Pendekatan *unsupervised* untuk Mendeteksi Serangan Tingkat Rendah pada Jaringan Komputer

Baskoro Adi Pratomo

Departemen Teknik Informatika,
Institut Teknologi Sepuluh Nopember
Jl. Teknik Kimia, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur, Indonesia

Email: [1]baskoro@if.its.ac.id

Tersedia Online di
http://www.jurnal.unublitar.ac.id/index.php/briliant

Sejarah Artikel
Diterima pada 13 Maret 2022
Disetuji pada 28 Mei 2022
Dipublikasikan pada 31 Mei 2022
Hal. 546-564

Kata Kunci:
Deteksi intrusi*; deep learning;* serangan tingkat rendah

DOI:
http://dx.doi.org/10.28926/briliant.v7i2.1004

Abstrak: Serangan tingkat rendah merupakan serangan yang diam-diam masuk ke dalam sistem tanpa mengirimkan paket dalam jumlah besar. Contoh dari serangan jenis ini adalah exploit, backdoors, dan worms. Untuk mencegah serangan jenis ini, kami mengusulkan sistem deteksi intrusi dengan menggunakan Recurrent Neural Network dan Autoencoders. Pendekatan *unsupervised* yang diusulkan mampu mengidentifikasi serangan tingkat rendah dalam koneksi jaringan, mengesampingkan persyaratan sampel berbahaya pada data pelatihan. Pendekatan yang diusulkan memberikan peningkatan *detection rate* setidaknya 12,04% dari penelitian sebelumnya.

PENDAHULUAN

Kerentanan baru muncul setiap hari, dan peretas selalu memiliki cara baru untuk mengeksploitasinya. Selain itu, sebagian besar serangan ini dapat dilakukan dari jarak jauh, dan musuh dapat meluncurkan serangan dari mana saja dan menargetkan berbagai protokol pada lapisan aplikasi. Misalnya, Cisco Data Center Network Manager memiliki kerentanan di mana pengguna yang diautentikasi dapat mengunggah file *Web Application Resources* (WAR) melalui HTTP yang berisi skrip berbahaya dan menjalankan perintah dari jarak jauh sebagai root (CVE-2019-1619, 2019). Dengan memanfaatkan celah ini, peretas dapat menguasai komputer kita tanpa harus hadir secara fisik di depan komputer. Kami menyebut serangan-serangan seperti ini sebagai *serangan tingkat rendah*. Tingkat rendah bukan diukur dari seberapa berbahaya serangan tersebut, tetapi dilihat dari jumlah pesan yang dikirimkan pada jaringan komputer.

Studi yang lebih baru menunjukkan peningkatan dari penelitian lama (Whalen, Boggs, & Stolfo, 2014), (Oza, Ross, Low, & Stamp, 2014), (Hamed, Dara, & Kremer, 2018), (Shen, Wei, Zhu, & Wang, 2017). Mereka juga mengevaluasi pendekatan mereka dengan dataset yang lebih baru yang didapat dari koneksi yang dihasilkan sendiri atau dibuat secara sintetis (seperti, dataset ISCX12

dan UNSW-NB15 (Moustafa & Slay, 2015)). Namun demikian, studi terbaru ini hanya fokus pada HTTP, meskipun pemodelan mereka tidak mempertimbangkan sebuah protocol secara spesifik dan dapat digunakan untuk mendeteksi serangan pada protokol lain.

Kedua masalah ini pun akhirnya memunculkan pertanyaan: mengingat penelitian terbaru menunjukkan bahwa penelitian sebelumnya pada deteksi serangan tingkat rendah di beberapa protokol sudah ketinggalan zaman dan kinerjanya turun ketika berhadapan dengan serangan tingkat rendah kontemporer, bagaimana meningkatkan kinerja model deteksi serangan tingkat rendah untuk menangani serangan tingkat rendah pada berbagai protokol yang semakin menyebabkan kerusakan pada jaringan perusahaan?

Algoritme Machine Learning (ML) mampu mengklasifikasikan objek dan artefak berdasarkan fitur yang ada dalam data dan menangani berbagai modalitas input. (ML) telah berhasil diterapkan di berbagai domain dengan tingkat keberhasilan yang tinggi, seperti klasifikasi gambar, pemrosesan bahasa alami, pengenalan suara, dan bahkan sistem deteksi intrusi. Ada juga berbagai penelitian tentang penerapan pembelajaran mesin untuk mengatasi deteksi intrusi jaringan (Sommer & Paxson, 2010).

Ada dua pendekatan utama untuk melatih model (ML), *supervised* dan *unsupervised*. *Unsupervised learning* lebih umum digunakan untuk menarik kesimpulan dari data tanpa label yang diketahui. Salah satu pekerjaan yang termasuk dalam kategori ini adalah deteksi anomali/pencilan. Deteksi anomali adalah kasus di mana model dilatih melalui sampel data 'normal' atau dalam kasus ini, koneksi yang sah, dan kemudian model ditugaskan untuk mencari data baru yang menyimpang dari pola perilaku normal. Pendekatan ini sangat berguna ketika nilai fitur dari kelas target sering berubah atau nilainya tidak pernah tetap (yaitu, serangan siber). Pendekatan *unsupervised* juga tidak memerlukan sampel koneksi berbahaya untuk melatih model, dan karena itu cenderung berkinerja lebih baik di lingkungan yang berubah-ubah seperti pada jaringan komputer. Manfaat ini membawa kami untuk fokus pada pendekatan *unsupervised* dalam artikel ini.

Kinerja algoritma ML sangat bergantung pada representasi data yang diberikan. Representasi ini, juga biasa disebut sebagai fitur, dapat dibuat secara manual dengan memanfaatkan masukan dari pengetahuan ahli pada domain yang diteliti. Fitur yang dibuat secara manual dari muatan jaringan akan sulit diperoleh karena ada banyak protokol lapisan aplikasi dengan berbagai format pesan. Fitur abstrak dapat menangkap variasi data sehingga pengamatan data menjadi lebih baik (Goodfellow, Bengio, & Courville, 2016). Misalnya, browser yang berbeda meminta halaman web yang sama dari server web mengirim pesan permintaan HTTP yang sedikit berbeda. Namun, pesan ini akan berbeda dari pesan yang dikirim oleh musuh untuk mengeksploitasi kerentanan di server web. Fitur tingkat tinggi dan abstrak dapat menangani variasi pesan yang dikirim oleh browser yang berbeda, dan model akan tetap mengklasifikasikannya sebagai koneksi yang sah.

Penelitian-penelitian tersebut kemudian menerapkan metode deteksi yang menggunakan representasi abstrak dari payload paket untuk mengidentifikasi serangan. Beberapa dari penelitian ini menggunakan model statistik berbasis ambang batas (threshold) (Wang & Stolfo, Anomalous Payload-Based Network

Intrusion Detection, 2004), (Bolzoni, Etalle, & Hartel, 2006), (Wang, Parekh, & Stolfo, Anagram: A content anomaly detector resistant to mimicry attack, 2006), (Rieck & Laskov, 2007), (Oza, Ross, Low, & Stamp, 2014), sementara yang lain menggunakan algoritme ML, misalnya Logistic Regression (Whalen, Boggs, & Stolfo, 2014), SVM (Perdisci, Ariu, Fogla, Giacinto, & Lee, 2009), Hidden Markov Model (Ariu, Tronci, & Giacinto, 2011). Terlepas dari kenyataan bahwa penelitian-penelitian ini, baik yang menggunakan representasi abstrak dari payload, model statistik dan algoritme ML konvensional, tidak bekerja dengan baik ketika menghadapi sejumlah besar data (LeCun, Bengio, & Hinton, 2015). Oleh karena itu, kami berpendapat bahwa ada ruang untuk perbaikan untuk metode deteksi. Deep Learning (DL), yang juga merupakan jenis algoritme ML, dapat menangani abstraksi data tingkat tinggi lebih baik daripada algoritme ML konvensional dan bekerja dengan baik pada volume data yang besar (Goodfellow, Bengio, & Courville, 2016). Meskipun, tidak semua arsitektur pembelajaran mendalam kompatibel dengan pembelajaran tanpa pengawasan.

