Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202072662
Vol. 7, No. 6, Desember 2020, hlm. 1187-1196 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

SENTRALISASI PENGAWASAN INFORMASI JARINGAN MENGGUNAKAN *BLOCKCHAIN* ETHEREUM

Muhammad Fajar Sidiq[1], Akbari Indra Basuki[*2], Halim Firdaus[3], Muhammad Aldi Baihaqi[4]

[1]Fakultas Teknologi Industri dan Informatika, Institut Teknologi Telkom Purwokerto
[2]Pusat Penelitian Informatika, Lembaga Ilmu Pengetahuan Indonesia
[3,4] Fakultas Teknik Telekomunikasi dan Elektro, Institut Teknologi Telkom Purwokerto
Email: [1]fajar@ittelkom-pwt.ac.id, [2]akbari@informatika.lipi.go.id, [3]16101094@ittelkom-pwt.ac.id, [4]16101100@ittelkom-pwt.ac.id
*Penulis Korespondensi

(Naskah masuk: 04 November 2019, diterima untuk diterbitkan: 26 November 2020)

Abstrak

Pengawasan jaringan pada beberapa kantor yang berlokasi berjauhan sangat sulit dilakukan kerana keterbatasan tenaga ahli, biaya, dan teknologi pendukung. Penelitian yang sudah ada tidak dapat menyediakan sistem pengawasan jaringan yang mampu menjamin tiga aspek keamanan sekaligus, yaitu: *availability*, *integrity*, dan *confidentiality*. Teknologi *blockchain* mampu menyediakan sistem pengawasan jaringan secara terpusat dan aman dengan menjamin keamanan sistem komunikasi pelaporan dan sistem basis data pelaporan. Penelitian ini bertujuan untuk menyajikan purwarupa sistem pengawasan konfigurasi dan statistik jaringan menggunakan jejaring *blockchain* Ethereum. Metode pengawasan mengharuskan program c*ontroller* pada setiap jaringan kantor cabang untuk secara berkala menarik informasi *flow-rules* dari setiap perangkat jaringan dan melaporkan data tersebut dalam sebuah transaksi ke jejaring *blockchain*. Pada penelitian ini dianalisa dua jenis skema pengiriman transaksi: transaksi berbasis *smart contract* dan transaksi berbasis *zero-payment*. Berdasarkan hasil pengujian, transaksi berbasis *zero-payment* secara rerata hanya membutuhkan sekitar 6 % dari biaya transaksi *smart contract*. Perkiraan biaya bulanan untuk pensamplingan informasi setiap 10 menit adalah sekitar 1,19 ether per-perangkat jaringan. Meskipun demikian, metode pada penelitian ini lebih sesuai untuk diterapkan pada jejaring Ethereum jenis *Proof-of-Authority* (PoA) dibandingkan jenis *Proof-of-Work* (PoW) karena harga Ether yang mahal.

Kata kunci: *Sentralisasi, Pengawasan, Jaringan, Blockchain, Ethereum*

CENTRALIZED MONITORING OF NETWORK INFORMATION USING ETHEREUM BLOCKCHAIN

Abstract

Centralized monitoring of remote networks is hard to implement due to the high cost, lack of experts, and the missing key technologies. The existing researches are unable to provide a secure, centralized monitoring system that satisfies three security aspects, namely: availability, integrity, and confidentiality. Blockchain technology meets those three requirements by providing a reliable reporting and immutable database system. In this paper, we proposed a prototype of a centralized monitoring system that records network statistics and configurations into the blockchain ledger. The method requires the network controller to periodically fetch network information from every network device and submit it as a single blockchain transaction. We compare two kinds of transaction schema, smart-contract based and zero-payment based reporting schemes. The evaluations show that zero-payment transactions only cost 6 % of the smart-contract transactions. The estimation of the monthly cost is 1.19 ether per-device for 10-minutes data sampling. Nevertheless, the proposed method is applicable only for the Proof-of-Authority (PoA) Ethereum networks. It is not feasible for the Ethereum main network that uses Proof-of-Work (PoW) due to the high cost of Ether.

Keywords: *Centralized, Network, Monitoring, Blockchain, Ethereum*

1. PENDAHULUAN

Pada kasus perusahaan memiliki banyak kantor cabang yang tersebar di berbagai lokasi, potensi serangan terhadap sistem jaringan dapat berasal dari berbagai arah, baik dari dalam maupun dari luar. Potensi serangan dari luar dapat diminimalkan dengan menggunakan *firewall* untuk memilah paket

(Fiessler dkk, 2018) (Phatak dkk, 2018), dan *virtual private network* (VPN) sebagai sistem komunikasi antar kantor (van der Pol dkk, 2016) (Patil dkk, 2018) (Subratie dkk, 2019). Di sisi lain, potensi serangan dari dalam sangat sulit dideteksi karena celah keamanan yang beragam. Celah keamanan dapat berasal dari program penyusup seperti jenis *trojan* dan *malware* ataupun karena kesalahan konfigurasi jaringan di kantor cabang yang diekploitasi oleh penyerang.

Pada umumnya pihak penyerang lebih memilih untuk menyerang kantor cabang karena minimnya pengawasan dari kantor pusat dan terbatasnya jumlah tenaga ahli jaringan. Penggunaan protokol VPN yang dari awal bertujuan untuk menangkal serangan dari luar dapat dieksploitasi oleh penyusup untuk mengakses seluruh jaringan perusahaan.

Untuk mencegah potensi serangan dari dalam, beberapa penelitian sebelumnya lebih berfokus pada penggunaan sistem autentifikasi pengguna seperti *proxy* server dan *firewall* (Germann dkk, 2018). Sistem pengamanan berbasis *proxy server* sangat mudah diretas. Apabila penyerang dapat memperoleh informasi *username* dan *password* dari komputer pegawai, maka penyusup dapat mengakses jaringan tanpa terdeteksi.

Di sisi lain, penggunaan *firewall* dapat meningkatkan keamanan jaringan dengan membatasi masuknya perangkat baru yang tidak terdaftar. Meskipun pihak penyerang dapat memperoleh akses yang sah/*valid*, identitas *MAC adress* dari perangkat penyerang dapat dideteksi oleh *firewall*. Meskipun demikian, pihak penyerang dapat mamalsukan informasi MAC *address* (*MAC address spoofing*) untuk menyamarkan perangkatnya dan masuk ke jaringan tanpa terdeteksi. Jenis serangan ini hanya dapat terdeteksi apabila pihak pengelola jaringan melakukan pengawasan menyeluruh terhadap informasi trafik dan statistik jaringan.

