu, dibutuhkan sebuah parameter N, yaitu batasan dari jumlah *inbound packets* per detik sebuah *IP pair*. Apabila nilai N terlewati, sistem akan memulai proses pendeteksian untuk FSIA.

      Parameter N hanya dapat digunakan untuk pendeteksian FSIA, dikarenakan peningkatan jumlah *inbound packets* secara massif dalam waktu singkat belum tentu merupakan sebuah serangan RSIA. Selain itu, RSIA juga terbilang sulit untuk dideteksi karena cukup mirip dengan

legitimate traffic. Untuk mendeteksi RSIA, dibutuhkan parameter T, yaitu jumlah *inbound packets* dari *IP pair* yang lolos dari parameter N.

2) *Feature Selection*

a) FSIA:

Berdasarkan analisis terhadap karakteristik dari protokol TCP dan serangan berbasis TCP, dipilih dua kategori fitur: 15 fitur statistik dasar dan 16 fitur rasio yang bisa memberikan *insight* mengenai terjadinya *malicious traffic* (Gambar 7).

No.	Feature	Description
1	in_pps	number of inbound TCP packets per sec.
2	out_pps	number of outbound TCP packets per sec.
3	syn_in_pps	number of inbound syn packets per sec.
4	synack_out_pps	number of outbound syn-ack packets per sec.
5	ack_in_pps	number of inbound ack packets per sec.
6	ack_out_pps	number of outbound ack packets per sec.
7	push_in_pps	number of inbound push packets per sec.
8	push_out_pps	number of outbound push packets per sec.
9	fin_in_pps	number of inbound fin packets per sec.
10	fin_out_pps	number of outbound fin packets per sec.
11	rst_in_pps	number of inbound rst packets per sec.
12	rst_out_pps	number of outbound rst packets per sec.
13	other_in_pps	number of inbound non-flag packets per sec.
14	port_num_RIP	number of port used by remote IP
15	port_num_LIP	number of port used by local IP

No.	Feature
16	in_pps / (in_pps + out_pps)
17	syn_in_pps / in_pps
18	syn_in_pps / (syn_in_pps + syn-ack_out_pps)
19	syn_in_pps / (syn_in_pps + ack_in_pps)
20	ack_in_pps / in_pps
21	ack_in_pps / (ack_in_pps + ack_out_pps)
22	ack_in_pps / (ack_in_pps + rst_out_pps)
23	push_in_pps / in_pps
24	push_in_pps / (push_in_pps + push_out_pps)
25	push_in_pps / (push_in_pps + rst_out_pps)
26	push_in_pps / (push_in_pps + ack_in_pps)
27	rst_in_pps / in_pps
28	rst_out_pps / out_pps
29	fin_in_pps / in_pps
30	fin_in_pps / (fin_in_pps + fin_out_pps)
31	other_in_pps / in_pps

Gambar 7 Fitur untuk mendeteksi FSIA

Fitur-fitur pada gambar di atas berhubungan dengan *control flags* yang mereprestasikan status komunikasi antara *local host* dan *remote host* (e.g. SYN, ACK, PUSH, RST, FIN). Kita bisa ambil contoh, apabila ada serangan dalam bentuk *SYN Flood*, maka server sasaran akan menerima banyak *SYN packets*. Perbandingan antara jumlah *SYN packets* dengan jumlah total *inbound packets* per detik akan sangat tinggi.

b) RSIA:

Fitur untuk mendeteksi RSIA didesain sedemikian sehingga memiliki similaritas dengan jumlah dari *inbound packets*.

Didefinisikan 10 fitur yang simpel dan efektif, dimana fitur ke-*i* adalah jumlah dari *remote hosts* yang mengirim setidaknya (*i*/10)*N packets* dan kurang dari ((*i*+1)/10)*N packets* ke *local hosts*, untuk $i \in \{0, 1, 2, \ldots, 9\}$, dimana *N* adalah batas jumlah dari *inbound packets* per detik.

c) *Chi-squared test*

Jumlah fitur yang digunakan dalam pendeteksian memengaruhi efisiensi klasifikasi serangan. *Chi-squared test* dimanfaatkan untuk menghasilkan sebuah himpunan fitur yang mumpuni untuk melakukan klasifikasi antara traffic yang *legitimate* dan *malicious*. Untuk pendeteksian FSIA, akan dipilih sejumlah *k* fitur paling atas dari pengurutan oleh *chi-squared test* untuk mencapai klasifikasi yang lebih akurat dan cepat.

3. *Classification Phase*

Pada fase ini, terdapat dua *decision tree classifiers*, satu didesain untuk FSIA dan yang lainnya untuk RSIA, yang dilatih oleh data eksperimen terlebih dahulu. *Decision tree classifiers* ini akan melabelkan suatu *traffic flow* sebagai "normal" atau "serangan".

