Rancangan Sistem Otomatisasi Packet Filtering berdasar Sinkronasi Data pada IP Profile Database menggunakan Python

Tri Multy Rizkilina[1], Nur Rohman Rosyid[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
trimulty98@mail.ugm.ac.id
[*]Korespondensi: nrohmanr@ugm.ac.id;

Abstract – Data traffic on a dense network is a threat to cybercrime and a high vulnerability for technology companies so that the challenges to prevent it will be more diverse. Adding a strengthening of the boundary wall or firewall and sorting data through packet filtering plus writing firewall rules to prevent malware and attacks from outside at the network device level is an alternative to protect the traffic you have. When heavy traffic makes data exchange uncontrollable, this research will create an automation design so that the sorting of incoming data packets through selection based on the specified rules runs in real-time so that the prevention of crime that enters the network is more swiftly handled. The system is running successfully by connecting the MQTT Collector as a subscriber that uses the python programming language to retrieve profiling data from the IP Profile database. The system was tested on a mikrotik router RB951Ui-2HnD which then the blocking track record will be stored in the Dynamic Firewall Data database in MongoDB. Also added a tool for controlling data in storage in the form of a program release. The results of the test show that data with an average base score above 20 is blocked and then stored in the block list in checking the collection in the database every 30 seconds. In addition, the data in the database will be checked every day within 30 days which will then be released and recorded in the release log in MongoDB.

Keywords – Packet Filltering, Firewall Rules, Python, Router Mikrotik

Intisari – Lalu lintas data pada jaringan yang padat menjadi ancaman kejahatan dunia maya serta kerentanan yang tinggi bagi perusahaan teknologi sehingga tantangan untuk mencegahnya akan semakin beragam. Menambahkan penguatan dinding batas atau firewall serta memilah data melalui packet filtering ditambah menuliskan firewall rules untuk mencegah malware dan serangan dari luar di tingkat perangkat jaringan menjadi salah satu alternatif melindungi lalu lintas yang dimiliki. Saat traffic padat membuat tidak terkontrolnya pertukaran data maka penelitian ini akan membuat suatu rancangan otomatisasi agar pemilahan packet data yang masuk melalui seleksi berdasar rules yang ditentukan berjalan real-time sehingga pencegahan kejahatan yang masuk dalam jaringan lebih sigap ditangani. Sistem berhasil berjalan dengan menghubungkan MQTT Collector selaku subscriber yang memanfaatkan bahasa pemrograman python untuk mengambil data hasil profiling dari IP Profile database. Sistem diujicobakan pada router mikrotik RB951Ui-2HnD yang kemudian rekam jejak pemblokiran akan disimpan pada database di MongoDB. Hasil dari pengujian menunjukkan data dengan average base score diatas 20 terblokir kemudian disimpan pada block list pada pengecekan collection di database setiap 30 detik. Selain itu, data pada database akan dicek setiap hari dalam kurun waktu 30 hari yang kemudian akan di release dan tercatat pada log release di MongoDB.

Kata kunci – Packet Filltering, Firewall Rules, Python, Router Mikrotik

I. PENDAHULUAN

Pesatnya perkembangan teknologi membuat lalu lintas data pada jaringan semakin padat yang menyebabkan pertukaran informasi rawan akan serangan. Ancaman kejahatan dunia maya adalah kenyataan sehari-hari bagi perusahaan dan membuat tantangan menjadi dinamis dan berkelanjutan. Celah dari kerentanan dapat dimanfaatkan oleh orang-orang yang tidak bertanggung jawab untuk melakukan tindakan ilegal [1]. Dampak buruk bila pernah atau terus terkena serangan dari pihak yang tidak bertanggungjawab tanpa adanya langkah tepat untuk mencegahnya adalah pengaksesan ilegal sehingga terjadi kebocoran data yang menyebabkan data rahasia terekspos, terjadinya kelumpuhan sistem hingga kerusakan fatal [2]. Salah satu sistem yang dapat membantu dalam pecegahan serangan jahat pada lalu lintas jaringan adalah *firewall*. *Firewall* mampu memonitor dan mengontrol semua kegiatan yang terjadi pada lalu lintas jaringan yang masuk dan keluar berdasar peraturan keamanan yang ditetapkan.

Sistem keamanan jaringan komputer dengan metode *port blocking* menggunakan *router* mikrotik mampu menghasilkan kestabilan jaringan serta meminimalkan risiko masuknya *malware* dan serangan dari luar yang dapat memicu terjadinya kelumpuhan jaringan lokal [3]. Sistem yang dibuat cenderung manual sehingga antisipasi serangan jahat tidak diperbaharui secara sigap. Penerapan sistem *packet filtering* melalui konfigurasi akan mendapatkan akses untuk mendeteksi data mencurigakan berdasar alamat IP karena masuk berkali-kali. Konfigurasi akan diterapkan pada *router* mikrotik yang terhubung pada program otomatisasi dengan bahasa *python* menggunakan *library paramiko*. Data serangan akan di proses dengan menuliskan *firewall rules* yang diatur secara dinamis sehingga data-data dari *IP Profile database* yang sesuai dengan ketentuan dapat terus diperbaharui.

