JURTEKSI (Jurnal Teknologi dan Sistem Informasi) ISSN 2407-1811 (Print)
Vol. VII No. 2, April 2021, hlm. 159 - 164 ISSN 2550-0201 (Online)
DOI: https://doi.org/10.33330/jurteksi.v7i2.933
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

IMPLEMENTASI TEKNOLOGI FIREWALL SEBAGAI KEAMANAN SERVER DARI SYN FLOOD ATTACK

Sahren
[1]Sistem Komputer, Sekolah Tinggi Manajemen Informatika dan Komputer Royal
email: sahren.one@gmail.com

Abstract: During the Covid-19 pandemic, the centralized academic information system is very vulnerable to various forms of attacks, such as SYN Flood, ICMP Flood, DoS, etc. This attack will cause the server to slow down, so that it takes a long time to access the application. For this reason, a method is needed so that server security is guaranteed. The method used is packet filtering firewall. The results obtained from this study can improve server security and capturing traffic that leads to the server. So this method can be used to increase security on the server.

Keywords: DoS; Firewall; ICMP Flood; Packet Filtering; SYN Flood; Server

Abstrak: Dimasa pandemi Covid-19 saat ini sistem informasi akademik secara terpusat sangat rawan terhadap berbagai bentuk serangan, seperti SYN Flood, ICMP Flood, DoS, dll. Serangan ini akan mengakibatkan server menjadi lambat, sehingga lama dalam mengakses aplikasi. Untuk itu, dibutuhkan suatu metode agar keamanan server lebih terjamin. Metode yang digunakan adalah packet filtering firewall. Hasil yang diperoleh dari penelitian ini dapat meningkatan kemanan server dan capturing traffic yang mengarah kepada server. Sehingga metode ini dapat digunakan untuk meningkatkan keamanan pada server.

Kata kunci: DoS; Firewall; ICMP Flood; Packet Filtering; SYN Flood; Server

PENDAHULUAN

Sistem informasi yang secara terpusat sangat rawan terhadap berbagai macam serangan seperti DoS *SYN Flood* dan lain sebagainya. Seorang penyerang akan menyerang sistem jaringan dengan maksud guna mengalahkan layanan keamanan pada fasilitas jaringan tersebut. Dengan mempertimbangkan fakta bahwa jaringan *public* pada awalnya dirancang untuk keterbukaan tanpa mempertimbangkan keamanan, jelas diikuti meningkatnya pula serangan *cybercriminals* dari tahun ketahuan[1].

Kelemahan dari *protocol Transmission Control Protocol (*TCP) terhadap *SYN Flood Attack* kembali ditemukan oleh Bill Checwick dan Steve Bellowin, memberikan saran untuk mencegak serangan ini. Serangan *SYN Flood* jauh lebih efektif jika dibandingkan dengan penyerangan dengan teknik serangan *Denial of Service* (DoS) lainnya [2]. Serangan ini memanfaatkan kelemahan yang memang ada pada *protocol Transmission Control Protocol/Internet* (TCP/IP) ketika dirancang sejak awal [2]. Serangan *SYN Flooding* merupakan metode DoS yang mempengaruhi *host* yang menjalankan proses *server* TCP/IP [3]. Serangan ini akan membanjiri *server* dengan *request* palsu secara bertubi-tubi, mengeksploitasi dan menghabiskan sum-

JURTEKSI (Jurnal Teknologi dan Sistem Informasi) ISSN 2407-1811 (Print)
Vol. VII No. 2, April 2021, hlm. 159 - 164 ISSN 2550-0201 (Online)
DOI: https://doi.org/10.33330/jurteksi.v7i2.933
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

