JEPIN
(Jurnal Edukasi dan Penelitian Informatika)
ISSN(e): 2548-9364 / ISSN(p) : 2460-0741

Vol. 9
No. 1
April 2023

Deteksi *Malware Ransomware* Berdasarkan Panggilan API dengan Metode Ekstraksi Fitur N-gram dan TF-IDF

Hartinah[#1], Ady Wahyudi Paundu[#2], Amil Ahmad Ilham[#3]

[#]*Departemen Teknik Informatika, Universitas Hasanuddin*
Jl.Poros Malino, Gowa, Sulawesi Selatan, Indonesia

[1]hartinah20d@student.unhas.ac.id

[2]adywp@unhas.ac.id

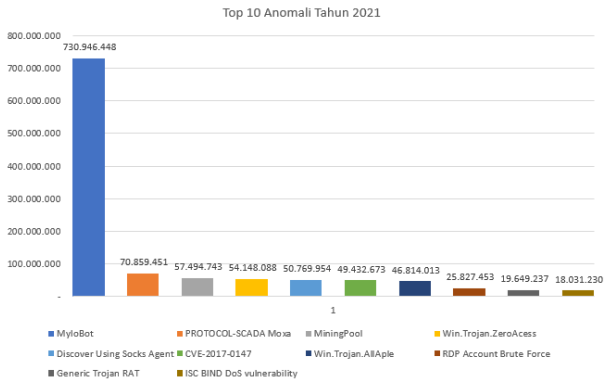
[3]amil@unhas.ac.id

Abstrak*—Ransomware merupakan ancaman malware yang paling menakutkan saat ini karena memiliki kemampuan mengenkripsi data, selain itu jumlah serangan ransomware yang terus meningkat mengakibatkan kerugian yang tidak sedikit. Penanganan atas serangan ini semakin sulit dilakukan dikarenakan varian ransomware yang terus berkembang. Dibutuhkan suatu sistem yang mampu mendeteksi ransomware bahkan untuk varian ransomware terbaru. Melalui penelitian ini kami membuat suatu sistem yang mampu mendeteksi ransomware dan normalware menggunakan metode *machine learning* dengan memanfaatkan data panggilan API dari ransomware dan normalware. Pada penelitian ini kami hanya melakukan *binary classification* untuk semua varian ransomware yang terdeteksi. Proses ekstraksi fitur terlebih dilakukan dengan metode N-gram dan TF-IDF pada panggilan API untuk membentuk subset fitur yang digunakan dalam proses pembelajaran model. Pembuatan model deteksi dilakukan dengan melatih data panggilan API dari beberapa varian ransomware. Pengujian model dilakukan baik terhadap varian ransomware yang sudah dilatih sebelumnya maupun varian ransomware diluar data latih. Proses pembelajaran model dilakukan untuk mencari kesamaan fitur dari data panggilan API berbagai varian ransomware pada data latih, kesamaan fitur ini akan dimanfaatkan untuk mendeteksi varian lain dari ransomware diluar data latih. Hasil penelitian menunjukkan bahwa akurasi rata-rata model terhadap varian ransomware dalam data latih adalah 94% dengan skor *error rate* tertinggi 10%. Adapun hasil deteksi ransomware untuk varian diluar data latih menunjukkan akurasi rata-rata 83% dengan skor *error rate* tertinggi 30%. Sehingga dengan demikian model yang dibuat pada penelitian ini dapat digunakan untuk mendeteksi ransomware meskipun varian dari ransomware mengalami perkembangan.

***Kata kunci*—Ransomware, Panggilan API, *Machine Learning*, *Binary Classification*, N-gram, TF-IDF**

I. PENDAHULUAN

Berdasarkan laporan kasus serangan siber yang dikeluarkan oleh Badan Siber dan Sandi Negara (BSSN), diketahui terdapat 1.637.973.002 serangan anomali yang terjadi di indonesia selama periode Januari hingga Desember tahun 2021, dimana salah satu tren serangan siber ini berupa serangan malware jenis ransomware yang dimasukkan kedalam MyloBot untuk menargetkan sistem operasi windows yang menyebar melalui spam E-mail dan unduhan file yang telah terinfeksi. Dari total 1.637.973.002 serangan siber yang masuk diketahui 730.946.448 merupakan serangan MyloBot berupa Ransomware atau setidaknya sebanyak 44,62% dari serangan yang masuk ke indonesia, dengan statistik serangan anaomali ditunjukkan pada Gambar 1.



Gambar. 1 Top 10 anomali tahun 2021

Ransomware sendiri merupakan jenis *malicious* yang melakukan enkripsi pada data pribadi penggunanya ketika menginfeksi sebuah komputer, malware ini kemudian akan menuntut tebusan dalam bentuk mata uang digital dari korbannya untuk mendapatkan kunci deskripsi pada komputer yang terinfeksi [1] [2], jika dalam jangka waktu tertentu korban tidak membayar tebusan yang telah ditentukan maka *threat actor* akan menyebarkan data pribadi pengguna pada situs Dark Web dengan alamat khusus. Salah satu perusahaan keamanan Emisoft menyatakan ada 369 varian ransomware yang berbeda

selama periode 2021[3], hal inilah yang membuat infeksi ransomware kini semakin meningkat dengan total kerugian yang beragam setiap variannya, contohnya, tahun 2014 dimana varian CryotoLocker mencapai total pembayaran $3 Million dan ditahun 2021 secara umum ransomware mengakibatkan kerugian mencapai $200 Million [4]. Kebanyakan infeksi ransomware ini dialami oleh pengguna sistem operasi windows. Korban ransomware tidak hanya pengguna rumahan atau individu tetapi ransomware telah menargetkan jaringan pemerintah, bisnis, dan layanan kesehatan. Hal ini menyebabkan kerusakan, kerugian finansial dan resiko tersebarnya informasi sensitif yang dapat menyebabkan kerugian lebih besar lagi [5][2].

