MEMBANGUN SISTEM PENGUJIAN KEAMANAN APLIKASI ANDROID MENGGUNAKAN MOBSF

[1]Aan Kartono, [2]Anang Sularsa, [3]Setia Juli Irzal Ismail.

[1] Computer Engginering, Telkom University, Bandung, Indonesia.

[1] aankartono@student.telkomuniversity.ac.id, [2]anang@tass.telkomuniversity.ac.id, [3]jul@tass.telkomuniversity.ac.id.

Abstrak: *Smartphone* sekarang sudah didukung oleh aplikasi-aplikasi yang beragam yang sangat membantu untuk kehidupan sehari-hari. Tapi dari banyaknya aplikasi *android* ada beberapa aplikasi yang tersusupi *malware* berbahaya yang tanpa disadari dapat merusak ponsel itu sendiri, aplikasi yang sudah tersusupi *malware* juga dapat mencuri data pengguna yang tersimpan pada ponsel tersebut. *Mobile Security Framework (MobSF)* dapat menganalisis *malware* pada aplikasi *android* dengan melakukan analisis statik dan dinamis, dalam melakukan proses analisis yang kemudian akan memberikan hasil berupa laporan yang menyatakan bahwa aplikasi *android* tersebut terbebas dari *malware* berbahaya oleh pihak yang tidak bertanggung jawab. Proyek akhir ini menggunakan *Mobile Security Framework (MobSF)* sebagai perangkat lunak untuk menganalisis aplikasi *android*. Berbagai macam jenis *malware* android dapat terdeteksi seperti diantaranya *Trojan, Ransomware, Adware, dan s*ebagainya. *Mobile Security Framework (MobSF)* juga dapat mengetahui *source code* yang berbahaya pada sebuah aplikasi *android.*

Kata Kunci *: Mobile Security Framework (MobSF)*, Aplikasi *android*

1. Pendahuluan

1.1 Latar Belakang

Perkembangan Pada era globalisasi ini, perkembangan teknologi semakin pesat. Penggunaan ponsel lebih didominasi oleh ponsel pintar yang lebih dikenal dengan nama *smartphone*, ada berbagai jenis sistem operasi yang digunakan *smartphone* seperti, *IOS, Windows Phone*, dan *Android*. S*martphone* dengan sistem operasi *android* merupakan *smartphone* yang paling banyak digunakan saat ini di seluruh dunia, dengan didukung oleh aplikasi-aplikasi beragam yang bertujuan untuk memudahkan kehidupan manusia sehari-hari. Namun dari banyaknya aplikasi yang tersedia untuk *smartphone android* ada beberapa aplikasi-aplikasi *android* yang berbahaya yang tanpa disadari dapat membahayakan *smartphone* penggunanya sendiri, beberapa contoh aplikasi-aplikasi yang dapat merugikan diantaranya aplikasi baterai palsu, aplikasi memori palsu, *antivirus* palsu, dan aplikasi pihak ketiga lainnya yang banyak tersebar di *google*. Alasan mengapa aplikasi-aplikasi tersebut termasuk berbahaya karena aplikasi tersusupi *malware* yang dapat merusak perangkat maupun mencuri data pengguna.

Mobile Security Framework (*MobSF*) adalah *framework* pengujian otomatis bersifat *open-source*, yang mampu melakukan analisis statis dan dinamis dalam melakukan proses analisis akan menampilkan hasil berupa laporan mengenai aplikasi *android* tersebut.

Berdasarkan hal tersebut penulis akan membangun sebuah sistem keamanan aplikasi *android* dengan menggunakan *Mobile Security Framework* (*MobSF*) untuk dapat mengetahui adanya *malware* berbahaya pada aplikasi *android*. Dengan menggunakan *Mobile Security Framework* (*MobSF*) akan dilakukan analisis statis dan dinamis pada beberapa aplikasi *android* yang ada di *Web* dan sampel yang sudah mengandung *malware*, yang kemudian akan dianalisis lebih lanjut mengenai hasilnya yang berupa *source code program*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang terdapat diatas, maka rumusan masalah pada proyek akhir ini adalah :

1. Bagaimana cara mengetahui adanya *malware* pada aplikasi *android* ?
2. Bagaimana cara menggunakan *Mobile Security Framework (MobSF)* untuk menganalisis *malware* yang ada di aplikasi *android* ?

