ANALISIS AKTIVITAS *MALWARE* PADA RAM ANDROID DAN SANDBOX ENVIRONMENT



Disusun Oleh:

N a m a : Rizky Laksamana Putra

NIM : 13523094

PROGRAM STUDI TEKNIK INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2019

HALAMAN PENGESAHAN DOSEN PEMBIMBING

ANALISIS AKTIFITAS MALWARE PADA RAM ANDROID DAN SANDBOX ENVIRONMENT

TUGAS AKHIR

Disusun Oleh:

N a m a : Rizky Laksamana Putra
NIM : 13523094

Yogyakarta, 12 Desember 2018

Pembimbing

(Fietyata Yudha S.Kom., M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

ANALISIS AKTIVITAS *MALWARE* PADA RAM ANDROID DAN SANDBOX ENVIRONMENT

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Teknik Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 7 Januari 2019

Tim Penguji

Fietyata Yudha, S.Kom., M.Kom

Anggota 1

Nur Wijayaning Rahayu, S.Kom., M.Cs.

Anggota 2

Almed Hamzah, S.T., M.Eng.

Mengetahui,
Ketua Program Studi Teknik Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Rizky Laksamana Putra

NIM : 13523094

Tugas akhir dengan judul:

ANALISIS AKTIVITAS *MALWARE* PADA RAM ANDROID DAN SANDBOX ENVIRONMENT

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 7 Desember 2018



(Rizky Laksamana Putra)

HALAMAN PERSEMBAHAN

Segala puji bagi Allah Subhanahu Wa Ta'ala, atas limpahan rahmat dan karunia yang tiada hentinya kepada saya, sehingga saya masih dapat merasakan nikmat Iman, Islam, Rezeki serta Kesehatan setiap harinya tanpa kurang suatu apapun.

Shalawat serta salam senantiasa tercurahkan untuk junjungan kita Nabi Muhammad Shalallahu "Alaihi Wasalam yang telah membawa kita dari zaman yang gelap menuju kepada zaman yang terang benderang. Semoga kita termasuk orang-orang yang mendapat syafaat dari Nabi Muhammad Shalallahu 'Alaihi Wasalam kelak pada hari akhir. Tugas Akhir ini saya persembahkan kepada:

1. Orang Tua saya, Alm. Bapak Ir. H. Sugiharto serta Ibu Iriana Safitri S.H.,M.M yang selalu memberikan dukungan serta doa untuk saya.
2. Kepada Adik saya, Mellinda Salsabila Putri serta keluarga besar yang selalu ada dan memberikan nasihat serta dukungan untuk saya.
3. Kepada Sahabat dan Teman-Teman yang ada disaat suka maupun duka selama masa perkuliahan saya.
4. Bapak Fietyata Yudha S.Kom., M.Kom., selaku pembimbing Tugas Akhir saya

Penulis menyadari bahwa skripsi ini masih jauh dari kata sempurna, oleh karena itu penulis mengharapkan saran agar skripsi ini menjadi lebih baik lagi.

HALAMAN MOTO

"Sesungguhnya sesudah kesulitan itu ada kemudahan."

(QS. Alam Nasyroh: 6)

"Hardwork beats talent when talent doesn't work hard."

(Tim Tebow, Through My Eyes)

"Why So Serious?"

(Joker, from The Dark Knight)

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Alhamdulillahirobbilalamin, puji syukur kami panjatkan kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul ***"Analisis Aktivitas Malware pada RAM Android dan Sandbox Environment"***.

Laporan Tugas Akhir ini dibuat sebagai syarat untuk memperoleh gelar sarjana Teknik Informatika Universitas Islam Indonesia. Penulis menyadari bahwa dalam pelaksanaan Tugas Akhir dan penyusunan laporan ini tidak dapat lepas dari bimbingan, dukungan dan bantuan dari berbagai pihak. Oleh karena itu perkenankanlah penulis untuk mengucapkan terima kasih dan penghargaan setinggi-tingginya kepada:

1. Allah SWT karena atas karunia-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik dan semoga Tugas Akhir ini dapat memberikan manfaat di kemudian hari.
2. Orang Tua dan keluarga penulis atas segala doa dan dukungan selama penulis melaksanakan Tugas Akhir.
3. Bapak Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia
4. Bapak R. Teduh Dirgahayu, S.T., M.Sc., Ph.D., selaku Kepala Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia
5. Ibu Elyza Gustri Wahyuni, S.T., M.Cs., selaku Dosen Pembimbing Akademik di Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia
6. Bapak Fietyata Yudha, S.Kom., M.Kom., selaku Dosen Pembimbing Tugas Akhir di Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
7. Bimo Prakoso dan Nelvin Faidzil Kurnia, yang telah bersedia meminjamkan perangkat penelitian yang dibutuhkan dalam menyelesaikan Tugas Akhir ini.
8. Sahabat-sahabat yang tergabung dalam grup YOUNG EXECUTIVE, terima kasih untuk kebersamaan serta motivasi yang telah diberikan selama ini.
9. Sahabat-sahabat selama kuliah yang tergabung dalam grup CYBERDOTA, ROAD TO WIMBLEDON, FPS KOK GAK KELIATAN yang tidak bisa disebutkan satu per satu. Terima kasih untuk kebersamaan dan keceriaan yang telah dilewati bersama.
10. Teman-teman angkatan 2013 (ETERNITY) di Jurusan Teknik Informatika Universitas Islam Indonesia yang telah memberikan kenangan indah selama kuliah.

11. Semua pihak yang telah banyak membantu dalam pelaksanaan Tugas Akhir yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa laporan Tugas Akhir ini masih belum sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun agar Tugas Akhir ini menjadi lebih baik. Akhir kata, semoga laporan ini dapat bermanfaat dan atas segala bantuan yang telah diberikan semoga mendapat imbalan yang setimpal dari Allah SWT, Amin.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Yogyakarta, 7 Desember 2017

(Rizky Laksamana Putra)

SARI

Indonesia menjadi negara dengan pengguna *smartphone* yang terkena serangan *malware* ketiga di dunia. *Malware* merupakan sebuah *software* yang dapat menyusup ke sistem operasi sehingga dapat merusak sistem, juga dapat mencuri file-file penting yang ada pada sistem. beberapa oknum menggunakan *malware* untuk mencuri data-data penting dari pengguna dengan berbagai macam tujuan. Untuk mengetahui aktivitas yang dilakukan *malware,* perlu dilakukan analisis terhadap *malware*.

Analisis *malware* ini menggunakan dua teknik yaitu menggunakan metode *sandbox* dan analisis RAM dari perangkat android. Analisis menggunakan *sandbox* dilakukan dengan membangun lingkungan kerja atau *environment* dengan mesin linux untuk melakukan automasi analisis. Sedangkan, analisis pada RAM android dilakukan dengan mengakuisisi RAM milik perangkat android yang telah dipasang dengan sampel *malware,* kemudian dianalisis menggunakan *tools*.

Hasil dari analisis menggunakan *sandbox* menemukan bahwa sampel *malware* meminta izin yang berbahaya bagi pengguna saat memasang aplikasi, kemudian sampel *malware* memiliki muatan berbahaya yang tersembunyi, dan sampel *malware* telah dikategorikan sebagai *malware* oleh beberapa mesin antivirus. Sedangkan hasil analisis pada RAM menunjukkan aktivitas mencurigakan dalam RAM yang mengindikasikan sampel *malware* melakukan eksekusi kode berbahaya ke dalam sistem milik *smartphone* android.

Kata Kunci: *Malware, tools, smartphone*, Android, *sandbox*, RAM

GLOSARIUM

Sandbox	: mekanisme keamanan untuk memisahkan program yang sedang berjalan
Malware	: sebuah *software* yang dapat menyusup ke sistem operasi sehingga dapat merusak sistem, juga dapat mencuri file-file penting yang ada pada sistem
Compile	: proses untuk mengubah berkas kode program dengan berkas lain yang terkait menjadi berkas yang siap untuk dieksekusi oleh system operasi secara langsung.
Plugin	: fungsi atau fitur tambahan yang digabungkan ke sebuah sistem untuk menambah kemampuan dan kinerja sistem tersebut.

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI ix
GLOSARIUM x
DAFTAR ISI xi
DAFTAR TABEL xiii
DAFTAR GAMBAR xiv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan Penelitian 3
1.5 Manfaat Penelitian 3
1.6 Metode Penelitian 3
1.7 Sistematika Penulisan 4
BAB II TINJAUAN PUSTAKA 2
2.1 Penelitian Terdahulu 2
2.2 Landasan Teori 6
2.2.1 *Malware* 6
2.2.2 Sandbox 9
2.2.3 Cuckoo Sandbox 9
2.2.4 LiME (Linux Memory Extractor) 12
2.2.5 Volatility 13
BAB III METODE PENELITIAN 14
3.1 Alur Penelitian 14
3.1.1 Alur Penelitian Cuckoodroid 14
3.1.2 Alur Penelitian Volatility 16
3.2 Analisis Kebutuhan Sistem 19
3.2.1 Analisis Kebutuhan Perangkat Keras 19
3.2.2 Analisis Kebutuhan Perangkat Lunak 19
3.2.3 Analisis Kebutuhan Proses 21
3.3 Eksaminasi 22
3.4 *Malware* Analysis 23
BAB IV IMPLEMENTASI DAN HASIL PENGUJIAN 24
4.1 Implementasi Perangkat Lunak 24
4.1.1 Membangun Lingkungan Kerja Simulasi dan Analisis 24
4.1.2 Instalasi Cuckoo dan Cuckoodroid 24
4.1.3 Akuisisi RAM 26
4.1.4 Implementasi Analisis Volatility 27
4.2 Hasil dan Pembahasan 28
4.2.1 Hasil Analisis Cuckoodroid 28
4.2.2 Hasil Analisis Volatility 36

BAB V KESIMPULAN DAN SARAN 43
5.1 Kesimpulan 43
5.2 Saran 44
DAFTAR PUSTAKA 45
LAMPIRAN 47

DAFTAR TABEL

Tabel 4.1 Tabel Hidden Payload.....30
Tabel 4.2 Hasil Antivirus.....32
Tabel 4.3 Tabel proc_maps.....39
Tabel 4.4 Tabel Perbandingan Hasil Temuan Analisis.....41

DAFTAR GAMBAR

Gambar 1.1 Peringkat negara dengan serangan *malware* tertinggi 1

Gambar 2.1 Alur Analisis CuckooDroid 11

Gambar 3. 1 Alur Analisis Cuckoodroid 15
Gambar 3.2 Alur pembuatan kernel 16
Gambar 3.3 Proses mengambil ramdisk 16
Gambar 3.4 Proses pembuatan boot.img baru 17
Gambar 3.5 Proses pembuatan modul LiME 17
Gambar 3.6 Proses instalasi bootable image baru 17
Gambar 3.7 Proses pemindahan modul lime.ko 18
Gambar 3.8 Instalasi modul dan Akuisisi RAM 18
Gambar 3.9 Alur Akuisisi RAM 18

Gambar 4.1 Instalasi Java JDK 24
Gambar 4.2 Cuckoo Sandbox 25
Gambar 4.3 Cuckoo dan AVD 25
Gambar 4.4 Halaman website hasil analisis Cuckoo Sandbox 26
Gambar 4.5 make profile Volatility 27
Gambar 4.6 Membuat zip profil 27
Gambar 4.7 Melihat profil Volatility 28
Gambar 4.8 Sub Bagian Signatures 29
Gambar 4.9 Izin Aplikasi BeNews 31
Gambar 4.10 Hosts IP 33
Gambar 4.11 Kolom DNS 33
Gambar 4.12 IP Tracker 34
Gambar 4.13 Kolom TCP 34
Gambar 4.14 Kolom UDP 35
Gambar 4. 15 Hasil Analisis PacketTotal 35
Gambar 4.16 Contoh Payload 36
Gambar 4.17 Plugin linux_pslist 37
Gambar 4.18 Proses BeNews.apk 37

Gambar 4.19 Plugin linux_psxview.....37
Gambar 4.20 proses benews pada plugin psxview 37
Gambar 4.21 Plugin linux_malfind.....38
Gambar 4.22 Plugin linux_procmaps 39
Gambar 4.23 Tabel ARP.....40
Gambar 4.24 Tabel Ifconfig.....40
Gambar 4.25 proses dump route-cache.....40
Gambar 4.26 Tabel *cache routing* 41

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia menjadi negara dengan pengguna *smartphone* yang terkena serangan *malware* ketiga di dunia sebanyak 41,14%. Peringkat pertama adalah negara Iran dengan persentase pengguna *smartphone* yang terkena serangan *malware* sebanyak 57,25% (Gambar 1.1).

	Country*	%**
1	Iran	57.25
2	Bangladesh	42.76
3	Indonesia	41.14
4	Algeria	38.22
5	Nigeria	38.11
6	China	37.63
7	Côte d'Ivoire	37.12
8	India	36.42
9	Nepal	34.03
10	Kenya	33.20

Gambar 1.1 Peringkat negara dengan serangan *malware* tertinggi

Malware merupakan sebuah *software* yang dapat menyusup ke sistem operasi sehingga dapat merusak sistem, juga dapat mencuri file-file penting yang ada pada sistem (Yesi Novaria, 2014). Beberapa oknum menggunakan *malware* untuk mencuri data-data penting dari pengguna dengan berbagai macam tujuan. *Smartphone* dengan sistem operasi Androidpun tidak luput dari serangan *malware*. Umumnya *malware* bekerja pada RAM milik perangkat, hal ini terjadi karena RAM memiliki data dan file-file yang penting dan spesifik sehingga para penjahat menargetkan informasi dalam RAM milik pengguna. Cara kerja *malware* juga beragam, oleh karena itu diperlukan analisis lebih lanjut terhadap *malware* tersebut. Analisis *malware* dibagi menjadi dua yaitu *static analysis* dan *dynamic analysis*.

Dalam melakukan analisis, hal yang pertama kali dilakukan adalah membangun sebuah *sandbox*. Sandbox adalah sebuah mekanisme keamanan untuk memisahkan program yang sedang berjalan. Sandbox ini sering digunakan untuk mengeksekusi kode yang belum teruji,

atau program yang tidak dipercaya dari pihak ketiga yang tidak diverifikasi, pemasok, pengguna yang tidak dipercaya dan situs web yang tidak dipercaya. Tools yang akan digunakan dalam *sandbox*ing yaitu Cuckoo *sandbox*. Cuckoo *sandbox* memungkinkan proses *dynamic analysis* yang dapat dimonitor secara *real-time*.

Selain itu, akan ada studi kasus dimana *malware* akan benar-benar diinjeksikan kedalam perangkat android. Analisis *malware* didalam perangkat android dilakukan dengan melakukan akuisisi terhadap RAM dari perangkat, kemudian hasil akuisisi RAM tersebut dianalisis dengan menggunakan *tools*. Hasil analisis ini nantinya akan menjadi perbandingan bagaimana cara kerja *malware* pada perangkat. Hasil analisis ini juga ingin membuktikan adanya kegiatan *malware* pada RAM milik perangkat yang terjangkit. Diharapkan dengan adanya penelitian tentang *malware* pada *smartphone* ini, dapat memberikan informasi dan edukasi kepada pengguna *smartphone* tentang ancaman yang mengintai dibalik aplikasi jika aplikasi tersebut adalah *malware* dan agar lebih waspada terhadap aplikasi yang akan di pasang kedalam *smartphone* mereka.

1.2 Rumusan Masalah

Dari uraian latar belakang diatas, perumusan masalah untuk penelitian yang akan dilakukan adalah:

a. Bagaimana cara melakukan analisis terhadap *malware* android yang dijalankan didalam Cuckoo Sandbox?
b. Bagaimana teknik melakukan akuisisi RAM pada perangkat Android?
c. Bagaimana melakukan analisis terhadap RAM perangkat android yang telah diakuisisi?
d. Bagaimana hasil perbandingan analisis antara *malware* yang berjalan pada sandbox dengan *malware* yang berjalan pada RAM?

1.3 Batasan Masalah

Berdasarkan rumusan masalah diatas, peneliti membataskan masalah ini pada:

a. Perangkat yang diinjeksi *malware* adalah perangkat dengan sistem operasi android.
b. Lab sandbox yang digunakan adalah CuckooDroid Sandbox.
c. Tools yang digunakan untuk melakukan akuisisi RAM dari perangkat android adalah LiME (Linux Memory Extractor).
d. Tools yang digunakan untuk melakukan analisis terhadap RAM yang telah diakuisisi adalah Volatility.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai penelitian ini adalah:

a. Membangun lingkungan penelitian berbasis Linux untuk menjalankan simulasi dan analisis menggunakan Cuckoo Sandbox yang terintegrasi dengan emulator android.
b. Mengimplementasikan teknik akuisisi RAM menggunakan tools LiME melalui koneksi port TCP dengan bantuan kabel USB.
c. Melakukan analisis terhadap RAM perangkat android menggunakan tools Volatility dengan memanfaatkan plugin yang tersedia.
d. Membandingkan hasil analisis dari eksekusi *malware* pada perangkat virtual yang memanfaatkan *sandbox* dan eksekusi *malware* pada perangkat android asli yang dianalisis melalui RAM untuk mencari persamaan dan perbedaan dari informasi yang didapatkan oleh kedua perangkat tersebut.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini antara lain:

a. Mengetahui adanya aktifitas *malware malware* pada RAM Android.
b. Mengetahui cara kerja *malware* dari proses analisis melalui sandbox dan RAM yang diakuisisi dari perangkat android yang terjangkit *malware*.
c. Berkontribusi dalam bidang penelitian forensics sebagai dokumentasi dan implementasi proses dimasa mendatang.