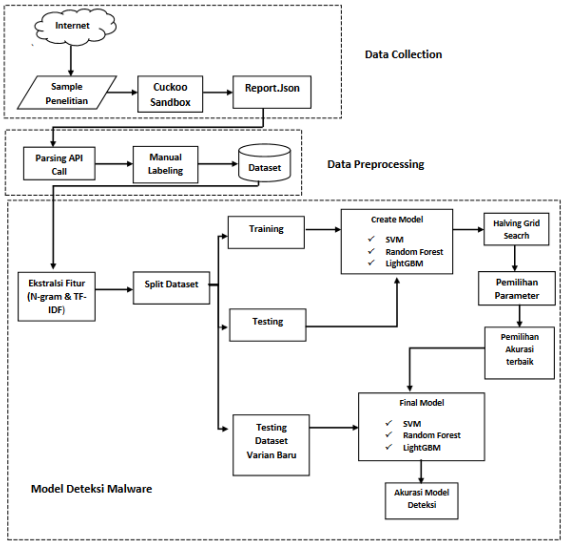
Dalam beberapa tahun penelitian untuk mendeteksi lebih awal infeksi ransomware dengan berbagai metode telah dilakukan. Penelitian [6] memanfaatkan perubahan yang terjadi pada komputer ketika terjadi proses infeksi ransomware seperti perubahan API, registry, string, dan lainnya, kemudian data ini diolah menggunakan markov chain dan algoritma pembelajaran mesin untuk mempelajari setiap perubahan. Hasilnya akurasi deteksi mencapai 97,3 % dengan nilai FPR 4,8%, dan FNR 1,5%. Penelitian [7] memanfaatkan panggilan API ransomware yang dihasilkan dari proses analisis statik malware ransomware, hasil analisis statik panggilan API ini kemudian dimasukkan kedalam beberapa algoritma untuk melalui proses pembelajaran dan pengujian. Hasil dari penelitian ini menempatkan algoritma Random Forest sebagai algoritma yang memiliki tingkat akurasi terbaik yakni mencapai 99%. Penelitian [8][9] menggunakan metode koefisien korelasi dalam menemukan korelasi dari setiap fitur panggilan API varian ransomware. Kemudian mengaplikasikan algoritma pembelajaran mesin yakni SVM dan LightGBM. Hasilnya metode yang diusulkan mampu mendeteksi varian ransomware dengan akurasi rata-rata 98%. Bukan hanya algoritma *machine learning*, penggunaan algoritma *deep learning* pun telah digunakan oleh beberapa peneliti, contohnya penelitian [10][11][12] yang menerapkan algoritma LSTM, ANN, TextCNN, dan DNN dalam proses pembelajaran panggilan API dari berbagai sampel ransomware yang kemudian membawa penelitian-penelitian ini pada akurasi rata-rata sebesar 98%. Penelitian lain yang juga memanfaatkan panggilan API dalam mendeteksi berbagai malware yakni penelitian [13] dengan memanfaatkan fitur analisis statis dan dinamis malware, penelitian ini berhasil mengekstrak urutan panggilan API sebanyak 4.684 sampel malware dari 1.821, dan ditemukan fakta bahwa analisis klaster hierarkis otomatis dapat dengan cepat di proses saat melakukan klasifikasi kelompok varian malware. Penelitian [14] melakukan sebuah terobosan baru dengan mengadopsi algoritma penyelarasan urutan DNA dan mengekstrak pola urutan panggilan API umum dari fungsi berbahaya malware dalam berbagai kategori. Dengan menggunakan panggilan API dan menggunakan metode penyelarasan urutan DNA, penelitian ini menghasilkan akurasi 98% dan diklaim bahkan dapat mendeteksi jenis malware yang belum diketahui. Penelitian [15] memanfaatkan panggilan API malware yang dikombinasikan dengan metode NLP yakni n-gram, doc2vec (atau Vektor paragraf), dan TF-IDF untuk mengubah urutan panggilan API menjadi vektor numerik sebelum dimasukkan kedalam algoritma klasifikasi. Penelitian ini mencoba tiga metode berbeda pada proses ekstraksi fitur panggilan API, yaitu TF-IDF, Paragraph Vector dengan Distributed Bag of Words dan Paragraph Vector dengan Distributed Memor, masing-masing metode memberikan hasil akurasi yang baik. Namun, karena kurangnya sampel, maka peneliti masih belum dapat mengevaluasi kualitas metode yang digunakan untuk membedakan antara malware dan program normal. Selanjutnya penelitian [16] menggunakan urutan panggilan Windows API (Win-API) untuk menangkap perilaku aplikasi berbahaya. Dalam penelitian ini lima kelas malware telah dianalisis yakni Worm, Trojan-Downloader, Trojan-Spy, Trojan-Dropper, dan Backdoor. Sebanyak 400 sampel untuk masing-masing kelas telah digunakan pada eksperimen, dengan 2.000 sampel digunakan sebagai data latih, sedangkan untuk pengujian digunakan 120 sampel untuk setiap kelas. Algoritma hashing fuzzy ssdeep diterapkan untuk menghasilkan *signature* berbasis fuzzy hash. Hasilnya akurasi metode ini diatas 95% dengan tingkat kesalahan yang rendah. Dan terakhir penelitian [17] menerapkan metode data mining dalam mendeteksi malware jenis ransomare dengan memanfaatkan panggilan API dari ransomware. Algoritma Random Forest, SVM, Simple Logistic (SL) dan algoritma Naive Bayes (NB) diterapkan untuk mendeteksi malware ini. Hasilnya Simple Logistic terbukti sebagai algoritma terbaik dalam mendeteksi ransomware. Oleh karena itu, penulis penelitian ini menyimpulkan bahwa sistem yang diusulkan efektif dan kredibel untuk mendeteksi berbagai ransomware.

Dari beberapa penelitian yang telah dipaparkan diatas, diketahui bahwa penelitian ini masih memiliki kekurangan dikarenakan ransomware telah berevolusi dari sisi varian yang semakin banyak serta cara menghindari deteksi dengan meniru perangkat normal [6], sehingga dibutuhkan lebih banyak sampel ransomware yang beragam jenisnya dalam proses pembelajaran, serta dibutuhkan sebuah model deteksi yang mampu mendeteksi ransomware meskipun varian baru muncul, dimana varian ini merupakan varian yang belum pernah dipelajari oleh model deteksi yang dibangun sebelumnya, hal ini dapat membantu dalam menguji apakah model pembelajaran dapat berfungsi baik dalam proses deteksi meskipun terdapat perubahan pada varian yang diuji. Maka dari itu berdasarkan penelitian sebelumnya, kamipun mencoba mengusulkan sebuah model deteksi dengan memanfaatkan panggilan API yang dihasilkan ransomware dan normalware. Panggilan API dipilih sebagai data yang akan diolah pada penelitian ini karena panggilan API memberikan gambaran yang sangat detail dari proses sebuah program biasa maupun malware yang sedang berjalan pada sistem komputer [18] sehingga dengan mempelajari panggilan ini sistem deteksi mampu membedakan program normal dan program berbahaya yang sedang berjalan.

Panggilan API yang dihasilkan oleh ransomware sendiri memiliki ciri yang hampir sama meskipun dengan varian yang berbeda [18], dengan fakta ini maka penelitian yang dilakukan juga akan memamfaatkan panggilan API ransomware. Pada proses ekstraksi fitur kami menggunakan metode NLP yakni N-gram dan TF-IDF dengan asusmsi bahwa panggilan API yang dihasilkan ransomware dan normalware merupakan sebuah kumpulan instruksi berupa "*text*", dimana setiap *text* ini mungkin memiliki nilai variable yang mempengaruhi proses deteksi dari ransomware dan normalware. Nilai yang dihasilkan pada proses N-gram dan TF-IDF kemudian dimasukkan kedalam algoritma pembelajaran mesin yakni SVM, Random Forest, dan LightGBM untuk selanjutnya melalui proses pelatihan dan pengujian data. Pada proses pembelajaran beberapa varian ransomware akan melalui tahap pelatihan dan pengujian data untuk menghasilkan model deteksi yang diharapkan, setelah model terbentuk, maka kami melakukan pengujian pada model untuk mendeteksi varian ransomware lain sebagai ransomware, dengan kata lain penelitian ini hanya melakukan *binary classification* untuk semua varian ransomware yang terdeteksi. Varian ransomware lain ini merupakan jenis ransomware yang belum pernah dipelajari oleh model sebelumnya, hal ini dilakukan guna menguji ketahanan model yang rancang ketika sebuah malware jenis ransomware baru menyerang, apakah model yang dibuat tetap mampu mendeteksi ransomware atau tidak. Selanjutnya peneliti akan menerapkan metode *confusion matrix* untuk melihat performa deteksi dari model yang dibuat.