1.3 Tujuan

Tujuan dari proyek akhir ini adalah :

1. Menggunakan *Mobile Security Framework (MobSF)* sebagai analasis keamanan dari aplikasi *android.*
2. Menganalisis *source code* yang didapat dengan *Mobile Security Framework (MobSF).*

1.4 Batasan Masalah

Batasan masalah yang digunakan untuk membatasi masalah dalam proyek akhir ini adalah :

1. Menggunakan *Mobile Security Framework (MobSF)* sebagai *framework* pendeteksi keamanan aplikasi *android* yang digunakan.

2. Menggunakan aplikasi berbasis *android* sebagai bahan pengujian.
3. Menggunakan *Virtual Machine* sebagai *software* pendukung untuk pengujian.
4. Menggunakan *Dynamic analysis* dan *Static analysis* sebagai metode analisis.

1.5 Definisi Operasional

Definisi operasional pada proyek akhir ini adalah :

1. **Sistem.** suatu paduan yang terdiri dari beberapa unsur yang tergabung satu sama lain agar mempermudah laju aliran informasi, energi ataupun materi hingga dapat mencapai tujuan tertentu. [2]
2. **Keamanan**. keadaan bebas dari bahaya. Istilah ini bisa digunakan dengan hubungan kepada kejahatan, segala bentuk kecelakaan, dan lain-lain**.** [2]
3. **Aplikasi.** dapat diartikan sebagai suatu program berbentuk perangkat lunak yang berjalan pada suatu sistem tertentu yang berguna untuk membantu berbagai kegiatan yang dilakukan oleh manusia. [1]
4. ***Android*.** adalah sistem operasi berbasis *Linux* yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. [2]
5. ***MobSF*.** *Mobile Security Framework* (*MobSF*) adalah *framework* pengujian otomatis bersifat *open-source*, yang mampu melakukan analisis statis dan dinamis. [8]

2. Tinjauan Pustaka

2.1 Malware

Malware atau *Malicious Software* merupakan sebuah program atau *software* jahat yang sengaja dibuat dengan tujuan tertentu oleh pembuatnya yang dapat mengakibatkan berbagai kerugian pada pengguna yang menjadi sasaran dari pembuat *malware* tersebut. Kata *malware* merupakan istilah umum yang digunakan untuk *software* maupun *program* yang dirancang bertujuan untuk menyusup atau merusak sebuah sistem secara diam-diam.[2]

2.1.1 Contoh-Contoh *Malware*

1. Virus
   Virus adalah jenis *malware* yang menyerang *file* eksekusi (*.exe*) yang akan menyerang dan menggandakan diri ketika *file exe* yang terinfeksi di jalankan.Virus komputer menyebar dengan cara menyisipkan program dirinya pada program atau dokumen yang ada dalam komputer.
2. Worm
   Worm adalah sebuah program komputer yang dapat menggandakan dirinya secara sendiri dalam sistem komputer. Sebuah *worm* dapat menggandakan dirinya dengan memanfaatkan jaringan (LAN/WAN/ Internet) tanpa perlu campur tangan dari user itu sendiri. *Worm* memanfaatkan celah keamanaan yang memang terbuka atau lebih dikenal dengan sebutan *vulnerability*.
3. Spyware
   Spyware adalah program yang bertindak sebagai mata-mata untuk mengetahui kebiasaan pengguna komputer dan mengirimkan informasi tersebut ke pihak lain. *Spyware* biasanya digunakan oleh pihak pemasang iklan.
4. Adware
   Adware adalah iklan yang dimasukan secara tersembunyi oleh pembuat program, biasanya pada program yang bersifat *freeware* untuk tujuan promosi atau iklan.
5. Trojan
   Trojan atau *trojan hourse* adalah program yang diam-diam masuk ke komputer kita, kemudian memfasilitasi *program* lain misalnya virus, *sypware*, *adware*, *keylogger* dan *malware* lainnya untuk masuk, merusak sistem, memungkinkan orang lain me-*remote* komputer dan mencuri informasi seperti *password* atau nomor kartu kredit.
6. Keylogger
   Keylogger adalah sebuah program yang dapat memantau penekanan tombol pada *keyboard*, sehingga orang lain dapat mengetahui *password* dan informasi apapun yang korban ketik.
7. Rootkit
   Rootkit adalah program yang menyusup kedalam sistem komputer, bersembunyi dengan menyamar sebagai bagian dari sistem (misalnya menempel pada *patch*, *keygen*, *crack* dan *game*), kemudian mengambil alih, memantau kerja sistem yang disusupinya. *Rootkit* dapat mencuri data yang lalu-lalang di jaringan, melakukan *keylogging*, mencuri *cookies* akun bank dan lain-lain.
8. Phising
   Phishing adalah suatu bentuk penipuan untuk memperoleh informasi pribadi seperti *userID*, *password*, *ATM*, kartu kredit dan sebagainya melalui *e-mail* atau *website* palsu yang tampak asli

