Penanganan Insiden Pada Malware Android

Dede Apriyandi
NIM 23215090
Tugas Makalah EL6115
Tahun 2016

Abstrak—Berangkat dari G DATA Mobile Malware Report, penetrasi perangkat mobil di dunia yang menggunakan sistem operasi Android pada kuartal 4 tahun 2015 menguasai 66 % dari sistem operasi yang digunakan oleh seluruh perangkat pintar di dunia. Tingginya penetrasi Android tentunya memicu tingginya minat pembuat malware. Laboratorium G Data mencatat 758.133 malware baru selama kuartal 4 tahun 2015. Sampai dengan kuartal 4 total malware baru Android yang ditemukan menembus angka 2,3 juta sehingga bisa dipastikan total malware baru Android di tahun 2015 mencapai sejarah baru yang belum di capai sebelumnya, menembus angka 2 juta malware Android baru dalam 1 tahun [1].

Berbagai jenis malware, seperti Botnet, Backdoor, rootkit, dan Trojan, menyerang ponsel pintar untuk melakukan kejahatan seperti penipuan, layanan penyalahgunaan, mencuri informasi, dan akses root. Pada umumnya, mereka memiliki beberapa karakteristik, seperti terus-menerus memindai Bluetooth untuk mempersingkat masa pakai baterai perangkat, mengakses GPS untuk mengirim informasi posisi ke Internet, dan menghambat komunikasi antara perangkat dan base station untuk melumpuhkan jaringan nirkabel [2].

Lalu bagaimana penanganan insiden yang muncul akibat dari berbagai macam malware Android. Oleh karena itu, makalah ini akan membahas mengenai bagaimana penanganan insiden pada malware Android.

Kata kunci—malware; android; penanganan insiden.

I. PENDAHULUAN

Angka penjualan smartphone berbasis sistem operasi Android telah menunjukkan adanya penjualan yang cukup signifikan. Penelitian terbaru menunjukkan jika ponsel yang memiliki sistem operasi Android ini mulai disukai oleh banyak masyarakat di benua Eropa hingga Cina. Data menunjukkan jika di Uni Eropa termasuk Jerman, Prancis, Inggris, Spanyol dan Italia, pangsa pasar ponsel Android yang ada di sana melonjak hingga 71 persen pada kuartal pertama tahun 2016 lalu, hingga 75,6 persen. Di lain sisi, Amerika Serikat bahkan terjadi lonjakan hingga 7,3 persen atau menjadi 65,5 persen. Ponsel Android yang ada di Cina juga naik hingga 6 persen, menjadi 77 persen [3].

G DATA *Mobile Malware Report* pun mendukung pernyataan di atas, dimana penetrasi perangkat mobil di dunia yang menggunakan sistem operasi Android pada kuartal 4 tahun 2015 menguasai 66 % dari sistem operasi yang digunakan oleh seluruh perangkat pintar di dunia. Tingginya penetrasi Android tentunya memicu tingginya minat pembuat malware. Laboratorium G Data mencatat 758.133 malware baru selama kuartal 4 tahun 2015. Sampai dengan kuartal 4 total malware baru Android yang ditemukan menembus angka 2,3 juta sehingga bisa dipastikan total malware baru Android di tahun 2015 mencapai sejarah baru yang belum di capai sebelumnya, menembus angka 2 juta malware Android baru dalam 1 tahun [1].

Melalui makalah penanganan insiden pada malware android ini, penulis mencoba mempelajari dan memaparkan bagaimana karakteristik malware android dan teknik untuk melakukan mitigasi malware android sebagai wujud penanganan insiden pada malware android.

