DETEKSI INTRUSI JARINGAN DENGAN *K-MEANS CLUSTERING* PADA AKSES LOG DENGAN TEKNIK PENGOLAHAN *BIG DATA*

Farid Ridho[1], Arya Aji Kusuma[2]

Program Studi Komputasi Statistika
Politeknik Statistika STIS
e-mail:, [1]faridr@stis.ac.id, [2]aryaaku999@gmail.com

Abstrak

Keamanan jaringan, adalah salah satu aspek penting dalam terciptanya proses komunikasi data yang baik dan aman. Namun, masih adanya serangan yang efektif membuktikan bahwa sistem keamanan yang berlaku belum cukup efektif untuk mencegah dan mendeteksi serangan. Salah satu metode yang dapat digunakan untuk mendeteksi serangan ini adalah dengan dengan *Intrusion Detection System* (IDS). Besarnya data (*volume*), cepatnya perubahan data (*velocity*), serta variasi data (*variety*) merupakan ciri-ciri dari *Big data*. Akses log, secara teori termasuk dalam kategori ini sehingga dapat dilakukan pemrosesan menggunakan teknologi bigdata dengan *Hadoop*. Hal ini mendorong penulis untuk dapat menerapkan metode pengolahan baru yang dapat mengatasi perkembangan data tersebut, yaitu *Big data*. Penelitian ini dilakukan dengan menganalisis akses log dengan *K-Means Clustering* menggunakan metode pengolahan bigdata. Penelitian menghasilkan satu model yang dapat digunakan untuk mendeteksi sebuah serangan dengan probabilitas deteksi sebesar 99.68%. Serta dari hasil perbandingan kedua metode pengolahan bigdata menggunakan pyspark dan metode tradisional menggunakan *python* standar, metode bigdata memiliki perbedaan yang signifikan dalam waktu yang dibutuhkan dalam eksekusi program.

Kata kunci*:* IDS, *big data*, akses log, *k-means*, *clustering*

Abstract

Good network security planning ensures the safety and comfort of user data. However, the existence of effective attacks proves that the current security system is not effective to prevent and detect attacks. One of methods that can be used to detect this attack is by using Intrusion Detection System (IDS). The amount of data (volume), speed of which data change (velocity), and variations in data (variety) are characteristics of big data. Log access, theoretically is also a form of big data so a new approach in statistical data processing is needed to overcome big data. This research was conducted by analyzing log access with K-Means Clustering using the big data processing technique. The study produced a model that can be used to detect an attack with a detection probability of 99.68%. As well as a comparison between big data using Pyspark and traditional processing technique using standard python, which big data technique has a significant difference in time needed to execute the program.

Keywords: *IDS, big data, log access, k-means, clustering*

PENDAHULUAN

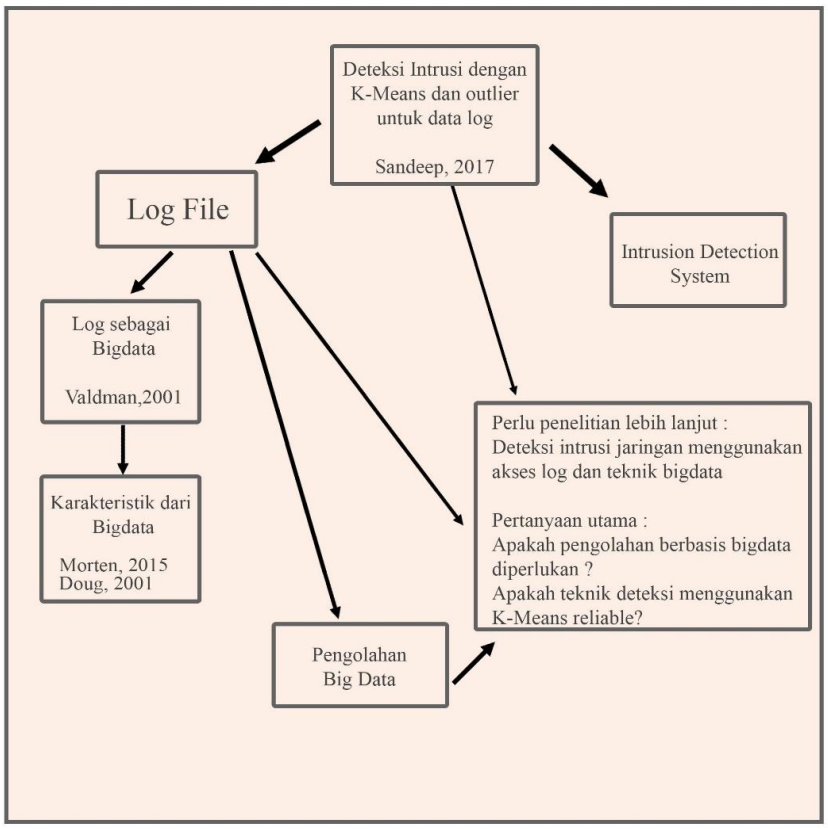
Komunikasi antar komputer merupakan hal sehari – hari yang sering ditemui saat ini. Komunikasi ini dapat terjadi karena adanya sebuah hubungan dalam jaringan komputer. Peranan jaringan komputer tidak hanya terletak pada komunikasi antar komputer, akan tetapi juga terletak pada pertukaran data yang terjadi, baik untuk masyarakat, industri maupun pemerintah. Pada implementasinya, perlu diperhatikan tiga aspek utama, yaitu *performance*, *reliability*, dan *security* agar kegiatan komunikasi dan pertukaran data dapat berjalan secara cepat, aman dan nyaman baik bagi seluruh kalangan.

Keamanan merupakan aspek utama dalam adanya jaringan komputer yang baik. Dalam penerapan keamanan jaringan, dapat dibagi menjadi empat tahapan yaitu prediksi, cegah, deteksi, dan tanggapan. Empat tahapan ini memberikan proteksi untuk menjaga data baik untuk server, maupun pengguna agar terhindar dari ancaman luar yang dapat merusak maupun merugikan pihak terkait. Namun, masih adanya kasus serangan pada jaringan membuktikan bahwa proteksi yang disediakan oleh sistem keamanan jaringan belum sempurna (Govscirt, 2018).

