PDK

LAPORAN AKHIR

PROGRAM PENELTIAN DAN PENGABDIAN MASYARAKAT INTERNAL

SKIM: Penelitian Dasar Keilmuan(PDK)

Analisis Hasil Obtain dan Scrub Data Malware Menggunakan Metode Klasifikasi untuk Mendapatkan Insight Data

Oleh:

Vinna Rahmayanti SN, S.Si, M.Si (0706079004)

Denar Regata Akbi, S.Kom., M.Kom (0701058601)

DIREKTORAT PENELITIAN DAN PENGABDIAN PADA

MASYARAKAT UNIVERSITAS MUHAMMADIYAH MALANG

FEBRUARI 2022

HALAMAN PENGESAHAN

Judul	:	Analisis Hasil Obtain dan Scrub Data Malware Menggunakan Metode Klasifikasi untuk Mendapatkan Insight Data
Peneliti/Pelaksana	:	
a. Nama Lengkap	:	Denar Regata Akbi S.Kom., M.Kom.
b. Perguruan Tinggi	:	Universitas Muhammadiyah Malang
c. NIDN	:	0701058601
d. Jabatan Fungsional	:	Asisten Ahli
e. Program Studi	:	Teknik Informatika
f. Nomor HP	:	081330220014
g. Alamat surel (e-mail)	:	dnarregata@umm.ac.id
Anggota (1)	:	
a. Nama Lengkap	:	Vinna Rahmayanti S S.Si., M.Si
b. NIDN	:	0706079004
c. Perguruan Tinggi	:	Universitas Muhammadiyah Malang
Anggota Mahasiswa	:	Ridhi Pratomo Pramudana (201610370311141), Muhammad Zahid Humam (201610370311073),
Tahun Pelaksanaan	:	Tahun ke 1 dari rencana 1 tahun
Biaya Penelitian	:	
Rincian Biaya	:	Tahun ke-1: diusulkan ke dppm Rp. 17,000,000 Total dana usulan: Rp. 17,000,000 Tahun ke-1: dana institusi lain Rp. - Total incash: Rp. - Tahun ke-1: Inkind Rp. - Total inkind: Rp. -
Biaya Disetujui	:	Rp. 15,300,000

Mengetahui
Dekan / Wakil

Dr. Ahmad Mubin M.T.
NIP. 196411171990031003

Malang, 28 Februari 2022
Ketua Pelaksana,

Denar Regata Akbi S.Kom., M.Kom.
NIP. 10816120591

Menyetujui
Direktur DPPM / Wakil,

Prof. Dr. Yus Mochamad Cholily M.Si
NIP.

Lembar pengesahan ini di*generate* oleh SIMPPM - UMM - 20220228141342

RINGKASAN

Malware merupakan perangkat lunak berbahaya yang dapat mengganggu kinerja dari suatu sistem, dan telah menjadi salah satu cyber threat yang perlu mendapat perhatian khusus. Semakin hari perkembangan malware semakin berbagai macam dan mengalami evolusi semakin canggih, sehingga mempunyai kemampuan untuk melindungi diri dari suatu acaman baik itu antivirus atau sistem pengamanan yang lain. Pesatnya perkembangan malware tersebut, memerlukan perhatian dari berbagai macam pihak, baik dari peneliti pada bidang malware, dan stakeholder – stakeholder, salah satunya seperti Pusat Malware Nasional (Pusmanas) BSSN, untuk dapat saling bekerja sama meminimalisir akibat dari malware cyber threat. Salah satu upaya awal yang dapat dilakukan adalah melakukan analisis terhadap malware – malware yang ada, analisis dalam hal ini merupakan suatu proses untuk melakukan identifikasi terhadap perilaku malware, mulai dari apa yang dilakukan, apa yang diinginkan, dan apa tujuan utama dari malware tersebut. Tujuan utama dilakukannya analisis terhadap data – data malware tersebut, untuk mengetahui dan memahami bagaimana malware bekerja, dengan didapatkannya pengetahuan tersebut, pihak organisasi, stakeholder, ataupun para peneliti malware, dapat memanfaatkannya untuk mengembangkan suatu framework, antimalware, ataupun antivirus. Oleh karena itu, analisis yang dilakukan pada data – data malware tersebut memerlukan teknik, metode ataupun data yang baik. Dengan melakukan analisis pada data – data malware dengan kualitas yang baik, hal tersebut dapat mendukung langkah dalam pembuatan suatu suatu framework, antimalware, ataupun antivirus yang baik pula dalam meminimalisir terjadinya malware cyber threat. Analisis secara mendalam sangat diperlukan untuk menghasilkan sistem pendeteksi yang baik, analisis yang bisa dilakukan salah satunya pada bagian fitur dari data malware, dimana data dari fitur tersebut mencerminkan karakterisktik dari suatu malware, karakteristik dari malware satu dengan malware yang lain berbeda. Perkembangan malware dengan perkembangan sistem pengamanan berjalan beriringan, banyak pembuat malware yang telah menerapkan dan membekali malware buatannya dengan kecerdasan buatan, disisi lain para peneliti menerapkan metode machine learning dan deep learning yang merupakan cabang ilmu dari kecerdasan buatan untuk melakukan penelitian terhadap karakteristik malware, dengan melakukan analisis terhadap karakteristik dari suatu varian malware, seperti menggunakan metode klasifikasi diharapkan hal tersebut dapat memberikan referensi untuk pembuatan sistem pengamanan terhadap malware yang lebih baik. Pada penelitian yang akan dilakukan peneliti mencoba untuk melakukan analisis terhadap data malware yang diambil dari Canadian Institute for Cybersecurity, dimana data – data malware tersebut diambil dari real phone, sehingga bisa mewakili kondisi nyata dari data malware yang sebenarnya. Langkah awal yang dilakukan oleh peneliti adalah melakukan input data malware pada sistem kemudian dilakukan pembagian data, pembagian dilakukan untuk membagi persentase data test dan data training. Pembagian tersebut nantinya akan dilakukan variasi dimana pembagian pertama disesuaikan dengan kondisi pada penelitian terdahulu serta pembagian kedua dan seterusnya peneliti mencoba untuk melakukan variasi persentase pembagian data test dan data training. Setelah itu akan dilakukan preprocessing, langkah tersebut dilakukan untuk mendapatkan fitur – fitur utama dari data - data malware yang digunakan, setelah itu dilakukan klasifikasi menggunakan metode deep learning, hasil dari metode ini nantinya akan bandingkan dengan penelitian terdahulu, dengan pemilihan metode ini diharapkan mendapatkan hasil akurasi lebih baik dari penelitian sebelumnya. sehingga bisa dijadikan acuan dalam pembuatan atau pengembangan suatu sistem pengamanan terhadap malware

Kata Kunci: Malware, Fitur, Data, Canadian Institute for Cybersecurity

PRAKATA

Alhamdulillah, puji syukur kehadirat Allah SWT yang telah memberikan rahmat dan inayah-Nya sehingga kami dapat menyelesaikan Laporan Akhir Penelitian internal yang berjudul Analisis Hasil Obtain dan Scrub Data Malware Menggunakan Metode Klasifikasi untuk Mendapatkan Insight Data.

Kami menyadari, bahwa laporan akhir ini masih jauh dari kata sempurna baik segi penyusunan, bahasa, maupun penulisannya. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari semua pembaca guna menjadi acuan agar penulis bisa menjadi lebih baik lagi di masa mendatang.

Semoga laporan akhir ini bisa menambah wawasan para pembaca dan bisa bermanfaat untuk perkembangan dan peningkatan ilmu pengetahuan.

Malang, 27 Februari 2022

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

Malware merupakan perangkat lunak yang dibuat untuk beberapa tujuan yang dapat merugikan, seperti merusak kinerja sistem, merusak data, tercurinya data, dan tujuan lainnya yang merugikan. Beberapa pihak seperti Fortinet, Tencent Mobile Security Lab's, mencatat peningkatan beberapa jenis malware sekitar 1.800 sampai 1.829.188 malware dalam kurun waktu cuma 2 tahun, peningkatan kurang lebih 12 kali lipat tersebut perlu mendapat perhatian [1][2]

Analisis terhadap jenis - jenis malware yang ada perlu dilakukan secara terus menerus, hasil dari analisis yang telah dilakukan nantinya dapat dimanfaatkan serta dijadikan referensi bagi pihak organisasi ataupun beberapa peneliti malware, untuk mengembangkan suatu framework, antimalware, ataupun antivirus, terlebih lagi saat ini Indonesia telah membuat Pusat Malware Nasional (Pusmanas) BSSN, dimana salah satu tugasnya melakukan analisis dan pengarsipan data malware, sehingga penelitian ini diharapkan memiliki kontribusi kepada berbagai macam pihak terkait.

Beberapa tahun terakhir terdapat banyak varian baru malware bermunculan dengan berbagai macam fitur [3][4][5], untuk meminimalisir aktivitas kejahatan dunia maya yang disebabkan oleh malware, maka salah satu usaha yang bisa dilakukan adalah melakukan analisis karakteristik dari malware dengan mengklasifikasikan berdasarkan fitur - fitur yang melekat[6].

Penelitian terdahulu, metode machine leaning digunakan untuk pengklasifikasian fitur malware, dimana pada kasus tertentu diperlukan metode tambahan untuk mendukung akurasi proses pengklasifikasian malware, metode seperti Principal Component Analysis (PCA), histogram, C4, C5, dan metode lainnya merupakan metode tambahan untuk ekstraksi fitur utama yang melekat pada malware[7][8]. Metode deep learning menawarkan hal yang sedikit berbeda, dimana pada beberapa metode deep learning telah menyediakan metode tambahan tersebut pada satu metode.

Penelitian ini menggunakan metode deep learning yaitu Convolutional Neural Network (CNN). CNN memiliki kemampuan yang lebih baik dibandingkan dengan pendekatan konvolusional karena memiliki arsitektur konvolusional layer, pooling layer, dan multi-connected layer. Hasil CNN dapat ditunjukkan pada penelitian yang bertujuan untuk mendeteksi aplikasi komputer yang didapatkan dari *honeypot*, Github, dan aplikasi dari sistem windows. Hasil akurasi yang didapatkan lebih dari 90% dalam mengidentifikasi aplikasi malware dan bukan *malware*. Penelitian ini juga menunjukkan bahwa CNN sangat efektif untuk mengidentifikasi *malware* yang tertanam ke dalam kode aplikasi yang aman, sehingga *malware* tidak memiliki tempat untuk bersembunyi.

