Pendeteksian Serangan *Denial of Service* (DoS) pada Perangkat Smartlock Berbasis Wifi Menggunakan SNORT IDS

Garrey Baldo Gunawan[1], Parman Sukarno[2], Aji Gautama Putrada[3]

[1,2,3]Prodi S1 Teknik Informatika, Fakultas Informatika, Universitas Telkom, Bandung
[1]garreybaldo@students.telkomuniversity.ac.id, [2]psukarno@telkomuniversity.ac.id,
[3]ajigp@telkomuniversity.ac.id

Abstrak

Serangan DoS menimbulkan ancaman kerugian yang serius, saat ini serangan DoS terus berkembang sehingga membuat suatu jaringan atau *resource* tidak dapat berfungsi sebagaimana mestinya. Serangan DoS pada *broker* perangkat *Smartlock* berbasis wifi ini dapat membuat perangkat tidak memberikan respon pada saat akan digunakan. *Broker* sangat berperan penting dalam berlangsungnya komunikasi antara *user* dan perangkat *Smartlock* itu sendiri. Oleh karena itu, langkah-langkah tepat harus diambil untuk memastikan keamanan dan kelancaran komunikasi perangkat. Dalam tugas akhir ini, diusulkan solusi untuk digunakan dalam menyaring serangan DoS pada perangkat *Smartlock* menggunakan SNORT IDS. Teknik yang diusulkan menggunakan SNORT dengan sejumlah *rules* yang dibuat. Penulis mengevaluasi solusi yang diusulkan dengan membandingkan tingkat presisi *rules* yang dibuat. Hasil pengujian menunjukkan bahwa *rule* terbaik didapatkan dengan nilai presisi 0.8.

Kata kunci: Smartlock, DoS, SNORT IDS

Abstract

DoS attacks pose a serious threat of harm, now DoS attacks continue to grow to make a network or resource can not function properly. DoS attacks on Wifi-based smartlock device brokers can make the device does not respond when it will be used. Brokers play an important role in the ongoing communication between the user and the Smartlock device itself. Therefore, appropriate measures must be taken to ensure the security and smoothness of device communications. In this final project, the authors propose a solution to be used in filtering DoS attacks on Smartlock devices using SNORT IDS. The proposed technique uses SNORT with a number of rules made. The author evaluates the proposed solution by comparing the level of rule precision made. Test results show that the best rule is obtained with a precision value of 0.8.

Keywords: Smartlock, DoS, SNORT IDS

1. Pendahuluan

Latar Belakang

Teknologi IoT terus berkembang seiring berjalannya waktu. Penyerang dan pengguna jahat kini mengalihkan target serangan mereka, yaitu dari server ke perangkat akhir yang dipakai oleh pengguna. Hal ini dilakukan penyerang karena ada beberapa hal yang memudahkan mereka, seperti aksesibilitas fisik pada perangkat IoT kurang terlindungi jika dibandingkan dengan server, dan memiliki akses fisik ke prangkat akhir memberi para penyerang keuntungan untuk menembus perangkat dengan tingkat kerumitan yang sedikit. Selain itu, karena perangkat IoT lebih dekat dengan pengguna, keamanan menyebabkan bocornya informasi berharga dan memiliki konsekuensi bencana[4].

Salah satu teknologi IoT yang saat ini banyak digunakan dan mudah dijumpai yaitu perangkat *Smartlock*. *Smartlock* merupakan perangkat IoT yang dapat diterapkan pada berbagai tempat, salah satunya pada pintu suatu ruangan. *Smartlock* dapat dijumpai dengan berbagai macam jenis, contohnya seperti *Smartlock* berbasis WiFi.

Serangan yang dapat terjadi pada perangkat *Smartlock* berbasis WiFi yaitu *Denial of Service*. *Denial of Service* merupakan serangan yang bertujuan untuk mempengaruhi koneksi jaringan, dan membuat jaringan tersebut tidak dapat digunakan oleh pengguna yang seharusnya. Serangan DoS dilakukan dengan membanjiri target serangan dengan lalu lintas, atau mengirim informasi untuk memicu *crash*[2]. Pada perangkat *Smartlock*, DoS dapat menyerang broker perangkat dengan berbagai cara, salah satunya dengan membanjiri perangkat dengan *request* yang akan membebankan tugas broker menjadi lebih berat.

Metode yang dapat digunakan untuk mendeteksi dan mencegah serangan DoS terjadi yaitu SNORT. SNORT merupakan sebuah metode yang bisa dibilang versi *open source* dari NIDS. SNORT dapat memeriksa analisis komunikasi dan alur data dalam sistem. Metode ini sangat pintar untuk memastikan prosedur psikoanalisis dan dapat mengidentifikasi jenis gangguan keamanan yang berbeda[5].

Topik dan Batasannya

Rumusan masalah pada tugas akhir ini yaitu, bagaimana melakukan pencegahan terhadap serangan *Denial of Service* (DoS) pada perangkat *Smartlock* berbasis Wifi dengan menggunakan *rules* SNORT IDS yang efektif.

Adapun batasan masalah pada tugas akhir ini yaitu sebagai berikut.