Dengan demikian, metode yang paling akurat untuk menangkal berbagai jenis serangan adalah dengan melakukan pengawasan jaringan secara menyeluruh. Pengelola jaringan pusat harus dapat menganalisa setiap statistik dan konfigurasi jaringan di seluruh kantor cabang. Pada jaringan *Software Defined Networking* (SDN), tugas utama program *controller* adalah untuk memprogram dan memantau kondisi jaringan. Skema pengawasan terpusat dapat direalisasikan dengan cara memerintahkan program *controller* di setiap kantor cabang untuk mengirimkan data statistik dan konfigurasi jaringan ke pengelola pusat.

Masalah utama pada sistem pengawasan terpusat adalah tentang bagaimana cara menjamin keamanan sistem, baik dari segi komunikasi pelaporan dan sistem penyimpanan data laporan. Berangkat dari masalah tersebut, diperlukan sistem pengawasan jaringan yang terpusat yang mampu menjamin tiga aspek keamanan sebagai berikut:

1. Ketersediaan data pelaporan (*availability*)
2. Integritas data pelaporan (*integrity*)
3. Kerahasiaan data pelaporan (*confidentiality*)

Penelitian ini bertujuan untuk menyajikan purwarupa sistem pengawasan jaringan secara terpusat menggunakan jejaring *blockchain* Ethereum (Buterin, 2014) sebagai penjamin kemanan untuk sistem komunikasi pelaporan dan sistem basis data pelaporan. Teknologi *blockchain* dipilih sebagai solusi pada penilitian ini karena mampu menjawab tiga syarat kemanan yang diperlukan, yaitu: *confidentiality*, *integrity*, dan *availability.*

Pada penelitian ini, bab 2 menyajikan metode penelitian yang berisi tentang persyaratan keamanann sistem, pemilihan teknologi *blockchain* yang sesuai, dan cara pengukuran performa sistem. Selanjutnya, bab 3 membahas perancangan dan implementasi sistem. Bab 4 menampilkan perhitungan skalabilitas sistem pencatatan berbasis analisa biaya pencatatan. Bab 5 memaparkan diskusi tentang peningkatan efisiensi biaya pencatatan dan perbandingan ketersediaan data. Terakhir, bab 6 menyajikan kesimpulan.

2. METODE PENELITIAN

2.1. Persyaratan kemanan sistem

Persyaratan kemanan sistem secara umum terdiri dari tiga komponen, yaitu: Kerahasiaan (*confidentiallity*), Integritas (*integrity*), dan ketersediaan sistem (*availability*).

Informasi statistik dan konfigurasi jaringan merupakan data sensitif dan krusial bagi sebuah perusahaan. Pengiriman data ke kantor pusat melalui internet sangat rentan akan pencurian data. Sistem pelaporan informasi jaringan harus mampu menjamin *confidentiallity* dari data yang dilaporkan. Cara paling lazim untuk menjaga aspek kerahasiaan data dan mencegah adanya pencurian data atau akses tidak sah adalah dengan cara mengenkripsi enkripsi.

Pengiriman melalui internet membuka celah bagi penyerang untuk memalsukan informasi jaringan. Sistem pelaporan yang aman harus dapat memvalidasi informasi jaringan mana saja yang secara otentik dihasilkan oleh program *controller* dan mana yang tidak.

Ketersediaan sistem untuk selalu siap memberikan informasi ke pengguna merupakan aspek yang paling sulit untuk dijamin. Perusahaan besar seperti Google, Facebook, dan lainya terkadang gagal menjaga ketersediaan server akibat serangan *Distributed Denial-of-Services* (DDoS). Sistem penyimpanan informasi jaringan harus tahan terhadap serangan jenis DDoS sehingga dapat memberikan informasi jaringan kapan saja ketika dibutuhkan. Alasan utama adalah karena jaringan merupakan suatu sistem dinamis yang mana serangan dapat terjadi kapan saja. Maka dari itu, sistem pengawasan jaringan harus tersedia setiap saat dan selalu melaporkan informasi jaringan secara periodik.

2.2. Pemilihan teknologi *blockchain*

Blockhcain bekerja secara terdistribusi, yang mana setiap *node*/komputer terhubung satu sama lain secara langsung atau *peer-to-peer*. Dengan skema terdistribusi, pihak penyerang tidak akan mampu untuk menghapus data jaringan yang dilaporkan karena data tersebut turut pula disimpan keseluruh *node*/komputer yang terhubung ke jejaring *blockchain*. Untuk menghapus data pelaporan, maka pihak penyerang harus menghapus seluruh salinan data yang tersimpan di berbagai belahan dunia yang mana sangat mustahil untuk dilakukan. Oleh karena itu, data jaringan yang dilaporkan melalui jejaring *blockchain* akan tetap terjaga ketersediannya karena dapat diperoleh dari berbagai sumber. Dengan demikian, jejaring *blockchain* telah memenuhi syarat *availability*.

Jejaring *blockchain* memvalidasi keabsahan suatu transaksi dengan cara merujuk pada nilai *signature* dari transaksi. Selama pihak penyerang tidak mengetahi kunci *private* program *controller*, maka mereka tidak akan dapat memalsukan informasi jaringan karena tidak dapat menghasilkan nilai *signature* yang sah. Dengan demikian, teknologi *blockchain* mampu menjamin nilai integritas (*integrity*) data yang dilaporkan.

Jejaring *blockchain* bersifat terbuka, yang mana siapa saja dapat membaca data yang terekam untuk menjamin akuntabilitas jejaring *blockchain*. Untuk menjaga kerahasiaan data yang dilaporkan, maka pada penelitian ini, program controller akan selalu mengenkripsi data yang dilaporkan. Dengan data yang terenkripsi, maka sistem pelaporan berbasis *blockchain* mampu memenuhi aspek *confidentiallity*.