Penelitian yang telah dijabarkan menggerakkan peneliti untuk melakukan analisis data terhadap data malware menggunakan Convolutional Neural Network. Peneliti mencoba untuk melakukan analisis dataset malware pada Canadian Institute for Cybersecurity, yang telah ditambahkan network-flows dan API Calls, yang nantinya hasil akurasi akan diperbandingkan dengan penelitian sebelumnya, diharapkan metode yang digunakan bisa mencapai akurasi lebih baik dari penelitian terdahulu.

Dataset malware publik dari Canadian Institute for Cybersecurity digunakan pada penelitian sebelumnya. Dataset tersebut banyak digunakan oleh beberapa peneliti malware, dikarenakan pada dataset publik lain seperti Maldozer, UCL, AMD, AAGM, Kharon, Andro-Profiler dan lainnya terdapat beberapa kekurangan, kekurangan - kekurangan tersebut pada dataset yang dimiliki oleh Canadian Institute for Cybersecurity telah diperbaiki [9]. Penelitian terdahulu, analisis dilakukan menggunakan metode klasifikasi machine learning

Pada awal penelitian ini, peneliti mencoba untuk melakukan analisis menggunakan metode deep learning pada 350 data malware yang memiliki 918 fitur. Hasil penelitian akan

dibandingkan dengan penelitian terdahulu. Penelitian berikutnya data malware akan ditambahkan sesuai dengan kebutuhan penelitian kemudian dianalisis menggunakan deep learning.

BAB 2 TINJAUAN PUSTAKA

Gambar 1. Roadmap penelitian untuk kasus penggunaan machine learning pada malware

Pada Gambar 1. Merupakan beberapa penelitian mulai tahun 2018 sampai tahun 2019, yang membahas terkait malware, penelitian yang akan dilakukan oleh peneliti kali ini, peneliti menggunakan data yang digunakan juga oleh Lashkari, A.H., et.al dan L, Taheri., et.al. Data malware yang digunakan oleh kedua peneliti tersebut telah melalui tahap uji coba dan beberapa peneliti malware lain juga menggunakan data yang sama untuk melakukan penelitian yang difokuskan pada malware yang menyerang sistem Android. Data tersebut selalu diperbaharui dan perbaiki oleh peneliti – peneliti pada Canadian Institute for Cybersecurity, agar data tersebut bisa digunakan oleh peneliti – peneliti lain, oleh karena itu penelitian pada skim Penelitian Dasar Keilmuan (PDK) kali ini menggunakan data tersebut. Terlebih lagi sistem Android sekarang merupakan salah satu sistem yang banyak digunakan oleh orang – orang hampir di seluruh dunia, sehingga data tersebut sangat membantu para peneliti malware untuk melakukan penelitiannya.

Pada tahun 2018, penelitian yang dilakukan oleh Lashkari, A.H., et.al [10], mengusulkan untuk menggunakan dataset yang diambil dari real-phone, dikarenakan dataset yang didapatkan dari emulator tidak cocok ketika digunakan untuk membuat sistem pengamanan pada kondisi real, sehingga data pada penelitian ini diambil dari real-phone, data – data malware yang diambil dari real-phone tersebut dikumpulkan menjadi satu pada lingkungan penelitian di Canadian Institute for Cybersecurity, pada penelitian ini peniliti mencoba untuk melakukan analisis pada data – data malware yang dimaksud dengan menggunakan tiga skenario, skenario yang pertama peneliti mencoba untuk melakukan analisis terkait malware binary detection, pada skenario kedua peneliti mencoba untuk melakukan analisis terkait malware category classification, pada skenario ketiga peneliti mencoba untuk melakukan analisis terkait malware family characterization. Pada penelitian ini peneliti mencoba untuk melakukan evaluasi hasil penelitian yang dilakukan menggunakan metode machine learning, metode machine learning yang digunakan terdapat tiga macam, diantaranya adalah Random Forest (RF), K-Nearest Neighbor (KNN), serta Decision Tree (DT). Data – data malware yang digunakan pada penelitian ini didapat dari GooglePlay dari tahun 2015, 2016, 2017. Topologi yang digunakan untuk mendapatkan data – data malware, diantaranya, peneliti menggunakan modem untuk terkoneksi internet, sebuah server untuk menampung data – data malware, kemudian terdapat tiga node yang berupa laptop dengan real phone yang terhubung, real phone ini digunakan untuk mendapatkan malware dari GooglePlay. Setelah data – data dari malware didapatkan, peneliti

menggunakan metode – metode machine learning seperti RF, KNN, serta DT untuk melakukan analisis, hasil yang didapatkan menunjukkan akurasi rata – rata 85% sampai 88% dengan menggunakan tiga metode machine leaning, yaitu Random Forest (RF), K-Nearest Neighbor, serta Decision Tree (DT).

Penelitian yang dilakukan M.Murtaz., et.al [11], pada tahun 2018 dilakukan pada varian malware yang memiliki kemampuan untuk menyembunyikan aktifitasnya terhadap sistem pengamanan yang ada. Tujuan dari penelitian ini adalah membuat suatu sistem yang diharapkan dapat melindungi pengguna perangkat seluler dan perusahaan infrastruktur seluler dari malware, sistem yang direncanakan nantinya dapat digunakan untuk mengamati perilaku aplikasi berbahaya yang telah terinfeksi malware serta adware. Peneliti melakukan analisis terhadap data malware yang ada menggunakan metode machine learning sebelum melakukan pembuatan sistem tersebut, hasil analisis yang telah dilakukan didapatkan akurasi rata – rata mencapai 94% dengan menggunakan 5 metode machine learning diantaranya Random Forest (RF), K-Nearest Neighbor (KNN), decision Tree (DT), Random Tree (RT) dan Regression (R).

F Noorbehbahani, et.al [12], melakukan penelitian yang dikhususkan pada malware varian ransomware, dimana dengan meningkatnya pertumbuhan internet dan jaringan komputer jumlah malware terus meningkat, salah satunya adalah ransomware, menurut peneliti ransomware merupakan ancaman terbesar dalam keamanan siber, dengan adanya hal tersebut, peneliti menggunakan metode machine learning untuk melakukan deteksi malware yang di khususkan pada ransomware, skenario pengujian yang dilakukan oleh peneliti ada 2 macam, yang pertama peneliti melakukan klasifikasi terhadap dataset yang berisi berbagai macam jenis ransomware, sedangkan pada skenario kedua peneliti melakukan klasifikasi pada 10 jenis data ransomware. Pada kedua skenario tersebut peneliti mendapatkan hasil klasifikasi menggunakan metode random forest lebih baik jika dibandingkan dengan metode machine learning yang lain.

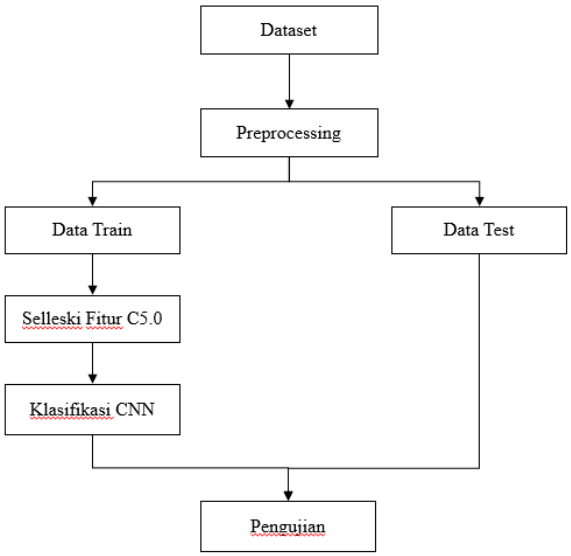
Penelitian yang dilakukan oleh L.Taheri, et.al [9], merupakan penelitian lanjutan yang dilakukan oleh Lashkari, A.H., et.al, penelitian yang dilakukan oleh Lashkari, A.H., et.al lebih berfokus pada mendapatkan data – data malware real, menggunakan real phone dan data malware diambil dari GooglePlay mulai tahun 2015 sampai 2017, sedangkan pada penelitian yang dilakukan oleh L.Taheri, et.al, mencoba untuk memperbaiki data – data yang didapatkan oleh Lashkari, A.H., et.al dengan menambahkan network-flows dan API Calls. Terdapat beberapa kontribusi perbaikan penelitian dari sisi data malware yang telah di dapatkan dari penelitian yang telah dilakukan oleh Lashkari, A.H., et.al. kontribusi perbaikan yang dilakukan oleh L.Taheri, et.al diantaranya: 1. Mengusulkan perbandingan antara beberapa kumpulan data malware Android yang di biasa dipakai oleh beberapa peneliti sebelumnya, perbandingan dilakukan berdasarkan parameter 15 kriteria yang dianggap penting, 2. Melakukan improvisasi perbaikan pada data set CICAndMal2017 dengan menambahkan fitur Permission dan Intent sebagai static fitur, dan penambahan API Calls sebagai dynamic fitur, 3. Menyajikan dua layer analisis, yang dilakukan menggunakan metode static dan dynamic analisis, 4. Melakukan improvisasi pada bagian analisis network-flow dengan menambahkan relasi sekuensial n-gram yang diekstraksi dari fitur API Calls. Dari hasil perbandingan beberapa dataset publik yang digunakan oleh peneliti – peneliti malware, data set CICAndMal2017 bisa dikatakan dataset yang cukup lengkap jika dibandingkan dengan dataset publik lain, terlebih lagi data yang terkumpul pada CICAndMal2017 diambil dari real phone, yang di klaim oleh peneliti, hal tersebut bisa mewakili data real dari malware. Jika dibandingkan data set lain yang diambil secara real menggunakan real phone seperti data set kharon, dan data set AAGM, data set CICAndMal2017 bisa dikatakan dataset yang paling lengkap. Hasil penelitian yang dilakukan oleh L.Taheri, et.al didapatkan akurasi pada Static-Based Malware Binary berkisar 95,3%, sedangkan pada Dynamic-Based Malware Category dan Dynamic-Based Malware Family didapatkan 83,3% dan 59,7%

BAB 3 TUJUAN DAN MANFAAT PENELITIAN

Tujuan utama dilakukannya analisis terhadap data – data malware tersebut, untuk mengetahui dan memahami bagaimana malware bekerja, dengan didapatkannya pengetahuan tersebut, pihak organisasi, stakeholder, ataupun para peneliti malware, dapat memanfaatkannya untuk mengembangkan suatu framework, antimalware, ataupun antivirus. Oleh karena itu, analisis yang dilakukan pada data – data malware tersebut memerlukan teknik, metode ataupun data yang baik. Dengan melakukan analisis pada data – data malware dengan kualitas yang baik, hal tersebut dapat mendukung langkah dalam pembuatan suatu suatu framework, antimalware, ataupun antivirus yang baik pula dalam meminimalisir terjadinya malware cyber threat. Analisis secara mendalam sangat diperlukan untuk menghasilkan sistem pendeteksi yang baik, analisis yang bisa dilakukan salah satunya pada bagian fitur dari data malware, dimana data dari fitur tersebut mencerminkan karakterisktik dari suatu malware, karakteristik dari malware satu dengan malware yang lain berbeda. Perkembangan malware dengan perkembangan sistem pengamanan berjalan beriringan, banyak pembuat malware yang telah menerapkan dan membekali malware buatannya dengan kecerdasan buatan, disisi lain para peneliti menerapkan metode machine learning dan deep learning yang merupakan cabang ilmu dari kecerdasan buatan untuk melakukan penelitian terhadap karakteristik malware, dengan melakukan analisis terhadap karakteristik dari suatu varian malware, seperti menggunakan metode klasifikasi diharapkan hal tersebut dapat memberikan referensi untuk pembuatan sistem pengamanan terhadap malware yang lebih baik.