2.2 Analisis Malware

Analisis *malware* adalah proses studi tentang *malware* dengan membedah komponen-komponen yang berbeda untuk memahami

perilaku dan karakteristik dari *malware* untuk mendeteksi dan mengetahui cara *malware* tersebut bekerja dan mencari celah dalam keamanan. [3]

2.2.1 Analisis Statis

Metode analisis statis adalah metode yang dilakukan tanpa menjalankan *malware* secara langsung namun dengan melihat *source code* dari file yang akan dianalisis dengan tujuan untuk melihat secara detail mengenai mekanisme kerja *malware*. Analisis statis pada *Mobile Security Framework* (*MobSF*) dapat mengidentifikasi beberapa hal pada sebuah file yang dianalisis seperti diantaranya, mendeteksi perizinan dan konfigurasi yang tidak aman yang dapat mengakibatkan rentan terhadap serangan malware, mendeteksi kode yang berbahaya, dan perubahan terhadap tempat penyimpanan file tersembunyi. [2]

2.2.2 Analisis Dinamis

Metode analisis dinamis dilakukan dengan menjalankan sampel *malware* pada sebuah ruang lingkup yang dikontrol dan di monitor secara langsung selama *malware* itu aktif dengan tujuan untuk mendapatkan informasi yang lebih banyak tentang *malware* tersebut. Analisis dinamis pada *Mobile Security Framework* (*MobSF*) dilakukan menggunakan bantuan *Virtual Machine*, hal yang dapat dideteksi diantaranya seperti, paket jaringan yang diambil, lalu lintas *HTTPS* yang didekripsi, log, laporan yang salah atau rusak, informasi *debug*, dan pengaturan *file*. [2]

2.3 Mobile Security Framework

Mobile Security Framework (*MobSF*) adalah *framework* pengujian otomatis bersifat *open-source* yang cerdas, yang mampu melakukan analisis statis dan dinamis. [8]

2.4 Kali Linux

Kali Linux adalah distro turunan *Debian* dan juga penerus *BackTrack* yang digunakan untuk melakukan penetrasi pada jaringan. *Kali Linux* memiliki lebih dari 300 *tools* di dalamnya dengan fungsi masing-masing. *Kali Linux* juga bersifat *Live CD* dan Instalasi manual. Dari segi tampilan *Kali Linux* memiliki tampilan sederhana dan tidak terlalu mencolok dan penggunaanya pun tergolong cukup mudah, sehingga sangat baik untuk para pemula dalam melakukan penetrasi pada jaringan. Karena hal tersebut *MobSF* sangat cocok dijalankan di sistem operasi *Kali Linux*. [8]

2.5 Android

Android adalah sistem operasi berbasis *Linux* yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet yang sekarang memiliki pengguna yang banyak diseluruh dunia. Namun tidak jarang sistem operasi *android* merupakan sasaran bagi para pihak yang tidak bertanggung jawab untuk menyebarkan *malware* yang berbahaya yang dapat merugikan pengguna *android*. [1]

2.6 Bahasa Pemrograman Python

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, *python* lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat *Python* sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain. *MobSF* menggunakan bahasa pemrograman *python* dalam konfigurasi dan penggunaannya. [1]

2.7 Java Devolopment Kit

Java Development Kit atau biasa disingkat dengan *JDK* adalah Perangkat lunak yang digunakan untuk melakukan proses kompilasi dari kode java ke *bytecode* yang dapat dimengerti dan dapat dijalankan oleh *JRE* (*Java Runtime Envirotment*). *MobSF* membutuhkan *Java Development Kit* sebagai *compiler* dan *debugger* yang diperlukan untuk mengembangkan konfigurasi yang digunakan pengguna. [6]

2.8 Virtual Machine

Virtual Machine adalah implementasi perangkat lunak dari sebuah mesin komputer yang dapat menjalankan program sama seperti layaknya sebuah komputer asli. MobSF menggunakan Virtual Machine agar dapat melakukan Analisis dinamis, mesin yang dipakai adalah Android 4.4.2. [6]

3. Analisis Dan Perancangan

3.1 Analisis

Pada bab ini menjelaskan mengenai proses analisis terkait dengan cara kerja sistem yang sedang berjalan, kemudian pada bab ini juga akan dibahas penjelasan mengenai gambaran umum sistem, dan blok diagram.