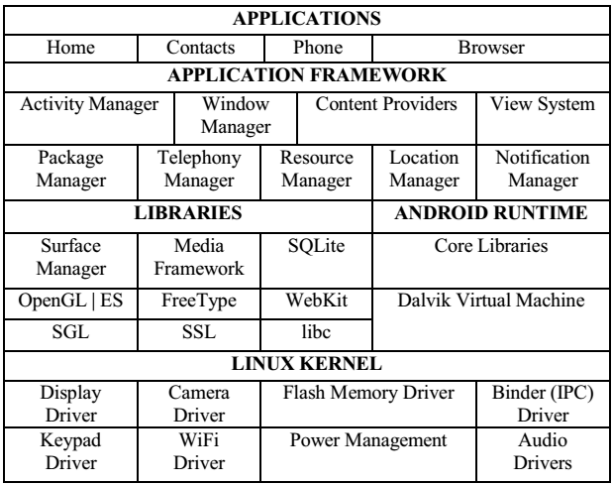
II. TINJAUAN

A. Android

Android merupakan sistem operasi *open source* berbasis pada OS Linux [6]. Pada Oktober 2015, versi terakhir yang secara resmi dirilis adalah android versi 6.0 (yang secara umum diketahui sebagai Android *Marshmallow* [7]. Dikembangkan dan didukung oleh Google, seluruh perangkat android memperbolehkan pengguna untuk melakukan akses *synchronize* ke penyimpanan dan pelayanan komunikasi yang disediakan oleh Google. Sebagai contoh, pengguna dapat *log-in* ke akun Gmail Google untuk mengecek email dan mengakses daftar kontak, kalender, dan aplikasi gratis lainnya secara otomatis. Android memperbolehkan pengguna untuk mengunduh dan memasang aplikasi baru untuk tujuan yang dibenarkan. Tempat umum untuk menemukan aplikasi adalah Google Play Store [8].

B. Sistem Operasi Android dan Fitur Keamanan

Kerangka OS Android memiliki beberapa layer untuk memberikan fasilitas berjalannya aplikasi [9]. Gambar 1 menunjukkan *snapshot* dari kerangka OS android. Layer paling bawah adalah kernel Linux. Di atas kernel, terdapat android *runtime* yang berisi *core libraries* (mencakup serangkaian inti *library* java) dan Dalvik *Virtual Machine* (Java *Virtual Machine* yang memberi kekuatan pada sistem android). Layer yang memiliki level yang sama dengan android *runtime* adalah *libraries* (set *library-library* dalam bahasa C/C++). Layer berikutnya *Application Framework*, yang mencakup program untuk mengatur fungsi-fungsi dasar *smartphone*. Layer teratas merupakan aplikasi itu sendiri. Di layer inilah ditemukan fungsi-fungsi dasar *smartphone* seperti telepon, SMS, *browser*, daftar kontak, dan lain-lain [10].



Gambar 1. Arsitektur Android [10]

Android memiliki beberapa mekanisme keamanan untuk melindungi data dan memori yang dimiliki oleh proses atau aplikasi yang berjalan pada perangkat [11]. Fitur keamanan inti yang akan dibahas, yaitu *sandbox, permission-based access control, secure Inter Process Communication (IPC), safe memory management,* dan *data encryption* [5].

Sandbox: Android mencegah aplikasi untuk mengakses konten data dan memori antar satu aplikasi dengan lainnya dan menjalankan tiap aplikasi di dalam *sandbox* (tiap proses memiliki salinan dari *virtual machine*). Hasilnya, aplikasi tidak dapat mengakses data dan kode aplikasi lainnya.

Permission-based access control: Izin yang diberikan oleh pengguna pada tiap aplikasi adalah dasar untuk memberikan atau melarang akses ke fitur sistem dan data pengguna. Saat instalasi aplikasi, izin yang dibutuhkan untuk beroperasi dideklarasikan dan pengguna diminta untuk memberikan atau menolak izin. Jika pengguna tidak memberikan izin, maka aplikasi tidak diinstal.

Secure IPC: Sebuah aplikasi tidak dapat secara langsung mengakses ruang memori aplikasi lainnya (yang mengandung data). Jadi, untuk mengakses data dari satu aplikasi ke aplikasi lain, mekanisme *Inter Process Communication* (IPC) memainkan fitur utamanya.

Safe memory management: Tiap aplikasi android berjalan di proses yang terpisah di dalam *instance* Dalvik aplikasi itu sendiri. Dalvik merupakan *register-based virtual machine* yang teroptimasi untuk memastikan perangkat dapat menjalankan beberapa *instances* secara efisien. Dalvik bertanggung jawab untuk manajemen memori dan proses saat *runtime* dan dapat menghentikan dan menghilangkan proses saat dibutuhkan. Manajemen memori terkait kerentanan seperti *buffer overflow*, kebocoran memori, dan penggunaan *pointer* yang tidak dikenali dieliminasi dengan menggabungkan beberapa teknologi yang terkenal seperti *Address Space Layout Randomization* (untuk mencegah serangan injeksi kode), NX (*non-executable stack* dikarenakan *buffer overflow*), dan *ProPolice* (mencegah korupsi pengembalian alamat ruang spasi).