BAB 4 METODE PENELITIAN

Penelitian ini menggunakan dataset pada Canadian Institute for Cybersecurity, yang diberi nama CICAndMal2017 [9] dimana pada dataset tersebut telah ditambahkan network-flows dan API Calls. Alur penelitian ini digambarkan pada Gambar 2



Gambar 2. Bagan Alur Penelitian

Pada Gambar 2. Dapat dijelaskan dataset yang digunakan CICAndMal2017, pada dataset tersebut akan dilakukan preprocessing terlebih dahulu, kemudian akan dilakukan pembagian pada dataset, pembagian tersebut dilakukan dengan membagi sebagian data sebagai data train sedangkan sebagian lagi sebagai data test, perbandingan data train dan data test dilakukan dengan beberapa skenario, diantaranya skenario pertama persentase pembagian data train dan data test disesuaikan dengan penelitian sebelumnya, skenario kedua peneliti mencoba untuk melakukan variasi persentase pembagian data train dan data test diluar persentase penelitian sebelumnya. Data malware CICAndMal2017 nantinya juga akan di proses menggunakan metode seleksi fitur C5.0, hal ini dilakukan untuk menganalisis fitur – fitur apa saja dari dataset tersebut yang merupakan fitur utama, yang nantinya fitur tersebut bisa digunakan untuk proses selanjutnya. Setelah itu akan dilakukan klasifikasi menggunakan metode CNN, dan yang terakhir adalah melihat akurasi yang dihasilkan, yang nantinya hasil akurasi tersebu akan dibandingkan dengan penelitian sebelumnya, diharapkan pada penelitian ini nilai akurasi bisa lebih baik dari penelitian sebelumnya.

Penelitian ini menggunakan dataset pada Canadian Institute for Cybersecurity, yang diberi nama CICAndMal2017 [9] dimana pada dataset tersebut telah ditambahkan network-flows dan API Calls. Dataset ini merupakan dataset yang diambil dan dikumpulkan pada kondisi real dengan menggunakan real-phone, hal tersebut dilakukan karena pada penelitian sebelumnya data yang diambil dari emulator handphone, tidak bisa mewakili kondisi data pada kenyataan, sehingga diambil langkah untuk mendapatkan dan mengumpulkan data malware pada lingkungan real. berikut merupakan perbandingan dataset malware publik yang digunakan oleh beberapa peneliti.

Tabel 1. Perbandingan dataset publik yang digunakan beberapa peneliti [9]

Dataset Name	Published Year	Number of Benignware	Number of Malware	Captured Static Features						Captured Dynamic Features					Installed On
				States	Permission	Intent	Component	Certification	SourceCode	API.Call	Network	Sys.Call	Infoflow	Log	
Genome [27]	2012	-	1260	No	Yes	No	Yes	No	Yes	No	No	No	No	No	-
Drebin [3]	2014	123,453	5560	No	Yes	Yes	Yes	No	Yes	No	No	No	No	No	-
AndroTracker [13]	2015	51,179	4,554	No	Yes	Yes	No	Yes	Yes	No	No	No	No	No	-
SAPIMMDS [11]	2016	1776	906	No	No	No	No	No	No	Yes	No	No	No	No	Emulator
Andro-Dumpsys [24]	2016	1776	906	No	Yes	Yes	No	Yes	Yes	Yes	No	No	No	No	Emulator
Andro-Profiler [12]	2016	8840	643	No	No	No	No	No	No	No	No	Yes	No	Yes	Emulator
Kharon [6]	2016	-	7	No	No	No	No	No	No	No	No	No	Yes	No	RealPhone
AAGM [15]	2017	1500	400	No	No	No	No	No	No	No	Yes	No	No	No	RealPhone
AMD [23]	2017	-	405	No	No	No	Yes	No	Yes	No	No	No	No	No	-
MalDozer [14]	2018	38,000	33,000	No	No	No	No	No	Yes	No	No	No	No	No	-
UCL [20]	2018	1.2M	-	No	No	No	Yes	No	Yes	No	No	No	No	No	-
CICAndMal2017 [16]	2018	1700	426	Yes	Yes	Yes	No	No	No	Yes	Yes	No	No	No	RealPhone

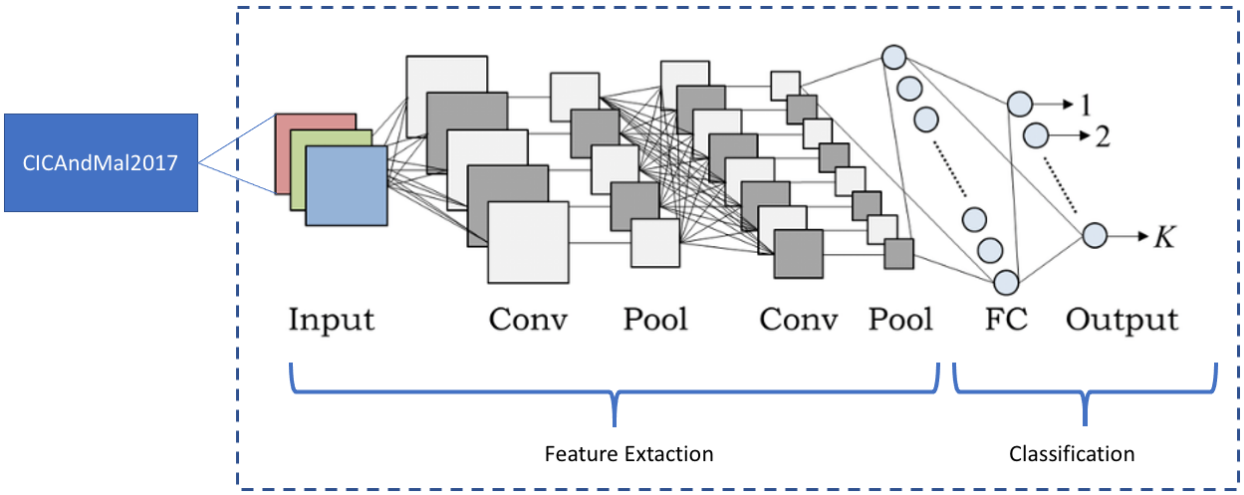
Pada Tabel 1. Merupakan perbandingan dataset yang dilakukan oleh L.Taheri, et.al. perbandingan tersebut dilakukan dengan membandingkan beberapa dataset publik yang digunakan oleh peneliti – peneliti malware. Dari table perbandingan tersebut bisa dikatakan dataset CICAndMal2017 merupakan dataset yang cukup lengkap.

Pada penelitian ini, hal pertama yang dilakukan adalah melakukan replikasi dari hasil penelitian sebelumnya dengan penggunaan metode, penggunaan data, serta pembagian data yang sama. Hal tersebut dilakukan untuk mendapatkan hasil yang nantinya bisa dibandingkan dengan penelitian yang akan dilakukan.

Pembagian data latih dan data uji pada penelitian lanjutan dengan menggunakan metode deep learning disesuaikan dengan penelitian sebelumnya yaitu 60:40. Hal tersebut dilakukan untuk melakukan claim hasil pengujian pada penelitian yang akan dilakukan. Hasil dari penelitian sebelumnya dan penelitian yang akan dilakukan akan diperbandingkan dari sisi akurasi hasil, sehingga dapat terlihat dataset CICAndMal2017 lebih baik menggunakan metode machine learning atau deep learning.

Data train selanjutnya dilakukan ekstraksi fitur menggunakan metode C5.0. Hasi dari ekstraki fitur akan diproses untuk tahap klasifikasi tiap *malware* menggunakan metode CNN.

Pada penelitian ini, peneliti mencoba menggunakan metode *Convloutional Neural Network* (CNN) yang merupakan salah satu metode yang ada pada deep learning. Manfaat yang paling signifikan dari CNN dibandingkan pendekatan konvensional adalah secara otomatis memperoleh fitur-fitur utama untuk klasifikasi dalam proses pembelajaran[13]. Gambar 2 menunjukkan tahapan yang dilakukan pada penelitian ini.



Gambar 3. Arsitektur metode CNN secara umum yang akan digunakan

Pada Gambar 2 merupakan arsitektur umum metode CNN yang akan digunakan pada penelitian yang akan dilakukan [14]. Langkah-langkah penggunaan algoritma CNN memasukkan dataset yang telah dilakukan pada proses preprocessing, CNN otomatis akan memisahkan tiap fitur sesuai dengan label, memasukkan fitur yang telah dipisahkan ke dalam *array*, melakukan proses max-pooling untuk mengecilkan ukuran *array*, dan membuat prediksi indikator dari setiap *malware* sesuai dengan *family*nya. Pengujian dilakukan untuk mengetahui hasil akurasi dari algoritma yang telah dilakukan. Penelitian sebelumnya yang dilakukan oleh L. Taheri et.al tidak dilakukan ekstraksi fitur pada dataset tersebut, sedangkan pada penelitian ini dilakukan dengan menggunakan metode CNN yang mana dengan menggunakan metode tersebut diharapakan bisa mencapai akurasi lebih baik dari penelitian terdahulu

BAB 5 HASIL DAN PEMBAHASAN

5.1. Implementasi

Untuk melakukan penelitian ini, perlu dilakukan *preprocessing* terlebih dahulu. Lalu akan berlanjut ke *labeling* untuk memberikan labeling kepada data yang sudah dilakukan *preprocessing* menggunakan C5.0 dikarenakan nama dari tiap atribut sudah berubah. Data yang sudah melalui proses ekstraksi fitur dan *preprocessing* akan dilakukan proses klasifikasi menggunakan metode CNN.