1.6 Metode Penelitian

Metode penelitian yang digunakan untuk melakukan penelitian ini diantaranya adalah:

a. Studi pustaka, yaitu teknik pengumpulan data dengan cara membaca dan mempelajari tentang analisis *malware*, sandbox, android forensic dan hal-hal yang berkaitan dengan penelitian ini. Sumber dari studi pustaka dapat berupa buku, paper, jurnal, makalah, dan referensi keilmuan lainnya.
b. Membangun Environtment dengan cara menginstall modul dan library yang dibutuhkan oleh tools agar bisa dijalankan untuk melakukan analisis *malware*. Modul dan library ini sangat penting karena kebutuhan dari tiap tools berbeda-beda.

c. Melakukan akuisisi RAM pada perangkat android yang terjangkit *malware*. Juga menyiapkan sandbox untuk melakukan analisis terhadap sampel *malware* yang diinjeksi kedalam perangkat.
d. Analisis data. Kegiatan analisis dilakukan terhadap hasil yang didapatkan dari eksekusi sampel *malware* pada sandbox dan RAM perangkat.
e. Laporan. Pada tahap ini, semua temuan yang didapat selama proses analisis akan dimasukkan kedalam laporan akhir.

1.7 Sistematika Penulisan

Sistematika penulisan penelitian disusun untuk memberikan gambaran umum tentang penelitian yang dijalankan. Sistematika penelitian ini adalah sebagai berikut:

BAB I PENDAHULUAN

Pada bagian ini berisi tentang latar belakang permasalahan, identifikasi masalah, menentukan batasan masalah yang akan dibahas, menjabarkan tujuan dan manfaat dari penelitian ini, asumsi metodologi penelitian, serta sistematika penulisan.

BAB II LANDASAN TEORI

Pada bagian ini berisi tentang teori yang digunakan sebagai landasan untuk menyelesaikan permasalahan yang diangkat pada penelitian ini. Bahasan dalam bagian ini mengenai pembahasan teori dasar yang digunakan dalam penelitian, terkait *malware* pada android.

BAB III METODOLOGI

Pada bab ini berisi tentang objek dan jenis penelitian, data dan sumber data, teknik pengumpulan data.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini berisi tentang pembahasan hasil setelah proses akuisisi RAM dan eksekusi *malware* pada sandbox dilakukan. Pembahasan berisi tentang temuan dari rangkaian analisis kedua tools tersebut.

BAB V KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan yang menjelaskan tujuan penelitian dapat tercapai serta menjelaskan kelebihan dan kekurangan dari penelitian ini. Sedangkan saran, berisi hal-hal yang dapat dilakukan sebagai kontribusi terhadap penelitian ini dikemudian hari.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Dalam jurnal berjudul ”*Accurate mobile malware detection and classification in the cloud”* (Xiaolei Wang, Yuexiang Yang & Yingzhi Zeng, 2015), peneliti mengusulkan sebuah sistem deteksi *malware* berdasarkan CuckooDroid, yang merupakan fitur dari Cuckoo Sandbox untuk melakukan analisis *malware* android menggunakan analisis statis dan dinamis. Sistem yang diusulkan ini terdiri dari dua bagian yaitu mesin mendeteksi keanehan yang ditimbulkan oleh aplikasi yang mencurigakan menggunakan analisis dinamis, dan mendeteksi keaslian *malware* dan klasifikasi *malware* menggunakan kombinasi dari analisis statis dan dinamis. Penelitian dilakukan menggunakan 5560 sampel *malware* dan 6000 sampel tidak berbahaya. Dengan prtimbangan sumber daya komputasi intensif yang diperlukan oleh analisis statis dan dinamis, sistem yang diusulkan harus digunakan diluar perangkat android, seperti di Cloud. Pasar aplikasi dan pengguna biasa dapat mengakses sistem untuk mendeteksi *malware* melalui layanan cloud.

(Joe Sylve, Andrew Case, Lodovico Marziale, Golden G.Richard, 2011) dalam jurnalnya yang berjudul “*Acquisition and analysis of volatile memory from android devices”* melakukan eksplorasi terkait masalah teknis yang berhubungan dengan akuisisi tangkapan fisik memori dari perangkat berbasis android serta analisis lanjutan dari data yang berhasil didapat. Mereka menyajikan sebuah metodologi untuk mengakuisisi keseluruhan memori dari android, kode untuk menganalisis struktur data kernel, dan skrip yang memungkinkan analisis dari sisi pengguna dan aktifitas *file* berdasarkan *system-based.* Mereka percaya bahwa android akan terus membutuhkan riset forensik di masa depan dan untuk menunjang agar hasil yang didapatkan dapat langsung digunakan oleh peneliti dan investigator, mereka telah terintegrasi dengan framework Volatility Memory Analysis (Volatility, 2011).

Dalam studi kasusnya, mereka menggunakan perangkat android yaitu HTC EVO 4G yang sudah dalam keadaan *root*. Kemudian akuisisi memori dilakukan melalui TCP dengan menggunakan kernel DMD (yang saat ini berubah nama menjadi LiME). Setelah memori berhasil diakuisisi, hasil tangkapan memori dianalisis menggunakan plugin milik Volatility.

Berikut ini beberapa penelitian terdahulu lainnya dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No.	Judul	Peneliti	Pencapaian
1.	An Adaptive Dataset for the Evaluation of Android *Malware* Detection Techniques	(Hreirati, Iqbal, & Zulkarnine, 2018)	Penelitian ini menggunakan 5000 sampel aplikasi (4500 berbahaya dan 500 bersih) dan membaginya kedalam kategori distribusi berdasarkan hasil analisis Cuckoodroid
2.	Mobile Sandbox: Having a deeper look into android applications.	(Spreitzenbarth, Freiling, Echtler, Schreck, & Hoffmann, 2013)	• Jurnal ini membahas tentang Mobile-Sandbox, yaitu sistem yang di desain untuk melakukan analisis otomatis terhadap aplikasi android dalam dua langkah yaitu (1) mengkombinasikan antara analisis statis dan dinamis, hasil dari analisis statis digunakan untuk memandu analisis dinamis dan memperluas jangkauan kode yang dieksekusi. (2) menggunakan teknik yang spesifik untuk mengidentifikasi log API. • Sistem ini memadukan analisis statis dan dinamis, melacak log API dan dapat diakses melalui web.
3.	Forensic Analysis on Android: A Practical Case	(Alonso-Parrizas, 2015)	• Jurnal ini membahas tentang bagaimana cara melakukan analisis forensik pada perangkat android yang mencakup aspek-aspek berikut: bukti-bukti dalam log, lalu lintas jaringan, file system dan analisis memori.
4.	LiME – Linux Memory Extractor Instructions v1.1	(504ENSICS Labs, 2012)	• Jurnal ini merupakan instruksi penggunaan LiME sebagai alat untuk melakukan akuisisi terhadap memori volatil.
5.	Acquisition and Analysis of Android Memory	(Xenakis & Ntantogian, 2013)	• Jurnal ini menjelaskan tentang apa itu Live Forensics, mengapa harus mengambil RAM, fungsi dari analisis RAM serta studi kasus akuisisi RAM perangkat android menggunakan LiME.
6.	*Smartphone* forensic analysis: a case study for obtaining root access of an android Samsung S3 device and analyse the image without an expensive tool.	(Faheem, Le-Khac, & Kechadi, 2014)	• Paper ini membahas tentang eksperimen terhadap perangkat Samsung S3 dalam proses akuiasisi memori untuk kebutuhan forensik. • Paper ini menunjukkan cara untuk mendapatkan akses root, membuat DD *image* dari memori, lalu menganalisis *image* menggunakan tools UFED Physical Analyzer Examination. • Tools bekerja pada sistem operasi Windows.
7.	The Art of Memory Forensics:	(Ligh, Case, Levy, & Walters, 2016)	• Buku ini membahas tentang memori forensik pada sistem operasi Windows, Linux dan Mac OS.

	Detecting *Malware* and Threats in Windows, Linux, and Mac Memory		• Buku ini menjadi acuan tentang teknik yang digunakan dalam melakukan memori forensik dan hasil yang didapatkan melalui studi kasus.
8.	Android Forensics by LCDI	(Katz, 2013)	• Paper ini mempelajari sebanyak mungkin tentang android forensik, yang memungkinkan analisis yang lebih berhasil dari perangkat mobile ini. • Metode yang digunakan adalah (1) Research, (2) Melakukan *factory reset* menggunakan cellebrite, (3) Generate data, (4) menggunakan Cellebrite untuk mengambil log dari *physical extraction*, *logical extraction* dan *file system extraction.* (5) Ekstrak data dengan menggunakan tools XRY. • Hasil dalam paper ini yaitu mengaplikasikan penggunaan tools yaitu Cellebrite dan XRY dalam melakukan aktifitas *mobile forensics*.
9.	Live acquisition of main memory data from Android *smartphone* and smartwatches	(Yang et al., 2017)	• Dalam paper ini peneliti membuat sebuah tools bernama AMD (Android Memory Dump). • Tools ini dapat melakukan akuisisi seluruh konten dalam milik *smartphone* dan smartwatch. • Akuisisi dilakukan dengan melakukan *reverse engineering* pada *firmware update protocols* dalam *Android Bootloader.* • Paper ini membandingnkan AMD dengan tools akuisisi lain yaitu LiME, AMExtractor dan BMA. • Hasil akuisisi menggunakan AMD kemudian dianalisis menggunakan Volatility untuk memeriksa integritas data yang dihasilkan.
10.	Android Forensics: Automated Data Collection and Reporting from a Mobile Device	(Grover, 2013)	• Paper ini membahas tentang sistem monitoring yang mengambil data dari responden, auditor keamanan, monitor keamanan dan investigator forensik. • Tools yang digunakan bernama DroidWatch. • Setelah mendapat persetujuan akses dari pengguna, DroidWatch terus mengumpulkan, menyimpan dan mentransfer data Android yang dapat dijadikan bukti forensik ke web server remote tanpa hak akses root. • Data tersebut berguna untuk mengetahui: (1) *General Usage Trend,* (2) Kontak dan komunikasi mencurigakan, (3) Memonitor lokasi, (4) *History* Internet, (5) Aplikasi berbahaya.

11.	New Acquisition Method Based on Firmware Update Protocols for Android *Smartphone*s	(Yang, Choi, Kim, & Chang, 2015)	• Paper ini membahas tentang teknik akuisisi baru yaitu *physical acquisition*menggunakan flash memory dengan melakukan reverse engineering terhadap *firmware update protocol* didalam android bootloader. • Paper ini ingin membuktikan teknik baru ini unggul dalam integritas, kecepatan akuisisi dan akuisisi dengan keadaan ponsel yang terkunci. • Tools yang digunakan untuk akuisisi adalah Android Physical Dump (APD). • Hasilnya adalah tools APD ini unggul dalam integritas, kecepatan akuisisi (32GB dalam 30 menit) dan dapat melakukan akuisisi dalam keadaan ponsel terkunci.
12.	Android Platform *Malware* Analysis	(Alghamdi, Alfalqi, & Waqdan, 2015)	• Paper ini membahas tentang cara untuk mendeteksi *malware* yang ada dalam perangkat android. • Mulai dari arsitektur android, model keamanan android, bagaimana cara *malware* menginfeksi perangkat, tipe dari analisis *malware* serta tools yang digunakan untuk analisis *malware.*
13.	A Study of User Data Integrity During Acquisition of Android Devices	(Son et al., 2013)	• Paper ini mengevaluasi cara kerja "Androud Recovery Mode" terkait dengan integritas data yang dihasilkan untuk kebutuhan forensik. • Selain itu, sebuah tools dihadirkan untuk menjamin integritas data yang diperoleh saat akuisisi. • Tools tersebut bernama Android Extractor.

Data pada Tabel 2.1 adalah referensi yang peneliti gunakan untuk mengetahui penelitian *malware* yang pernah dilakukan pada lingkungan *sandbox* dan perangkat android. Penelitian menggunakan *sandbox* yaitu Cuckoodroid meneliti banyak sampel aplikasi untuk melihat dan menganalisis persebaran *malware* pada aplikasi yang beredar, sedangkan penelitian pada perangkat android menekankan pada implementasi akuisisi RAM menggunakan *tools* untuk mendapatkan salinan RAM yang dapat diakui sebagai barang bukti forensik.

Peneliti ingin mengusulkan ide untuk melakukan simulasi dan membandingkan hasil analisis sampel *malware* pada *sandbox* yang menggunakan mesin virtual dan RAM dari perangkat android asli untuk mengetahui persamaan dan perbedaan yang didapatkan dari simulasi analisis yang menggunakan dua *platform* berbeda.

Setelah mempelajari beberapa jurnal penelitian terdahulu yang terdapat pada Tabel 2.1, maka peneliti ingin mengetahui perbandingan hasil analisis *malware* yang dieksekusi melalui Cuckoodroid dan *malware* yang dieksekusi pada perangkat android asli. Seperti yang telah disebutkan sebelumnya, Cuckoodroid harus dijalankan diluar perangkat android seperti pada Cloud. Hal ini berarti *malware* akan dijalankan pada sebuah mesin virtual. Sedangkan, integritas suatu data sangat penting dalam kepentingan forensik. Hal ini berarti peneliti juga harus mendapatkan data yang nyata dari perangkat yang benar-benar sedang digunakan. Oleh karena itu peneliti akan mengeksekusi *malware* pada perangkat android yang memang sedang digunakan, lalu melakukan akuisisi RAM karena RAM merupakan tempat dari informasi yang penting dalam studi memori forensik, menggunakan tools akuisisi yang menjaga integritas kesamaan hasil akuisisi dengan konten didalam perangkat android yang digunakan.

Perbandingan hasil eksekusi *malware* pada perangkat virtual dan perangkat asli ini diharapkan membuka sudut pandang baru untuk dunia forensik khususnya bidang memori forensik dan *smartphone* forensik.

2.2 Landasan Teori

Bagian ini menjelaskan tentang teori-teori dasar yang berkaitan tentang sanboxing *malware*, tools yang digunakan untuk sandboxing *malware*, akuisisi RAM dari perangkat android, serta tools untuk menganalisis RAM yang telah berhasil diakuisisi.

2.2.1 *Malware*

Malware merupakan istilah umum untuk program atau kode yang berbahaya bagi sistem. *malware* dapat menyusup ke sistem operasi sehingga dapat merusak sistem dan juga dapat mencuri file-file penting yang ada pada sistem. Berikut ini adalah beberapa jenis *malware* yang paling sering di temui saat ini yaitu:

a. Adware

   Adware adalah program yang tidak diinginkan yang di desain untuk memunculkan iklan pada perangkat. Biasanya program ini menyamar menjadi program yang terlihat meyakinkan untuk mengelabui pengguna dan mendukung program lain untuk menipu pengguna agar memasang program pada perangkat mereka.

b. Spyware

   Spyware adalah *malware* yang diam-diam mengamati aktivitas pengguna tanpa izin dan melaporkannya kepada penyerang atau pembuat spyware tersebut.

c. Virus

Virus adalah *malware* yang melekat pada program lain dan ketika dieksekusi, virus akan mereplikasi dirinya dengan memodifikasi program komputer lain dan menginfeksi mereka dengan kode yang dibawa oleh virus tersebut.

d. Worms

Worms adalah jenis *malware* yang mirip dengan virus, mereplikasi diri untuk menyebar ke komputer lain melalui jaringan. Tujuan dari worms biasanya menyebabkan kerusakan dengan menghancurkan data dan file milik pengguna yang berada dalam sistem.

e. Trojan

Trojan atau biasa dikenal dengan Trojan Horse adalah salah satu jenis *malware* paling berbahaya. Biasanya mewakili dirinya sebagai sesuatu yang berguna untuk menipu pengguna. setelah berhasil masuk kedalam sistem, penyerang yang berada dibalik Trojan tersebut mendapatkan akses tanpa izin ke komputer yang telah terjangkit. Dari sana, Trojan dapat digunakan untuk mencuri informasi keuangan atau memasang ancaman seperti virus dan ransomware.

f. Ransomware

Ransomware adalah bentuk *malware* yang mengunci perangkat milik korban dan mengenkripsi file milik korban, kemudian memaksa korban untuk membayar uang tebusan untuk dapat membuka akses terhadap perangkat dan file-file milik program. Ransomware disebut sebagai senjata penjahat dunia maya karena menuntut pembayaran cepat dan menguntungkan dalam mata uang kripto yang sulit dilacak. Kode program ransomware ini mudah di peroleh melalui pasar kriminal online dan untuk melindungi diri dari ransomware sangat sulit.

g. Rootkit

Rootkit adalah bentuk *malware* yang memberikan hak administrator dari sistem yang terinfeksi kepada penyerang. Biasanya rootkit juga dirancang untuk tetap tersembunyi dari pengguna, program lain pada sistem dan sistem operasi itu sendiri.

h. Keylogger

Keylogger adalah *malware* yang merekam semua penekanan tombol pengguna pada keyboard, biasanya menyimpan informasi yang dikumpulkan dan mengirimkannya ke penyerang. Keylogger dibuat dengan tujuan mencari informasi sensitif seperti nama pengguna, kata sandi, atau detail kartu kredit yang diketikkan kedalam sistem.

i. Malicious Cryptomining

Malicious Cryptomining atau dikenal juga dengan nama “drive-by mining” atau “cryptojacking” adalah *malware* yang memungkinkan penyerang menggunakan komputer milik korban untuk menambang Cryptocurrency seperti Bitcoin atau Monero. Jadi, daripada penyerang membangun perangkat untuk menambang cryptocurrency yang harganya mahal, penyerang memilih menggunakan komputer korban untuk menambang dan mengirimkannya ke alamat cryptocurrency miliknya.

j. Exploits

Exploits adalah jenis *malware* yang memanfaatkan bug dan kerentanan dalam suatu sistem yang memungkinkan pencipta exploit untuk mengambil kendali.

