Analisis Perbedaan Pengaruh Penggunaan *Iptables Chains* dalam Mencegah *Denial of Service* (DoS) pada Jaringan IoT

Hanifatun Nida[1], Ronald Adrian[1*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada
hanifatun.nida@mail.ugm.ac.id
*Korespondensi: ronald.adr@ugm.ac.id;

Abstract – The growth of home IoT device is aligned with the growth of security vulnerabilities in home network. IoT has limited resources which makes them an ideal target to Denial of Service (DoS) attack, including SYN flood. This attack tends to targeting crucial resources, such as CPU. When this attack continuously happen massively, this attack can exhaust the resources and make IoT device lost their functionality. Iptables implementation on Raspberry Pi comes to rescue to reduce SYN flood effects. Iptables has various chains and tables with their own function. This research aims to analyze the difference effect of iptables chains usage against Raspberry Pi CPU usage. The lower CPU usage, the lower DoS will likely to happen. This results then will be compared to when Raspberry Pi use previous solution by other researcher. The results show no significant difference in CPU usage for both rules.
Keywords : Internet of Things (IoT), Raspberry Pi, Iptables, Denial of Service (DoS)

Intisari – Peningkatan penggunaan perangkat *Internet of Things* (IoT) rumahan dibarengi dengan berkembangnya berbagai kerentanan keamanan pada jaringan rumah. Keterbatasan sumber daya yang dimiliki perangkat IoT menjadikannya target ideal untuk diluncurkan serangan *Denial of Service* (DoS), termasuk *SYN flood*. Serangan ini cenderung menargetkan sumber daya IoT yang krusial, termasuk CPU. Apabila dilakukan secara terus menerus, serangan ini mampu menguras sumber daya IoT dan membuatnya kehilangan fungsionalitasnya. Penerapan *iptables* pada *Raspberry Pi* mampu meminimalkan dampak serangan *SYN flood*. *Iptables* memiliki berbagai *chains* dan *tables* dengan kemampuan dan fungsi yang berbeda. Penelitian ini bertujuan untuk menganalisis perbedaan pengaruh penggunaan *iptables chains* terhadap penggunaan CPU *Raspberry Pi*. Penggunaan CPU yang semakin rendah memperkecil kemungkinan terjadinya DoS. Hasil penelitian ini kemudian akan dibandingkan dengan penggunaan CPU ketika aturan *iptables* pada penelitian terdahulu diterapkan. Hasilnya, kedua aturan mengonsumsi CPU dalam persentase yang hampir sama.
Kata kunci : *Internet of Things* (IoT), *Raspberry Pi, Iptables*, *Denial of Service* (DoS)

I. PENDAHULUAN

Internet of Things (IoT) merupakan teknologi yang memungkinkan terjadinya komunikasi antara perangkat elektronik dengan sensor melalui internet. Berdasarkan data Statistik pada Oktober 2021, terdapat sekitar 11,3 miliar perangkat IoT pada Tahun 2021 dan akan bertambah menjadi 19,1 miliar pada Tahun 2025 [1]. Sementara itu, berdasarkan prediksi yang dilakukan Insider Intelligence pada Oktober 2021, sekitar 60 juta rumah tangga di Amerika Serikat akan menggunakan perangkat IoT pada Tahun 2021 [2].

Masifnya penggunaan IoT ternyata dibarengi dengan meningkatnya berbagai ancaman keamanan, termasuk *Denial of Service/Distributed Denial of Service* (DoS/DDoS). Berdasarkan analisis Tahun 2016 terkait insiden siber terdahulu, disimpulkan bahwa 96% perangkat yang terkena DDoS adalah perangkat IoT. Salah satu kasus serangan DDoS terjadi pada Oktober 2016 yang disebabkan terinfeksinya perangkat IoT oleh *malware* Mirai. Hal ini berimplikasi pada kegagalan akses ke sebagian besar *platform* dan layanan di Amerika Utara dan Eropa. Serangan DoS/DDoS sendiri cenderung menargetkan sumber daya IoT yang krusial, salah satunya CPU [3].

Banyaknya kasus serangan terhadap perangkat IoT dikarenakan perangkat IoT sendiri memiliki kemampuan yang terbatas dan datang tanpa fitur keamanan. Keterbatasan sumber daya IoT seperti rendahnya memori, komputasi, dan konsumsi baterai membuatnya rentan terhadap serangan yang bertujuan menghabiskan sumber daya itu sendiri. Hal ini menjadikannya target yang ideal untuk dilakukan penyerangan seperti DoS [4].