Jejaring *blockchain* terdiri dari berbagai jenis dan varian. Berdasarkan partisipasi pengguna dan metode pembuatan blok baru, jaringan *blockchain* terdiri dari tiga jenis yaitu: *permissionless* (Buterin, 2014), *permission-based* (Helebrandt, P. dkk, 2018) (Košťál, K., 2019) dan *private* (Mendez Mena, D. M. dkk, 2018). *Blockchain* jenis *permissionless* mengizinkan semua orang untuk bergabung baik dalam mencatatkan transaksi atau dalam berlomba-lomba dalam membuat blok baru dengan cara memecahkan tingkat kesulitan terkini. *Blokchain* jenis *permission-based* dan *private* memiliki kesamaan dalam hal partisipasi pengguna yang bersifat terbatas. Perbedaan keduanya terletak pada prosedur pembuatan blok. *Blockchain* jenis *permission-based* membuat blok baru berdasarkan konsensus dari pengguna, sedangkan pada jenis *private* pembutan blok baru ditentukan secara terpusat oleh pengelola *blokchain*.

Berdasarkan aspek integritas, ketiga jenis *blockchain* tersebut memilliki tingkat keamanan yang sama karena menggunakan *digital signature* yang dihasilkan menggunakan kriptografi asimetris.

Berdasarkan aspek kerahasiaan, *blokchain* jenis *permission-based* dan *private* memiliki tingkat kerahasian yang lebih baik dari pada jenis *permissionless* karena informasi jaringan yang tersimpan di dalam transaksi hanya dapat dibaca oleh kalangan terbatas. Meskipun demikian, hal tersebut tidak menjamin kerahasian data karena pengguna jejaring *blockchain* yang lain masih dapat membaca informasi jaringan. Untuk menjamin kerahasiaan data secara mutlak, maka ketiga jenis jejaring *blockchain* tersebut harus menggunakan sistem enkripsi data tambahan.

Berdasarkan aspek ketersediaan (*availablity*), *blockchain* jenis *permissionless* jauh lebih baik daripada *blockchain* jenis *permission-based* dan *private* karena data tersimpan di berbagai belahan dunia. Apabila suatu *node blockchain* atau koneksi internet di salah satu daerah terputus, maka pengguna masih dapat mengambil data dari *node* yang berada di daerah atau negara lain.

Penelitian ini menggunakan *blockchain* jenis *permissionless* berbasis jejaring Ethereum sebagai mana pada penelitian sebelumnya (El Houda dkk, 2019) (Kamboj, P. dkk, 2019) (Niya, S. R., 2018). Keterbaruan dari sistem yang diajukan adalah adanya penggunaan enkripsi AES untuk menjamin aspek kerahasiaan data yang dilaporkan. Penelitian ini menggunakan *blockchain* Ethereum karena dua faktor utama. Pertama, jejaring Ethereum menyediakan jaringan pengujian (*test network*) yang dapat digunakan tanpa harus membayar biaya transaksi. Kedua, jejaring Ethereum memiliki fitur *smart contract* yang dapat mengotomasikan proses autentifikasi transaksi. Pada penelitian ini, akan diuji coba dua jenis transaksi, yaitu transaksi berbasis *smart contract* dan transaksi berbasis pembayaran/*transfer*.

2.3. Metode pengukuran performa sistem

Pengujian sistem dilakukan dengan cara mengemulasikan sistem jaringan SDN menggunakan program Mininet (Kaur dkk, 2014). Untuk skenario jaringan multi-lokasi, satu lokasi jaringan diimplementasikan dalam satu buah *virtual machine* (VM). Setiap jaringan dikendalikan oleh sebuah *controller* yang terhubung ke jejaring *blockchain* melalui internet. *Controller* mencatatkan konfigurasi dan statistik jaringan ke jejaring *blockchain* setiap rentang waktu tertentu (t).

Analisa skalabilitas dihitung dengan cara menganalisa implikasi dari nilai resolusi pengambilan data (*update time* / U_{time}) dan besaran data (T_{size}) terhadap biaya pencatatan ke jejaring *blockchain* (*update cost* / U_{cost}).

Pada penelitian ini, akan dianalisa dua jenis kasus dan dua jenis skema pelaporan. Kedua kasus tersebut adalah: perangkat jaringan dengan jumlah *flow-rules* yang rendah (10 aturan), dan jumlah *flow-rules* tinggi (100 aturan). Sedangkan kedua jenis skema pelaporan adalah skema transaksi berbasis *smart contract* (Buterin, 2014) dan transaksi

berbasis *zero-payment* atau pembayaran dengan nilai nol (0).

Aturan dasar mengenai pengiriman transaksi ke jejaring *blockchain* adalah pengirim harus memastikan bahwa transaksi yang dikirim berhasil tercatatkan pada *block* terbaru sebelum mengirim transaksi berikutnya. Jika syarat tersebut tidak terpenuhi, maka transaksi sebelumnya boleh jadi tidak akan pernah tercatatkan selamanya atau tercatatkan secara tidak terurut. Untuk memastikan transaksi berhasil tercatatkan dengan baik, waktu pensamplingan dan pencatatan data ke jejaring *blockchain* (U_{time}) harus lebih besar dari waktu pembuatan blok baru (*block creation time* / B_{CT}) (persamaan 1).

$$U_{time} \geq B_{CT} \quad (1)$$

Hal lain yang perlu diperhatikan adalah adanya pembatasan ukuran transaksi untuk mencegah serangan *denial-of-services* (DoS) terhadap jejaring *blockchain*. Maka dari itu, ukuran input data sebuah transaksi tidak boleh melebihi batas kapasitas input data (*data size*) dan biaya transaksi per-blok (*block gas limit*).

Besaran informasi jaringan yang akan dicatakan (T_{size}) tergantung pada dua hal, yaitu: jumlah *flow-rule* yang dipasang pada setiap perangkat (N_R), dan frekuensi sampling data (U_{time}). Hubungan antara variabel ditunjukkan oleh persamaan 2. Nilai F_i merujuk kepada ukuran *flow-rule* nomor ke-i dalam satuan *byte*.

$$T_{size} = \sum_{i=1}^{N_R} F_i \quad (2)$$

Semakin besar informasi jaringan yang akan dicatatkan (T_{size}), maka biaya pencatatan dari transaksi (U_{cost}) juga semakin besar. Total biaya pencatatan (U_{cost}) bergantung pada ukuran input data transaksi (T_{size}), nilai *gas limit per byte* (GLPB), dan *gas price* (persamaan 3). Nilai GLPB bersifat konstan dan ditentukan oleh kode program *blockhcain*. Sedangkan nilai *gas price* pada umumnya ditentukan oleh pengirim dengan nilai minimal 1 Gwei (*giga wei*, 1 *wei* = satu per-milyar Ether).