Saat *malware* telah menginfeksi sistem, *malware* dapat bekerja sesuai dengan tujuan *malware* itu dibuat. Beberapa *malware* dibuat untuk merusak sistem saat setelah *malware* tersebut berjalan. Ada juga *malware* yang membawa muatan atau payload, yang akan melakukan eksekusi kode berbahaya saat sudah berada didalam sistem milik korban, untuk membuka akses dari sistem milik korban ke penyerang agar dapat melakukan serangan lain atau mengunduh *malware* lainnya. Visualisasi serangan *malware* adalah:

a. Langkah pertama adalah penyerang menginfeksi sistem korban dengan *malware*. Cara yang dilakukan oleh penyerang untuk menginfeksi perangkat korban beragam, dapat melalui link *phising*, *clickbait,* e-mail, dan sebagainya.
b. Langkah kedua, setelah *malware* berada dalam perangkat korban dan biasanya secara tidak sengaja korban melakukan aktivitas yang membuat *malware* tersebut berjalan (melakukan instalasi program tidak dikenal, menjalankan aplikasi mencurigakan), *malware* akan mengeksekusi kode berbahaya sesuai dengan tujuannya. Saat ini *malware* lebih banyak menanamkan backdoor agar membuka akses dari perangkat korban agar dapat mengontrol perangkat korban.
c. Langkah ketiga, setelah penyerang mendapatkan akses untuk mengontrol perangkat korban, penyerang dapat melakukan aksi tergantung dengan tujuannya. Tujuannya dapat berupa pencurian data sensitif milik korban, menghapus data korban, mengenkripsi data dan sistem korban, menyebarkan *malware* ke perangkat sekitarnya, atau juga dapat menghapus dirinya sendiri untuk menghindari deteksi.

2.2.2 Sandbox

Sandbox adalah sebuah mekanisme keamanan untuk memisahkan program yang sedang berjalan. Sandbox ini sering digunakan untuk mengeksekusi kode yang belum teruji, atau program yang tidak dipercaya dari pihak ketiga yang tidak diverifikasi, pemasok, pengguna yang tidak dipercaya dan situs web yang tidak dipercaya. Proposisi dari teknik analisis dengan menggunakan *Malware* Sandbox adalah cara yang tepat untuk menjawab tantangan yang diberikan oleh serangan *malware*. Konsep dibalik sistem analisis Sandbox *Malware* ini adalah untuk menangkap sampel program jahat dalam lingkungan pengujian terkontrol (sandbox) dimana perilaku dapat dipelajari dan dianalisis secara mendalam (Messimer, 2013). Lingkungan kerja otomatis membuat lebih mudah untuk mengikuti perkembangan *malware* dan menyiapkan basis data yang bisa digunakan untuk pembelajaran dimasa depan dan mengkategorikan mereka kedalam famili. Meskipun terdapat langkah-langkah ini, para pelaku kejahatan siber terus mengembangkan teknik yang dapat digunakan untuk mengelakkan langkah-langkah keamanan yang diberlakukan oleh analisis berbasis deteksi yang memanfaatkan sandbox. Mereka terus menyebarkan *malware* yang canggih untuk menghindari teknik deteksi menggunakan sandbox (Singh, 2015).

Teknik sandboxing ini tentunya memiliki kelebihan dan kekurangan, tetapi teknik ini adalah teknik yang baik untuk mendapatkan detail tambahan tentang *malware*, seperti perilaku jaringannya. Dan akan lebih baik jika sandboxing ini dilakukan bersamaan dengan analisis statis terhadap *malware* untuk mendapatkan pemahaman yang lebih dalam.

2.2.3 Cuckoo Sandbox

Cuckoo sandbox adalah tool untuk melakukan analisis *malware* secara otomatis. Digunakan untuk menjalankan dan menganalisa file secara otomatis dan mengumpulkan hasil analisis komprehensif yang menguraikan apa yang dilakukan *malware* saat berjalan di dalam sistem operasi. Cuckoo Sandbox memiliki banyak fitur termasuk pelacakan file yang dibuat, dihapus atau diunduh selama eksekusi, mengambil dump memori dari mesin dan menangkap lalu lintas jaringan yang terjadi selama proses eksekusi (Oktavianto, D., & Muhardianto, I., 2013). Pada penelitian ini, penulis menggunakan CuckooDroid, yaitu ekstensi pada Cuckoo Sandbox yang berfungsi khusus untuk menganalisis *malware* yang bekerja pada perangkat android. Fitur yang dimiliki oleh Cuckoo Sandbox adalah:

a. Dapat menganalisis banyak tipe file berbahaya (file eksekutabel, dokumen *office*, file pdf, e-mails, dan lain-lain) dan juga halaman web berbahaya di Windows, Linux, Mac OS X dan lingkungan virtualisasi Android.
b. Melacak panggilan API dan perilaku umum dari file dan menjadikannya informasi yang dapat dipahami oleh siapa saja.
c. Mengambil dan menganalisis lalu lintas jaringan, bahkan ketika dienksripsi dengan SSL atau TLS. Dengan dukungan "native network routing" untuk mendapatkan lalu lintas jaringan atau membuat rute melalui InetSIM, antarmuka jaringan atau VPN.

Beberapa hasil analisis yang didapatkan oleh Cuckoo Sandbox adalah:

a. Jejak panggilan API Win32 yang dihasilkan oleh semua proses yang dilakukan oleh *malware*.
b. File yang dibuat, dihapus dan diunduh oleh *malware* selama proses eksekusi.
c. Penumpukan memori dari proses *malware*.
d. Jejak lalu lintas jaringan dalam format PCAP.
e. Screenshots dari desktop yang diambil selama proses eksekusi *malware.*
f. Seluruh memori dari perangkat.

Setiap analisis dilakukan pada mesin virtual baru yang terisolasi. Infrastruktur Cuckoo terdiri dari mesin host (management software) dan sejumlah mesin guest (mesin virtual yang melakukan analisis). Mesin host menjalankan komponen inti dari sandbox yang mengelola seluruh proses analisis, sementara mesin guest adalah lingkungan terisolasi dimana sampel *malware* dieksekusi dengan aman dan kemudian dianalisis.

Setiap guest terdiri dari mesin virtual linux yang menjalankan android yang dikendalikan oleh modul mesin. Komponen tambahan yang dipasang didalam mesin linux untuk mendukung proses analisis adalah:

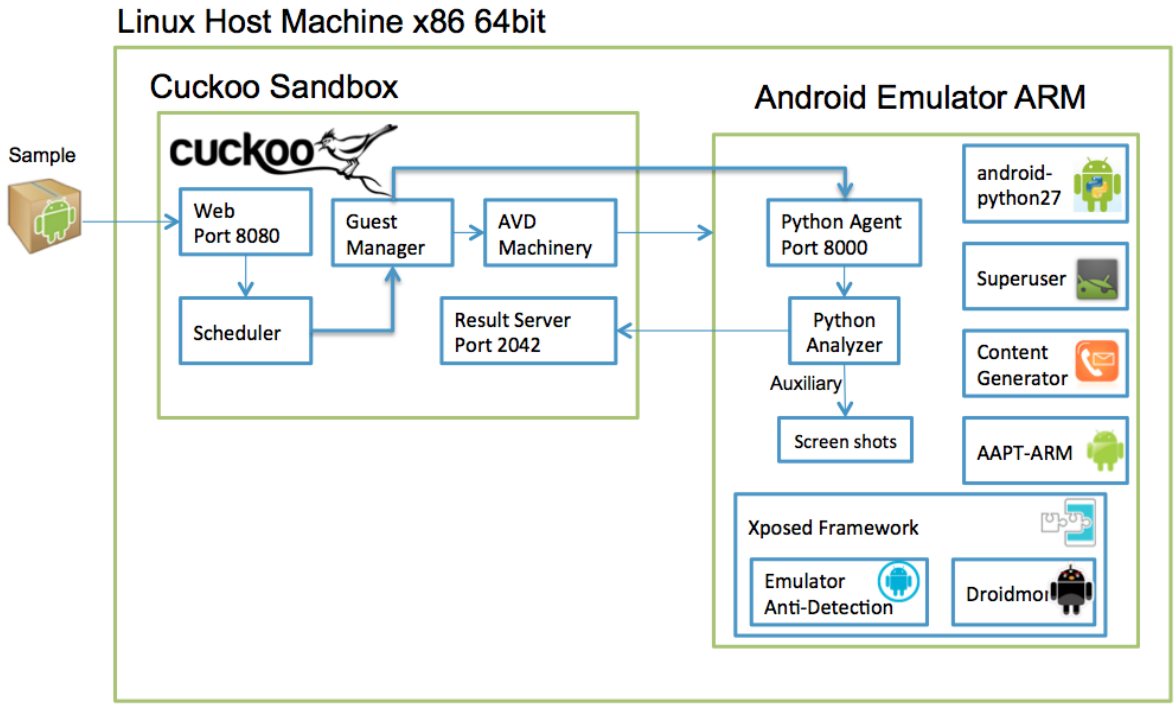
a. Python 2.7.
b. Skrip python Agent.py yang dimodifikasi untuk emulator android.
c. Komponen penganalisis android yang dikirim ke mesin guest pada awaln analisis.
d. Xposed: yaitu sebuah framework untuk modulyang dapat mengubah perilaku sistem dan aplikasi tanpa mempengaruhi APK. Ada dua modul tambahan yang dibuat melalui Xposed yaitu:
    12. Droidmon – yaitu modul untuk memantau API panggilan.
    13. Emulator anti-detection – koleksi dari teknik anti-deteksi untuk menyembunyikan emulator android.

e. Aplikasi Superuser – memberi dan mengelola hak superuser untuk ponsel.

f. Content Generator – menghasilkan daftar kontak acak untuk tampilan yang lebih realistis.

g. AAPT Arm – alat pengemas aset android yang di kompilasi untuk mengekstrak main activity dan package name dari APK.

Alur analisis dan hubungan antara mesin host Linux dengan CuckooSandbox dan emulator android dapat dilihat pada Gambar 2.1.



Gambar 2.1 Alur Analisis CuckooDroid

Command-line atau perintah yang dieksekusi pada terminal digunakan untuk melakukan semua tugas dalam Cuckoo Sandbox. Namun, kita juga dapat menggunakan bentuk web dari Cuckoo Sandbox untuk mempercepat proses analisis dari penyerahan sampel hingga meninjau laporan. Data dari laporan dapat diekspor untuk digunakan pada alat atau laporan lain dan diarsipkan menggunakan artefak forensik lainnya.

Cuckoo Sandbox yang digunakan adalah Cuckoo versi 1.2 dan Cuckoodroid versi 1.0. Versi ini digunakan karena pada versi setelahnya yaitu versi 2.0 keatas, Cuckoo Sandbox belum mendukung analisis terhadap aplikasi android dan Cuckoodroid tidak dapat di pasang kedalam Cuckoo sandbox versi terbaru.

2.2.4 LiME (Linux Memory Extractor)

LiME (Linux Memory Extractor) adalah kernel modul yang dapat diload (LKM) yang memungkinkan akuisisi memori volatil dari Linux, dan perangkat berbasis Linux seperti Android. Hal ini yang menjadikan LiME unik karena merupakat tool pertama yang memungkinkan akuisisi memori penuh pada perangkat android. LiME juga meminimalisir interaksi antara pengguna dengan proses ruang kernel selama proses akuisisi, yang menghasilkan tangkapan memori lebih forensik dibanding tool akuisisi lainnya.

Menurut Joe Sylve, dkk., (2012) ada tiga alasan mengapa LiME memenuhi standar dasar forensik. Pertama, LiME mencoba untuk meminimalisir dampak pada perangkat target saat melakukan transfer data dari dan kepada target. Kedua, LiME hanya membutuhkan koneksi melalui USB untuk interaksi perangkat. Saat perangkat telah terhubung, hanya dibutuhkan satu biner (modul kernel) yang perlu ditransfer dan dieksekusi untuk melakukan akuisisi. Ketiga, pemuatan modul membutuhkan minimal footprint karena modul LiME berukuran kecil (kurang lebih 70kb) dan membutuhkan beberapa fungsi kernel untuk memperoleh artefak memori.

Fitur dari LiME antara lain:

a. Akuisisi memori dari perangkat android secara keseluruhan.
b. Akuisisi melalui network interface.
c. Jejek proses minimal.
d. Memori yang diakuisisi memiliki Hash.

504ENSICS (2013) mengungkapkan dua tipe dari teknik akuisisi pada LiME. Pertama adalah teknik akuisisi memori melalui koneksi TCP. Pada teknik ini, pemeriksa harus mentransfer modul kernel kedalam kartu SD dari ponsel target melalui ADB. Semua proses transfer dilakukan melalui koneksi USB. Setelah itu, ponsel target harus terhubung pada port spesifik (port 4444) kemudian komputer host milik pemeriksa dihubungkan kepada ponsel target. Ketika soket terhubung, modul kernel secara otomatis mengirimkan *image* RAM yang diakuisisi kepada komputer host. Kedua adalah akuisisi memori melalui kartu SD milik ponsel target. Teknik ini dilakukan apabila pemeriksa ingin memastikan tidak ada buffer yang ditimpa. Untuk melaksanakan prosesnya, penguji harus melakukan *imaging* terhadap kartu SD terlebih dahulu untuk menghemat ruang yang tidak terisi. Setelah itu, penguji harus menghubungkan perangkat ke mesin Linux dan mengaktifkan penyimpanan USB yang mengekspos direktori /dev/sd?. Perangkat dapat dicitrakan menggunakan cara tradisional (contohnya: menggunakan dd pada mesin Linux). Jika kartu SD dapat dilepas, maka penguji dapat melepas kartu SD dari

perangkat untuk diakuisisi, tetapi apabila SD tidak dapat dilepas, maka penguji harus menggunakan ADB untuk mentransfer dump memori ke mesin investigasi penguji.

2.2.5 Volatility

Volatility framework adalah kumpulan tools yang diimplementasikan dengan python dibawah lisensi publik GNU, untuk ekstraksi artefak digital dari sampel volatil memori (RAM). Teknik ekstraksi dilakukan sepenuhnya independen dari sistem yang sedang tetapi menawarkan visibilitas ke status runtime sistem. Framework ini dimaksudkan untuk memperkenalkan orang kepada teknik dan kompleksitas yang terkait dengan ekstraksi artefak digital dari sampel memori yang mudah menguap dan menyediakan platform untuk bidang penelitian ini dimasa depan.

Volatility yang digunakan adalah versi 2.6. Keunggulan dari Volatility adalah Volatility sudah mendukung analisis terhadap volatil memori dari android dengan menyediakan plugin khusus untuk android. Volatility tidak memiliki kemampuan untuk melakukan akuisisi volatil memori. Oleh karena itu, dibutuhkan tool lain untuk melakukan akuisisi, baik secara gratis maupun komersil. Salah satu tool untuk akuisisi ini adalah LiME (Linux Memory Extractor). Volatility telah mendukung ekstensi .lime sebagai salah satu format volatil memori yang dapat dianalisis. Keunggulan lainnya adalah dukungan komunitas dan dokumentasi yang cukup lengkap.

Volatility mendukung berbagai format file sampel dan kemampuan untuk mengkonversi antara format ini, antara lain:

a. Raw linear sample (dd)
b. Hibernation file (dari Windows 7 dan sebelumnya)
c. Crash dump file
d. VirtualBox ELF64 core dump
e. Vmware saved state dan file snapshot
f. EWF format (E01)
g. LiME format
h. Mach-O format file
i. QEMU virtual machine dumps
j. Fireware
k. HPAK (FDPro)

BAB III
METODE PENELITIAN

3.1 Alur Penelitian

3.1.1 Alur Penelitian Cuckoodroid

Pada penelitian menggunakan tools Cuckoodroid ini, langkah pertama yang harus dilakukan adalah membangun lingkungan penelitian. Lingkungan penelitian ini menggunakan mesin Linux yaitu Ubuntu Desktop 16.04. Agar simulasi dan analisis dapat dilakukan, mesin Linux harus diatur agar dapat membangun file sumber android dengan cara menginstall JDK pada mesin Linux tempat simulasi dan analisis akan dijalankan. JDK atau *Java Development Kit* adalah lingkungan pengembangan perangkat lunak yang digunakan untuk mengembangkan aplikasi dan applet Java.