Berdasarkan pada keterbatasan dan kerentanan perangkat IoT tersebut, maka diperlukan suatu cara untuk memproteksi perangkat IoT rumahan dari serangan DoS seperti *SYN flood*. *Salah* satu cara yang dapat digunakan untuk menangani *SYN flood* adalah dengan menggunakan *iptables* sebagai *firewall*. Penerapan *iptables* sebagai *firewall* pernah dilakukan oleh Marek Majkowski dalam menganalisis perbandingan jumlah paket yang ditolak ketika menggunakan *chain* PREROUTING tabel RAW dan *chain* INPUT tabel FILTER dalam rentang waktu yang sama [5]. Hasilnya, *chain* PREROUTING tabel RAW lebih banyak menolak paket dibanding *chain* INPUT tabel FILTER. Sementara itu, Dmitrij Melkov berfokus pada pengaruh jumlah aturan yang diterapkan pada tiap *chain* terhadap *throughput* [6]. Hasilnya, *chain* INPUT tabel FILTER mampu menangani lebih banyak aturan dengan *throughput* lebih besar dibanding *chain* PREROUTING tabel RAW. Berbeda dengan keduanya, penelitian ini akan membandingkan pengaruh kinerja tiap *chain* dalam menangani *SYN flood* terhadap penggunaan CPU Raspberry Pi. Penggunaan CPU dijadikan tolok ukur keefektifan suatu *iptables chains* dalam menangani *SYN flood.* Aturan yang pernah digunakan AL-Musawi [7] dalam memitigasi DoS/DDoS pun dianalisis.

II. DASAR TEORI

A. Raspberry Pi

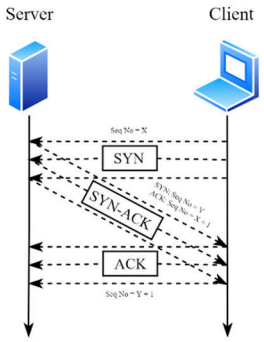
Raspberry Pi merupakan sebuah *single-board* komputer yang dapat digunakan bersama perangkat keras, seperti *mouse, keyboard,* dan monitor. Raspberry Pi memiliki CPU yang kompatibel dengan ARM, *on-chip graphics,* dan slot untuk memasukkan *SD card. SD card* ini digunakan sebagai penyimpanan memori bagi sistem operasi dan aplikasi secara keseluruhan [8].

Ukuran Raspberry Pi yang sekecil kartu kredit dilengkapi dengan *wireless chip* yang mampu menjadikannya *Wireless Access Point* (WAP) bagi perangkat-perangkat IoT, empat *USB port,* sebuah *ethernet port,* sebuah *HDMI port,* dan empat puluh pin *General-purpose input/output* (GPIO). Fitur-fitur resebut mampu menjadikan Raspberry Pi sebagai perangkat koneksi IoT, karena kemampuannya mengumpulkan, menyimpan, memproses, hingga mengunggah data yang dihasilkan perangkat-perangkat IoT yang terhubung padanya [9].

B. Serangan *SYN Flood*

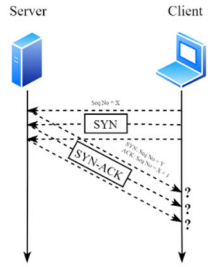
Serangan DoS maupun Serangan *SYN flood* merupakan salah satu tipe serangan *Denial of Service* (DoS) yang mampu menyibukkan seluruh *port* target [10]. Serangan ini dilakukan dengan mengirimkan paket SYN dalam jumlah besar dengan memanfaatkan kerentanan dari *TCP connection sequence,* yaitu *three-way handshake*. *Three-way handshake* adalah suatu metode yang digunakan klien dan server untuk bertukar paket SYN *(synchronization)* dan ACK *(acknowledgement)* sebelum komunikasi data antar keduanya terjadi [11].

Normalnya, untuk memulai koneksi TCP, klien akan mengirim paket SYN ke server. Server kemudian membalasnya dengan mengirim paket SYN-ACK. Paket ACK kemudian dikirimkan oleh klien ke server, yang menandakan koneksi TCP antara klien dengan server telah terbangun [12].



