# ANALISIS DAN DETEKSI MALWARE DENGAN METODE HYBRID ANALYSIS MENGGUNAKAN FRAMEWORK MOBSF

Edward Tansen, Deris Wahyu Nurdiarto

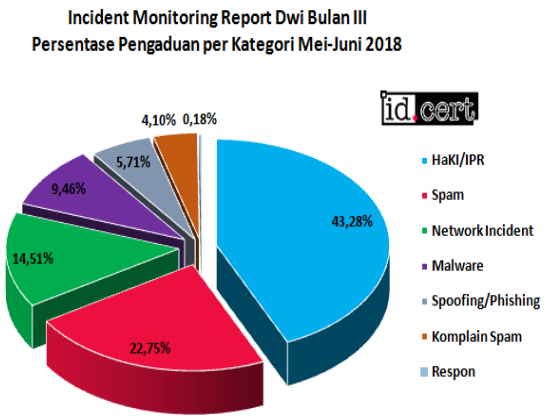
*Program Studi Teknik Komputer, Universitas Amikom Yogyakarta*
Jl. Ring Road Utara, Ngringin, Condongcatur, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta
edward.tansen77@students.amikom.ac.id, deris.n@students.amikom.ac.id

***Abstract*** - The increase in the popularity of smartphones is followed by an increase in the number of users each year. In this case, smartphones with the Android platform are still ranked number one in the percentage of the highest number of users in the world. This fact is also followed by an increasing number of attacks by malicious programs or malware on the Android platform. These rogue application developers take advantage of the loopholes in the Android platform by inserting their malicious programs in the form of source code in Android applications and disseminating them through internet blogs and the Android application market. Ignorance and carelessness in lay users in installing android applications make the main target by malicious application developers. It is crucial to know by users what functions are performed by the Android application, especially in providing permissions or access rights to the Android system. This study, using a sample of Bouncing Golf and Riltok Banking Trojan malware. The study was conducted to know the characteristics and behavior using a combination of static analysis and dynamic analysis, or what is referred to in this study is a hybrid analysis using the MobSF framework. The analysis showed that Bouncing Golf stole information and was able to hijack infected Android devices effectively and Riltok Banking Trojan could take over mobile phones to steal information from credit cards through phishing techniques.
***Keywords*** - Android; Dynamic Analysis; Hybrid Analysis; Malware; Static Analysis.

***Abstrak*** - Peningkatan popularitas *smartphone* diikuti dengan kenaikan jumlah pengguna pada setiap tahunnya. Dalam hal ini, *smartphone* dengan platform *android* masih menjadi urutan nomor satu dalam persentase jumlah pengguna terbanyak di dunia. Fakta ini juga diikuti dengan meningkatnya jumlah serangan program jahat atau *malware* terhadap *platform android*. Para pengembang aplikasi jahat ini memanfaatkan celah yang ada pada platform *android* dengan menyisipkan program jahat mereka dalam bentuk *source code* di dalam aplikasi *android* dan menyebarluaskannya melalui *blog-blog* internet serta pasar aplikasi *android*. Tidak adanya kewaspadaan dan lengahnya pada pengguna awam dalam memasang aplikasi *android* menjadikan target utama oleh pengembang aplikasi jahat. Sangat penting untuk diketahui oleh para pengguna terkait apa saja fungsi yang dilakukan oleh aplikasi *android*, terutama dalam memberikan perizinan atau hak akses terhadap sistem *android*. Dalam penelitian ini menggunakan sampel *malware Bouncing Golf* dan *Riltok Banking Trojan*. Penelitian dilakukan dengan tujuan mengetahui karakteristik dan perilaku dengan menggunakan kombinasi analisis statis dan analisa dinamis, atau yang disebut dalam penelitian ini adalah analisis *hybrid* menggunakan *framework* MobSF. Analisis yang dilakukan menunjukkan bahwa *Bouncing Golf* melakukan pencurian informasi dan dapat secara efektif membajak perangkat *android* yang terinfeksi dan *Riltok Banking Trojan* memiliki kemampuan dalam mengambil alih *smartphone* untuk mencuri informasi dari kartu kredit melalui teknik *phishing*.
***Kata Kunci*** - *Android; Dynamic Analysis; Hybrid Analysis; Malware; Static Analysis.*

## I. PENDAHULUAN

Menganalisa suatu *malware* sudah menjadi hal yang umum dilakukan dan menjadi ilmu dasar dari *digital forensic*. Untuk menjadi seorang investigator, menganalisa suatu perangkat lunak yang berbahaya adalah hal yang menantang dalam setiap melakukan investigasi [1]. Hal ini dikarenakan serangan *malware* yang meningkat dengan pesat dari dari tahun ke tahun.




Gambar 1. Kategori Pengaduan Cybercrime

Gambar 1 menunjukkan hasil penelitian dari ID-CERT yang menjadi laporan Dwi tahun 2018 menemukan pengaduan terkait *cybercrime* berasal dari *malware* sekitar 4.26% di bulan I dan meningkat hingga 9,46% di bulan III [2][3][4]. *Malware* atau yang sering disebut *malicious software* adalah sebuah perangkat lunak yang sengaja diciptakan untuk menyusup ke dalam sistem komputer beserta jaringannya tanpa diketahui oleh pemilik [5]. Tidak hanya merusak sistem secara langsung *malware* juga dapat membuat sistem rusak secara perlahan dengan cara memperbanyak aktivitas dari sistem atau menggandakan dirinya sendiri sehingga sistem akan melambat dan rusak secara perlahan. Saat ini *malware* tidak hanya berupa software saja tapi sudah merambah ke ranah aplikasi dengan sistem operasi *android* [6]. Berdasarkan *statcounter system* operasi *android* pada *smartphone* menembus angka 72.26% dari keseluruhan *market* sistem operasi *mobile* yang ada di dunia [7], hal ini memperkuat alasan mengapa perkembangan *malware* di Indonesia sangat cepat khususnya *malware* yang menyerang *smartphone*.

Pada umumnya dalam mendeteksi penyalahgunaan dan anomali pada sebuah aplikasi menggunakan *static* analisis atau *dynamic* analisis untuk mengekstraksi fitur dari sampel *malware* [8]. Analisis *static* bertujuan untuk menganalisa kode dari sumber atau aplikasi dengan pola mencurigakan tanpa menjalankannya seperti mengurai *file manifest* pada aplikasi *android* [9]. Selain *static*, teknik analisa *dynamic* juga memiliki peran yang sangat penting dalam menganalisa *malware* pada *android* dengan menjalankan aplikasi pada sebuah perangkat virtual (*sandbox*) sehingga dapat menampilkan perilaku dari *malware* yang dijalankan.