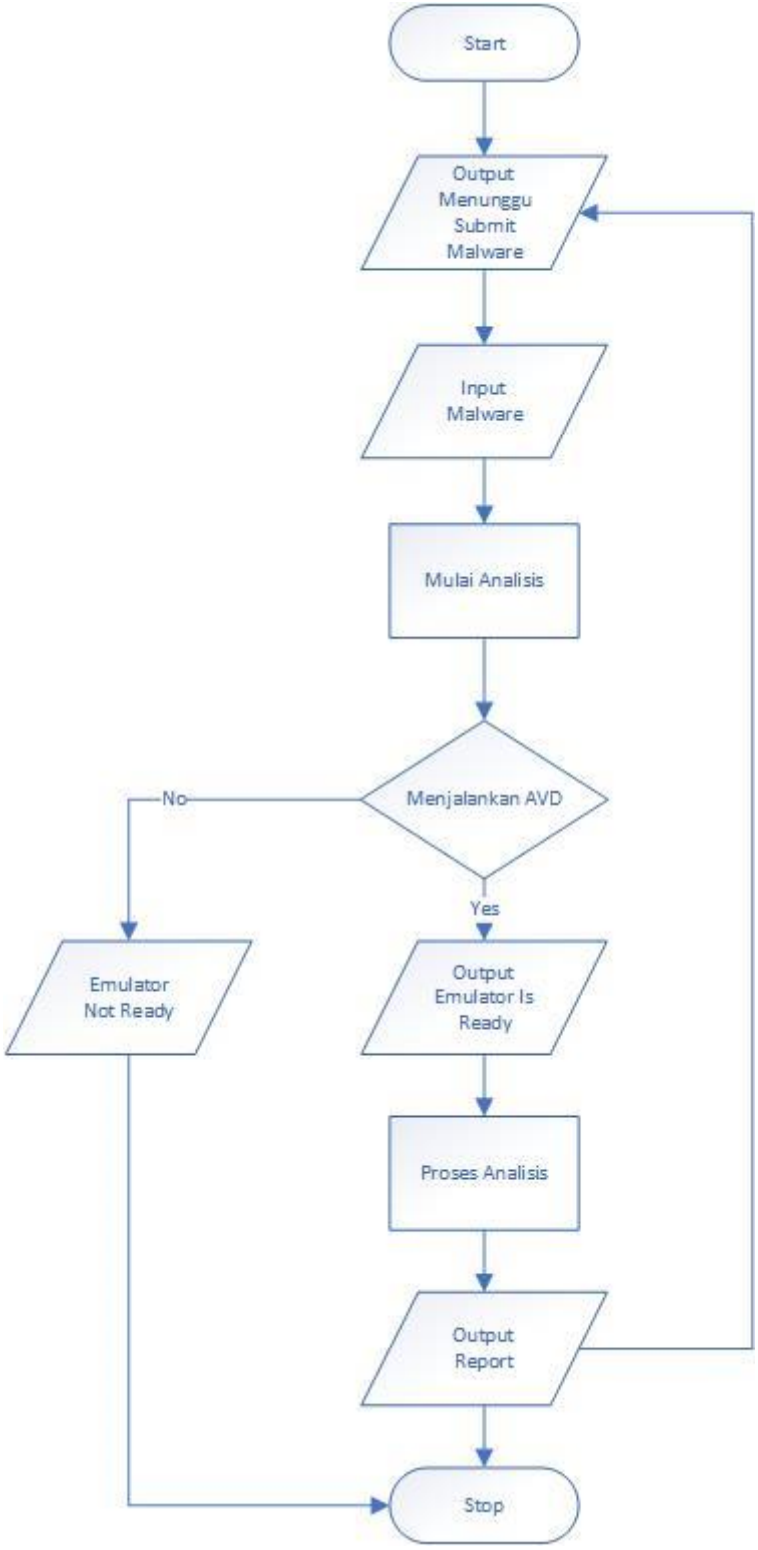
Langkah selanjutnya adalah melakukan instalasi Cuckoo sandbox dan Cuckoodroid dari repositori github milik Cuckoo. Seperti yang kita tahu bahwa Cuckoodroid merupakan ekstensi dari Cuckoo sandbox yang khusus untuk menganalisis *malware* milik android. Oleh karena itu, instalasi Cuckoodroid dilakukan dengan melakukan instalasi Cuckoo sandbox terlebih dahulu.

Setelah Cuckoo dan Cuckoodroid telah terinstall, langkah selanjutnya adalah melakukan konfigurasi terhadap beberapa file di dalam direktori Cuckoo sandbox. Konfigurasi yang dilakukan terdiri dari nama mesin, emulator yang digunakan, database yang digunakan, tools analisis lain yang digunakan seperti analisis untuk jaringan, dan lain-lain.

Hal yang perlu dipersiapkan selanjutnya adalah android SDK. Pada penelitian ini, android SDK berfungsi untuk membangun emulator android yang akan digunakan oleh Cuckoodroid sebagai wadah instalasi dan analisis *malware* android. Hal yang harus diperhatikan adalah penempatan *path* atau lokasi pada konfigurasi Cuckoo harus sesuai. Jika tidak maka akan mengalami error.

Cuckoodroid memerlukan beberapa tools tambahan untuk menunjang analisis yang akan dilakukan, beberapa tools berupa modul python yang dapat diinstall menggunakan pip, namun ada juga yang dapat diinstall menggunakan perintah apt-get install.

Alur kerja Cuckoo saat melakukan analisis sampel *malware* android dapat dilihat pada Gambar 3. 1.

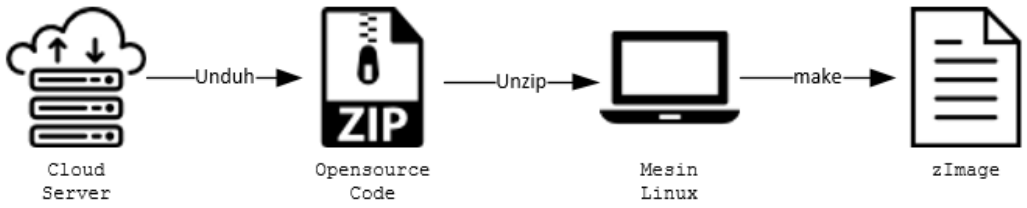


Gambar 3. 1 Alur Analisis Cuckoodroid

Setelah Cuckoo berhasil dijalankan, kita dapat melakukan *submit malware* melalui *web browser* dengan mengakses halaman web milik Cuckoodroid. Setelah *submit*, Cuckoo akan mulai melakukan analisis, setelah analisis selesai, hasil analisis dapat dilihat pada bagian *result*.

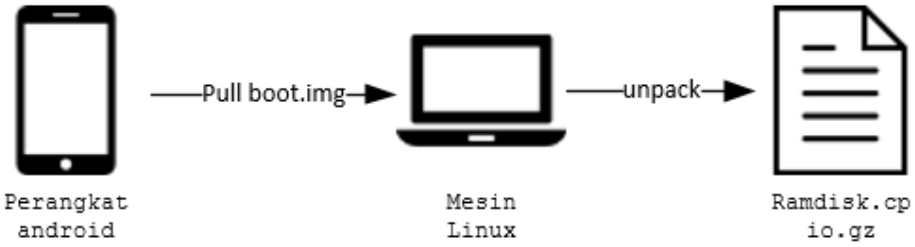
3.1.2 Alur Penelitian Volatility

Untuk melakukan analisis terhadap RAM *smartphone* android yang terinfeksi *malware*, penelitian diawali dengan mempersiapkan opensource code dari tipe *smartphone* yang akan diakuisisi. *Opensource code* memuat kernel dari *smartphone* yang diperuntukkan untuk pengembangan lebih lanjut dari developer untuk melakukan kustomisasi terhadap kernel. *Opensource code* ini dapat diunduh pada server milik vendor *smartphone*. Pada penelitian ini, *Opensource code* diunduh dari server milik Sony Xperia. Setelah mengunduh *Opensource code*, penguji membuat kernel dari *Opensource Code* tersebut. Alur pembuatan kernel dapat dilihat pada Gambar 3.9.



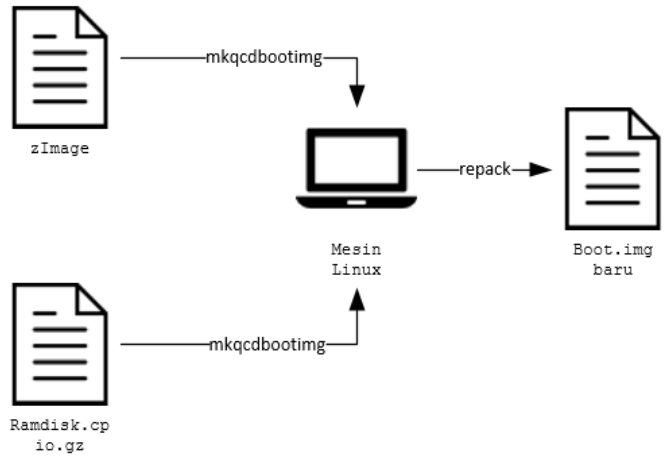
Gambar 3.2 Alur pembuatan kernel

Kernel yang dihasilkan oleh *Opensource code* ini bernama "zImage". Kernel ini kemudian akan di *compile* kedalam *bootable image* yang diambil dari *smartphone*, kemudian *bootable image* tersebut akan diinstal kedalam *smartphone*. Tujuan dari kompilasi zImage dengan *bootable image* ini adalah sebagai jembatan agar *smartphone* dapat menginstal kernel modul Lime. Untuk membuat bootable image yang baru, penguji harus mengambil bootable image terdahulu dari perangkat untuk mengambil file ramdisk.cpio.gz. proses pengambilan ramdisk dapat dilihat pada Gambar 3.3.



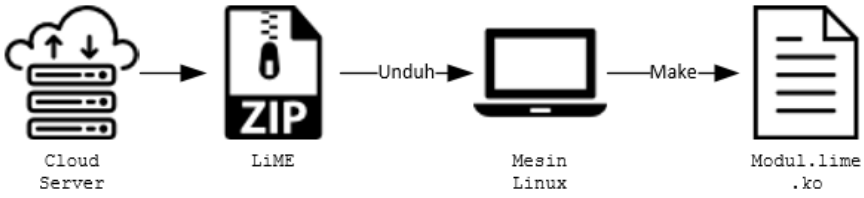
Gambar 3.3 Proses mengambil ramdisk

Kemudian, file ramdisk dan file zImage dimuat ulang kedalam satu bootable image yang baru dengan menggunakan tools mkqcdtbootimg. Proses memuat ulang bootable image ini dapat dilihat pada Gambar 3.4.



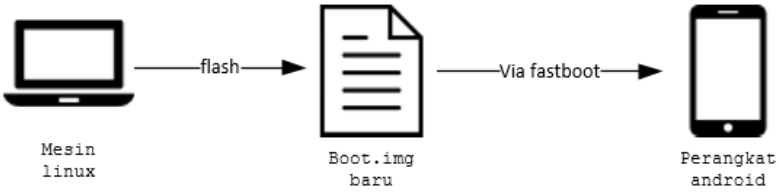
Gambar 3.4 Proses pembuatan boot.img baru

Setelah *bootable image* baru yang berisi kernel hasil kompilasi dari opensource code telah terinstall didalam *smartphone*, langkah selanjutnya adalah membuat kernel modul oleh tools bernama Lime. Kernel modul milik Lime ini berfungsi agar komputer dapat berkomunikasi dengan *smartphone* melalui jaringan TCP untuk melakukan akuisisi RAM. Proses pembuatan kernel modul LiME dapat dilihat pada Gambar 3.5



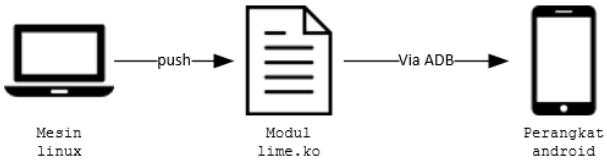
Gambar 3.5 Proses pembuatan modul LiME

Bootable image baru yang telah dibuat akan diinstal pada perangkat agar dapat membaca modul milik LiME. Proses instalasi bootable image baru dilakukan fastboot. Proses instalasi bootable image baru dapat dilihat pada Gambar 3.6.



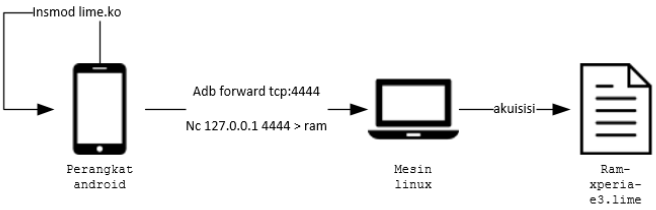
Gambar 3.6 Proses instalasi bootable image baru

Kemudian, kernel modul LiME bernama *lime.ko* ini akan dipindahkan kedalam *smartphone* melalui ADB seperti Gambar 3.7.



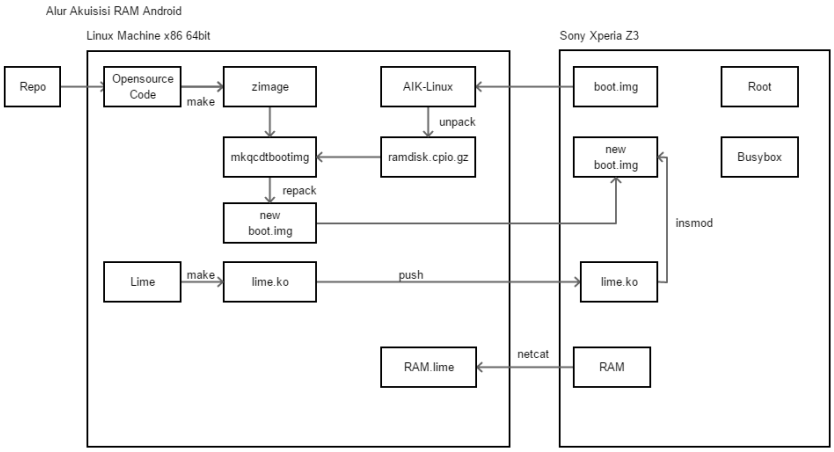
Gambar 3.7 Proses pemindahan modul lime.ko

Selanjutnya, kernel modul diinstal kedalam *smartphone* dengan perintah insmod (*insert module)*. Setelah kernel modul *lime.ko* telah terinstal, akuisisi RAM dapat dilakukan. Proses instalasi modul dan akuisisi RAM dapat dilihat pada Gambar 3.8.



Gambar 3.8 Instalasi modul dan Akuisisi RAM

Akuisisi RAM dilakukan menggunakan tools netcat, melalui jaringan TCP. Akuisisi dinyatakan berhasil dilakukan apabila kapasitas atau besar RAM yang diakuisisi sesuai dengan besar RAM yang tersedia. Alur teknik melakukan akuisisi RAM pada perangkat android ini digambarkan pada Gambar 3.9



Gambar 3.9 Alur Akuisisi RAM

Langkah selanjutnya adalah mempersiapkan tools untuk analisis RAM yang telah diakuisisi. Tools yang digunakan pada penelitian ini adalah tools bernama Volatility. Volatility adalah tools yang digunakan untuk menganalisis hasil ekstraksi artefak digital dari volatil memori (RAM). Volatility memiliki banyak plugin, beberapa diantaranya berfungsi untuk menganalisis apakah artefak digital terinfeksi *malware* atau tidak. Penelitian ini bertujuan untuk menganalisis apakah RAM yang diakuisisi terinfeksi *malware* atau tidak, dan apabila RAM tersebut terinfeksi *malware*, bagaimana aktifitas *malware* didalam RAM tersebut.

3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan dalam penyelesaian penelitian analisis *malware* ini di dasarkan pada kebutuhan sistem untuk menjalankan tools akuisisi dan analisis *malware* tersebut. Hasil dari analisis kebutuhan ini adalah:

3.2.1 Analisis Kebutuhan Perangkat Keras

Pada penelitian ini, penguji membutuhkan sebuah mesin Linux yang akan menjadi host dari Cuckoo Sandbox untuk simulasi eksekusi *malware* pada emulator android, serta menjadi host untuk analisis RAM dari *smartphone* target yang telah diakuisisi. Perangkat keras yang dibutuhkan terdiri dari Laptop yang akan menjadi mesin host, *smartphone* yang akan menjadi target studi kasus eksekusi *malware*, serta kabel USB untuk mentransfer data dari *smartphone* target ke mesin host. Spesifikasi dari kebutuhan perangkat keras yang digunakan adalah:

a. Laptop

Laptop yang digunakan pada penelitian ini adalah laptop ASUS dengan seri A43S yang memiliki spesifikasi prosesor intel core i3-2330M, 2.2GHz dengan RAM 8GB. Laptop ini mengadaptasi *dual-boot* dengan sistem operasi Windows 8.1 dan Linux Ubuntu 16.04 LTS.

b. *Smartphone* Android

Smartphone yang digunakan untuk penelitian ini adalah Sony E3 D2202 dengan Build number 18.5.A.0.22 dengan ROM android 4.4 Kitkat. Untuk penelitian ini, *smartphone* yang digunakan dalam keadaan *Unlocked Bootloader*, *Root* dan telah terinstall aplikasi *Superuser* dan *busybox* didalamnya.