Oleh karena itu, dalam paper ini, kami akan menyelidiki seberapa baik model DL tanpa pengawasan dengan fitur berbasis muatan dapat mengidentifikasi serangan tingkat rendah dalam jaringan. Paper ini tersusun sebagai berikut: Bagian 2 kami akan membahas arsitektur DL mana yang dapat digunakan untuk deteksi outlier. Setelah itu, metode deteksi yang diusulkan dijelaskan di Bagian 3. Pada Bagian 4, kami menunjukkan bagaimana kinerja metode yang kami usulkan pada kumpulan data terbaru dan beberapa protokol. Untuk membuktikan peningkatan metode yang kami usulkan dibandingkan yang paling mutakhir, kami membandingkan hasil eksperimen kami dengan NIDS lainnya. Penelitian-penelitian yang ada dipilih berdasarkan kode sumber atau ketersediaan kode semu dan apakah pendekatannya tidak diawasi. Semua penelitian ini kemudian dievaluasi menggunakan kumpulan data terbaru, UNSW-NB15 (Moustafa & Slay, 2015) dan BlattaSploit. Bagian 5 menyimpulkan hasil penelitian ini.

METODE

Di bagian ini, kami akan membahas lebih detail bagaimana RNN dan Autoencoders digunakan untuk mendeteksi serangan tingkat rendah. Kami kemudian akan mencari tahu kinerja kedua pendekatan tersebut, mana yang lebih cocok untuk mengidentifikasi serangan tingkat rendah di Bagian Eksperimen dan Hasil.

Rekonstruksi Pesan Layer Aplikasi

Pesan pada lapisan aplikasi (seperti HTTP, FTP, SMTP) biasanya lebih panjang dari ukuran maksimum dari sebuah paket IP. Oleh karena itu, pesan tersebut perlu dibagi menjadi beberapa paket IP dan dikirim satu per satu. Di tujuan, mereka bisa tiba secara berurutan, bisa tidak. Sebuah Network Intrusion Detection System (NIDS) berbasis *payload* butuh membaca pesan lapisan aplikasi secara utuh sehingga memiliki gambaran yang lebih lengkap tentang pesan yang ditransmisikan dan dengan demikian mampu mendeteksi serangan dengan lebih akurat. Oleh karena itu, dalam usulan kami, ketika sebuah paket IP tiba, paket tersebut tidak langsung diproses oleh detektor outlier. Paket dimasukkan ke dalam antrian dan

kemudian, bersama dengan paket lain dari koneksi yang sama (diidentifikasi dari alamat IP dan port-nya), disusun kembali menjadi pesan lapisan aplikasi.

Kami menerapkan penyusunan ulang pesan lapisan aplikasi dengan mengikuti standar RFC 793 (Transmission Control Protocol, 1981), karena ada bug di pustaka PyNIDS (Mitrecnd, 2014). PyNIDS menambahkan byte acak di akhir pesan yang direkonstruksi. Hal ini akan membingungkan model RNN dan Autoencoder kami. PyNIDS adalah Python wrapper untuk libNIDS yang ditulis dalam C dan tidak dipelihara sejak delapan tahun yang lalu, oleh karena itu menanti perbaikan bug akan memakan waktu lama. Kami memverifikasi implementasi kami dengan membandingkan pesan lapisan aplikasi yang direkonstruksi dengan fitur Follow TCP Stream di Wireshark, dan hasilnya sesuai.

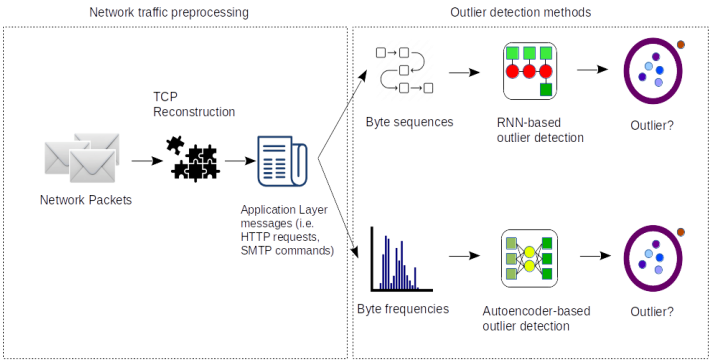
Pesan dari setiap protokol lapisan aplikasi perlu dikelompokkan karena setiap protokol dianalisis secara independen. Dengan kata lain, kita akan memiliki model untuk setiap protokol. Model-model ini dilatih dengan metode yang sama, tetapi data pelatihan yang digunakan berbeda. Oleh karena itu, metode yang kami usulkan dapat mendeteksi serangan tingkat rendah pada banyak protokol, tidak seperti metode-metode sebelumnya

Model Deteksi Outlier

Di bagian ini, kami fokus pada bagaimana kami membangun struktur Deep Learning dan bagaimana kami membuat model sesuai data yang kami gunakan, pesan lapisan aplikasi. Bagian ini juga merumuskan cara kami mengubah pesan lapisan aplikasi menjadi kumpulan fitur, melatih model, dan memasukkannya ke algoritme.

Karena metode yang kami usulkan unsupervised, pertama-tama kami mengumpulkan koneksi yang sah dari dataset, merekonstruksi pesan lapisan aplikasinya, dan melatih model dengan data pelatihan tersebut. Pada fase pelatihan, model diharapkan mempelajari pola payload yang sah/normal. Setelah itu, model akan dapat mengidentifikasi setiap penyimpangan dari pola normal dan menandainya sebagai berbahaya.

Kami mendefinisikan M_p sebagai kumpulan pesan m dari protokol lapisan aplikasi p (seperti, HTTP dan SMTP). Kedua model yang kami usulkan (DO-RNN dan DO-AE) bekerja dengan pesan M_p, tetapi setiap model memiliki representasi *payload*-nya sendiri sebagai input karena setiap model memiliki metode yang berbeda untuk mendeteksi outlier. Model berbasis RNN menggunakan deret nilai byte untuk mewakili m_p, sedangkan model berbasis Autoencoders menggunakan frekuensi byte sebagai representasi. Kedua pendekatan dijelaskan lebih lanjut di bagian berikut.



Gambar 1 Ilustrasi cara kerja DO-RNN dan DO-AE

Deteksi Outlier dengan Recurrent Neural Network

Model deteksi outlier (DO-RNN) berbasis RNN kami bekerja berdasarkan prinsip bahwa serangan tingkat rendah akan memiliki deret byte yang tidak seperti biasanya. DO-RNN mengambil deret byte yang tumpang tindih dengan panjang tertentu. Kemudian dilatih untuk memprediksi byte berikutnya dari setiap deret. Proses pelatihan memungkinkan DO-RNN untuk mempelajari deret byte yang biasanya muncul dalam koneksi yang sah. Serangan tingkat rendah diharapkan membuat model RNN sering mengeluarkan prediksi byte berikutnya yang salah karena payload mungkin berisi deret byte yang jarang muncul dalam koneksi yang sah.