Analisa dengan metode *static* lebih menguntungkan karena tidak terlalu banyak menggunakan *resource* akan tetapi sangat rentan terhadap penyamaran kode (*code obfuscation*), yang merupakan teknik umum dengan tujuan untuk mempersulit pembacaan sumber kode. Sebaliknya untuk analisis *dynamic*, *code obfuscation* tidak terlalu berpengaruh karena pembacaan aplikasi melalui aktivitas *runtime*, akan tetapi teknik ini memerlukan lebih banyak sumber daya [10]. Meningkatnya serangan *malware* pada sistem operasi *android* saat ini menjadi topik menarik untuk melakukan penelitian ini [11]. Seperti penguraian di atas tentang teknik analisa *malware* memiliki kelemahan satu sama lain, oleh sebab itu menggabungkan kedua teknik analisa yang sebelumnya diuraikan yaitu teknik *static analysis* dan *dynamic analysis* yang sering disebut *hybrid analysis* menggunakan *framework* MobSF yang kemudian akan dianalisa kembali bagian-bagian mana saja yang dicurigai sebagai *malware* dari aplikasi yang akan peneliti analisa.

## II. TINJUAN PUSTAKA

### *A. Malware*

*Malware* atau *malicious software* adalah sebuah perangkat lunak berbahaya yang digunakan untuk menyusupi komputer orang lain beserta jaringannya tanpa izin dari pemilik [5]. *Malware* terbagi menjadi beberapa bentuk seperti *code,script, active content*, dan perangkat lunak, *malware* juga mampu membuat perangkat keras bekerja lebih keras dari yang seharusnya karena perlu memfasilitasi jalannya program dari *malware* tersebut [12].

### *B. Analisis Static*

Metode statis analis dilakukan tanpa benar-benar menjalankan programnya dan lebih seperti menyelidiki apa yang terjadi pada *source code* dengan tujuan utama yaitu untuk mengetahui kode berbahaya seperti apa yang tertanam dalam aplikasi tersebut [13].

### *C. Analisis Dinamis*

Analisa *malware* secara dinamis dapat dilakukan dengan menjalankan virtual *sandbox* yang aman dan dalam tujuan guna mengetahui perilaku *malware* terhadap sistem seperti penambahan *service*, pengumpulan data, pembukaan *port*, dan penambahan jaringan [13].

### *D. Hybrid Analysis*

Metode *hybrid analysis* adalah sebuah penggabungan dari teknik analisa statis dan teknik analisa dinamis dengan memeriksa *source code* yang diduga sebagai *malware* kemudian melihat perilaku dari *malware* tersebut setelah menginfeksi sistem [14]. Analisa *hybrid* ini dilakukan guna menutupi kekurangan dari kedua teknik tersebut.

### *E. Sistem Operasi Android*

Sistem operasi *android* adalah sistem operasi *mobile open source* berbasis *java* dengan *kernel linux* 2.6 dan berfitur lengkap yang diciptakan untuk memenuhi kebutuhan konsumen. Aplikasi *android* yang di kembangkan dengan *java* lebih fleksibel terhadap platform yang digunakan, hal ini memberikan dorongan besar bagi pasar aplikasi *android*. Selain itu fitur-fitur yang dimiliki oleh *android* lebih menarik sehingga memiliki pertumbuhan yang sangat cepat terhadap pasar global.

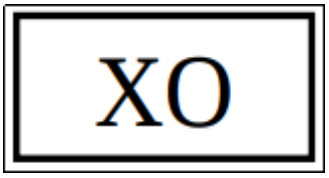
Sistem operasi *android* memiliki keamanan salah satunya penambahan *permission* yang terdapat pada *file* "*Androidmanifest.xml*" di dalam aplikasinya hal ini bertujuan untuk memverifikasi penginstalan aplikasi kepada pengguna dengan memberitahukan *permission* yang diminta oleh aplikasi tersebut [15].

*F. MobSF*

*Mobile Security Framework* (MobSF) adalah *framework* pengujian otomatis bersifat *open-source*, yang mampu melakukan analisis statis dan dinamis dengan sangat mudah [16].

## III. METODOLOGI PENELITIAN

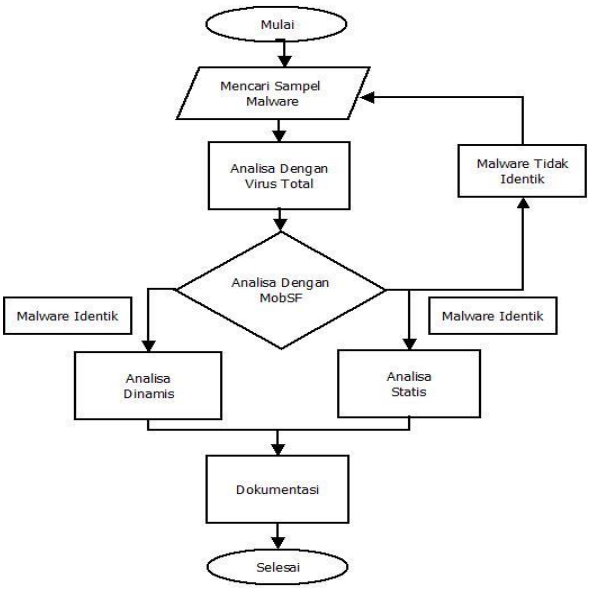
Metodologi yang digunakan pada penelitian ini adalah kuantitatif dengan jenis *pre-experimental design* menggunakan *one-shot case study*. Maksud penelitian *one-shot case study* dalam penelitian ini yaitu tidak adanya kelas kontrol yang digunakan sebagai perbandingan dari kelas eksperimen. Lalu menelusuri lebih jauh hasil yang didapatkan serta melakukan pengukuran pada akhir penelitian. Adapun desain dari *one-shot case study* menurut Sugiyono dalam bukunya (2014:110) dapat digambarkan sebagai berikut :




Gambar 2. One-Shot Case Study

Keterangan :
X = *Treatment*/perlakuan dengan menggunakan *framework* MobSF (Variabel Independen)
O = Observasi terhadap *malware* yang mendapatkan perlakuan (Variabel Dependen)

Dari paragraf di atas didapati sebuah kerangka berpikir yang digunakan pada penelitian ini yaitu untuk mencari hal-hal yang janggal pada *malware* yang menginfeksi perangkat *android*, dalam melakukan penelitian *hybrid analysis*, MobSF berperan penuh terhadap penelitian ini.




