ANALISIS MALWARE PADA SISTEM OPERASI ANDROID MENGGUNAKAN MEMORY FORENSICS BERDASARKAN API

MALWARE ANALYSIS IN ANDROID OPERATING SYSTEM USING MEMORY FORENSICS BASED ON API

Rifyandaru Wibisono, Avon Budiono, S.T., M.T.[2], Ahmad Almaarif., S.Kom., M.T.[3]

[1,2,3]Program Studi Sistem Informasi, Fakultas Rekayasa Industri, Universitas Telkom

[1]rifyandaruwibisono@student.telkomuniversity.ac.id , [2]avonbudi@telkomuniversity.ac.id, [3]ahmadalmaarif@telkomuniversity.ac.id

Abstrak

***Malware* adalah sebuah program yang berada pada sistem operasi dan dapat membahayakan sistem operasi yang terjangkit. Dalam perkembangan internet banyak sekali jenis *malware* yang dapat ditemukan dalam internet seperti *android malware*. *Android malware* adalah *malware* yang membuat pengaruh negatif pada sistem operasi android. *Malware* memiliki tujuan untuk merugikan pengguna pada sistem operasi yang terjangkit. Oleh sebab itu digunakanlah *malware analysis* untuk mengidentifikasi *malware*. *Malware analysis* adalah cara untuk mendapatkan informasi dari *malware* untuk mengatasi serangan terhadap korban yang terinfeksi. Dalam melakukan *malware analysis* bisa digunakan beberapa cara untuk mendeteksi *malware* seperti mendeteksi berdasarkan penggunaan *memory*. Dalam mendeteksi malware berdasarkan memory bisa digunakannya memory forensic untuk melakukan pendeteksian. Setelah mendapatkan hasil dari deteksi *malware* akan digunakannya sebuah cara untuk melakukan memberikan dampak untuk *malware* berdasarkan API. Dalam menggunakan *memory forensics* digunakan tools volatility untuk mendapatkan hasil analisis *malware* dan menggunakan *reverse engineering* dengan tools APKtools untuk mengetahui *malicious activity* berdasarkan penggunaan API dari aplikasi tersebut. Dalam penelitian ini digunakan 10 *malware* untuk dilakukan análisis menggunakan volatility dan APKtools untuk memberikan dampak mengunakan hasil dari análisis dan juga berdasarkan *malicious activity* dari API. Maka dari tu hasil dari penelitian ini adalah dampak yang berkaitan dengan API dan hasil análisis.**

Kata kunci : ***malware, malware analysis, memory, memory usage.***

Abstract

Malware is a program that is on an operating system and can endanger the affected operating system. In the development of the internet there are many types of malware that can be found on the internet such as Android malware. Android malware is malware that makes a negative influence on the Android operating system. Malware has the purpose of harming users of the affected operating system. Therefore malware analysis is used to identify malware. Malware analysis is a way to get information from malware to deal with attacks on infected victims. In doing analysis malware can be used several ways to detect malware such as detecting based on memory usage. In detecting malware based on memory you can use forensic memory to detect it. After getting the results of malware detection a method will be used to make an impact on malware based on API. In using memory forensics tools used volatility to get the results of malware analysis and use reverse engineering with APKtools tools to find out malicious activity based on the use of the API from the application. In this study 10 malware were used to analyze the use of volatility and APK tools to have an impact on using the results of analysis and also based on malicious activity from the API. So from the results of this study are the impacts related to the API and the results of analysis.

Keywords: ***malware, malware analysis, memory, memory forensics.***

1. Pendahuluan

Dalam era ini internet sudah menjadi sebuah hal yang sangat lumrah bagi manusia dikarenakan dapat membantu manusia dalam melakukan berbagai aktifitas. Banyak sekali komputer saling terhubung dengan komputer lainnya menggunakan internet. Internet memiliki banyak sekali hal positif yang dapat diambil dalam era ini tetapi internet juga memiliki hal yang negatif seperti cyber crime. Cycber crime adalah sebuah penyalahgunaan menggunakan internet untuk mendapatkan keuntungan ataupun menyebarkan malware menggunakan internet (Saini, Rao, 2012).