Secara formal, kami mendefinisikan deret $B = \{b_1, b_2, \dots, b_l\}$ dengan panjang *l* sebagai deret byte, di mana setiap byte b_i adalah byte dalam pesan lapisan aplikasi $m_p$dengan panjang *l*. Kemudian, kami memilih beberapa nilai *n*, dimana *n* > *l*. Kemudian kita mendefinisikan himpunan *S* seperti pada persamaan **(*1*)** di mana setiap s_i adalah bagian dari deret *B* dengan panjang *n* + *1*.

$$S = \{s_i | \, s_i = (b_i, b_{i+1}, b_{i+2}, \dots, b_{i+n+1}) \; untuk \; semua \; 1 \leq i \leq l - n + 2\} \quad \textbf{(1)}$$

Kami kemudian membagi setiap subderet s_i menjadi *x* dan *y* di mana *i* adalah input untuk model DO-RNN kami dan *y* adalah output target:

$$x = \{s_k \mid 0 \leq k < n\} \quad \textbf{(2)}$$

$$y = s_n \quad \textbf{(3)}$$

Oleh karena itu, kami melatih fungsi F_p, di mana fungsi F_p adalah pengklasifikasi berbasis Recurrent Neural Network untuk memprediksi elemen berikutnya $\hat{y}$ dari deret *x*, di atas set *S* (karena *S* sekarang didefinisikan sebagai himpunan semua deret s_i) dalam protokol *p* dalam fase pelatihan. Fungsi F_p kemudian dioptimasi dengan *backpropagation* sehingga prediksi keluaran $\hat{y}$ sama dengan nilai sebenarnya dari *y*. Misalnya, selama fase pelatihan, F_{HTTP} membaca semua pesan HTTP, F_{FTP} menerima pesan FTP, dan F_{SMTP} menangani SMTP pesan. Setelah mendapatkan pesan, masing-masing fungsi tersebut akan mengumpulkan sekumpulan subderet *s*.

Menyatakan pengklasifikasi RNN kami sebagai $F_p(x)$ akan menjadi penyederhanaan yang berlebihan karena ada banyak operasi yang ada di dalam fungsi tersebut. Jadi, kami merumuskan fungsi kami $F_p(x)$ secara lebih rinci sebagai:

$$F_p(x) = argmax(SF(R(E(x))))$$

(4)

Di mana *E(x)* adalah fungsi yang mengubah setiap *x* menjadi vektor dengan ukuran dimensi tertentu. Fungsi itu juga bisa disebut sebagai lapisan Embedding. Fungsi *R* adalah lapisan Recurrent yang mengambil vector *embedded* sebagai input dan mengeluarkan nilai vektor yang kemudian diproses oleh fungsi Softmax *SF*. Output dari fungsi Softmax *SF* adalah probabilitas sebuah byte menjadi elemen berikutnya dari deret tersebut. Fungsi *argmax* mengeluarkan elemen dengan probabilitas tertinggi.

Sampai saat ini kami selalu menggunakan istilah RNN. Akan tetapi, kami menggunakan LSTM dan GRU - jenis RNN yang dapat bekerja lebih baik dengan deret yang lebih panjang karena mereka dapat menyaring informasi mana yang harus diteruskan ke perhitungan langkah waktu berikutnya dengan memanfaatkan gerbang (*gate*) (Hochreiter & Schmidhuber, 1997), (Chung, Gulcehre, Cho, & Bengio, 2014). Namun secara umum, tujuan fungsi *R* tidak berubah, begitu pula dimensi input dan outputnya.

Untuk mengukur seberapa jauh koneksi baru dari model normal, DO-RNN perlu menghitung skor anomali. Pertama-tama kita mendapatkan skor anomali dari setiap pesan lapisan aplikasi yang digunakan pada fase pelatihan untuk menghitung ambang batas yang nantinya akan digunakan pada fase deteksi. Pesan lapisan aplikasi yang memiliki skor anomali di atas ambang batas akan dianggap berbahaya.

Seperti yang diilustrasikan pada **Gambar *1***, model DO-RNN kami bekerja dengan mengambil deret *x* dan memprediksi nilai berikutnya $\hat{y}$. Skor anomali pada dasarnya adalah perbedaan antara output yang diprediksi dan nilai aktual, namun perbedaan ini dapat diukur secara terpisah atau terus menerus. Oleh karena itu, pada bagian ini, kami mengusulkan dua metode untuk menghitung skor anomali a_p dari pesan lapisan aplikasi m_p, yaitu skor biner dan skor *floating*.

Skor anomali biner dihitung dengan menghitung jumlah prediksi yang benar yang dibuat model. $\hat{Y}$ menjadi kumpulan elemen yang diprediksi $\hat{y}$:

$$\hat{Y} = \{\hat{y}_i \,|0 \leq i < (l-n)\}$$

(5)

Kemudian skor anomali biner $a_p^{\{binary\}}$ dari pesan m_p didefinisikan sebagai berikut:

$$a_p^{\{binary\}} = \frac{\sum_{i=0}^{l-n} v_i}{l} \begin{cases} v_i = 1, \hat{y}_i = y_i \\ v_i = 0, sebaliknya \end{cases}$$

(6)

Skor anomali *floating* dihitung dengan mengakumulasikan selisih probabilitas prediksi $\hat{y}$, yang dapat diperoleh dari output fungsi Softmax, dan probabilitas aktual *y*. Oleh karena itu, jika $Prob(y)$ adalah probabilitas *y*, skor anomali floatin $a_p^{\{float\}}$ dari pesan m_p didefinisikan sebagai:

$$a_p^{\{float\}} = \frac{\sum_{i=0}^{l-n} \left(Prob(\widehat{y}) - Prob(y) \right)^2}{l}$$

(7)

Pada fase deteksi, pertama-tama kita mendapatkan satu set deret *s* dari pesan lapisan aplikasi seperti yang dijelaskan oleh Persamaan **(*1*)**. Setiap baris *s* dibagi menjadi *x* dan *y* seperti yang dijelaskan oleh Persamaan **(*2*)** dan **(*3*)**. Selanjutnya, skor anomali (yaitu, biner dan *floating*) dihitung. Peringatan dimunculkan ketika skor anomali melampaui ambang batas (metode mendefinisikan ambang batas dijelaskan secara rinci di Bagian Menentukan Ambang Batas).

Deteksi Outlier dengan Autoencoders

Seperti yang dijelaskan di Bagian sebelumnya, Autoencoders adalah model yang mencoba menyalin input ke outputnya. Perilaku ini membuat Autoencoders dapat mempelajari representasi data. Dengan kata lain, Autoencoders dilatih untuk menghafal pola dari data pelatihan. Dalam bagian ini, kami mengembangkan model yang dapat mengingat pola data pada koneksi yang sah yang kami sebut sebagai DO-AE. **Gambar *1*** umumnya menunjukkan cara kerja DO-AE.

Kami merepresentasikan pesan lapisan aplikasi sebagai frekuensi byte karena serangan tingkat rendah akan menunjukkan distribusi frekuensi byte yang berbeda. Kita bisa saja menggunakan deret byte sebagai representasi data, seperti yang kita lakukan dengan DO-RNN, tetapi Autoencoder tidak dirancang untuk bekerja dengan data sekuensial dengan panjang bervariasi. Karena nilai sebuah byte bervariasi dari 0 hingga 255, frekuensi byte adalah jumlah kemunculan setiap nilai byte yang mungkin dalam pesan lapisan aplikasi dibagi dengan panjang pesan. Pembagian diperlukan agar frekuensi satu byte relatif terhadap panjang pesan. Oleh karena itu, model tidak akan bias terhadap pesan yang pendek maupun yang panjang. Mirip dengan DO-RNN, DO-AE menganalisis koneksi yang sah dalam fase pelatihan.

Kami definisikan m_p untuk menunjukkan pesan lapisan aplikasi yang sah yang dikirimkan dengan protokol *p*. Kami kemudian mendefinisikan $X = \{x_i \mid 0 \leq i \leq 255\}$ sebagai satu set frekuensi byte m_p. Vektor frekuensi byte *X* adalah input ke Autoencoder. Model Autoencoder pada dasarnya adalah fungsi non-linear G_p yang mengumpulkan vektor input *X* dari pesan protokol *p* dan mengubahnya menjadi $\hat{X}$. Fungsi tersebut dapat didefinisikan sebagai berikut:

$$\hat{X} = G_p(X)$$

(8)

, di mana $\hat{X} = \{\hat{x}_i \mid 0 \leq i \leq 255\}$. Fungsi G_p kemudian dioptimalkan dengan backpropagation sehingga $\hat{X}$ sedekat mungkin dengan X, dan mirip dengan model DO-RNN.