Gambar 3 Tahapan Penelitian Hybrid Analysis

Gambar 3 merupakan tahapan-tahapan yang akan dilakukan dalam penelitian ini. Menindaklanjuti hasil penelitian dari ID-CERT dengan meningkatnya pengaduan terkait *cybercrime* berasal dari *malware* sebesar 9,46% pada tahun 2018, maka dalam penelitian ini akan berfokus pada analisa *malware*. Pencarian sampel *malware* dilakukan pada aplikasi yang sudah mengandung *malware* untuk dilakukan analisa selanjutnya. Dalam pemilihan sampel *malware* tersebut, didasarkan pada banyaknya serangan *malware* jenis *adware* dan *trojan* pada sistem operasi *android*. Selanjutnya, sampel dari *malware* yang sudah ditemukan akan dilakukan analisa dengan metode *hybrid analysis*.

*Hybrid analysis* yaitu menggabungkan dari 2 analisa, yaitu *static analysis* dan *dynamic analysis.* Metode analisa statis yang digunakan pada sampel *malware* adalah *reverse engineering*. Sampel *malware* dengan format *file* APK akan dilakukan *reverse engineering* untuk menelusuri sumber kode yang diduga *malware* dan agar dapat mengetahui lebih detail karakteristik dari perizinan yang ada di dalam aplikasi, sehingga dengan kombinasi *dynamic analysis* akan didapatkan hasil dari fungsionalitas dan perilaku terkait tingkat akurasi pendeteksian *malware*.

Dokumentasi akan menyimpan hasil dari setiap analisa yang dilakukan terhadap sampel aplikasi *malware*. Berbagai jenis *tools* yang digunakan seperti, MobSF dan *scan* sampel *malware VirusTotal* akan mendukung dalam proses pengumpulan informasi, dan kemudian akan disajikan dalam laporan penelitian.

## IV. HASIL DAN PEMBAHASAN

*A. Preparation Analysis*

*Malware* yang digunakan dalam objek penelitian adalah *Bouncing Golf* dan *Riltok*. Kedua *malware* tersebut dapat dilihat dalam Tabel 1 dan 2. Informasi lebih lengkap mengenai *malware* yang akan digunakan dalam penelitian sebagai berikut :

Tabel 1. Informasi malware bouncing golf (sumber: virustotal)

| No | Malware Bouncing Golf | |
|---|---|---|
| 1 | SHA256 | 55123ed4982fa135dbeda49969ab68444125143e36930fe1612d367f2fa615fc |
| 2 | Rasio Deteksi | 24/61 |
| 3 | Waktu Analisis | 2020-06-03 06:25:12 UTC |
| 4 | Ukuuran File | 14.07 MB |
| 5 | Tipe File | APK |

Informasi diperoleh dari *VirusTotal* yaitu sampel *malware Bouncing Golf* memiliki nilai *checksum* SHA256 atau *Secure Hash Algorithm* 55123ed4982fa135dbeda49969ab68444125143e36930fe1612d367f2fa615fc. Deteksi dengan rasio dari 61 *antimalware* dan hasil 24 *antimalware* dapat mendeteksi sampel

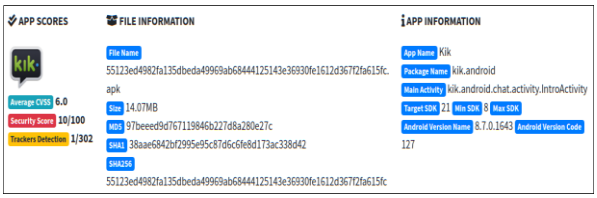
*malware Bouncing Golf*. *File* sampel *malware* berukuran 14.07 MB dan tipe *file* APK. *File* APK merupakan tipe *file* yang umum digunakan pada sistem operasi *android* sebagai *file* distribusi dan pemasangan perangkat lunak atau aplikasi pada sistem *android*.

Tabel 2. Informasi malware riltok (sumber: virustotal)

| No | Malware Riltok | |
|---|---|---|
| 1 | SHA256 | 0497b6000a7a23e9e9b97 472bc2d3799caf49cbbea1 627ad4 d87ae6e0b7e2a98 |
| 2 | Rasio Deteksi | 40/61 |
| 3 | Waktu Analisis | 2019-09-27 09:36:56 UTC |
| 4 | Ukuuran File | 992.55 KB |
| 5 | Tipe File | APK |

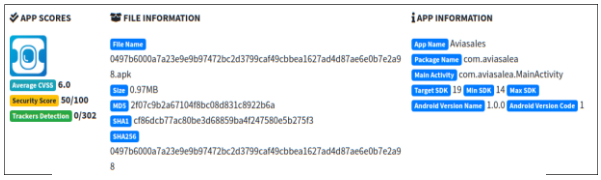
Informasi didapatkan dari *VirusTotal* yaitu sampel *malware Riltok* memiliki nilai *checksum* SHA256 atau *Secure Hash Algorithm* 0497b6000a7a23e9e9b9747 2bc2d3799caf49cbbe a1627ad4d87ae6e0b7e2a98. Deteksi dengan rasio dari 61 *antimalware* dan hasil 40 *antimalware* dapat mendeteksi sampel *malware Riltok*. *File* sampel *malware* berukuran 992.55 KB dan tipe *file* APK. *File* APK merupakan tipe *file* yang umum digunakan pada sistem operasi *android* sebagai *file* distribusi dan pemasangan perangkat lunak atau aplikasi pada sistem *android*.

Keaslian atau *data integrity* dari kedua sampel *malware* dapat dilakukan pemeriksaan dengan membandingkan informasi nilai *checksum* yang diperoleh dari Virus Total dan nilai *checksum* yang diperoleh menggunakan *framework* MobSF. Hasil pemeriksaan dari nilai *checksum* kedua sampel *malware* menggunakan *framework* MobSF dapat dilihat pada gambar 3 dan 4.



Gambar 4 Checksum Malware Bouncing Golf MobSF

Gambar 4 menunjukkan nilai *checksum* SHA256 dari sampel *malware Bouncing Golf* menggunakan *framework* MobSF. Tidak hanya terbatas *pada file information*, namun terlihat juga nilai *application scores* berdasar dari *framework* MobSF dan juga *application information* dari *file* APK atau sampel *malware* yang akan dilakukan analisa.