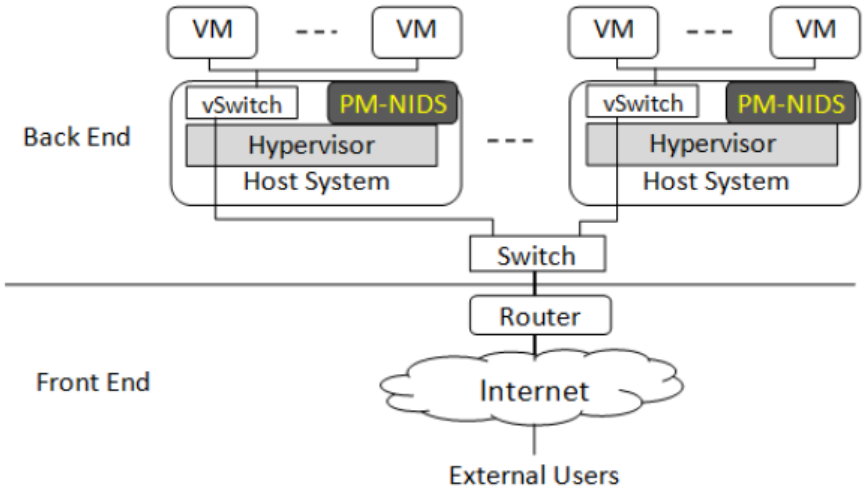
4. *Attack Alert Phase*

Berlaku untuk pendeteksian FSIA. Adanya *remote-local host IP Pair* memberikan kemampuan untuk mengeluarkan sebuah *alert* dengan memberikan *IP address* dari *malicious users* tersebut dan membuat sistem mempersiapkan mekanisme pertahanan yang tepat.

B. *Protocol-specific Multi-threaded Network Intrusion Detection System* (PM-NIDS) [15]

Protocol-specific Multi-threaded Network Intrusion Detection System atau disingkat PM-NIDS adalah sistem pendeteksian serangan DDoS yang membuat beberapa *threads* dan mengaplikasikan *selection* dan *classifiers* yang *protocol specific*. *Classifiers* yang digunakan pada sistem ini ada tiga, yaitu: *decision tree* dengan algoritma ID3, RandomForest, dan OneR, berdasarkan *network protocol*-nya. *Decision tree* menangani *continuous* dan *discrete attributes*. RandomForest menjadi

decision maker mengenai data yang masuk. OneR memberikan sebuah *rule* untuk setiap jenis protokol. *Rule* dengan *margin error* paling kecil akan terus menerus digunakan untuk klasifikasi. Desain dari *framework* PM-NIDS dapat dilihat pada Gambar 8:

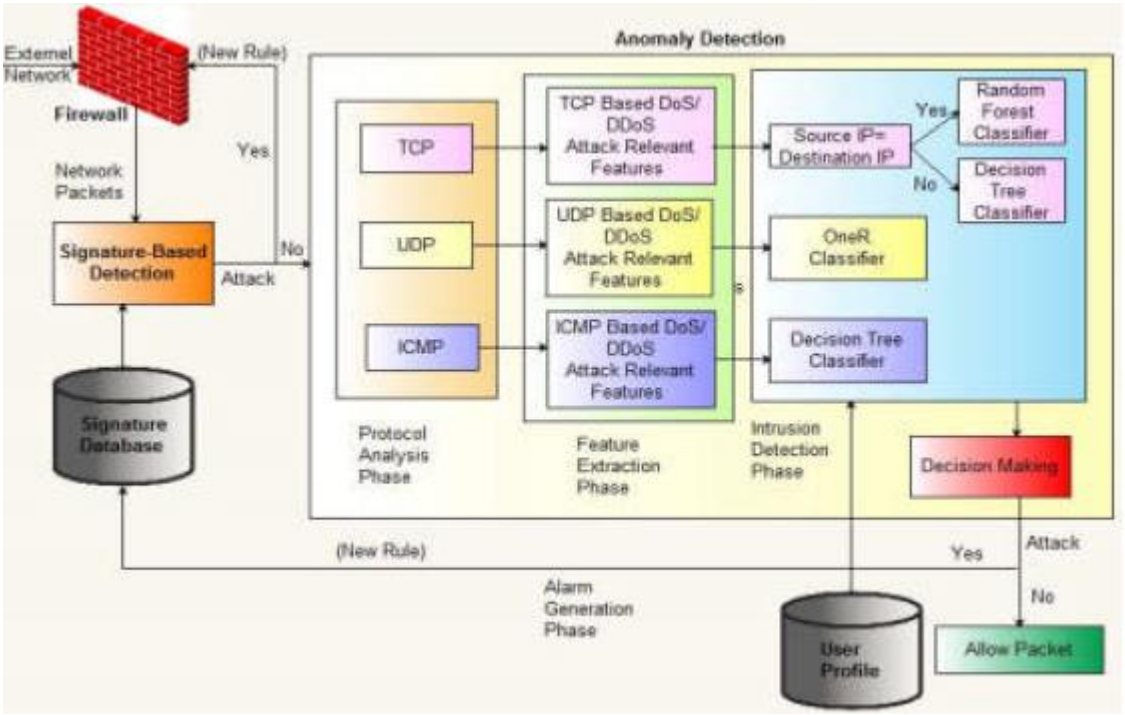


Gambar 8 Desain dan *deployment* dari PM-NIDS [15]

Seperti yang terlihat pada gambar di atas, *back-end* dari suatu *cloud* terdiri dari beberapa *physical servers*, yang masing-masingnya terdiri dari *multi-core processors*, Hypervisor, *host operating system*, *virtual machines*, dan *guest operating system*. Antar-*physical servers* dihubungkan oleh sebuah *virtual network*. PM-NIDS di-*deploy* di *backend* dari *cloud computing* agar mampu mendeteksi serangan internal ataupun eksternal pada *virtual network*. Selain itu, PM-NIDS juga bisa memonitor *traffic* dari beberapa *Virtual Machine* secara bersamaan.

Untuk melaksanakan *intrusion detection*, system mengombinasikan *firewall*, *signature-based detection*, dan *anomaly detection modules* (Gambar 9). *Firewall* akan bertindak sebagai garis pertahanan pertama dengan me-*reject packets* yang bersumber dari *malicious IP address* yang ada pada aturan *filter*-nya. Aturan *filter* akan selalu diperbarui sesuai dengan *alerts* yang dihasilkan oleh semua *intrusion detection system* di *cloud*. *Packets* yang lolos dari *firewall* akan melalui *signature-based detection* untuk disaring menjadi *malicious packets* dan *safe packets*. Aturan *filter* akan diperbarui dengan menambahkan *malicious IP address* yang ada pada *malicious packets*. Apabila sebuah *packet* dikategorikan sebagai *safe packets*, *packet* tersebut

akan dikirim ke *anomaly detection module* yang akan mendeteksi serangan yang tidak diketahui.



Gambar 9 *Working flow* dari PM-NIDS [15]

Anomaly detection module terdiri dari empat modul:

1. *Protocol analysis*

   Pada modul ini, *packets* yang diterima sistem akan dipisahkan sesuai dengan protocol dari *packets* tersebut, seperti TCP, UDP, dan ICMP. Setiap *packet* akan berada pada *queue* yang berbeda sesuai dengan protokolnya. *Queue* yang dibedakan akan membantu mempercepat pemrosesan *packets* dalam jumlah besar.

2. *Feature Extraction and Selection*

   Pada modul ini, atribut seperti *protocol*, *port number*, dan atribut *network* lain (e.g. *flag*, *source bytes*, *destination bytes*, *etc*.) akan diekstrak dari *packets*. Setelah diekstrak, atribut yang dianggap relevan terhadap serangan DDoS akan dipilih. Hal ini bertujuan untuk mengurangi *training* dan *testing time* pada *detection phase*. Sebagai tambahan, hal tersebut juga

membantu meningkatkan akurasi pendeteksian secara keseluruhan. Atribut yang dipilih akan dikirim ke fase selanjutnya untuk *intrusion detection*.

3. *Intrusion Detection Phase*

   Pada fase ini, sistem mengumpulkan *network behavioural data* dari waktu ke waktu dan melakukan tes statistik terhadap *network behaviour* yang sedang diperiksa untuk mengetahui apakah *behaviour* tersebut berbahaya. Untuk mendeteksi serangan DDoS berbasis TCP, sistem menggunakan RandomForest dan *decision tree*. Tiga tipe serangan DDoS berbasis TCP yang sistem perhatikan adalah *Neptune, Back,* dan *Land*. Jika *source IP address* dan *destination IP address* sama, maka pendeteksian dilanjutkan oleh RandomForest, bila tidak maka *decision tree* akan mengambil alih untuk melakukan *intrusion detection*. Serupa dengan serangan DDoS berbasis TCP, atribut dari *packets* UDP akan diberikan kepada OneR dan atribut dari *packets* ICMP akan menggunakan *decision tree* untuk mendeteksi serangan DDoS

4. *Alert Generation Phase*

   Pada fase ini, sistem meng-*generate alert* sebagai respons terhadap aktivitas yang intrusif di jaringan *cloud*. Jika serangan terdeteksi, maka *alert* akan disebarkan kepada *physical server* lain melalui komponen *framework* mereka. Sebagai tambahan, *intrusion signature* akan dihasilkan dan diberikan kepada Snort untuk pendeteksian dini di waktu mendatang. Selain itu, profil *user* (VM) juga diperbarui.