Sistem Deteksi Intrusi Jaringan merupakan salah satu pendekatan dalam deteksi serangan, metode ini mengklasifikasikan aktivitas jaringan yang sedang terjadi ke dalam model yang telah dibangun sistem ke dalam *library* sehingga dapat dikategorikan sebagai aktifitas normal atau serangan (Chakraborty, 2017). Pada praktiknya, sistem intrusi jaringan dibagi menjadi dua yaitu berbasis anomali dan berbasis aturan. Pada Sistem Deteksi Intrusi jaringan berbasis aturan, aktivitas atau lalu lintas akan dicek dengan membandingkan data tersebut dengan aturan yang telah dicatat sebelumnya menggunakan pola yang sering ditemukan pada aktivitas serangan. Sistem Akan tetapi, metode ini memiliki kekurangan yaitu metode ini tidak dapat mendeteksi serangan tipe baru yang belum pernah tercatat sebelumnya, yaitu *false-negative*. Ditambah metode ini sering mengklasifikasikan aktifitas atau lalu lintas baru sebagai serangan, dapat disebut sebagai *false-positive* yaitu aktivitas normal yang tercatat sebagai serangan.

Sedangkan Sistem deteksi intrusi jaringan berbasis anomali mengklasifikasikan aktivitas atau trafik ke dalam klasifikasi data normal maupun data anomali. Hal ini dapat dihasilkan dengan cara statistik. Dengan melihat data secara statistik, dapat dihasilkan model yang tepat dapat mendeteksi serangan dan dapat selalu terbaharui. Aktivitas dan lalu lintas dalam komunikasi antar server dan pengguna tercatat dalam data log (Iversen, 2015). Akses log, merupakan catatan data transaksi pengguna dan server yang meliputi URL, HTML, gambar, file, browser yang digunakan dalam kejadian transaksi tersebut. Secara umum, akses log dapat dianalisis secara statistik untuk informasi perihal monitoring jaringan, seperti banyaknya pengunjung, banyaknya jumlah akses data, maupun melihat popularitas halaman setiap harinya (Valdman, 2001). Karena data tercatat secara keseluruhan transaksi, maka dapat dilakukan analisis yang lebih lanjut. Akan tetapi, perlu diketahui bahwa dengan penggunaan akses log maka dapat menimbulkan permasalahan pada saat prosesing data karena file akses log memiliki ukuran yang besar, memiliki perubahan data yang sangat cepat, dan memuat berbagai macam informasi (Grace, 2011).

Besarnya ukuran data, cepatnya perubahan data, serta data yang bervariasi merupakan ciri – ciri dari *Big Data*. Sehingga, secara teori akses log merupakan *Big Data* maka dapat dilakukan metode pengolahan dengan *Big Data* untuk mengatasi permasalahan tersebut. Pengolahan dengan teknik *Big Data* akan melibatkan *Hadoop*, yaitu aplikasi berbasis open-source yang dapat mengatur dan memproses data secara terdistribusi. Dengan arsitektur *Hadoop*, juga akan digunakan *Spark* yaitu aplikasi pengolahan data secara terdistribusi (Scott, 2015).



Gambar 1. Peta Literatur

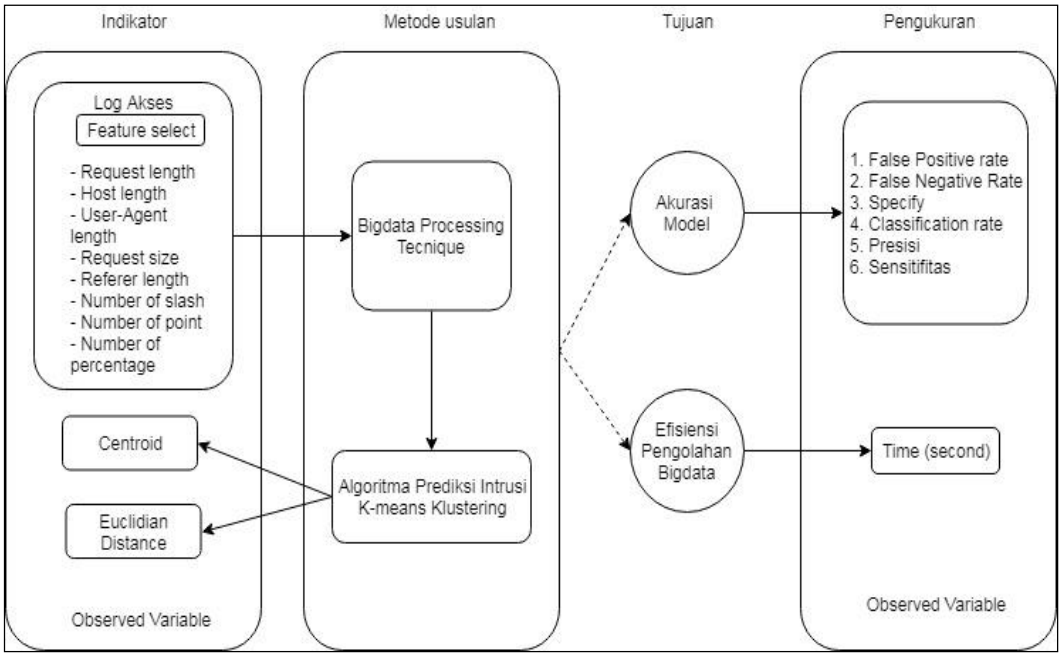
Algoritma analisis yang digunakan adalah *K-Means Clustering*. Dimana data akan di klaster kan menjadi 2 dan lalu akan diklasifikasikan sebagai cluster normal dan cluster anomali. Dari cluster bentukan itu akan dibangun sebagai model yang nantinya dapat digunakan untuk menganalisis aktivitas atau trafik baru yang masuk.

METODOLOGI

Logfile adalah catatan atas seluruh kejadian atau transaksi yang terjadi pada suatu sistem. Seiring perkembangan jaman, transaksi antar pengguna terjadi dalam hitungan menit bahkan detik. Sehingga untuk memenuhi kebutuhan atas pengolahan dan prosesing data, diperlukan teknik yang tepat yaitu teknik *Big Data*. Dalam penelitian ini akan digunakan akses log dari website www.stis.ac.id dalam periode waktu 27 Februari 2017 hingga 31 Oktober 2017. Akses log akan diproses menggunakan *Big Data* dengan *K-Means Clustering* sebagai metode analisisnya (Chandel, 2017).