berdaya jaringan. Pada dasarnya ketika sebuah komputer yang terhubung ke pada suatu server maka akan terjadi yang disebut koneksi TCP Ke *server*. Dimana *client* mengirim *SYNchronize* ke *server* dan *server* akan mengenali *acknowledge* (ACK) *request* ini dengan mengirim balik *SYN*-ACK ke *client* dan *client* mengirim ACK maka koneksi akan terbentuk [4]. Proses hubungan ini juga dikenal dengan sebutan TCP *Three Way Handshake*. Namun, pada kasus *SYN Flood* kode yang seharusnya dikirim kembali oleh *client* pada fase terakhir, tidak dikirim kembali justru komputer membuat *request* baru kesemua *port* yang ada pada *server*. Akibatnya koneksi masih terbuka dan tidak bisa ditutup oleh *server* hal ini akan terjadi secara terus-menerus akan mengakibatkan *server* menjadi sangat sibuk [5]. Dengan besarnya dampak yang akan ditimbulkan oleh serangan *SYN Flood* maka diperlukan suatu metode untuk menjamin keamanan *server*, salah satunya adalah dengan memanfaatkan metode *packet filtering firewall.*

Firewall disini suatu mekanisme keamanan mendasar yang digunakan untuk jaringan komputer. Baik yang bersifat *open source* hingga komersil [6]. *Firewall* dapat untuk memperkuat dan melindungi sumberdaya dalam jaringan dengan menolak setiap akses berbahaya dari luar. *Firewall* digolongkan menjadi dua yaitu: *packet filtering* dan aplikasi *firewall* [7]. Kinerja dari *firewall* menjadi perhatian utama dari sistem keamanan dikarenakan penurunan kinerjanya bisa menghancurkan integritas data bahkan ketersediaan layanan dapat terganggu [7]. *Packet filtering* merupakan elemen yang terdapat pada *firewall* yang akan menganalisa dan mengontrol setiap paket data yang masuk maupun keluar pada lapisan *network access*, *network* dan transfort baik lewat media fisik maupun non fisik [8]. Saat *packet filtering* terpasang setiap pengguna atau penyerang dalam jaringan tidak akan menyadarinya kecuali pada saat *firewall* menolak akses atau paket data yang dikirimnya [9].

Didalam rangkaian tugasnya akan tersusun dalam bentuk tabel filtrasi, yang merupakan konfigurasi default sistem, yang terdiri dari 3 chain yaitu: *INPUT, FORWARD, dan OUTPUT* [10]. Fungsi dari *packet filtering* dapat dianalogikan sebagai penjaga kemanan [8]. Menyediakan berbagai persyaratan keamanan seperti kualitas dan ketersediaan layanan informasi yang diperlukan [10]

Beberapa penelitian terdahulu telah banyak dilakukan berkaitan dengan metode serangan *Danial of Service* (DoS) *SYN Flood*, analisa dampak, deteksi serangan maupun tindakan pencegahan yang dapat dilakukan. [1] Melakukan penelitian tentang metode pertahanan *web server* terhadap *Distributed Slow* HTTP DoS *Attack*. [2] Melakukan penelitian sebuah pendekatan untuk mengurangi efek serangan DDoS TCP *SYN Flood*. [3] Melakukan penelitian tentang deteksi *SYN Flood* berdasarkan Bayes *Estimator* pada jaringan *MANET*. [4] Melakukan penelitian mengenai pengujian sistem keamanan jaringan melalui serangan DoS. [5] Melakukan penelitian pendeteksian serangan *SYN Flood* pada jaringan *cloud computing* menggunakan dukungan *vector machine.* [8] Melakukan sebuah penelitian untuk pengembangan model optimalisasi *firewall* menggunakan *packet filtering*. [10] Melakukan penelitian optimalisasi dan implementasi set aturan iptables di *linux.*