Melalui penelitian ini, tujuan sistem otomatisasi yang dijalankan pada *router* mikrotik berdasar sinkronasi data dari *IP Profile database* mampu melakukan pemblokiran otomatis terhadap data *malware* tanpa harus di konfigurasi manual. Pengkonfigurasian manual akan mengabiskan waktu, tenaga dan juga kurang tepatnya data-data serangan yang tercatat. Otomatisasi *packet filltering* pada penelitian ini akan sangat mempermudah administrator jaringan karena pengambilan data dari *IP Profile database* akan dilakukan secara rel-time untuk pengecekan database sehingga pembaruan data terkontrol, kemudian riwayat pemblokiran serta pelepasan

data yang sudah baik akan disimpan rapi di dalam *log collection* pada MongoDB.

II. TINJAUAN PUSTAKA

A. Keamanan Jaringan

Keamanan jaringan yaitu bagaimana suatu jaringan mampu mengamankan jaringannya, dalam penerapan keamanan tersebut perlu dibuatkan kebijakan teknis yang digunakan untuk mengelola *user*, mencegah akses yang tidak perlu yang nantinya dapat membebani jaringan [4]. *Traffic* jaringan yang ramai akan menimbulkan bermacam celah kejahatan seperti pencurian data atau peretasan yang mampu melumpuhkan sumber daya jaringan. Langkah untuk meminimalisi terjadinya penyalahgunaan jaringan dengan meningkatkan keamanan menggunakan beragam cara seperti penguatan *firewall*, memperkuat *password* dan memasang *anti virus* ataupun cara lainnya.

Salah satu penguatan keamanan *network traffic* dalam mengatisipasi terjadinya penyalahgunaan, menggunakan *filtering rule* dengan menerapkan metode *filtering rule* yang mampu melakukan *block url* yang ada pada *protocol* HTTP maupun HTTPS dengan hasil cukup baik. Hasil analisa yang didapatkan melalui simulasi menggunakan *tool network packet anayzer wireshark* menunjukkan setiap paket yang dikirim tidak dapat dibaca (blokir) baik pada *protocol* http maupun https [5].

B. *Firewall Packet Filtering*

Firewall jaringan adalah sistem yang membatasi akses dari dalam maupun keluar jaringan. Biasanya diposisikan diantara jaringan pribadi tepercaya dan terlindungi dan jaringan publik yang tidak terpercaya. *Firewall* hanya mengizinkan lalu lintas yang disetujui masuk dan keluar sesuai dengan kebijakan atau *rules firewall* yang dibuat [6].

Firewall umumnya terdiri dari bagian *filtering* yang berfungsi untuk membatasi akses untuk mengurangi fungsi jaringan, mempersempit kanal, atau untuk memblok kelas trafik tertentu dan bagian *gateway* (*gate*). Cara untuk mengoptimalisasi suatu *firewall* dengan menentukan kebijakannya atau *rules* seperti membatasi apa dan siapa saja yang perlu dilayani, layanan-layanan apa yang dibutuhkan oleh tiap pengguna jaringan serta menerapkan konfigurasi menggunakan beberapa parameter yang tercantum dalam *header* paket data: arah (*inbound atau outbound*), *address* asal dan tujuan, *port* asal dan tujuan, serta jenis *protocol transport* yang tepat sehingga kebijakan dapat diterapkan dengan baik dan keamanan jauh lebih baik dari ancaman yang ada [7].

C. *MongoDatabase*

Basis data relasional tidak digunakan dalam manajemen data NoSQL karena kerangka kerja ini menggunakan struktur penyimpanan data non-konvensional. Biasanya, sistem non-konvensional ini tidak mendukung operasi gabungan dan cara eksekusi kuerinya secara produktif. MongoDB adalah *database* NoSQL berbasis dokumen yang dibuat oleh MongoDB Inc, yang tersedia sebagai *open source*. Sistem yang digunakan oleh MongoDB ini merupakan sistem basis data berbasis dokumen yang digambarkan oleh penyimpanan data yang sangat besar, pada saat yang sama dengan kinerja kueri yang tinggi dan lebih baik [8].

IP Profile Database memanfaatkan layanan yang disediakan MongoDB untuk menyimpan data hasil *profiling*. Data yang tersimpan berupa hasil *generate* IOC *(Indicators of Compromise*) yang didapat dari MISP yang telah diolah berdasar perhitungan tingkat kejahatan. Memanfaatkan hasil profiling pada *IP Profile Database* akan mempermudah pengolahan data pemblokiran berdasar *average base score* yang di tentukan.

D. Otomasi Jaringan

Otomatisasi jaringan adalah proses berkelanjutan untuk menghasilkan dan menerapkan perubahan konfigurasi, manajemen, dan pengoperasian ditingkat perangkat jaringan [9]. Memanfaatkan otomatisasi akan mampu mempercepat waktu operasional, menghemat biaya serta menekan *human error* yang sering terjadi. Otomatisasi jaringan yang diimplementasikan membutuhkan kombinasi *hardware* dan *software* yang diatur agar dapat menjalankan tugas secara otomatis dan berulang dalam jaringan yang ada.

Menggunakan bahasa *scripting* yang secara luas digunakan oleh *administrator* jaringan dan *administrator* sistem untuk mengotomatisasikan tugas atau pekerjaan. Bahasa pemrograman yang biasa digunakan oleh administrator untuk proses otomatisasi adalah bahasa *phyton.* Selain mudah diterapkan, bahasa *python* memiliki beragam *library* atau pustaka yang mendukung otomatisasi pada jaringan [10].