Malware adalah sebuah kode yang berada pada perangkat lunak yang dapat menyebabkan kerusakan pada fungsi sistem (McGraw, Morrisett, 2000). Malware tetap menjadi ancaman yang berbahaya dan kemunculannya dapat menyebabkan masalah pada korban yang terkena malware. Dikarenakan hal tersebut keamanan internet merupakan hal yang penting dalam era internet ini untuk menjaga semua aktifitas yang kita lakukan di internet.

Keamanan internet pada era ini telah berubah sebelumnya selalu berfokus pada perangkat yang ada pada komputer menjadi perangkat yang berada mobile (Milosevic, Malek, Ferrante, 2016). Perangkat mobile pada era ini merupakan perangkat yang terlemah sehingga dapat terjangkit malware khusus dengan mudah yaitu mobile malware (Symantec Corporation, 2015). Mobile malware terus naik dan mencapai 6 juta sample yang terdeteksi pada tahun 2014, naik 14% pada tahun yang sama (Mcaffe Labs, 2015).

Malware analysis merupakan sebuah cara untuk mendapatkan informasi dari malware untuk mengatasi serangan terhadap korban yang terinfeksi (Gutmann, 2007). Dari informasi yang didapat, infeksi malware bisa dikenali lewat signature dan dapat menggambarkan bagaimana cara sebuah malware itu bekerja. Ada dua cara untuk melakukan analisis malware yaitu static analisis dan dynamic analisis. Static analisis dapat digunakan untuk melakukan investigasi terhadap static features seperti Permission-Based. Biasanya static analisis digunakan pada perangkat lunak yang memiliki sumber daya yang minimal. Dynamic analisis juga merupakan sebuah pendekatan yang sangat efektif dengan melakukan percobaan pada malware yang akan diidentifikasi. Dynamic analisis pada mobile malware menggunakan cpu dan juga memory. Dalam menggunakan cpu dan memory pada dynamic analisis akan dimonitor cpu usage dan virtual memory untuk melakukan pendeteksian malware.

Selain itu, aka nada kasus dimana malware akan benar-benar diinjeksikan kedalam android. Analisis malware didalam perangkat android dilakukan dengan melakukan akuisisi terhadap memory dari android, kemudian hasil akuisisi memory akan dianalisis dengan menggunakan tools voloatility. Dalam hasil analisis ini bisa dibuktikan adanya kegiatan malware pada memory milik perangkat yang terjangkit. Diharapkan dengan adanya penelitian tentang malware pada smartphone ini, dapat memberikan informasi dan edukasi kepada pengguna bahwa terdapat ancaman pada perangkat android dengan cara malware melakukan injeksi terhadap memory pada android.

Memory Forensics adalah sebuah cara untuk mendapatkan bukti proses untuk mendapatkan suatu kejanggalan yang ada pada proses. Kelebihan dari *memory forensics* ini yaitu dapat mengetahui *malware* yang terhubung langsung dengan proses yang berada pada *memory*. Setelah mendapatkan proses yang mencurigakan akan dilakukan *reverse engineering* untuk mendapatkan API yang ada pada suatu proses.

API adalah beberapa kumpulan dari sebuah perintah untuk melakukan program aplikasi. Dalam API bisa digunakan untuk mendapatkan malicious activity dari suatu malware dan dalam melakukan analisis API adalah cara yang sangat efektif untuk mendapatkan bagaimana cara kerja malware (Alquraishi, Batarfi, 2017). Dengan menggunakan API bisa dilakukan analisis *malicious activity* dengan beberapa API yang didapatkan.

Berdasarkan data yang telah dianalisis tersebut maka hasil dari penelitian ini adalah dampak dari sample malware menggunakan API yang didapatkan sebelumnya dengan menggunakan *reverse engineering* dan bagaimana proses berjalan dalam suatu *environtment* yang ada. Dampak tersebut berdasarkan *malicious activity* dan proses menggunakan plugin *psxview* dan plugin *malfind*

2. Dasar Teori

2.1 *Malicious Software* (*Malware*)

Malware adalah software yang sengaja dirancang untuk membahayakan, menyusup, atau merusak sebuah komputer (Provos, Mavrommatis, Rajab, 2008). Software ini bisa menyebabkan kerusakan pada sebuah sistem operasi, memungkinkan pelaku untuk memiliki akses ke informasi rahasia dan sensitif serta memata-matai komputer korban.