5.1.1 Preprocessing

Sebelum dilakukan klasifikasi dan ekstraksi fitur menggunakan CNN, perlu dilakukan tahap *feature selection* agar jumlah atribut yang sangat banyak sebelumnya dapat diringkas dan memudahkan untuk dilakukan pemrosesan. Tahap yang dilalui yaitu:

a. Feature Selection menggunakan C5.0

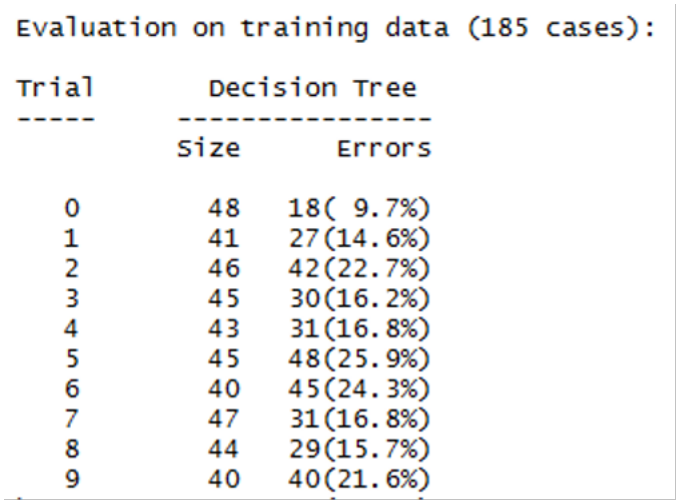
Pada tahap *feature selection*, metode yang digunakan adalah *C5.0* yang berfungsi untuk meringkas dari sebanyak 918 atribut menjadi 129 atribut.

Penggunaan C5.0 dikarenakan metode ini dapat memunculkan seberapa pengaruh dari tiap atribut untuk dilakukan proses klasifikasi menggunakan aplikasi *Anaconda Studio* dan bahasa pemrograman R. Untuk dokumentasi proses *feature selection* dapat dilihat dari gambar 5.1

```
5
6  ##import dataset
7  malware_dataset = read.csv("Datasetfull.csv")
8
9  ##convert atribut family dari char -> factor
10 malware_dataset$Family <- as.factor(malware_dataset$Family)
11 str(malware_dataset$Family)
12
13
14 ##split data
15 library(caTools)
16 split = sample.split(malware_dataset$Family, SplitRatio = 0.60)
17 training_set = subset(malware_dataset[-1], split == TRUE)
18 test_set = subset(malware_dataset[-1], split == FALSE)
19
20 ##training data
21 library(C50)
22 malw_model = C5.0(formula = Family ~ ., data = training_set )
23 malw_model
24 plot(malw_model)
25 summary(malw_model)
26
27 ##training data dengan 10x percobaan
28 malw_model = C5.0(formula = Family ~ ., data = training_set, trials=10 )
29 plot(malw_model)
30 summary(malw_model)
31
```

Gambar 4. Source Code dari Feature Selection menggunakan C5.0

Dari pemrosesan pada gambar 5.1, menghasilkan total 40 fitur yang paling berpengaruh dalam dataset dengan nilai *error* 9.7% untuk percobaan pertama dan dilakukan percobaan sebanyak 10x sesuai dengan gambar 5.2. Proses ini dilakukan guna menunjukkan seberapa besar persentase pengaruh tiap fitur terhadap tiap *family* yang ditampilkan pada gambar 5.3 sesuai dengan *Attribute Usage*, untuk fitur yang tidak masuk kedalam daftar *Attribut Usage* nantinya dapat dihapus dan tidak perlu digunakan.



```
Evaluation on training data (185 cases):

Trial        Decision Tree
-----      ----------------
           Size      Errors

   0         48   18( 9.7%)
   1         41   27(14.6%)
   2         46   42(22.7%)
   3         45   30(16.2%)
   4         43   31(16.8%)
   5         45   48(25.9%)
   6         40   45(24.3%)
   7         47   31(16.8%)
   8         44   29(15.7%)
   9         40   40(21.6%)
```

Gambar 5. Daftar nilai error pada tiap percobaan

```
Attribute usage:

100.00% Down.Up_Ratio
 80.54% X2Grammed_APICalls_.416
 74.59% X2Grammed_APICalls_.41
 71.89% X2Grammed_APICalls_.30
 68.11% X2Grammed_APICalls_.411
 65.95% X2Grammed_APICalls_.142
 64.86% X2Grammed_APICalls_.59
 60.54% X2Grammed_APICalls_.50
 50.27% Bwd_Packets.s
 28.65% X2Grammed_APICalls_.125
 27.03% X2Grammed_APICalls_.86
 24.32% X2Grammed_APICalls_.301
 22.70% Idle_Std
 21.62% X2Grammed_APICalls_.35
 20.00% X2Grammed_APICalls_.264
 19.46% X2Grammed_APICalls_.213
 18.92% X2Grammed_APICalls_.283
 18.38% X2Grammed_APICalls_.77
 17.30% X2Grammed_APICalls_.109
 17.30% X2Grammed_APICalls_.170
 17.30% Fwd_PSH_Flags
 16.76% X2Grammed_APICalls_.408
 15.68% Total_Length_of_Fwd_Packets
 14.59% X2Grammed_APICalls_.40
 14.05% X2Grammed_APICalls_.24
 12.97% X2Grammed_APICalls_.18
 10.27% X2Grammed_APICalls_.4
 10.27% X2Grammed_APICalls_.99
 10.27% X2Grammed_APICalls_.217
 10.27% X2Grammed_APICalls_.357
  8.65% X2Grammed_APICalls_.221
  7.57% X2Grammed_APICalls_.12
  7.57% Fwd_Packet_Length_Min
  7.03% X2Grammed_APICalls_.1
```

Gambar 6. Daftar attribute usage

5.1.2 Ekstraksi Fitur

Pada tahap ini setelah data melalui proses *training* menggunakan C5.0 akan muncul beberapa daftar atribut yang paling berpengaruh terhadap tiap *family malware* menggunakan cara C5.0 akan melakukan *decision tree* sebanyak 10x seperti pada gambar 5.2 dengan berbeda *root*. Pada gambar 5.4, *root* yang digunakan pada percobaan pertama adalah atribut `Down.Up_Ratio`. Pada percobaan kedua dan ketiga, root yang digunakan adalah `Min_Packet_Length` sesuai dengan gambar 5.5 dan 5.6. Pada percobaan keempat, root yang digunakan adalah `X2Grammed_APICalls_.7` seperti pada gambar 5.7. Untuk atribut `X2Grammed_APICalls_.53` dijadikan root pada percobaan kelima dan sesuai dengan gambar 5.8. Untuk percobaan keenam sampai dengan percobaan kedelapan, atribut root yang digunakan tetap sama, yaitu `PSH_Flag_Count` sesuai dengan gambar 5.9 sampai dengan gambar 5.11. Dan untuk

percobaan kesembilan sampai dengan kesepuluh, root yang digunakan adalah `X2Grammed_APICalls_.416` seperti pada gambar 5.12 dan 5.13.

```
Down.Up_Ratio <= 9:
:...X2Grammed_APICalls_.213 > 0: fakemart (5)
:   X2Grammed_APICalls_.213 <= 0:
:   :...X2Grammed_APICalls_.408 > 1: penetho (2)
:       X2Grammed_APICalls_.408 <= 1:
:       :...Total_Length_of_Fwd_Packets > 32747: pletor (5)
:           Total_Length_of_Fwd_Packets <= 32747:
:           :...X2Grammed_APICalls_.142 > 0: mazarbot (5)
:               X2Grammed_APICalls_.142 <= 0:
:               :...X2Grammed_APICalls_.357 > 0: zsone (4)
:                   X2Grammed_APICalls_.357 <= 0:
:                   :...X2Grammed_APICalls_.109 <= 1:
:                       :...X2Grammed_APICalls_.24 <= 0: fakeinst (2/1)
:                       :   X2Grammed_APICalls_.24 > 0: jifake (5)
:                       X2Grammed_APICalls_.109 > 1:
:                       :...Fwd_Packet_Length_Min <= 85: jifake (2/1)
:                           Fwd_Packet_Length_Min > 85: beanbot (6)
```

Gambar 7. Decision tree percobaan pertama

```
-----  Trial 1:  -----

Decision tree:

Min_Packet_Length <= 227:
:...X2Grammed_APICalls_.408 > 1: penetho (2.3)
:   X2Grammed_APICalls_.408 <= 1:
:   :...X2Grammed_APICalls_.213 > 0: fakemart (3.9)
:       X2Grammed_APICalls_.213 <= 0:
:       :...Active_Std > 966754: pletor (3.9)
:           Active_Std <= 966754:
:           :...X2Grammed_APICalls_.108 > 0: beanbot (3.9)
:               X2Grammed_APICalls_.108 <= 0:
:               :...X2Grammed_APICalls_.14 > 0: zsone (8.5)
:                   X2Grammed_APICalls_.14 <= 0:
:                   :...Idle_Std <= 3759739: jifake (7/2.3)
:                       Idle_Std > 3759739: mazarbot (7.7/1.6)
```

Gambar 8.Decision tree percobaan kedua

```
-----  Trial 2:  -----

Decision tree:

Min_Packet_Length <= 227:
:...X2Grammed_APICalls_.138 > 28: fakeinst (2.1)
:   X2Grammed_APICalls_.138 <= 28:
:   :...X2Grammed_APICalls_.213 > 0: fakemart (3)
:       X2Grammed_APICalls_.213 <= 0:
:       :...act_data_pkt_fwd > 22:
:           :...Fwd_PSH_Flags <= 0: pletor (3)
:           :   Fwd_PSH_Flags > 0: lockerpin (3.9/1.8)
:           act_data_pkt_fwd <= 22:
:           :...X2Grammed_APICalls_.357 > 0: zsone (6.3)
:               X2Grammed_APICalls_.357 <= 0:
:               :...X2Grammed_APICalls_.13 > 0: zsone (2.4)
:                   X2Grammed_APICalls_.13 <= 0:
:                   :...X2Grammed_APICalls_.108 > 0: beanbot (3)
:                       X2Grammed_APICalls_.108 <= 0:
:                       :...Bwd_IAT_Total <= 1.345803e+007: beanbot (2.1)
:                           Bwd_IAT_Total > 1.345803e+007:
:                           :...X2Grammed_APICalls_.171 > 0: mazarbot (3.9)
:                               X2Grammed_APICalls_.171 <= 0:
:                               :...Fwd_IAT_Total <= 8.531263e+007: jifake (3.6)
:                                   Fwd_IAT_Total > 8.531263e+007: mazarbot (3)
```