Pada fase pelatihan, DO-AE membaca pesan lapisan aplikasi yang sah dan mengelompokkannya berdasarkan protokol yang digunakan. Untuk setiap protokol, kami melatih sebuah model autoencoder untuk merekonstruksi frekuensi byte dari pesan lapisan aplikasi.

Seperti halnya dengan DO-RNN, diperlukan untuk mengukur jarak antara model normal dan koneksi masuk yang baru. Skor anomali untuk setiap protokol di DO-AE ${a_p}^{ae}$ diperoleh dari kuadrat rata-rata selisih antara masing-masing elemen di X dan $\hat{X}$. Seperti disebutkan di Bagian sebelumnya, ini juga dapat disebut sebagai *reconstruction error*, tetapi mulai saat ini kami akan menyebutnya skor anomali untuk singkatnya. Kami mendefinisikan skor anomali DO-AE sebagai berikut:

$$a^{ae} = \frac{1}{256}\sum_{i=0}^{255}(x_i - \hat{x}_i)^2$$

(9)

Pada fase deteksi, DO-AE menganalisis pesan lapisan aplikasi, mengambil frekuensi byte, memasukkannya ke model autoencoder, dan menghitung skor anomali. Setiap pesan lapisan aplikasi dengan skor anomali lebih tinggi dari ambang batas dianggap berbahaya. Ambang batas, mirip dengan DO-RNN, dapat ditentukan secara manual. Namun, kami mengusulkan untuk menggunakan pendekatan statistik untuk menentukan ambang batas sehingga nilainya juga dipelajari dari data pelatihan (lihat Bagian Mendefinisikan Ambang Batas).

HASIL DAN PEMBAHASAN

Bagian ini merinci eksperimen kami dengan DO-RNN dan DO-AE. Kami pertama-tama menjelaskan lingkungan eksperimen kami, termasuk parameter yang digunakan, pekerjaan sebelumnya yang akan dibandingkan, dan metrik yang digunakan untuk mengukur kinerja. Kami kemudian menjelaskan dataset yang digunakan untuk mengevaluasi metode yang kami usulkan. Pendekatan untuk mendefinisikan ambang batas secara statistik juga dibahas nanti di bagian ini. Terakhir, kami akan menunjukkan hasil percobaan yang kami lakukan. Untuk memvalidasi hasil kami, kami juga membandingkannya dengan hasil pekerjaan sebelumnya.

Persiapan Eksperimen

Kami menerapkan metode yang diusulkan menggunakan Python 2.7.12 dengan Keras 2.0.6 (Keras: The Python Deep Learning library), Tensorflow 1.2.1 dengan dukungan GPU, dan pcapy 0.10.8. Semua eksperimen dilakukan pada komputer personal dengan Intel Core i7-6700 @3,40 GHz, RAM 16 GB, dan NVIDIA GeForce GT-730.

Nilai hyperparameters DO-RNN ditunjukkan pada **Tabel *1*** dan DO-AE ditunjukkan pada **Tabel *2***. Ada beberapa parameter lain yang kami eksplorasi untuk melihat pengaruhnya terhadap kinerja, yaitu, panjang subderet *n* untuk DO-RNN dan jumlah neuron di lapisan tersembunyi DO-AE.

Kami menetapkan fungsi aktivasi lapisan Recurrent ke tangen hiperbolik karena lebih cepat konvergen daripada Sigmoid (Anastassiou, 2011)dan bekerja lebih baik di lapisan Recurrent daripada ReLU. Sementara di DO-AE, kami menggunakan ReLU sebagai fungsi aktivasi untuk lapisan tersembunyi karena bukan lapisan Recurrent, dan ini memberikan kecepatan pelatihan yang lebih baik dan efisien pada dataset yang besar daripada Sigmoid (Glorot, Bordes, & Bengio, 2011). Sebagai *optimiser*, kami memilih Adadelta untuk kedua model karena memiliki *overhead* komputasi yang lebih sedikit daripada Stochastic Gradient Descent biasa (Zeiler, 2012). Kemudian, kami menetapkan jumlah *epoch* menjadi sepuluh karena nilai kerugian mulai stabil setelah sepuluh epoch. Untuk mencegah *overfitting*, kami juga menambahkan lapisan Dropout dengan probabilitas 0,2.

Tabel 1 Hyperparameters pada DO-RNN

Hyperparameter	**Nilai**
Dimensi Output Lapisan Recurrent	32
Fungsi Aktivasi pada Lapisan Tersembunyi	Tangen Hiperbolik
Fungsi Aktivasi pada Lapisan Output	Sigmoid
Loss Function	Categorical Crossentropy

Tabel 2 Hyperparameters pada DO-AE

Hyperparameter	**Nilai**
Jumlah Hidden Layer	1, 3, dan 5
Fungsi Aktivasi pada Lapisan Tersembunyi	ReLU
Fungsi Aktivasi pada Lapisan Output	Sigmoid
Loss Function	Binary Crossentropy

Kami membandingkan metode yang kami usulkan dengan PAYL (Wang & Stolfo, Anomalous Payload-Based Network Intrusion Detection, 2004), Kitsune (Mirsky, Doitshman, Elovici, & Shabtai, 2018), Decanter (Bortolameotti, et al., 2017), OCPAD (Swarnkar & Hubballi, 2016) dan (Wang, Liu, Pitsilis, & Zhang, 2018). Secara umum, kami memilih penelitian-penelitian ini untuk dievaluasi karena ketersediaan kode atau program binernya dan karena mereka menggunakan pendekatan *unsupervised*. Kitsune (Mirsky, Doitshman, Elovici, & Shabtai, 2018) dan (Wang, Liu, Pitsilis, & Zhang, 2018) adalah NIDS berbasis *header*, sedangkan PAYL, Decanter, dan OCPAD adalah NIDS berbasis *payload*. NIDS berbasis *header* dan berbasis *payload* disertakan dalam eksperimen sehingga kami juga dapat membandingkan hasil penggunaan fitur dari header paket dan payload jaringan.

Kami mengukur kinerja sistem yang kami usulkan dan penelitian sebelumnya dengan kombinasi *detection rate* dan *false positive rate* karena lebih tahan terhadap *imbalanced data* daripada akurasi. Jumlah koneksi yang sah selalu jauh lebih besar daripada serangan. Metrik akurasi hanya mengukur jumlah prediksi yang benar terlepas dari jenis data aktualnya. Jika kumpulan data evaluasi berisi 99% koneksi yang sah, dengan menebak secara acak semua koneksi masuk sebagai sah, metode ini akan mendapatkan akurasi 99% meski melewatkan semua serangan.