Gambar 1. Proses *TCP three-way handshake*

Sementara itu, serangan *SYN flood* terjadi ketika klien mengirimkan sejumlah besar paket SYN ke tiap *port* mesin target dan tidak merespons paket SYN-ACK server dengan ACK [13]. Akibatnya, server terus menunggu paket ACK yang akhirnya menyebabkan koneksi setengah terbuka *(half-open connection)* antara klien dan server. Koneksi ini akan terus terbuka dan menyebabkan server kebanjiran paket SYN hingga membuatnya kehabisan sumber daya. Habisnya sumber daya pada server mampu menyebabkan terjadinya DoS, yakni kondisi ketika server kehilangan fungsionalitasnya dalam menyediakan sumber daya jaringan bagi pengguna yang sah [14].



Gambar 2. Proses *TCP half-open connection*

C. Iptables

Iptables merupakan *default firewall* di Linux dan beroperasi di atas *Netfilter*. Dengan kata lain, *iptables* merupakan *front end* dari *Netfilter* [15]. *Netfilter* merupakan *framework* yang digunakan pada pemrosesan paket di kernel Linux [16]. *Framework Netfilter* menyediakan *hooks* penyaringan paket *(packet filtering hooks)* pada *network stack* kernel Linux. Untuk memproses penyaringan paket, iptables berinteraksi dengan *hooks* ini. *Hooks* kemudian akan mencegat paket dan meneruskannya ke aturan-aturan pemrosesan paket. Aturan-aturan ini sendiri dikelola oleh tabel Netfilter yang memiliki beberapa *chains* bawaan *(built-in chains)* [12]. Beberapa tabel pada iptables di antaranya FILTER, NAT, MANGLE, dan RAW. Nama tiap *chain* dari tabel-tabel Netfilter sendiri mencerminkan nama *Netfilter hook* yang berasosiasi dengannya.

Paket atau lalu lintas yang datang dapat memicu aktifnya *Netfilter hooks*. Aktifnya *hook* bergantung pada kondisi paket atau lalu lintas seperti dijelaskan pada Tabel 1 [17]. Setiap *Netfilter hook* mampu mengaktifkan satu *chain.*

Tabel 1. *Netfilter hooks*

Netfilter hook	**Kondisi diaktifkannya *hook***	***Chain* yang diaktifkan**
NF_IP_PRE_ROUTING	*Hook* ini akan diaktifkan segera ketika terdeteksi adanya lalu lintas yang masuk *(incoming traffic)* pada antarmuka jaringan. *Hook* ini diproses sebelum dibuatnya keputusan perutean *(routing decision)* mengenai ke mana paket ini harus dikirim.	PREROUTING
NF_IP_LOCAL_IN	*Hook* ini akan diaktifkan setelah paket yang masuk telah dirutekan dan ditujukan ke sistem lokal.	INPUT
NF_IP_FORWARD	*Hook* ini akan diaktifkan setelah paket yang masuk telah dirutekan dan akan diteruskan ke *host* lain.	FORWARD
NF_IP_LOCAL_OUT	*Hook* ini akan diaktifkan ketika terdeteksi ada lalu lintas atau paket yang dihasilkan sistem lokal yang akan dikirim ke luar.	OUTPUT
NF_IP_POST_ROUTING	*Hook* ini akan diaktifkan oleh lalu lintas keluar *(outgoing traffic)* atau lalu lintas yang diteruskan *(forwarded traffic)* setelah dirutekan dan sesaat sebelum meninggalkan *host*.	POSTROUTING

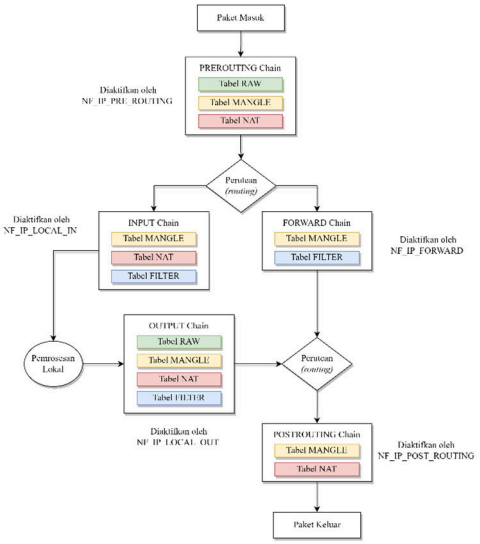
Gambar 3 menunjukkan arsitektur *Netfilter* yang merangkum pemrosesan paket di *Netfilter*.

D. Ipset

Ipset digunakan untuk mengelola sekelompok atau kumpulan *(set)*. Kumpulan tersebut dapat berupa alamat IP, alamat jaringan, alamat MAC, nomor *port,* dan antarmuka jaringan. Ipset dapat digunakan bersama iptables [18].