1. Peta literatur bahasan

Peta literatur diatas menjelaskan adanya kesinambungan atas beberapa topik yang telah dijelaskan sebelumnya. Untuk mengatasi permasalahan deteksi intrusi jaringan dapat menggunakan analisis akses log, dimana akses log adalah catatan seluruh transaksi. Karena besarnya ukuran, cepatnya perubahan serta data yang memiliki variasi, akses log dapat dikategorikan sebagai bigdata. Maka dalam proses pengolahannya dapat digunakan teknik pengolahan *Big Data*. Pengolahan *Big Data* dilakukan dengan tujuan memeriksa keseluruhan data untuk menemukan pola didalamnya (Mukherjee dkk, 2016). Untuk melakukan proses komputasi secara cepat untuk data besar tersebut, tidak dapat menggunakan sistem konvensional seperti RDBMS maupun sistem penyimpanan lainnya, karena dapat memberikan beban yang terlalu besar pada komputer pengolah. Untuk menanggulangi hal ini dapat digunakan *Hadoop* sebagai arsitektur pengolahan *Big Data* (Parthiban, 2016).



Gambar 2. Kerangka Pikir Penelitian

2. Kerangka Pikir Penelitian

Dari penjelasan diatas, setiap tahapan dan langkah penelitian dapat digambarkan pada kerangka pikir (Gambar 2).

Dari kerangka pikir di atas, dapat dilihat bahwa dari log akses dilakukan clustering dengan fitur yang terpilih untuk mendapatkan pusat cluster untuk dibentuk model yang nantinya akan di evaluasi untuk melihat ukuran dari model tersebut (Fink dkk, 2012). Serta dalam proses pengolahan menggunakan bigdata, akan diuji perbedaannya dengan metode tradisional.

3. Metode Analisis

Dalam penelitian ini dilakukan serangkaian tahapan guna mencapai hasil analisis yang representatif. Analisis yang digunakan adalah dengan metode K-Means *Clustering*. K-Means *Clustering* dilakukan pada variabel akses log yang dipilih. Berikut tahapan yang dilewati dalam pengerjaan penelitian ini :

Data *preprocessing*

Data akses log berekstensi log akan di parsing terlebih dahulu menggunakan bahasa pemrograman *python*, hal ini dilakukan agar memudahkan sistem untuk mengolah data tersebut. Parsing data dilakukan dengan bantuan regex, yaitu regular expression memanfaatkan pola yang terdapat pada data akses log sendiri. Proses data *preprocessing* yang dilakukan adalah sebagai berikut:

1. Data cleaning*:* Pembersihan data dengan mengisi missing value, dan mengatasi inkonsistensi data yang ada pada log.
2. Data integrasi*:* Menghilangkan konflik dan menggabungkan data dengan tipe berbeda
3. Data selection*:* Memilih data sesuai dengan yang dibutuhkan, dalam istilah lain sering disebut dengan istilah feature selection
4. Data transformation*:* Data di normalisasi, agregasi, dan generalisasi

Ekstraksi dan Pemilihan Fitur

Analisis K-Means dilakukan pada beberapa variabel pilihan. Pemilihan variabel tersebut dilakukan dengan tahapan Ekstraksi dan Pemilihan Fitur. Ektraksi fitur untuk melakukan transformasi data menjadi fitur yang sesuai dengan model. Dalam penelitian terkait tentang ekstraksi fitur pada data log server web, didapatkan 30 fitur yang dapat diambil dari sebuah file log untuk deteksi serangan (Nguyen, dkk, 2011).

Feature Name	Feature Name
Length of the request ⋆ ◇	Length of the path ⋆
Length of the arguments ⋆ ◇	Length of the header “Accept” †
Length of the header “Accept-Encoding” †	Length of the header “Accept-Charset” †
Length of the header “Accept-Language” †	Length of the header “Cookie” †
Length of the header “Content-Length” †	Length of the header “Content-Type”
Length of the Host †	Length of the header “Referer” †
Length of the header “User-Agent” †	Method identifier
Number of arguments ⋆	Number of letters in the arguments ⋆
Number of digits in the arguments ⋆	Number of ’special’ char in the arguments ⋆ † • ◇
Number of other char in the arguments • ◇	Number of letters char in the path ⋆
Number of digits in the path ⋆ †	Number of ’special’ char in the path ⋆
Number of other char in path †	Number of cookies †
Minimum byte value in the request ◇	Maximum byte value in the request ⋆ †
Number of distinct bytes	Entropy ◇
Number of keywords in the path	Number of keywords in the arguments

Gambar 3. Fitur yang Dianggap Relevan pada Deteksi Serangan

Tabel 1. Fitur Spesial

No	Serangan	Contoh	Karakter Spesial
(1)	(2)	(3)	(4)
1	*Directory Transversal*	/admin/./index.html	*Slash* (/), *dot* (.)
2	*Hex-Encode HTTP Evasion*	/%69%6E%64%65	Persentase (%)
3	*Regex Attack*	^(([a-z])+.)+$	Semua simbol

Keterangan simbol untuk Gambar 3 adalah sebagai berikut: ⋆ fitur yang dipilih oleh CFS dari dataset CSIC-2010, † fitur yang dipilih oleh mRMR dari dataset CSIC 2010; • fitur yang dipilih oleh CFS dari dataset ECML/PKDD 2007 ; ◊ fitur yang dipilih oleh mRMR dari dataset ECML/PKDD 2007. Untuk fitur dalam akses log serta melihat faktor yang sering muncul pada serangan pada penelitian terkait, dipilih 6 variabel, yaitu: (a)Panjang karakter pada request, (b)Panjang karakter pada host, (c)Panjang karakter pada User-agent, (d)Besaran ukuran byte dalam transaksi, (e) panjang karakter pada Referer, serta (f) Banyaknya karakter spesial pada request. Beberapa karakter spesial merupakan ciri – ciri adanya serangan tertentu. Dengan detail pada karakter spesial mengikuti banyaknya okurensi yang mewakili serangan tersebut seperti yang ditampilkan pada Tabel 1.

Dari beberapa karakter spesial, dipilih karakter garis miring (/), titik(.), serta symbol persen (%). Karakter spesial tersebut dipilih karena dianggap dapat mendeteksi anomali yang ada pada transaksi pengguna dan *server*. Karakter ini di ekstraksi dari variabel request yang merupakan halaman yang dituju oleh pengguna terkait.

Setelah melakukan ekstraksi fitur kemudian akan dilakukan proses pemilihan fitur yang dilakukan untuk memilih fitur yang berpengaruh terhadap data dengan tujuan memberikan hasil analisis yang akurat sesuai masalah yang ada. Sehingga, setelah proses pemilihan fitur ini didapatkan 8 fitur utama yang akan digunakan untuk keperluan pengolahan dan analisis tujuan penelitian (Tabel 2).