Data encryption: Android memperbolehkan pengguna untuk mengenkripsi data mereka dan informasi profil lainnya. Mengenkripsi akun, aplikasi yang diunduh, file media, dan pengaturan dapat dilakukan. Perangkat yang terenkripsi dapat didekripsi berdasarkan *password* pengguna (saat setiap perangkat dinyalakan). Proses enkripsi menggunakan daya yang tidak sedikit saat pemrosesan dan waktu yang lebih lama.

C. Tipe malware

Malware atau "*malicious software*" diimplementasikan dengan tujuan berbahaya. Malware diinstal tanpa sepengetahuan pengguna (misalnya korban biasanya diminta melihat daftar izin yang dibutuhkan untuk menjalankan malware dan secara sukarela memberikan izin tanpa paham efek dari tindakan malware). Malware memiliki beberapa kategori yang terkenal, termasuk virus (program berbahaya uang dapat menyalin diri sendiri pada komputer yang terinfeksi), *worms* (mirip virus, kecuali memiliki kemampuan berkembang biak di mesin yang baru), dan *Trojan horses* (program yang menginstal *backdoor* di sebuah komputer yang terinfeksi untuk berkomunikasi dengan komputer yang dikendalikan *hacker*) [12].

Pada makalah ini, akan diklasifikasikan aplikasi malware yang dapat berjalan pada platform android (lihat tabel 1).

Tabel 1. Tipe malware, tindakan dan izin yang dibutuhkan

Tipe malware	Contoh tindakan	Izin
Mengubah desktop	Hal yang baru dan menghibur dengan mengubah	SET_WALLPAPER BIND_WALLPAPER

	wallpaper.	
Menjual *credentials* pengguna	Mengakses informasi kontak dari perangkat dan mengirimkan informasi tersebut via internet.	USE_CREDENTIALS READ_PROFILE MANAGE_ACCOUNTS
Panggilan dan SMS *Premium rate*	Menggunakan panggilan telepon dan pesan teks ke nomor premium dari perangkat korban.	READ_SMS WRITE_SMS SEND_SMS
Pengaturan ponsel berubah	Mengubah pengaturan seperti suara, kunci ponsel, dan pengaturan lainnya.	WRITE_SETTINGS
***Hacking* akun jaringan sosial**	Secara diam-diam mengakses dan memutakhirkan informasi profil pengguna.	READ_SOCIAL_STREAM WRITE_SOCIAL_STREAM

III. MENGENAL KARAKTERISTIK MALWARE ANDROID

Pada subbab ini, penulis memaparkan karakterisasi sistematik dari malware android yang ada, dimulai dari instalasi, aktivasi, sampai muatan berbahaya yang dibawanya [4].

A. Instalasi malware

Dari penelitian Yajin Zhou dan Xuxian Jiang [4], terdapat malware android yang melakukan instalasi pada telepon pengguna dan dikategorikan menjadi tiga teknik berdasarkan *social engineering*, yaitu *repackaging*, *update attack*, dan *drive-by download*.

1) ***Repackaging*** merupakan satu dari kebanyakan teknik yang digunakan pembuat malware untuk melakukan *piggyback* muatan berbahaya pada aplikasi populer (atau aplikasi sederhana). Pada dasarnya, pembuat malware mencari dan mengunduh aplikasi populer, membongkar aplikasi tersebut, memasukkan muatan berbahaya dan kemudian mengirimkan aplikasi yang telah diperbaharui tersebut ke pasar android resmi dan/atau alternatif.

2) ***Update attack*** merupakan teknik yang lebih sulit dideteksi dari teknik *repackaging*. Secara spesifik, teknik ini masih melakukan *repackage* pada aplikasi populer. Namun alih-alih memasukkan muatan berbahaya, teknik ini hanya mengandung komponen *update* yang akan mengambil atau mengunduh muatan berbahaya pada saat *runtime*.