E. *Python*

Pengembang pertama bahasa pemrograman *python* adalah Guido van Rossum pada tahun 1990 di CWI, Amsterdam. Versi terakhir yang dikeluarkan CWI adalah 1.2. *Python* dapat diperoleh dan dipergunakan secara bebas karena lisensi *Python* tidak bertentangan baik menurut definisi *Open Source* maupun *General Public License* (GPL). Hingga saat ini pengembangan *Python* terus dilakukan oleh kumpulan pemrogram secara terpusat dikoordinir oleh Guido dan *Python Software Foundation*. *Python Software Foundation. Python* dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan diberbagai macam sistem operasi karena sifatnya yang *multiplatform* sehingga mencegah *Python* dimiliki oleh perusahaan komersial [11].

Python adalah bahasa pemrograman yang kuat dan mudah dipelajari. *Python* memiliki struktur data tingkat tinggi yang efisien dan pendekatan yang sederhana namun efektif untuk pemrograman berorientasi objek. Sintaks *Python* yang elegan dan pengetikan dinamis, bersama dengan sifat interpretasinya, menjadikannya bahasa yang ideal untuk pembuatan skrip dan pengembangan aplikasi yang cepat di banyak area pada sebagian besar *platform* [12].

Penelitian mengenai otomatisasi menggunakan bahasa *python* menjadi pengantar ide yang akan dikembangkan banyak pihak. Penelitian tentang metode ringkas dalam mengkonfigurasi perangkat jaringan atau disebut otomatisasi akan mampu mengurangi waktu untuk konfigurasi peralatan dan memudahkan perawatan. Memanfaatkan *library* Netmiko dan Paramiko, infrastruktur skrip yang tepat mampu mewujudkan sistem baik disertai tingkat kerentanan yang minim [13].

F. Paramiko

Paramiko adalah pustaka yang mengimplementasikan protokol SSH, biasanya digunakan untuk mengelola sistem UNIX dari jarak jauh. Paramiko membungkus protokol dan memungkinkan abstraksi tingkat tinggi dan tingkat rendah [14].

Saat paramiko telah berhasil berjalan, maka proses kerjanya SSH *client* sebagai objek akan meminta *bind* pada *local port* sehingga ssh *server* dan ssh *client* saling terkoneksi. *Client* meminta *public key* dan *host key* milik *server*. *Server* bertanggung jawab untuk membatasi pengguna dengan mengizinkan data apa saja yang boleh digunakan seperti kata sandi, dan atau jenis saluran. *Client* dan *server* menyetujui algoritma enkripsi yang akan dipakai. Setelah itu *Client* akan membentuk *session key* yang didapat sesuai persetujuan keduanya dan dienkripsikan menggunakan public *key* milik *server*. *Server* akan men-decrypt *session key* yang didapat dari *client* lalu mengenkripsi ulang dengan public *key* milik *client*, dan mengirimkannya kembali ke *client* untuk verifikasi. Langkah terakhir user akan mengautentifikasi ke *server* melalui *session key* yang tersedia untuk melakukan pengaksesan data [15].

G. MQTT

MQTT adalah *publish* atau *subscribe protocol* berbasis topik yang menggunakan string karakter untuk mendukung topik hierarki. Protokol ini memfasilitasi langganan ke berbagai topik. Protokol MQTT bersifat terbuka yang dirancang untuk perangkat terbatas yang digunakan dalam aplikasi telemetri. MQTT tidak menentukan teknik perutean atau jaringan apa pun yang berarti diasumsikan bahwa jaringan yang mendasari menyediakan layanan transportasi data *point-to-point*, *session-oriented*, *auto-segmenting* dengan *in-order delivery* (seperti TCP / IP) dan menggunakan layanan ini untuk pertukaran pesan [16].

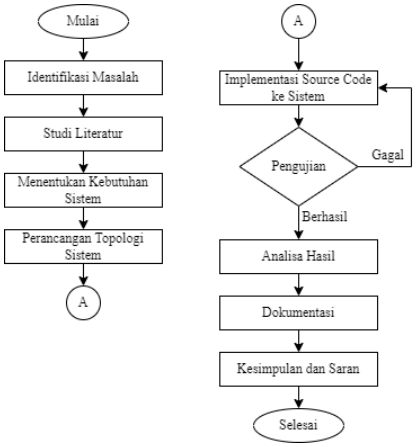
Implementasi Protokol MQTT untuk Sistem *Monitoring* Perangkat IoT. Penelitian menguji suatu sistem *monitoring* IoT memanfaatkan protokol MQTT yaitu *publish-subscribe* sangat berpengaruh dalam berjalannya proses transfer data yang dibutuhkan. Pada MQTT terdapat *library* paho yang dimanfaatkan *client* untuk men *subscribe* MQTT *broker* sebagai sumber data. Keunggulan lainnya dari MQTT salah satunya kebutuhan *resource* pada protokol MQTT lebih sedikit dibanding protokol UDP yang membuat sistem mampu berjalan pada keadaan *bandwith* yang rendah dan *latency* yang tinggi [17].

III. METODE PENELITIAN

Bab ini akan menjabarkan mengenai metode penelitian yang digunakan dalam penelitian Rancangan Sistem Otomatisasi *Packet Filtering* Berdasar Sinkronasi Data pada *IP Profile Database* Menggunakan *Python*.