1. Perangkat *Smartlock* menggunakan protokol MQTT,
2. Implementasi diterapkan pada perangkat keras *Raspberry Pi 3 Model B*,
3. Serangan DoS dilakukan menggunakan tool mqtt stresser pada sistem operasi ubuntu 64 bit dengan RAM 1GB,
4. Parameter yang digunakan yaitu *False Positive* (FP), *False Negative* (FN), *Precision*, dan *Recall*.

Tujuan

Tujuan yang ingin dicapai dari penelitian tugas akhir ini yaitu sebagai berikut.

1. Mengimplementasikan SNORT rules yang diusulkan kedalam sistem *Smartlock* yang telah dibuat
2. Analisis SNORT IDS rules yang diusulkan
3. Mengetahui tingkat efektivitas dari masing-masing rules terhadap serangan DoS pada perangkat *Smartlock*

2. Studi Terkait

Pada karya tulis [3] yang berjudul *Detecting SQL Injection Attacks Using SNORT IDS*, diusulkan solusi inovatif untuk menyaring serangan injeksi SQL menggunakan IDS SNORT. Teknik pendeteksian yang diusulkan menggunakan alat SNORT dengan menambahkan sejumlah aturan SNORT tambahan. Penulis mengevaluasi solusi yang diusulkan dengan membandingkan metode penulis dengan beberapa teknik yang ada. Hasil eksperimen menunjukkan bahwa metode yang diusulkan melebihi teknik serupa lainnya menggunakan kumpulan data yang sama. Aturan SNORT yang penulis sajikan menunjukkan peningkatan signifikan dalam kinerja dalam mendeteksi serangan injeksi SQL. Dalam beberapa kasus aturan yang diajukan melakukan keberhasilan 100% dalam deteksi dengan nilai *false alarm* nol.

Karya tulis [7], penulis menyajikan prosedur untuk meningkatkan aturan Snort-IDS untuk deteksi serangan probe jaringan. Untuk menguji evaluasi kinerja, penulis menggunakan kumpulan data dari MIT-DAPRA 1999, yang mencakup lalu lintas normal dan abnormal. Pertama, hal yang dilakukan penulis yaitu menganalisis dan mengeksplorasi aturan Snort-IDS yang ada untuk memperbaiki aturan Snort-IDS yang diusulkan. Kedua, penulis menerapkan perangkat lunak WireShark untuk menganalisis paket data berupa serangan dalam kumpulan data. Setelah itu, Snort-IDS ditingkatkan, dan dapat mendeteksi serangan probe jaringan. Pada karya tulis ini, penulis telah mengklasifikasikan serangan ke beberapa kelompok berdasarkan sifat serangan probe jaringan. Selain itu, penulis juga membandingkan keampuhan serangan deteksi antara aturan Snort-IDS untuk diperbarui dengan *Detection Scoring Truth*. Sebagai hasil eksperimen, Snort-IDS yang diusulkan secara efisien mendeteksi serangan probe jaringan dibandingkan dengan *Detection Scoring Truth*. Dapat mencapai akurasi yang lebih tinggi. Namun, ada beberapa peringatan mendeteksi yang terjadi selama serangan *Detection Scoring Truth*, karena beberapa serangan terjadi dalam beberapa waktu tetapi *Detection Scoring Truth* mengidentifikasi sebagai satu waktu.

2.1 Dasar Teori

2.1.1 Serangan *Denial of Service*

Denial of Service atau bisa disebut dengan DoS merupakan serangan yang bertujuan untuk mempengaruhi koneksi jaringan, dan membuat jaringan tersebut tidak dapat digunakan oleh pengguna yang seharusnya. Serangan DoS dilakukan dengan membanjiri target serangan dengan lalu lintas, atau mengirim informasi untuk memicu *crash*. Ini adalah salah satu metode serangan *cyber* paling populer dalam keamanan jaringan[10]. Serangan Denial of Service dicirikan oleh upaya penyerang untuk mencegah pengguna layanan yang sah menggunakan sumber daya

yang diinginkan[9]. Kelompok serangan DoS memiliki dua kelas yang berbeda: serangan logika dan serangan sumber daya[11].

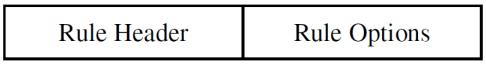
Serangan logika menyerang dengan mengeksploitasi kelemahan perangkat lunak yang ada untuk menyebabkan perangkat jauh crash atau secara substansial menurunkan kinerja. Contoh yang terkenal adalah *Ping of Death*, yang menyebabkan sistem operasi *crash* dengan mengirimkan paket ping ICMP yang lebih besar dari 65.535 byte[11].

Serangan sumber daya dilakukan dengan membanjiri sumber daya komputer korban (contoh CPU dan memori) atau sumber daya jaringan dengan mengirim aliran paket palsu yang tak henti-hentinya. Serangan jenis ini yang dapat terjadi pada sistem perangkat *Smartlock*. Serangan dilakukan dengan mengirim paket palsu dalam, sehingga broker akan bekerja lebih untuk menanggapi paket atau pesan tersebut. Hal ini yang membuat sistem (broker) mengalami *crash*, dan *user* akan sulit untuk mengakses perangkat.