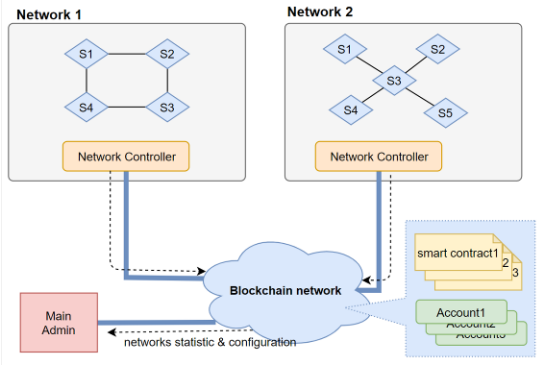
$$U_{cost} = T_{size} * GLPB * gas\ price \quad (3)$$

Hal lain yang perlu diperhatikan adalah nilai *gas price* menentukan seberapa cepat transaksi akan tercatatkan ke jejaring *blockchain*. Penambang/*miner* cenderung untuk terlebih dahulu memilih transaksi-transaksi dengan nilai *gas price* yang tinggi. Untuk mencegah pencatatan yang tidak terurut, maka nilai *gas price* harus bersifat konstan.

3. PERANCANGAN DAN IMPLEMENTASI SISTEM

Secara umum sistem terbagi kedalam tiga bagian utama sebagaimana ditampilkan pada Gambar 1. Pada sistem yang kami ajukan, hanya pengelola pusat (*Main admin*) dan program *controller* di setiap kantor cabang yang terhubung ke jejaring *blockchain*. Tiga bagian sistem adalah sebagai berikut:

1. Sistem jaringan SDN,
2. Aplikasi *controller* pengirim transaksi untuk pencatatan informasi jaringan ke jejaring *blockchain,*
3. Aplikasi pengawasan pusat.



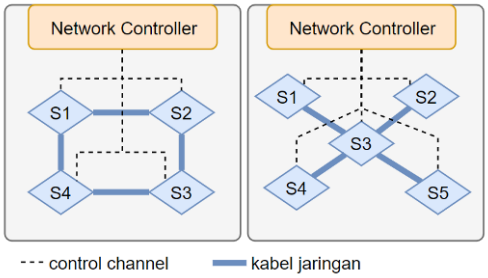
Gambar 1. Skema umum pengawasan jaringan terpusat berbasis *blockchain smart conract*.

3.1. Jaringan SDN

Pada jaringan SDN, sebuah *controller* jaringan bertugas untuk mengontrol seluruh perilaku perangkat jaringan. Pengelola jaringan dapat memprogram dan memonitor aturan-aturan (*flow-rules*) yang terpasang pada setiap perangkat jaringan menggunakan program *controller*.

Pada metode yang kami ajukan, untuk menghindari kesalahan konfigurasi perangkat jaringan oleh pengelola jaringan lokal, program *controller* secara berkala akan mengirimkan informasi mengenai konfigurasi dan statistik jaringan ke pengelola pusat. Program *controller* di jaringan lokal memiliki tambahan aplikasi berupa perekam konfigurasi dan statistik jaringan yang kemudian melaporkan rekaman tersebut melalui jejaring *blockchain*.

Pada penelitian ini digunakan protokol *OpenFlow* (McKeown dkk, 2008) untuk menghubungkan antara program *controller* dengan perangkat jaringan. Sedangkan jenis jaringan yang digunakan adalah jaringan *out-of-band* (Gambar 2). Setiap perangkat jaringan terhubung ke *controller* melalaui sebuah koneksi pengontrol (*control channel*) terdedikasi.



Gambar 2. Ssitem jaringan *out-of-band*, seluruh *switch* terkoneksi langsung ke *controller*.

Controller jaringan mengambil informasi konfigurasi dan statistik *flow-rules* dari setiap perangkat jaringan setiap t detik sekali dan menyimpan data tersebut kedalam database lokal. Sebuah proses anakan (*subprocess*) bertugas untuk mengirimkan data tersebut ke jejaring *blockchain* sesuai dengan alamat pengiriman yang telah ditentukan sebelumnya

Pada penelitian ini, digunakan tiga nilai t, yaitu: 15 detik, 1 menit, dan 10 menit. Semakin cepat waktu pengambilan sampel, maka semakin besar pula data yang harus dicatatkan ke jejaring *blockchain*. Meskipun hal ini akan meningkatkan biaya pencatatan (U_{cost}), waktu sampling yang cepat memungkinkan penangkapan perubahan konfigurasi jaringan secara lebih akurat. Sebagai contoh, pihak penyerang dapat merubah pengaturan jaringan dalam waktu singkat untuk kemudian mengembalikannya ke kondisi semula ketika sudah berhasil menyerang target. Apabila waktu sampel lebih besar dari pada waktu perubahan, maka perubahan pada jaringan tersebut tidak dapat terdeteksi oleh pengelola pusat.

Spesifikasi jaringan SDN yang diamati adalah sebagai berikut. Program *controller* jaringan yang digunakan adalah Ryu (Ryu SDN Framework, 2011) sedangkan perangkat jaringan yang digunakan adalah OpenVSwitch (Pfaff dkk, 2015) dengan protokol OpenFlow versi 1.3.

3.2. Aplikasi pengirim transaksi

Aplikasi pengirim transaksi merupakan *subprocess* dari *controller* Ryu yang diprogram menggunakan bahasa Nodejs dan pusataka ethers (Ethers.js, 2016). Sedangkan jejaring Ethereum yang digunakan untuk mencatatakan informasi jaringan adalah jejaring *test-netwok* Rinkeby yang berbasis *Proof-of-Authority* (PoA) (POA, 2014).

Terdapat dua jenis data yang dicatatkan oleh program ini ke jejaring *blockchain*: 1) data konfigurasi jaringan, dan 2) data statistik jaringan.

Data konfigurasi jaringan yang dikirim ke jejaring *blockchain* bertujuan untuk mencatat riwayat perubahan konfigurasi jaringan (*flow-rules* pada setiap perangkat jaringan) di setiap kantor cabang dan memberikan identitas penomoran khusus pada setiap *flow-rules* yang dipasang.

Data statistik jaringan berisi nomor identitas dari setiap *flow-rules* beserta data statistik dari paket jaringan yang dikenai oleh *flow-rules* tersebut. Data statistik paket terdiri dari: waktu pencatatan data, jumlah paket, dan ukuran paket dalam *bytes*.