Gambar 5. Checksum Malware Riltok MobSF

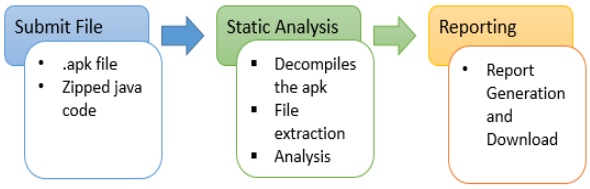
Gambar 5 menunjukkan hasil dari nilai *checksum* SHA256 pada sampel *malware Riltok* dan memperlihatkan nilai yang diperoleh melalui *framework* MobSF identik dengan informasi nilai *checksum* yang diperoleh melalui *VirusTotal*, juga berarti bahwa sampel *pada malware Riltok* yang digunakan pada penelitian ini asli atau identik serta tidak mengalami modifikasi atau perubahan apapun. Perbandingan nilai *checksum* kedua sampel *malware* dapat dilihat pada Tabel 3.

Tabel 3. Perbandingan nilai checksum virustotal dengan MobSF

| No | Malware | Nilai Checksum | | Definisi |
|---|---|---|---|---|
| | | Virus Total | MobSF | |
| 1 | Bouncing Golf | 55123ed4 982fa135d beda4996 9ab68444 125143e3 6930fe161 2d367f2fa 615fc | 55123ed4 982fa135d beda4996 9ab68444 125143e3 6930fe161 2d367f2fa 615fc | Identik |
| 2 | Riltok | 0497b600 0a7a23e9e 9b97472b c2d3799c af49cbbea 1627ad4d 87ae6e0b 7e2a98 | 0497b600 0a7a23e9e 9b97472b c2d3799c af49cbbea 1627ad4d 87ae6e0b 7e2a98 | Identik |

*B.* Static Analysis

Analisis statis yang dilakukan dalam penelitian ini menerapkan metode *reverse engineering* pada sampel *malware* untuk mendapatkan *java source code*. *Tool* yang digunakan dalam *reverse engineering* ini adalah MobSF, kemudian analisis dilakukan pada *source code* untuk mendapatkan kode kode yang *terindikasi* merusak atau terdapat program berbahaya.



Gambar 6. Alur Analisis Static file APK dengan MobSF

Gambar 6 merupakan alur *reverse engineering* pada sampel *malware* menggunakan *framework* MobSF. Tahapan pertama dalam analisis statis adalah dengan melakukan *reverse engineering* pada sampel *malware* dengan melakukan perubahan ekstensi *file* yang sebelumnya adalah **.apk* menjadi **.zip* menggunakan *framework* MobSF, hasil dari proses ini berupa ekstraksi dari *file* sampel *malware* menjadi data yang dapat dianalisa seperti pada alur proses gambar 6. Hasil *file* ekstraksi dari *framework* MobSF adalah *report.pdf* atau hasil berupa laporan yang selanjutnya akan dianalisa untuk lebih mengetahui perizinan apa saja yang diminta oleh aplikasi dari sampel *malware* terhadap sistem serta *source code* dari *file* aplikasi yang sudah diubah melalui proses *decompiler*.

### *C. Bouncing Golf Static Analysis*

1) *Application Permission Analysis: File report.pdf* dari hasil *reverse engineering* menggunakan *framework* MobSF mengandung seluruh informasi dari sebuah aplikasi, seperti *file information*, *application information*, *application components*, *certificate information*, *application permissions*, APKID *analysis*, *manifest analysis*, *code analysis*, *domain malware check*, *firebase database*, *emails*, *dan trackers.* Seluruh informasi yang didapatkan tersebut akan dianalisa dengan fokus utama pada *application permission* dan *code analysis*. *Application permission* adalah informasi hak akses atau perizinan yang diminta *oleh* sampel *malware.*

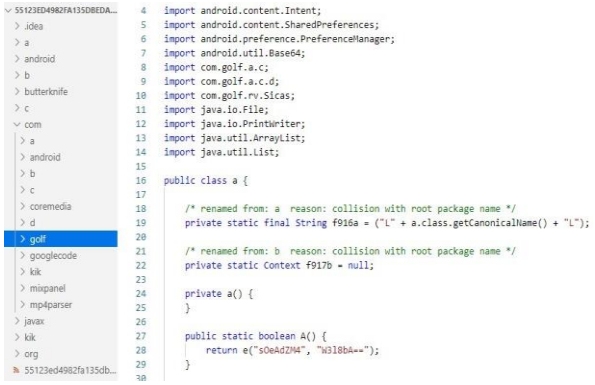
Informasi perizinan yang didapatkan selanjutnya dilakukan pembagian berdasar pada tingkat keamanan sesuai dengan potensi atau hasil kerusakan yang disebabkan oleh *malware Bouncing Golf*. Detail lengkap mengenai informasi perizinan sampel *malware Bouncing Golf* dapat dilihat pada Tabel 4.

Tabel 4. Tingkat keamanan perizinan malware bouncing golf

| No | Permission | Status | Information |
|---|---|---|---|
| 1 | android.permission .INTERNET | Dangerous | Full internet access |
| 2 | android.permission .READ_PHONE_STATE | Dangerous | Read phone state and identity |
| 3 | android.permission .READ_CONTACTS | Dangerous | Read contact data |
| 4 | android.permission .WRITE_EXTERNAL_STORAGE | Dangerous | Read/modify /delete SD card contents |
| 5 | android.permission .WAKE_LOCK | Dangerous | Prevent phone from sleeping |
| 6 | kik.android.permission.CONTACT | Dangerous | Unknown permission from android reference |
| 7 | com.android.vending.BILLING | Dangerous | Unknown permission from android reference |
| 8 | android.permission .ACCESS_FINE_LOCATION | Dangerous | Fine (GPS) location |
| 9 | android.permission .ACCESS_COARSE_LOCATION | Dangerous | Coarse (network based) location |
| 10 | ancdroid.permisson .CAMERA | Dangerous | Take pictures and videos |
| 11 | android.permission .RECORD_AUDIO | Dangerous | Record audio |
| 12 | android.permission .STORAGE | Dangerous | Unknown permission from android reference |
| 13 | android.permission .SYSTEM_ALERT_WINDOW | Dangerous | Display system alerts |
| 14 | android.permission .QUICKBOOT_POWERON | Dangerous | Unknown permission from android reference |
| 15 | android.permission .READ_SMS | Dangerous | Read SMS or MMS |
| 16 | android.permission .WRITE_SMS | Dangerous | Edit SMS or MMS |
| 17 | com.android.browser.permission.READ_HISTORY_BO | Dangerous | Read browser history and |