A. Tahap Penelitian

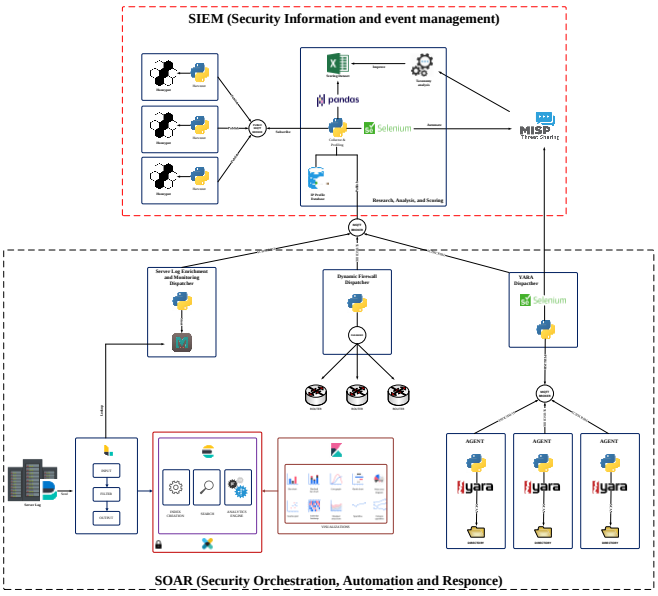
Penelitian ini melalui proses dan tahapan yang digambarkan pada Bagan Alir pada Gambar 1.



Gambar 1. Alur Penelitian

B. Arsitektur Sistem Keseluruhan

Penelitian *Threat Intelegence System* ini terdiri dari beberapa topik pembahasan yang saling terkait. Pada Gambar 2 digambarkan arsitektur topik penelitian atau *framework* keamanan informasi yang akan diimplementasikan dan diuji yang mengimplementasikan konsep SIEM (*Security Information and Event Management*) dan SOAR (*Security Orchestration, Automation and Responce*).



Gambar 2. Arsitektur Keseluruhan Sistem

C. Alat dan Bahan

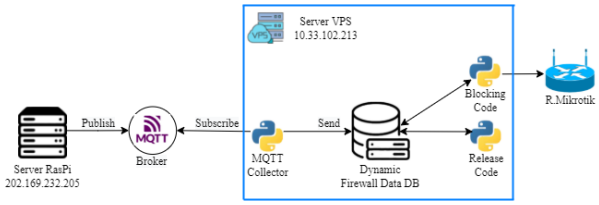
Penelitian ini menggunakan beberapa perangkat keras dan perangkat lunak sebagai penunjang proses simulasi penelitian yang dilakukan. Berikut peralatan yang digunakan

1) Perangkat Keras
- Satu (1) Komputer (Laptop)
- Satu (1) Router Mikrotik RB951Ui-2HnD

2) Perangkat Lunak
- Windows 10
- OpenVPN
- VPS UGM
- MongoDB Compass
- *Python*
- *Library* pymongo
- *Library* paramiko
- *Library* paho-mqtt *client*
- *Python* Idle

D. Arsitektur Sistem dan Fokus Penelitian

Proyek Akhir yang dikerjakan ini berfokus pada manajemen respon implementasi dari sistem SOAR untuk melakukan pengamanan gerbang antara jaringan lokal dengan jaringan internet. Sistem bernama *Dynamic Firewall* akan berjalan secara otomatis mengelola data yang akan di blokir. Arsitektur sistem pada *Dynamic Firewall* memiliki tahapan dalam berjalannya sistem yang dapat dilihat pada Gambar 3.



Gambar 3. Arsitektur Sistem Penelitian

Penjelasan pengaturan topologi dijabarkan menjadi 4 tahap.

1) *Setup* VPS dan Mikrotik RB951Ui-2HnD

Tahapan menjalankan sistem pada perangkat jaringan, diawali dengan installasi kebutuhan sistem pada VPS. *Virtual Private Server* yang digunakan menggunakan layanan VPS milik ugm sehingga jaringan yang terhubung bersama dengan router dalam jaringan ugm. Menginstall kebutuhan *library python* pada VPS seperti paramiko, *pymongo* dan paho MQTT *client*. Pada router mikrotik di atur konfigurasi IP serta memastikan router dapat berkomunikasi dengan VPS, selain itu dilakukan pengaturan penting yaitu mengatur *command firewall rules* sejak awal pengkonfigurasian. Setelah mengkonfigurasi router, tak lupa melakukan konfigurasi akses *Secure Shell* (SSH) protokol agar router bisa di *remote* atau di kontrol jarak jauh. Pengaturan SSH yang dilakukan ini agar router mikrotik dapat di otomatisasi oleh program *python* yang akan di hubungkan nanti. Jika pengaturan SSH berhasil maka router akan bisa di *remote* dan diakses melalui program dengan paramiko.

2) *Setup* MQTT *Collector*

Tahap pengambilan data selanjutnya dengan menjalankan *Collector* yang akan bertugas membangun koneksi dengan MQTT *Broker* yang terkoneksi dengan RasPi *Server* selaku *Publisher*. RasPi akan mengirimkan data berisi satu persatu file utuh berformat *json* yang telah melalui tahap *profiling*, fokus data yang akan dikumpulkan ada di parameter *average base score*. Melalui minimal nilai *average base score* yang ditentukan pada program *blocking code* akan menentukan data mana yang akan di blokir. Seluruh data yang masuk akan di simpan ke MongoDB secara terpusat.

3) *Setup Blocking Code*

Tahapan berikutnya, setelah data yang dibutuhkan dalam proses pemfilteran masuk ialah menjalankan perintah otomatisasi *packet filtering* atau *blocking code* menggunakan bahasa pemrograman *python*. Memanfaatkan *library* paramiko agar bisa melakukan kontrol jarak jauh pada *router* mikrotik yang digunakan, *library* ini masuk menggunakan protokol SSHv2 sebagai *server* maupun *client*. Selain itu, pengaturan perintah pada kode diinputkan *firewall rules* sesuai *command* yang dimiliki oleh *router* mikrotik. *Firewall Rules* tersebut akan berkorelasi dengan nilai *average base score* yang akhirnya terjadi proses pemblokiran. Saat program dijalankan perintah-perintah yang dituliskan akan otomatis tercatat didalam mongodb di *collection* blog_log_history.