Dari Tabel 2 terlihat beberapa fitur yang digunakan untuk keperluan analisis. Terdapat 8 variabel yang digunakan untuk keperluan analisis, yaitu besaran request pengguna, panjang karakter dari host, panjang karakter dari request line, panjang karakter dari User-Agent, panjang karakter dari Referer, banyaknya okurensi garis miring, dot, serta persentase. Variabel tersebut diasumsikan telah dapat mewakili beberapa karakteristik dari aktivitas jaringan, baik aktivitas normal maupun anomali.

Tabel 2. Fitur Terpilih sebagai Variabel Penelitian

No	Feature Name
(1)	(2)
1	Panjang dari request line
2	Panjang dari IP Host
3	Panjang header dari User-Agent
4	Besarnya ukuran byte setiap request
5	Panjang header dari Referer
6	Jumlah okurensi garis miring
7	Jumlah okurensi titik
8	Jumlah okurensi persentase

Tabel 3. *Confusion Matrix*

		Predicted value	
True value		P	N
	P	TP	FN
	N	FP	TN

Pengolahan dan Analisis Data

Setelah dilakukannya seleksi fitur untuk K-Means, maka hasil olahan akan siap diolah menggunakan *Hadoop* dan *Spark*. Hal ini dilakukan dengan menjalankan dua aplikasi yang bersangkutan, yaitu *Hadoop* Distributed File System (HDFS) serta *Spark*. HDFS adalah media penyimpanan secara terdistribusi antar komputer sehingga memungkinkan *Hadoop* dan *Spark* melakukan tugas pengolahan data secara terdistribusi. Pada percobaan ini digunakan 2 komputer, yaitu satu sebagai master node dan satunya sebagai slave node. Master node merupakan komputer pengatur jalannya pengolahan terdistribusi, sedangkan slave merupakan komputer pekerja. Dalam pengolahan datanya, digunakan *PySpark*, yaitu *Python* in *Spark*. Merupakan program yang memungkinkan untuk menggunakan bahasa pemrograman *python* dalam *Spark* melalui API yang telah disediakan oleh *Spark*.

Dari serangkaian proses diatas akan menghasilkan hasil akhir berupa sebuah model K-Means dan Bisecting K-Means sebagai pembandingnya pada *PySpark*. Sesuai dengan tujuan penelitian ini, evaluasi model ini dilakukan menggunakan metrik yang sama seperti penelitian – penelitian sebelumnya, yaitu dengan menghitung false-positive rate, false-negative rate, sensitifitas atas model, spesifik dari model, classification rate serta presisi atas hasil dari model. Keenam metrik ini dapat didapatkan dari perhitungan 4 aspek, yaitu *True positive*, *true negative*, *false positive* dan *false negative* seperti pada *confusion matrix* (Tabel 3).

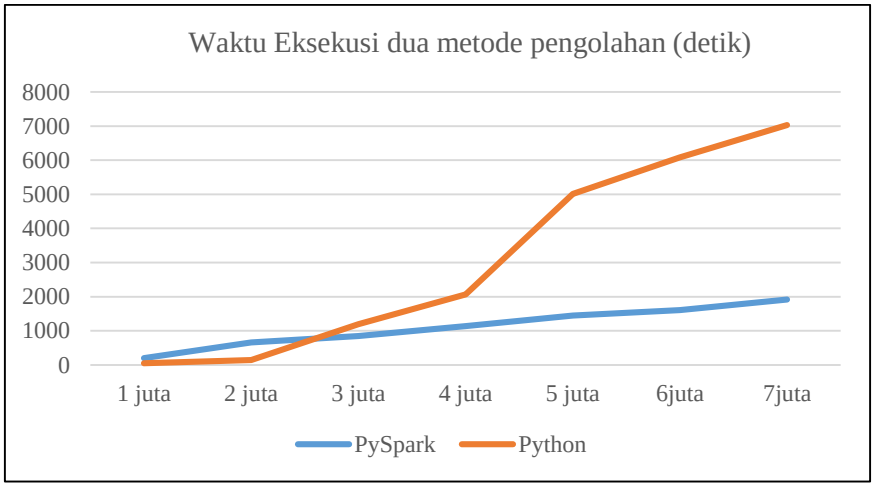
Beberapa nilai di atas dapat menggambarkan kondisi yang sesuai untuk bahasan keamanan jaringan komputer, baik dalam hal ketepatan dan akurasinya.

Spesifikasi yang digunakan

Dalam tahapan pengolahan diatas, digunakan dua perangkat komputer yang terhubung dalam virtualBox. Dimana satu komputer berperan sebagai master node, dan yang lainnya adalah slave node. Master node bertugas untuk mengendalikan dan membagi tugas antar komputer yang terhubung. Sedangkan slave node bertugas hanya sebagai penerima perintah dari master *node*. Kedua komputer tersebut memiliki spesifikasi yang sama, yaitu dengan 2GB RAM dan dengan sistem operasi Ubuntu 16.04. Perbedaannya terletak pada alokasi memori yang diberikan ke ekosistem *Hadoop*, dimana master node memberikan 512MB baik ke executor serta driver memory, sedangkan pada slave node hanya 512MB pada executor memory saja.

Tabel 4. Tabel Waktu Eksekusi Dua Metode Berbeda dalam Detik

Size	1 juta	2 juta	3 juta	4 juta	5 juta	6 juta	7 Juta
Pyspark	201.1	655.7307	841.1226	1140.61	1449.246	1607.348	1920.184
Python	51.78	140.7595	1197.417	2062.988	5011.45	6082.78	7032.3



Gambar 4. Grafik Perbandingan Waktu Eksekusi Dua Metode Pengolahan

Sedangkan piranti lunak yang digunakan dalam pengolahan dan uji coba bahan penelitian adalah sebagai berikut :

1. Bahasa pemrograman*: Python* 2.72
2. *Hadoop* 3.1.0 – *Hadoop Distributed File System*
3. *Spark* 2.3.0, dengan menggunakan *PySpark* (*Spark Python* API)