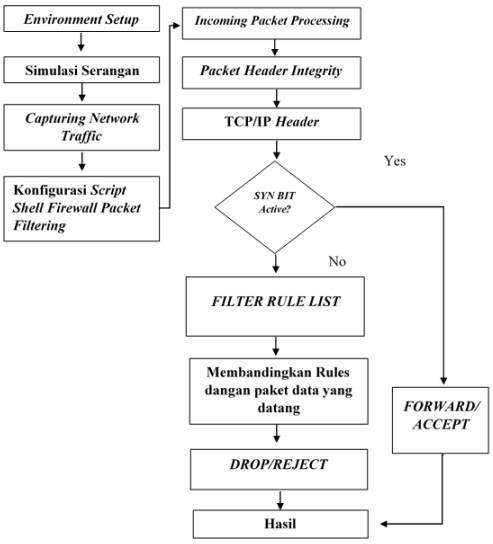
Tujuan pada penelitian ini adalah menggunakan metode *packet filtering*

JURTEKSI (Jurnal Teknologi dan Sistem Informasi) ISSN 2407-1811 (Print)
Vol. VII No. 2, April 2021, hlm. 159 - 164 ISSN 2550-0201 (Online)
DOI: https://doi.org/10.33330/jurteksi.v7i2.933
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

firewall untuk meningkatkan keamanan *server* di Sekolah Tinggi Manajemen Informatika (STMIK) Royal kisaran sehingga keamanannya lebih terjamin.

METODE

Pada bagian ini menjelaskan mengenai tahapan-tahapan sistematis yang dilakukan oleh penulis dalam melakukan penelitian untuk mengimplementasikan teknologi firewall sebagai suatu sistem keamanan guna mengatasi *SYN Flood Attack.* Adapun tahapan proses pembangunan sistem dan tahapan proses pada metode *firewall packet filtering* yang digunakan dapat di lihat pada gambar 1 berikut ini.



Gambar 1. Tahapan Penelitian

Berdasarkan tahapan penelitian yang terdapat pada gambar 1. Diuraiakan tahapan-tahapannya sebagai berikut:

Pertama Simulasi Serangan. Pada tahapan ini akan dilakukan percobaan serangan DoS *SYN Flood* pada *server* yang belum dipasang sistem kemanan *firewall.* Serangan akan dilakukan dengan mengirim sejumlah paket data ke dalam *server* secara terus menerus sehingga kinerja *server* akan terganggu di karenakan habisnya sumberdaya yang ada. Untuk melakukan simulasi serangan ini dapat di lakukan dengan menggunakan *tools Hping3*. Setelah dilakukannya serangan maka akan di analisa *traffic* yang mengarah pada *server*.

Kedua Konfigurasi *Script Shell Firewall* Tahap penulisan *script shell* ini adalah tahapan konfigurasi *firewall packet filtering* pada *server* dimana penulisan *script shell* ini dilakukan pada *tools iptables*. Konfigurasi yang di lakukan adalah untuk meningkatkan kemanan pada *server* dengan membuat *rules* atau aturan-aturan untuk memfilter setiap paket data yang masuk maupun keluar. Pada tahapan ini setiap paket akan disaring, paket yang masuk dan keluar harus sesuai dengan aturan. Tahapan pertama adalah mengekstrak IP *Header* dan melakukan pemerikasaan *protocol* TCP. Jika bit paket telah sesuai maka paket akan langsung di izinkan *(FORWARD* atau *ACCEPT).* Sedangkan paket yang tidak sesuai sistem akan membandingkan paket dengan aturan yang telah dibuat. Apabila protokol yang masuk sama akan diperiksa pada alamat tujuan, alamat sumber, port sumber dan port tujuan. Setelah itu akan diputuskan apa yang harus di lakukan dengan paket tersebut apakah di *DROP* ataupun di *REJECT*. Berikut sintaks penulisan *rule* di*firewall* dengan *iptables.*

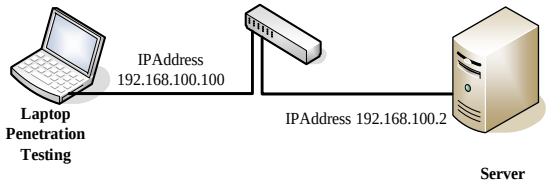
HASIL DAN PEMBAHASAN

Didalam sebuah layanan jaringan terpusat terdapat layanan-layanan penting seperti *WEB* SSH dan lain-lain, layanan tersebut memiliki jalur-jalur atau *port-port* tertentu seperti pada *WEB* dengan *port* 80 dan SSH dengan *port* 22. *Port-*