4) *Setup Release Code*

Tahap keempat ini menjadi penentu kapan IP yang terblokir oleh sistem boleh dikeluarkan. Menggunakan bahasa permograman *python* yang memanfaatkan *library time* dan *datetime*, diatur agar waktu data masuk dalam *collection* blog_log_history dengan data yang akan di keluarkan berjarak sesuai dengan tenggat waktu yang ditentukan. Data yang berhasil keluar dari list pemblokiran akan di simpan ke MongoDB di *collection* log_release.

E. Penentuan Batas Nilai *Average Base Score*

Nilai *average base score* yang dimiliki oleh data *profiling* ditentukan berdasarkan pada jumlah kemunculan IoC yang sama pada Honeypot (penyerang melakukan *brute-force attack*), yang berarti bila semakin banyak IoC yang sama muncul dideteksi bahwa penyerang berusaha masuk terus menerus secara paksa. Perhitungan nilai *risk score* dilakukan oleh Arfan di penelitian "Perancangan Metode Profiling pada Honeypot Indicator of Compromise (IoC) berbasis Korelasi pada *Malware Information Sharing Platform* (MISP)" yang menghasilkan *average base score* yang akan digunakan sebagai acuan nilai batas pemblokiran berdasar Tabel 1 berikut.

Tabel 1. Risk Score Protokol Mysql

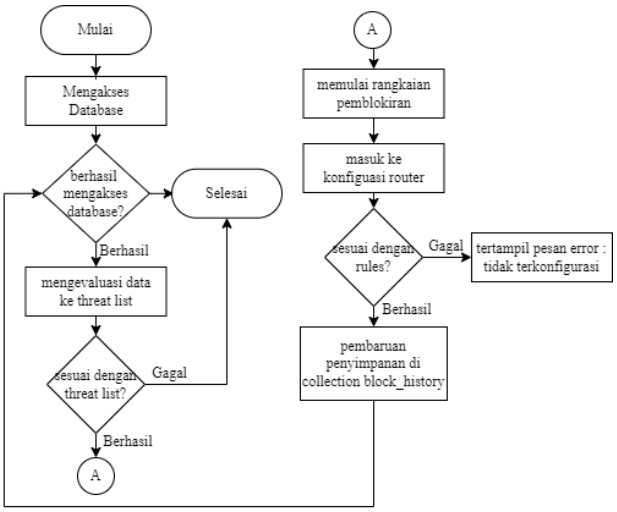
No	Protokol	Risk Score	Presentase
1	Mysql	15	25%
2	Mysql	15.45	22.06%
3	Mysql	17.25	17.65%
4	Mysql	21.3	16.18%
5	Mysql	100	19.12%

Average base score didapatkan dari hasil kali antara rerata *risk score* protocol mysql dengan banyaknya data serangan yang masuk. Data serangan masuk ke protokol *mysql* sebanyak 61.1% sehingga didapat batas nilai *average base score* sebagai acuan pemblokiran di kisaran nilai 20,6.

F. Cara Kerja Sistem

1) Skema Berjalannya Proses Pemblokiran

Diagram alir berjalannya *blocking code* dijabarkan pada Gambar 4. Penjelasan diagram alir dimulai dari pengambilan data *profiling* dari MongoDB di *collection* data_to_block, ketika data berhasil di dapat maka akan masuk ke proses seleksi kondisi, jika nilai *average base score* data sesuai ketentuan batas minimal maka akan diteruskan ke proses selanjutnya yaitu diteruskan ke *router* mikrotik melalui paramiko. Saat berhasil masuk ke *router* dan melewati proses pemblokiran maka alamat IP penyerang tidak punya hak akses di lalu lintas jaringan *router* dan masuk ke catatan pada MongoDB di *collection* block_log_history.



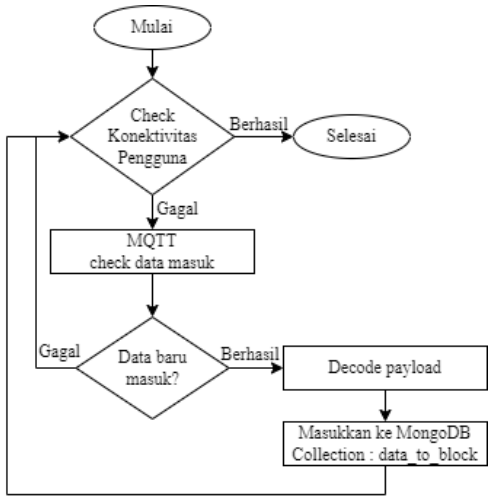
Gambar 4. Alur Sistem Pemblokiran

2) Cara Pengambilan Data

Mengimplementasikan kode yang saling terkait melalui tahapan berikut

a. Implementasi MQTT *Collector*

Program MQTT *Collector* menggunakan bahasa pemrograman *python* berfungsi untuk mengambil data yang dikirimkan oleh *server* RasPi melalui *protokol* MQTT *publisher*. Data akan disimpan pada *database* MongoDB mengunakan *library pymongo*. Program ini akan membangun koneksi untuk men-*subscribe* MQTT *Broker*. Proses berjalannya program *collector* saat menghimpun data dari *publisher* dijabarkan pada Gambar 5.