II. METODE PENELITIAN

Alur penelitian deteksi ransomware dengan metode ekstraksi fitur N-gram dan TF-IDF dijabarkan pada Gambar 2 berikut ini.



Gambar. 2 Gambaran alur penelitian

A. *Data Collection*

Tahapan *Data collection* seperti ditunjukkan oleh Gambar 2 merupakan proses pengumpulan setiap sampel ransomware dan normalware yang akan digunakan pada penelitian ini. Sampel ransomware yang telah dikumpulkan memiliki 16 varian berbeda yang berasal dari tiga situs repositori malware yakni Virus Share, Malware Bazar, dan Any Run dengan jumlah setiap varian yang bervariasi, sedangkan untuk sampel normalware berasal dari penelitian [19]. Total sampel ransomware secara keseluruhan sebanyak 958 dan normalware sebanyak 514, jadi total keseluruhan sampel yang digunakan sebanyak 1.472 sampel, yang kemudian akan dibagi menjadi empat dataset berbeda, dimana sebanyak 12 varian ransomware ditambah dengan 395 sampel normalware akan di jadikan sebagai dataset pembelajaran model (pelatihan dan pengujian), sedangkan sisanya yakni 4 varian ransomware ditambah dengan 119 sampel normalware akan dijadikan data uji, dataset ini diasumsikan sebagai data ransomware terbaru yang tidak diikut sertakan dalam proses pembelajaran model sebelumnya (pelatihan dan pengujian), hal ini bertujuan untuk menguji ketahanan model yang dibuat ketika sebuah malware jenis ransomware baru menyerang. Contoh pembagian dataset dapat dilihat pada Table I.

TABEL I
PEMBAGIAN DATASET

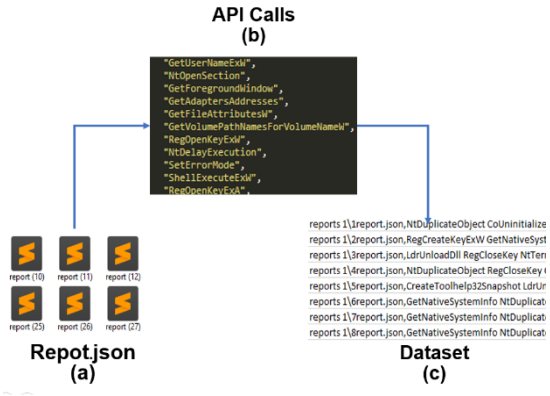
Dataset	Pembelajaran Model	Jumlah sampel	Data Uji Model	Jumlah Sampel
Dataset	Cerber	33	Ryuk	19
	Cryptolocker	13	Satan	31
	Cryptowall	41	Teslacrypto	27
	Kollah	113	Wannacry	80
	Kovter	34	Normalware	119
	Lockbit	78		
	Locky	63		
	Medusa	67		
	Netwalker	124		
	Petya	3		
	Pysa	21		
	Reveton	211		
	Normalware	395		
	Total	**1.196**	**Total**	**276**

Sampel ransomware dan normalware yang telah dikumpulkan kemudian akan dijalankan satu per satu kedalam aplikasi cuckoo sandbox untuk melalui proses analisis aktivitas lebih detail. Hasil analisis dari setiap sampel akan berupa sebuah file report.json. Report.json ini

kemudian akan diolah pada tahapan kedua yakni pada tahapan *Preprocessing.*

B. Data *Preprocessing*

Tahap selanjutnya merupakan proses *preprocessing* data yakni tahap mengelolah file report.json yang dihasilkan dari cuckoo sandbox (*Data Collection*).



Gambar. 3 Proses parsing panggilan API

Pada Gambar 3 terdapat tiga tahapan preprosesing data yang dilakukan yakni :

a. Tahapan pertama pengumpulan seluruh file report.json ransomware dan normalware dari hasil cuckoo sandbox.
b. Setelah seluruh file dikumpulkan, maka selanjutnya adalah memasukkan file report.json kedalam program *parsing* python yang dibuat, hal ini dilakukan sebab report.json berisi laporan menyeluruh dari sampel yang dijalankan sedangkan pada penelitian ini hasil analisis yang dibutuhkan hanyalah panggilan API dari ransomware dan normalware.
c. Setelah panggilan API dari ransomware dan normalware berhasil ditarik, maka selanjutnya program *parsing* akan menyusun secara otomasis panggilan API dari setiap ransomware dan normalware kedalam bentuk file CSV (dataset) agar dapat digunakan dalam pengolahan selanjutnya.

C. Pembuatan Model Deteksi Malware

Tahapan awal dari pembuatan model deteksi yakni proses ekstraksi fitur. Pada dasarnya metode ekstraksi fitur yang digunakan dalam penelitian ini adalah metode yang sama pada proses *text mining*, hal ini berlaku karena panggilan API diasumsikan sebagai sebuah instruksi dalam bentuk "*text*", meskipun pada implementasinya "*text*" yang dimaksud disini tidak mampu diartikan oleh manusia melainkan sebuah instruksi "*text*" yang hanya dipahami oleh sistem operasi komputer. Tujuan dari proses ini sendiri agar panggilan API yang berupa "*text*" akan ditransformasi dalam bentuk angka atau vektor sehingga mesin dapat memahami data yang akan diolah serta menemukan fitur penting pada setiap data yang diproses melalui proses perhitungan kemunculan kata dalam sebuah dokumen. Metode ekstraksi fitur yang diimplementasikan pada penelitian ini adalah N-gram dan TF-IDF.

N-Gram digunakan karena sifat dari panggilan API yang dihasilkan ransomware terjadi dalam jumlah yang banyak dan berulang sehingga diperlukan mekanisme N-gram untuk membagun subset fitur panggilan API yang tidak berulang dan dapat digunakan pada proses selanjutnya tanpa menghilangkan esensi panggilan API yang dihasilkan oleh sampel.