E. Hping3

Hping3 merupakan alat jaringan yang dapat mengirim paket TCP/IP secara *custom* dan mampu menampilkan jawaban dari target. Alat ini tersedia di Kali Linux secara *pre-installed* [19]. Beberapa hal yang dapat dilakukan menggunakan *hping3* adalah menguji aturan *firewall,* melakukan *port scanning,* dan menguji performa jaringan. *Hping3* dapat juga digunakan untuk mengirim paket secepat mungkin, yakni dengan menggunakan opsi *flood* [20].



Gambar 3. Arsitektur Netfilter

F. Ksoftirqd

Ksoftirqd merupakan suatu utas kernel *(kernel thread)* yang menangani *softirqs* yang tertunda *(pending)*. *Softirqs* atau *software interrupt request* sendiri digunakan untuk menangani penjadwalan suatu proses yang terjadi pada sistem [21]. Sibuknya pemrosesan yang dilakukan oleh kernel dapat terjadi salah satunya ketika antarmuka jaringan dibanjiri oleh paket. Hal ini akan mengaktifkan *softirqs* dengan frekuensi yang sangat tinggi. Ketika terdapat proses-proses yang masih tertunda untuk ditangani, *softirqs* akan mengaktifkan *ksoftirqd* dan membuatnya mengantrikan proses-proses yang masih tertunda tersebut, serta menanganinya seefisien mungkin [22].

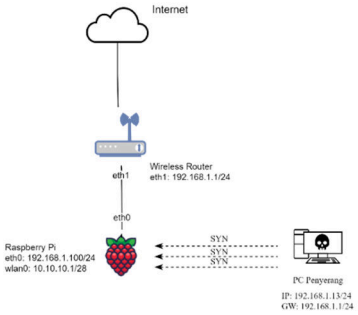
Setiap CPU dalam suatu mesin memiliki *ksoftirqd/n*-nya masing-masing, yang *n* sendiri merupakan angka logis CPU. *Ksoftirqd/0* artinya *ksoftirqd* pada CPU 0. Setiap utas *ksoftirqd/n* mendapatkan bagian untuk menangani *softirqs* yang tertunda, sehingga menyebabkan aktivitas pada penggunaan CPU, baik peningkatan maupun penurunan, tergantung pada jumlah proses yang perlu ditangani [22].

III. METODOLOGI

Penelitian ini dilakukan dengan mengidentifikasi masalah keamanan yang marak terjadi, melakukan studi literatur terhadap masalah yang ditemukan beserta solusinya, melakukan perancangan terhadap sistem yang akan dibuat sebagai solusi atas permasalahan yang ditemukan, pengaplikasian sistem, pengujian sistem, serta pencatatan dan analisis hasil. Perancangan sistem meliputi perancangan dan pengaplikasian topologi jaringan. Apabila Raspberry Pi sebagai objek yang akan dikenai serangan telah siap digunakan, *firewall* kemudian diterapkan padanya dan diuji dengan serangan *SYN flood*. Kegagalan *firewall* dapat berupa penulisan aturan yang salah maupun koneksi jaringan yang tidak stabil. Lalu pada tahap akhir, dilakukan pencatatan dan analisis terhadap hasil pengujian *firewall.*

A. Perancangan Topologi

Gambar 4 menunjukkan topologi jaringan yang digunakan dalam penelitian. Raspberry Pi terhubung ke WAP rumah menggunakan kabel *ethernet* untuk dapat terhubung ke jaringan internet. Sementara itu, mesin penyerang ditempatkan juga dalam jaringan rumah yang akan membanjiri Raspberry Pi dengan paket SYN.



Gambar 4. Topologi jaringan

B. Pembuatan Aturan Iptables

Aturan iptables yang akan dibandingkan adalah aturan iptables yang mengutilisasi ipset dan aturan iptables yang dibuat oleh AL-Musawi [7] yang mengutilisasi penggunaan *limit* dan *limit burst*. Seluruh *chains* menggunakan *default policy* ACCEPT.