Detection rate (DR) dan *False positive rate* (FPR) dihitung seperti pada persamaan **(*10*)** dan **(*11*)**. *True positive* (TP) adalah jumlah koneksi berbahaya yang terdeteksi. *False-positive* (FP) adalah jumlah koneksi sah yang dianggap berbahaya. *True Negatif* (TN) adalah jumlah koneksi sah yang dianggap sah. *False-negative* (FN) adalah jumlah koneksi berbahaya yang tidak terdeteksi.

$$DR = \frac{TP}{TP + FN} \quad \textbf{(10)}$$

$$FPR = \frac{FP}{TN + FP} \quad \textbf{(11)}$$

Karena kami memiliki dua metrik untuk mengukur kinerja, sangat penting untuk menemukan keseimbangan antara dua metrik ini karena peningkatan DR juga dapat disertai dengan peningkatan FPR. Mencari keseimbangan ini tergantung dari kasus yang dihadapi dan tergantung juga pada akibat dari serangan yang tidak terdeteksi. Ketika serangan memiliki efek yang parah pada sistem, dimana ini merupakan kasus untuk deteksi serangan tingkat rendah, akan lebih baik untuk memprioritaskan DR daripada FPR karena *false positive* tidak berdampak besar (Sommer & Paxson, 2010). Karena jumlah sampel yang sah dan berbahaya sangat tidak seimbang (lihat Bagian Dataset di bawah ini) dan untuk menekankan *false negative* yang lebih rendah, kami menggunakan metrik lain, F2-score, yang diturunkan dari Fβ $-score dengan β sama dengan dua. Metrik ini menemukan keseimbangan antara DR dan FPR dengan memberikan prioritas lebih pada DR. Oleh karena itu, metrik ini membantu kami untuk menyimpulkan mana metode yang lebih baik ketika DR dan FPR sama-sama meningkat. F2-score dihitung seperti pada persamaan **(*12*)**.

$$F2 = \frac{(1 + 2^2) * TP}{(1 + 2^2) * TP + (2^2) * FN + FP\}} \quad \textbf{(12)}$$

Dataset

Dataset koneksi jaringan yang tersedia untuk umum mengalami beberapa masalah. Dataset DARPA99 sudah sangat lama dan tidak mewakili koneksi kontemporer, baik yang sah maupun yang berbahaya. Dataset ISCX12 tidak memiliki informasi tentang jenis serangan apa yang ada. Ini membuat tidak mungkin bagi kami untuk mengumpulkan serangan tingkat rendah dari dataset karena ada berbagai serangan di dalamnya dan berisi beberapa serangan tingkat

tinggi (yaitu, Distributed Denial of Services dan Brute force). Dataset UNSW-NB15 juga tidak sempurna. Namun demikian, ini adalah dataset terbaru yang berisi koneksi sah yang kontemporer dan representatif. Oleh karena itu, kami masih menggunakannya untuk mengevaluasi metode kami.

Dataset UNSW-NB15 terdiri dari trafik jaringan dari dua hari penangkapan, yang pertama diambil pada 22 Januari 2015 (UNSW-JAN) dan lalu bagian kedua dikumpulkan pada 17 Februari 2015 (UNSW-FEB). Perbedaan antara dua hari itu ada pada jumlah koneksi karena yang terakhir memiliki kira-kira sepuluh kali lipat lebih banyak. Keduanya berisi sepuluh kelas, yaitu Normal, Analisis, *Backdoors, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode*, dan *Worms*. Untuk mengevaluasi metode deteksi serangan tingkat rendah, kami hanya menggunakan kelas *Normal, Backdoors, Exploits*, dan *Worms* dalam eksperimen.

Kami juga menggunakan dataset BlattaSploit, yang berisi berbagai serangan tingkat rendah untuk mengevaluasi DO-RNN dan DO-AE. Karena datanya berbeda dari yang ada di UNSW-NB15, kami juga dapat melihat bagaimana performa model kami di hadapan data yang belum pernah terlihat sebelumnya di fase pelatihan.

Dalam percobaan kami, data pelatihan diperoleh dari koneksi yang sah di UNSW-JAN. Set pengujian terdiri dari koneksi yang sah di UNSW-FEB, koneksi HTTP dan SMTP berbahaya dari UNSW-JAN dan UNSW-FEB, dan koneksi berbahaya di BlattaSploit. Kami hanya menguji semua metode pada HTTP, FTP, dan SMTP, karena mereka adalah protokol dengan proporsi koneksi tertinggi dalam dataset UNSW-NB15.

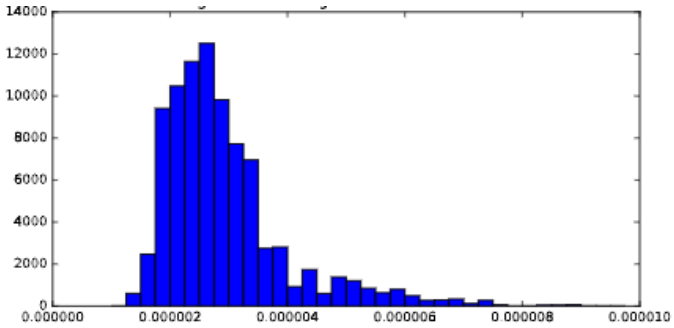
Dataset UNSW-NB15 memiliki 17.041 HTTP, 1.232 FTP, dan 4.631 serangan SMTP. BlattaSploit berisi 5515 HTTP, 9 FTP, dan 74 serangan SMTP. Kami akan menganalisis false positive (koneksi yang sah diklasifikasikan sebagai berbahaya) dengan koneksi yang sah di UNSW-FEB karena koneksi dari UNSW-JAN telah digunakan untuk melatih model. Ada 153.718 koneksi HTTP, FTP, dan SMTP yang sah di UNSW-FEB.

Menentukan Ambang Batas

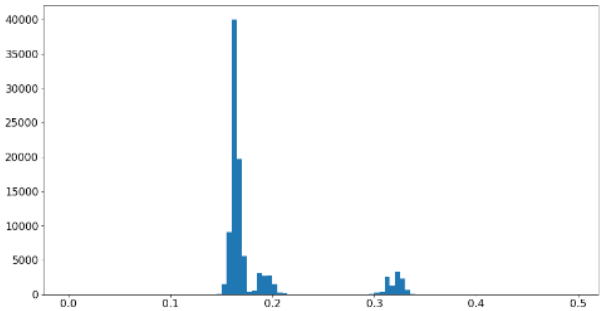
Metode yang diusulkan dilatih pada data pelatihan (yaitu, koneksi-koneksi di UNSW-JAN yang sah). Setelah pelatihan selesai, sampel-sampel yang sama dimasukkan kembali ke dalam model untuk mendapatkan skor anomali yang digunakan untuk menghitung ambang batas. Nilai ambang batas dihitung berdasarkan kumpulan skor anomali ini. DO-RNN dan DO-AE memiliki nilai ambang batas tersendiri namun cara perhitungannya sama.

Ide dasarnya adalah bahwa skor anomali serangan tingkat rendah akan jauh dari skor anomali rata-rata koneksi yang sah di data pelatihan. Serangan tingkat rendah seharusnya menunjukkan deret byte atau frekuensi yang berbeda dan belum pernah terlihat oleh model. Oleh karena itu, DO-RNN akan kesulitan memprediksi elemen berikutnya dalam deret dan DO-AE akan kesulitan merekonstruksi frekuensi byte serangan tingkat rendah. Pertanyaannya adalah, bagaimana kita mendapatkan skor anomali rata-rata? Metode statistik tradisional menggunakan mean (μ) dan standar deviasi (σ). Pendekatan ini bekerja dengan asumsi bahwa

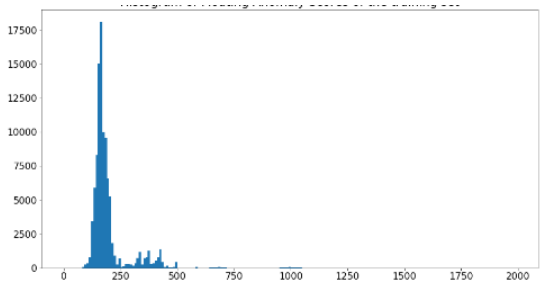
outlier biasanya berada di luar kisaran ini: $\mu - 2 * \sigma > a_p > \mu + 2 * \sigma$ atau $\mu - 3 * \sigma > v > \mu + 3 * \sigma$. Namun, pendekatan ini membutuhkan datanya memiliki distribusi normal. Oleh karena itu, untuk memverifikasi bahwa data kami terdistribusi normal, kami melakukan serangkaian uji statistik.