Pembobotan TF-IDF membantu menghitung frekuensi kemunculan setiap panggilan API yang dihasilkan oleh ransomware dan normalware. TF-IDF bekerja dengan dua metode, pertama TF (*Term Frequency*) digunakan untuk menghitung banyaknya kata yang muncul dalam sebuah dokumen, kemudian IDF (*Inverse Document Frequency*) berfungsi menghitung kata yang jarang muncul dalam sebuah dokumen. Maka dengan demikian fitur jarang muncul pada data dapat dianggap sebagai fitur penting dan harus diberi skor tinggi. Tetapi ketika fitur tersebut muncul terlalu sering, maka fitur tersebut tidak terindikasi sebagai fitur unik dan diberikan skor yang lebih rendah. Hanya panggilan API dengan skor tinggi yang dianggap sebagai profil perilaku data [20]. Persamaan yang digunakan dalam penerapan TF-IDF yakni :

1. Menghitung *Term Frequency* (TF)

$$TF(x) = \log(x+1) \quad [21]$$

2. Menghitung *Inverse Document Frequency* (IDF)

$$IDF(x) = \log((1+p)/(1+d)) + 1 \quad [21]$$

3. Menghitung TF-IDF

$$TF\text{-}IDF = TF * IDF \quad [22]$$

Keterangan :

x = vektor frekuensi mentah

p = jumlah trek di himpunan x

d = vektor yang menghitung trek di mana setiap n-gram muncul.

D. Split Dataset

Setelah proses ekstraksi fitur selesai, maka selanjutnya dilakukan pemisahan dataset yakni data latih dan data uji menggunakan *library* sklearn. Saat dilakukan pemisahan data latih (*training*) dan data uji (*testing*) diketahui bahwa *split test* terbaik adalah 80% untuk data latih dan 20% untuk data uji.

Proses selanjutnya adalah pencarian parameter terbaik atau *tunning hyperparameter* untuk setiap algoritma yang digunakan dalam penelitian ini. *Hyperparameter* terbaik untuk masing-masing algoritma didapatkan melalui pencarian parameter menggunakan *library Halving GridSearch. Library* ini berfungsi memilih parameter terbaik dari setiap algoritma yang digunakan untuk menghasilkan model dengan akurasi terbaik. Setelah model terbentuk maka selanjutnya dataset baru yang berisi varian lain dari ransomware dan normalware akan diuji kedalam model yang dibuat sebelumnya guna melihat ketahanan model ketika sebuah serangan malware baru muncul,

apakah model tetap mampu mendeteksi ransomware atau tidak.

Dalam penelitian ini, kami melakukan evaluasi kinerja akurasi model dengan metode *confusion matrix*. Adapun perhitungan yang kami lakukan mengacuh pada TN, TP FP, FN.

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+TN+FN+FP)} X\ 100\% \text{ [23]}$$

$$\text{Recall} = \frac{(TP)}{(TP+FN)} X\ 100\% \text{ [24]}$$

$$\text{Presisi} = \frac{(TP)}{(TP+FP)} X\ 100\% \text{ [25]}$$

Keterangan :

- TN mengacuh pada jumlah data Negatif (Normalware) yang di prediksi benar
- TP mengacuh pada jumlah data Positif (Ransomware) yang di prediksi benar
- FP mengacuh pada jumlah data Negatif (Normalware) yang di prediksi sebagai data Positif (Ransomware)
- FN mengacuh pada jumlah data Positif (Ransomware) yang di prediksi sebagai data Negatif (Normalware)

III. HASIL PENELITIAN

Bagian ini akan membahas mengenai *environment* penelitian yang digunakan dalam melakukan proses ekspreimen meliputi ekstraksi fitur, pembuatan dan pengujian model pada masing-masing algoritma yang telah dibangun.

A. Environment Sistem Penelitian

Penelitian ini dikerjakan pada sebuah *personal computer* dengan spesifikasi utama prosesor core i7 gen 8, Ram 16 GB dan penyimpanan 1TB dengan sistem operasi utama Linux 18.04 untuk proses analisis dinamis pada cuckoo sandbox. Sedangkan untuk proses *preprocessing* kami membuat program *parsing* menggunakan bahasa pemograman python. Terakhir proses ekstraksi fitur hingga pembuatan model menggunakan Jupyter Notebook dengan beberapa *library* yang dimanfaatkan yakni:

1. Pandas merupakan *library* yang paling umum digunakan untuk membaca dataset dalam bentuk file CSV.
2. Sklearn dimanfaatkan cukup banyak pada penelitian ini, mulai dari proses split dataset, fitur ekstraksi, dan pencarian parameter menggunakan *Halving GridSearch*, serta *classification report.*

B. Ekstraksi Fitur

Panggilan API yang diasumsikan sebagai sebuah "*text*" merupakan permasalahan klasifikasi *text* yang cukup rumit karena panggilan API ini tidak mampu dipahami manusia secara tekstual belum lagi panggilan API ini memiliki jumlah yang cukup banyak dan berulang ketika sebuah program melakukan aktivitas didalam komputer, maka dari itu diterapkan teknik N-gram untuk membangun subset fitur panggilan API yang tidak berulang dan dapat digunakan pada proses selanjutnya tanpa menghilangkan esensi panggilan API yang dihasilkan oleh sampel. Berdasarkan penelitian [26] proses pemilihan nilai N-gram harus sesuai dengan data yang diolah. Meskipun pada teorinya semakin besar nilai N-gram yang digunakan maka jumlah data yang berulang akan semakin sedikit, akan tetapi hal ini akan membuat sulitnya data untuk digeneralisasi sehingga bisa terjadi *overfitting*, untuk mencegah hal ini maka pencarian nilai N terbaik harus dilakukan.

Setelah serangkaian percobaan yang dilakukan dalam mencari nilai N terbaik menggunakan *Halving GridSearch*, ditemukan hasil nilai N-gram terbaik berdasarkan dataset yang digunakan adalah 1,2. Nilai N=1,2 merupakan gabungan dari unigram dan bigram, gabungan dari N-gram ini menghasilkan fitur yang lebih beragam. N-gram 1,2 bekerja dengan cara mengambil teks pertama untuk dibentuk menjadi satu fitur (unigram), kemudian teks pertama ini akan digabungkan dengan teks kedua untuk dibentuk menjadi satu fitur (bigram) dan seterusnya. Setelah nilai N-gram didapatkan maka selanjutnya membatasi fitur yang terbentuk, hal ini berguna untuk mencegah fitur yang terbentuk cukup banyak sehingga akan menghasilkan ruang fitur yang lebih jarang (banyaknya nilai fitur bernilai 0), untuk itu *max feature* yang digunakan adalah 800.

Setelah proses N-gram maka selanjutnya TF-IDF akan menghitung frekuensi kemunculan setiap *term* atau yang menjadi fitur pada panggilan API, hal ini berguna untuk melihat fitur mana yang akan menjadi fitur penting dan dapat digunakan pada proses *training* pada model yang dibuat dengan algoritma pembelajaran mesin. Berikut salah satu hasil perhitungan panggilan API dengan menggunakan TF-IDF.