Gambar 9.Decision tree percobaan ketiga

```
-----  Trial 3:  -----

Decision tree:

X2Grammed_APICalls_.7 > 0:
:...X2Grammed_APICalls_.66 > 108: virusshield (2.6/1)
:   X2Grammed_APICalls_.66 <= 108:
:   :...X2Grammed_APICalls_.41 <= 0: feiwo (5.9)
:       X2Grammed_APICalls_.41 > 0: jisut (2.4)
X2Grammed_APICalls_.7 <= 0:
:...Fwd_Packets.s <= 67208:
    :...X2Grammed_APICalls_.408 > 1: penetho (4.6/0.5)
    :   X2Grammed_APICalls_.408 <= 1:
    :   :...Idle_Max > 9.964572e+007: mobidash (5.5)
    :       Idle_Max <= 9.964572e+007:
    :       :...X2Grammed_APICalls_.213 > 0: fakemart (2.4)
    :           X2Grammed_APICalls_.213 <= 0:
    :           :...Total_Length_of_Fwd_Packets > 32747: pletor (2.4)
    :               Total_Length_of_Fwd_Packets <= 32747:
    :               :...X2Grammed_APICalls_.357 > 0: zsone (5)
    :                   X2Grammed_APICalls_.357 <= 0:
    :                   :...X2Grammed_APICalls_.14 > 0: zsone (2.4/0.5)
    :                       X2Grammed_APICalls_.14 <= 0:
    :                       :...X2Grammed_APICalls_.1 > 0: mazarbot (3.6)
    :                           X2Grammed_APICalls_.1 <= 0:
    :                           :...X2Grammed_APICalls_.109 <= 1: jifake (6/3.6)
    :                               X2Grammed_APICalls_.109 > 1: beanbot (4.1/0.5)
```

Gambar 10 Decision tree percobaan keempat

```
-----  Trial 4:  -----

Decision tree:

X2Grammed_APICalls_.53 > 0: charger (8)
X2Grammed_APICalls_.53 <= 0:
:...Bwd_Packets.s > 31106:
    :...X2Grammed_APICalls_.7 > 0: feiwo (6.7/2)
    :   X2Grammed_APICalls_.7 <= 0:
    :   :...X2Grammed_APICalls_.109 > 0:
    :       :...X2Grammed_APICalls_.343 > 0: shuanet (3.1)
    :       :   X2Grammed_APICalls_.343 <= 0:
    :       :   :...X2Grammed_APICalls_.50 > 17: charger (2.9/1.3)
    :       :       X2Grammed_APICalls_.50 <= 17:
    :       :       :...X2Grammed_APICalls_.331 > 4: avpass (4.6)
    :       :           X2Grammed_APICalls_.331 <= 4:
    :       :           :...Fwd_PSH_Flags <= 2: virusshield (2.6)
    :       :               Fwd_PSH_Flags > 2:
    :       :               :...Idle_Std > 1.987329e+007: nandrobox (6/0.4)
    :       :                   Idle_Std <= 1.987329e+007: [S1]
```

Gambar 11.Decision tree percobaan kelima

```
-----  Trial 5:  -----

Decision tree:

PSH_Flag_Count <= 5:
:...X2Grammed_APICalls_.138 > 28: fakeinst (2.2)
:   X2Grammed_APICalls_.138 <= 28:
:   :...X2Grammed_APICalls_.13 > 0: zsone (3.6)
:       X2Grammed_APICalls_.13 <= 0:
:       :...X2Grammed_APICalls_.74 > 0: penetho (2.3/0.6)
:           X2Grammed_APICalls_.74 <= 0:
:           :...X2Grammed_APICalls_.108 > 0: beanbot (2.5)
:               X2Grammed_APICalls_.108 <= 0:
:               :...Bwd_IAT_Mean <= 2504950: beanbot (3.6/1.2)
:                   Bwd_IAT_Mean > 2504950:
:                   :...Fwd_IAT_Min <= 1.78557e+007: jifake (8.9/2.7)
:                       Fwd_IAT_Min > 1.78557e+007: mazarbot (3.7/0.3)
```

Gambar 12.Decision tree percobaan keenam

```
-----  Trial 6:  -----

Decision tree:

PSH_Flag_Count <= 5:
:...X2Grammed_APICalls_.213 > 0: fakemart (5.1)
:   X2Grammed_APICalls_.213 <= 0:
:   :...act_data_pkt_fwd > 22: pletor (6.4/1.4)
:       act_data_pkt_fwd <= 22:
:       :...X2Grammed_APICalls_.142 > 0: mazarbot (4.9)
:           X2Grammed_APICalls_.142 <= 0:
:           :...URG_Flag_Count > 2: mazarbot (3.1/1.2)
:               URG_Flag_Count <= 2:
:               :...X2Grammed_APICalls_.14 > 0: zsone (4)
:                   X2Grammed_APICalls_.14 <= 0:
:                   :...Flow_Bytes.s <= 984963: fakeinst (2.6/0.8)
:                       Flow_Bytes.s > 984963:
:                       :...X2Grammed_APICalls_.109 <= 3: jifake (5.4/0.5)
:                           X2Grammed_APICalls_.109 > 3: beanbot (2.1)
```

Gambar 13.Decision tree percobaan ketujuh

```
-----  Trial 7:  -----

Decision tree:

PSH_Flag_Count <= 5:
:...X2Grammed_APICalls_.408 > 1: penetho (2.6)
:   X2Grammed_APICalls_.408 <= 1:
:   :...X2Grammed_APICalls_.213 > 0: fakemart (4.1)
:       X2Grammed_APICalls_.213 <= 0:
:       :...Total_Length_of_Fwd_Packets > 32747: pletor (4.1)
:           Total_Length_of_Fwd_Packets <= 32747:
:           :...X2Grammed_APICalls_.108 > 0: beanbot (4.5)
:               X2Grammed_APICalls_.108 <= 0:
:               :...Min_Packet_Length <= 61: beanbot (2.9/1.4)
:                   Min_Packet_Length > 61:
:                   :...X2Grammed_APICalls_.14 > 0: zsone (3.3)
:                       X2Grammed_APICalls_.14 <= 0:
:                       :...X2Grammed_APICalls_.91 > 0: zsone (2.3/0.8)
:                           X2Grammed_APICalls_.91 <= 0:
:                           :...X2Grammed_APICalls_.24 <= 3: mazarbot (5.8/0.4)
:                               X2Grammed_APICalls_.24 > 3: jifake (2.8)
```

Gambar 14.Decision tree percobaan kedelapan

```
-----  Trial 8:  -----

Decision tree:

X2Grammed_APICalls_.416 > 1: AndroidDefender (16.4/0.6)
X2Grammed_APICalls_.416 <= 1:
:...PSH_Flag_Count <= 5:
    :...X2Grammed_APICalls_.138 > 28: fakeinst (2.5)
    :   X2Grammed_APICalls_.138 <= 28:
    :   :...X2Grammed_APICalls_.408 > 1: penetho (2.1)
    :       X2Grammed_APICalls_.408 <= 1:
    :       :...X2Grammed_APICalls_.213 > 0: fakemart (3.2)
    :           X2Grammed_APICalls_.213 <= 0:
    :           :...act_data_pkt_fwd > 40: pletor (3.2)
    :               act_data_pkt_fwd <= 40:
    :               :...X2Grammed_APICalls_.108 > 0: beanbot (3.5)
    :                   X2Grammed_APICalls_.108 <= 0:
    :                   :...X2Grammed_APICalls_.14 > 0: zsone (2.6)
    :                       X2Grammed_APICalls_.14 <= 0:
    :                       :...Idle_Std <= 3759739: jifake (8.4/1.7)
    :                           Idle_Std > 3759739: mazarbot (4.9/1.1)
```

Gambar 15. Decision tree percobaan kesembilan

```
-----  Trial 9:  -----

Decision tree:

X2Grammed_APICalls_.416 > 1: AndroidDefender (14.2/1.8)
X2Grammed_APICalls_.416 <= 1:
:...X2Grammed_APICalls_.411 > 0: fakejoboffer (9.6/1.5)
    X2Grammed_APICalls_.411 <= 0:
    :...PSH_Flag_Count <= 4:
        :...X2Grammed_APICalls_.213 > 0: fakemart (2.5)
        :   X2Grammed_APICalls_.213 <= 0:
        :   :...X2Grammed_APICalls_.357 > 0: zsone (4.1/0.8)
        :       X2Grammed_APICalls_.357 <= 0:
        :       :...act_data_pkt_fwd > 40: pletor (2.5)
        :           act_data_pkt_fwd <= 40:
        :           :...X2Grammed_APICalls_.142 > 0: mazarbot (3.7)
        :               X2Grammed_APICalls_.142 <= 0:
        :               :...X2Grammed_APICalls_.109 > 1: beanbot (7.7/2.5)
        :                   X2Grammed_APICalls_.109 <= 1:
        :                   :...X2Grammed_APICalls_.24 <= 0: fakeinst (2.9/0.9)
        :                       X2Grammed_APICalls_.24 > 0: jifake (3.8)
```

Gambar 16.Decision tree percobaan kesepuluh

Hasil dari 10x percobaan tersebut adalah sebuah daftar atribut yang paling berpengaruh dan terdapat 129 atribut dari total 918 atribut. Untuk atribut tersebut dapat dilihat pada tabel 5.1 dan untuk atribut yang tidak masuk kedalam daftar berarti tidak berpengaruh dan dapat dihapus.