Gambar 5. Alur Proses Program MQTT-Collector

b. *Setting* Perangkat Jaringan

Memasukkan 2 *firewall rules* selaku *intial config* yang perlu disiapkan dari awal konfigurasi sebagai syarat program dapat berjalan semestinya. Berikut penjabaran *rules* yang di tuliskan

- *Rules* Satu

  Menuliskan rules ini kan membuat *address-list* yang sudah terdaftar dalam *drop-traffic* akan di *block*

  /ip firewall mangle add action=add-src-to-address-list address-list=drop_traffic chain=prerouting protocol=tcp

- *Rules* Dua

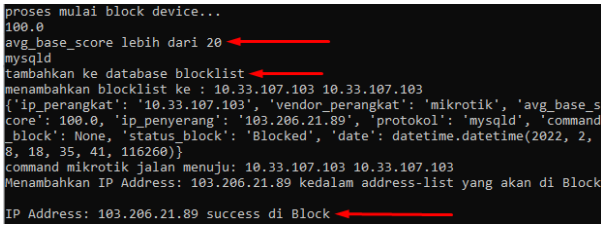
  Setelah berhasil masuk *drop-traffic* maka dipasang *firewall filter* ke *interface ether2* sehingga *traffic* yang masuk melalui *ether2* akan di *filter* sesuai daftar *address list*

  /ip firewall filter add action=drop chain=input src-address-list=drop_traffic comment=WORM in-interface=ether2

c. Implementasi *Blocking Code*

Saat *bloking code* berhasil dijalankan maka akan tertampil keterangan proses pemblokiran, letak dan keterangan alamat IP perangkat dimana data terblokir, serta data apa saja yang akan masuk pada tahap pemblokiran. Terminal tempat program berjalan pada Gambar 6 menampilkan keterangan hasil dari proses

pemblokiran. Keterangan bahwa alamat IP trekait terporses dalam *filtering* terblokir tertulis jelas.



Gambar 6. Terminal *Blocking-Code*

d. Implementasi *Release Code*

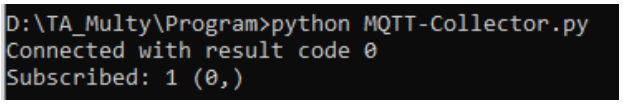
Menjalankan kode ini maka, pengecekan akan terus dilakukan pada data-data yang ada di *collection* block_log_history sehingga akan terus muncul keterangan data di jendela depan apakah data itu masih tersimpan atau bila sudah di *release* akan ditampilkan keterangan '*database* kosong'.

IV. HASIL DAN PEMBAHASAN

A. Hasil *Running* MQTT *Collector*

1) Terminal

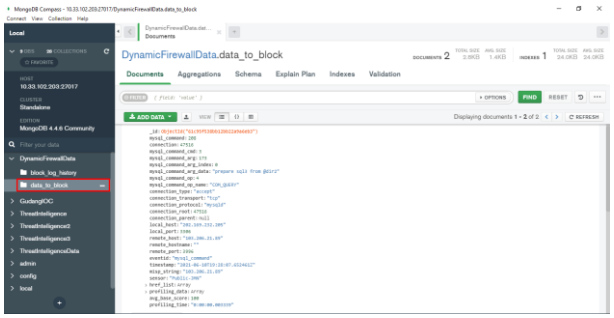
Saat MQTT *Collector* berhasil terhubung dengan *broker* akan menampilkan keterangan seperti pada Gambar 7. Saat data masuk, jendela MQTT-*Collector* akan menampilkan tiap-tiap file utuh json hasil *profiling* yang dikirm dari *IP Profile Database* secara berkala dan akan tercatat di *collection Dynamic Firewall Data database.*



Gambar 7. Hasil MQTT-*Collector* Terhubung *Broker*

2) MongoDatabase

Terbuat *collection* baru dengan nama data_to_block yang akan menyimpan masuknya data-data yang akan di proses pada *blocking code*. Data yang tersimpan disini tidak bersifat permanen untuk mencegah penumpukan data yang akan terus menerus bertambah. Tampilan data yang berhasil masuk dan tercatat pada *collection* terdapat pada Gambar 8.



Gambar 8. Tampilan Isi *Collection* data_to_block

B. Hasil *Input Firewall Rules*

Tampilan pada router ketika *firewall rules* berhasil di *input*kan seperti pada Gambar 9.

C. Hasil *Running Blocking Code*

1) MongoDatabase

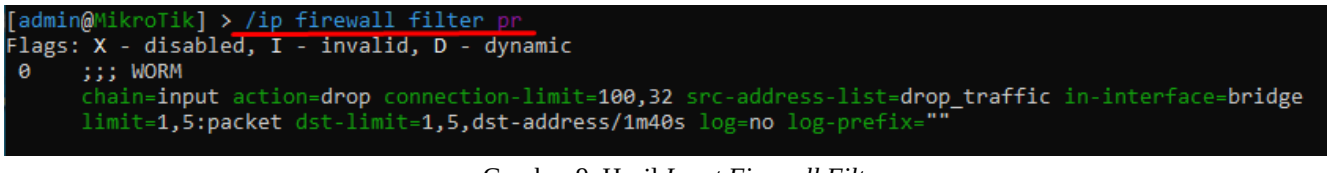
Data akan hilang dari *database* setelah berhasil masuk pada tahap pemblokiran di *router* sehingga isi dari *collection* data_to_block kosong. Data-data hasil pemblokiran tercatat dan tersimpan seperti pada tampilan Gambar 10 di block_log_history.