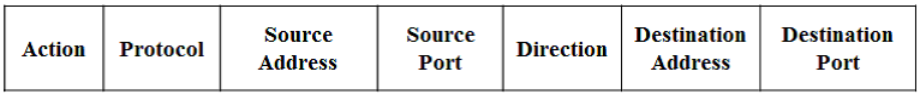
2.1.2 SNORT IDS

Snort adalah *Intrusion Detection and Protection System* (IDS / IPS) populer yang digunakan untuk melindungi risiko sistem dari penyerang. Snort adalah perangkat lunak ringan *open source*, dikembangkan oleh Martin Roesch dengan bahasa C pada tahun 1998 [1]. Snort dapat diinstal pada hampir arsitektur komputer dan platform sistem operasi. Selain itu, Snort-IDS juga menghasilkan peringatan secara *real-time* [13][6]. Snort mencari dan mencocokkan paket data lalu lintas jaringan dengan aturan untuk memeriksa lalu lintas paket data abnormal. Rule Snort-IDS dibuat dalam bentuk garis tunggal. Mudah dibaca dan dipahami, dan juga dapat dimodifikasi. Komponen dasar Snort-IDS terdiri dari: *Packet Decoder, Preprocessor, Detection Engine, Logging and Alerting System*, dan *Output Modules* [8][12].

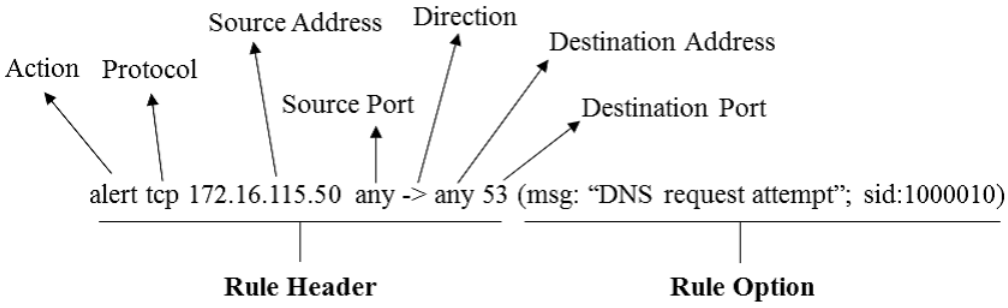
Snort-IDS menggunakan *rule* yang sesuai dengan jaringan lalu lintas paket data. Gambar 1 menunjukkan struktur dasar *rule* Snort-IDS yang dibagi menjadi dua bagian, yaitu *rule header* dan rule opsi. Gambar 2 menunjukkan setiap bidang dalam *header* Snort-IDS *rules*. Ini berisi definisi kriteria untuk pencocokan antara *rule* dan jaringan lalu lintas paket data. Selain itu, bidang tindakan dari rule header juga dapat menentukan jenis tindakan seperti pass, log alert dll. Opsi rule mengikuti rule header dan mereka berada dalam pasangan tanda kurung. Secara umum, setiap opsi-opsi rule terdiri dari dua bagian: kata kunci dan argumen. Opsi kata kunci diambil dari argumen dengan tanda titik dua. Argumen kata kunci adalah opsi di dalam tanda kutip ganda dan setiap *rule* dipisahkan oleh tanda titik koma.



Gambar 1. Struktur aturan Snort-IDS



Gambar 2. Struktur header aturan Snort-IDS



Gambar 3. Contoh dari aturan Snort-IDS[7]

Adapun mekanisme SNORT dalam mendeteksi sebuah serangan yaitu sebagai berikut[3].

1. Modul Packet Capture

Modul ini dibangun pada pemrograman paket populer *library* libpcap, yang menyediakan implementasi independen akses ke fasilitas pengambilan paket yang disediakan oleh sistem operasi, untuk menyediakan antarmuka tingkat tinggi untuk menangkap paket.

2. Decoder

   Modul ini membedah paket-paket yang diambil ke berbagai struktur data dan mengidentifikasi tautan yang akan diperiksa di modul berikutnya, seperti upaya koneksi yang mencurigakan untuk beberapa port TCP / UDP, atau terlalu banyak paket yang dikirim dalam waktu singkat.

3. Preprosesor

   Preprocessors SNORT terbagi dalam dua kategori. Mereka bisa digunakan untuk memeriksa paket baik untuk aktivitas yang mencurigakan atau memodifikasi paket sehingga modul berikutnya dapat menafsirkan dengan benar mereka. Para *preprocessors* lainnya bertanggung jawab untuk mengkategorikan lalu lintas sehingga modul berikutnya dapat mencocokkan *signature* secara akurat. *Preprocessors* ini mengalahkan serangan yang berusaha menghindari mesin deteksi SNORT dengan memanipulasi pola lalu lintas.

4. Mesin Deteksi

   Modul ini menggunakan *plug-in* deteksi, dan mencocokkan paket terhadap aturan yang dimuat ke dalam memori selama inisialisasi SNORT.

5. Plug-in Deteksi

   Definisi *plug-in* deteksi ada dalam file aturan. Mereka digunakan untuk mengidentifikasi pola.

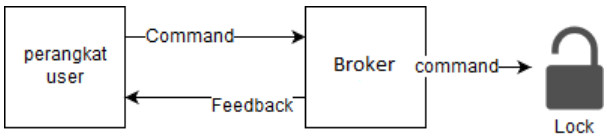
6. File Aturan

   Ini adalah file teks yang berisi daftar aturan dengan sintaksis yang diketahui. Sintaksnya mencakup protokol, alamat, dan beberapa data penting lainnya.