Tabel 2.Daftar attribute usage

No	Persentase	Daftar *attribute*
1.	100.00%	X2Grammed_APICalls_.7
2.	100.00%	X2Grammed_APICalls_.53
3.	100.00%	X2Grammed_APICalls_.416
4.	100.00%	Min_Packet_Length
5.	100.00%	PSH_Flag_Count
6.	100.00%	Down.Up_Ratio
7.	97.30%	Bwd_Packets.s
8.	94.05%	X2Grammed_APICalls_.142
9.	94.05%	X2Grammed_APICalls_.411
10.	94.05%	Fwd_Packets.s
11.	82.70%	X2Grammed_APICalls_.357
12.	81.08%	X2Grammed_APICalls_.41
13.	81.08%	X2Grammed_APICalls_.50
14.	81.08%	X2Grammed_APICalls_.329
15.	80.00%	X2Grammed_APICalls_.30
16.	75.68%	X2Grammed_APICalls_.301
17.	71.89%	X2Grammed_APICalls_.59
18.	66.49%	X2Grammed_APICalls_.263
19.	65.41%	X2Grammed_APICalls_.125
20.	64.32%	X2Grammed_APICalls_.342
21.	63.78%	X2Grammed_APICalls_.364
22.	61.62%	Fwd_PSH_Flags
23.	57.84%	Total_Length_of_Fwd_Packets
24.	57.84%	Idle_Min
25.	55.68%	X2Grammed_APICalls_.109
26.	55.14%	X2Grammed_APICalls_.217
27.	53.51%	X2Grammed_APICalls_.796
28.	52.97%	Fwd_IAT_Min

29.	52.43%	X2Grammed_APICalls_.90
30.	49.19%	X2Grammed_APICalls_.1
31.	47.57%	X2Grammed_APICalls_.255
32.	44.32%	X2Grammed_APICalls_.532
33.	40.00%	act_data_pkt_fwd
34.	38.92%	X2Grammed_APICalls_.48
35.	35.68%	X2Grammed_APICalls_.160
36.	35.68%	Idle_Std
37.	35.14%	X2Grammed_APICalls_.408
38.	34.05%	X2Grammed_APICalls_.4
39.	34.05%	X2Grammed_APICalls_.89
40.	33.51%	X2Grammed_APICalls_.331
41.	31.89%	X2Grammed_APICalls_.14
42.	31.89%	X2Grammed_APICalls_.38
43.	31.35%	X2Grammed_APICalls_.116
44.	31.35%	Active_Std
45.	29.19%	X2Grammed_APICalls_.138
46.	28.65%	X2Grammed_APICalls_.22
47.	28.65%	X2Grammed_APICalls_.34
48.	28.11%	X2Grammed_APICalls_.111
49.	27.57%	X2Grammed_APICalls_.5
50.	27.57%	X2Grammed_APICalls_.35
51.	27.03%	X2Grammed_APICalls_.86
52.	25.95%	X2Grammed_APICalls_.193
53.	25.95%	X2Grammed_APICalls_.222
54.	25.95%	X2Grammed_APICalls_.353
55.	25.41%	X2Grammed_APICalls_.121
56.	25.41%	X2Grammed_APICalls_.296
57.	25.41%	X2Grammed_APICalls_.344
58.	24.86%	X2Grammed_APICalls_.240
59.	24.86%	X2Grammed_APICalls_.283
60.	24.86%	X2Grammed_APICalls_.303
61.	23.78%	X2Grammed_APICalls_.13
62.	23.78%	X2Grammed_APICalls_.363
63.	23.24%	X2Grammed_APICalls_.99
64.	23.24%	X2Grammed_APICalls_.320
65.	22.70%	X2Grammed_APICalls_.2
66.	22.16%	X2Grammed_APICalls_.12
67.	22.16%	Flow_Packets.s
68.	21.62%	X2Grammed_APICalls_.24
69.	21.62%	X2Grammed_APICalls_.66
70.	21.62%	X2Grammed_APICalls_.309
71.	21.62%	Flow_Bytes.s
72.	21.62%	Init_Win_bytes_backward
73.	21.08%	X2Grammed_APICalls_.213
74.	21.08%	Fwd_Packet_Length_Min
75.	21.08%	Idle_Max
76.	20.54%	X2Grammed_APICalls_.56
77.	20.00%	X2Grammed_APICalls_.264
78.	19.46%	X2Grammed_APICalls_.101
79.	19.46%	X2Grammed_APICalls_.108
80.	19.46%	X2Grammed_APICalls_.343
81.	18.92%	X2Grammed_APICalls_.265
82.	18.38%	X2Grammed_APICalls_.77
83.	17.30%	X2Grammed_APICalls_.0

84.	17.30%	X2Grammed_APICalls_.74
85.	17.30%	X2Grammed_APICalls_.170
86.	16.22%	X2Grammed_APICalls_.225
87.	15.68%	X2Grammed_APICalls_.3
88.	15.68%	X2Grammed_APICalls_.211
89.	15.14%	X2Grammed_APICalls_.28
90.	14.59%	X2Grammed_APICalls_.17
91.	14.59%	X2Grammed_APICalls_.40
92.	14.59%	X2Grammed_APICalls_.192
93.	13.51%	X2Grammed_APICalls_.8
94.	12.97%	X2Grammed_APICalls_.18
95.	12.97%	X2Grammed_APICalls_.165
96.	12.97%	X2Grammed_APICalls_.197
97.	12.97%	X2Grammed_APICalls_.293
98.	12.97%	URG_Flag_Count
99.	12.43%	Bwd_IAT_Mean
100.	11.89%	X2Grammed_APICalls_.311
101.	10.27%	X2Grammed_APICalls_.221
102.	10.27%	Flow_Duration
103.	9.73%	ACK_Flag_Count
104.	9.19%	X2Grammed_APICalls_.36
105.	9.19%	X2Grammed_APICalls_.393
106.	9.19%	X2Grammed_APICalls_.524
107.	8.65%	X2Grammed_APICalls_.19
108.	8.65%	X2Grammed_APICalls_.64
109.	8.65%	X2Grammed_APICalls_.84
110.	8.11%	X2Grammed_APICalls_.281
111.	7.57%	X2Grammed_APICalls_.137
112.	7.57%	X2Grammed_APICalls_.209
113.	7.57%	X2Grammed_APICalls_.220
114.	7.03%	X2Grammed_APICalls_.43
115.	7.03%	X2Grammed_APICalls_.91
116.	7.03%	X2Grammed_APICalls_.230
117.	7.03%	X2Grammed_APICalls_.328
118.	7.03%	Bwd_IAT_Total
119.	6.49%	X2Grammed_APICalls_.171
120.	5.95%	X2Grammed_APICalls_.51
121.	5.95%	X2Grammed_APICalls_.87
122.	5.95%	X2Grammed_APICalls_.205
123.	5.95%	X2Grammed_APICalls_.368
124.	5.41%	X2Grammed_APICalls_.212
125.	4.32%	X2Grammed_APICalls_.39
126.	4.32%	X2Grammed_APICalls_.229
127.	4.32%	Fwd_IAT_Total
128.	3.78%	X2Grammed_APICalls_.275
129.	3.78%	X2Grammed_APICalls_.663

5.1.3 Klasifikasi menggunakan CNN

Setelah proses *preprocessing* selesai dilakukan, maka lanjut ke proses klasifikasi menggunakan metode CNN. Proses klasifikasi dilakukan menggunakan bahasa pemrograman *python.* Berikut ini *source code* untuk melakukan klasifikasi menggunakan metode CNN.

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import sklearn
```

Gambar 17.Source code import library

Pada gambar 5.14 terdapat *source code* "import matplotlib.pyplot as plt" yang memiliki fungsi untuk membuat diagram dengan di-inisialisasi menjadi plt. *Source code* "import pandas as pd" adalah berfungsi sebagai membaca file masukan dan di-inisialisasi menjadi pd. Untuk *source code* "import numpy as np" berguna sebagai pendukung untuk *array* multi dimensi dan matriks. Sedangkan *source code* "import sklearn" berfungsi untuk mengimpor beberapa proses perintah pada *machine learning* agar dapat melakukan proses. Setelah *import library* selesai maka lanjut ke proses untuk memasukkan dataset yang akan dilakukan klasifikasi seperti pada gambar 5.15.

```
train = pd.read_csv("malwaredataset.csv", delimiter=";")
```

Gambar 18.Source code import file

Dataset dimasukkan kedalam variable *train* menggunakan *library pandas* dan diberi *delimiter* ";" yang berguna untuk memisahkan antar atribut dalam 1 csv. Proses selanjutnya yaitu dilakukan "*standard scaler*" yang berfungsi untuk menghapus *mean* dan memberikan skala kepada tiap unit varian seperti pada gambar 5.16. Proses selanjutnya setelah dilakukan "*standard scaler*" adalah melakukan "*Label Encoder*" menggunakan *source code* pada gambar 5.17.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# extract numerical attributes and scale it to have zero mean and unit variance
cols = train.select_dtypes(include=['float64','int64']).columns

sc_train = scaler.fit_transform(train.select_dtypes(include=['float64','int64']))

# turn the result back to a dataframe
sc_traindf = pd.DataFrame(sc_train, columns = cols)
```

Gambar 19.Proses standard scaler

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

# extract categorical attributes from both training and test sets
classtrain = train.select_dtypes(include=['object']).copy()

# encode the categorical attributes
trainclass = classtrain.apply(encoder.fit_transform)
```

Gambar 20.Proses Label Encoder

Label Encoder berfungsi untuk mengubah label menjadi numerik agar dapat mudah dipahami dan dapat diproses oleh *machine learning*. Pada *source code* gambar 5.17, proses *encoder* yang berhasil dilakukan dimasukkan kedalam variable *trainclass* yang nantinya akan dilakukan proses *split data* pada gambar 5.21.

```
train_x = pd.concat([sc_traindf,trainclass],axis=1)
train_y = train['Family']
train_x.shape
```

Gambar 21.Proses concat, train dan mengecek shape

Tiap baris *source code* pada gambar 5.18 memiliki fungsi masing-masing. Untuk "train_x = pd.concat([sc_traindf,trainclass],axis=1)" berfungsi untuk melakukan penggabungan antara variable "sc_traindf" yang diinisialisasi pada gambar 5.16 dan "trainclass" pada gambar 5.17 dimasukkan kedalam *axis* 1, seluruh proses tersebut dimasukkan kedalam variable "train_x". Baris kedua memiliki fungsi melakukan *train* pada atribut *Family* dan dimasukkan kedalam variabel "train_y". "train_x.shape" berguna untuk melihat dimensi pada *array* yang digunakan.