Dalam menggunakan malware tersebut dengan mengirimkan malware tersebut kedalam internet dan akan menyerang target yang dituju akan mengalami bahaya yang sangat serius jika dibiarkan. Malware memiliki beberapa jenis berdarsarkan tingkah perilakunya seperti virus, worm, trojan horse, backdoor yang memiliki kriteria sendiri (Sharp, 2007). Berikut ini adalah beberapa jenis *malware* yang paling sering ditemui saat ini yaitu :

2.2 *Static Analysis*

Static analysis adalah metode malware analysis yang digunakan dengan cara mengamati secara langsung kode malware tanpa harus menjalankan malware tersebut. Dalam melakukan static analysis sebuah file akan diperiksa melalui tahapan dan metode yang telah dirancang sedemikian rupa untuk mampu mendapatkan informasi malware pada file yang diperiksa.

Static analysis bisa membantu bagaimana cara menemukan sebuah kode yang aneh pada suatu program yang terjangkit malware. Ada beberapa teknik yang digunakan untuk melakukan static analysis seperti file fingerprint dengan melihat binary dalam suatu program (Ghadiya, Bhasavar, 2013).

2.3 Dynamic Analysis

Dynamic analysis adalah sebuah proses analisis dimana kita akan menjalankan sample *malware* pada komputer lalu akan diamati bagaimana perilakunya(Ghadiya, Bhasavar, 2013). Dalam menjalankan *malware* tersebut akan digunakan environment yang aman lalu akan mengumpulkan informasi mengenai dampak yang terjadi ketika menjalankan prosesnya.

Dalam menggunakan analisis ini bisa digunakan berdarsarkan memory dan juga cpu dari perangkat keras yang terjangkit *malware*. Dalam menggunakan *memory* dan juga *cpu* kita dapat mengetahui hal aneh yang ada pada suatu program yang kita analisa.

2.4 Memory

Memory adalah sebuah tempat penyimpanan sebuah proses yang dapat disimpan secara sementara ataupun permanen (Ulrich Drepper, 2007). Memory menyimpan data yang sangat penting untuk diproses menggunakan *CPU* dan kapasistas yang besar untuk menyimpan data. Kapasitas memory yang ada adalah 8 bit, 16 bit, dan 32 bit.

Dalam melakukan analisa menggunakan *memory* kita bisa menganalisa menggunakan sebuah *virtual memory* untuk melihat kombinasi RAM dan juga ada sebagian harddrive dan melihat apakah terdapat kejanggalan. *Malware* biasanya menjalankan sebuah program dan dapat mempengaruhi *virtual memory* tersebut.

2.5 Metodologi Penelitian

Metode penelitian ini menunjukan alur dalam membuat suatu penelitian dan menentukan masalah yang terjadi terhadap suatu lingkungan. Metode penelitian ini harus memiliki pemahaman terhadap teknik penelitian Metode penelitian yang digunakan adalah model konseptual.

Model Konseptual adalah suatu hubungan antara faktor faktor yang memberikan dampak penting untuk melakukan penelitian dalam menganalisis suatu masalah. Dalam melakukan model konseptual dibuat kerangka menggunakan teori dan dibentuk secara terstruktur sehingga membuat suatu kesatuan untuk menyelasaikan masalah sesuai faktor yang ada.

Permasalahan dari penelitian ini adalah bagaimana *malware* bisa menjalankan aplikasinya dengan tidak terdeksi dengan teknik lainnya. Dengan permasalahan tersebut didapatkan bagaimana cara untuk mengidentfikasi sebuah *malware* untuk membantu computer pengguna. Cara yang dapat dilakukan adalah dengan melakukan analisis dengan menggunakan *memory.*