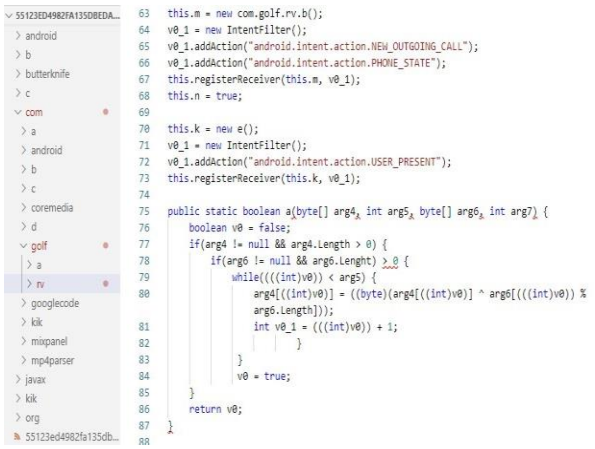
| | | | |
|---|---|---|---|
| | OKMARKS | | bookmarks |
| 18 | android.permission .GET_TASK | Dangerous | Retrieve running applications |
| 19 | android.permission .CALL_PHONE | Dangerous | Directly call phone numbers |
| 20 | android.permission .SEND_SMS | Dangerous | Send SMS messages |
| 21 | android.permission .WRITE_SETTIN GS | Dangerous | Modify global system settings |
| 22 | android.permission .READ_EXTERN AL_STORAGE | Dangerous | Read SD card contents |
| 23 | android.permission _RECEIVE_SMS | Dangerous | Receive SMS |
| 24 | android.permission .PROCESS_OUTG OING_CALLS | Dangerous | Intercept outgoing calls |
| 25 | android.permission .READ_CALL_L OG | Dangerous | Read user call log |
| 26 | android.permission .WRITE_CALL_L OG | Dangerous | Write user call log data |

2) *Code Review:* Tahap ini merupakan proses lanjutan dari hasil *permission analysis*, setelah semua perizinan yang diminta oleh aplikasi *malware* terhadap sistem ditemukan maka dilakukan analisa lebih lanjut dalam bentuk *java source code. Report* yang didapatkan dari hasil analisis menggunakan *framework* MobSF tidak hanya dalam bentuk laporan PDF, namun juga *file* ekstraksi dari sampel *malware* melalui proses *decompile* dan *file extraction* menjadi *source code java*, dengan ini struktur dan *source code* dapat dianalisa.




Gambar 7. Struktur Source Code Malware Bouncing Golf

Gambar 7 menunjukkan struktur dari sampel *malware Bouncing Golf* setelah melalui proses *reverse engineering* menggunakan *framework* MobSF. Analisis statis digunakan dalam melakukan analisis fungsi-fungsi yang ada pada *source code* dari sampel *malware*. Tahapan ini diperlukan dalam mencari kode program yang dapat bersifat *malicious* atau berbahaya, dan sampel *malware Bouncing Golf* masuk dalam kategori *spyware*. Beberapa fungsi program *malicious* ditemukan dalam static analysis, di mana program *malicious* disisipkan dalam folder golf.




Gambar 8. Analisa Fungsi Source Code Call Phone Monitor

Gambar 8 merupakan hasil dari analisa statis pada *source code malware Bouncing Golf* dan ditemukan sampel *malware* memiliki fungsi untuk melakukan pemantauan dan merekam panggilan telepon yang sedang terjadi pada perangkat *smartphone* pengguna, selain itu ada *data call logs, contacts, accounts, browser history* dan *bookmarks*, SMS *messages*, *phone information*, dan *file* media seperti *image* dan *video* yang menjadi sasaran dari program *Bouncing Golf*. Data yang dicuri akan dikumpulkan dan dilakukan *enkripsi* menggunakan operasi XOR sederhana, kemudian data tersebut akan dikirimkan ke *server Command & Control* menggunakan metode HTTP POST.

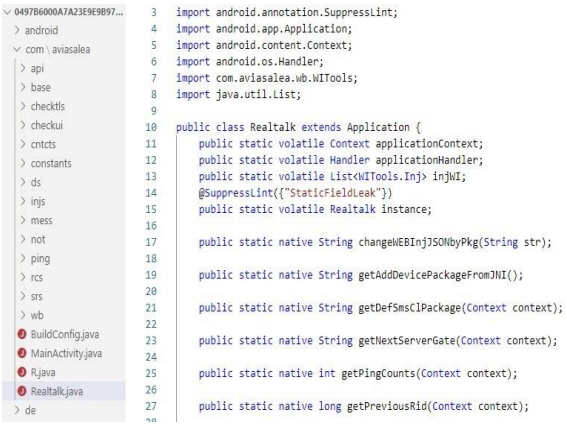
*D. Riltok Static Analysis*

1) *Application Permission Analysis:* Dalam proses *permission analysis* yang akan dilakukan sama seperti pada sampel *malware Bouncing Golf*. Teknik *reverse engineering* dilakukan dengan menggunakan *framework* MobSF untuk melakukan *decompile file* APK *pada malware Riltok*. Hasil dari proses *decompile* berupa *file report.pdf* yang selanjutnya akan dianalisa berdasar hasil laporan tersebut. Analisa berfokus pada *application permission analysis* untuk mengetahui perizinan yang diminta oleh sampel *malware* dan dampak terhadap sistem, seperti terlihat pada Table 5.

Tabel 5. Tingkat keamanan perizinan malware riltok

| No | Permission | Status | Information |
|---|---|---|---|
| 1 | android.permission .INTERNET | Dangerous | Full internet access |
| 2 | android.permission .USES_POLICY_FORCE_LOCK | Dangerous | Unknown permission from android reference |
| 3 | android.permission .WAKE_LOCK | Dangerous | Prevent phone from sleeping |
| 4 | android.permission .WRITE_SETTINGS | Dangerous | Modify global system settings |
| 5 | android.permission .SYSTEM_ALERT_WINDOW | Dangerous | Display system-level alerts |
| 6 | android.permission .REAL_GET_TASK | Dangerous | Unknown permission from android reference |
| 7 | android.permission .READ_PHONE_STATE | Dangerous | Read phone state and identity |
| 8 | android.permission .RECEIVE_SMS | Dangerous | Receive SMS |
| 9 | android.permission .READ_SMS | Dangerous | Read SMS or MMS |
| 10 | android.permission .SEND_SMS | Dangerous | Send SMS messages |
| 11 | android.permisson. CALL_PHONE | Dangerous | Directly call phone numbers |
| 12 | android.permission .CHANGE_WIFI_STATE | Dangerous | Change Wi-Fi status |
| 13 | android.permission .CHANGE_NETWORK_STATE | Dangerous | Change network connectivity |
| 14 | android.permission .GET_TASK | Dangerous | Received running applications |