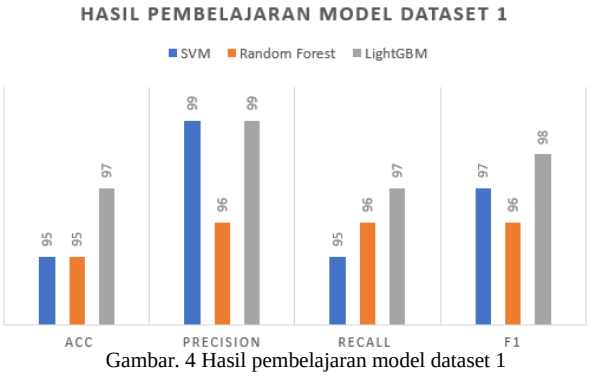
TABEL II
SALAH SATU HASIL EKSTRAKSI FITUR DENGAN TF-IDF

Term	TF	IDF	TF-IDF
cocreateinstance	0.004542	1.715962	0.007794
cocreateinstance getsystemdirectoryw	0.002899	2.190420	0.006350
coinitializeex	0.004124	1.558155	0.006426
coinitializeex createthread	0.004324	3.267292	0.014128
coinitializeex regenumvaluew	0.004231	3.197225	0.013529
coinitializesecurity	0.002818	2.129149	0.006000
couninitialize	0.004292	1.621429	0.006959
couninitialize regclosekey	0.004538	1.714254	0.007778
createactctxw	0.003913	1.478224	0.005784
Createactctx getfilesize	0.003200	2.417693	0.007736

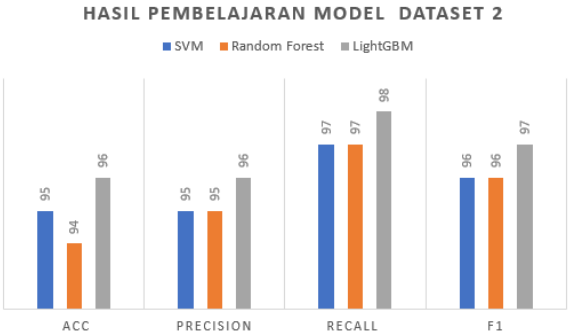
Tabel II diatas menampilkan 10 hasil pembobotan fitur dengan metode TF-IDF yang memanfaatkan *Library* Sklearn pada dokumen pertama dataset 1, ada banyak fitur yang terbentuk dengan bobot yang beragam tergantung kelangkaan dari fitur tersebut.

C. *Hasil Pembuatan Model Untuk masing-masing Algoritma*

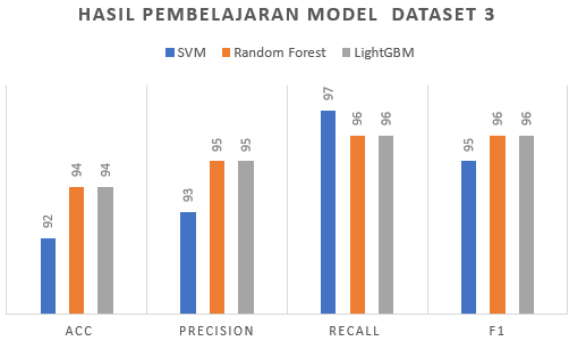
Hasil pembuatan model dievaluasi dengan metode *confusion matrix*, dimana metode ini akan memberikan hasil terkait akurasi, *precision*, *recall*, dan F1 skor untuk masing-masing dataset.



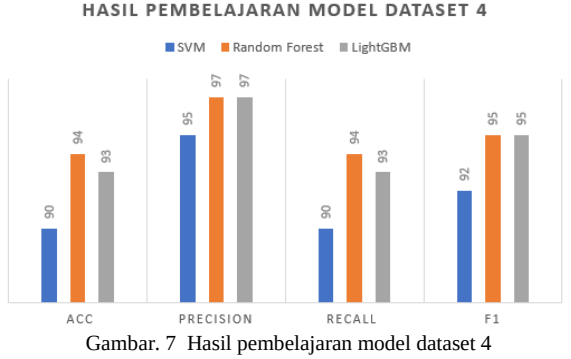
Gambar. 4 Hasil pembelajaran model dataset 1



Gambar. 5 Hasil pembelajaran model dataset 2



Gambar. 6 Hasil pembelajaran model dataset 3

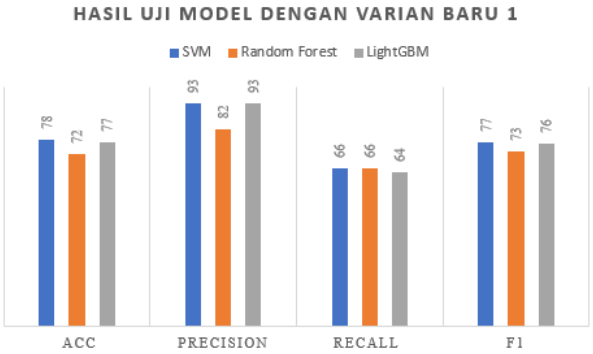


Gambar. 7 Hasil pembelajaran model dataset 4

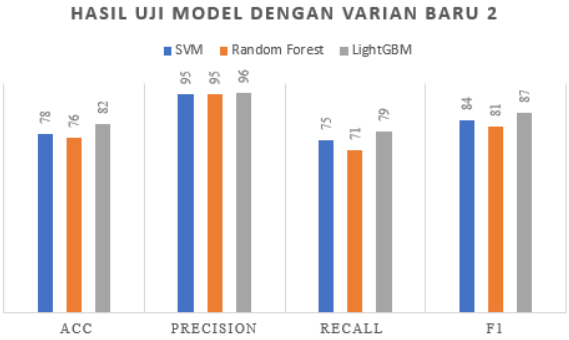
Dari hasil pembelajaran model ketiga algoritma diketahui bahwa akurasi rata-rata yang dihasilkan sebesar 94%, dimana akurasi paling baik dari semua algoritma dimiliki oleh LightGBM.

D. *Hasil Uji Coba Model dengan Data Varian Baru*

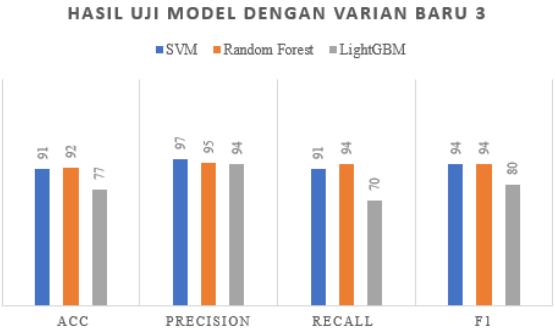
Pada tahap ini, setelah model terbentuk dengan akurasi diatas 90% ditunjukkan pada Gambar 4, 5, 6, dan 7, maka langkah selanjutnya adalah menguji model dengan data varian baru, dimana data ini merupakan data yang belum pernah ditemui oleh model sebelumnya, hal ini dilakukan untuk menguji model apakah mampu mendeteksi ransomaware meskipun ransomware tersebut merupakan sampel baru. Hasil uji coba untuk setiap algoritma pada setiap dataset yakni :