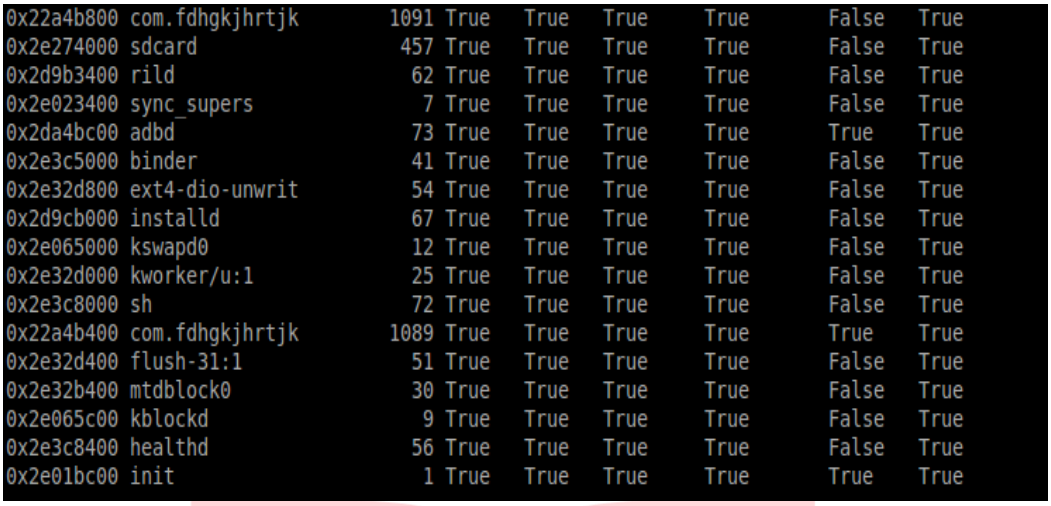
Dengan permasalahan yang ada pada penelitian ini akan diberikan cara solusi untuk mendapatkan hasil dari analisis dengan menggunakan *memory forensics.* Untuk meghasilkan analisis tersebut akan digunakan sebuah tools volatility dan APKtools untuk mencari hasil analisis dari suatu proses dan *malicious activity.* Cara yang digunakan adalah dengan mengunakan teknik *static analysis.*

Analisis yang akan dihasilkan yaitu berupa hasil referensi dari setiap proses dan juga kode yang telah diinjeksikan pada suatu *memory range* dan *malicious activity* dari API.

3. Pengujian Sistem dan Analisis

3.1 Pengujian Menggunakan Plugin Psxview

Plugin psxview hasil yang kita dapatkan berfungsi untuk memberikan referensi proses dari berbagai sumber dan dapat menemukan proses yang tersembunyi pada sistem.

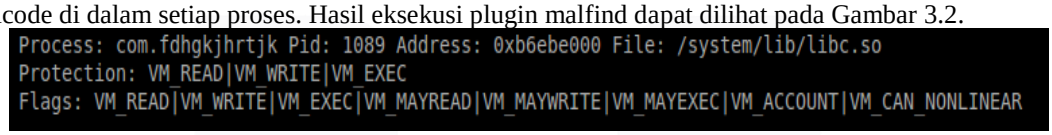


Gambar 3-1 Hasil Psxview

Gambar 3-1 Terlihat pada gambar diatas ini adalah sample2.apk yaitu dengan nama package com.gone60 dengan memiliki referensi proses yaitu label PSLIST, PSSCAN, PID_HASH dan LEADERS yang bernilai true yang berarti proses yang dimilik sample ini terlihat di semua proses pencarian. Salah satu tanda jika sebuah proses tersembunyi adalah nilai pada label PSLIST bernilai FALSE dan dapat dicurigai sebagai sebuah malware, namun pada kasus ini, sample ini belum terlihat sebagai malware.

3.2 Pengujian Menggunakan Plugin Malfind

Dalam menggunakan plugin malfind hasil yang didapat adalah sebuah file executable berbahaya dan shellcode di dalam setiap proses. Hasil eksekusi plugin malfind dapat dilihat pada Gambar 3.2.