3) ***Drive-by download*** merupakan teknik ketiga yang mengaplikasikan serangan tradisional melalui unduhan ke perangkat *mobile*. Meskipun tidak langsung mencari kerentanan *browser*, biasanya pembuat malware memikat pengguna untuk mengunduh aplikasi yang "menarik" atau "banyak fitur".

B. Aktivasi

Selanjutnya, dikaji *events* keseluruhan sistem android yang menarik bagi malware android. Dengan melakukan register ke *events* sistem terkait, malware android dapat bergantung pada dukungan *built-in* notifikasi *events* otomatis dan *callbacks* pada android untuk secara fleksibel memicu atau meluncurkan muatannya. Untuk mempermudah, kami menyingkat beberapa *events* android yang sering digunakan ke dalam tabel 2. Untuk setiap famili malware yang berkaitan dengan *events* juga dapat dilihat pada tabel tersebut.

Tabel 2. (Singkatan) ***Events*** **Android yang menarik bagi malware**

Singkatan	***Events***
BOOT **(Boot Completed)**	BOOT_COMPLETED
CALL **(Phone Events)**	PHONE_STATE NEW_OUTGOING_CALL
PKG **(Package)**	PACKAGE_ADDED PACKAGE_REMOVED PACKAGE_CHANGED PACKAGE_REPLACED PACKAGE_RESTARTED PACKAGE_INSTALL
SMS **(SMS/MMS)**	SMS_RECEIVED WAP_PUSH_RECEIVED
USB **(USB Storage)**	UMS_CONNECTED UMS_DISCONNECTED
BATT	ACTION_POWER_CONNECTED

(Power/Battery)	ACTION_POWER_DISCONNECTED BATTERY_LOW BATTERY_OKAY BATTERY_CHANGED_ACTION
NET **(Network)**	CONNECTIVITY_CHANGE PICK_WIFI_WORK
MAIN **(Main Activity)**	ACTION_MAIN
SYS **(System Events)**	USER_PRESENT INPUT_METHOD_CHANGED SIG_STR SIM_FULL

Dari semua *events* sistem, BOOT_COMPLETED merupakan yang paling digemari oleh malware android. Hal ini tidak mengejutkan karena *event* ini akan dipicu saat proses *booting* sistem selesai – waktu yang tepat untuk malware untuk memulai *background services*-nya. Urutan kedua yang paling digemari malware android adalah SMS_RECEIVED. Hal ini juga masuk akal karena banyak malware akan tertarik dalam melakukan *intercept* atau menanggapi pesan SMS yang masuk.

C. Muatan berbahaya

Melalui muatan yang dibawanya, malware android dapat dikategorikan menjadi 4 kategori, yaitu *privilege escalation, remote control, financial charges,* dan *personal information stealing*.

1) ***Privilege Escalation*** – *Platform* android merupakan sistem rumit yang terdiri tidak hanya dari kernel Linux, tetapi juga termasuk seluruh kerangka android dengan lebih dari 90 *open-sources libraries,* seperti WebKit, SQLite, dan OpenSSL. Kompleksitasnya secara alami menunjukkan kerentanan perangkat lunak yang berpotensi dieksploitasi untuk *privilege escalation*.

2) ***Remote Control*** – Ditemukan malware yang mampu mengubah ponsel yang terinfeksi menjadi *bots* untuk *remote control*. Banyak malware yang menggunakan trafik web berdasarkan HTTP untuk menerima perintah *bot* dari server C&C mereka. Beberapa famili malware juga secara diam-diam mengenkripsi URL dari *remote* server C&C sebagaimana mereka berkomunikasi dengan server C&C mereka.

3) ***Financial Charge*** – Di samping *privilege escalation* dan *remote control*, juga dilihat motivasi dibalik infeksi malware. Ternyata ditemukan bahwa malware secara sengaja menyebabkan kerugian finansial pada pengguna yang terinfeksi. Pada android terdapat fungsi *sendTextMessage* yang memungkinkan malware untuk mengirim pesan SMS tanpa sepengetahuan pengguna.

4) ***Information Collection*** – Ditemukan malware yang secara aktif mendapatkan berbagai informasi pada ponsel yang terinfeksi, termasuk pesan SMS, nomor telepon serta akun pengguna.