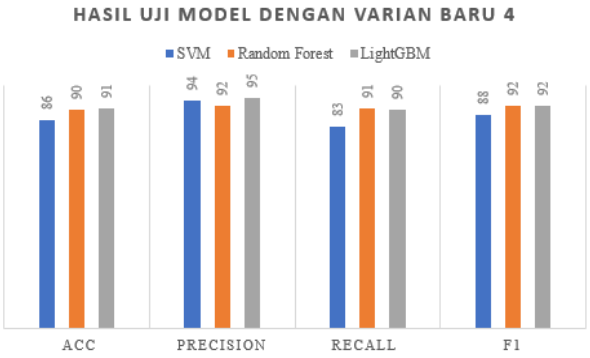
Gambar. 8 Hasil Uji Model dengan Varian Baru 1



Gambar. 9 Hasil Uji Model dengan Varian Baru 2



Gambar. 10 Hasil Uji Model dengan Varian Baru 3



Gambar. 11 Hasil Uji Model dengan Varian Baru 4

Gambar 8, 9, 10 dan 11 menunjukkan hasil pengujian model dengan data ransomware baru, dimana data ini belum pernah dipelajari sebelumnya. Dari hasil ini diketahui bahwa model yang dibuat tetap mampu mendeteksi ransomware baru dengan rata-rata akurasi sebesar 83%.

E. Hasil Error rate untuk Pembelajaran Model dan Pengujian Model Masing-Masing Algoritma

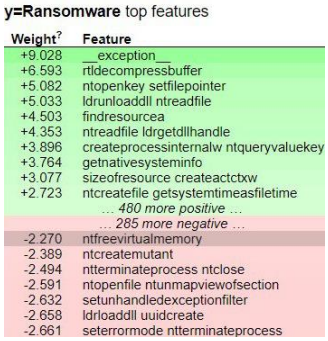
Tabel III menunjukkan tingkat kesalahan deteksi yang dihasilkan oleh model saat proses pembelajaran dan juga proses pengujian dengan data ransomware jenis baru. Pada pembelajaran model nilai *error rate* tertinggi yakni 0,1 atau sebesar 10%, sedangkan pada pengujian model dengan dataset ransomware baru nilai *error rate* tertinggi yakni 0,3 atau sebesar 30%.

TABEL III
HASIL ERROR RATE UNTUK PEMBELAJARAN MODEL DAN PEGUJIAN MODEL DEGAN DATA RANSOMWARE BARU

Algoritma	Dataset 1	Dataset 2	Dataset 3	Dataset 4
SVM Model	0,1	0,1	0,1	0,1
SVM Model Uji	0,2	0,2	0,1	0,1
RF Model	0,1	0,1	0,1	0,1
RF Model Uji	0,3	0,2	0,1	0,1
LightGBM Model	0,03	0,04	0,1	0,1
LightGBM Model Uji	0,2	0,2	0,2	0,1

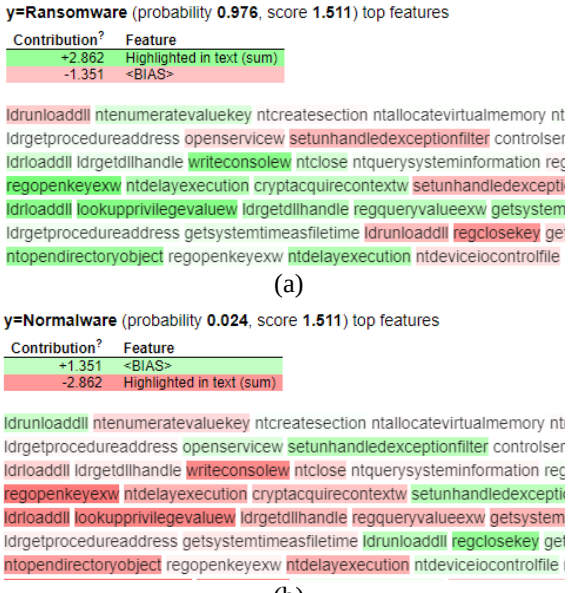
F. Interpretasi Model Deteksi Ransomware

Pada proses ini peneliti mencoba melihat interpretasi dari salah satu algoritma yang digunakan dalam mendeteksi malware yakni algoritma SVM dengan memanfaatkan librari eli5, hal ini bertujuan untuk menganalisis bagaimana algoritma SVM mengenali dataset berdasarkan hasil ekstraksi fitur yang telah dipelajari pada proses *training* sebelumnya. Adapun hasilnya sebagai berikut



Gambar. 12 Top Fitur untuk Ransomware

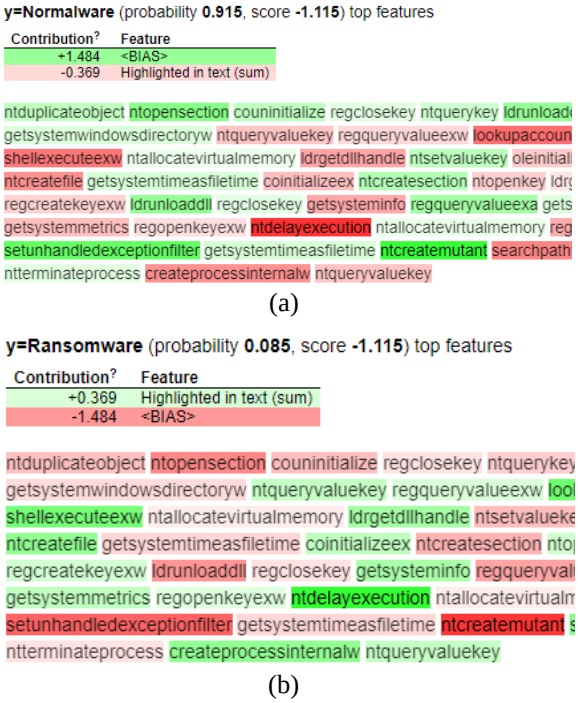
Gambar 12 menunjukkan fitur yang diidentifikasi oleh algoritma SVM selama proses pembelajaran sebagai fitur berpengaruh dalam mendeteksi ransomware. Fitur inilah yang akan menjadi penentu dari proses deteksi ketika dataset ransomware baru yang belum pernah ditemui oleh model dalam proses *training* diuji coba kedalam model. Model akan mengukur seberapa penting fitur yang ada, tidak hanya berdasarkan penilaian kinerja fitur tetapi bagaimana setiap fitur itu sendiri berkontribusi pada proses pengambilan keputusan untuk proses uji coba model pada dataset ransomware baru.