Gambar 2 Histogram skor anomali yang dihasilkan dengan menjalankan DO-AE



Gambar 3 Histogram skor anomali biner yang dihasilkan dengan menjalankan DO-RNN



Gambar 4 Histogram skor anomali floating yang dihasilkan dengan menjalankan DO-RNN

Seperti disebutkan sebelumnya, DO-RNN dan DO-AE dilatih dengan koneksi UNSW-JAN yang sah, dan data yang sama dimasukkan lagi ke dalam model untuk menghitung skor anomali. Kami kemudian menyelidiki apakah skor anomali yang dihasilkan oleh DO-RNN dan DO-AE terdistribusi normal. Kami pertama-tama memplot histogram dari skor anomali. **Gambar *2***, **Gambar *3***, dan **Gambar *4*** menunjukkan histogram skor anomali yang dihasilkan oleh DO-AE, DO-RNN dengan skor anomaly Biner, dan DO-RNN dengan skor anomali Floating

masing-masing. Gambar-gambar ini menunjukkan pola yang sama, skor anomali sangat *skewed* dan memiliki banyak puncak.

Untuk memastikan bahwa skor anomali set pelatihan tidak terdistribusi normal, kami juga menjalankan uji normalitas (d'Agostino, 1971), (Pearson, D ‘‘'AGOSTINO, & Bowman, 1977). Kami memperoleh p-value nol untuk setiap set skor anomali, yang berarti hipotesis nil ditolak, dan datanya **tidak** terdistribusi normal. Dengan demikian, penggunaan mean dan standar deviasi untuk menemukan ambang tidak cocok untuk mengidentifikasi outlier.

(Diez, Barr, & Cetinkaya-Rundel, 2012) menyarankan rentang interkuartil (IQR) untuk mendeteksi outlier. Seharusnya lebih tahan terhadap outlier karena bergantung pada median (m). Namun, pendekatan ini masih tidak cocok untuk data yang *skewed*. Oleh karena itu, kami mengusulkan untuk menggunakan pendekatan IQR yang dimodifikasi untuk data *skewed* yang diusulkan oleh (Hubert & Vandervieren, 2008). Ini menggunakan medcouple (*MC*) untuk mengukur kemiringan data.

Misalkan *F* menjadi datanya, *MC(F)* adalah pasangan-pasangan dari *F* yang diurutkan di mana $\$x_1 < x_2 < x_3 < \ldots < x_n\$$. *MC(F)* seperti yang didefinisikan dalam persamaan **(*13*)**. x_i dan x_j diambil sampelnya secara terpisah.

$$MC(F) = median_{x_i < m < x_j} h(x_i, x_j) \quad \textbf{(13)}$$

$$h(x_i, x_j) = \frac{(x_j - m) - (m - x_i)}{x_j - x_i} \quad \textbf{(14)}$$

Setelah itu, misalkan Q_3 menjadi kuartil ketiga dari data, kemudian T_{IQR}, ambang batas metode IQR yang dimodifikasi didefinisikan seperti dalam persamaan **(*15*)** dan **(*16*)**.

$$T_{\{IQR\}} = Q_3 + e^{\{3*MC\}} * 1{,}5 * IQR\,, jika\ MC \geq 0 \quad \textbf{(15)}$$

$$T_{\{IQR\}} = Q_3 + e^{\{4*MC\}} * 1{,}5 * IQR\,, jika\ MC < 0 \quad \textbf{(16)}$$

Metode lain untuk mendeteksi outlier adalah dengan menggunakan Z-score yang dimodifikasi, yang bekerja berdasarkan deviasi absolut median (Iglewicz & Hoaglin, 1993). Pengusulnya berpendapat bahwa Median Absolute Deviation dan Median adalah metrik yang mampu menunjukkan tendensi sentral dan dispersi. Namun, cara penerapannya dalam bab ini berbeda dengan pendekatan $T_{\{IQR\}}$.

Pertama-tama kita perlu menghitung Z-score dari skor anomali. Misal $E = \{e_i \mid i < length(F)\}$ adalah sekumpulan nilai anomali yang diperoleh dari data latih. Median Absolute Deviation (*MAD*) dapat dihitung seperti pada persamaan **(*17*)**. Jadi Z-score *z* dari E_i dihitung seperti dalam persamaan **(*18*)** (Iglewicz & Hoaglin, 1993). Saat menggunakan pendekatan ini, sebuah pesan dianggap berbahaya jika *z* lebih besar dari 3,5 (Iglewicz & Hoaglin, 1993).

$$MAD = median(\{(|e_i - median(E)|), e_i \in E, 0 < i < n\}) \quad \textbf{(17)}$$

$$z = \frac{0.6745 * (|e - median(R)|)}{MAD} \quad \textbf{(18)}$$

Diskusi

Secara umum, DO-RNN dengan skor anomali biner dan metode threshold $T_{\{IQR\}}$ memiliki performa terbaik di antara semua metode, seperti terlihat pada **Tabel *3***. Ini mencapai tingkat deteksi yang lebih tinggi dari 99% untuk dataset UNSW-NB15 dan BlattaSploit sambil mempertahankan FPR yang relatif rendah (3,57%). Model ini mencapai F2-score tertinggi sebesar 0,95. DO-AE dengan metode ambang batas Z-score, DO-RNN dengan skor anomali *floating* dan metode ambang batas Z-score, dan One-class-SVM (Wang, Liu, Pitsilis, & Zhang, 2018) mungkin memiliki *detection rate* 100% untuk kedua set data. Namun, FPR mereka juga sangat tinggi, membuat mereka tidak layak untuk diterapkan dalam situasi kehidupan nyata. Metode lain hanya bekerja dengan baik dengan salah satu dataset saja (yaitu, Decanter, dan K-Nearest Neighbour (Wang, Liu, Pitsilis, & Zhang, 2018) atau menunjukkan kinerja yang tidak memuaskan (<50% tingkat deteksi) di kedua dataset (yaitu, OCPAD, DO-RNN yang lain, dan Kitsune).

Tabel 3 Hasil ujicoba

Metode	DR-UNSW (%)	DR-BS (%)	FPR (%)	F2
AE-OD (T_IQR)	51.55	96.83	0.89	0.67
AE-OD (Z-Score)	100	100	23.61	0.74
RNN-OD (Skor anomaly biner & T_IQR)	**99.13**	**99.97**	**3.57**	**0.95**
RNN-OD (Skor anomaly biner & Z-score)	0	0	0	0
RNN-OD (Skor anomaly *floating* & T_IQR)	34.24	58.07	1.12	0.41
RNN-OD (Skor anomaly *floating* & Z-score)	100	100	99.98	0.38
OCPAD (1-gram)	19.88	16.65	0	0.12
OCPAD (3-gram, HTTP)	29.08	23.31	8.85	0.23
PAYL	87.09	83.93	0.05	0.86
One-Class SVM (Wang, Liu, Pitsilis, & Zhang, 2018)	100	100	46.83	0.52
KNN (Wang, Liu, Pitsilis, & Zhang, 2018)	36.41	100	0.03	0.41
Kitsune	0	0	0.0004	0
Decanter	68.13	7.62	0.02	0.15

Selama percobaan dengan DO-RNN dengan skor anomali biner dan metode ambang batas $T_{\{IQR\}}$, kami menemukan bahwa DO-RNN memiliki performa terbaik dalam menganalisis koneksi SMTP dengan tingkat deteksi hingga 100% dan FPR 1\%. Itu karena deret byte dalam pesan SMTP di dataset kami terdistribusi lebih seragam daripada protokol lain. Oleh karena itu ketika ada kode

eksploit, deretnya sangat tidak biasa dan mudah dideteksi oleh metode tersebut. Di protokol lain, metode ini memiliki performa terburuk dalam menganalisis serangan FTP di BlattaSploit dengan DR 77,78%. Kami menduga bahwa hasil ini disebabkan oleh pesan FTP dalam data pelatihan yang relatif lebih pendek daripada pesan HTTP atau SMTP. Oleh karena itu, model memiliki lebih sedikit sampel untuk pelatihan. Adapun HTTP, metode ini memiliki FPR tertinggi di antara protokol lain di data pengujian.