2) Terminal

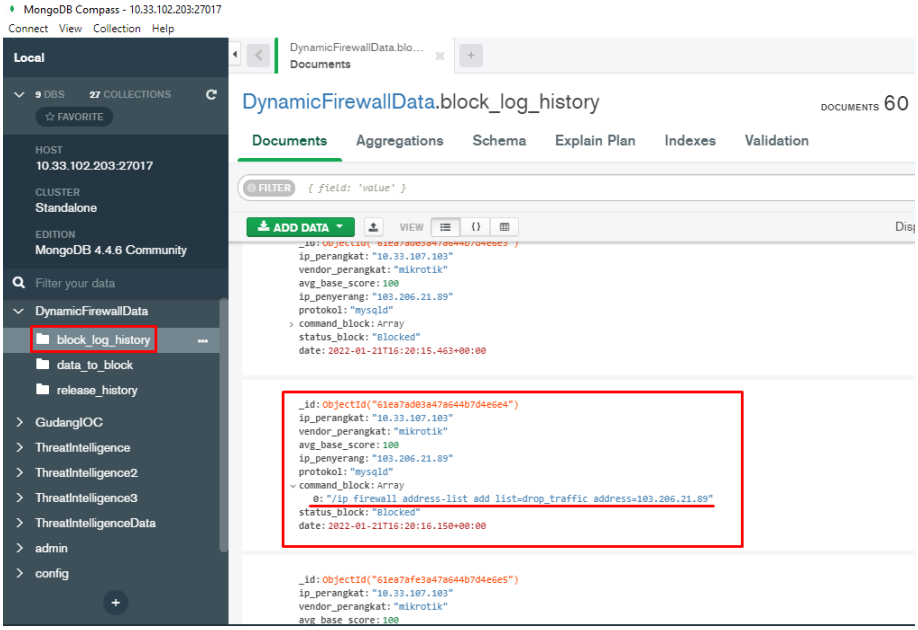
Rangkaian proses pemblokiran selesai dan berhasil, maka akan muncul keterangan pada terminal di Gambar 11 bahwa data penyerang masuk pada *log* sesuai format yang telah di atur pada *code*.

3) Router

Saat *Blocking Code* berhasil masuk dan menjalankan perintah serta *rules* diterapkan. Maka untuk pengecekan apakah alamat IP penyerang masuk dalam catatan terblokir dituliskan perintah */ip firewall address print.* Ditampilan seperti pada Gambar 12.



Gambar 9. Hasil *Input Firewall Filter*



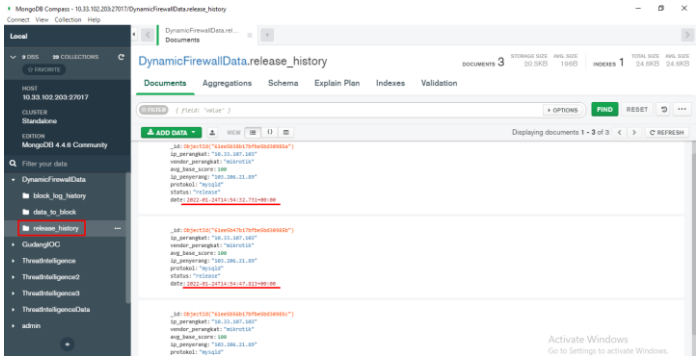
Gambar 10. Tampilan *Collection* block_log_history

```
Log tercatat di dalam database:
{'ip_perangkat': '10.33.107.103', 'vendor_perangkat': 'mikrotik', 'avg_base_score': 100, 'ip_penyerang': '103.206.21.89'
, 'protokol': 'mysqld', 'command_block': ['/ip firewall address-list add list=drop_traffic address=103.206.21.89'], 'sta
tus_block': 'Blocked', 'date': datetime.datetime(2022, 1, 20, 14, 20, 18, 17098), '_id': ObjectId('61e90d32952e72ccb4da9
72f')}
```

Gambar 11. Tampilan *Log Format*

```
[admin@MikroTik] > /ip firewall address pr
Flags: X - disabled, D - dynamic
 #    LIST                ADDRESS                 CREATION-TIME         TIMEOUT
 0    drop_traffic        103.206.21.89           jan/02/1970 03:17:40
 1 D drop_traffic        10.70.132.5             jan/06/1970 00:06:52
```

Gambar 12. Alamat IP Penyerang Tercatat



Gambar 13. Tampilan isi log_release

D. Hasil *Running Release Code*

1) MongoDatabase

Data-data hasil *blocking* di *collection* block_log_history yang siap di *release*, disororoti bagian waktu masukknya data akan menjadi parameter penting dalam proses pe-*release*-an. *Collection* log_release pada Gambar 13 akan berisi data hasil pengeluaran data dari block_log_history yang sudah tersimpan selama 30 hari atau lebih disana. tercatat tanggal keterangan data dikeluarkan

E. Hasil Pengujian

Program pemblokiran menyimpan data berdasar rerata *base score* lebih dari sama dengan 20. Pada pengamatan selama 30 menit, didapat data masuk pada *collection* data_to_block sebanyak 60 *document* yang langsung terhapus dan berpindah ke *collection* block_log_history saat *blocking code* dijalankan dengan jeda 45 detik karena memenuhi batas nilai rerata *base score*. Data yang berusia lebih dari dan sama dengan 30 hari berpindah pada *log release* saat program *release* dijalankan. Pada Tabel 2 terangkum hasil pengujian dari keempat tahap menjalankan sistem.