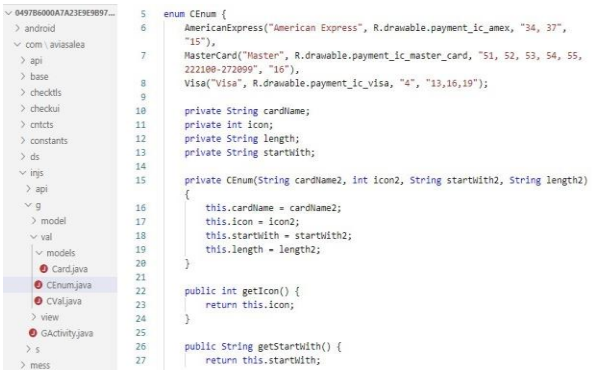
2) *Code Review:* Tahap ini merupakan proses lanjutan dari hasil *permission analysis*, setelah semua perizinan yang diminta oleh aplikasi sampel *malware* terhadap sistem ditemukan maka akan dilakukan analisa lebih lanjut dalam bentuk *java source code*. *Report* yang didapatkan dari hasil analisis menggunakan *framework* MobSF tidak hanya dalam bentuk laporan PDF, namun juga *file* ekstraksi dari sampel *malware* melalui proses *decompile* dan *file extraction* menjadi *source code java*, dengan ini struktur dan *source code* dapat dianalisa.



Gambar 9 Struktur Source Code Java Malware Riltok

Gambar 9 merupakan *struktur dari* sampel *malware Riltok* setelah melalui proses *reverse engineering* menggunakan *framework* MobSF. Analisa statis digunakan dalam melakukan menganalisa fungsi-fungsi yang ada pada *source code*, tahapan ini sama seperti pada sampel *malware* sebelumnya yang diperlukan dalam mencari kode program yang dapat bersifat *malicious* atau berbahaya, dan sampel *malware Riltok* masuk dalam kategori *phising*. Beberapa fungsi program *malicious* ditemukan pada sampel *malware Riltok* diantaranya adalah menjadikan *malware* sebagai *default* SMS *application*, dan

menampilkan halaman pembayaran dengan detail kartu bank.




Gambar 10. Analisa Fungsi Source Code Fake Google Play Request Bank Card Details

Gambar 10 menunjukkan hasil dari analisa statis pada *source code malware Riltok* dan ditemukan sampel *malware* memiliki fungsi utama dalam mencuri data kartu kredit dan mengirimkan data tersebut ke *server Command & Control*. *Malware Riltok* menampilkan halaman pembayaran palsu dari *Service Google Play* yang meminta detail kartu bank serta menampilkan halaman pembayaran melalui *browser* dengan menyerupai pembayaran dari *mobile banking*. Setelah pengguna memasukan detail kartu bank pada halaman pembayaran palsu, *malware* akan melakukan validasi dasar seperti periode kartu, nomor kartu, jumlah nomor kartu, panjang CVC dan melakukan pencocokan data dengan daftar *blacklist* kartu yang telah disiapkan oleh *malware Riltok.*

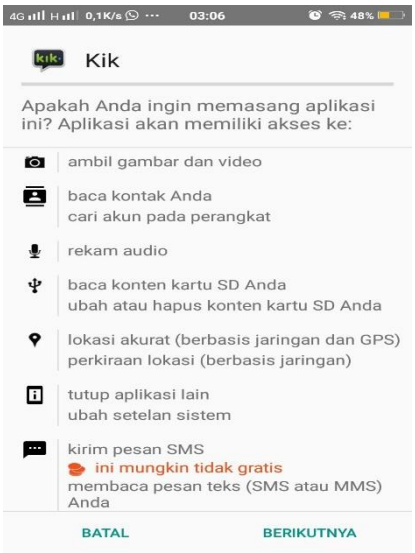
*E. Dynamic Analysis*

Dalam menganalisis sebuah *malware android* khususnya analisa dinamis pada umumnya memerlukan sebuah *environment* yang aman atau biasanya disebut *virtual lab*. Perangkat virtual dalam proses analisa dinamis ini menggunakan *virtual lab* (*emulator*) *android* dari *Genymotion* dengan versi *android* 8.1 dan API level 27 (Oreo) yang dihubungkan dengan MobSF untuk proses analisa dinamis dengan menggunakan jaringan *local area network* (LAN).

Sampel *malware* yang digunakan dalam analisis dinamis sama dengan sampel pada analisis statis, yang kemudian dijalankan (*install*) dan dilakukan analisa *permission*, juga dilakukan analisa kembali terhadap karakteristik dari *malware Bouncing Golf dan Riltok.*

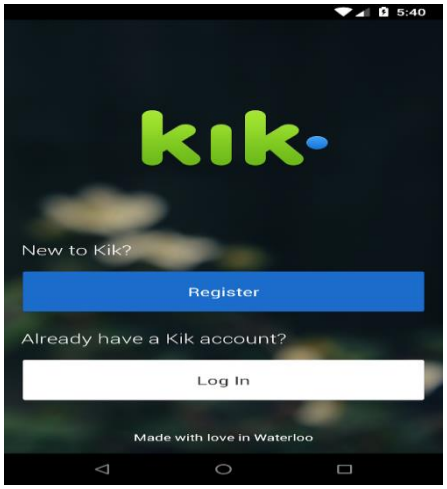
*F. Bouncing Golf Dynamic Analysis*

1) *Installing and Running*: Pemasangan *malware Bouncing Golf* yang menyerupai aplikasi KIK (aslinya adalah sebuah aplikasi media *chatting* online) menampilkan beberapa perizinan yang diminta oleh *malware Bouncing Golf* sebagai syarat penginstalan. Melihat beberapa perizinan yang diminta oleh *malware Bouncing Golf* untuk melakukan *install* atau pemasangan aplikasi, beberapa perizinan yang pada dasarnya tidak dibutuhkan dan dapat dipastikan bahwa *malware Bouncing Golf* sedang mencoba menarik informasi dari penggunanya.