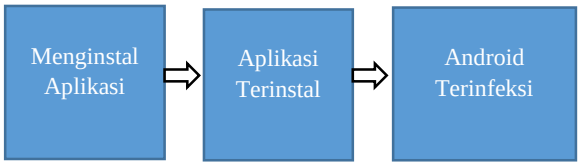
3.1.1 Gambaran Sistem Saat Ini

Gambar 3. 1 Gambaran Sistem Saat Ini.

Pada gambar 3. 1 Merupakan gambaran sistem saat ini, *Andoid* yang akan menginstal sebuah aplikasi yang kemungkinan tersusupi *malware* didalamnya tanpa sepengetahuan pengguna yang akhirnya *android* terjangkit *malware* tersebut dan dapat

merugikan pengguna seperti terjadi kerusakan terhadap *smartphone android* maupun pencurian data privasi pengguna oleh pihak yang tidak bertanggung jawab.

3.1.2 Blok Diagram



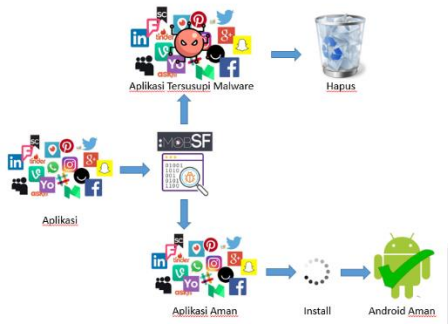
Gambar 3. 2 Blok Diagram.

Pada gambar 3. 2 merupakan tampilan dari blok diagram sistem saat ini, yang merupakan cara kerja sistem saat ini adalah dengan menginstal aplikasi *android* tanpa meyadari aplikasi tersebut sudah tersusupi *malware*.

3.2 Perancangan

Perancangan menjelaskan mengenai proses cara kerja sistem yang akan dibuat, Kemudian juga akan dibahas penjelasan mengenai gambaran umum sistem usulan, blok diagram, cara kerja, dan analisis kebutuhan fungsional dan *non* fungsional.

3.2.1 Gambaran Sistem Usulan



Gambar 3. 3 gambaran Sistem Usulan.

Pada gambar 3. 3 merupakan gambaran sistem usulan dalam Proyek Akhir ini, sistem ini menggunakan *MobSF* sebagai *framework* pengujian untuk melakukan analisis *malware*. Untuk menjalankan sistem ini diperlukan beberapa alat yang sudah tersedia di dalam sistem operasi, serta diperlukan beberapa alat atau *software* tambahan dan juga beberapa konfigurasi.

3.2.2 Blok Diagram

Pada gambar 3. 4 merupakan blok diagram sistem saat ini ketika menganalisis *malware* adalah seperti berikut.

Gambar 3. 4 Blok Diagram.

1 *INPUT*

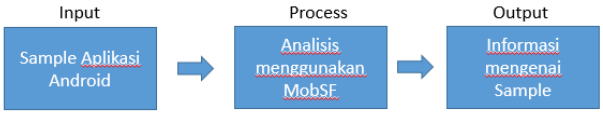
Pada tahap ini pengguna melakukan instalasi *Mobile Security Framework* terlebih dahulu dengan cara mengambil *framework* dari *https://github.com/MobSF/Mobile-Security-Framework-MobSF*.

2 *PROCESS*

Pada tahap ini akan dilakukan konfigurasi terlebih dahulu pada *Mobile Security Framework* dengan menginstal beberapa program yang diperlukan seperti *python*, *Java*, dan *Virtual Machine* yang akan dijelaskan lebih lanjut pada bagian implementasi.

3 *OUTPUT*

Pada tahap ini akan dihasilkan *output* berupa *Mobile Security Framework* siap digunakan untuk menganalisis aplikasi *android*.



Gambar 3. 5 Blok Diagram.