7. Output Plug-in

   Modul ini memformat pemberitahuan (peringatan, log) untuk pengguna untuk mengaksesnya dengan berbagai cara (basis data, konsol, dan file eksternal)

3. Sistem yang Dibangun



Gambar 4. Gambaran Umum Sistem

Sistem yang dibangun dalam tugas akhir ini merupakan desain perangkat *Smartlock* dengan menggunakan Wi-fi. Adapun serangan yang akan disimulasikan yaitu terhadap MQTT broker sebagai *control center*, dan SNORT rules akan diimplementasikan pada control center untuk mendeteksi anomali yang terjadi pada saat serangan dilakukan.

Serangan DoS dilakukan menggunakan tool MQTT Stresser pada sistem operasi ubuntu dengan menggunakan *Virtual Machine*. *Script* DoS tersebut berisikan paket data yang menyerupai paket data yang sebenarnya, tetapi dengan jumlah yang banyak dan dalam waktu yang singkat. Script yang digunakan yaitu sebagai berikut.

```
/mqtt-stresser -broker tcp://192.168.0.100:1883 -num-clients 50 -num-messages 4 -log-level 2
```

SNORT rules dibuat dalam script berdasarkan parameter-parameter yang terdapat didalam lalu lintas data pada saat user menggunakan perangkat Smartlock. SNORT rules yang diimplementasikan dan dibandingkan yaitu sebagai berikut.

1. Rule-1

   alert tcp any any ->$HOME_NET 1883 (msg: ”MQTT detect”; detection_filter:track by_dst, count 250, seconds 60; SID:0;rev:1;)

2. Rule-2

   alert tcp any any ->$HOME_NET 1883 (msg: "MQTT detect"; detection_filter:track by_dst, count 250, seconds 60; flow:to_server, established; SID:0;rev:1;)

3. Rule-3

   alert tcp any any ->$HOME_NET 1883 (msg: "MQTT detect"; threshold: type limit, track by_dst, count 250, seconds 60; flow:to_server, established; SID:0;rev:1;)

Berdasarkan struktur aturan Snort-IDS pada Gambar 2 dan 3, maka pada *rule* yang diusulkan dengan aksi *alert*, yaitu aksi yang hanya memberikan peringatan, tidak mengambil tindakan seperti *drop packet* yang terdeteksi atau menolak paket tersebut. Kemudian aksi *alert* tersebut diperuntukkan untuk pengguna yang mengakses jaringan lokal dengan port 1883 (port TCP/IP yang digunakan untuk MQTT) melalui protokol tcp dengan *source address* dari manapun, dan *source port* apapun. Pengguna yang terdeteksi berdasarkan *rule* yang diusulkan yaitu, pengguna yang mengirim dan meminta lebih dari 250 pesan dalam jangka waktu 60 detik. Angka 250 pada *rule* didapatkan karena keterbatasan perangkat yang digunakan. Perangkat yang digunakan untuk melakukan serangan DoS pada tugas akhir ini hanya dapat melakukan serangan sampai dengan 200 message, dan jika *count* yang digunakan adalah 250 dalam jangka waktu 60 detik, maka lalu lintas yang terdeteksi serangan yaitu jika dalam 1 detik ada lebih dari 4 pesan. Hal itu dikarenakan, dalam alur data yang terekam pada *tools* wireshark, satu kali penggunaan perangkat, maka akan ada 4 pesan yang masuk ke dalam sistem. Dalam kasus yang umum, MQTT pada broker *Smartlock* (Raspberry Pi 3 Model B) mengalami penurunan performa jika menerima paket mencapai jumlah 1 juta paket, dan oleh karena itu, maka rule untuk kasus umum tanpa ada keterbatasan perangkat akan mengalami perbedaan pada *count* yang dimana komponen tersebut yang akan menentukan seberapa banyak paket yang masuk dan dianggap serangan DoS.

Perbedaan diantara *rule* 1 sampai dengan *rule* 3 yaitu pada isi *rule* 1, tidak menggunakan aturan *flow:to_server, established* dan tidak menggunakan aturan *threshold*. Pada *rule* 2, ditambahkan aturan *flow:to_server, established*, dan pada rule 3 menggunakan *threshold* dengan tipe limit. Komponen-kompenen pembeda tersebut memiliki fungsi masing-masing, yang dimana *flow:to_server, established* berguna untuk memverifikasi lalu lintas yang menuju ke server pada suatu sesi, dan *threshold* berguna untuk mengurangi jumlah peringatan yang masuk. *Threshold* dapat dipakai untuk mengurangi alarm palsu secara signifikan, dan ini juga dapat digunakan untuk menulis aturan generasi yang lebih baru. Perintah Thresholding membatasi berapa kali peristiwa tertentu dicatat selama interval waktu tertentu.