3.2.2 Analisis Kebutuhan Perangkat Lunak

a. Python 2.7

Python adalah Bahasa pemrograman yang dapat melakukan eksekusi sejumlah instruksi multiguna secara langsung (interpretative) dengan metode orientasi objek (Object Oriented Programming) serta menggunakan semantic dinamis untuk memberikan tingkat keterbacaan syntax (Advernesia, 2018). Python yang digunakan adalah python dengan versi 2.7 dan diinstal pada system operasi Ubuntu.

b. Java JDK

Java JDK (Java Development Kit) adalah sebuah software yang digunakan untuk membuat aplikasi dengan Bahasa pemrograman Java. Fungsi dari java JDK dalam penelitian ini adalah sebagai salah satu kebutuhan software untuk membangun lingkungan kerja local untuk membangun android source code.

c. Opensource Kernel

Opensource kernel adalah kernel dari perangkat yang sengaja dibuka aksesnya kepada para pengembang untuk membangun dan memungkinkan akses kepada fitur-fitur tertentu dan melakukan modifikasi pada perangkat. Pada penelitian ini, opensource kernel dibutuhkan untuk proses akuisisi RAM pada perangkat android.

d. Cross Compile Software

Cross compiler adalah compiler yang dapat membuat kode yang dapat dieksekusi untuk platform selain dari platform host. Cross compiler diperlukan untuk mengkompilasi suatu kode untuk berbagai platform dari satu host. Dalam penelitian ini, cross compiler diperlukan untuk mengkompilasi opensource code dari *smartphone* untuk menghasilkan kernel baru yang akan diinstal kedalam perangkat agar akuisisi RAM dapat berkerja dengan baik. Cross compiler yang digunakan pada penelitian ini adalah arm-eabi-4.6.

e. Android Kitchen

Android Kitchen adalah sebuah tool untuk melakukan packing ulang paket. Dalam penelitian ini, Android Kitchen digunakan untuk memasukkan kernel yang telah dikompilasi ke dalam boot.img, untuk selanjutnya di *flash* kedalam perangkat agar proses akuisisi berjalan dengan baik.

f. Android SDK Manager

Android SDK Manager adalah sebuah *command line tool* yang memungkinkan pengguna untuk melakukan *view, install* dan *uninstall package* untuk Android SDK. Dalam penelitian ini, Android SDK Manager berfungsi untuk membuat dan melakukan konfigurasi android emulator atau mesin AVD sebagai wadah eksekusi sampel *malware* yang akan dilakukan oleh CuckooDroid.

g. ADB

Android Debugging Bridge atau lebih dikenal dengan nama ADB adalah sebuah tools untuk berkomunikasi antara *client* dan *server*. Dengan kata lain, ADB diperlukan agar komputer host dapat berkomunukasi dengan *smartphone*. Dalam penelitian ini, ADB berfungsi untuk mengambil boot image dari dalam *smartphone*, serta untuk memasukkan modul lime kedalam *smartphone*.

h. Fastboot

Fastboot adalah alat diagnostic yang digunakan untuk memodifikasi sistem file android dari komputer ketika *smartphone* dalam mode bootloader. Dalam penelitian ini, mode fastboot digunakan untuk melakukan instalasi boot image yang telah dimodifikasi kedalam *smartphone*.

i. Netcat

Netcat adalah sebuah program berbasis python yang berfungsi untuk membaca atau menulis dari soket TCP dan UDP menggunakan antarmuka yang mudah. Dalam penelitian ini, netcat digunakan untuk melakukan akuisisi RAM dari *smartphone* android kedalam mesin linux melalui port TCP.

3.2.3 Analisis Kebutuhan Proses

Pada saat melakukan penelitian analisis *malware* menggunakan Cuckoodroid, terdapat beberapa proses yang terjadi. Proses berawal dari mengunduh sampel *malware* dari repositori Github. Kemudian sampel *malware* di *submit* kepada Cuckoodroid. Setelah proses *submit*, selanjutnya Cuckoodroid akan menjalankan android emulator untuk selanjutnya melakukan proses analisis. Setelah analisis selesai, Cuckoodroid akan memproses hasil analisis yang didapatkan kedalam *report* melalui halaman website.

Kemudian, proses untuk melakukan analisis *malware* pada RAM android berbeda dengan proses melakukan analisis *malware* menggunakan Cuckoodroid. Analisis *malware* pada RAM android dimulai dengan mengunduh opensource code dari repositori milik Sony. Opensource ini berisi kernel dari *smartphone* yang akan dibuild dan digunakan agar RAM dapat diakuisisi. Proses selanjutnya adalah mengekstrak opensource code ini, kemudian menjalankan perintah "make" untuk membuat kernel. Setelah itu, kernel dibongkar menggunakan tools AIK Linux, untuk kemudian dimasukkan kedalam boot image milik *smartphone*. Proses selanjutnya yaitu menginstall boot image yang telah berisi kernel baru kedalam *smartphone* yang akan diakuisisi ramnya. setelah boot image terinstall didalam *smartphone*, langkah selanjutnya adalah membuat modul lime. Modul lime digunakan untuk

membuka akses *smartphone* agar ram dapat diakuisisi. Setelah modul lime terbentuk, modul tersebut diinstall kedalam *smartphone*. Setelah itu, proses akuisisi RAM dapat dijalankan dengan menggunakan netcat.

Proses selanjutnya setelah akuisisi RAM adalah analisis RAM menggunakan tools Volatility. Proses analisis dimulai dengan membuat profil RAM, kemudian analisis dilakukan dengan menjalankan beberapa plugin yang disediakan oleh Volatility untuk mencari dan mengidentifikasi aktifitas *malware* dalam RAM android.

3.3 Eksaminasi

Eksaminasi atau pengujian yang akan dilakukan terhadap RAM *smartphone* yang telah diakuisisi dilakukan dengan menggunakan Volatility. Volatility framework adalah kumpulan tools yang diimplementasikan dengan python dibawah lisensi publik GNU, untuk ekstraksi artefak digital dari sampel volatil memori (RAM). Dalam penelitian ini, Volatility digunakan untuk mencari indikasi adanya *malware* yang berjalan pada RAM android.

Volatility menyediakan berbagai plugin untuk kebutuhan yang berbeda-beda. dalam penelitian ini, terdapat beberapa plugin yang akan dijalankan sesuai dengan fungsi dan keluaran yang diinginkan, antara lain sebagai berikut.

a. Linux_pslist

Plugin linux_pslist ini berfungsi untuk menampilkan semua proses yang berjalan dengan mengikuti daftar EPROCESS. Dengan menjalankan plugin linux_pslist, kita akan mencari apakah terdapat proses milik *malware* didalam RAM.

b. Linux_psxview

Plugin linux_psxview ini berfungsi untuk mencari proses tersembunyi didalam berbagai daftar proses.

c. Linux Malfind

Plugin linux_malfind ini berguna untuk mencari kode tersembunyi yang telah diinjeksi.

d. Linux_proc_maps

Plugin linux_proc_maps ini berfungsi untuk mencetak detail memori proses, termasuk *stacks* dan pustaka bersama.

e. Linux_arp

Plugin linux_arp berfungsi untuk mencetak tabel ARP pada RAM.

f. Linux_ifconfig

Plugin linux_ifconfig berfungsi untuk mengetahui ip address dari perangkat saat RAM diakuisisi.

g. Linux_route_cache

Plugin linux_route_cache ini berfungsi untuk mencetak data dalam cache tabel routing yang dapat menunjukkan kepada kita sistem mana yang berkomunikasi dengan perangkat dimasa lalu.

3.4 *Malware* Analysis

Sampel *malware* yang digunakan dalam penelitian ini adalah BeNews. BeNews merupakan aplikasi *backdoor* yang menyamar sebagai portal berita. BeNews telah diunduh sebanyak lima puluh kali sebelum akhirnya dihapus dari *Google Play* pada 7 Juli 2015. BeNews dapat menginfeksi versi android 2.2 Froyo hingga 4.4 KitKat. Peneliti menggunakan aplikasi BeNews sebagai sampel karena saat peneliti mencoba untuk melakukan eksekusi terhadap beberapa sampel, BeNews ini yang memperlihatkan aktivitas yang mencurigakan pada RAM. Hal ini tidak ditemukan pada beberapa sampel yang di uji pada saat yang sama, oleh karena itu peneliti memilih BeNews sebagai sampel *malware* yang akan dilihat hasilnya.

Tujuan dari penelitian ini adalah untuk mengetahui perbandingan hasil analisis sampel *malware* yang dieksekusi pada sandbox dan RAM android. Maka, langkah awal dari penelitian ini adalah melakukan eksekusi sampel *malware* kedalam CuckooDroid. Sampel *malware* yang sama juga kita eksekusi pada sisi *smartphone*. Awal eksekusi *malware* pada Cuckoodroid adalah dengan melakukan submit pada direktori Cuckoo, kemudian Cuckoodroid akan mengeksekusi dan melakukan analisis secara otomatis. Laporan dari hasil analisis Cuckoodroid dapat dilihat dalam sebuah halaman website lokal.

Untuk analisis pada sisi RAM, diawali dengan melakukan akuisisi RAM dari perangkat yang sudah diinjeksi *malware*. Setelah RAM berhasil diakuisisi, RAM tersebut dianalisis menggunakan Volatility. Cara melakukan analisis dengan Volatility adalah dengan menentukan plugin apa yang akan digunakan sesuai dengan informasi apa yang ingin dicari dan ditampilkan. Sesuai dengan keterangan pada subbab Eksaminasi, plugin yang akan digunakan adalah linux_pslist, linux_psxview, linux_malfind serta linux_proc_maps, linux_arp, linux_ifconfig serta linux_route_cache.

Hasil akhir dari rencana analisis ini adalah diharapkan dapat menemukan dan membuktikan adanya kerja *malware* pada sisi RAM android, serta menemukan hubungan antar kedua tools yang digunakan yaitu Cuckoodroid dan Volatility dalam menganalisis *malware*.

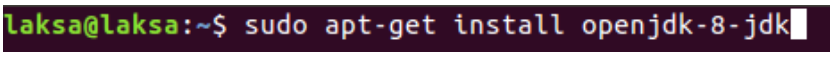
BAB IV
IMPLEMENTASI DAN HASIL PENGUJIAN

4.1 Implementasi Perangkat Lunak

4.1.1 Membangun Lingkungan Kerja Simulasi dan Analisis

Langkah pertama yang harus dilakukan pada penelitian ini adalah membangun lingkungan penelitian. Lingkungan penelitian ini menggunakan mesin Linux yaitu Ubuntu Desktop 16.04. Agar simulasi dan analisis dapat dilakukan, mesin Linux harus diatur agar dapat membangun file sumber android dengan cara menginstall JDK pada mesin Linux tempat simulasi dan analisis akan dijalankan. JDK atau *Java Development Kit* adalah lingkungan pengembangan perangkat lunak yang digunakan untuk mengembangkan aplikasi dan applet Java.

Installasi Java JDK pada Ubuntu dapat dilakukan melalui terminal dengan perintah seperti pada Gambar 4.1.



Gambar 4.1 Instalasi Java JDK

Setelah JDK berhasil diinstall, langkah selanjutnya adalah melakukan instalasi Cuckoo dan Cuckoodroid pada mesin Linux.

4.1.2 Instalasi Cuckoo dan Cuckoodroid

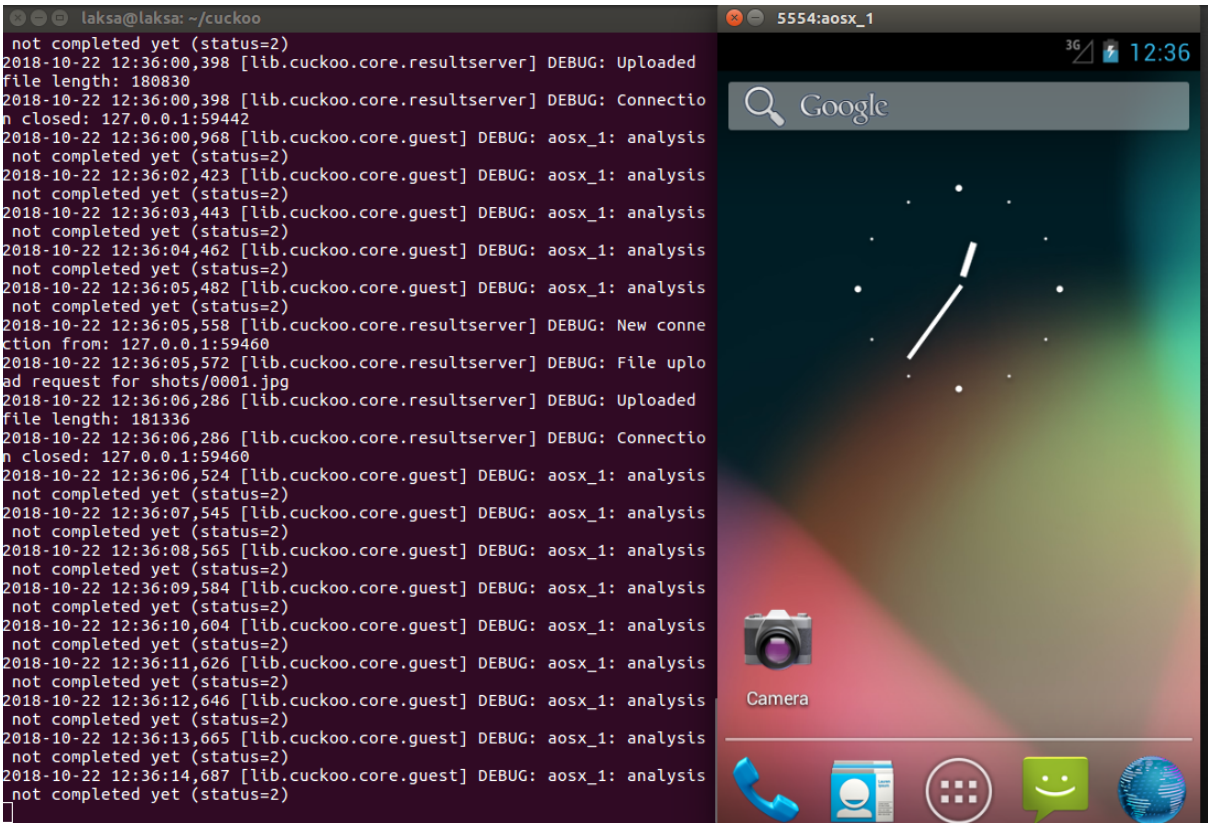
Tools yang digunakan oleh penguji pada penelitian *malware* ini adalah Cuckoodroid yaitu sebuah ekstensi yang berdiri diatas Cuckoo Sandbox untuk melakukan analisis terhadap aplikasi android dalam ekstensi APK (Android Package Kit).

Instalasi tools diawali dengan menginstall Cuckoo Sandbox pada mesin Linux. Cuckoo Sandbox yang digunakan pada penelitian ini adalah Cuckoo Sandbox versi 1.2. Proses instalasi Cuckoo dan Cuckoodroid dapat dilihat pada Lampiran A. Tampilan Cuckoo Sandbox dapat dilihat pada gambar Gambar 4.2



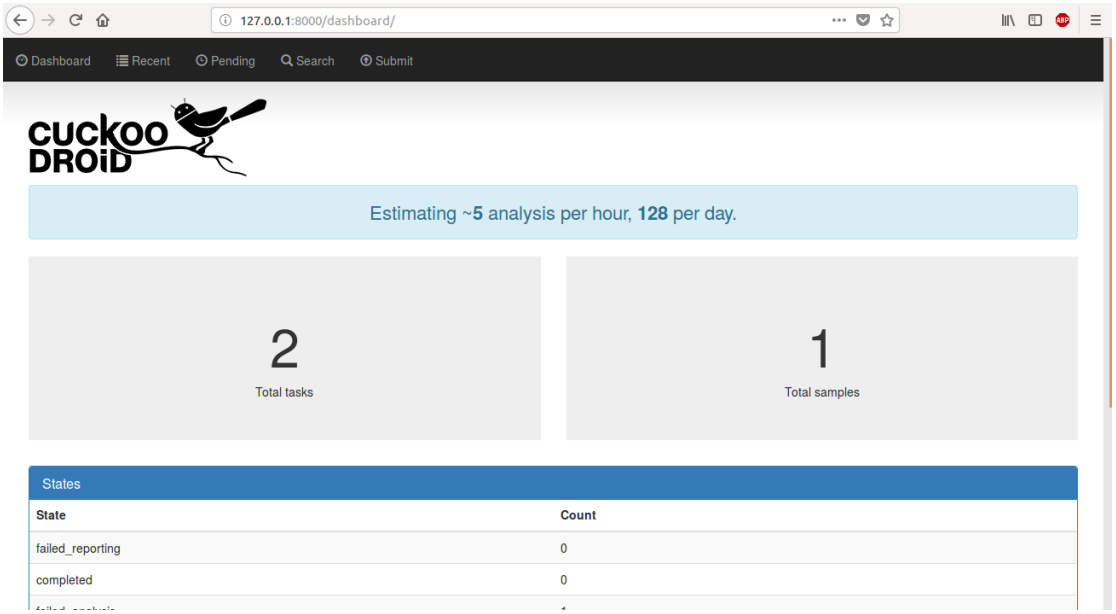
Gambar 4.2 Cuckoo Sandbox

Agar Cuckoo Sandbox dapat melakukan analisis terhadap aplikasi android, Cuckoo Sandbox harus di pasang dengan framework Cuckoodroid. Setelah framework Cuckoodroid terpasang pada Cuckoo Sandbox, perlu dilakukan beberapa konfigurasi agar Cuckoo dapat terintegrasi dengan Android Virtual Device (AVD) sebagai wadah analisis. Tampilan Cuckoo yang terintegrasi dengan AVD dapat dilihat pada gambar Gambar 4.3.