1. Aturan 1: Penggunaan ipset

   Aturan ini berfokus pada penolakan seluruh paket SYN dari perangkat yang tidak didefinisikan dalam *set* WHITELIST.

```
# ipset -N WHITELIST iphash

# ipset -A WHITELIST <alamat IP>

# iptables -N SYNFLOOD

# iptables -t <filter/mangle/raw> -A <INPUT/
  PREROUTING> -m set ! --match-set WHITELIST src
  -p tcp --syn -j SYNFLOOD

# iptables <filter/mangle/raw> -A SYNFLOOD -j
  DROP
```

2. Aturan 2: Penggunaan limitasi (*limit* dan *limit burst*)

   Aturan ini berfokus pada pembatasan paket SYN yang diterima *firewall* Raspberry Pi.

Ini ditunjukkan pada *limit 1/s limit-burst 3,* yang berarti aturan tersebut hanya akan menerima maksimal tiga paket SYN dalam satu detik.

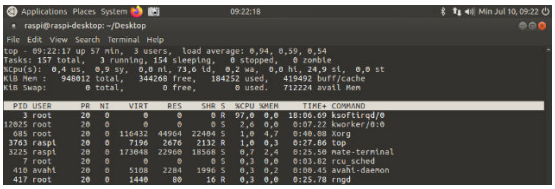
```
# iptables -N SYNFLOOD
# iptables <filter/mangle/raw> -A <INPUT/PREROUTING> -p tcp --syn -j SYNFLOOD
# iptables <filter/mangle/raw> -A SYNFLOOD -m limit --limit 1/s --limit-burst 3 -j RETURN
# iptables <filter/mangle/raw> -A SYNFLOOD -j DROP
```

C. Pengujian Aturan Iptables

Pengujian dilakukan dengan metode *black box testing*. *Black box testing* merupakan metode yang digunakan untuk menguji fungsionalitas tiap aturan yang dibuat. Tiap *chain* diuji satu per satu untuk mengetahui dampaknya terhadap penggunaan CPU Raspberry Pi ketika dikenai *SYN flood*. *SYN flood* dalam penelitian ini dilakukan dengan membanjiri Raspberry Pi dengan paket SYN dari Kali Linux dengan perintah berikut.

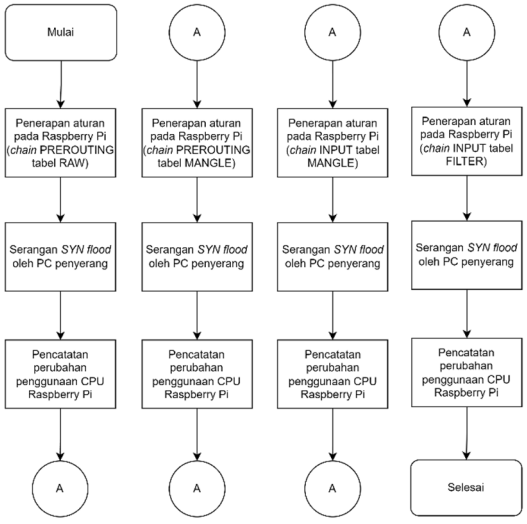
```
hping3 -S --flood -V 192.168.1.100
```

Ksoftirqd/0 dijadikan sebagai indikator kenaikan penggunaan CPU ketika terjadi *SYN flood.* Gambar 5 menunjukkan contoh penggunaan CPU Raspberry Pi mencapai 97% ketika terjadi serangan. Pengamatan terhadap CPU dilakukan dalam satu menit atau dua puluh perubahan nilai *ksoftirqd/0.*



Gambar 5. Penggunaan CPU Raspberry Pi saat *SYN flood*

Gambar 6 menunjukkan alur pengujian setiap aturan iptables yang dibuat, baik aturan 1 maupun aturan 2. Untuk setiap aturan yang diterapkan, serangan dilakukan selama satu menit. Selama satu menit pula perubahan penggunaan CPU diamati.



Gambar 6. Metode Pengujian Aturan Iptables

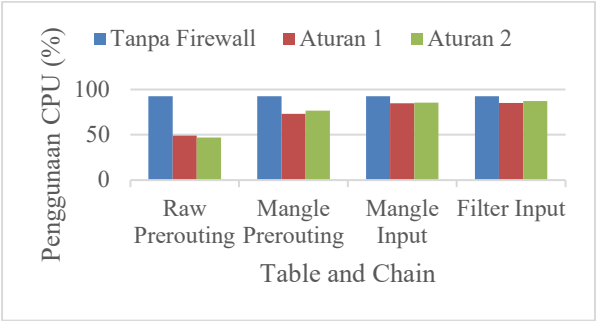
IV. HASIL DAN PEMBAHASAN

Tabel 2 menunjukkan perbandingan rata-rata penggunaan CPU Raspberry Pi ketika digunakan aturan dan *chain* dari tabel berbeda.