Gambar. 13 Hasil Interpreter algoritma SVM untuk data Ransomware

Gambar 13 bagian (a) menunjukkan bagaimana model melakukan deteksi data berdasarkan fitur dari proses pembelajaran model sebelumnya. Fitur yang berwarna hijau merupakan fitur milik ransomware yang memiliki pengaruh pada proses pembelajaran. Sedangkan untuk fitur berwarna merah merupakan fitur yang tidak memiliki pengaruh terhadap deteksi ransomware atau bisa dikatakan fitur ini merupakan fitur berpengaruh untuk data normalware. Data baru yang diuji pada Gambar 13 merupakan data pertama, yang pada kenyataannya merupakan data API ransomware dan pada proses uji coba, model dengan benar mendeteksi data tersebut sebagai ransomware, adapun Gambar 13 bagian (b) merupakan data

yang sama hanya saja kami mencoba menunjukkan bagaimana jika model diberi instruksi untuk mendeteksi data tersebut sebagai data normalware, maka terlihat dengan jelas bagaimana model mengenali perbedaan fitur antara ransomware dan normalware berdasarkan fitur berpengaruh saat proses pembelajaran dilakukan. Pada Gambar 13 diatas hasil deteksi memiliki probabilitas yang berbeda untuk data yang sama, dimana untuk bagian (a) model mendeteksi data tersebut sebagai ransomware dengan nilai probabilitas 0.976, sedangkan bagian (b) saat model diberi instaruksi mengenali data tersebut sebagai normalware, skor nilai probabilitas sebesar 0.024, sehingga probabilitas data tersebut merupakan ransomware lebih tinggi dibandingkan jika data tersebut adalah normalware.



Gambar. 14 Hasil Interpreter algoritma SVM untuk data Normalware

Contoh lain seperti Gambar 14 (a) yang menunjukkan bagaimana model melakukan pengujian untuk data ke 162 yang sebenarnya merupakan data normalware dan benar model mendeteksi data tersebut sebagai normalware. Adapun Gambar 14 bagian (b) merupakan data yang sama hanya saja kami mencoba menunjukkan bagaimana jika model diberi instruksi untuk mendeteksi data tersebut sebagai data ransomware, maka terlihat bahwa kedua hasil deteksi diatas memiliki probabilitas yang berbeda untuk data yang dideteksi, dimana untuk deteksi data bagian (a) memiliki nilai probabilitas 0,915 sedangkan hasil deteksi bagian (b) memiliki nilai probabilitas 0,085 sehingga data yang diuji pada Gambar 14 merupakan normalware lebih akurat dibandingkan jika data tersebut adalah ransomware.

Pada penelitian ini diketahui ketiga algoritma yang digunakan memiliki beberapa perbedaan dalam menilai fitur penting panggilan API ransomware selama proses pembelajaran dilakukan, lebih jauh kami mencoba mencari kesamaan fitur yang berpengaruh pada proses deteksi ransomware untuk keseluruhan fitur yang ditelah ditemukan pada setiap model sebelumnya, dan hasilnya kami menemukan 26 fitur yang mempengaruhi proses deteksi ransomware pada model yang dibuat menggunakan tiga jenis algoritma klasifikasi dengan memanfaatkan empat pola dataset yang telah dibagi pada penelitian ini. Pada implementasinya terdapat banyak fitur yang digunakan oleh algoritma dalam mengidentifikasi data baru, tetapi pada tulisan ini kami menampilkan 26 fitur yang mempengaruhi proses deteksi ransomware pada model yang dibuat menggunakan tiga jenis algoritma klasifikasi dengan memanfaatkan empat pola dataset yang telah dibagi. Adapun 26 fitur panggilan API ransomware ini ditunjukkan Table IV

TABEL IV
TOP 15 FITUR RANSOMWARE BERDASARKAN KETIGA ALGORITMA

26 Fitur Panggilan API Ransomware Berpengaruh	
__exception__	closesocket
findresourcea	ntcreatefile
ntcreatefile getsystemtimeasfiletime	writeconsolew
ntreadfile	Loadstringw ldrgetprocedureaddress
ntdelayexecution	oleinitialize
ntprotectvirtualmemory shgetfolderpathw	ntallocatevirtualmemory regopenkeyexa
lookupprivilegevaluew	ntresumethread
ntopendirectoryobject	ntenumeratevaluekey
wsastartup	regopenkeyexw
regsetvalueexa	ntqueryinformationfile
createprocessinternalw	ntdelayexecution seterrormode
findfirstfileexw	coinitializeex
getsystemtimeasfiletime findfirstfileexw	ntopenkey

IV. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan dengan memanfaatkan panggilan API ransomware dan normalware dalam proses pembelajaran model, dimana model yang telah berhasil dibuat kemudian melalui proses pengujian baik terhadap varian ransomware yang sudah dilatih maupun untuk varian ransomware diluar data latih, menghasilkan fakta bahwa model yang dihasilkan tetap mampu mengenali ransomware dengan akurasi rata-rata sebesar 83%. Fakta ini telah sejalan dengan hipotesis awal yang kami dibangun oleh yakni ransomware tetap mampu dideteksi meskipun terdapat varian yang belum pernah ditemui atau dipelajari oleh model sebelumnya, hal ini disebabkan karena adanya kesamaan fitur panggilan API pada malware ransomware, kesamaan fitur ini didapatkan pada proses *training* pada ketiga algoritma untuk setiap dataset. Dari proses analisis keseluruhan model dan dataset diketahui terdapat 26 fitur panggilan API yang sama ditunjukkan Table IV. Dari ketiga algoritma yang digunakan pada proses pembelajaran model menunjukkan nilai akurasi rata-rata model terhadap varian ransomware pada data latih (*training*) sebesar 94% dengan skor *error*

rate tertinggi sebesar 10%. Adapun hasil deteksi ransomware dengan varian diluar data latih menunjukkan nilai akurasi rata-rata sebesar 83% dengan skor *error rate* tertinggi sebesar 30%. Sehingga kami yakin bahwa metode yang diusung dapat mendeteksi ransomware meskipun varian dari ransomware terus mengalami perkembangan.

UCAPAN TERIMA KASIH / ACKNOWLEDGMENT

Terima kasih kami ucapkan kepada keluarga, sahabat, dan seluruh dosen serta staf Departemen Magister Teknik Informatika Universitas Hasanuddin Makassar yang telah mendukung langkah kami selama ini, terutama kepada kedua dosen pembimbing kami yang telah mengarahkan dan membantu kami dalam menyelesaikan penelitian ini. Semoga proyek penelitian ini dapat bermanfaat bagi pengembangan ilmu pengetahuan di bidang keamanan jaringan khususnya di bidang analisis forensik malware.

REFERENSI

[1] D. Morato, E. Berrueta, E. Magaña, And M. Izal, "Ransomware Early Detection By The Analysis Of File Sharing Traffic," *J. Netw. Comput. Appl.*, Vol. 124, No. September, Pp. 14–32, 2018, Doi: 10.1016/J.Jnca.2018.09.013.

[2] A. Palisse, H. Le Bouder, J. L. Lanet, C. Le Guernic, And A. Legay, "Ransomware And The Legacy Crypto Api," *Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, Vol. 10158 Lncs, Pp. 11–28, 2017, Doi: 10.1007/978-3-319-54876-0_2.

[3] Emsisoft, "Ransomware Statistics For 2021 Q1 Report," Pp. 1–7, 2021.

[4] M. Humayun, N. Z. Jhanjhi, A. Alsayat, And V. Ponnusamy, "Internet Of Things And Ransomware: Evolution, Mitigation And Prevention," *Egypt. Informatics J.*, Vol. 22, No. 1, Pp. 105–117, 2021, Doi: 10.1016/J.Eij.2020.05.003.