HASIL DAN PEMBAHASAN

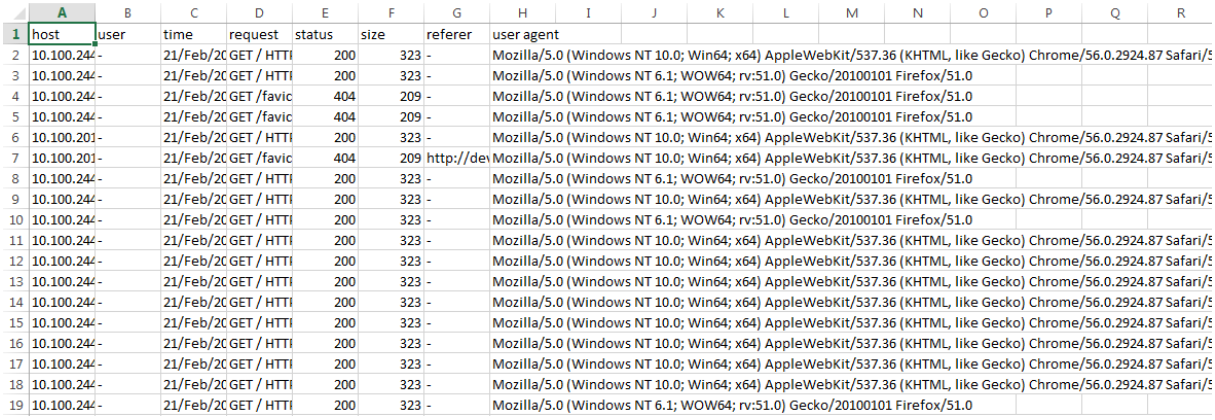
1. Perbandingan antar 2 Metode

Pada bagian sebelumnya telah dijelaskan bahwa dengan penggunaan teknik pengolahan *Big Data* dapat mempersingkat waktu eksekusi yang dibutuhkan untuk tiap data. Pernyataan ini diuji terlebih dahulu menggunakan beberapa data dummy dengan perbedaan pada banyaknya *record* yang ada pada data tersebut. Data yang digunakan mencakup data dengan 1 juta *record*, 2 juta hingga 7 juta *record* dengan masing – masing memiliki 10 variabel. Data tersebut diperlakukan sama seperti data olahan yang nantinya dipakai, yaitu dengan mengolahnya pada *PySpark* dengan menggunakan algoritma K-Means. Hal ini ditujukan untuk mencatat waktu yang dibutuhkan metode tradisional dan metode *Big Data* dalam memproses sebuah data dengan ukuran yang berbeda.

Pengukuran waktu eksekusi dalam percobaan menggunakan bantuan package time yang dimiliki oleh *python*. Time() memberikan nilai waktu terkini dalam detik yang dihitung dari awal masa. Perintah ini dimasukkan saat script dijalankan dan saat script selesai berjalan. Dalam pengukurannya, waktu eksekusi didapat dari selisih antara waktu selesai script berjalan dengan waktu script mulai dijalankan.

Dari Tabel 4 tersebut dapat dilihat bahwa setiap ukuran data akan menghasilkan ukuran waktu eksekusi yang berbeda. Pada metode tradisional, dapat dilihat bahwa pada ukuran data yang kecil proses eksekusi yang dibutuhkan lebih sedikit. Namun untuk kelipatan berikutnya, membutuhkan waktu yang secara drastis naik. Berbeda pada metode bigdata, di mana waktu eksekusi yang dibutuhkan naik secara stabil.

Dari hasil tersebut juga telah dilakukan uji T untuk membuktikan kedua metode adalah signifikan berbeda. Dengan tingkat signifikansi 5% dan df yaitu 6, didapatkan nilai t tabel sebesar 1,943. Dan berdasarkan perhitungan, didapat nilai t-



	A	B	C	D	E	F	G	H
1	host	user	time	request	status	size	referer	user agent
2	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
3	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
4	10.100.24	-	21/Feb/20	GET /favic	404	209	-	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
5	10.100.24	-	21/Feb/20	GET /favic	404	209	-	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
6	10.100.201	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
7	10.100.201	-	21/Feb/20	GET /favic	404	209	http://dev	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
8	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
9	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
10	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
11	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
12	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
13	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
14	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
15	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
16	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
17	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
18	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/5
19	10.100.24	-	21/Feb/20	GET / HTTI	200	323	-	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0

Gambar 5. Hasil Parsing *Raw Data* Akses Log Website Politeknik Statistika STIS

18819	48	14	72	42	3	2	0
225	52	13	72	88	5	2	0
16762	46	15	165	22	3	2	0
18819	48	15	65	46	3	2	0
225	52	15	65	88	5	2	0
20396	46	14	137	48	3	2	0
24212	33	14	65	46	3	2	0
17492	80	11	72	1	11	1	0
24032	40	11	110	18	3	2	0
0	15	10	160	1	2	1	0
17298	14	12	160	1	2	1	0
225	25	15	100	28	2	2	0
20396	46	15	165	22	3	2	0
225	52	14	65	88	5	2	0
0	15	12	160	1	2	1	0
21067	78	14	65	41	3	2	0
17298	14	12	108	1	2	1	0
225	52	12	108	1	5	2	0
24276	40	14	90	80	3	2	0
225	52	14	90	48	5	2	0
18819	48	14	1	48	3	2	0

Gambar 6. Hasil Ekstraksi Fitur

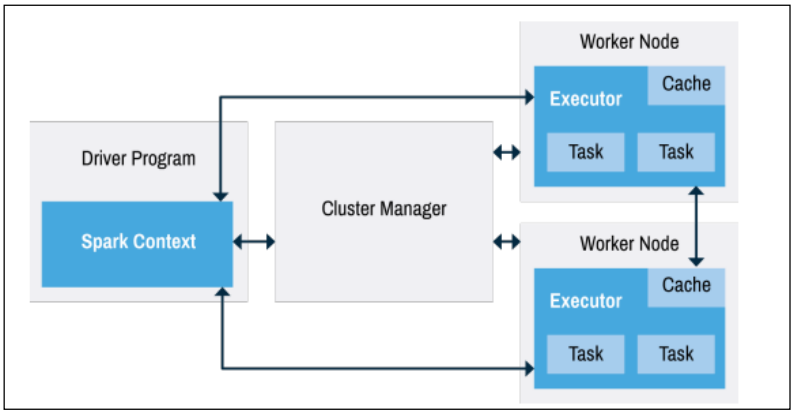
hitung sebesar 2.2165, dari kedua nilai ini dapat dilihat perbedaan kedua metode. Karena t hitung > t tabel, maka H0 ditolak, sehingga dapat diambil kesimpulan bahwa terdapat perbedaan signifikan antara waktu yang dibutuhkan untuk eksekusi script atau program pada metode tradisional menggunakan *python* dengan metode pengolahan bigdata.