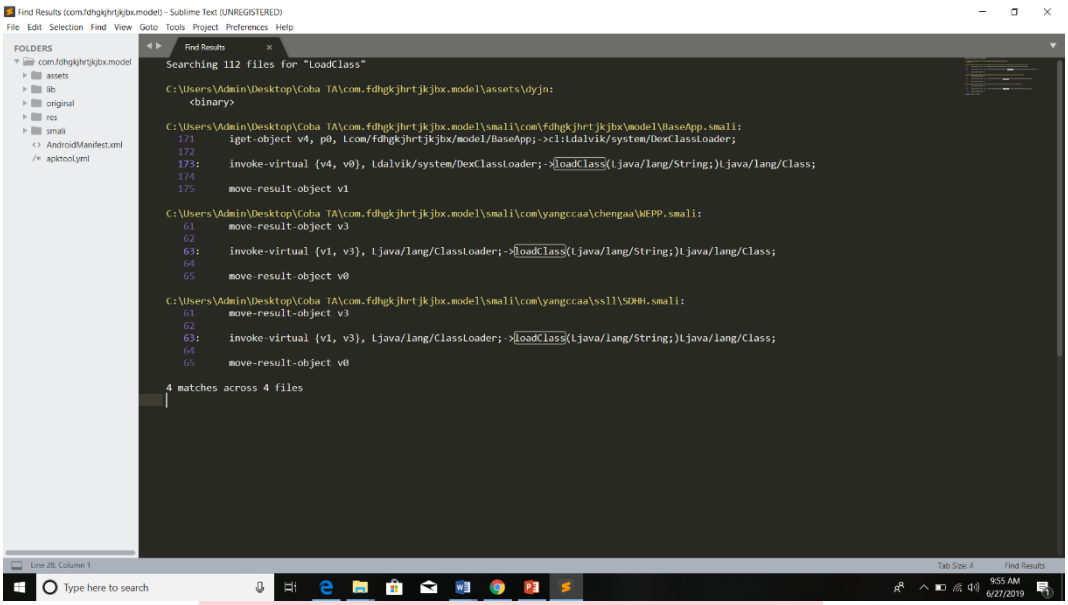


Gambar 3-2 Hasil ShowString

Gambar 3-2 adalah hasil eksekusi plugin linux_malfind, kita menemukan bahwa target dari proses yang telah dinjeksikan oleh kode dengan PID 532, memiliki wilayah memori pada alamat 0x0xa88d9000 yaitu berbentuk file /dev/ashmem/dalvik-jit-code-cache yang dapat dibaca, ditulis, dan dieksekusi. Hal ini terlihat dari vm_flags yang terdapat pada bagian protection dan flags. Vm_flags adalah bit-bit area yang menentukan perilaku dari sebuah *memory* dan memberikan informasi pada setiap halaman *memory*. Pada area memori dengan alamat 0xa88d9000 ini, vm_flags yang digunakan adalah VM_Read yang berarti halaman ini dapat dibaca, VM_Write yang berarti halaman ini dapat ditulis, VM_Exec yang berarti halaman dapat dieksekusi, VM_Mayread yang berarti *flags* VM_Read dapat diatur, VM_Mywrite yang berarti *flags* VM_WRITE dapat diatur, VM_Mayexec yang berarti *flags* VM_Exec dapat diatur, VM_Account yang berarti dalam memori ini adalah sebuah objek yang memiliki akun, VM_Can_Nonlinier yang berarti dalam memori ini adalah memiliki pemetaan yang nonlinear. Dengan adanya perintah ini VM_Read, VM_Write dan VM_Exec yang berarti memori dapat dibaca, ditulis dan dieksekusi dalam sebuah memori.

3.3 Pengujian Menggunakan APKtools

Dalam menggunakan APKtools hasil yang didapat dalam melakukan *reverse engineering* ini adalah mendapatkan fungsi dari sebuah kode yang telah dinjeksikan terhadap *memory address*. Dengan mendapatkan fungsi tersebut bisa diketahui bagaiman kode yang telah diinjeksi ini berkerja.



Gambar 3-3 Hasil APKtools

Pada Gambar 3-3 didapatkan merupakan informasi dari metode yang berhasil didapatkan dengan menggunakan APKtools. Metode yang didapatkan bisa diketahui dengan mencari berasal dari dex file yang telah kita reverse enginering. Dari sample tersebut terdapat beberapa metode seperti LoadLibary, dan LoadClass. LoadClass adalah sebuah fungsi menjalankan class dari file yang berbentuk .jar dan juga .apk yang berasal dari dari classes.dex. Method ini biasanya dijalankan untuk mengeksekusi kode yang bukan bagian dari aplikasi.

3.4 Hasil Analisis dengan Plugin Psxview, malfind dan APKtools

Dibawah ini adalah hasil dari plugin psxview yang telah kita dapat dari hasil pengujian sebelumnya dengan menggunakan 10 sample malware.