[5] D. Y. Kao, S. C. Hsiao, And R. Tso, "Analyzing Wannacry Ransomware Considering The Weapons And Exploits," *Int. Conf. Adv. Commun. Technol. Icact*, Vol. 2019-Febru, No. 1, Pp. 1098–1107, 2019, Doi: 10.23919/Icact.2019.8702049.

[6] J. Hwang, J. Kim, S. Lee, And K. Kim, "Two-Stage Ransomware Detection Using Dynamic Analysis And Machine Learning Techniques," *Wirel. Pers. Commun.*, Vol. 112, No. 4, Pp. 2597–2609, 2020, Doi: 10.1007/S11277-020-07166-9.

[7] S. Sheen And A. Yadav, "Ransomware Detection By Mining Api Call Usage," *2018 Int. Conf. Adv. Comput. Commun. Informatics, Icacci 2018*, Pp. 983–987, 2018, Doi: 10.1109/Icacci.2018.8554938.

[8] J. Zhou, M. Hirose, Y. Kakizaki, And A. Inomata, "Evaluation To Classify Ransomware Variants Based On Correlations Between Apis," In *Icissp 2020 - Proceedings Of The 6th International Conference On Information Systems Security And Privacy*, 2020, No. Icissp 2020, Pp. 465–472, Doi: 10.5220/0008959904650472.

[9] D. T. Nguyen And S. Lee, "Lightgbm-Based Ransomware Detection Using Api Call Sequences," *Int. J. Adv. Comput. Sci. Appl.*, Vol. 12, No. 10, Pp. 138–146, 2021, Doi: 10.14569/Ijacsa.2021.0121016.

[10] Y. A. Ahmed, B. Koçer, And B. A. S. Al-Rimy, "Automated Analysis Approach For The Detection Of High Survivable Ransomware," *Ksii Trans. Internet Inf. Syst.*, Vol. 14, No. 5, Pp. 2236–2257, 2020, Doi: 10.3837/Tiis.2020.05.021.

[11] B. Qin, Y. Wang, And C. Ma, "Api Call Based Ransomware Dynamic Detection Approach Using Textcnn," *Proc. - 2020 Int. Conf. Big Data, Artif. Intell. Internet Things Eng. Icbaie 2020*, Pp. 162–166, 2020, Doi: 10.1109/Icbaie49996.2020.00041.

[12] A. Ashraf, A. Aziz, U. Zahoora, And A. Khan, "Ransomware Analysis Using Feature Engineering And Deep Neural Networks," Pp. 1–15, 2019, [Online]. Available: Http://Arxiv.Org/Abs/1910.00286.

[13] K. Iwamoto And K. Wasaki, "Malware Classification Based On Extracted Api Sequences Using Static Analysis," In *Asian Internet Engineeering Conference, Aintec 2012*, 2012, No. November 2012, Pp. 31–38, Doi: 10.1145/2402599.2402604.

[14] Y. Ki, E. Kim, And H. K. Kim, "A Novel Approach To Detect Malware Based On Api Call Sequence Analysis," *Int. J. Distrib. Sens. Networks*, Vol. 2015, 2015, Doi: 10.1155/2015/659101.

[15] T. K. Tran And H. Sato, "Nlp-Based Approaches For Malware Classification From Api Sequences," *Proc. - 2017 21st Asia Pacific Symp. Intell. Evol. Syst. Ies 2017*, Vol. 2017-Janua, Pp. 101–105, 2017.

[16] S. Gupta, H. Sharma, And S. Kaur, "Malware Characterization Using Windows Api Call Sequences," *J. Cyber Secur. Mobil.*, Vol. 7, No. 4, Pp. 363–378, 2018, Doi: 10.13052/Jcsm2245-1439.741.

[17] Z. G. Chen, H. S. Kang, S. N. Yin, And S. R. Kim, "Automatic Ransomware Detection And Analysis Based On Dynamic Api Calls Flow Graph," *Proc. 2017 Res. Adapt. Converg. Syst. Racs 2017*, Vol. 2017-Janua, Pp. 196–201, 2017, Doi: 10.1145/3129676.3129704.

[18] P. Bajpai And R. Enbody, "An Empirical Study Of Api Calls In Ransomware," *Ieee Int. Conf. Electro Inf. Technol.*, Vol. 2020-July, Pp. 443–448, 2020, Doi: 10.1109/Eit48999.2020.9208284.

[19] Y. Fang, Y. Zeng, B. Li, L. Liu, And L. Zhang, *Deepdetectnet Vs Rlattacknet: An Adversarial Method To Improve Deep Learningbased Static Malware Detection Model*, Vol. 15, No. 4. 2020.

[20] A. Ninyesiga And J. Ngubiri, "Malware Classification Using Api System Calls," *Int. J. Technol. Manag.*, Vol. 3, No. 2, 2018, [Online]. Available: Https://Utamu.Ac.Ug/Ijotm/Index.Php/Ijotm/Article/View/41.

[21] R. Canzanese, S. Mancoridis, And M. Kam, "Run-Time Classification Of Malicious Processes Using System Call Analysis," *2015 10th Int. Conf. Malicious Unwanted Software, Malware 2015*, No. October, Pp. 21–28, 2016, Doi: 10.1109/Malware.2015.7413681.

[22] M. Schofield *Et Al.*, "Convolutional Neural Network For Malware Classification Based On Api Call Sequence," In *Computer Science & Information Technology (Cs & It)*, Jan. 2021, Pp. 85–98, Doi: 10.5121/Csit.2021.110106.

[23] Z. Yan, P. Zhang, And A. V. Vasilakos, "A Survey On Trust Management For Internet Of Things," *J. Netw. Comput. Appl.*, Vol. 42, Pp. 120–134, 2014, Doi: 10.1016/J.Jnca.2014.01.014.

[24] P. Casaseca-De-La-Higuera, M. Martín-Fernández, And C. Alberola-López, "Weaning From Mechanical Ventilation: A Retrospective Analysis Leading To A Multimodal Perspective," *Ieee Trans. Biomed. Eng.*, Vol. 53, No. 7, Pp. 1330–1345, 2006, Doi: 10.1109/Tbme.2006.873695.

[25] W. M. Coplin, D. J. Pierson, K. D. Cooley, D. W. Newell, And G. D. Rubenfeld, "Implications Of Extubation Delay In Brain-Injured Patients Meeting Standard Weaning Criteria," *Am. J. Respir. Crit. Care Med.*, Vol. 161, No. 5, Pp. 1530–1536, 2000, Doi: 10.1164/Ajrccm.161.5.9905102.

[26] B. A. S. Al-Rimy, M. A. Maarof, And S. Z. M. Shaid, "Crypto-Ransomware Early Detection Model Using Novel Incremental Bagging With Enhanced Semi-Random Subspace Selection," *Futur. Gener. Comput. Syst.*, Vol. 101, Pp. 476–491, 2019, Doi: 10.1016/J.Future.2019.06.005.