Tabel 1 menampilkan contoh format data yang dikirim ke jejaring *blockchain*. Untuk menghemat biaya pengiriman transkasi, daat konfigurasi jaringan hanya akan dikirim apabila *flow-rules* yang bersangkutan merupakan *flow-rules* baru. Sedangkan untuk pengiriman normal hanya dikirim data statistik jaringan. Transaksi *smart contract* menggunakan enkoding Unicode yang mana 1 digit data ASCII memerlukan kapasitas 2 *bytes*. Pada penenlitian ini akan digunakan enkripsi jenis AES yang memerlukan tambahan kapasitas data sekitar 1,33 kali ukuran paket asli. Kapasitas yang diperlukan oleh setiap jenis transaksi dalam *bytes* dapat ditunjukkan oleh Table 1.

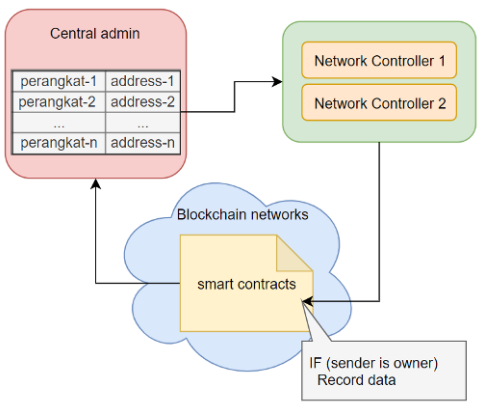
Tabel 1. Kapasitas total setiap jenis transkasi dalam *bytes*

Jenis informasi	*Flow-rules* id (digit)	Data (digit)	Total data (F_i) (bytes)
Statistik	4	24	75
Konfigurasi	4	300	809

Pada penelitian ini akan diamati dua jenis skema transaksi yaitu transaksi *smart contract* (T_{SC}), dan transaksi *zero-payment* (T_{ZP}). Pemilihan ini berdasarkan pertimbangan nilai ekonomis dan keaman dari kedua jenis transaksi tersebut.

Dari segi integritas data, kedua jenis transaksi sama-sama mampu menjaga keamanan data. Hanya saja dari segi *availability*, transaksi berbasis *smart contract* lebih bagus dari pada transaksi jenis *zero-payments*. Pada transaksi *smart contract,* hanya pemilik *smart contract* yang dapat mencatatakan transkasi. Sedangkan pada transaksi *zero-payment* semua pemilik akun/alamat di jejaring Ethereum dapat mengirimkan transaksi ke alamat penerima. Pengelola pusat harus melakukan pemilahan transaksi berdasarkan alamat blockchain dari setiap perangkat *switch* yang diawasi.

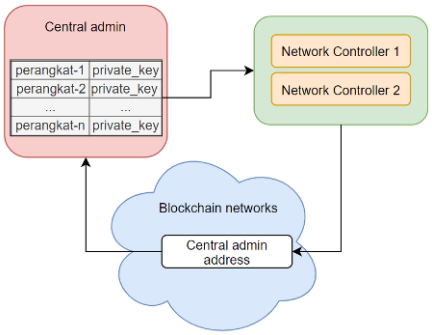
Secara ringkas, diagram transaksi berbasis *smart contract* ditampilkan oleh Gambar 3.



Gambar 3. Diagram pencatatan informasi jaringan berbasis transaksi *smart contract*

Pada pengiriman berbasis *smart contract,* setiap perangkat jaringan memiliki satu buah alamat *contract*. Pengelola pusat menentukan alamat *contract* untuk setiap perangkat *switch* dan mengatur program *controller* yang mengendalikan perangkat *switch* sebagai pemilik sah dari dari *contract* tersebut. Dengan demikian, *controller* dapat menuliskan data konfigurasi dan statistik perangkat tersebut ke masing-masing *smart contract* dari setiap perangkat *switch* karena memiliki *signature* yang sah. Di sisi lain, pengelola pusat dapat menganalisa informasi jaringan dengan merujuk pada data yang tersimpan pada *smart contract* tersebut.

Pada pengiriman berbasis transaksi *zero-payment*, program *controller* mengirimkan informasi jaringan menggunakan sebuah transaksi bernilai nol (0) ke alamat pengelola pusat (Gambar 4). Pertama-tama, pengelola pusat menginformasikan program *controller* mengenai alamat/akun *blockchain* dari setiap perangkat *switch* yang dikendalikan. Program *controller* menggunakan akun *blockchain* tersebut untuk mengirimkan informasi jaringan dari setiap perangkat *switch*. Data konfigurasi dan statistik jaringan disimpan kedalam input data dari transaksi *zero-payment*. Pengelola pusat dapat menganalisa informasi jaringan dengan mengunduh semua transaksi yang ditujukan ke alamatnya. Data dari setiap perangkat dapat dibedakan berdasarkan pada alamat pengirim transaksi.



Gambar 4. Diagram pencatatan informasi jaringan berbasis transaksi *zero-payment.*

3.3. Aplikasi pengawasan pusat

Aplikasi pengawasan pusat bertugas untuk merekap dan menganalisa perubahan konfigurasi dan statistik dari setiap perangkat dari seluruh jaringan *remote* atau kantor cabang. Pertama-tama, aplikasi harus mengekstrak seluruh transaksi *smart contract* (T_{SC}) atau transaksi *zero-payment* (T_{ZP}) yang ditujukan kepadanya. Pengelola pusat dapat langsung mengekstrak data tersebut apabila pengelola pusat juga menjalankan sebuah *node blockchain*. Apabila tidak berperan sebagai *node blockchain*, maka pengelola pusat harus terlebih dahulu mengunduh daftar transaksi tersebut dari penyedia yang terpecaya seperti "*etherscan.io"* dan "*infura.io"*. Karena informasi yang dikirim terenkripsi, maka aplikasi pengawasan pusat harus mendekripsi data laporan.

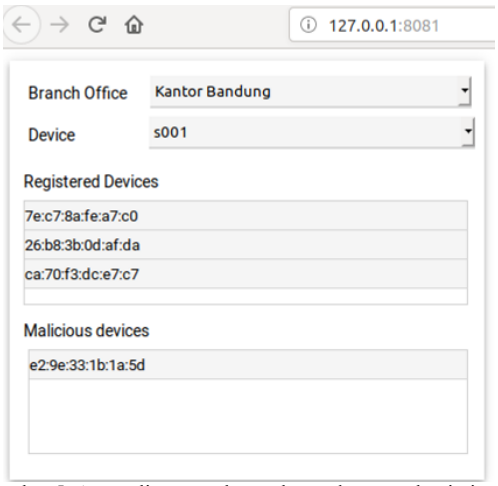
Terdapat dua aspek yang dianalisa oleh aplikasi pengawasan pusat, yaitu: 1) kesesuaian *flow-rule* di setiap perangkat jaringan dengan *flow-rule* yang ditetapkan, dan 2) anomali nilai pencacah dari paket jaringan yang dikenai oleh *flow-rules* tersebut. Dengan menganalisa kedua variabel tersebut, pengelola jaringan dapat memastikan bahwa perangkat jaringan di berbagai lokasi dapat berjalan sesuai dengan yang diharapkan dan tidak ada penyerang yang masuk ke jaringan.