Tabel 3 - 1 Hasil plugin *psxview*

Sample	PID	Pslist	Psscan	Kmem_chace	Parents
Sample1.apk	1025	True	True	True	False
Sample2.apk	731	True	True	True	False
Sample3.apk	1089	True	True	True	True
Sample4.apk	843	True	True	False	True
Sample5.apk	994	True	True	False	True
Sample6.apk	1236	False	True	False	False
Sample7.apk	998	True	True	False	True
Sample8.apk	993	True	True	False	True
Sample9.apk	754	True	True	False	True
Sample10.apk	846	True	True	False	True

Dari Tabel V - 1 diatas terdapat 10 sample memory yang terdeteksi dengan menggunakan tools dengan volatility. Referensi proses yang telah terdeksi dengan tools tersebut memiliki proses yang sedang dijalankan. Sample tersebut memiliki beberapa yang proses yang bisa untuk diinvestigasi lebih lanjut. Dari sample yang telah dianalisis terdapat 4 referesi proses yang telah ditampilkan untuk semua proses agar bisa untuk diinvestigasi. Berikut jumlah referensi proses yang telah dianalisis dari sample malware tersebut dan memiliki referensi proses yang bisa dibaca.

Dibawah ini adalah hasil dari plugin malfind yang telah kita dapat dari hasil pengujian sebelumnya dengan menggunakan 10 sample malware. Berikut ini adalah analisis yang didapatkan dengan menggunakan plugin malfind.

Tabel 3 - 2 Hasil plugin *malfind*

No	PID	Protection	Flags	Address	File
1	1025	VM_Read|VM_Write|VM_Exec	VM_Read|VM_Write|VM_Exec|VM_Mayread|VM_Maywrite|VM_Mayexec|VM_Account	0xa8740000	Anonymous Mapping
2	731	VM_Read|VM_Write|VM_Exec	VM_Read|VM_Write|VM_Exec|VM_Mayread|VM_Maywrite|VM_Mayexec|VM_Accout|VM_Nonlinier	0x5fd47000	Dev/ashmem/dalvik-jit-code-cache
3	1089	VM_Read|VM_Write|VM_Exec	VM_Read|VM_Write|VM_Exec|VM_Mayread|VM_Maywrite|VM_Mayexec|VM_Accout|VM_Nonlinier	0xb6ebe000	System/lib/libc.so
4	843	VM_Read|VM_Exec	VM_Read|VM_Write|VM_Exec|VM_Mayread|VM_Maywrite|VM_Mayexec|VM_Account	0x36c16000	Anonymous Mapping
5	994	VM_Read|VM_Write|VM_Exec	VM_Read|VM_Write|VM_Exec|VM_Mayread|VM_Maywrite|VM_Mayexec|VM_Account	0x2790a000	Anonymous Mapping
6	1236	VM_Read|VM_Write|VM_Exec	VM_Read|VM_Write|VM_Exec|VM_Mayread|VM_Maywrite|VM_Mayexec|VM_Accout|VM_Nonlinier	0xb5b78000	Anonymous Mapping
7	998	VM_Read|VM_Write	VM_Read|VM_Exec|VM_Mayread|VM_Maywrite|VM_Mayexec|VM_Accout	0x3df5f000	Anonymous Mapping
8	993	VM_Read|VM_Write|VM_Exec	VM_Read|VM_Write|VM_Exec|VM_Mayread|VM_Maywrite|VM_Mayexec|VM_Accout|VM_Nonlinier	0xb627e000	Anonymous Mapping
9	754	VM_Read|VM_Write|VM_Exec	VM_Read|VM_Exec|VM_Mayread|VM_Maywrite|VM_Mayexec|VM_Accout	0x5820a000	Anonymous Mapping
10	846	VM_Read|VM_Write|VM_Exec	VM_Read|VM_Exec|VM_Mayread|VM_Maywrite|VM_Mayexec|VM_Accout	0x3da0a000	Anonymous Mapping

Dari Tabel diatas terdapat 10 sample memory yang terdeteksi dengan menggunakan tools dengan volatility. Dari hasil eksekusi plugin kita menemukan bahwa proses yang ditemukan oleh linux_malfind terindikasi berbahaya yaitu proses dengan alamat address, flags, protection dan file. Disini kita bisa melihat bagaiman kode yang telah melakukan injeksi tersebut dalam berprilaku dengan melihat flags dan juga protection.