3.1 Skenario Pengujian

3.1.1 Skenario 1 - Fungsionalitas Perangkat *Smartlock*

Tujuan dari skenario 1 – Fungsionalitas Perangkat *Smartlock* yaitu untuk memastikan bahwa sistem *Smartlock* berjalan dengan lancar tanpa hambatan baik dari perangkat keras maupun perangkat lunak. Sistematika pengujian pada skenario ini yaitu dilakukan simulasi penggunaan perangkat *Smartlock* seperti apa yang *end user* lakukan. Pengujian dilakukan menggunakan *smartphone* yang dihubungkan dengan jaringan yang digunakan pada perangkat *Smartlock*.

3.1.2 Skenario 2 - Serangan DoS dan Performa Broker

Tujuan dari skenario 2 - Serangan DoS dan Performa bBroker yaitu untuk memastikan apakah *script* DoS dapat menjalankan tugasnya dengan baik atau perlu dilakukan improvisasi pada isi *script*. Serangan dilakukan menggunakan sistem operasi linux Ubuntu 4 bit yang dijalankan pada *virtual machine*.

Setelah serangan dos berhasil dilakukan, kemudian dilakukan analisis dampak serangan dos terhadap performa dari broker perangkat *Smartlock*. Performa broker untuk menangani lalu lintas protokol mosquitto dilihat berdasarkan *cpu memory usage*.

3.1.3 Skenario 3 - Snort Rule pada Broker

Tujuan dari skenario 3 – Snort Rule pada Broker yaitu untuk menguji parameter-parameter yang telah ditentukan pada masing-masing dari ketiga *rule* yang dibuat. Pengujian dilakukan dengan penerapan konfigurasi snort pada broker, kemudian serangn dos dilakukan untuk mendapatkan nilai dari parameter FP, FN, TP, TN, Pesisi, dan Recall.

4. Evaluasi

4.1 Hasil Pengujian

Pengujian dilakukan dengan *request message* sebanyak 4 *message* per 1 *client*, dan jumlah *client* yaitu sebanyak 50. Tabel 1 menunjukkan hasil dari percobaan penerapan SNORT rules pada perangkat *Smartlock* terhadap serangan DoS yang dilakukan. Terdapat beberapa parameter yang menunjukkan tingkat efektivitas dari *rules* yang dibuat. Pada kasus tugas akhir ini, FP atau *False Positive* merupakan keadaan dimana kondisi normal yang terdeteksi kedalam serangan DoS, FN atau *False Negative* merupakan kondisi serangan DoS yang tidak terdeteksi, TP atau *True Positive* merupakan paket serangan yang terdeteksi, TN atau *True Negative* merupakan kondisi normal (bukan serangan) yang tidak terdeteksi sebagai serangan, *Precision* merupakan ketepatan antara informasi yang diminta dengan jawaban yang diberikan oleh sistem dengan rumus:

$$Precision = \frac{TP}{FP+TP}$$

dan *Recall* merupakan tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi dengan rumus:

$$Recall = \frac{TP}{FN+TP}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Pada pengujian serangan DoS dengan *script* yang telah disebutkan sebelumnya, *rule* 1 berhasil mendeteksi paket sebanyak 548 buah. Pada *rule* 2, paket yang terdeteksi sebanyak 265 buah, dan pada *rule* 3 sebanyak 250 buah. Berdasarkan penjelasan masing-masing parameter, maka didapatkan nilai sebagai berikut.

Tabel 1. Tabel Parameter Pengujian

Nomor Rule	Total Paket DoS	FP	FN	TP	TN	Precision	Recall	Accuracy
Rule-1	200	345	0	200	0	0.36	1	36.49%
Rule-2	200	65	0	200	283	0.75	1	88.13%
Rule-3	200	50	0	200	298	0.8	1	90.87%

Tabel 2. Tabel Pemetaan Parameter

		Nilai sebenarnya	
		True	False
Nilai prediksi	True	TP	TN
	False	FP	FN

Tabel 2 menunjukkan pemetaan dari parameter yang digunakan. Kolom True dan False (Nilai sebenarnya) merupakan nilai yang seharusnya dihasilkan atau diinformasikan oleh sistem (Snort IDS). Kemudian baris True dan False (Nilai prediksi) merupakan nilai yang dihasilkan atau diinformasikan oleh sistem (Snort IDS) pada kenyataannya.

4.2 Analisis Hasil Pengujian

Berdasarkan Tabel 2, telah ditunjukkan bahwa *rule* nomor 3 yang mendapatkan hasil lebih baik berdasarkan parameter FP dan *Precision*. Nilai parameter tersebut menunjukkan bahwa *rule* nomor 3 melakukan pendeteksian lebih optimal, dikarenakan ketepatan pendeteksian serangan DoS mendekati angka 1, dari skala 0 sampai 1. Nilai parameter FP pada rule nomor 3 pun lebih rendah, dimana hal itu menunjukkan bahwa *rule* lebih dapat membedakan anomali yang terdapat pada sistem ketika serangan DoS dilakukan.

Rule nomor 3 mengandung aturan *flow:to_server, established* dan *threshold* yang membuat paket yang terdeteksi menjadi terbatas berdasarkan interval yang ditentukan dan hanya paket yang menuju ke server yang dilakukan pengecekan, sehingga peringatan tidak akan dilakukan secara terus menerus dan akan lebih efisien. FN pada setiap *rule* memiliki nilai yang sama dikarenakan semua *rule* mendeteksi lebih dari jumlah serangan yang dilakukan, artinya pendeteksian terhadap serangan tidak ada yang terlewat.