JURTEKSI (Jurnal Teknologi dan Sistem Informasi) ISSN 2407-1811 (Print)
Vol. VII No. 2, April 2021, hlm. 159 - 164 ISSN 2550-0201 (Online)
DOI: https://doi.org/10.33330/jurteksi.v7i2.933
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi

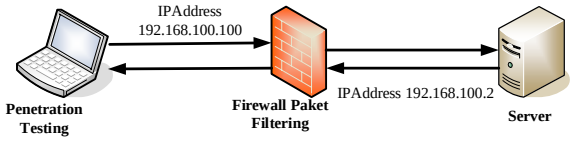
port ini lah yang menjadi jalan masuk untuk melakukan koneksi dan juga dimanfaatkan sebagai jalur serangan dengan membanjiri *port-port* tersebut dengan banyak *request*. Dalam penelitian ini serangan akan dilakukan dengan menggunakan *tools hping3*.

Percobaan menggunakan satu komputer sebagai komputer *server*, dengan ipaddress 192.168.100.2 dan satu komputer sebagai *penetration testing* dengan ipaddress 192.168.100.100. Adapun topologinya adalah sebagai berikut:



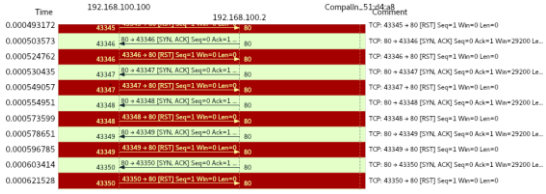
Gambar 2. Topologi Awal

Pada gambar 2 memperlihatkan bahwa koneksi antara laptop penetration testing dan komputer *server* di lakukan secara langsung tanpa ada dinding keamanan yang dilewati terlebih dahulu. Hal ini mengakibakan tidak adanya penyaringan terhadap lalu lintas data yang lewat. Sedangkan pada gambar 3 dapat kita lihat bahwa hubungan antara *penetration testing* dengan komputer *server* sudah dibatasi oleh *firewall* sehingga lalu lintas data yang masuk akan di cek atau di filter terlebih dahulu.



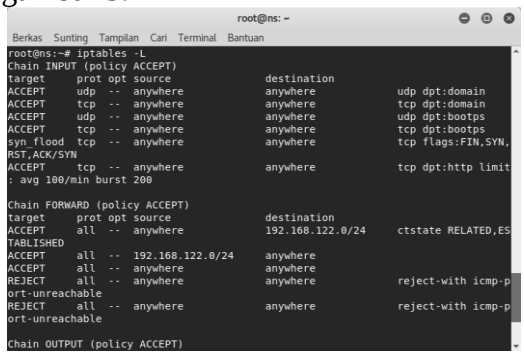
Gambar 3 Topologi Akhir

Serangan dengan mode *Flooding* data pada komputer *server* dijalankan dengan menjalankan *command* pada *tools hping3* yaitu *hping3 –flood -S -p 80 192.168.100.2* Setelah dilakukan serangan pada komputer *server* akan di lakukan analisa lalu lintas data dengan menggunakan *tools wireshark*. Dalam simulasi serangan ini komputer *server* akan merespon paket *SYN* dan mengirimkan kembali *SYN flags* ACK, akan tetapi komputer penyerang tidak merespon untuk memenuhi syarat *three way handshake* dengan mengirim ACK, tetapi mengirim *flags* RST sehingga koneksi akan setengah terbuka. Seperti yang ditampilkan pada gambar 4. Ketika koneksi seperti itu dibuat dalam beberapa detik akan mengakibatkan sumberdaya proses akan habis dalam waktu singkat.