Dibawah ini adalah hasil dari penggunaan APKtools yang telah kita dapat dari hasil pengujian sebelumnya dengan menggunakan 10 sample malware. Berikut ini adalah analisis yang didapatkan dengan menggunakan plugin APKtools.

Tabel 3-3 Hasil Analisis API

No	PID	Address	API
1	1025	0xa8740000	- LoadLibrary - getRuntime
2	731	0x5fd47000	- getDeviceId - LoadLibary
3	1089	0xa874000	- LoadClass - LoadLibrary
4	843	0x36c16000	- sendTextMessage - getDeviceId - getNetworkOperator - getSimCountryIso - getSimOperator
5	994	0x2790a000	- sendTextMessage - getDeviceId - getNetworkOperator - getSimCountryIso - getSimOperator
6	1236	0xb5b78000	- LoadLibrary
7	998	0x3df5f000	- sendTextMessage - getDeviceId - getNetworkOperator
8	993	0xb627e000	- sendTextMessage - getDeviceId - openConnection
9	754	0x5820a000	- loadLibrary - getRuntime - getSubscriberId - getLine1Number - getDeviceId - sendTextMessage - getNetworkOperator
10	846	0x3da0a000	- getSubscriberId - getSimSerialNumber - getDeviceId - openConnection - sendTextMessage

Dari Tabel diatas terdapat 10 sample memory yang terdeteksi dengan menggunakan tools dengan APKtools. Dari hasil tools APKtools bisa diketahui perilaku sebuah malware yang sedang berjalan dalam sebuah environtmet yang telah disiapkan. Didapatkan juga method yang digunakan dalam malware ini dalam melakukan kode yang telah dinjeksikan terhadap memory yang sedang berjalan ini. Berikut adalah penjelasan dari method yang telah didapat dari sample malware.

3.5 Dampak Hasil Malware

Dari hasil analalisis diatas bisa didapatkan malicious activity dari sebuah malware dengan menggunakan reverse engineering. Dari hasil tersebut bisa kita simpulkan dengan memberikan dampak terhadap malware tersebut.

Tabel 3 - 3 Hasil dampak *malware*

No	Sample	Malicious Activity	Dampak
1	Sample1.apk	Malware ini dapat melakukan link libabry kedalam sebuah proses tanpa sepengetahuan user. Sehingga malware ini bisa menghindari sebuah forensic yang sedang dijalankan dikarenakan dapat menyembunyikan diri. Dan menjalankan perintah shell yang tidak diketahui oleh pengguna.	Malware ini dapa melakukan tindakan yang dapat merugikan pengguna dikarenakan dapat menyembunyikan cara mereka bekerja tersebut sehingga sulit untuk dilakukan penelitian yang lebih menjauh. Tindakan yang dapat dilakukan adalah dengan menjalankan antivirus atau antimalware untuk dilakukan scanning secara berkala supaya tidak terjadi hal yang merugikan.
2	Sample2.apk	Malware ini dapat melakukan link libabry kedalam sebuah proses tanpa sepengetahuan user. Sehingga malware ini bisa mengindari sebuah forensic dikarenakan dapat menyembunyikan diri. Dan malware dapat mencuri data pribadi dari pengguna dan mengirimkannya ke remote server.	Malware ini dapat melakukan tindakan yang dapat merugikan pengguna dikarenakan dapat menyembunyikan cara mereka bekerja tersebut sehingga sulit untuk dilakukan penelitian yang lebih menjauh. Malware ini dapat mengambil data berupa IMEI dan juga CDMA. Tindakan yang dapat dilakukan adalah dengan tidak memberikan permission pada malware yang mencurigakan.
3	Sample3.apk	Malware ini dapat melakukan dan berinterkasi terhadap *environtment* yang digunakan dan memberikan link library kedalam sebuah proses tanpa sepengetahuan user agar bisa menjalankan kode yang berbahaya. Sehingga malware ini bisa mengindari sebuah forensic dikarenakan dapat menyembunyikan diri. Dan menjalan kode tersebut melalui file yang bukan bagian dari aplikasi.	Malware ini dapa melakukan tindakan yang dapat merugikan pengguna dikarenakan dapat menyembunyikan cara mereka bekerja dengan menyembunyikan eksekusi perintah cara tersebut sehingga sulit untuk dilakukan penelitian yang lebih menjauh. Tindakan yang dapat dilakukan adalah dengan mencegah untuk tidak menginstall aplikasi yang mencurigakan.
4	Sample4.apk	Malware ini dapat mengatur berbagai macam operasi sms dan mengirimkan berbagai macam sms kepada nomor yang dimiliki oleh pengguna tanpa sepengetahuan user. Dan malware ini dapat mencuri data pribadi dari pengguna dan mengirimkannya ke remote server.	Malware ini dapa melakukan tindakan yang dapat merugikan pengguna dikarenakan dapat menyembunyikan cara mereka bekerja tersebut sehingga sulit untuk dilakukan penelitian yang lebih menjauh. Malware ini dapat mengambil data berupa IMEI, kode Negara ISO, data provider, IMSI dan juga CDMA tanpa sepengetahuan user. tidak