Pengujian pada penelitian ini dilakukan pada dua kasus:

1. Deteksi akses ilegal ke jaringan tanpa izin,
2. Deteksi penyerangan ke perangkat *switch*.

Pada kasus pertama, pengelola pusat menerapkan konsep *honey trap* dengan mengizinkan akses masuk berdasarkan identifikasi alamat mesin perangkat (MAC *address*). Di sisi lain, pengelola pusat mewajibkan pegawai untuk mendaftarkan MAC *address* dari perangkat yang mereka gunakan. Dengan membandingkan data MAC *address* yang didaftarkan dengan data MAC *address* yang diproses oleh setiap perangkat *switch*, pengelola pusat dapat mendeteksi adanya upaya penyusupan oleh penyerang.

Dengan skema *honey-trap,* pihak penyerang merasa bahwa jaringan tidak diproteksi karena mereka dapat melakukan perintah *ping* ke perangkat lain di jaringan. Hanya saja, peyerang tidak dapat mengakses jaringan perusahaan lebih lanjut karena penggunaan *firewall* di setiap kantor cabang. Gambar 5 menunjukkan antarmuka pada progeram pengawasan pusat. Melalui program tersebut, pengelola pusat dapat mengetahui jumlah perangkat yang bersifat jahat/*malicious*, yaitu perangkat yang belum terdaftarkan tetapi sedang mengakses jaringan perusahaan.

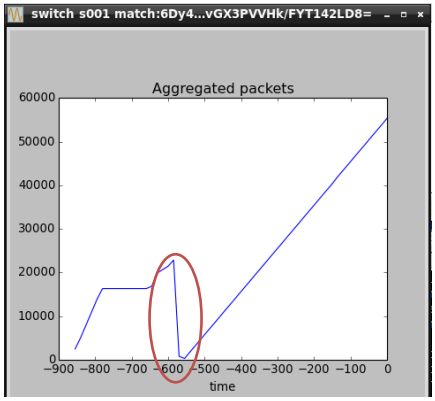


Gambar 5. Anomali pencacahan paket pada perangkat jaringan

Sebagai langkah penanganan, program controller dapat memasang kembali *flow-rules*

tersebut. Sebagai akibatnya, data pencacahan paket untuk *flow-rules* tersebut kembali bermula dari nol.

Pada kasus kedua, pengelola pusat dapat mengetahui kejadian penyerangan berdasarkan anomali data pencacahan paket (Gambar 6). Pengelola pusat dapat turun tangan dengan cepat untuk mengatasi insiden penyerangan tersebut. Lokasi masuk dari penyerang juga dapat diketahui karena sistem pelaporan statistik jaringan dilakukan per-perangkat jaringan. Pada pengujian ini penyerang masuk ke jaringan melalui *switch* berinisial s001.



Gambar 6. Anomali pencacahan paket pada perangkat jaringan

4. SKALABILITAS SISTEM PENCATATAN

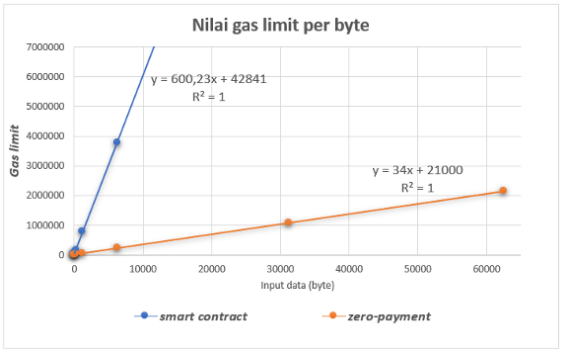
Skalabilitas sistem diukur berdasarkan biaya pengiriman transaksi pencatatan data. Berdasarkan persamaan 3, biaya transaksi (U_{cost}) ditentukan oleh besarnya ukuran transaksi yang dikirim (T_{size}) dengan cara menyesuaikan nilai *gas limit* dari transaksi. Dengan menggunakan nilai *gas limit* yang cukup, maka kemungkinan besar transaksi yang dikirim dapat tercatatkan ke jejaring *blockchain*. Nilai minimum *gas limit* yang diperlukan dapat dihitung dengan mengkalikan besar ukuran input data transaksi (T_{size}) dengan nilai *gas limit per byte* (GLPB).

Gambar 7 menunjukkan hasil percobaan pencatatan transaksi ke jejaring *blockchain* dengan ukuran input data yang bervariasi. Pengujian dilakukan pada dua jenis skema pengiriman: transaksi berbasis *smart contarct* (T_{SC}) dan *zero-payment* (T_{ZP}). Nilai *gas limit* minimal untuk kedua skema pengiriman tersebut dapat dituliskan sebagai sebuah fungsi terhadap input data (T_{size}), sebagaimana ditunjukkan oleh persamaan 5 dan 6. Satuan T_{size} adalah *byte*.

$$GL_{SC} = 600{,}2\,T_{size} + 42481 \quad (5)$$

$$GL_{ZP} = 34\,T_{size} + 21000 \quad (6)$$

Hasil pengujian menunjukkan bahwa skema *zero-payment* memiliki biaya yang lebih rendah daripada skema *smart contract* untuk kedua kasus tersebut. Hal ini dikarenakan pada transaksi *smart contract*, *node-node* pada jejaring *blockchain* harus menjalankan fungsi *smart contract* seperti mengecek alamat pengirim, menaikkan nilai pencacahan data, serta menyimpan data yang dikirim. Sedangkan untuk pengiriman berbasis *zero-payment*, *node-node blockchain* hanya sekedar mencatat transaksi tanpa melakukan eksekusi suatu fungsi. Sebagaimana telah dijelaskan pada sub bab 3.2, meskipun transaksi *smart contract* lebih mahal, akan tetapi transaksi *smart contract* memiliki hasil pencatatan yang lebih rapi dan aman dari serangan transaksi *spam*.