Gambar 4.3 Cuckoo dan AVD

Analisis dilakukan dengan mengajukan sampel *malware* kedalam Cuckoo Sandbox. Setelah *malware* diajukan kedalam Cuckoo Sandbox, mesin akan memulai analisis dengan membuka AVD. Selanjutnya tinggal menunggu analisis selesai dilakukan. Hasil analisis dari Cuckoo Sandbox ini dapat dilihat dalam bentuk halaman website seperti pada Gambar 4.4.



Gambar 4.4 Halaman website hasil analisis Cuckoo Sandbox

4.1.3 Akuisisi RAM

Setelah analisis dengan Cuckoo dan Cuckoodroid berhasil dilakukan, kita menuju ke tahap selanjutnya yaitu melakukan analisis *malware* dengan studi kasus RAM dari perangkat android. Pada penelitian ini, sampel *malware* yang dianalisis menggunakan Cuckoodroid diinstal juga kedalam perangkat android.

Langkah awal untuk melakukan akusisi RAM perangkat android adalah dengan mempersiapkan beberapa kebutuhan yaitu:

a. Opensource Kernel

Opensource Kernel adalah kernel yang disediakan oleh vendor secara *opensource* yang dapat digunakan untuk membuat atau memodifikasi kernel sesuai dengan seri perangkat yang dimiliki. Pada penelitian ini, opensource kernel yang digunakan adalah seri 23.0.2.A untuk perangkat Sony Xperia E3.

b. Cross-Compiler

Cross-Compiler adalah sebuah aplikasi untuk melakukan compile kernel agar kernel dapat digunakan oleh perangkat android. Cross-compiler yang digunakan dalam penelitian ini adalah arm-eabi-4.6.

c. Unpack/Repack Tools

Unpack dan Repack Tools adalah aplikasi yang digunakan untuk mengambil sebuah file bernama “ramdisk.cpio.gz” yang dibutuhkan untuk menggabungkan modul lime dan boot.img milik perangkat agar modul lime dapat dijalankan didalam perangkat. Tools yang digunakan untuk melakukan unpack adalah tools bernama “build_tools” dan tools untuk melakukan repack adalah “mkqcdtbootimg”.

d. Fastboot

Fastboot adalah sebuah tools yang menghubungkan perangkat dengan komputer client dalam mode fastboot. Tools ini digunakan untuk melakukan flashing boot.img kedalam perangkat.

Setelah seluruh kebutuhan telah siap, akuisisi RAM dapat dilakukan. Langkah-langkah akuisisi RAM dapat dilihat pada lampiran B. setelah akuisisi RAM berhasil dilakukan, RAM android telah siap untuk dianalisis.

4.1.4 Implementasi Analisis Volatility

Setelah RAM berhasil diakuisisi, langkah selanjutnya adalah melakukan eksekusi RAM dengan Volatility. Untuk menjalankan Volatility, pertama-tama kita harus membuat profil Volatility sesuai dengan jenis perangkat yang digunakan. Untuk membuat profil Volatility milik Xperia E3, kita dapat menjalankan perintah make seperti pada Gambar 4.5.

```
make[1]: Entering directory '/home/laksa/Xperia_E3/18.5.A.0.22/kernel'
  CC [M]  /home/laksa/android-volatility/tools/linux/module.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/laksa/android-volatility/tools/linux/module.mod.o
  LD [M]  /home/laksa/android-volatility/tools/linux/module.ko
make[1]: Leaving directory '/home/laksa/Xperia_E3/18.5.A.0.22/kernel'
dwarfdump -di module.ko > module.dwarf
```

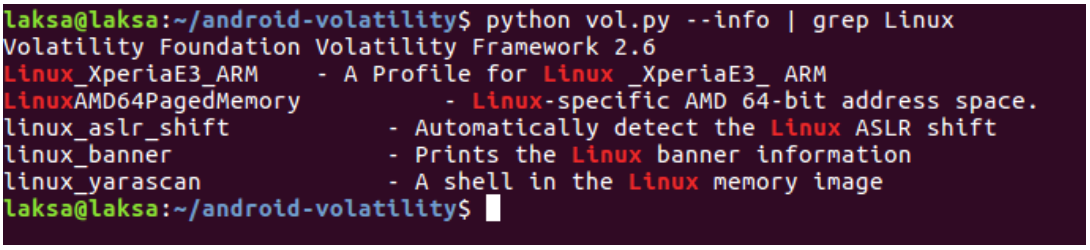
Gambar 4.5 make profile Volatility

Kemudian kita membuat zip yang menggabungkan module.dwarf dengan System.map milik kernel yang kita buat sebelumnya seperti pada Gambar 4.6.

```
laksa@laksa:~/android-volatility/tools/linux$ zip /home/laksa/android-volatility/volatility/plugins/overlays/linux/_E3_.zip module.dwarf /home/
laksa/Xperia_E3/18.5.A.0.22/kernel/System.map
  adding: module.dwarf (deflated 90%)
  adding: home/laksa/Xperia_E3/18.5.A.0.22/kernel/System.map (deflated 75%)
laksa@laksa:~/android-volatility/tools/linux$
```

Gambar 4.6 Membuat zip profil

Sampai pada tahap ini, profil Volatility kita telah dapat kita pakai. Untuk melihat profil kita, kita dapat menjalankan “python vol.py –info | grep Linux”. Beberapa profil yang telah terinstal dapat dilihat pada Gambar 4.7.



Gambar 4.7 Melihat profil Volatility

Untuk melakukan penelitian, Volatility menyediakan berbagai plugin sesuai dengan kebutuhan user. Pada penelitian ini, plugin yang digunakan adalah:

a. Linux_pslist
b. Linux_psxview
c. Linux_malfind
d. Linux_proc_maps
e. Linux_arp
f. Linux_ifconfig
g. Linux_route_cache

Untuk menjalankan Volatility, perintah yang digunakan adalah “python vol.py –profile=[nama_profil] -f [letak ram] [plugin].

4.2 Hasil dan Pembahasan

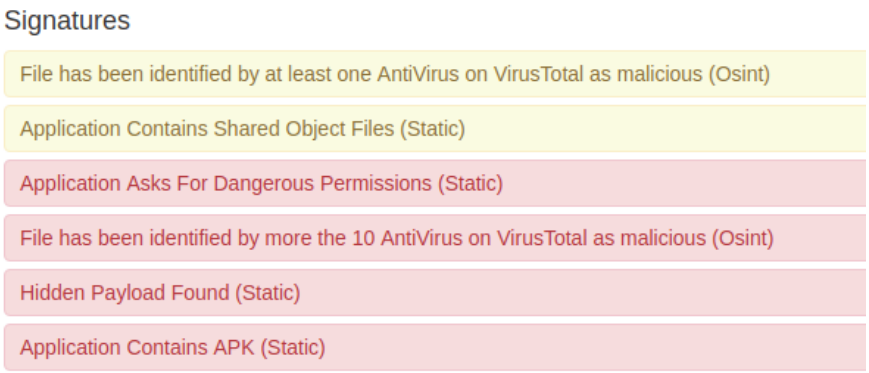
Setelah melakukan implementasi dengan menjalankan analisis pada Cuckoodroid dan Volatility, kita akan masuk kedalam tahap analisis hasil yang telah didapatkan dari implementasi sebelumnya.

4.2.1 Hasil Analisis Cuckoodroid

Implementasi menggunakan Cuckoodroid menganalisis file apk dari aplikasi BeNews. Dari hasil analisis secara otomatis, kita mendapatkan beberapa informasi. Hasil yang dilaporkan oleh Cuckoodroid terbagi kedalam tiga bagian yaitu Quick Overview, Static Analysis dan Network Analysis.

Quick Overview berisi garis besar dari aplikasi yang dianalisis, berisi beberapa sub bagian yaitu Android Application, Analysis, Machine, File Details Signatures, Screenshots dan Hosts.

Dari menu Quick Overview ini, kita langsung disajikan data yang mengindikasikan bahwa aplikasi BeNews ini adalah sebuah *malware*. Hal ini terlihat pada Sub Bagian Signatures, indikasi bahwa aplikasi adalah *malware* ditandai dengan warna merah (Gambar 4.8).



Gambar 4.8 Sub Bagian Signatures

Sub Bagian Signatures ini berisi beberapa informasi sebagai berikut:

a. *File has been identified by at least one Antivirus on VirusTotal as malicious (Osint)*

Hal ini berarti file benews.apk ini telah teridentifikasi sebagai *malware* oleh paling tidak satu antivirus yang digunakan pada aplikasi VirusTotal. Virus Total adalah sebuah website layanan gratis untuk menganalisis file dan URL yang mencurigakan dan memfasilitasi deteksi cepat virus, worm, trojan dan segala jenis *malware*

b. *Application containts shared object files (static).*

Yang berarti, terdapat *shared object files* didalam aplikasi BeNews.apk ini. *Shared Object Files* adalah unit tak terpisahkan yang dihasilkan dari satu atau lebih objek yang dapat dipindahkan. Shared object ini dapat diikat dengan eksekutabel dinamis untuk membentuk proses yang dapat dijalankan.

c. *Application ask for dangerous permission (static).*

Yang berarti, terdapat permintaan *permission* yang berbahaya apabila diizinkan oleh pengguna.

d. *File has been identified by more the 10 Antivirus on VirusTotal as malicious (Osint).*

Yang berarti, aplikasi BeNews.apk ini telah teridentifikasi sebagai *malware* oleh lebih dari sepuluh AntiVirus yang terdapat di VirusTotal.

e. *Hidden payload found (Static).*

Payload adalah fungsi dari *malware* yang menyebabkan kerusakan. Payload dapat berupa bagian dari *malware* itu sendiri, atau bisa berbentuk pengunduh tersembunyi yang kemudian menciptakan *backdoor* dan memungkinkan serangan oleh *malware* lainnya. Cuckoodroid menemukan beberapa file yang dicurigai sebagai payload tersembunyi yang dimiliki oleh aplikasi BeNews. File-file yang merupakan payload tersembunyi dapat dilihat pada Tabel 4.1.

Tabel 4.1 Tabel Hidden Payload

Nama	Type	Md5	Size
res/layout/ abc_action_menu_item_layout.xml	Android binary XML	d1134391eef349a17c332df6cf11f642	812
res/layout/ fragment_detail_image_view.xml	Android binary XML	61949eb18725dff51b31906182202e4	2288
res/anim/ abc_slide_in_bottom.xml	Android binary XML	c4b64b15bae99397bddf123a083758e2	400
res/anim/abc_fade_out.xml	Android binary XML	3a781127d972ca6805e360696872ce3a	396
res/anim/support_simple_ spinner_dropdown_item.xml	Android binary XML	fc4a6b4cf17c868b2aefb51d1a3912a7	508
res/drawable/abc_ item_background_holo_light.xml	Android binary XML	ef06a1aab4d5eaa03e8058ca65ba8dbf	400
res/layout/abc_activity_chooser_ view_include.xml	Android binary XML	2f0b26bd300ebb34322f123d7e8e8934	400

Dari Tabel 4.1, terlihat beberapa file dengan tipe Android binary XML dicurigai sebagai payload yang tersembunyi dari aplikasi BeNews. File XML dapat berisi informasi tentang apa yang harus diunduh, servis mana yang harus dijalankan. Penyerang dapat menyalahgunakan fitur XML untuk melakukan serangan seperti *Denial Of Service*, mengakses file logis, membuat koneksi ke mesin lain, atau menghindari *firewall*. Kemungkinan serangan akan dilakukan pada saat kita mengakses salah satu file yang tercantum pada Tabel 4.1.

Menu kedua adalah Static Analysis. Bagian ini memberikan informasi penting yaitu hasil analisis statis yang dilakukan oleh Cuckoodroid. Terdapat beberapa sub bagian dari Static

Analysis yaitu Android Static Analysis, Android Static API Calls, Certificate, Files, Strings dan Antivirus.

Dari menu Static Analysis ini, informasi yang mengindikasikan bahwa aplikasi BeNews ini adalah sebuah *malware* yaitu pada kolom Permissions dan kolom Antivirus.

Kolom Permissions berisi informasi tentang izin apa saja yang diminta oleh aplikasi BeNews saat diinstall dan dijalankan. Informasi terkait izin yang diminta oleh aplikasi BeNews dapat dilihat pada Gambar 4.9.



Gambar 4.9 Izin Aplikasi BeNews

Pada kolom Permissions ini semuanya ditulis dengan warna merah. Itu berarti bahwa izin yang dimiliki oleh aplikasi BeNews terindikasi berbahaya. Aplikasi BeNews memiliki tiga Permissions yaitu:

a. android.permission.WRITE_EXTERNAL_STORAGE

Perintah ini meminta izin kepada pengguna agar aplikasi BeNews.apk ini dapat diizinkan untuk menulis didalam *SD Card* dari perangkat.

b. android.permission.INTERNET

Perintah ini meminta izin kepada pengguna agar aplikasi BeNews.apk ini dapat diizinkan untuk membuat *network sockets.*

c. android.permission.READ_PHONE_STATE

Perintah ini meminta izin kepada pengguna agar aplikasi BeNews.apk ini dapat mengakses fitur dari perangkat. Aplikasi dengan izin ini dapat mengetahui nomor telepon yang digunakan perangkat, nomor serial dari perangkat, apakah perangkat sedang melakukan panggilan aktif, nomor telepon yang sedang terhubung dengan perangkat, dan sebagainya.

Kolom selanjutnya adalah kolom Antivirus. Kolom ini berisi informasi data tentang mesin antivirus yang sudah pernah menganalisis aplikasi BeNews ini. Terdapat 60 mesin antivirus dari berbagai perusahaan yang pernah mendeteksi aplikasi BeNews ini. Tidak semua antivirus mendeteksi aplikasi ini sebagai *malware*, namun Cuckoo menyimpulkan bahwa

aplikasi BeNews ini adalah *malware* karena sebanyak 34 mesin antivirus mendeteksinya sebagai *malware* (Tabel 4.2).

Tabel 4.2 Hasil Antivirus

No.	Antivirus	Signatures
1.	MicroWorls-eScan	Android.Riskware.Agent.VW
2.	CAT-Quickheal	Android.Hoaber.GEN22718
3.	AegisLab	SUSPICIOUS
4.	Trustlook	Android.*Malware*.General(8)
5.	BitDefender	Android.Riskware.Agent.VW
6.	K7GW	Trojan (004c8f171)
7.	NANO-Antivirus	Trojan.Android.Hoaber.duivoe
8.	SymantecMobileInsight	Trojan.BeNews
9.	Symantec	Android.BeNews
10.	ESET-NOD32	Android/Nebews.A
11.	Avast	Android.Agent-ITZ[Trj]
12.	Kaspersky	Heur:Trojan.AndroidOS.Hoaber.a
13.	Babable	*Malware*.HighConfidence
14.	Ad-Aware	Android.Riskware.Agent.VW
15.	Emsisoft	Android.Riskware.Agent.VW (B)
16.	F-Secure	Android.Riskware.Agent
17.	DrWeb	Android.Downloader.213
18.	Zillya	Trojan.Nebews:.1
19.	McAfee-GW-Edition	Android/Mekir
20.	Ikarus	Trojan.AndroidOS.Nebews
21.	Cyren	AndroidOS/GenBl.6C120179!Olympus
22.	Avira	ANDROID/Hoaber.A
23.	Fortinet	Android/Hoaber.A!tr
24.	Antivy-AVL	Trojan/Android.Hoaber
25.	Arcabit	Android.Riskware.Agent.VW
26.	ZoneAlarm	HEUR:Trojan.AndroidOS.Hoaber.a
27.	Avast-Mobile	Android:Agent-ITZ[Trj]
28.	Sophos	Andr/Benews-A
29.	AhnLab-V3	Android-Trojan/Htbnews.f32b
30.	McAfee	Artemis!6C120179688CB
31.	Tencent	Dos.Trojan.Hoaber.Wopu
32.	Gdata	Android.Riskware.Agent.VW
33.	AVG	Android!:Agent-ITZ[trj]
34.	Qihoo-360	Trojan.Android.Gen

Dari data pada Tabel 4.2, dapat dilihat bahwa aplikasi BeNews ini memiliki banyak nama. Hal ini berarti aplikasi BeNews hanya menggunakan nama “BeNews” sebagai kedok untuk mengelabui pengguna agar mempercayai dan memasang aplikasi ini pada *smartphone* milik pengguna. 11 dari 34 mesin antivirus mengklasifikasikan aplikasi BeNews kedalam jenis Trojan.

Menu ketiga adalah Network Analysis. Network analysis terdiri dari beberapa kolom yaitu Hosts, DNS, TCP, UDP, HTTP, ICMP, IRC.

Kolom pertama yaitu kolom *Hosts*. Kolom ini berisi informasi tentang IP Address yang digunakan oleh aplikasi BeNews.apk ini. IP yang ditampilkan dapat dilihat pada Gambar 4.10.


Hosts

IP
ff02::16
ff02::1:ff12:3456
ff02::2

Gambar 4.10 Hosts IP

IP yang ditampilkan pada kolom Hosts adalah ff02::16, ff02::1:ff12:3456, ff02::2. Ini adalah merupakan alamat multicast Ipv6 dari aplikasi benews.apk.

Kolom kedua yaitu kolom DNS. DNS adalah kepanjangan dari Domain Name Server, diibaratkan sebagai buku telepon dari internet. Manusia mengakses informasi online melalui nama domain, browser web berinteraksi melalui alamat Internet Protocol (IP), DNS menerjemahkan nama domain ke alamat IP sehingga browser dapat memuat sumber daya internet. Informasi yang tersedia pada kolom DNS ini dapat dilihat pada Gambar 4.11.