```
train_x=train_x.drop(["Family",0], axis=1)
```

Gambar 22. Proses drop atribut yang tidak digunakan

Setelah proses pada gambar 5.18 selesai dilakukan, pasti dataset akan bertambah 1 atribut bernama 0, maka proses pada gambar 5.19 diperlukan untuk membuang atribut 0 dan *"Family"* karena tidak diperlukan. Tetapi untuk varibel "train_y" masih dalam bentuk *dataframe* sehingga perlu diubah menjadi sebuah *array* seperti pada gambar 5.20 agar dapat dilakukan proses *split* data.

```
train_y=train_y.to_numpy()
```

Gambar 23. Proses merubah variable "train_y" ke array

Proses split data dapat dilakukan menggunakan *library* dari "sklearn" dengan *source code* seperti pada gambar 5.21 dan 5.22.

```
from sklearn.model_selection import train_test_split
X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(train_x, train_y, test_size=0.2, random_state=30
```

Gambar 24.Proses split data 1

```
X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test, Y_val_and_test, test_size=0.2, shuffle=False)
```

Gambar 25Proses split data 2

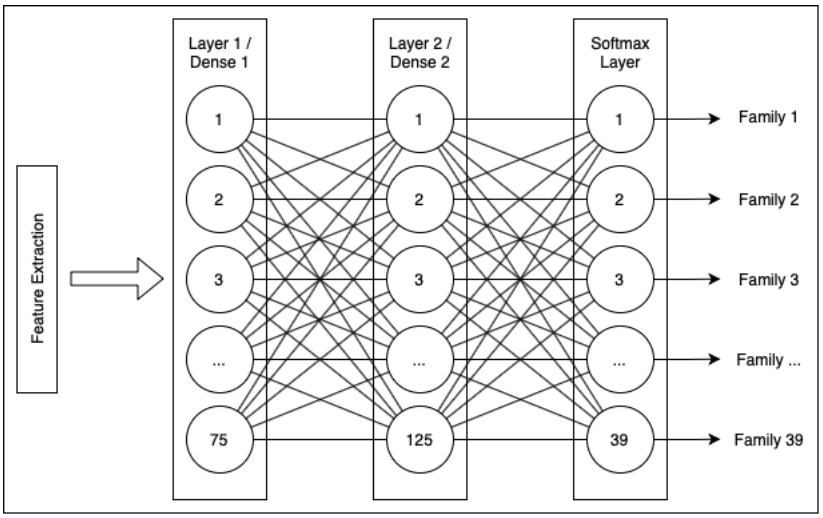
Proses *split* data dilakukan 2x karena untuk proses *split* yang kedua diperlukan hasil dari proses *split* yang pertama, pada gambar 5.21 dilakukan *split* data *train* 80% dan data *test* 20% dan kemudian dimasukkan kedalam variable "X_train, X_val_and_test, Y_train, Y_val_and_test". Sedangkan pada proses *split* yang kedua adalah data yang displit hasil dari proses *split* pertama yaitu variabel "X_val_and_test" dan "Y_val_and_test" untuk dimasukkan kedalam variabel "X_val, X_test, Y_val, Y_Test" sebanyak 20% untuk data *test* dan 80% untuk data *train* lalu untuk "shuffle=False" berguna agar data yang displit tidak *random*.

Untuk melakukan proses CNN yang diperlukan adalah *library* yang harus diimpor sesuai dengan gambar 5.23 seperti *library* "from keras.models import Sequential" yang berfungsi untuk menyiapkan tempat untuk daftar *hidden layer*. Selanjutnya adalah "from keras.layers import Dense, Dropout, Flatten, MaxPooling1D" berfungsi untuk membuat *hidden layer* yang akan digunakan untuk melakukan proses klasifikasi. *Library* "from keras.utils import np_utils" berfungsi untuk mengkonversikan *array* untuk diproses ke proses selanjutnya. Untuk *library* "import h5py" berfungsi untuk membaca data biner dan memungkinkan untuk melakukan manipulasi pada dataset. *Code* "from keras.optimizer import adam" adalah suatu *code* untuk menggunakan optimasi dari Adam melalui *library* keras. "model = Sequential()" adalah code untuk menginisialisasi bahwa model yang akan digunakan adalah *sequential* dan menambahkan hidden layer sebanyak 3, yaitu seperti skema CNN pada gambar 5.24. untuk *activation* yang dapat digunakan sesuai dengan data ini adalah *sigmoid* dikarenakan untuk klasifikasi biner pada model regresi. Sedangkan untuk *activation softmax* digunakan di akhir karena digunakan untuk multi klasifikasi yang totalnya ada 39.

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, MaxPooling1D
from keras.utils import np_utils
import h5py
from keras.optimizers import Adam

model = Sequential()
model.add(Dense(75, input_shape=(131,), activation='sigmoid'))
model.add(Dense(125, activation='sigmoid'))
model.add(Dense(39 ,activation='softmax'))
model.summary()
```

Gambar 26. Code layering CNN



Gambar 27. Gambar 5.24 Skema CNN

Setelah proses pembuatan *hidden layer* untuk melakukan proses klasifikasi, selanjutnya adalah menggunakan optimasi dengan code seperti pada gambar 5.25 dengan *learning rate* sebesar 0.01 atau 1e-2.

```
opt=Adam(learning_rate=1e-2)
```

Gambar 28. Code optimizer learning rate

Proses klasifikasi CNN mulai dilakukan seperti pada gambar 5.26, tiap *line* memiliki kegunaan masing-masing seperti pada line 1 yang berguna untuk melakukan *compile* dengan menggunakan "sparse_categorical_crossentropy" dikarenakan label yang digunakan lebih dari 2, menggunakan *optimizer* yang sudah diinisialisasi pada gambar 5.25 dan *metrics* yang dicari adalah "*accuracy"*. Sedangkan untuk *line* 2 berfungsi untuk menampilkan model yang telah di*compile* pada *line* 1 menggunakan *verbose 2* agar yang ditampilkan tidak terlalu banyak dan dilakukan sebanyak 50 kali pada perintah *epochs.*

```
model.compile(loss='sparse_categorical_crossentropy', optimizer=opt, metrics = ['accuracy'])
hist = model.fit(X_train,Y_train,validation_data=(X_val, Y_val),verbose=2,epochs=50)
```

Gambar 29.Code klasifikasi CNN

Hasil dari klasifikasi pada gambar 5.26 memiliki akurasi yang bagus dan tergantung dari *hidden layer* yang kita gunakan seperti pada gambar 5.23. Untuk hasil klasifikasinya seperti pada gambar 5.27

```
 - 0s - loss: 0.9486 - accuracy: 0.7664 - val_loss: 1.5362 - val_accuracy: 0.4583
Epoch 13/50
 - 0s - loss: 0.8282 - accuracy: 0.7787 - val_loss: 1.4290 - val_accuracy: 0.5417
Epoch 14/50
 - 0s - loss: 0.7288 - accuracy: 0.7828 - val_loss: 1.4539 - val_accuracy: 0.5417
Epoch 15/50
 - 0s - loss: 0.6548 - accuracy: 0.8074 - val_loss: 1.3971 - val_accuracy: 0.5000
Epoch 16/50
 - 0s - loss: 0.5786 - accuracy: 0.8648 - val_loss: 1.3106 - val_accuracy: 0.6250
Epoch 17/50
 - 0s - loss: 0.5258 - accuracy: 0.8484 - val_loss: 1.4355 - val_accuracy: 0.5208
Epoch 18/50
 - 0s - loss: 0.4715 - accuracy: 0.8648 - val_loss: 1.3582 - val_accuracy: 0.6667
Epoch 19/50
 - 0s - loss: 0.4478 - accuracy: 0.8934 - val_loss: 1.3801 - val_accuracy: 0.6042
Epoch 20/50
 - 0s - loss: 0.3964 - accuracy: 0.8811 - val_loss: 1.3628 - val_accuracy: 0.6042
Epoch 21/50
 - 0s - loss: 0.3671 - accuracy: 0.9139 - val_loss: 1.3765 - val_accuracy: 0.6042
```

Gambar 30.Hasil klasifikasi

Untuk mengetahui seberapa besar *loss* dan akurasi dari proses klasifikasi pada gambar 5.26 yaitu dengan menggunakan *source code* pada gambar 5.28 dan akan keluar hasilnya seperti pada gambar 5.29.

```
test_eval = model.evaluate(X_test, Y_test)
print('Test loss:', test_eval[0])
print('Test accuracy:', test_eval[1])
```

Gambar 31.Source code evaluate

```
13/13 [==============================] - 0s 297us/step
Test loss: 2.7607812881469727
Test accuracy: 0.692307710647583
```

Gambar 32.Hasil evaluate

Tetapi untuk mengetahui hasil yang lebih detail, bisa seperti *source code* pada gambar 5.30 menggunakan *code* “from sklearn.metrics import classification_report” dan menginisialisasi variabel yang akan dilakukan prediksi seperti *line* 2 diambil dari variabel “X_test” sehingga hasilnya akan seperti pada gambar 5.31.

```
from sklearn.metrics import classification_report
predicted_cnn = model.predict_classes(X_test)
print(classification_report(Y_test, predicted_cnn))
```

Gambar 33.Source code classification report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1
2	0.50	1.00	0.67	1
3	1.00	1.00	1.00	1
4	0.00	0.00	0.00	0
7	1.00	1.00	1.00	2
8	0.00	0.00	0.00	1
13	1.00	1.00	1.00	1
15	1.00	1.00	1.00	1
18	0.00	0.00	0.00	0
20	0.00	0.00	0.00	0
22	0.00	0.00	0.00	1
25	1.00	1.00	1.00	2
27	0.00	0.00	0.00	1
30	0.00	0.00	0.00	1
accuracy			0.69	13
macro avg	0.46	0.50	0.48	13
weighted avg	0.65	0.69	0.67	13

Gambar 34. Hasil dari classification report

Dataset:	**Evaluation (Testing set)**				**Evaluation (Testing set)**		
Features:	**Network_Flows**				**Permissions+Intents**	**API_Calls+Network_Flows**	
Scenario:	**A (Malware Binary)**	**B (Malware Category)**	**C (Malware Family)**		**A (Malware Binary)**	**B (Malware Category)**	**C (Malware Family)**
Algorithm:	**RF**	**RF**	**RF**		**RF**	**RF**	**RF**
Precision(%):	85.80	49.90	27.50		95.30	83.30	59.70
Recall(%):	88.30	48.50	25.50		95.30	81.00	61.20
	First part of the CICAndMal2017 [16] dataset				**Second part of the CICAndMal2017 dataset**		

Gambar 35.Hasil dari Penelitian sebelumnya

Pada penelitian sebelumnya mendapatkan hasil seperti pada gambar 5.32, jika dilihat hasil akurasi pada malware family lebih baik penelitian yang saat ini, tetapi setelah ditelaah oleh peneliti, hal tersebut tidak sebanding, dikarenakan terdapat preprocessing ulang yang dilakukan oleh peneliti, oleh karena itu perlu dilakukan kaji ulang terhadap data yang digunakan sehingga bisa sebanding jika dibandingkan dengan penelitian sebelumnya.