Gambar 7. Nilai *gas limit* sebagai fungsi besaran input data transaksi pada jaringan *blockchain* Ethereum-Rinkeby

Jejaring *blockchain* membatasi ukuran transaksi untuk mencegah serangan *denial-of-services* (DoS) (Bab 2). Jejaring Ethereum-Rinkeby memiliki batas *gas limit* sebesar 7 juta dan batas kapasitas input data transaksi sebesar 65 *Kbyte*. Berdasarkan persamaan 5, transaksi *smart contract* akan terlebih dahulu mengalami keterbatasan biaya daripada keterbatasan kapasitas. Ukuran input data maksimal yang didukung adalah sekitar 11500 *bytes*. Berdasarkan persamaan 6, transaksi *zero-payment* akan terlebih dahulu memiliki keterbatasan kapasitas (65 *Kbytes*).

Tabel 2 menunjukkan nilai *gas limit* yang dibutuhkan untuk mengirimkan informasi jaringan pada perangkat dengan jumlah *flow-rules* rendah (10 *flow-rules*) dan tinggi (100 *flow-rules*). Nilai T_{size} dalam *bytes* adalah N_R x F_i.

Tabel 2. Perbandingan *gas limit* pada transaksi jenis *smart contract* dan *zero-payment*

(N_R)	Jenis	T_{size} (bytes)	Gas limit (T_{SC})	Gas limit (T_{ZP})
10	*Stats*	750	492.631	46.500
10	*Config*	8.090	4.898.099	296.060
100	*Stats*	7.500	4.543.981	276.000
100	*Config*	80.900	48.598.661	2.771.600

Berdasarkan Tabel 2, pengiriman data konfigurasi (*Configs*) pada jaringan dengan *flow-rules* tinggi (100) tidak dapat dilakukan dengan satu buah transaksi karena melebihi kapasitas input data maksimum 65 *Kbyte*. Namun demikian, karena data konfigurasi hanya perlu dikirim ketika terjadi

perubahan, hal ini tidak terlalu berpengaruh pada skalabilitas sistem. Karena perubahan konfigurasi jaringan tidak terlalu sering terjadi, maka potensi terjadinya *race-condition* pada pengiriman transaksi dapat diminimalkan.

Hal yang paling mempengaruhi skalabilitas sistem adalah pada pengiriman statistik jaringan (*Stats*) karena harus dicatat secara periodik. Ancaman skalabilitas pada pengiriman data statistik berasal dari biaya pengiriman transaksi yang tinggi. Sedangkan ancaman skalabilitas dari *race condition* sangat minimal karena data statistik dapat dikirimkan dalam sekali transaksi tanpa melebihi ambang batas ukuran input data ataupun *gas limit*.

Tabel 3. Perkiraan biaya pencatatan statistik jaringan per-hari untuk $N_R = 100$.

Jenis transaksi	U_{cost}/Trx (Ether)	U_{cost}/day @15 dtk	U_{cost}/day @1 mnt	U_{cost}/day @10 mnt
T_{SC}	0,004544	392,6016	6,54336	0,654336
T_{ZP}	0,000276	23,8464	0,39744	0,039744

Tabel 3 menunjukkan perkiraan biaya pengiriman transaksi untuk pencatatan informasi jaringan ke jejaring *blockchain*. Nilai *gas-price* per-transkasi yang digunakan adalah 1 Gwei. Perkiraan dilakukan pada tiga jenis waktu pensampelan yang berbeda (15 detik, 1 menit, dan 10 menit). Kasus yang diamati adalah perangkat dengan jumlah *flow-rules* tinggi (100 buah).

Berdasarkan Tabel 3, transaksi *zero-payment* hanya memerlukan biaya pengiriman sekitar 6 % dari biaya transaksi *smart contract*. Dengan demikian, transaksi berbasis *zero-payment* memiliki skalabilitas biaya pengiriman transaksi yang lebih bagus. Pada transaksi *zero-payment*, biaya per-bulan per-perangkat yang diperlukan adalah 30 x 0,039744 = 1,19232 ethers.

Meskipun biaya bulanan per-perangkat hanya berkisar 1,19232 Ethers, skema pencatatan tersebut tidak sesusai untuk diimplementasikan pada jejaring Ethereum *main-network* yang berbasis *Proof-of-Work* (PoW) (Buterin, 2014) karena biaya riil yang diperlukan sangat tinggi. Pada saat penelitian ini dilakukan, harga 1 Ether adalah sekitar $ 200. Dengan demikian, biaya bulanan yang diperlukan untuk mencatatkan informasi per-perangkat jaringan adalah sekitar $ 238,464. Apabila perusahaan memiliki N-buah perangkat jaringan, maka total biaya yang diperlukan menjadi N kali lipat.

5. DISKUSI

Alternatif skema pengiriman data yang lebih ekonomis adalah dengan menggunakan jejaring penyimpanan file terdistribusi seperti *Inter-planetary File Sytem* (IPFS). Penggunaan jejaring IPFS untuk meminimalisir pencatatan data telah diimplementasikan pada beberapa penelitian terdahulu (Gries, S. dkk, 2018) (de Tazoult dkk, 2019). Data yang akan dilaporkan disimpan di jejaring IPFS sedangkan nilai hash dari data tersebut dicatatkan di jejaring *blockchain*. Perubahan data pelaporan dapat diketahui dengan membandingkan nilai hash dari data yang tersimpan di jejaring IPFS dengan nilai hash yang tercatat di jejaring *blockchain*. Biaya pencatatan dapat diminimalisir karena hanya nilai hash dari data pelaporan yang perlu dicatatkan ke jejaring *blockchain*.

Penelitian ini tidak menggunakan penyimpanan berbasis jejaring IPFS karena sistem penyimpanan berbasis IPFS tidak dapat sepenuhnya menjamin ketersediaan (*availability*) data. Data yang tersimpan di jejaring IPFS tidak otomatis terreplikasi di berbagai tempat yang berbeda. Pengguna harus memastikan sendiri proses replikasi data ke berbagai server yang berbeda lokasi agar terjaga ketersediaan salinan datanya. Pemilihan *node-node* yang bertugas untuk menduplikasi data pelaporan perlu ditentukan dengan seksama dan merupakan topik penelitian tersendiri.