2. *Clustering* pada *Big Data*

Preprocessing

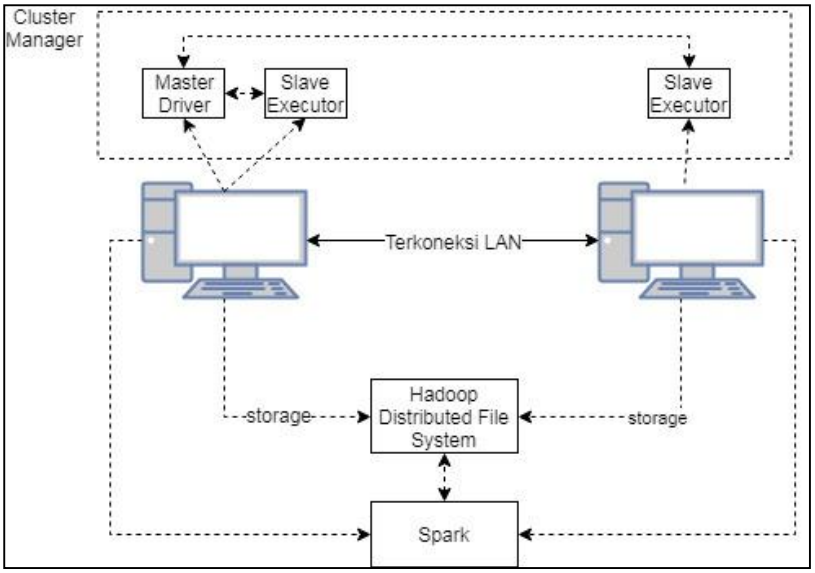
Sesuai yang telah dijelaskan pada sub-bab sebelumnya, tahapan ini terbagi menjadi 4 bagian yaitu data pre-prosesing, seleksi fitur dan ekstraksi fitur, pengolahan data, serta analisis hasil olahan. Pre-prosesing data dilakukan dengan parsing data file berekstensi .log menjadi bentuk yang lebih dapat dibaca yaitu .csv. Hal ini dilakukan dengan menggunakan regex yang telah disediakan pada salah satu script sebelumnya. Script ini akan menghasilkan file .csv baru yang diambil dari parsing data log.

Data hasil pre-prosesing tidak mengalami penambahan atau pengurangan record, di mana hanya terdapat catatan transaksi pada rentang waktu Februari 2017 hingga Oktober 2017. Dari data tersebut dilakukan seleksi dan ekstraksi fitur. Dari fitur yang telah dipilih, dilakukan ekstraksi dengan menghitung panjang dari variabel tertentu, serta menghitung banyak okurensi dari karakter tertentu. Penghapusan beberapa variabel yang tidak digunakan juga dilakukan.



Gambar 7. Alur Kerja Logical *Spark*

Sumber: Getting Started with Apache Spark, hal. 38



Gambar 8. Alur Kerja Fisik *Spark*

Pengolahan data

Setelah data siap olah, data tersebut dimasukkan kedalam ekosistem pengolahan *Hadoop* dengan melalui HDFS. Hal ini memungkinkan data tersebut dapat diolah secara terdistribusi, agar memenuhi tujuan dari penelitian ini. Pengolahan data dilakukan dengan *Spark*, yang dapat dijelaskan dalam Gambar 7 dan Gambar 8.

Pengolahan pada *Spark*, tidak melihat seluruhnya pada jumlah komputer yang digunakan. Namun melihat pada banyaknya node yang digunakan. Komposisi node pada *Spark* meliputi 1 master node dan beberapa slave node. Pada penelitian ini, menggunakan 2 slave node. Pengalokasian tugas pengolahan dilakukan pada master node dengan driver. Alokasi tugas dari driver lalu akan diberikan ke cluster manager, pada penelitian ini *Spark* menjadi cluster manager. Cluster manager lalu akan membagikan tugas dari driver ke slave node untuk dilakukan pengolahan secara terdistribusi. Pembagian tugas ini melewati HDFS, karena data yang akan diolah harus masuk kedalam HDFS untuk dapat dilakukan komputasi terdistribusi. Cluster manager akan memonitoring tugas yang telah diberikan ke slave node, dengan memonitoring secara terus menerus konsistensi dari pengolahan dapat terjaga. Setelah data selesai diolah, maka akan diberikan kembali ke cluster manager untuk disatukan kembali dan hasil tersebut akan diberikan ke driver untuk dapat ditampilkan pada layar atau file *output*.

Hasil pengolahan data

1. K-Means *Clustering*

Hasil dari pengolahan data di atas berupa pusat cluster dan model yang dapat

```
In [2]: from pyspark.mllib.clustering import KMeans, KMeansModel
        from numpy import array
        from math import sqrt

        data = sc.textFile("///user/hadoop/revisi_4.csv")
        parsedData = data.map(lambda line: array([float(x) for x in line.split(',')]))

        clusters = KMeans.train(parsedData, 2)

        for x in clusters.clusterCenters: print(x)

[1.14941476e+04 5.80004820e+01 1.32577988e+01 1.01088818e+02
 2.53420899e+01 4.68543727e+00 1.99703187e+00 1.47047355e-01]
[1.72912425e+06 5.76777824e+01 1.33801584e+01 1.09877345e+02
 3.97445811e+01 5.12390579e+00 2.06638183e+00 1.46540225e+00]
```

Gambar 9. Hasil Running Algoritma *K-Means* dengan *Pyspark*

```
In [1]: from pyspark.mllib.clustering import BisectingKMeans, BisectingKMeansModel
        from numpy import array
        from math import sqrt

        data = sc.textFile("///user/hadoop/revisi_4.csv")
        parsedData = data.map(lambda line: array([float(x) for x in line.split(',')]))

        model = BisectingKMeans.train(parsedData, 2, maxIterations=5)

        for x in model.clusterCenters: print(x)

[7.51238183e+02 6.79834723e+01 1.30996041e+01 8.99985370e+01
 2.14752879e+01 5.60538117e+00 1.88649086e+00 1.38719273e-01]
[3.25604176e+04 4.46917369e+01 1.34691553e+01 1.15906847e+02
 3.05531600e+01 3.46086011e+00 2.14465707e+00 1.63326513e-01]
```

Gambar 10. Hasil Running Algoritma *Bisecting K-Means* dengan *Pyspark*

dipanggil di dalam *PySpark*. Dari 5.700.375 record akses log dengan 8 variabel menghasilkan dua pusat cluster yaitu: [11.494,147 ; 58,0004 ; 13,257 ; 101,08 ; 25,342 ; 4,685 ; 1,997 ; 0,147] serta [1.729.124,25 ; 57,677 ; 13,38 ; 109,8 ; 39,744 ; 5,123 ; 2,066 ; 1,465] (Gambar 9).