Pada *rule* nomor 2 tidak menggunakan limit threshold, sehingga pengecekan dilakukan pada semua paket yang menuju ke server, hal itu yang menyebabkan nilai presisi dan akurasi menjadi lebih rendah. Dengan semakin

banyaknya paket yang terdeteksi dan paket tersebut bukan merupakan serangan, maka nilai parameter FP akan semakin tinggi. Tingginya nilai parameter FP berdampak pada perhitungan presisi dan akurasi berdasarkan rumus akan semakin rendah.

Pada *rule* nomor 1, ini merupakan *rule* dengan isi aturan yang masih umum, maksudnya *rule* ini melakukan pengecekan terhadap semua lalu lintas baik dari *client* menuju *server* dan sebaliknya. Hal ini tidak efisien, karena serangan hanya terjadi dari *client* ke *server*. Sama halnya dengan kasus pada *rule* nomor 2, nilai FP juga akan semakin besar dan berdampak pada perhitungan nilai presisi dan akurasi berdasarkan rumus yang telah dipaparkan sebelumnya. Pada *rule* 1 ini nilai FP akan lebih besar dari *rule* nomor 2 dan 3, dikarenakan pengecekan dilakukan terhadap lalu lintas dari dan menuju *server*.

Begitupula dengan *recall*, ketiga rule (rule 1, 2, dan 3) menghasilkan nilai yang sama dikarenakan berdasarkan rumus *recall* yang telah disebutkan sebelumnya, maka jumlah data (paket) serangan yang dipisahkan dengan benar jumlahnya sama bahkan lebih besar dibandingkan dengan jumlah data (paket) serangan yang sebenarnya, dan *recall* memiliki nilai 0-1 karena recall merupakan sebuah probabilitas.

5. Kesimpulan

Berdasarkan skenario pengujian yang telah dilakukan dapat diambil kesimpulan sebagai berikut:

1. Snort *rules* dapat diimplementasikan pada sistem *Smartlock* berbasis Wifi, dengan menempatkan konfigurasi Snort pada broker
2. Berdasarkan hasil pengujian, Snort *rule* dengan nilai presisi lebih baik yaitu *rule* nomor 3 dengan nilai 0.8
3. Tingkat efektivitas rule yang dibuat dapat dipengaruhi oleh beberapa faktor penting, yaitu penyaringan *flow* pesan atau paket yang ada pada lalu lintas sistem, dan spesifikasi perangkat yang dipakai untuk penyaringan paket yang masuk ke dalam sistem

Daftar Pustaka

[1] Snort. available at: http://www.snort.org/snort-downloads?. [accessed 27 juli 2018].

[2] S. Alanazi, J. Al-Muhtadi, A. Derhab, K. Saleem, A. N. AlRomi, H. S. Alholaibah, and J. J. Rodrigues. On resilience of wireless mesh routing protocol against dos attacks in iot-based ambient assisted living applications. In *E-health Networking, Application & Services (HealthCom), 2015 17th International Conference on*, pages 205–210. IEEE, 2015.

[3] H. Alnabulsi, M. R. Islam, and Q. Mamun. Detecting sql injection attacks using snort ids. In *Computer Science and Engineering (APWC on CSE), 2014 Asia-Pacific World Congress on*, pages 1–7. IEEE, 2014.

[4] R. Buyya and A. V. Dastjerdi. In *Internet of Things Principles and Paradigms*, page 3, 2016.

[5] A. Garg and P. Maheshwari. Performance analysis of snort-based intrusion detection system. In *Advanced Computing and Communication Systems (ICACCS), 2016 3rd International Conference on*, volume 1, pages 1–5. IEEE, 2016.

[6] C. Huang, J. Xiong, and Z. Peng. Applied research on snort intrusion detection model in the campus network. In *Robotics and Applications (ISRA), 2012 IEEE Symposium on*, pages 596–599. IEEE, 2012.

[7] N. Khamphakdee, N. Benjamas, and S. Saiyod. Improving intrusion detection system based on snort rules for network probe attack detection. In *Information and Communication Technology (ICoICT), 2014 2nd International Conference on*, pages 69–74. IEEE, 2014.

[8] V. Kumar and O. P. Sangwan. Signature based intrusion detection system using snort. *International Journal of Computer Applications & Information Technology*, 1(3):35–41, 2012.

[9] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovic. Distributed denial of service attacks. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 3, pages 2275–2280. IEEE, 2000.