Penelitian ini menggunakan jejaring Ethereum Rinkeby yang berbasis *Proof-of-Authotrity* (PoA). Kelemahan dari skema PoA adalah tingkat ketersediaan data (*availability*) lebih rendah dibandingkan dengan skema PoW. Data pelaporan hanya tersimpan pada beberapa *nodes authority*. Untuk meningkatkan ketersedian data, maka jumlah *node authority* perlu ditambah.

Pengelola pusat dapat berpatisipasi dengan berperan sebagai salah satu *node authority,* sehingga memiliki salinan data pelaporan secara lokal. Apabila penyerang berhasil memutus koneksi internet di kantor pusat, pengelola pusat masih memiliki salinan rekaman data pelaporan. Rekaman data tersebut dapat digunakan untuk melacak skema penyerangan yang telah dilakukan, sehingga perbaikan sistem jaringan dapat dilakukan secara lebih cepat dan terarah.

6. KESIMPULAN

Pada penelitian ini dikembangkan purwarupa pengawasan informasi jaringan secara terpusat menggunakan jejaring Ethereum-Rinkeby. Sistem jaringan yang diawasi adalah jaringan berbasis SDN. Skema transaksi untuk pencatatan informasi jaringan terdiri dari dua jenis, yaitu transaksi berbasis *smart contract,* dan transaksi berbasis *zero-payment*. Hasil pengujian menunjukkan bahwa, transaksi *zero-payment* lebih efisien dengan hanya membutuhkan sekitar 6 % dari biaya transaksi *smart contract*. Metode pencatatan yang diajukan pada penelitian ini lebih sesuai untuk diimplementasikan pada jejaring Ethereum *Proof-of-Authority* dibandingkan *Proof-of-Work* karena biaya pencatatan bulanan per-perangkat yang tinggi, sekitar 1,12 Ether.

DAFTAR PUSTAKA

BUTERIN, V. 2014. A next-generation smart contract and decentralized application platform. white paper, 3, 37.

DE TAZOULT, C. T., CHIKY, R., & FOLTESCU, V. 2019. A Distributed Pollution Monitoring System: The Application of Blockchain to Air Quality Monitoring. In International Conference on Computational Collective Intelligence (pp. 688-697). Springer, Cham.

EL HOUDA, Z. A., HAFID, A. S., & KHOUKHI, L. 2019. Cochain-SC: An intra-and inter-domain Ddos mitigation scheme based on blockchain using SDN and smart contract. IEEE Access, 7, 98893-98907.

Ethers.js. 2016. What is ethers.js. [online] Tersedia di <https://github.com/ethers-io/ethers.js/> [Diakses 2 Oktober 2019]

FIESSLER, A., LORENZ, C., HAGER, S., & SCHEUERMANN, B. . 2018. FireFlow-High Performance Hybrid SDN-Firewalls with OpenFlow. In 2018 IEEE 43rd Conference on Local Computer Networks (LCN) (pp. 267-270). IEEE.

GERMANN, B., SCHMIDT, M., STOCKMAYER, A., & Menth, M. 2018. OFFWall: A Static OpenFlow-Based Firewall Bypass. In 11. DFN-Forum Kommunikationstechnologien. Gesellschaft für Informatik eV.

GRIES, S., MEYER, O., WESSLING, F., HESENIUS, M., & GRUHN, V. 2018. Using Blockchain Technology to Ensure Trustful Information Flow Monitoring in CPS. In IEEE International Conference on Software Architecture Companion (ICSA-C) (pp. 35-38).

HELEBRANDT, P., BELLUS, M., RIES, M., KOTULIAK, I., & KHILENKO, V. 2018. Blockchain Adoption for Monitoring and Management of Enterprise Networks. In IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) (pp. 1221-1225).

KAMBOJ, P., & PAL, S. 2019. QoS in software defined IoT network using blockchain based smart contract. In Proceedings of the 17th Conference on Embedded Networked Sensor Systems (pp. 430-431). ACM.

KAUR, K., SINGH, J., & GHUMMAN, N. S. 2014. Mininet as software defined networking testing platform. In International Conference on Communication, Computing & Systems (ICCCS) (pp. 139-42).

KOŠŤÁL, K., HELEBRANDT, P., BELLUŠ, M., RIES, M., & KOTULIAK, I. 2019. Management and Monitoring of IoT Devices Using Blockchain. Sensors, 19(4), 856.

MENDEZ MENA, D. M., & YANG, B. (2018, September). Blockchain-Based Whitelisting for Consumer IoT Devices and Home Networks. In Proceedings of the 19th Annual SIG Conference on Information Technology Education (pp. 7-12). International World Wide Web Conferences Steering Committee.

MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., ... & TURNER, J. 2008. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2), 69-74.

NIYA, S. R., JHA, S. S., BOCEK, T., & STILLER, B. 2018. Design and implementation of an automated and decentralized pollution monitoring system with blockchains, smart contracts, and LoRaWAN. In NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium (pp. 1-4).

PATIL, S., & SUBHEDAR, M. S. 2019. Optimizing MPLS Tunnel Creation Performance by Using SDN. In Soft Computing and Signal Processing (pp. 527-536). Springer, Singapore.

PFAFF, B., PETTIT, J., KOPONEN, T., JACKSON, E., ZHOU, A., RAJAHALME, J., ... & AMIDON, K. 2015. The design and implementation of open vswitch. In 12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15) (pp. 117-130).

PHATAK, A., KADIKAR, R., VIJAYAN, K., & AMUTHA, B. ,2018. Performance Analysis of Firewall Based on SDN and OpenFlow. In 2018 International Conference on Communication and Signal Processing (ICCSP) (pp. 0611-0615). IEEE.

POA. 2014, Welcome to POA. [online] Tersedia di <https://www.poa.network/ PoA> [Diakses 2 Oktober 2019]

Ryu SDN Framework. 2011. [online] Tersedia di <https://osrg.github.io/ryu/> [Diakses 1 Oktober 2019]

SUBRATIE, K., & FIGUEIREDO, R. (2018, October). Towards Island Networks: SDN-Enabled Virtual Private Networks with Peer-to-Peer Overlay Links for Edge Computing. In International Conference on Internet and Distributed Computing Systems (pp. 122-133). Springer, Cham.

VAN DER POL, R., GIJSEN, B., ZURANIEWSKI, P., ROMÃO, D. F. C., & KAAT, M. 2016. Assessment of SDN technology for an easy-to-use VPN service. Future Generation Computer Systems, 56, 295-302.

Halaman ini sengaja dikosongkan