Tabel 2. Perbandingan penggunaan *chain* terhadap penggunaan CPU

Kondisi Raspberry Pi	Penggunaan CPU (%)
Tidak diproteksi *firewall*	92,41
Diproteksi *firewall*: Aturan 1	
Chain PREROUTING tabel RAW	49,04
Chain PREROUTING tabel MANGLE	72,82
Chain INPUT tabel MANGLE	84,62
Chain INPUT tabel FILTER	84,89
Diproteksi *firewall*: Aturan 2	
Chain PREROUTING tabel RAW	46,81
Chain PREROUTING tabel MANGLE	76,45
Chain INPUT tabel MANGLE	85,34
Chain INPUT tabel FILTER	86,83

Sementara itu, Gambar 7 menunjukkan grafik pengaruh penggunaan *chain* terhadap penggunaan CPU Raspberry Pi. Berdasarkan persentase penggunaan CPU tersebut, dapat diamati bahwa tidak terdapat perbedaan yang signifikan maupun drastis pada penggunaan CPU Raspberry Pi ketika diterapkan aturan 1 maupun aturan 2. Selain itu, terlihat juga bahwa *chain* PREROUTING tabel RAW menekan penggunaan CPU paling besar dibanding *chain* dari tabel lainnya, yakni sekitar 43-45% dari penggunaan CPU ketika Raspberry Pi tidak diproteksi *firewall.*



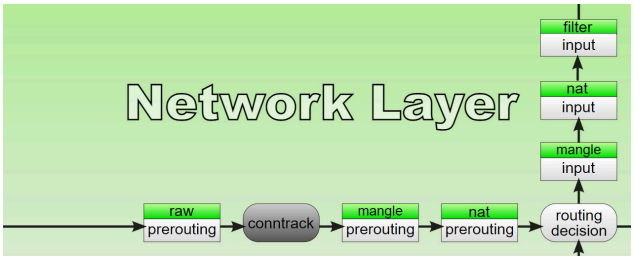
Gambar 7. Grafik pengaruh penggunaan *chain* terhadap penggunaan CPU saat *SYN flood*

Arsitektur *Netfilter* pada Gambar 3 menunjukkan bahwa aturan yang terdapat pada *chain* PREROUTING diproses sebelum dirutekan, tidak seperti *chain* INPUT. Perbedaan ini ternyata membawa dampak terhadap penggunaan CPU Raspberry Pi ketika diterapkan aturan pada salah satu dari *chain* tersebut seperti terlihat pada Tabel 2.

Selain itu, meski sama-sama menggunakan *chain* PREROUTING, penggunaan CPU ketika diterapkan aturan pada tabel RAW dan MANGLE memiliki perbedaan yang cukup besar, yakni 23-30%. Hal ini mengindikasikan adanya perbedaan pemrosesan paket antara dua tabel tersebut. Gambar 8 menunjukkan alur pemrosesan paket pada *Netfilter* lapisan jaringan.

Berdasarkan Gambar 8, terlihat bahwa *chain* PREROUTING tabel RAW diproses sebelum *connection tracking (conntrack),* sementara *chain* PREROUTING tabel MANGLE diproses tepat setelahnya.

Connection tracking sendiri digunakan untuk menyimpan informasi terkait koneksi yang masuk. Ini memungkinkan kernel untuk melacak semua koneksi atau alur logis dari koneksi jaringan, sehingga dapat ditangani bersamaan secara konsisten [23].



Gambar 8. Alur pemrosesan paket pada Netfilter di lapisan jaringan
Sumber: diambil dari [24]

Gambar 9 dan Gambar 10 membuktikan bahwa aturan pada *chain* PREROUTING tabel RAW menginspeksi paket terlebih dahulu sebelum *chain* PREROUTING tabel MANGLE.