1. *INPUT*

Pada tahap ini digunakan perangakat keras yaitu laptop yang menggunakan *sistem operasi Kali linux*. Kemudian *Mobile Security Framework (MobSF)* dijalankan untuk melakukan analisis *malware*. Dilakukan dengan memasukkan sampel aplikasi *android* yang didapat dari beberapa tempat seperti *Web* maupun sampel aplikasi *android* yang memang sudah berisi *malware.*

2. *PROCESS*

Pada tahap ini akan dilakukan analisis dengan memasukkan sampel aplikasi android yang sudah disediakan sebelumnya yang kemudian dianalisis dengan *Mobile Security Framework (MobSF)*. Metode analisis yang digunakan adalah anaslis statis dan analisis dinamis.

3. *OUTPUT*

Pada tahap ini akan dihasilkan *output* berupa laporan yang berisi mengenai informasi tentang isi yang terkandung dalam sampel yang diperiksa.

3.2.3 Cara Kerja

Berikut ini adalah cara kerja sistem yang akan dibuat dalam Proyek Akhir ini adalah sebagai berikut.

1 Pengguna mengambil beberepa sampel aplikasi *android*, kemudian dilakukan analisis dengan menggunakan *Mobile Security Framework* (*MobSF*).

2 *Mobile Security Framework* (*MobSF*) dijalankan dan melakukan analisis terhadap aplikasi *android*, Kemudian akan memberikan hasil analisis dari sampel tersebut berupa laporan.
3 Hasil dari analisis akan berupa laporan mengenai informasi sampel aplikasi *android* tersebut, pengguna juga bisa melanjutkan analisis dengan menggunakan metode analisis dinamis untuk mendapatkan hasil yang lebih lengkap.

3.2.4 Analisis Kebutuhan Fungsional Dan Non Fungsional

A. Kebutuhan Fungsional

Berikut ini adalah kebutuhan sistem yang diperlukan untuk menyelesaikan Proyek Akhir ini adalah sebagai berikut.

1 *Mobile Security Framework* (*MobSF*) dibangun untuk dapat menganalis *malware* pada aplikasi *android*.
2 *Python* adalah Bahasa pemrograman yang digunakan untuk konfigurasi *Mobile Security Framework* (*MobSF*)..
3 Mesin virtual digunakan sebagai analisis dinamis.
4 *Android* yang akan dijadikan sampel untuk uji coba didapatkan dari *Web*.

B. Kebutuhan *Non* Fungsional

Kebutuhan non fungsional pada sistem yang akan dibangun terdiri dari dua bagian yaitu *hardware* dan *software*, adapun rincian dari kedua bagian tersebut adalah sebagai berikut.

1 *Hardware*

Pada Tabel 3. 1 merupakan kebutuhan *hardware* atau perangkat keras yang dibutuhkan untuk membangun sistem adalah sebagai berikut.

Tabel 3. 1 Spesifikasi *Hardware*.

No	Nama	Keterangan
1	Laptop	Menjalankan sistem operasi Linux.

2 *Software*

Pada Tabel 3. 2 merupakan kebutuhan *software* atau perangkat lunak yang dibutuhkan untuk membangun sistem adalah sebagai berikut.

Tabel 3. 2 Spesifikasi *Software*.

No	Nama	Keterangan
1	Kali Linux	Sistem operasi yang digunakan untuk menjalankan *MobSF*.
2	*Mobile Security Framework (MobSF)*	*Framework* yang digunakan untuk menganalisis Aplikasi *android*
3	*Virtual Machine*	Mesin yang digunakan untuk menjalankan analisis dinamis.
4	Aplikasi *Android*	Aplikasi *android* yang didapat dari *Web* yang kemudian dijadikan sampel untuk menganalis *malware*.
5	*Java Development Kit*	*Java Development Kit* sebagai *compiler* dan *debugger* yang diperlukan untuk konfigurasi *Mobile Security Framework (MobSF).*
6	*Python*	Bahasa Pemrograman *Python* digunakan untuk konfigurasi dalam penggunaan *Mobile Security Framework (MobSF).*
7	*Malware*	Kode jahat yang ada di aplikasi *android.*

4 Pengujian

Pada Tabel 4. 1 merupakan sampel yang didapat dari *web* yang akan diujikan untuk analisis *malware* menggunakan *Mobile Security Framework (MobSF).*

Tabel 4. 1 Sampel Android.

No	Sampel	Versi	Keterangan
1	Blackmart	0.99	Aplikasi penyedia aplikasi *android.*
2	*Krep.itmtd.ywtjexf*	3.0	Aplikasi bank palsu.
3	*Earthquake*	1.1.5	Aplikasi pendeteksi gempa.