Tabel 2. Hasil Pengujian

No.	Proses	Pengujian	Hasil
1.	MQTT-*Collector* menghimpun data dengan men-*subscribe* topik "UGM_ProFeedAlert " kemudian menyimpannya di MongoDB.	Menjalankan program MQTT-*Collector* dan mengamati terminal program serta mongodb compass.	Berhasil
2.	Memastikan settingan ssh betul sehingga pramiko bisa mengontrol router.	Login SSH router mikrotik pada putty atau terminal.	Berhasil
3.	Data proofing yang tersimpan pada *collection* akan di seleksi average base score yang dimiliki kemudian masuk pada router untuk pemblokiran dan dicatat pada block history pada MongoDB.	Menjalankan program blocking code, mengamati terminal dan pembaruan data pada mongodb.	Berhasil
4.	Data akan dicek setiap hari selama 30 hari untuk siap di *release.*	Menjalankan program log *release* serta mengamati terminal dan pembaruan data pada *collection* log release.	Berhasil

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Rancangan otomastisasi ini berhasil mengembangkan sistem pemblokiran alamat IP terduga sumber *malware* berdasar data *profiling*. Melalui program otomasi berbahasa python, pemblokiran data berjalan secara *real-time* melalui koneksi paramiko yang mengimplementasikan protokol ssh. Program otomatis mengakses router mikrotik dan mencatat alamat IP terduga, didapat efisiensi waktu dalam melakukan pengkonfigurasian. Pada pengamatan selama 30 menit, data masuk pada *database* sebanyak 60 document yang terdistribusi langsung saat batas rerata *base score* data terpenuhi ke log yang telah disediakan. Data pada *block list* akan dikontrol selama 30 hari dan setelahnya perbolehkan keluar sembari rekam jejaknya dicatat.

B. Saran

Beberapa saran yang dapat diterapkan dalam pengembangan penelitian berikutnya :

1) Melakukan uji fungsionalitas terhadap versi *router* mikrotik lainnya

2) Melakukan uji otomatisasi *packet filltering* pada *vendor* lainnya

3) Menambahkan *interface* yang menarik untuk tampilan hasil otomatisasi di terminal

REFERENSI

[1] S. Iovan and A. Iovan, "From Cyber Threats To Cyber-Crime". *Romanian Economic Business Review*, vol. 10(2), p. 425-434, 2016.

[2] M. Rizki, "Implementasi dan Analisis Perbandingan Performa IDS *Snort* dan IDS Suricata pada Raspberry Pi". Yogyakarta: Universitas Gadjah Mada, 2019.

[3] D. Irawan, "Keamanan Jaringan Komputer dengan Metode *Blocking Port* pada Laboratorium Komputer Program Diploma-III Sistem Informasi Universitas Muhammadiyah Metro". *Mikrotik: Jurnal Manajemen Informatika*, vol 5 (2), 2015.

[4] I.G.K.O. Mardiyana, "Keamanan Jaringan dengan *Firewall Filter* Berbasis Mikrotik pada Laboratorium Komputer STIKOM Bali". *Konferensi Nasional Sistem dan Informatika 2015, Denpasar, Indonesia. STMIK STIKOM Bali*.

[5] A. Muzakir and M. Ulfa, "Analisis Kinerja *Packet Filtering* Berbasis Mikrotik *Routerboard* Pada Sistem Keamanan Jaringan", *Jurnal SIMETRIS*, vol. 10 (1), 2015.

[6] T. Katic and P. Pale., "Optimization of Firewall Rules". 2007 29th *International Conference on Information Technology Interfaces*, p. 685-690, 2007

[7] F.A. Purwaningrum, E.A. Darmadi, A. Purwanto, "Optimalisasi Jaringan Menggunakan *Firewall". Jurnal IKRA-ITH Informatika*, vol 2 (3), 2018.

[8] B. Jose and S. Abraham, "Performance analysis of NoSQL and relational databases with MongoDB and MySQL". *Materials Today: Proceedings*, 24, p. 2036–2043, 2020.

[9] M. Ulinic and S. House, *Network automation at scale*. O'Reilly Media, Inc, 2017

[10] R.A. Wiryawan, "Pengembangan Aplikasi Otomatisasi Administrasi Jaringan Berbasis *Website* Menggunakan Bahasa Pemrograman *Python". SIMETRIS*, 741- 752, 2019

[11] D. Triasanti. *Konsep Dasar Python* ,Surabaya: Sulita Jaya, 2002.

[12] G. Rossum and the Python development team. *Python Tutorial*. *Python Software Foundation*. 2020.

[13] P. Mihăilă, T. Bălan, R. Curpen and F. Sandu, F. *Network Automation and Abstraction Using Python Programming Methods.* Macro 2015, vol 2 (1), p. 95-103, 2019.

[14] M. Zadka, *DevOps in Python*: *Infrastructure as Python*. Belmont, CA, USA: Apress Media, p.111, 2019.

[15] I.D. Cahyani, I.D, "Sistem Keamanan Enkripsi *Secure Shell* (SSH) untuk Keamanan Data", *Jurnal Universitas Pandanaran*, vol 8 (16), 2010.

[16] U. Hunkeler, H. L. Truong and A. Stanford-Clark., "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks". *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, Bangalore, India, 2008. p. 791-798, 2008.

[17] Z. Abilovani, W. Yahya and F. Bakhtiar, "Implementasi Protokol MQTT untuk Sistem Monitoring Perangkat IOT". *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, vol 2 (12), p. 7521-7527, 2018.