IV. TEKNIK PENANGANAN INSIDEN PADA MALWARE ANDROID

Pada bagian ini akan dibahas berbagai teknik penanganan insiden pada malware android yang tersedia pada literatur yang ada. Selain teknik, disediakan pula kelebihan dan kekurangan dari masing-masing teknik tersebut.

A. Sandboxing

Sandbox [13] menyediakan lingkungan eksekusi realistik namun dalam keadaan terisolasi. Sebagai hasilnya, potensial aplikasi berbahaya tidak memiliki efek pada lingkungan di luarnya. Hal ini sangat berguna tidak hanya untuk identifikasi tanda tangan, tetapi juga untuk disinfeksi malware. *Sandbox* memiliki dua tahap: analisis statis dan dinamis.

Pada analisis statis, *sandbox* melakukan dekompresi file instalasi dan membongkar file *executable* untuk identifikasi fragmen kode berbahaya. Pada khususnya, file manifest dan file dex diubah menjadi format yang dapat dibaca manusia. Kode kemudian discan untuk melihat pola mencurigakan. Pola mungkin mengandung: (i) memuat panggilan metode pemanggilan *library* bawaan menggunakan *System.loadLibrary()*, (ii) meminta perintah level sistem menggunakan metode pemanggilan *Runtime.getRuntime()* yang mengembalikan daftar direktori (“ls –l”), (iii) meminta pembangun kelas *String* menggunakan teknik refleksi, (iv) meminta pelayanan dari kelas *Activity* yang berjalan pada dibelakang, dan (v) memeriksa set dari izin dalam file *Manifest* yang mungkin terkait pada aktivitas berbahaya seperti SET_WALLPAPER (untuk pengaturan *wallpaper*), WRITE_SMS dan SEND_SMS (menulis dan mengirim pesan SMS), dan READ_PROFIL (membaca profil pengguna).

Fase analisis dinamis dari sistem *sandbox* dimaksudkan untuk memonitor sistem dan pemanggilan *library* dengan argumen. Secara umum, pemanggilan sistem adalah fungsi permintaan yang dibuat dari ruang spasi pengguna ke dalam kernel untuk meminta pelayanan atau sumber daya dari sistem operasi [5]. *Loadable kernel module* (LKM) diimplementasikan dan ditempatkan di lingkungan *emulator* android. Kernel yang dimodifikasi tetap mencatat fungsi pemanggilan yang diminta oleh aplikasi dan argumen mereka untuk dianalisis kemudian. Hal ini

memberikan urutan pemanggilan sistem level rendah bertanggung jawab untuk aktivitas berbahaya.

Kelebihan: *Sandbox* mengurangi pembangkitan dari tanda tangan berbasis pada pelacakan pemanggilan level sistem. Pada rata-rata, dibutuhkan 48 hari untuk mendapatkan tanda tangan dari malware baru, yang menyisakan *window* dari peluang kerusakan oleh malware [14].

Kekurangan: Sebagai level terendah dari pemanggilan sistem yang diintercept dan dicatat, implementasi dari *loadable kernel module* (LKM) merupakan tugas menakutkan dan rawan kesalahan. Jika perubahan level rendah dilakukan, *emulator* cenderung sangat tidak stabil, dan perhatian khusus dibutuhkan untuk memonitor keluaran.

B. Machine learning

Shabtai dkk. [15] mengaplikasikan teknik pembelajaran mesin untuk membedakan karakteristik aplikasi diantara dua kategori: alat dan permainan. Mereka mengekstrak fitur dari kode *byte* (file dex) dan XML (izin). Fitur pembelajaran digunakan untuk identifikasi tipe umum dari aplikasi, dimana dapat digunakan sebagai indikator untuk aktivitas berbahaya potensial.

Kelebihan: Pendekatannya otomatis dan dapat memungkinkan deteksi statis dari aplikasi malware.

Kekurangan: Bergantung pada tipe algoritma klasifikasi, kinerja akan bervariasi. Juga, akurasi pelatihan merupakan hal penting. Set data awal yang baik yang merepresentasikan seluruh tipe aplikasi dibutuhkan. Jika aplikasi cocok didalam kategori yang bertumpang tindih (misalnya aplikasi permainan harus mengirimkan informasi melalui internet untuk menyimpan skor dari pengguna online dimana mungkin mirip dengan aplikasi yang dimaksudkan untuk *browsing* di jaringan), kemudian pembelajaran mesin rawan terhadap peringatan *false positif* untuk aplikasi tidak berbahaya.