Pada gambar 9, terlihat sekitar 1.077 ribu paket berukuran total 43 MB ditolak oleh aturan pada *chain* PREROUTING tabel MANGLE. Sejumlah paket ini tampaknya juga melintasi *chain* PREROUTING tabel RAW *(policy ACCEPT 1077k packets, 43M bytes).*

```
root@raspi-desktop:/home/raspi# iptables -t mangle -vL
Chain PREROUTING (policy ACCEPT 184 packets, 10672 bytes)
 pkts bytes target     prot opt in     out     source
1077K   43M SYNFLOOD   tcp  --  any    any     anywhere

Chain INPUT (policy ACCEPT 184 packets, 10672 bytes)
 pkts bytes target     prot opt in     out     source

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source

Chain OUTPUT (policy ACCEPT 191 packets, 13092 bytes)
 pkts bytes target     prot opt in     out     source

Chain POSTROUTING (policy ACCEPT 191 packets, 13092 bytes)
 pkts bytes target     prot opt in     out     source

Chain SYNFLOOD (1 references)
 pkts bytes target     prot opt in     out     source
   71  2840 RETURN     all  --  any    any     anywhere
    5   200 LOG        all  --  any    any     anywhere
1077K   43M DROP       all  --  any    any     anywhere
root@raspi-desktop:/home/raspi# iptables -t raw -vL
Chain PREROUTING (policy ACCEPT 1077K packets, 43M bytes)
 pkts bytes target     prot opt in     out     source
```

Gambar 9. Paket ditolak pada *chain* PREROUTING tabel MANGLE

Sementara itu pada Gambar 10, terlihat sekitar 1.615 ribu paket berukuran total 65 MB ditolak oleh aturan pada *chain* PREROUTING tabel RAW. Kemudian pada *chain* PREROUTING tabel MANGLE, jumlah paket yang melintasi *chain* tersebut berkurang drastis *(policy ACCEPT 179 packets, 10036 bytes).* Hal ini menunjukkan paket telah berhasil ditolak oleh aturan sebelumnya, yang dalam hal ini aturan pada *chain* PREROUTING tabel RAW.

```
root@raspi-desktop:/home/raspi# iptables -t raw -vL
Chain PREROUTING (policy ACCEPT 140 packets, 7727 bytes)
 pkts bytes target     prot opt in     out     source
1615K   65M SYNFLOOD   tcp  --  any    any     anywhere

Chain OUTPUT (policy ACCEPT 138 packets, 8836 bytes)
 pkts bytes target     prot opt in     out     source

Chain SYNFLOOD (1 references)
 pkts bytes target     prot opt in     out     source
   69  2760 RETURN     all  --  any    any     anywhere
    1    40 LOG        all  --  any    any     anywhere
1615K   65M DROP       all  --  any    any     anywhere
root@raspi-desktop:/home/raspi# iptables -t mangle -vL
Chain PREROUTING (policy ACCEPT 179 packets, 10036 bytes)
 pkts bytes target     prot opt in     out     source
```

Gambar 10. Paket ditolak pada *chain* PREROUTING tabel RAW

Berdasarkan alur pemrosesan paket tersebut dan hasil perbandingan penggunaan CPU antara *chain* PREROUTING tabel RAW, *chain* PREROUTING tabel MANGLE, *chain* INPUT tabel MANGLE, dan *chain* INPUT tabel FILTER, dapat dikatakan bahwa *chain* yg digunakan mempengaruhi bagaimana CPU Raspberry Pi dikonsumsi. Semakin cepat paket SYN ini ditangani, maka semakin rendah penggunaan CPU Raspberry Pi ketika terjadi *SYN flood.*

V. SIMPULAN

Berdasarkan hasil penelitian tersebut, diketahui bahwa penggunaan ipset maupun *limit* dan *limit-burst* pada iptables sama-sama efektif dalam menyaring paket *SYN* ketika terjadi *SYN flood.* Selain itu, *chain* PREROUTING tabel RAW terbukti mampu menekan penggunaan CPU Raspberry Pi paling besar, yakni sekitar 43-45% dari penggunaan CPU ketika Raspberry Pi tidak diproteksi *firewall.* Hal ini dikarenakan *chain* PREROUTING tabel RAW merupakan *chain* yang pertama kali dilintasi paket ketika paket memasuki jaringan. Berikutnya, penggunaan CPU berturut-turut meningkat ketika aturan diterapkan pada *chain* PREROUTING tabel MANGLE, *chain* INPUT tabel MANGLE, dan terakhir *chain* INPUT tabel FILTER yang hanya mengurangi penggunaan CPU sekitar 5-7% dari penggunaan CPU ketika Raspberry Pi tidak diproteksi *firewall.* Hal ini menunjukkan bahwa semakin cepat paket SYN ditangani, maka semakin rendah penggunaan CPU Raspberry Pi ketika terjadi *SYN flood.* Semakin rendah penggunaan CPU, maka semakin kecil pula kemungkinan terjadinya DoS ketika perangkat IoT terserang *SYN flood*.