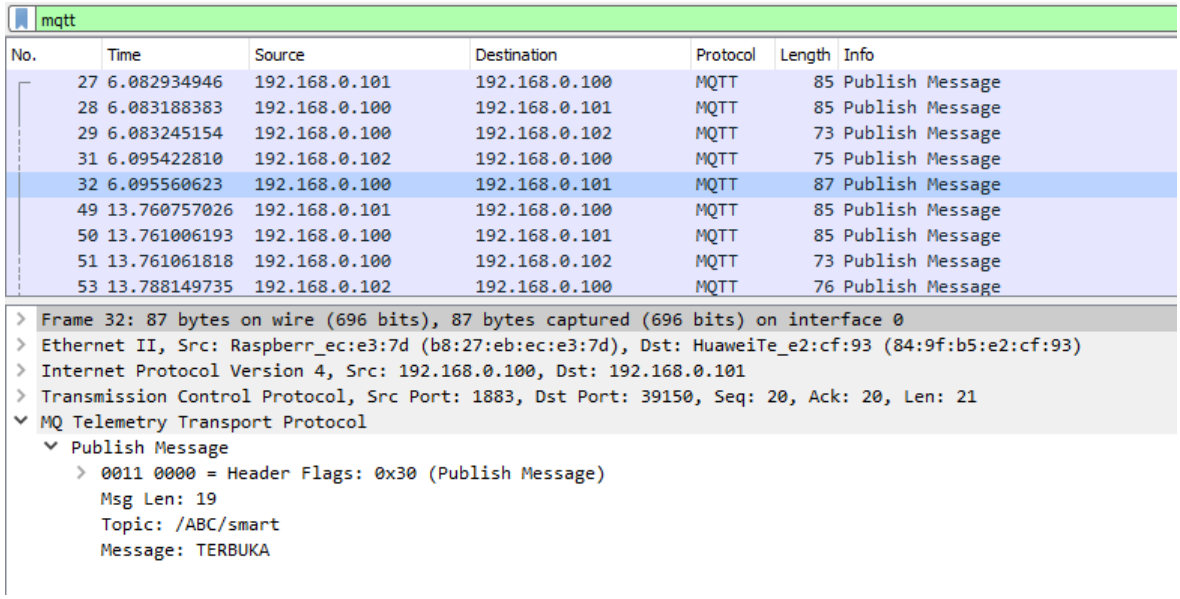
[10] L. Liang, K. Zheng, Q. Sheng, and X. Huang. A denial of service attack method for an iot system. In *Information Technology in Medicine and Education (ITME), 2016 8th International Conference on*, pages 360–364. IEEE, 2016.

[11] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage. Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139, 2006.

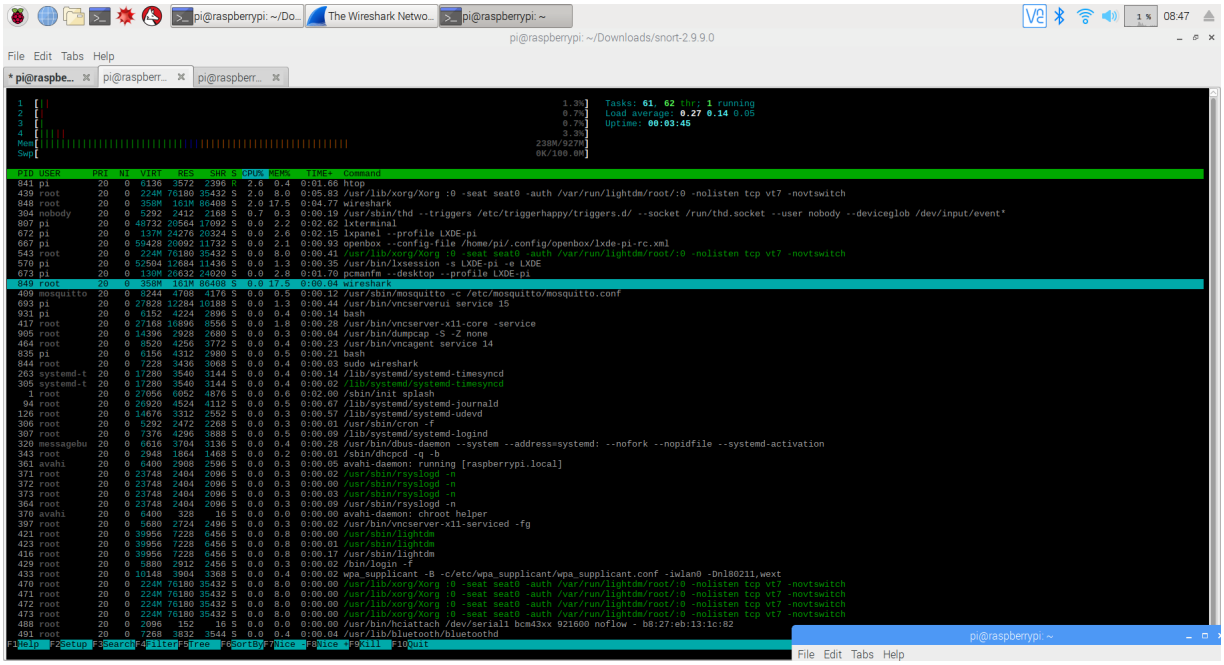
[12] S. N. Shah and M. P. Singh. Signature-based network intrusion detection system using snort and winpcap. *International Journal of Engineering Research & Technology (IJERT)*, 1(10):1–7, 2012.

[13] Z. Zhou, C. Zhongwen, Z. Tiecheng, and G. Xiaohui. The study on network intrusion detection system of snort. In *Networking and Digital Society (ICNDS), 2010 2nd International Conference on*, volume 2, pages 194–196. IEEE, 2010.