2. Bisecting K-Means *Clustering*

Dari hasil running script terlihat bahwa dari 5.700.375 record akses log dengan 8 variabel menghasilkan dua pusat cluster yaitu: [751,238 ; 67,983 ; 13,099 ; 89,99 ; 21,475 ; 5,605 ; 1,8864 ; 0,1387] serta [3,256 ; 4,469 ; 13,469 ; 115,906 ; 30,55 ; 3,46 ; 2,144 ; 0,1633] (Gambar 10).

Evaluasi hasil Cluster terhadap IDS

Dari hasil clustering didapatkan sebuah model yang dapat digunakan untuk mendeteksi suatu data tergolong ke cluster normal atau anomali. Untuk menguji coba keefektifan model tersebut, maka dihitung serangkaian metrik yaitu *false positive rate* (FPR), *false negative rate* (FNR), sensitifitas, presisi, classification rate, dan spesifik (Tabel 5, Tabel 6 dan Tabel 7).

1. Interpretasi FPR

FPR adalah rasio IDS mengklasifikasikan aktivitas normal sebagai aktivitas serangan. Pada algoritma *K-Means Clustering*, didapatkan nilai FPR sebesar 0,0191% sedangkan pada Bisecting *K-Means Clustering* mendapat 53,58%. Semakin besarnya nilai FPR, maka model tersebut dapat dikatakan lebih sensitif dalam mendeteksi aktivitas jaringan. Namun, semakin tinggi nilai FPR juga menggambarkan bahwa akan terlalu banyak aktivitas normal yang terdeteksi sebagai anomali. Maka, dari kedua nilai berikut dapat menggambarkan bahwa algoritma K-Means lebih baik dalam aktivitas.

2. Interpretasi FNR

FNR adalah rasio yang mengklasifikasikan aktifitas serangan sebagai aktifitas normal. Semakin tinggi nilai FNR menunjukkan bahwa model akan lebih rentan terhadap serangan. Pada kedua algoritma, mendapatkan nilai FNR yang sama yaitu sebesar 79,51%. Namun, mengingat nilai FPR *K-Means Clustering* lebih rendah daripada Bisecting *K-Means Clustering* maka algoritma *K-Means*

Tabel 5. *Confusion Matrix* dari *K-Means*

K-Means	***Predicted value***		
True value		P	N
	P	76	295
	N	19	99.609

Tabel 6. *Confusion Matrix* dari *Bisect K-Means*

Bisect	***Predicted value***		
True value		P	N
	P	76	295
	N	53.390	46.238

Tabel 7. Ukuran Evaluasi Kedua Metode

Algoritma	**FPR**	**FNR**	**Sensitifitas**	**Spesifikasi**	**R. Klasifikasi**	**Presisi**
K-Means	0,019%	79,515%	0,205	0,999	99,68%	80%
Bisecting	53,58%	79,515%	0,205	0,464	46,31%	0,14%

Clustering tetap lebih unggul daripada Bisecting K-means.

3. Interpretasi dari Sensitifitas

Yaitu rasio model mengkategorikan aktivitas serangan sebagai serangan atau juga dapat disebut sebagai *True Positive Rate*. Semakin tinggi nilai sensitifitas, maka semakin tinggi FPR yang ada pada model tersebut. Kedua nilai ini memiliki keterkaitan satu sama lain. Terlihat bahwa dari kedua algoritma, bahwa keduanya memiliki nilai sensitifitas yang sama. Namun, K-Means memiliki FPR yang lebih rendah dari pada Bisect K-Means hal ini menggambarkan bahwa algoritma Bisect K-means lebih sensitif dan akan memberikan lebih banyak notifikasi kepada administrator.

4. Interpretasi dari Spesifikasi

Seperti halnya sensitifitas, spesifikasi menggambarkan nilai *True Negative Rate* (TNR) yaitu mengkategorikan aktivitas normal sebagai normal. Dapat dilihat pada table diatas bahwa nilai spesifikasi dari K-Means memiliki perbedaan yang sangat jauh dibanding Bisect *K-Means Clustering*. Hal ini menggambarkan bahwa algoritma *K-Means Clustering* lebih sanggup untuk mengidentifikasi aktifitas normal sebagai normal.

5. Interpretasi dari Rasio Klasifikasi

Rasio klasifikasi menggambarkan seberapa besar akurasi yang dihasilkan dari permodelan data. Dalam kasus deteksi intrusi jaringan, maka nilai rasio klasifikasi yang tinggi akan meningkatkan kualitas dari model tersebut, didorong dengan kecilnya nilai FPR dan FNR dari model tersebut. Sehingga dapat disimpulkan bahwa algoritma *K-Means Clustering* lebih baik dalam memberi gambaran besar atas aktivitas serangan ataupun normal.

6. Interpretasi dari Presisi

Presisi yang dimaksud adalah kemampuan model untuk mendeteksi sebuah aktivitas serangan. Algoritma *K-Means Clustering* memiliki nilai yang lebih besar daripada algoritma Bisect *K-Means Clustering*, sehingga dapat diambil kesimpulan bahwa *K-Means Clustering* lebih mampu atas mendeteksi aktivitas serangan.

Dari hasil evaluasi diatas, dapat terlihat bahwa algoritma *K-Means Clustering* memiliki keunggulan sebagai model pendeteksian intrusi jaringan.