REFERENSI

[1] Statista, “IoT connected devices worldwide 2019-2030,” *Statista*, 2022. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed May 22, 2023).

[2] J. Lis, “Smart Home Forecast 2021,” *Insider Intelligence*. https://www.insiderintelligence.com/content/smart-home-forecast-2021 (accessed May 22, 2023).

[3] N. N. Thilakarathne, “Security and privacy issues in IoT Environment,” *Int. J. Eng. Manag. Res.*, vol. 10, 2020.

[4] C. Wheelus and X. Zhu, “IoT Network Security: Threats, Risks, and a Data-Driven Defense Framework,” *IoT*, vol. 1, no. 2, pp. 259–285, 2020.

[5] M. Majkowski, “How to drop 10 million packets per second,” *Cloudflare*, 2018.

[6] D. Melkov, A. Šaltis, and Š. Paulikas, “Performance Testing of Linux Firewalls,” in *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, IEEE, 2020, pp. 1–4.

[7] B. Q. M. AL-Musawi, “Mitigating DoS/DDoS attacks using iptables,” *Int. J. Eng. Technol.*, vol. 12, no. 3, pp. 101–111, 2012.

[8] R. Karunamoorthi *et al.*, “Design and Development of IoT based Home Computerization using Raspberry Pi,” *Mater. Today Proc.*, 2020.

[9] W. J. McBride and J. R. Courter, “Using Raspberry Pi Microcomputers to Remotely Monitor Birds and Collect Environmental Data,” *Ecol. Inform.*, vol. 54, p. 101016, 2019.

[10] T. A. Ahanger, A. Aldaej, M. Atiquzzaman, I. Ullah, and M. Yousufudin, “Federated Learning-Inspired Technique for Attack Classification in IoT Networks,” *Mathematics*, vol. 10, no. 12, p. 2141, 2022.

[11] D. Nashat and F. A. Hussain, “Multifractal detrended fluctuation analysis based detection for SYN flooding attack,” *Comput. Secur.*, vol. 107, p. 102315, 2021.

[12] Techopedia, "Three-Way Handshake," *Techopedia*, Nov. 10, 2020. https://www.techopedia.com/definition/10339/three-way-handshake (accessed May 22, 2023).

[13] R. Nagai, W. Kurihara, S. Higuchi, and T. Hirotsu, "Design and implementation of an openflow-based tcp syn flood mitigation," in *2018 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, IEEE, 2018, pp. 37–42.

[14] D. Kshirsagar, S. Sawant, A. Rathod, and S. Wathore, "CPU load analysis & minimization for TCP SYN flood detection," *Procedia Comput. Sci.*, vol. 85, pp. 626–633, 2016.

[15] P. Likhar and R. S. Yadav, "Impacts of Replace Venerable Iptables and Embrace Nftables in a new futuristic Linux firewall framework," in *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, 2021, pp. 1735–1742.

[16] L. Ceragioli, P. Degano, and L. Galletta, "Can my firewall system enforce this policy?," *Comput. Secur.*, vol. 117, p. 102683, 2022.

[17] J. Ellingwood, "A Deep Dive into Iptables and Netfilter Architecture | DigitalOcean," 2015. https://www.digitalocean.com/community/tutorials/a-deep-dive-into-iptables-and-netfilter-architecture (accessed May 22, 2023).

[18] "Man page of IPSET." https://ipset.netfilter.org/ipset.man.html (accessed May 22, 2023).

[19] W. B. W. Mariam and Y. Negash, "Performance evaluation of machine learning algorithms for detection of SYN flood attack," in *2021 IEEE AFRICON*, IEEE, 2021, pp. 1–6.

[20] "hping3(8) - Linux man page." https://linux.die.net/man/8/hping3 (accessed May 22, 2023).

[21] S. Van Rossem, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester, "Vnf performance modelling: From stand-alone to chained topologies," *Comput. Netw.*, vol. 181, p. 107428, 2020.

[22] D. P. Bovet and M. Cesati, *Understanding the Linux Kernel: from I/O ports to process management*. O'Reilly Media, Inc., 2005.

[23] A. Pollitt, "Linux Conntrack: Why it breaks down and avoiding the problem," *Tigera*, Apr. 26, 2019. https://www.tigera.io/blog/when-linux-conntrack-is-no-longer-your-friend/ (accessed May 22, 2023).

[24] J. Engelhardt, *Schematic for the packet flow paths through Linux networking and Xtables*. 2019. Accessed: May 22, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Netfilter-packet-flow.svg.