Untuk menganalisis lebih lanjut mengapa beberapa metode kami menghasilkan kinerja yang lebih rendah, kami memilah hasil berdasarkan protokol (yaitu, HTTP, FTP, dan SMTP) dan jenis serangan (yaitu, *Backdoors*, *Exploits*, *Worms*). DO-AE dengan ambang batas $T_{\{IQR\}}$ tampaknya kesulitan mendeteksi serangan tingkat rendah di dataset UNSW-NB15. Ketika kami melihat hasilnya secara lebih rinci, terungkap bahwa DO-AE dengan ambang batas $T_{\{IQR\}}$ mengalami kesulitan dalam mendeteksi eksploitasi berbasis HTTP, hanya 30,5% dari jenis serangan ini yang teridentifikasi. Namun, metode ini mencapai DR 80% untuk jenis serangan lain di HTTP. Ia bahkan mendeteksi 100% serangan tingkat rendah pada FTP dan SMTP. DO-RNN dengan skor anomali *floating* dan ambang $T_{\{IQR\}}$ mengalami masalah serupa. Dalam analisis kami lebih lanjut, tingkat deteksinya yang rendah disebabkan oleh hilangnya banyak eksploitasi berbasis HTTP.

Seperti yang disebutkan sebelumnya, model one-class SVM (Wang, Liu, Pitsilis, & Zhang, 2018) dengan fitur berbasis *header* memberikan DR yang lebih tinggi daripada metode berkinerja terbaik kami, tetapi ia hadir dengan 46,83% FPR, yang tidak dapat diterima di kondisi riil. Model berbasis KNN (Wang, Liu, Pitsilis, & Zhang, 2018) menangkap 100% koneksi berbahaya di dataset BlattaSploit, meskipun performanya buruk pada dataset UNSW-NB15. Metode ini dapat dianggap sebagai metode berkinerja terbaik karena kemampuannya untuk mengidentifikasi lebih banyak koneksi berbahaya di BlattaSploit, kumpulan data dengan serangan tingkat rendah yang lebih representatif dengan *false-positive* yang lebih rendah, tetapi perlu dicatat bahwa BlattaSploit dihasilkan di lingkungan yang berbeda dari UNSW-NB15. Mereka memiliki topologi jaringan yang berbeda dan, dengan demikian, memiliki nilai fitur berbasis header yang sangat berbeda, seperti jumlah hop dan waktu paket antar kedatangan. Oleh karena itu, metode ini mungkin menganggap semua pesan di BlattaSploit sebagai berbahaya hanya karena perbedaan itu. Ini seperti melatih metode dengan data dari satu perusahaan dan mengevaluasinya dengan data berbahaya dari perusahaan lain. Performa model akan tinggi, tetapi menyesatkan. Pada akhirnya, F2-score menunjukkan bahwa DO-RNN mengungguli model berbasis KNN (Wang, Liu, Pitsilis, & Zhang, 2018). Ini menunjukkan bahwa DO-RNN bekerja lebih baik dalam mengenali serangan terlepas dari kumpulan data yang digunakan untuk mengevaluasi.

NIDS berbasis *header* lainnya, Kitsune (Mirsky, Doitshman, Elovici, & Shabtai, 2018), gagal mendeteksi serangan tingkat rendah sama sekali. Ketika kami menganalisis skor anomali dari data pelatihan yang dihasilkan oleh Kitsune, kami menemukan bahwa skor anomali koneksi yang sah memiliki rentang yang sangat besar. Jadi semua skor anomali serangan tingkat rendah berada di bawah ‘rata-rata’. Meskipun menggunakan gabungan beberapa Autoencoders, Kitsune dengan fitur berbasis header melewatkan semua serangan tingkat rendah di kumpulan data kami.

Karena pendekatan berbasis *payload* kami, khususnya DO-AE yang juga menggunakan Autoencoders, memberikan kinerja yang lebih baik, temuan ini mendukung argumen kami bahwa fitur berbasis *header* mungkin tidak cocok untuk menangkap perilaku serangan tingkat rendah.

Model dengan performa terbaik kami (DO-RNN) juga memberikan peningkatan dibandingkan NIDS berbasis *payload* lainnya. Antara lain, OCPAD tampil paling buruk. Perfoma terbaik OCPAD hanya mendeteksi kurang dari 30% serangan tingkat rendah. Bahkan kinerjanya lebih buruk daripada PAYL, pendekatan sebelumnya. PAYL dapat menunjukkan hasil yang layak dengan DR yang tinggi dan FPR yang rendah. F2-score-nya adalah 0,86, tertinggi kedua setelah model terbaik kami. Namun, analisis lebih lanjut menunjukkan bahwa itu hanya dapat mendeteksi 42,93% serangan tingkat rendah berbasis HTTP. PAYL kesulitan untuk mendeteksi Exploit dalam koneksi HTTP. Kami juga menunjukkan bahwa Decanter (Bortolameotti, et al., 2017) mengalami kesulitan mengidentifikasi serangan tingkat rendah HTTP di dataset UNSW-NB15, apalagi serangan di dataset BlattaSploit. Selain itu, ini hanya berfungsi pada HTTP.

KESIMPULAN

Dalam paper ini, kami telah menyajikan dua metode (yaitu, DO-RNN dan DO-AE) untuk mendeteksi serangan tingkat rendah seperti *exploit, backdoor, dan worm*. Keduanya memiliki pendekatan yang sedikit berbeda untuk mendeteksi serangan ini. Untuk mendeteksi apakah pesan lapisan aplikasi berbahaya, DO-RNN mengambil deret byte dari pesan lapisan aplikasi dan memprediksi byte berikutnya untuk setiap deret. Pesan lapisan aplikasi berbahaya akan menyebabkan DO-RNN membuat prediksi yang salah. Ketika jumlah prediksi yang salah melampaui nilai ambang batas yang dihitung dalam fase pelatihan, maka akan ditandai sebagai berbahaya. Di sisi lain, DO-AE menghitung frekuensi byte dari pesan lapisan aplikasi. Frekuensi byte kemudian dimasukkan ke dalam Autoencoders di mana output akan dibandingkan dengan frekuensi byte dari input. DO-AE menganggap pesan lapisan aplikasi berbahaya ketika perbedaan antara input dan outputnya melampaui ambang batas yang telah dihitung sebelumnya dalam fase pelatihan.

Eksperimen kami menunjukkan bahwa model berkinerja terbaik, Long Short-Term Memory, dikombinasikan dengan skor anomali biner dan pendekatan ambang batas statistik, mampu mengidentifikasi serangan tingkat rendah di berbagai protokol lapisan aplikasi (yaitu, HTTP, FTP, dan SMTP). Dalam hal *detection rate*, DO-RNN yang diusulkan melampaui semua penelitian sebelumnya dengan peningkatan F2-score minimal 0,09 dan DR sebesar 12,04%. Oleh karena itu, tidak ada penelitian sebelumnya yang dapat menandingi kinerja DO-RNN.

SARAN

Juga telah ditunjukkan bahwa NIDS berbasis *header* kesulitan dalam mendeteksi serangan tingkat rendah. (Wang, Liu, Pitsilis, & Zhang, 2018), yang telah dievaluasi dengan dataset KDD99 dan menunjukkan hasil yang baik, sekarang dievaluasi dengan dataset UNSW-NB15 yang lebih baru. Eksperimen kami menunjukkan bahwa model mengalami FPR yang tinggi atau DR yang rendah.

Argumen ini juga didukung oleh kegagalan Kitsune untuk mendeteksi serangan tingkat rendah.

DAFTAR RUJUKAN

Anastassiou, G. A. (2011). Multivariate hyperbolic tangent neural network approximation. *Computers & Mathematics with Applications, 61*, 809–821.