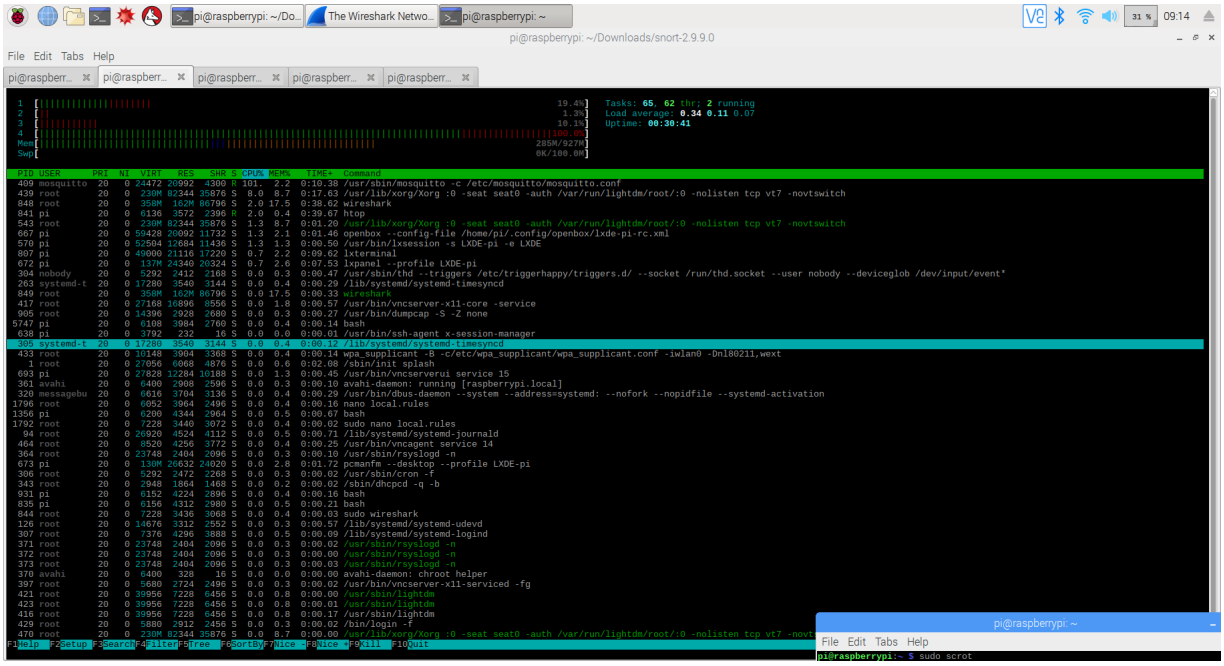
LAMPIRAN



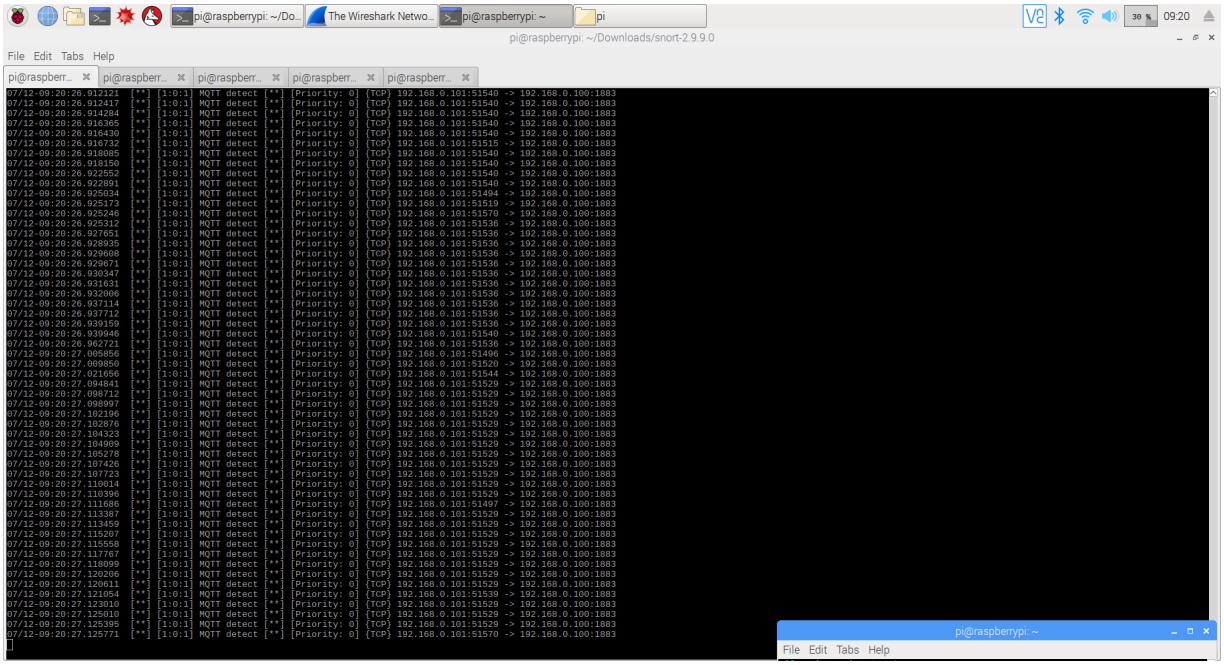
Gambar 5. Alur lalu lintas data pada sistem



Gambar 6. CPU Memory Broker Normal



Gambar 7. CPU Memory Broker saat Serangan DoS



Gambar 8. Paket data yang terdeteksi oleh SNORT