Gambar 11. Perizinan Instalasi Malware Bouncing Golf

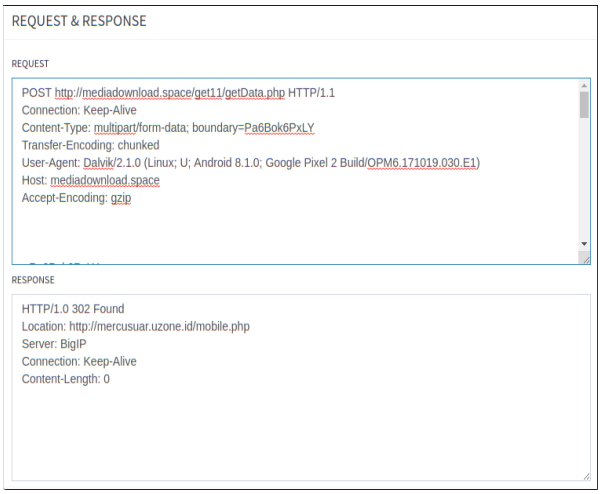
Gambar 11 menunjukkan tampilan perizinan saat pemasangan *malware Bouncing Golf*. Sampel *malware Bouncing Golf* yang sebelumnya telah terpasang kemudian dijalankan pada *emulator* dan dilakukan analisa *activity*. Pada tampilan pertama dari *malware Bouncing Golf*, didapati tampilan yang menyerupai aplikasi KIK dan terdapat dua pilihan yaitu *register* dan *login*, dari ke-2 pilihan tersebut tidak ada yang benar-benar bekerja 100% dikarenakan menu *register* dan menu *login* tidak terhubung atau tidak terkoneksi dengan *database* yang kemudian menyebabkan tidak dapat *login* ke dalam aplikasi tersebut.




Gambar 12. Register dan Log In Malware Bouncing Golf

Gambar 12 menunjukkan tampilan dari menu *login* dan *register* dari *malware Bouncing Golf*. Terlihat bahwa akses masuk atau menjalankan aplikasi ini diharuskan untuk mendaftar terlebih dahulu dengan memberikan informasi pribadi pengguna seperti pada umumnya, namun karena tidak terkoneksi dengan *database* maka tidak didapatkan akses untuk masuk ke dalam aplikasi tersebut.

2) *Traffic Analisis:* HTTP *tools* adalah sebuah *tool* perekam *traffic* HTTP yang terdapat pada analisa dinamis MobSF. Analisa *traffic* dari sampel *malware Bouncing Golf* yang dilakukan dengan MobSF menunjukkan sebuah *request* pada suatu halaman *website*.

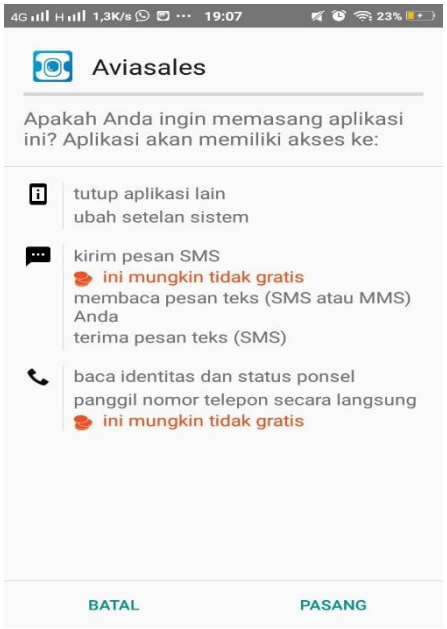



Gambar 13. Monitoring Traffic Malware Bouncing Golf menggunakan HTTP tools MobSF

Gambar 13 menunjukkan *monitoring* yang dilakukan menggunakan HTTP *tools* pada *framework* MobSF. Hasil dari *monitoring* ini menemukan sebuah respons ke alamat *website* yaitu http://mediadownload.space/get11/getdata.php dengan metode POST. Respons yang didapat dari *request* sebelumnya yaitu 302, yang artinya *website* sementara dialihkan hal tersebut disebabkan karena adanya gangguan terhadap *Domain Name Server* (DNS) yang mengarah pada *website* tersebut.

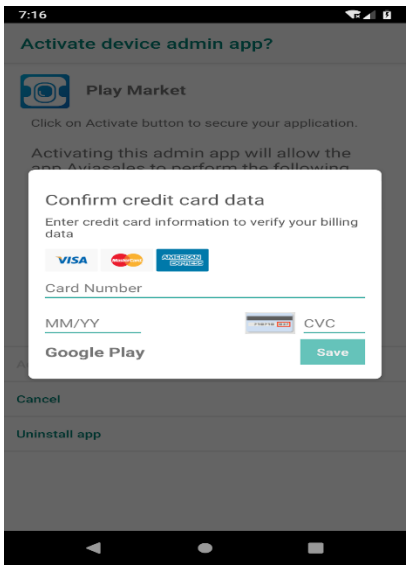
*G. Riltok Dynamic Analysis*

1) *Installing and Running*: Pemasangan atau proses *install malware Riltok* yang menyerupai aplikasi AVIASALES(aslinya adalah sebuah aplikasi pembelian tiket pesawat *online*) menampilkan beberapa perizinan yang diminta oleh sampel *malware Riltok* sebagai syarat aplikasi dapat dipasang atau *install*. Namun perizinan yang diminta oleh *malware Riltok* sangat tidak sesuai dengan fungsi dari aplikasinya, seperti membaca dan menerima SMS atau telepon yang pada dasarnya tidak diperlukan untuk sebuah aplikasi pembelian tiket *online*.




Gambar 14. Perizinan Instalasi Malware Riltok

Gambar 14 merupakan tampilan perizinan saat pemasangan *malware Riltok*. Sampel *malware* yang sebelumnya telah terpasang kemudian dijalankan pada perangkat virtual dan dilakukan analisa *activity*. Pada tampilan utama pada *malware Riltok* menampilkan *activation devices admin* dan meminta kepada pengguna untuk segera melakukan aktivasi yang kemudian memunculkan sebuah *form credit card* dan terdapat notifikasi *play protect disable*.




Gambar 14. Form Credit Card Aktivasi pada Malware Riltok

Gambar 14 menunjukkan tampilan *form credit card* pada saat aktivasi *malware Riltok*. Setelah pengguna memasukan detail kartu bank pada halaman pembayaran palsu, *malware* akan melakukan validasi

dengan kecocokan pada nomor kartu debit atau credit yang dimasukkan.

2) *Traffic Analysis*: Hasil monitoring traffic dari malware Riltok yang dijalankan pada perangkat virtual menunjukkan sebuah usaha komunikasi antara malware dengan server C&C dengan metode GET, Terlihat pada gambar 14.


Gambar 15. Monitoring Traffic Malware Riltok menggunakan HTTP tools MobSF

Pada gambar 15 *traffic* HTTP yang dilakukan proses *request* oleh *malware Riltok* dengan metode GET pada alamat http://le22999a.pw /1324273/gate.php memberikan respons 302 dan *Domain Name Server* (DNS) dibelokkan pada sebuah *website* http://mercusuar.uzone.id, yang artinya DNS tidak dapat mentranslasikan alamat IP dari *server malware Riltok* sehingga kemungkinan besar akses C&C pada sampel *malware Riltok* dan server telah dimatikan.