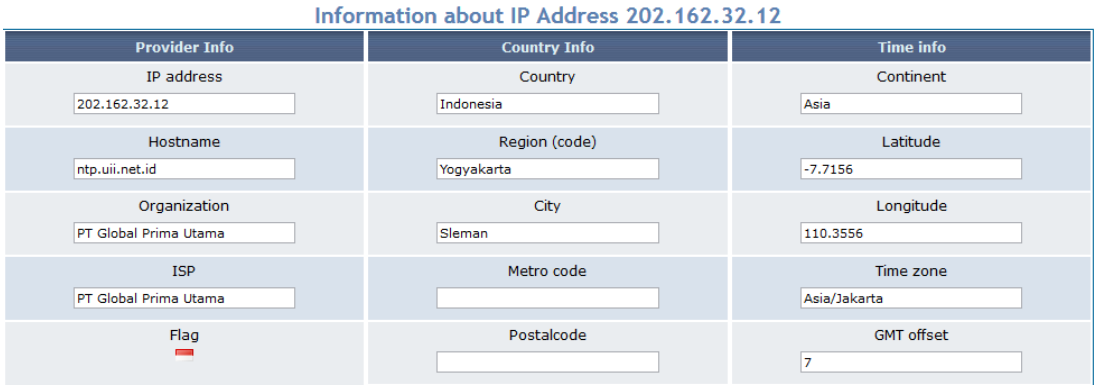

DNS

Name	Response	Post-Analysis Lookup
2.android.pool.ntp.org		202.162.32.12

Gambar 4.11 Kolom DNS

Informasi yang didapatkan adalah nama DNS yaitu 2.android.pool.ntp.org, kolom Response tidak terisi, dan kolom Post-Analysis Lookup berisi 202.162.32.12. 2.android.pool.ntp.org merupakan domain NTP. NTP adalah Network Time Protocol, yaitu protokol internet yang digunakan untuk sinkronisasi waktu milik komputer yang mengacu pada sebuah referensi waktu tertentu. Sebagian besar aplikasi menggunakan infrastruktur server NTP yang disediakan oleh ntp.org. beberapa pengecualian adalah aplikasi yang menggunakan server waktu NIST dan waktu server Qualcomm.izatcloud.net, kemungkinan terkait dengan layanan A-GPS. Pada kolom Post-Analysis Lookup terdapat IP address 202.162.32.12, penguji mencoba melacak IP address tersebut menggunakan website IP Tracker, dan hasilnya adalah

IP address tersebut merupakan IP address milik PT. Global Prima Utama dengan hostname ntp.uii.net.id. Informasi IP Tracker dapat dilihat pada Gambar 4.12.



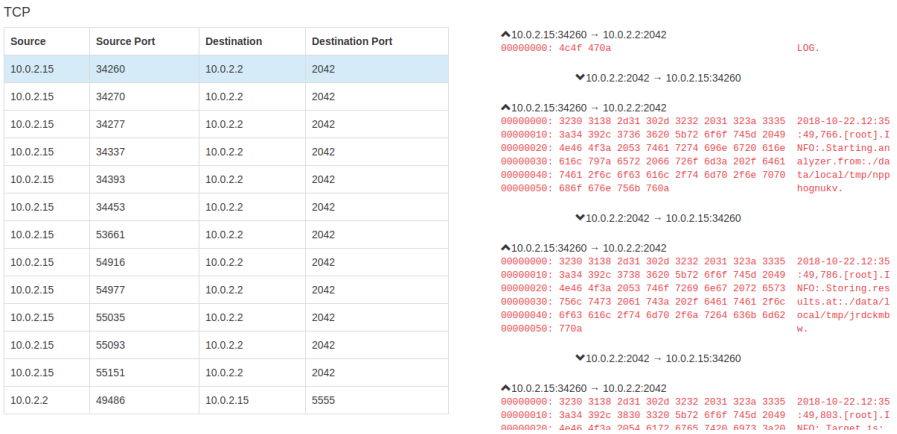
Information about IP Address 202.162.32.12

Provider Info	Country Info	Time info
IP address: 202.162.32.12	Country: Indonesia	Continent: Asia
Hostname: ntp.uii.net.id	Region (code): Yogyakarta	Latitude: -7.7156
Organization: PT Global Prima Utama	City: Sleman	Longitude: 110.3556
ISP: PT Global Prima Utama	Metro code:	Time zone: Asia/Jakarta
Flag	Postalcode:	GMT offset: 7

Gambar 4.12 IP Tracker

Analisis saya, aplikasi benews.apk menggunakan layanan NTP milik PT. Global Prima Utama dengan hostname ntp.uii.net.id sebagai acuan zona waktu saat analisis oleh Cuckoodroid dilakukan.

Selanjutnya adalah kolom TCP. Kolom TCP ini berisi aktifitas IP address yang berkomunikasi melalui port TCP pada saat analisis berlangsung. Kolom TCP ini terdiri dari kolom Source, Source Port, Destination dan Destination Port. Kolom TCP ini dapat dilihat pada Gambar 4.13.



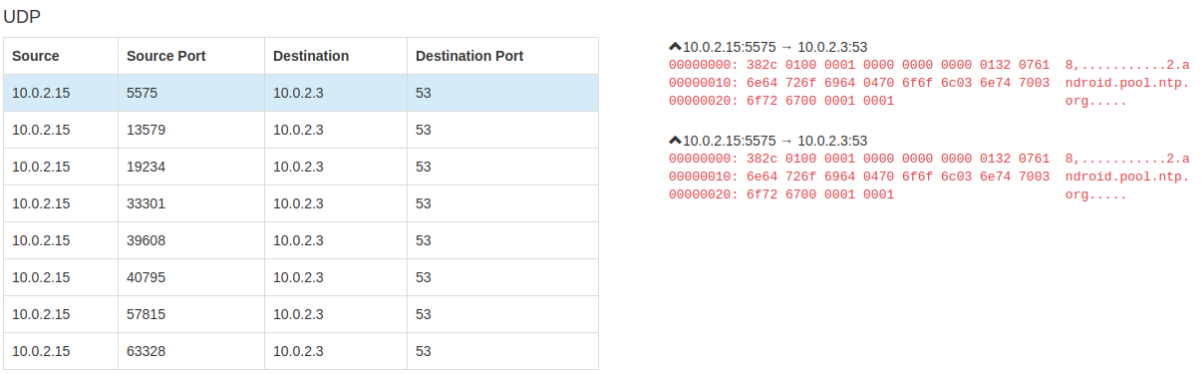
TCP

Source	Source Port	Destination	Destination Port
10.0.2.15	34260	10.0.2.2	2042
10.0.2.15	34270	10.0.2.2	2042
10.0.2.15	34277	10.0.2.2	2042
10.0.2.15	34337	10.0.2.2	2042
10.0.2.15	34393	10.0.2.2	2042
10.0.2.15	34453	10.0.2.2	2042
10.0.2.15	53661	10.0.2.2	2042
10.0.2.15	54916	10.0.2.2	2042
10.0.2.15	54977	10.0.2.2	2042
10.0.2.15	55035	10.0.2.2	2042
10.0.2.15	55093	10.0.2.2	2042
10.0.2.15	55151	10.0.2.2	2042
10.0.2.2	49486	10.0.2.15	5555

Gambar 4.13 Kolom TCP

Pada kolom TCP, kita dapat melihat aktifitas yang menggunakan port TCP yaitu proses analisis pada saat Cuckoodroid dijalankan. Seperti yang terlihat pada baris kedua yaitu alamat IP 10.0.2.15 pada port 34260 menuju alamat IP 10.0.2.15 pada port 2042 dengan informasi pada 2018-10-22 jam 12:35:49, memberikan informasi analisis akan dijalankan.

Kemudian terdapat kolom UDP, yang berisi alamat IP yang berkomunikasi melalui transmisi UDP atau User Datagram Protocol. Informasi yang tersedia pada kolom UDP dapat dilihat pada Gambar 4.14.



UDP

Source	Source Port	Destination	Destination Port
10.0.2.15	5575	10.0.2.3	53
10.0.2.15	13579	10.0.2.3	53
10.0.2.15	19234	10.0.2.3	53
10.0.2.15	33301	10.0.2.3	53
10.0.2.15	39608	10.0.2.3	53
10.0.2.15	40795	10.0.2.3	53
10.0.2.15	57815	10.0.2.3	53
10.0.2.15	63328	10.0.2.3	53

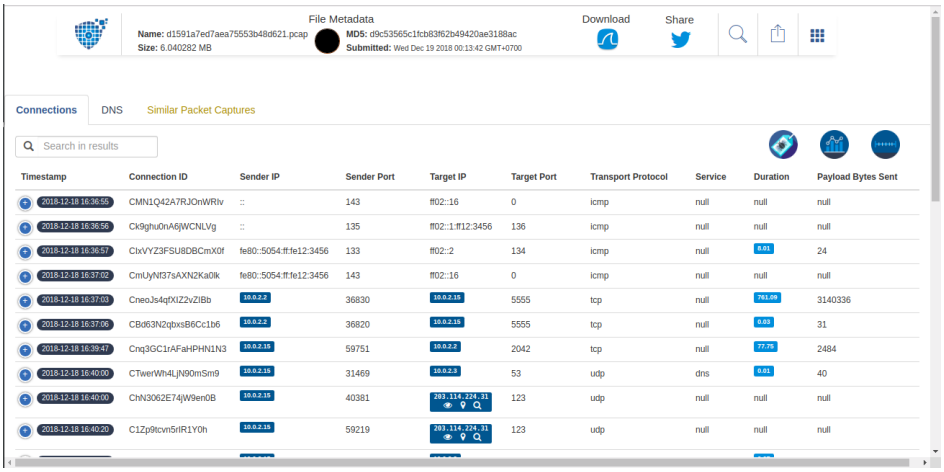
10.0.2.15:5575 → 10.0.2.3:53
00000000: 382c 0100 0001 0000 0000 0000 0132 0761 8,...........2.a
00000010: 6e64 726f 6964 0470 6f6f 6c03 6e74 7003 ndroid.pool.ntp.
00000020: 6f72 6700 0001 0001 org.....

10.0.2.15:5575 → 10.0.2.3:53
00000000: 382c 0100 0001 0000 0000 0000 0132 0761 8,...........2.a
00000010: 6e64 726f 6964 0470 6f6f 6c03 6e74 7003 ndroid.pool.ntp.
00000020: 6f72 6700 0001 0001 org.....

Gambar 4.14 Kolom UDP

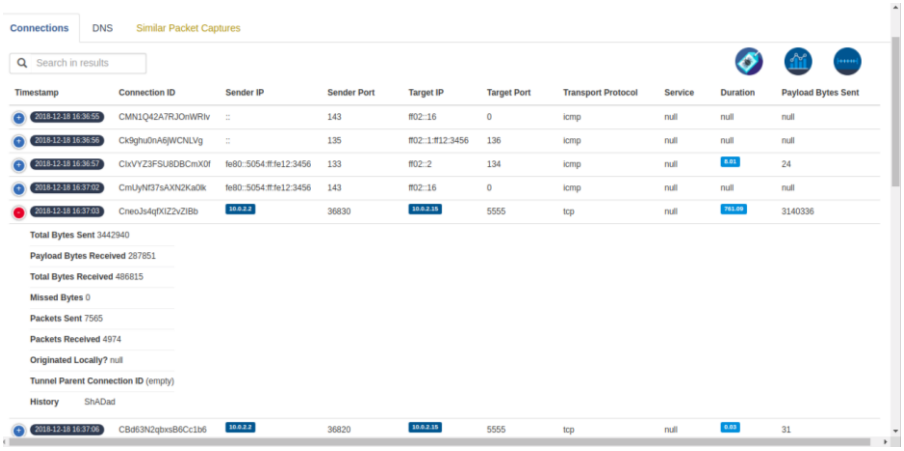
Terlihat pada kolom UDP ini, informasi yang disajikan terkait dengan NTP atau Network Time Protocol, yaitu protokol internet yang digunakan untuk sinkronisasi waktu milik komputer yang mengacu pada sebuah referensi waktu tertentu. Servis NTP yang digunakan adalah 2.android.pool.ntp.org.

Selain itu, dari Network Analysis ini, kita juga bisa mengunduh file PCAP dari sampel *malware*. PCAP atau *Packet Capture* adalah hasil tangkapan dari lalu lintas jaringan. Untuk memvalidasi hasil tangkapan jaringan yang dilakukan oleh cuckoodroid, penguji melakukan analisis melalui halaman web packettotal.com. PacketTotal adalah mesin berbasis web yang khusus menganalisis file berbentuk PCAP. Hasil analisis PacketTotal dapat dilihat pada Gambar 4. 15.



Gambar 4. 15 Hasil Analisis PacketTotal

Yang menjadi perhatian penguji adalah terdapat kolom bernama "*Payload Bytes Sent*". Payload merupakan data yang dibawa didalam *packet* atau data jaringan lainnya seperti *frames* atau *segments*. Ketika dalam konteks untuk sebuah *malware*, payload merujuk pada setiap kode eksekusi berbahaya yang dikirimkan pada target pada saat *malware* telah berhasil menginfeksi targetnya. Salah satu contohnya adalah pada koneksi dengan ID "CneoJs4qfXIZ2vZIBb" yang terlihat pada gambar . Dengan menggunakan protokol TCP, *Payload Bytes Sent* atau bit payload yang terkirim sebanyak 3.140.336 payload. Dan *Payload Bytes Received* atau bit payload yang diterima adalah sebanyak 2.878.851 payload. Hal ini mengindikasikan jika benews.apk ini memuat banyak sekali payload yang berbahaya (Gambar 4.16).



Gambar 4.16 Contoh Payload

4.2.2 Hasil Analisis Volatility

Untuk melakukan analisis pada RAM android, kita menggunakan tools Volatility. Pada tools Volatility ini, terdapat berbagai plugin yang dapat kita gunakan sesuai dengan kebutuhan analisis yang dibutuhkan. Pada analisis kali ini, penulis menggunakan 5 plugin yaitu linux_pslist, linux_psxview, linux_malfind, linux_proc_maps, linux_arp, linux_ifconfig dan linux_route_cache.

Langkah pertama adalah mengeksekusi plugin linux_pslist. Plugin linux_pslist berfungsi untuk menyebutkan proses dengan menjalankan daftar proses yang aktif yang ditunjukkan oleh variabel init_task. Variabel init_task ini dialokasikan secara statis didalam kernel, diinisialisasi saat boot, memiliki PID 0 dan memiliki nama *swapper*. Penggunaan plugin linux_pslist dapat dilihat pada Gambar 4.17.

```
laksa@laksa:~/android-volatility$ python vol.py --profile=Linux_XperiaE3_ARM -f /home/laksa/Xperia_E3/E3-malware.lime linux_pslist
Volatility Foundation Volatility Framework 2.6
Offset     Name                 Pid             PPid            Uid             Gid    DTB        Start Time
---------- -------------------- --------------- --------------- --------------- ------ ---------- ----------
0xf1c58000 init                 1               0               0               0      0x30d00000 2018-10-18 06:11:28 UTC+0000
0xf1c58500 kthreadd             2               0               0               0      ---------- 2018-10-18 06:11:28 UTC+0000
```

Gambar 4.17 Plugin linux_pslist

Pada saat mengeksekusi plugin linux_pslist, kita dapat melihat proses milik aplikasi BeNews.apk yang sudah diinstall dan dijalankan. Proses milik BeNews.apk ini adalah BeNews, dengan PID 20380, berjalan pertama kali pada 2018-10-18 pukul 15.52.34. informasi proses BeNews yang terekam pada linux_pslist dapat dilihat pada Gambar 4.18.

```
0xd28d0a00 kworker/u:17         20288           2               0               0      ---------- 2018-10-18 15:52:32 UTC+0000
0xc702a300 org.benews           20380           210             -               10189  0x07bdc000 2018-10-18 15:52:34 UTC+0000
0xc2402800 .managerservice      20530           210             -               10149  0x17c00000 2018-10-18 15:53:10 UTC+0000
```

Gambar 4.18 Proses BeNews.apk

Plugin kedua yang digunakan adalah linux_psxview. Plugin linux_psxview ini berfungsi untuk memberikan referensi proses dari berbagai sumber dan dapat menemukan proses yang tersembunyi pada sistem. Eksekusi plugin linux_psxview dapat dilihat pada Gambar 4.19.

```
laksa@laksa:~/android-volatility$ python vol.py --profile=Linux_XperiaE3_ARM -f /home/laksa/Xperia_E3/E3-malware.lime linux_psxview
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                    PID pslist psscan pid_hash kmem_cache parents leaders
---------- -------------------- ------ ------ ------ -------- ---------- ------- -------
INFO    : volatility.debug    : SLUB is currently unsupported.
0x18afd500 erprise.service        2246 True   True   True     False      False   True
0x31c5f300 khelper                  21 True   True   True     False      False   True
```

Gambar 4.19 Plugin linux_psxview

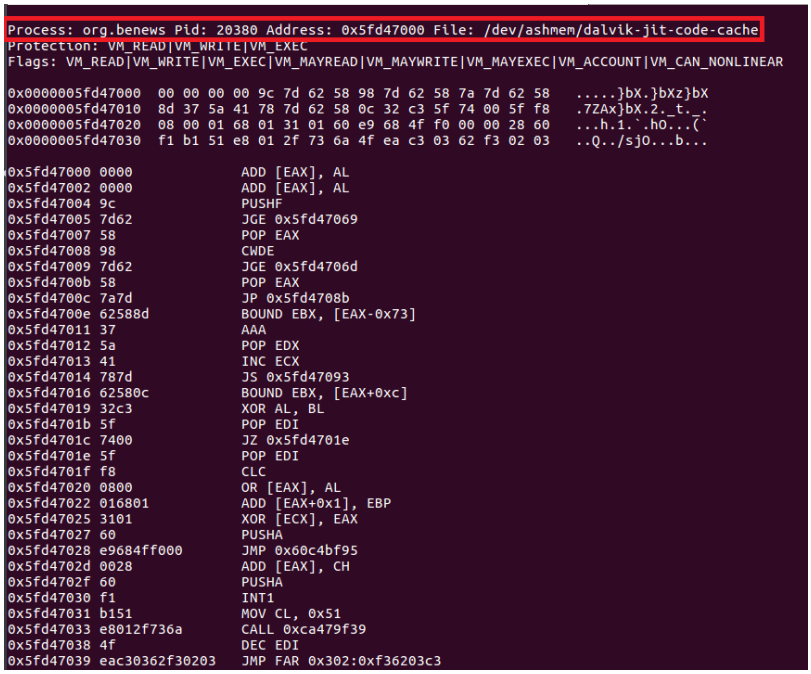
Proses milik BeNews yang tertangkap melalui plugin linux psxview dapat dilihat pada Gambar 4.20.

```
0x2596f800 d.process.media       1344 True   True   True     False      False   True
0x0702a300 org.benews           20380 True   True   True     False      False   True
0x2f429900 qrngd                  233 True   True   True     False      False   True
```

Gambar 4.20 proses benews pada plugin psxview

Pada Gambar 4.20, terlihat informasi milik proses BeNews.apk yaitu offset 0x0702a300 dengan nama BeNews dan PID 20380. Pada beberapa label yaitu label PSLIST, PSSCAN, PID_HASH dan LEADERS bernilai true yang berarti proses milik benews 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, aplikasi benews belum terlihat sebagai *malware*.