C. Decompiler-based static analysis

Enck dkk. [16] menganalisa banyak set aplikasi android yang dikoleksi dari pasar untuk identifikasi set dari pola aliran data, struktur, dan semantik. Pola aliran data mengidentifikasi apakah ada informasi data sensitif yang tidak boleh dikirim keluar (misalnya IMEI, IMSI, ICC-ID). Analisis struktur mencatat berbagai penggunaan API untuk mengambil informasi sensitif seperti ID perangkat atau manajer telepon. Analisis semantik melakukan argumen dari parameter metode pemanggilan. Sebagai contoh, saat pesan teks dikirim, akan dicek jika pesan dikirim ke nomor konstan atau dinamik. Sebelumnya mungkin merepresentasikan aktivitas aplikasi berbahaya. Observasi mereka dari aplikasi yang tampaknya tidak berbahaya dapat dipertimbangkan sebagai fitur untuk mengembangkan tanda tangan.

Kelebihan: Pendekatan ini sangan berguna dalam mengidentifikasi bagaimana informasi pengguna sebenarnya digunakan. Dalam mengetahui bagaimana informasi ditangani, kita dapat menyimpulkan sendiri mengenai bahayanya aplikasi.

Kekurangan: Catatan penulis bahwa ada beberapa perhatian logistik dan teknis [16] dengan metode khusus ini. Yang lebih penting lagi, metode ini tidak melakukan verifikasi aplikasi berbahaya secara jelas. Beberapa aplikasi bagus menggunakan praktis *coding* yang buruk, dan metode ini tidak dapat membedakan antara aplikasi bagus dan berbahaya.

D. Rule-based certification

Enck dkk. [17] mengembangkan teknik sertifikasi berdasarkan aturan dinamakan Kirin yang dapat mengecek kehadiran dari properti yang tidak diinginkan dalam aplikasi yang dicurigai sebagai malware. Pendekatan dimulai dari kebutuhan fungsionalitas umum dan kemudian analisis apakah izin yang dibutuhkan dapat membuat konflik operasi yang digunakan dalam operasi malware. Sebagai contoh. Aplikasi tidak boleh memiliki izin RECEIVE_SMS dan WRITE_SMS secara bersamaan. Kesuksesan dari proses sertifikasi bergantung pada tipe aturan spesifik oleh sistem dan kebutuhan.

Kelebihan: Pendekatan ini sesuai dengan prinsip desain android [17] dan mengirim hasil berhasil/gagal sederhana ke pengevaluasi. Sebagai tambahan, aturan yang dilanggar ditampilkan.

Kekurangan: Meskipun *interfaces* metode ini langsung dengan pelayanan keamanan [17], metode ini merupakan aplikasi *mobile*. Oleh karena itu, aplikasi harus dijalankan sepanjang waktu untuk melakukan verifikasi status dari aplikasi yang baru saja diunduh.

E. Binary code analysis

Batyuk dkk. [18] mengusulkan tidak hanya deteksi tanda tangan aplikasi berbahaya tetapi juga mengusulkan pendekatan mitigasi fleksibel. Mereka melakukan analisis statis pada kode biner dari aplikasi android (setelah dekompresi APK dan *decoding* kode byte Java ke dalam bahasa mesin Smali). Mereka mencari keberadaan APIs yang mungkin terkait dengan membaca informasi sensitif (misalnya IMEI atau pengidentifikasi perangkat, IMSI atau pengidentifikasi *subscriber*, nomor ponsel, ICC-ID atau nomor serial kartu SIM, menulis informasi ke keluaran *stream*) sebaik fungsionalitas untuk penggunaan pihak ketiga terkait dengan "Iklan" dan "Analitik". Pendekatan mitigasi dapat mengakomodasi kebutuhan pengguna; dapat menolak instalasi aplikasi berdasarkan laporan yang dibuat atau mengaplikasikan *patch* untuk mitigasi risiko keamanan potensial.