			memberikan permission pada malware yang mencurigakan.
5	Sample5.apk	Malware ini dapat mengatur berbagai macam operasi sms dan mengirimkan berbagai macam sms kepada nomor yang dimiliki oleh pengguna tanpa sepengetahuan user. Dan malware ini dapat mencuri data pribadi dari pengguna dan mengirimkannya ke remote server.	Malware ini dapa melakukan tindakan yang dapat merugikan pengguna dikarenakan dapat menyembunyikan cara mereka bekerja tersebut sehingga sulit untuk dilakukan penelitian yang lebih menjauh. Malware ini dapat mengambil data berupa IMEI, kode Negara ISO, data provider, IMSI dan juga CDMA tanpa sepengetahuan user. tidak memberikan permission pada malware yang mencurigakan.

4. Kesimpulan

4.1 Berhasil dalam melakukan analisis pada memory android dengan menggunakan tools Volatility dan memanfaatkan beberapa plugin yang tersedia yaitu malfind dan juga psxview.

4.2 Hasil dari Volatility menunjukkan adanya proses mencurigakan dari sample malware yang berjalan pada RAM. Proses yang dilakukan oleh volatility adalah plugin psxview dan juga plugin malfind. Proses mencurigakan ini terdeteksi oleh plugin linux_malfind. Proses mencurigakan tersebut berada pada file dan juga anonymous mapping.

4.3 Mendapatkan malicious activity dengan menerapkan reverse engineering untuk mengetahui aktifitas yang dilakukan oleh malware berdasarkan penggunaan API.

Daftar Pustaka:

[1] Bangerter, E. et al. (no date) 'Memory Based Dynamic Malware Analysis'.
Butty, L. (2012) 'Android Malware Analysis Based On Memory Forensics', pp. 1–6.

[2] Dunham, K. (2009) Mobile Malware Attacks and Defense, Mobile Malware Attacks and Defense. doi: 10.1016/B978-1-59749-298-0.X0001-8.

[3] Hsiao, H.-W., Lin, C. S. and Chang, S.-Y. (2009) 'Constructing an ARP attack detection system with SNMP traffic data mining', Proceedings of the 11th International Conference on Electronic Commerce - ICEC '09, p. 341. doi: 10.1145/1593254.1593309.

[4] Leeds, M., Keffeler, M. and Atkison, T. (2017) 'Examining Features for Android Malware Detection', Int'l Conf. Security and Management, pp. 217–223. Available at: http://csce.ucmss.com/books/LFS/CSREA2017/SAM9710.pdf.

[5] Milosevic, J., Malek, M. and Ferrante, A. (2016) 'A Friend or a Foe? Detecting Malware using Memory and CPU Features', Proceedings of the 13th International Joint Conference on e-Business and Telecommunications, 4(Icete), pp. 73–84. doi: 10.5220/0005964200730084.

[6] Nejati, S. et al. (2012) 'Modeling and analysis of CPU usage in safety-critical embedded systems to support stress testing', Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7590 LNCS, pp. 759–775. doi: 10.1007/978-3-642-33666-9_48.

[7] Sharp, R. (2009) 'An Introduction to Malware Classification of Malware', Network, pp. 1–28. doi: 10.1017/CBO9780511623806.