V. KESIMPULAN

Dengan berkembangnya penggunaan *cloud*, maka *security* yang melingkupinya pun harus terus berkembang. Ancaman terbesar yang dihadapi *cloud computing* diyakini adalah serangan DDoS, yang akan membuat *cloud* tidak mampu digunakan secara optimal. Untuk mengetahui apakah sistem *cloud* kita sudah dimasuki oleh *malicious packets* yang menjadi awal serangan DDoS, tentu kita memerlukan sistem pendeteksian serangan yang baik. Mekanisme sistem pendeteksian serangan DDoS bermacam-macam, *victim-end detection approach* dan *protocol-specified multithreads intrusion detection system* adalah contohnya. Tujuan dari kedua sistem tersebut ketika diimplementasikan pada *cloud* sama, yaitu mendapat *attack detection rate* yang tinggi dan *false alarm rate* rendah

REFERENSI

[1] “Rightscale 2019 State of The Cloud Report from Flexera™,” Flexera. Accessed: May 1, 2020. [Online]. Available: https://resources.flexera.com

[2] “2H 2019 NetScout Threat Intelligence Report,” NetScout. Available: https://www.netscout.com/sites/default/files/2020-02/SECR_001_EN-2001_Web.pdf

[3] G. Gousios, B. Vasilescu, A. Serebnik, A. Zaidman, *"*Lean GHTorrent: GitHub Data on Demand,*" The Netherlands: Delft University of Technology & †Eindhoven University of Technology: 1. Retrieved July 9, 2014*, Eindhoven, 2014.

[4] L. H. Newman, “GitHub Survived the Biggest DDoS Attack Ever Recorded,” Wired, 2018. Available: https://www.wired.com/story/github-ddos-memcached/

[5] “What is a DDoS Attack?”, CloudFlare. Accessed: May 4, 2020. [Online]. Available: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

[6] “DDoS Attacks,” Imperva. Accessed: May 4, 2020. [Online]. Available: https://www.imperva.com/learn/application-security/ddos-attacks/

[7] “What is IP Spoofing?”, CloudFlare. Accessed: May 4, 2020. [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/ip-spoofing/

[8] M. L. Sang, S. K. Dong, J. H. Lee, and J. S. Park, “Detection of DDoS attacks using optimized traffic matrix,” *Computers & Mathematics with Applications, vol. 63, no. 2*, pp. 501–510, 2012.

[9] T. Thapngam, S. Yu, W. Zhou, and S. K. Makki, “Distributed denial of service (DDoS) detection by traffic pattern analysis,” *Peer-to-Peer Networking and Applications*, vol. 7, no. 4, pp. 346–358, 2014.

[10] P. Xiao, W. Qu, H. Qi, and Z. Li, “Detecting DDoS attacks against data center with correlation analysis,” *Computer Communications*, vol. 67, no. C, pp. 66–74, 2015.

[11] B. Chaitanya, N. Medhi, “FlowTrApp: An SDN based architecture for DDoS attack detection and mitigation in data centers,” *3rd IEEE International Conference on Signal Processing and Integrated Networks* (SPIN), pg. 519-524, 2016.

[12] G. Anteneh, M. Garuba, J. Li, C. Liu, “Analysis of DDoS Attacks and an Introduction of A Hybrid Statistical Model to Detect DDoS Attacks on Cloud Computing Environment,” *12th*

IEEE International Conference on Information Technology-New Generation (ITNG), 2015, pg. 212-217.

[13] B. Habib, F. Khurshid, A. H. Dar and Z. Shah, "DDoS Mitigation in Eucalyptus Cloud Platform Using Snort and Packet Filtering — IP-Tables," *2019 4th International Conference on Information Systems and Computer Networks* (ISCON), Mathura, India, 2019, pp. 546-550.

[14] J. Jiao et al., "Detecting TCP-Based DDoS Attacks in Baidu Cloud Computing Data Centers," *2017 IEEE 36th Symposium on Reliable Distributed Systems* (SRDS), Hong Kong, 2017, pp. 256-258, doi: 10.1109/SRDS.2017.37.

[15] R. Patil, H. Dudeja, S. Gawade and C. Modi, "Protocol Specific Multi-Threaded Network Intrusion Detection System (PM-NIDS) for DoS/DDoS Attack Detection in Cloud," *2018 9th International Conference on Computing, Communication and Networking Technologies* (ICCCNT), Bangalore, 2018, pp. 1-7.