Gambar 4. *Flowgraph* Saat Ada Serangan Sebelum Ada *Firewall*

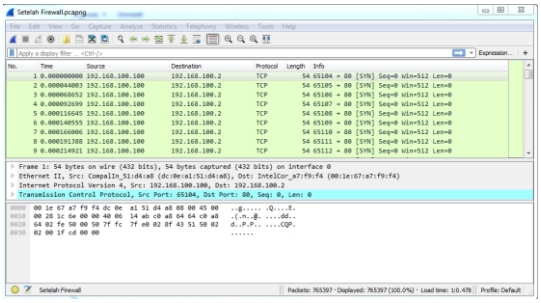
Untuk pertahanan dari serangan *SYN Flood* dengan melakukan konfigurasi *firewall* packet *filtering* pada *server* dengan menuliskan *script iptables* dan dapat dilakukan pengecekan yang akan menghasilkan *output* seperti pada gambar 5.



Gambar 5 Hasil *Firewall*

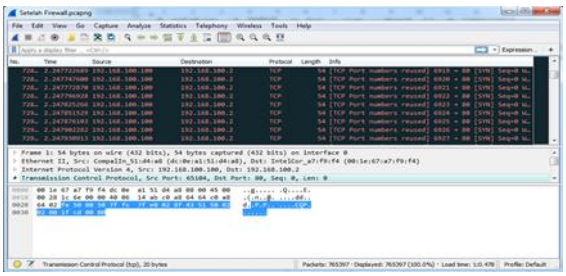
Setelah konfigurasi *firewall* dilakukan dengan menambahkan rules maka di lakukann *recapture packet* menggunakan *wireshak* sebagai *test* bahwa *script* atau *rule* yang ditambahkan telah bekerja dan untuk memperoleh hasil dari percobaan yang telah dilakukan.

JURTEKSI (Jurnal Teknologi dan Sistem Informasi) ISSN 2407-1811 (Print)
Vol. VII No. 2, April 2021, hlm. 159 - 164 ISSN 2550-0201 (Online)
DOI: https://doi.org/10.33330/jurteksi.v7i2.933
Available online at http://jurnal.stmikroyal.ac.id/index.php/jurteksi



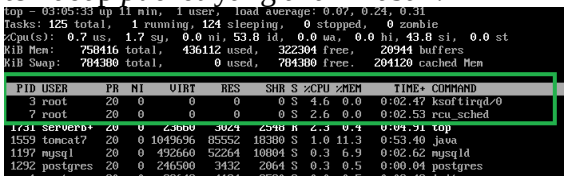
Gambar 6 *Capturing Traffic*

Gambar 6 merupakan hasil *capturing* pada *traffic* jaringan menuju *server*.



Gambar 7 *Conection Reused*

Pada gambar 7 dapat dilihat koneksi yang datang ke *server* secara terus menerus tidak dapat masuk ke *server* dikarenakan telah di lakukan filter terhadap *packet* yang akan masuk.



Gambar 8 Hasil Monitoring Proses

Pada gambar 8 merupakan tampilan proses yang berjalan dan penggunaan *resource* yang selama terjadi serangan ketika server sudah terpasang sistem keamanan dengan teknologi *firewall.*

Dalam penelitian ini, di uji kemampuan dari *firewall packet filtering* ketika terjadi eksploitasi terhadap *server* dan bertahan dari serangan yang terjadi. Untuk menentukan apakah sebuah lalu lintas jaringan sah atau tidak, *firewall* bergantung pada set aturan yang telah ditentukan oleh administrastor jaringan atau sistem untuk mengambil keputusan akhir yaitu mengizinkan atau menolah koneksi yang ada. Berdasarkan uji coba dan analisa *traffic* jaringan yang di lakukan terlihat perbedaan sebelum dan sesudah *firewall* di konfigurasi pada *server* ketika terjadi serangan sebagaimana terlihat pada hasil *flowgraph* pada gambar 4 dan gambar 7. Pada monitoring penggunaan cpu saat serangan sedang terjadi dengan memasang *firewall* mampu menurunkan usage cpu 46% hingga turun menjadi 4,6 %.