Ariu, D.;Tronci, R.;& Giacinto, G. (2011). HMMPayl: An intrusion detection system based on Hidden Markov Models. *computers & security*, *30*, 221–241.

Bolzoni, D.;Etalle, S.;& Hartel, P. (2006). POSEIDON: a 2-tier anomaly-based network intrusion detection system. *Fourth IEEE International Workshop on Information Assurance (IWIA'06)*, (ss. 10–pp).

Bortolameotti, R.;van Ede, T.;Caselli, M.;Everts, M. H.;Hartel, P.;Hofstede, R.;. . . Peter, A. (2017). Decanter: Detection of anomalous outbound http traffic by passive application fingerprinting. *Proceedings of the 33rd Annual Computer Security Applications Conference*, (ss. 373–386).

Carrasco, R. S.;& Sicilia, M.-A. (2018). Unsupervised intrusion detection through skip-gram models of network behavior. *Computers & Security*, *78*, 187–197.

Chiba, Z.;Abghour, N.;Moussaid, K.;El Omri, A.;& Rida, M. (2018). A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection. *Computers & Security*, *75*, 36–58.

Chitrakar, R.;& Huang, C. (2014). Selection of candidate support vectors in incremental SVM for network intrusion detection. *computers & security*, *45*, 231–241.

Chung, J.;Gulcehre, C.;Cho, K.;& Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS 2014 Workshop on Deep Learning, December 2014*.

CVE-2014-6271. (2014). *CVE-2014-6271*. Noudettu osoitteesta https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271

CVE-2019-1619. (2019). *CVE-2019-1619*. Noudettu osoitteesta https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-1619

d'Agostino, R. B. (1971). An omnibus test of normality for moderate and large size samples. *Biometrika, 58*, 341–348.

Davis, J. J.;& Clark, A. J. (2011). Data preprocessing for anomaly based network intrusion detection: A review. *Computers & Security*, *30*, 353–375.

Diez, D. M.;Barr, C. D.;& Cetinkaya-Rundel, M. (2012). *OpenIntro statistics*. CreateSpace.

Elkhadir, Z.;& Mohammed, B. (2019). A cyber network attack detection based on GM Median Nearest Neighbors LDA. *Computers & Security*.

Feng, C.;Li, T.;& Chana, D. (2017). Multi-level anomaly detection in industrial control systems via package signatures and lstm networks. *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, (ss. 261–272).

Glorot, X.;Bordes, A.;& Bengio, Y. (2011). Deep sparse rectifier neural networks. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, (ss. 315–323).

Goodfellow, I.;Bengio, Y.;& Courville, A. (2016). *Deep Learning.* MIT Press.

Hadžiosmanović, D.;Simionato, L.;Bolzoni, D.;Zambon, E.;& Etalle, S. (2012). N-gram against the machine: On the feasibility of the n-gram network analysis for binary protocols. *International Workshop on Recent Advances in Intrusion Detection*, (ss. 354–373).

Hamed, T.;Dara, R.;& Kremer, S. C. (2018). Network intrusion detection system based on recursive feature addition and bigram technique. *Computers & Security*, *73*, 137–155.

Hao, Y.;Sheng, Y.;& Wang, J. (2019). Variant gated recurrent units with encoders to preprocess packets for payload-aware intrusion detection. *IEEE Access*, *7*, 49985–49998.

Hawkins, S.;He, H.;Williams, G.;& Baxter, R. (2002). Outlier detection using replicator neural networks. *International Conference on Data Warehousing and Knowledge Discovery*, (ss. 170–180).

Hochreiter, S.;& Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*, 1735–1780.

Hubert, M.;& Vandervieren, E. (2008). An adjusted boxplot for skewed distributions. *Computational statistics & data analysis*, *52*, 5186–5201.

Iglewicz, B.;& Hoaglin, D. C. (1993). *How to detect and handle outliers* (Osa/vuosik. 16). Asq Press.

Keras: The Python Deep Learning library. (ei pvm). *Keras: The Python Deep Learning library*. Noudettu osoitteesta https://keras.io/

Khammassi, C.;& Krichen, S. (2017). A GA-LR wrapper approach for feature selection in network intrusion detection. *computers & security*, *70*, 255–277.

LeCun, Y.;Bengio, Y.;& Hinton, G. (2015). Deep learning. *nature*, *521*, 436–444.

Liu, H.;Lang, B.;Liu, M.;& Yan, H. (2019). CNN and RNN based payload classification methods for attack detection. *Knowledge-Based Systems*, *163*, 332–341.

Malhotra, P.;Vig, L.;Shroff, G.;& Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series. *Proceedings*, *89*.

Mirsky, Y.;Doitshman, T.;Elovici, Y.;& Shabtai, A. (2018). Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. *machine learning*, *5*, 2.

Mitrecnd. (July 2014). PyNIDS. *PyNIDS*. Noudettu osoitteesta https://github.com/MITRECND/pynids

Moustafa, N.;& Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *Military Communications and Information Systems Conference (MilCIS), 2015*, (ss. 1–6).

Oza, A.;Ross, K.;Low, R. M.;& Stamp, M. (2014). HTTP attack detection using n-gram analysis. *Computers & Security*, *45*, 242–254.

Pearson, E. S.;D ‘'AGOSTINO, R. B.;& Bowman, K. O. (1977). Tests for departure from normality: Comparison of powers. *Biometrika, 64*, 231–246.

Perdisci, R.;Ariu, D.;Fogla, P.;Giacinto, G.;& Lee, W. (2009). McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks, 53*, 864–881.

Qin, Z.-Q.;Ma, X.-K.;& Wang, Y.-J. (2018). Attentional Payload Anomaly Detector for Web Applications. *International Conference on Neural Information Processing*, (ss. 588–599).

Rieck, K.;& Laskov, P. (2007). Language models for detection of unknown attacks in network traffic. *Journal in Computer Virology, 2*, 243–256.

Shen, M.;Wei, M.;Zhu, L.;& Wang, M. (2017). Classification of encrypted traffic with second-order markov chains and application attribute bigrams. *IEEE Transactions on Information Forensics and Security, 12*, 1830–1843.

Sommer, R.;& Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *2010 IEEE symposium on security and privacy*, (ss. 305–316).

Swarnkar, M.;& Hubballi, N. (2016). OCPAD: One class Naive Bayes classifier for payload based anomaly detection. *Expert Systems with Applications, 64*, 330–339.

Transmission Control Protocol. (1981). *Transmission Control Protocol*. Noudettu osoitteesta https://tools.ietf.org/html/rfc793

Wang, K.;& Stolfo, S. J. (2004). Anomalous Payload-Based Network Intrusion Detection. Teoksessa E. Jonsson;A. Valdes;& M. Almgren (Toim.), *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15 - 17, 2004. Proceedings* (ss. 203–222). Berlin: Springer Berlin Heidelberg. doi:10.1007/978-3-540-30143-1_11

Wang, K.;Parekh, J. J.;& Stolfo, S. J. (2006). Anagram: A content anomaly detector resistant to mimicry attack. *International Workshop on Recent Advances in Intrusion Detection*, (ss. 226–248).

Wang, W.;Liu, J.;Pitsilis, G.;& Zhang, X. (2018). Abstracting massive data for lightweight intrusion detection in computer networks. *Information Sciences, 433*, 417–430.

Whalen, S.;Boggs, N.;& Stolfo, S. J. (2014). Model aggregation for distributed content anomaly detection. *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, (ss. 61–71).

Wressnegger, C.;Schwenk, G.;Arp, D.;& Rieck, K. (2013). A close look on n-grams in intrusion detection: anomaly detection vs. classification. *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, (ss. 67–76).

Xiang, J.;Westerlund, M.;Sovilj, D.;& Pulkkis, G. (2014). Using extreme learning machine for intrusion detection in a big data environment. *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, (ss. 73–82).

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.