Kelebihan: Pendekatan ini menawarkan kompleksitas penyimpanan hasil ke dalam basis data non-relasi. Laporan kemudian dengan mudah diberikan dan ditampilkan dalam format terbaca.

Kekurangan: Metode ini cenderung lebih fokus pada bagaimana informasi digunakan daripada tingkah laku berbahaya aplikasi. Untuk dapat mengidentifikasi lingkup yang lebih besar dari aktivitas berbahaya, pendekatan ini akan ideal.

V. KESIMPULAN

Dalam makalah ini, kita meninjau dan menyoroti ancaman keamanan yang ditimbulkan oleh malware android. Secara khusus, kita fokus pada karakteristik yang ditemukan secara umum pada aplikasi malware dan memahami fitur yang memperbolehkan kita mendeteksi tanda-tanda bahaya. Kita juga mendiskusikan beberapa teknik pertahanan umum untuk penanganan insiden dari aplikasi malware termasuk kelebihan dan kekurangannya.

REFERENSI

[1] G DATA Software AG, *G DATA Mobile Malware Report*, Threat report: Q4/2015

[2] W. C. Hsieh, C. C. Wu, and Y. W. Kao, "A Study of Android Malware Detection Technology Evolution," in *The 49th Annual IEEE International Carnahan Conference on Security Technology (ICCST)*, 2015, pp. 135–140.

[3] Penjualan Android Naik, iPhone Malah Turun, diakses dari http://ekoran.co.id/penjualan-android-naik-iphone-malah-turun/4617/

[4] Y. Zhou, and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," in *IEEE Symposium on Security and Privacy*, 2012, pp. 95–109.

[5] V. N. Cooper, H. Shahriar, and H. M. Haddad, "A Survey of Android Malware Characteristics and Mitigation Tegniques," in *The 11th International Conference on Information Technology: New Generations*, 2014, pp. 327–332.

[6] A Guide to Android OS, diakses dari http://connect.dpreview.com/post/8437301608/guide-to-android-os.

[7] Android Marshmallow, diakses dari https://en.wikipedia.org/wiki/Android_Marshmallow.

[8] Google Play, diakses dari https://play.google.com/store?hl=en.

[9] A. Shabtai, Y. Feldel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer, "Google Android: A Comprehensive Security Assessment," *IEEE Security & Privacy*, March 2010, pp. 35-44.

[10] Ini dia Arsitektur Android yang wajib diketahui, diakses dari http://www.androidnajwa.net/2015/08/ini-dia-arsitektur-android-yang-wajib.html.

[11] Security Tips – Android Developers, diakses dari http://developer.android.com/training/articles/security-tips.html

[12] Perbedaan Malware, Virus, Trojan, Spyware, dan Worm, diakses dari https://www.maxmanroe.com/perbedaan-malware-virus-trojan-spyware-dan-worm.html

[13] T. Blasing, L. Batyuk, A. Schmidt, S. Camtepe, and S. Albayrak, "An Android Application Sandbox System for Suspicious Software Detection," *Proc. of 5th IEEE Malicious and Unwanted Software*, 2010, pp. 55-62.

[14] J. Oberheide, E. Cooke, and F. Jahanian. Cloudav: Nversion antivirus in the network cloud. *In Proceedings of the 17th USENIX Security Symposium (Security'08)*, San Jose, CA, July 2008.

[15] A. Shabtai, Y. Feldel, Y. Elovici, "Automated Static Code Analysis for Classifying Android Applications Using Machine Learning,"
Proc. of Intl. Conference on Computational Intelligence and Security, 2010, pp. 329-333.

[16] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A Study of Android Application Security," *Proc. of USENIX Security Symposium*, August 2011.

[17] W. Enck, M. Ongtang, and P. McDaniel, "On Lightweight Mobile Phone Application Certification," *Proc. 16th ACM Conf. Computer and Communications Security (CCS 09)*, ACM, 2009, pp. 235-245.

[18] L. Batyuk, M. Herpich, S. Camtepe, K. Raddatz, A. Schmidt, and S. Albayrak, "Using Static Analysis for Automatic Assessment and Mitigation of Unwanted and Malicious Activities within Android Applications," *Proc. of 6th International Conference on Malicious and Unwanted Software (MALWARE)*, October 2011, pp. 66-72.