*H. Pencegahan*

Upaya dalam melakukan pencegahan dan melindungi *smartphone Android* dari infeksi *malware* adalah sebagai berikut :

- Pastikan hanya memasang aplikasi *android* dari sumber terpercaya seperti *Google Play Store* dan hindari pemasangan aplikasi dari sumber yang tidak dipercaya.
- Hiraukan SMS yang berisi tautan mencurigakan seperti *link* dari sebuah alamat *website*.
- Perhatikan juga *permission* atau perijinan yang diminta aplikasi terhadap sistem *android*.
- Pastikan selalu *update firmware* sistem *android* pada *smartphone*.
- Pasang antivirus untuk platform *android* dan *update* secara berkala *database* dari antivirus tersebut.

## V. KESIMPULAN

Dari hasil analisis sampel *malware Bouncing Golf* dan *Riltok* menggunakan MobSF dengan metode *static analysis* dan *dynamic analysis* menggunakan virtual lab (emulator) android dari Genymotion dengan versi *android* 8.1 dan API level 27 (Oreo) telah suskses mendapatkan beberapa karakteristik dari kedua *malware*. *Bouncing Golf* melakukan pemantauan dan merekam aktivitas yang dilakukan pengguna *smartphone android*. Sedangkan *malware Riltok* memiliki karakteristik dalam mencuri data kartu kredit dengan menampilkan halaman *billing* palsu. Kedua *malware* memiliki karakteristik yang sama, yaitu data akan dikirimkan ke *server* C&C*(Command & Control Server*). Upaya pencegahan yang dapat dilakukan untuk menghindari infeksi *malware* pada *smartphone android* yaitu dengan memastikan hanya memasang aplikasi *android* dari sumber terpercaya seperti *Google Play Store*, dan hiraukan SMS yang berisi tautan mencurigakan, serta pastikan selalu *update firmware* sistem *android* pada *smartphone*.

## SARAN

Saran yang dapat diberikan untuk pengerjaan pada penelitian berikutnya, diharapkan melalui laporan yang didapatkan ini dapat dilakukan pengembangan dalam melakukan analisa yang lebih efektif dan otomatis, seperti penerapan *machine learning* pada *malware* yang bersifat masif. Penelitian dengan teknik yang sama juga bisa dilakukan dalam investigasi serta analisis *malware* khususnya pada *smartphone* dengan platform *android* untuk mempelajari karakteristik *malware* yang terus berkembang.

## DAFTAR PUSTAKA

[1] W. Pranoto, "Malicious Software Analysis," *Cyber Secur. dan Forensik Digit.*, vol. 1, no. 2, pp. 62–66, 2019, doi: 10.4018/9781605660608.ch030.

[2] ID-CERT, "Laporan Dwi Bulan I 2018." [Online]. Available: https://cert.id/media/files/01_-_UMUM_Dwi_Bulan_I_2018.pdf. [Accessed: 26-Mar-2020].

[3] ID-CERT, "Laporan Dwi Bulan II 2018." [Online]. Available: https://cert.id/media/files/02_-_UMUM_Dwi_Bulan_II_2018.pdf. [Accessed: 26-Mar-2020].

[4] ID-CERT, "Laporan Dwi Bulan III 2018." [Online]. Available: https://www.cert.or.id/media/files/03_-_UMUM_Dwi_Bulan_III_2018.pdf. [Accessed: 26-Mar-2020].

[5] Y. A. Utomo, S. Juli, I. Ismail, and T. Z. S. T, "Membangun Sistem Analisis Malware Pada

Aplikasi Android Dengan Metode Reverse Engineering Menggunakan Remnux," vol. 4, no. 3, pp. 2000–2012, 2018.

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

[7] GlobalStats, "Mobile Operating System Market Share Worldwide on Statcounter." [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide. [Accessed: 27-Mar-2020].

[8] F. Ghaffari, M. Abadi, and A. Tajoddin, "AMD-EC: Anomaly-based Android malware detection using ensemble classifiers," *2017 25th Iran. Conf. Electr. Eng. ICEE 2017*, no. lCEE, pp. 2247–2252, 2017, doi: 10.1109/IranianCEE.2017.7985436.

[9] B. A. Wichmann, A. A. Canning, D. L. Clutterbuck, L. A. Winsborrow, N. J. Ward, and D. W. R. Marsh, "Industrial perspective on static analysis," *Softw. Eng. J.*, vol. 10, no. 2, pp. 69–75, 1995, doi: 10.1049/sej.1995.0010.

[10] V. Rastogi, Y. Chen, and X. Jiang, "DroidChameleon: Evaluating Android antimalware against transformation attacks," *ASIA CCS 2013 - Proc. 8th ACM SIGSAC Symp. Information, Comput. Commun. Secur.*, pp. 329–334, 2013, doi: 10.1145/2484313.2484355.

[11] A. F. Febrianto, A. Budiono, P. Studi, S. Informasi, F. R. Industri, and U. Telkom, "Analisis Malware Pada Sistem Operasi Android Menggunakan Metode Network Traffic Analysis Malware Analysis in Android Operating System Using Network," vol. 6, no. 2, pp. 7837–7844, 2019.

[12] Mobliciti, "The current state of mobile malware on Mobliciti." [Online]. Available: https://mobliciti.com/the-current-state-of-mobile-malware. [Accessed: 29-Mar-2020].

[13] K. Alfalqi, R. Alghamdi, and M. Waqdan, "Android Platform Malware Analysis," *Int. J. Adv. Comput. Sci. Appl.*, vol. 6, no. 1, pp. 140–146, 2015, doi: 10.14569/ijacsa.2015.060120.

[14] N. Zalavadiya and P. Sharma, "A Methodology of Malware Analysis, Tools and Technique for windows platform–RAT Analysis," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 5, no. 3, pp. 8198–8205, 2017, doi: 10.15680/IJIRCCE.2017.

[15] A. S. Rusdi, N. Widiyasono, and H. Sulastri, "Analisis Infeksi Malware Pada Perangkat Android Dengan Metode Hybrid Analysis," no. 24, 2019.

[16] A. Kartono, A. Sularsa, and S. J. I. Ismail, "Membangun Sistem Pengujian Keamanan Aplikasi Android Menggunakan Mobsf," vol. 5, no. 1, pp. 146–151, 2019.

[17] Sugiyono, *Metode Penelitian Pendidikan Pendekatan Kuantitatif, Kualitatif, dan R&D*, Bandung: Alfabeta, 2014.