SIMPULAN

Berdasarkan hasil yang diperoleh dari penelitian ini dapat meningkatan kemanan *server* dilihat dari *capturing traffic* yang mengarah kepada *server* dan monitoring terhapat proses resource CPU saat terjadi serangan mampu menurunkan *usage* cpu 46% hingga turun menjadi 4,6 %. Dengan mampunya *firewall packet filtering* menjaga penggunaan sum-berdaya CPU pada *server* tetap rendah dan memfilter koneksi yang tidak sah. Maka dapat disimpulkan bahwa dengan metode ini dapat menjaga aspek ketersedian atau *avability*. Sehingga metode ini dapat digunakan untuk meningkatkan keamanan pada *server.*

DAFTAR PUSTAKA

[1] M. Arman, “Metode Pertahanan Web Server Terhadap Distributed Slow HTTP DoS Attack,” *JATISI (Jurnal Tek. Inform. dan Sist. Informasi)*, vol. 7, no. 1, pp. 56–

70, 2020, doi: 10.35957/jatisi.v7i1.284.

[2] M. Bogdanoski, A. Toshevski, D. Bogatinov, and M. Bogdanoski, "A novel approach for mitigating the effects of the TCP *SYN* flood DDoS attacks," *World J. Model. Simul.*, vol. 12, no. 3, pp. 217–230, 2016.

[3] K. Hussain, S. J. Hussain, N. Z. Jhanjhi, and M. Humayun, "*SYN* flood attack detection based on bayes estimator (SFADBE) for MANET," *2019 Int. Conf. Comput. Inf. Sci. ICCIS 2019*, pp. 1–4, 2019, doi: 10.1109/ICCISci.2019.8716416.

[4] A. Maraj, G. Jakupi, E. Rogova, and X. Grajqevci, "Testing of network security systems through DoS attacks," *2017 6th Mediterr. Conf. Embed. Comput. MECO 2017 - Incl. ECYPS 2017, Proc.*, no. June, pp. 11–15, 2017, doi: 10.1109/MECO.2017.7977239.

[5] Z. Mašetić, D. Kečo, N. Doğru, and K. Hajdarević, "*SYN* Flood Attack Detection in Cloud Computing Using Support Vector Machine," *TEM J.*, vol. 6, no. 4, pp. 752–759, 2017, doi: 10.18421/TEM64-15.

[6] C. Diekmann, L. Hupel, J. Michaelis, M. Haslbeck, and G. Carle, "Verified iptables Firewall Analysis and Verification," *J. Autom. Reason.*, vol. 61, no. 1–4, pp. 191–242, 2018, doi: 10.1007/s10817-017-9445-1.

[7] P. S. Kadam, "Adaptive Packet Filtering Techniques for Linux Firewall," vol. 3, pp. 171–174, 2017.

[8] M. Thant, K. M. Thu, K. Z. Ye, and S. T. T. Sin, "Development of firewall optimization model using by packet filter," *Proc. - 2016 UKSim-AMSS 18th Int. Conf. Comput. Model. Simulation, UKSim 2016*, pp. 273–278, 2016, doi: 10.1109/UKSim.2016.45.

[9] A. A. Ali, S. M. Darwish, and S. K. Guirguis, "An Approach for Improving Performance of a Packet Filtering Firewall Based on Fuzzy Petri Net," *J. Adv. Comput. Networks*, vol. 3, no. 1, pp. 67–74, 2015, doi: 10.7763/JACN.2015.V3.144.

[10] L. Xuan and P. Wu, "The Optimization and Implementation of Iptables Rules Set on Linux," *2015 2nd Int. Conf. Inf. Sci. Control Eng.*, pp. 988–991, 2015, doi: 10.1109/ICISCE.2015.223.