Pada Tabel 4. 2 merupakan implementasi hasil pengujian oletools untuk melakukan analisis *malware* adalah sebagai berikut.

Tabel 4. 2 Pengujian Mobile Security Framework.

No	Sampel	Perbandingan	Keterangan
1	Blackmart	Ekstraksi	Mendapakan *Function* dan beberapa kode yang dicurigakan
2	*Krep.itmtd.ywtjexf*	Ekstraksi	Mendapakan *Function* dan beberapa kode yang dicurigakan
3	*Earthquake*	Ekstraksi	Mendapakan *Function* dan beberapa kode yang dicurigakan
4	Blackmart	Hasil Analisis	"*https://api.airpush.com/inappads/inappadcall.php*" Mendapatkan sebuah alamat atau *url* saat membuka aplikasi.
5	*Krep.itmtd.ywtjexf*	Hasil Analisis	"*https://play.googleapis.com/play/log*" Mendapatkan sebuah alamat atau *url* berbentuk *phising*.
6	*Earthquake*	Hasil Analisis	"*http://pari.securedapinetworks.com/api/input.php?type=Master&data=*" Mendapatkan sebuah alamat atau *url* saat membuka aplikasi.
7	Blackmart	Scan VirusTotal	44% terdeteksi sebagai *malware*.
8	*Krep.itmtd.ywtjexf*	Scan VirusTotal	61% terdeteksi sebagai *malware*.
9	*Earthquake*	Scan VirusTotal	0% terdeteksi sebagai *malware*.

5 Kesimpulan

Pada proyek akhir yang berjudul "Membangun Sistem Pengujian Keamanan Aplikasi Android Menggunakan MobSF" didapatkan kesimpulan sebagai berikut. Dalam melakukan analisis *malware* dapat dilakukan dengan berbagai tahap analisa untuk mendapatkan informasi tentang file yang dianalisis. Dari sampel *Blackmart* diperoleh informasi bahwa sampel tersebut merupakan sebuah *malware* dengan jenis *malware* bertipe *Adware*. Sedangkan untuk sampel *Krep.itmtd.ywtjexf* diperoleh informasi bahwa sampel tersebut merupakan sebuah *malware* dengan jenis *malware* bertipe *Trojan*. Dan yang terakhir sampel *Earthquake* bukan merupakan sebuah *malware* dikarenakan sampel tersebut merupakan sampel yang didapat dari *google play store* yang memiliki keamanan tinggi karena merupakan tempat untuk mengunduh aplikasi resmi untuk *android*. Hasil yang didapatkan dari *file* yang dianalisis berbeda-beda, tergantung dari sampel, metode analisis dan *tool*s yang digunakan.

Daftar Pustaka

[1] Habibi, Muhammad. 2017. Implementasi Layanan Deteksi *Malware* Berbasis *Android*. Retrieved from https://openlibrary.telkomuniversity.ac.id/pustaka/files/137447/jurnal_eproc/implementasi-layanan-deteksi-malware-berbasis-android.pdf

[2] Utomo. Yudha Aprianto. 2018. Membangun Sistem Analisis *Malware* Pada Aplikasi *Android* Dengan Metode *Reverse Engineering* Menggunakan *Remnux*. Retrieved from https://openlibrary.telkomuniversity.ac.id/pustaka/files/146794/ jurnal_eproc/membangun-sistem-analisis-malware-pada-aplikasi-android-dengan-metode-reverse-engineering-menggunakan-remnux.pdf.

[3] Muladi. Agus Dwi. 2018. *Membangun Sistem Analisis Malware Dengan Menggunakan FAME*. Bandung, *Open Library*.

[4] Shabtai, A. 2010. *Malware Detection on Mobile Devices. 11th International Conference on Mobile Data Management.*

[5] Aung, Z., Zaw, W. 2013. *Permission-Based Android Malware Detection. International Journal of Scientific & Technology Research*

[6] Pradana, Mada R. 2011. *Harmless Hacking Malware Analysis and Vulnerability Devolopment.*

[7] Sirkosi, Michael dan Andrew Honig. 2012. *Practical Malware Analysis. San Fransisco: William Pallock*

[8] Abraham, A. 2015. Mobile Security Framwork, https://github.com/MobSF/Mobile-Security-Framework-MobSF , [Diakses 3 Februari 2018].