BAB 6 KESIMPULAN DAN SARAN

Hasil penelitian yang dilakukan jika dibandingkan dengan penelitian sebelumnya sedikit ada perbaikan, tetapi hal tersebut perlu dikaji ulang dikarenakan data yang digunakan pada penelitian lanjutan telah melalui pre-processing ulang, sehingga bisa mendapatkan hasil seperti pada bab hasil dan pembahasan, Pada penelitian sebelumnya mendapatkan hasil akurasi lebih kecil jika dibandingkan dengan penelitian saat ini, setelah ditelaah oleh peneliti, hal tersebut tidak sebanding, dikarenakan terdapat preprocessing ulang yang dilakukan oleh peneliti, oleh karena itu perlu dilakukan kaji ulang terhadap data yang digunakan sehingga bisa sebanding jika dibandingkan dengan penelitian sebelumnya.

DAFTAR PUSTAKA

[1] C. Da, H. Zhang, and X. Zhang, "Detection of Android malware security on system calls," *Proc. 2016 IEEE Adv. Inf. Manag. Commun. Electron. Autom. Control Conf. IMCEC 2016*, pp. 974–978, 2017.

[2] M. Jaiswal, Y. Malik, and F. Jaafar, "Android gaming malware detection using system call analysis," *6th Int. Symp. Digit. Forensic Secur. ISDFS 2018 - Proceeding*, vol. 2018-Janua, pp. 1–5, 2018.

[3] Faruki, Parvez, et al. "Android security: a survey of issues, malware penetration, and defenses." *IEEE communications surveys & tutorials* 17.2 (2014): 998-1022.

[4] Black, Paul, Iqbal Gondal, and Robert Layton. "A survey of similarities in banking malware behaviours." *Computers & Security* 77 (2018): 756-772

[5] Yu, Bo, et al. "A survey of malware behavior description and analysis." *Frontiers of Information Technology & Electronic Engineering* 19.5 (2018): 583-603.

[6] Lin, Chih-Ta, et al. "Feature Selection and Extraction for Malware Classification." *J. Inf. Sci. Eng.* 31.3 (2015): 965-992.

[7] Akbi, Denar Regata, Vinna Rahmayanti Setyaning Nastiti, and Achmad Rizal Yogaswara. "SELEKSI FITUR MALWARE FAMILY MENGGUNAKAN METODE C5. 0." *Prosiding SENTRA (Seminar Teknologi dan Rekayasa)*. No. 6. 2021.

[8] Zhang, Jason. "Machine learning with feature selection using principal component analysis for malware detection: a case study." *arXiv preprint arXiv:1902.03639* (2019).

[9] Taheri, Laya, Andi Fitriah Abdul Kadir, and Arash Habibi Lashkari. "Extensible android malware detection and family classification using network-flows and API-calls." *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019.

[10] Lashkari, Arash Habibi, et al. "Toward developing a systematic approach to generate benchmark android malware datasets and classification." *2018 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2018.

[11] Murtaz, Muhammad, et al. "A framework for Android Malware detection and classification." *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*. IEEE, 2018

[12] Noorbehbahani, Fakhroddin, Farzaneh Rasouli, and Mohammad Saberi. "Analysis of machine learning techniques for ransomware detection." *2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*. IEEE, 2019.

[13] C. Hasegawa and H. Iyatomi, "One-dimensional convolutional neural networks for Android malware detection," *Proc. - 2018 IEEE 14th Int. Colloq. Signal Process. its Appl. CSPA 2018*, no. March, pp. 99–102, 2018.

[14] Hidaka, Akinori, and Takio Kurita. "Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks." *Proceedings of the ISCIE international symposium on stochastic systems theory and its applications*. Vol. 2017. The ISCIE Symposium on Stochastic Systems Theory and Its Applications, 2017.

UNIVERSITAS MUHAMMADIYAH MALANG

Fakultas Agama Islam - Fakultas Ilmu Sosial dan Ilmu Politik - Fakultas Hukum - Fakultas Psikologi - Fakultas Teknik
Fakultas Keguruan dan Ilmu Pendidikan - Fakultas Ekonomi dan Bisnis - Fakultas Pertanian dan Peternakan
Fakultas Kedokteran - Fakultas Ilmu Kesehatan - Program Pasca Sarjana

Kampus I : Jl. Bandung 1 Telp. 0341-551253 Fax. 0341-562124 Malang 65113
Kampus II : Jl. Bendungan Sutami No. 188A Telp. 0341-552443,551149 (Hunting) Fax. 0341-582060 Malang 65145
Kampus III : Jl. Raya Tlogomas No. 246 Malang Telp.0341-464318-319 Fax. 0341-460435,460782 Malang 65144
E-mail : webmaster@unix.umm.ac.id Website: www.umm.ac.id

SURAT TUGAS

Nomor : E.2.a/ 133 /BAA-UMM/II/2021

1. Pejabat yang memberi tugas	:	Rektor Universitas Muhammadiyah Malang
2. Nama yang diberi tugas	:	Terlampir
3. Jabatan yang diberi tugas	:	Dosen Universitas Muhammadiyah Malang
4. Alamat / kedudukan	:	di Malang
5. Yang bersangkutan diberi tugas	:	Melaksanakan Program Penelitian Internal Skim Penelitian Pengembangan Ipteks (P2I), Penelitian Dasar Keilmuan (PDK), Penelitian Berorientasi Produk (PBP), Program Unggulan Pusat Studi / Lembaga (PUPS/L), Penelitian Peningkatan Kualitas Pembelajaran di Perguruan Tinggi (PPKP-PT), Penelitian Penugasan Khusus Bidang AIK (PPK-AIK) / Bidang Lingkungan (PPK-L), Program Pengembangan Karya Ilmiah Doktor (PKID), dan Program Pengembangan Karya Ilmiah Profesor (PKIP) Periode Tahun Akademik 2020/2021
6. Tugas tersebut dilaksanakan	:	mulai Tanggal 1 Maret 2021 sd. 10 Desember 2021
7. Keterangan lain-lain	:	Tunaikan tugas dengan penuh rasa tanggung jawab sebagai amanah.

Malang, 13 Pebruari 2021
An. Rektor,
Wakil Rektor I,

Prof. Dr. Syamsul Arifin, M.Si

Tembusan :
1. Yth. Rektor (sebagai laporan),
2. Yth. Wakil Rektor I, II, III dan IV
3. Yth Ka.BAKKU
4. Yth. Dekan / Direktur,
5. Arsip.

Lampiran surat tugas: Nomor : E.2.a/ 132 /BAA-UMM/III/2021

No	TIM PENELITI	Anggota Mahasiswa	Fakultas	Judul	Skim	Dana
22.	Ir. Alik Ansyori Alamsyah M.T. (NIDN:0726036402) Ir. H. Hari Eko Meiyanto MT. (NIDN. 0701055601);	Salma Puteri Dwi Alfian (NIM. 201810340311014); Ajeng Darma Yanti (NIM. 201810340311027);	Teknik	Model Korelasi Antara Lama Perendaman Dan Jumlah Tumbukan pada Campuran Perkerasan Atb (Asphalt Treated Base) Terhadap Stabilitas Marshall	PBP	Rp 18.000.000
23.	Dian Palupi Restuputri ST, M.T (NIDN:0727078501) Rahmad Wisnu Wardana S.Pd., M.Eng. (NIDN. 0703049301)	Tri Ratna Indriani (NIM. 201710140311018); Achmad Mahardhika Febriansyah (NIM. 201710140311010);	Teknik	Pengembangan Model Pendekatan Ergonomi : Prevention By Design Sebagai Upaya Pencegah Kecelakaan Kerja (Tahun Ke-2)	PBP	Rp 18.000.000
24.	Yufis Azhar S.Kom., M.Kom. (NIDN:0728088701) Dr. Dwi Anggraini Puspita R S.Kom, M.IT (NIDN. 0726108201)	Fachry Fathurahman (NIM. 201810370311257); Ulfah Nur Oktaviana (NIM. 201810370311261);	Teknik	Optimasi Hasil Prediksi Pembatalan Pemesanan Hotel Dengan Menggunakan Pendekatan Augmentasi Data	PBP	Rp 17.000.000
25.	Agus Eko Minarno S.Kom., M.Kom. (NIDN:0729118203) Hariyady S.Kom, MT. (NIDN. 0717067307)	Moch. Chamdani Mustaqim (NIM. 201710370311285); Kharisma Muzaki Ghufron (NIM. 201710370311079);	Teknik	Klasifikasi Tumor Otak Menggunakan Convolutional Neural Network	PBP	Rp 18.500.000
26.	Denar Regata Akbi S.Kom., M.Kom. (NIDN:0701058601) Vinna Rahmayanti S S.Si., M.Si (NIDN. 0706079004)	Ridhi Pratomo Pramudana (NIM. 201610370311141); Muhammad Zahid Humam (NIM. 201610370311073);	Teknik	Analisis Hasil Obtain dan Scrub Data Malware Menggunakan Metode Klasifikasi untuk Mendapatkan Insight Data	PDK	Rp 15.300.000
27.	Ir. H. Mulyono MT. (NIDN:0701086601) Dini Kurniawati ST., MT (NIDN. 0724018205)	Achmad Rizky Amrullah Kulle (NIM. 201610120311206); Bagus Krisdiantoro (NIM. 201610120311186);	Teknik	Studi Eksperimen Dan Analisa Performansi Pembangkit Listrik Tenaga Gelombang Dengan Daya 500 Watt	PDK	Rp 15.300.000
28.	Fauzi Dwi Setiawan Sumadi ST., M.CompSc. (NIDN:0707069202) Christian Sri Kusuma Aditya S.Kom., M.Kom (NIDN. 0727029101);	Ahmad Akbar Maulana (NIM. 201410370311263); Achmad Irfani Nur Iman (NIM. 201710370311316);	Teknik	DDoS Detection and Mitigation in SDN-Honeypot Using Pseudo-labeling Method	PDK	Rp 16.150.000