KESIMPULAN DAN SARAN

Dari hasil pengolahan, analisis, hingga evaluasi data di atas maka dapat diambil kesimpulan berdasarkan tujuan dari penelitian ini, yaitu :

1. Implementasi pengolahan bigdata pada Deteksi Intrusi Jaringan dengan memanfaatkan akses log telah dilakukan. Hal ini dapat dilakukan karena akses log memiliki karakteristik dari bigdata, sehingga cocok untuk dilakukannya teknik pengolahan bigdata pada data tersebut, untuk tujuan Deteksi Intrusi Jaringan. Serta diperkuat dengan adanya hasil uji perbedaan antara metode tradisional dengan metode pengolahan bigdata yang menghasilkan bahwa metode pengolahan bigdata lebih baik dalam aspek efisiensi waktu eksekusi.
2. Implementasi *K-Means Clustering* serta Bisect *K-Means Clustering* dalam deteksi intrusi jaringan telah dilakukan dan menghasilkan model yang masing – masing mewakili algoritmanya. Karakteristik yang muncul dari model tersebut berbeda untuk kedua algoritma tersebut, pada model hasil algoritma *K-Means Clustering*, aktivitas anomali memiliki besaran request dari pengguna yang besar, berasal dari referrer dengan jumlah karakter yang lebih banyak, serta memiliki okurensi karakter spesial yang lebih banyak dibanding aktivitas normal. Aktivitas normal pada model *K-Means Clustering* memiliki kecenderungan bahwa pada request line, sedikit ditemukan karakter spesial yaitu persentase, berasal dari referrer yang memiliki jumlah karakter lebih sedikit, serta diakses melalui User-Agent yang hamper sama dengan aktivitas lain. Sedangkan pada algoritma Bisecting *K-Means Clustering*, aktivitas anomali memiliki besaran request dari pengguna yang besar, request pengguna yang lebih panjang, di akses dari User-Agent yang memiliki karakter lebih pendek daripada aktivitas normal, serta memiliki okurensi garing miring lebih banyak dibanding aktivitas normal. Pada aktivitas normal, memiliki perbedaan dibanding dari *K-Means Clustering*, yaitu pada Bisecting *K-Means Clustering* aktivitas normal memiliki okurensi persentase yang lebih banyak dari model *K-Means Clustering*, memiliki besaran request yang sangat kecil, serta panjang karakter request juga yang sedikit.
3. Dari hasil evaluasi yang telah di jabarkan pada Bab Hasil dan Pembahasan, telah terlihat bahwa algoritma *K-Means Clustering* memiliki keunggulan dibandingkan algoritma Bisect *K-Means Clustering*. Keunggulan yang dimaksud adalah dalam perihal kekuatan dan akurasi dari model bentukan dalam mendeteksi aktivitas normal maupun anomali. Hal ini dilihat dari interpretasi nilai FPR *K-Means Clustering* yang lebih kecil dengan nilai sebesar 0,019%, serta Spesifikasi, Rasio Klasifikasi dan Presisi yang lebih baik dibanding dari hasil Bisect *K-Means Clustering* dengan masing – masing nilainya adalah 0,99 untuk spesifikasi , rasio klasifikasi sebesar 99,68% serta 80% presisi model bentukan dari *K-Means Clustering*.

Dari proses dan hasil penelitian yang telah dilakukan pada data akses log dengan menggunakan pengolahan bigdata untuk tujuan deteksi intrusi jaringan, penulis memberikan beberapa poin saran yang mungkin dapat dikembangkan lebih lanjut :

1. Penggunaan metode statistik yang lebih kompleks untuk mendapatkan model yang lebih akurat.
2. Penambahan node dalam ekosistem *Hadoop* dan *Spark*, untuk memaksimalkan kinerja dan efisiensi dari pengolahan *Big Data*.
3. Menggunakan dataset yang lebih besar.
4. Membangun sistem yang dapat memanfaatkan hasil penelitian ini untuk memberi notifikasi ke administrator secara streaming.

DAFTAR PUSTAKA

Chakraborty, N. (2013). *Intrusion Detection System* and Intrusion Prevention System: A Comparative Study. International Journal of Computing and Business Research.

Chandel, S. K. (2017). *Intrusion Detection System* using K-Means Data Mining and Outlier Detection Approach. Bangalore: Faculty of Informatics, Masaryk University.

Fink, G., Chappell, B., Turner, T., & O'Donoghue, K. (2002). A Metrics-Based Approach to *Intrusion Detection System* Evaluation for Distributed Real-Time Systems. Florida: WPDRTS.

Grace, L. J., Maheswari, V., & Nagamalai, D. (2011). Analysis of Web Logs and Web User in Web Mining. International Journal of Network Security and Its Application, 99-110.

GovScirt. (2018). Statistik Insiden Respon Domain .Go.Id. govcsirt.kominfo.go.id. 22 Februari 2018. https://govcsirt.kominfo.go.id/statistik-insiden-respon-domain-go-id/

Iversen, M. A. (2015). When Logs Become *Big Data*. Oslo: Department of Informatics, University of Oslo.

Meyer, R. (2008, January 26). Detecting Attacks on Web Applications from Log Files. SANS Institute Infosec Reading Room, pp. 1-42.

Mukherjee, S., & Shaw, R. (2016). Big Data - Concept, Applications, Challenges, and Future Scope. International Journal of Advanced Research in Computer and Communication Engginering, 66-74.

Nguyen, H. T., Torrano-Gimenez, C., Alvarez, G., Petrovic, S., & Franke, K. (2011). Application of the Generic Feature Selection Measure in Detection of Web Attack. Computatuional Intelligence in Security for Information Systems, 25-32.

Parthiban, P., & Selvakumar, S. (2016). Big Data Architecture for Capturing, Storing, Analyzing and Visualizing of Web Server Logs. Indian Journal of Science and Technology, 1-9.

Scott, J. A. (2015). Getting Started With Apache *Spark*. San Jose: MapR Technologies, Inc.

Seyyar, M. B. (2017). Detection of Attack-Targeted Scans from Apache HTTP Server Access Log. Istanbul: Istanbul SEHIR University.

Suneetha, K., & Krishnamoorthi, R. (2009). Identifying User Behavior by Analyzing Web Server Access Log File. International Journal of Computer Science and Network Security, 327-332.

Troesch, M., & Walsh, I. (2014). Machine Learning for Network Intrusion Detection. Stanford.

Ularu, E. G., Puican, F. C., Apostu, A., & Velicanu, M. (2012). Perspective on Big Data and Big Data Analytics. Database Systems Journal, 3-14.

Valdman, J. (2001). Log File Analysis. Pilsen*:* Department of Computer Science and Engineering, University of West Bohemia.

Vijayalakshmi, S., Mohan, V., & Raja, S. (2010). Mining of Users Access Behaviour for Frequent Sequential Pattern from Web Logs. International Journal of Database Management System (IJDMS), 31-45.

Wei, L. (2007, Oktober 23). Evaluation of *Intrusion Detection System*s. pp. 1-10.

Zhong, S., Khoshgoftaar, T., & Seliya, N. (2007). *Clustering*-based Network Intrusion Detection. International Journal of Reliability, Quality, and Safety Engineering.