Selanjutnya adalah mengeksekusi plugin linux_malfind. Plugin linux_malfind adalah plugin yang mencari file executable berbahaya dan shellcode di dalam setiap proses. Hasil eksekusi plugin linux_malfind dapat dilihat pada Gambar 4.21.



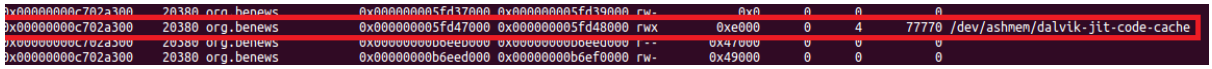
Gambar 4.21 Plugin linux_malfind

Dari hasil eksekusi plugin linux_malfind, kita menemukan bahwa target dari proses yang kita cari yaitu BeNews dengan PID 20380, memiliki wilayah memori pada alamat 0x5fd47000 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 area yang berisi bit-bit yang menentukan perilaku area memori dan memberikan informasi tentang halaman-halaman yang terdapat di area memori. Pada area memori dengan alamat 0x5fd47000 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 bendera VM_READ dapat diatur, VM_MAYWRITE yang berarti bendera VM_WRITE dapat diatur, VM_MAYEXEC yang berarti bendera VM_EXEC dapat diatur, VM_ACCOUNT yang berarti area memori ini adalah objek yang memiliki akun, VM_CAN_NONLINEAR yang berarti area memori ini adalah pemetaan nonlinear. Adanya perintah VM_READ, VM_WRITE dan VM_EXEC yang berarti memori dapat dibaca, ditulis dan dieksekusi, mengindikasikan bahwa program dapat menulis payload didalam alokasi memori dan mengeksekusinya.

Terdapat indikator bahwa bagian memori ini melakukan kegiatan berbahaya yaitu terdapat PUSH dan POP. Kegiatan PUSH dan POP ini mengindikasikan bahwa program akan berpindah ke stack memori lain, namun di antara proses PUSH dan POP, program melakukan eksekusi kode berbahaya dulu.

Kemudian terdapat aktifitas yang dilakukan oleh proses BeNews dari area memori 0x5fd47000 sampai dengan area memori 0x5fd47030. Aktifitas ini terindikasi berbahaya yang terlihat dari adanya signature pada area itu yang ditandai dengan "}bx" dan seterusnya. Dibawahnya terdapat kode dalam bahasa assembly yang menjelaskan maksud dari aktifitas area memori ini.

Untuk membuktikan apakah benar terdapat area memori 0x5fd47000 yang ditemukan oleh plugin linux_malfind karena terindikasi berbahaya pada proses BeNews, kita akan menggunakan plugin linux_procmaps. Plugin linux_procmaps ini berfungsi untuk mencetak detail proses memori, termasuk tumpukan-tumpukan dan pustaka bersama. Kita akan melihat detail proses dari proses BeNews dengan PID 20380 pada Gambar 4.22.



Gambar 4.22 Plugin linux_procmaps

Dari hasil dari eksekusi plugin linux_proc_maps, kita menemukan bahwa proses yang ditemukan oleh linux_malfind karena terindikasi berbahaya yaitu proses dengan alamat 0x5fd47000 memang berjalan pada ram dan bagian dari proses milik BeNews. informasi yang didapat dari eksekusi plugin linux_proc_maps ini adalah:

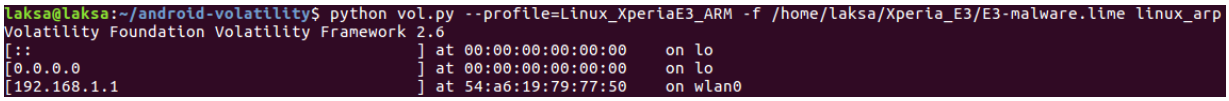
Tabel 4.3 Tabel proc_maps

Offset	PID	Name	Start	End	Flags	Pgoff	Major	Minor	Inode	File Path
0x00000000c702a300	20380	BeNews	0x000000005fd47000	0x000000005fd48000	Rwx	0xe000	0	4	77770	/dev/ashmem/dalvik-jit-code-cache

Dengan hasil pada Tabel 4.3, maka temuan dari analisis plugin linux_malfind dapat dibuktikan dengan eksekusi plugin linux_proc_maps. Dan dapat disimpulkan bahwa file /dev/ashmem/dalvik-jit-code-cache melakukan aktifitas berbahaya hingga terdeteksi oleh plugin linux_malfind.

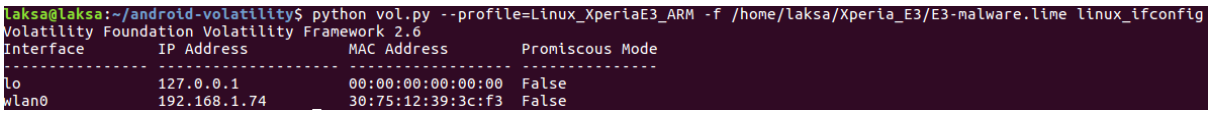
Pada pengujian untuk bagian jaringan, penguji menggunakan plugin linux_arp, linux_ifconfig dan linux_route_cache. Address Resolution Protocol (ARP) adalah protokol untuk memetakan alamat protokol internet (alamat IP) ke alamat mesin fisik yang dikenali pada jaringan lokal. Sebuah tabel yang biasa disebut ARP cache digunakan untuk menjaga korelasi antara setiap alamat IP yang sesuai. ARP menyediakan aturan protokol untuk membuat korelasi dan memberikan konversi alamat di kedua arah. Plugin linux_arp pada Volatility berfungsi

untuk mencetak tabel ARP pada RAM. Tabel ARP dari RAM perangkat target dapat dilihat pada Gambar 4.23.



Gambar 4.23 Tabel ARP

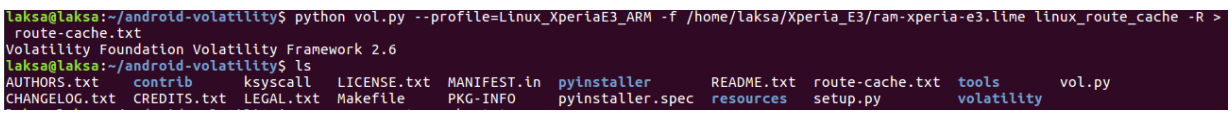
Kemudian penguji menggunakan plugin linux_ifconfig untuk mengetahui ip address dari perangkat saat RAM diakuisisi. Plugin linux_ifconfig berfungsi untuk mencetak informasi antarmuka aktif, termasik alamat IP, nama antarmuka, alamat MAC, dan apakah NIC dalam mode promiscuous atau tidak (sniffing). Informasi yang ditampilkan plugin linux_ifconfifg dapat dilihat pada Gambar 4.24.



Gambar 4.24 Tabel Ifconfig

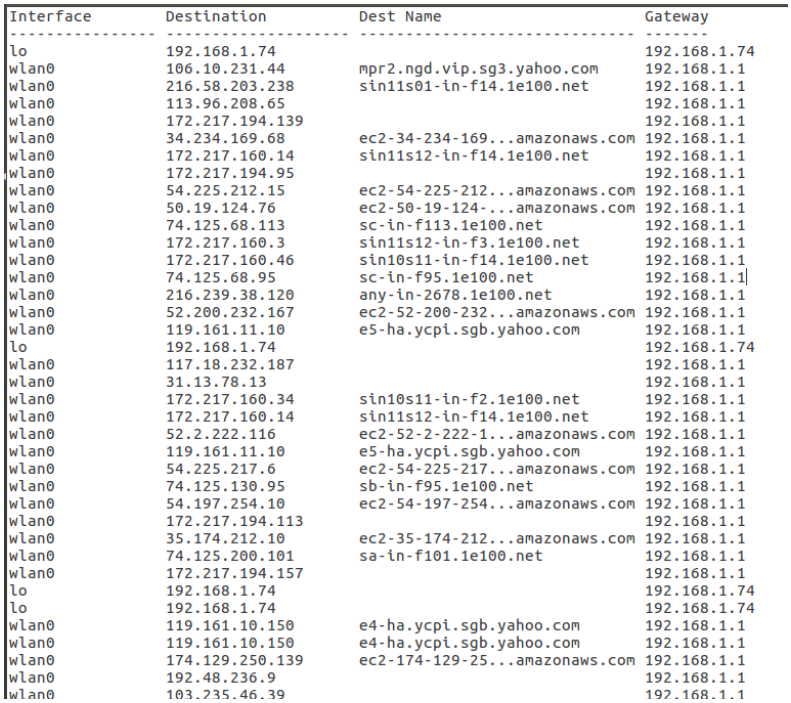
Dari Gambar 4.24, kita dapat mengetahui bahwa perangkat saat itu terhubung pada jaringan wifi dengan menggunakan IP 192.168.1.74 dengan alamat MAC 30:75:12:39:3c:f3 dan tidak sedang dalam mode *sniffing*. *Sniffing* adalah proses memantau dan menangkap semua paket data yang melewati jaringan yang diberikan. Penyerang menggunakan sniffer untuk menangkap paket data yang berisi informasi sensitif seperti kata sandi, informasi akun, dll. Sniffer dapat berupa perangkat keras atau perangkat lunak yang dipasang pada sistem. Dengan menempatkan paket sniffer pada jaringan dalam mode promiscuous, penyusup atau penyerang dapat menangkap dan menganalisis semua lalu lintas jaringan. Hal ini berarti file *malware* yang sedang diuji tidak sedang melakukan sniffing kepada perangkat.

Kemudian penguji menggunakan plugin linux_route_cache. Plugin ini berfungsi untuk mencetak data dalam tabel *cache routing* yang dapat menunjukkan kepada kita sistem mana yang berkomunikasi dengan perangkat dimasa lalu. Hasil dari tabel *cache routing* ini penguji cetak kedalam file yang dinamakan route-cache.txt yang terlihat pada Gambar 4.25.



Gambar 4.25 proses dump route-cache

Hasil dari *dumping* tabel *cache routing* ini adalah kita dapat meihat komunikasi dari *smartphone* target saat sebelum diakuisisi. Data dari tabel *cache routing* ini terlihat dalam Gambar 4.26.



Interface	Destination	Dest Name	Gateway
lo	192.168.1.74		192.168.1.74
wlan0	106.10.231.44	mpr2.ngd.vip.sg3.yahoo.com	192.168.1.1
wlan0	216.58.203.238	sin11s01-in-f14.1e100.net	192.168.1.1
wlan0	113.96.208.65		192.168.1.1
wlan0	172.217.194.139		192.168.1.1
wlan0	34.234.169.68	ec2-34-234-169...amazonaws.com	192.168.1.1
wlan0	172.217.160.14	sin11s12-in-f14.1e100.net	192.168.1.1
wlan0	172.217.194.95		192.168.1.1
wlan0	54.225.212.15	ec2-54-225-212...amazonaws.com	192.168.1.1
wlan0	50.19.124.76	ec2-50-19-124-...amazonaws.com	192.168.1.1
wlan0	74.125.68.113	sc-in-f113.1e100.net	192.168.1.1
wlan0	172.217.160.3	sin11s12-in-f3.1e100.net	192.168.1.1
wlan0	172.217.160.46	sin10s11-in-f14.1e100.net	192.168.1.1
wlan0	74.125.68.95	sc-in-f95.1e100.net	192.168.1.1
wlan0	216.239.38.120	any-in-2678.1e100.net	192.168.1.1
wlan0	52.200.232.167	ec2-52-200-232...amazonaws.com	192.168.1.1
wlan0	119.161.11.10	e5-ha.ycpi.sgb.yahoo.com	192.168.1.1
lo	192.168.1.74		192.168.1.74
wlan0	117.18.232.187		192.168.1.1
wlan0	31.13.78.13		192.168.1.1
wlan0	172.217.160.34	sin10s11-in-f2.1e100.net	192.168.1.1
wlan0	172.217.160.14	sin11s12-in-f14.1e100.net	192.168.1.1
wlan0	52.2.222.116	ec2-52-2-222-1...amazonaws.com	192.168.1.1
wlan0	119.161.11.10	e5-ha.ycpi.sgb.yahoo.com	192.168.1.1
wlan0	54.225.217.6	ec2-54-225-217...amazonaws.com	192.168.1.1
wlan0	74.125.130.95	sb-in-f95.1e100.net	192.168.1.1
wlan0	54.197.254.10	ec2-54-197-254...amazonaws.com	192.168.1.1
wlan0	172.217.194.113		192.168.1.1
wlan0	35.174.212.10	ec2-35-174-212...amazonaws.com	192.168.1.1
wlan0	74.125.200.101	sa-in-f101.1e100.net	192.168.1.1
wlan0	172.217.194.157		192.168.1.1
lo	192.168.1.74		192.168.1.74
lo	192.168.1.74		192.168.1.74
wlan0	119.161.10.150	e4-ha.ycpi.sgb.yahoo.com	192.168.1.1
wlan0	119.161.10.150	e4-ha.ycpi.sgb.yahoo.com	192.168.1.1
wlan0	174.129.250.139	ec2-174-129-25...amazonaws.com	192.168.1.1
wlan0	192.48.236.9		192.168.1.1
wlan0	103.235.46.39		192.168.1.1

Gambar 4.26 Tabel *cache routing*

Dapat kita lihat bahwa mesin yang terhubung pada wlan0 melakukan komunikasi kepada beberapa destinasi. Namun tidak terdapat indikasi bahwa komunikasi tersebut dilakukan oleh BeNews.com karena aplikasi BeNews.com hanya menggunakan ip address 202.162.32.12 yang merupakan *network time protocol*.

Hasil temuan dari kedua analisis ini dapat dilihat pada Tabel 4.4.

Tabel 4.4 Tabel Perbandingan Hasil Temuan Analisis

No.	Cuckoodroid	RAM Android
1.	Cuckoodroid menemukan permission berbahaya yang diajukan oleh aplikasi BeNews saat diinstall kedalam perangkat, yaitu: • Android.permission.WRITE_EXTERNAL_STORAGE • Android.permission.INTERNET • Android.permission.READ_PHONE_STATE	Eksekusi plugin linux_malfind menemukan alamat 0x5fd47000 pada memori yang dapat dibaca, ditulis dan dieksekusi. Hal ini mengindikasikan bahwa program dapat menulis payload didalam

		alokasi memori dan kemudian mengeksekusinya.
2.	Cuckoodroid menemukan payload tersembunyi dari aplikasi BeNews yang dapat dilihat pada menu *Hidden Payload Found*.	Alamat memori 0x5fd4700 terdapat kegiatan PUSH dan POP yang mengindikasikan program melakukan eksekusi kode berbahaya kedalam memori.
3.	Cuckoodroid menemukan aplikasi BeNews telah dikategorikan sebagai *malware* oleh 34 mesin antivirus.	Eksekusi plugin linux_arp, linux_ifconfig dan linux_route_cache tidak menemukan adanya pengaruh aplikasi BeNews terhadap komunikasi jaringan milik *smartphone*.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melakukan penelitian menggunakan tools Cuckoodroid dan Volatility dengan objek *malware* yang digunakan adalah file BeNews, penguji menarik kesimpulan sebagai berikut:

a. Berhasil membangun lingkungan penelitian menggunakan mesin linux dengan sistem operasi Ubuntu 16.04 LTS. Cuckoodroid dapat diinstal dengan baik diatas Cuckoo Sandbox dan dapat terintegrasi dengan baik kepada emulator android sebagai wadah eksekusi dan analisis sampel *malware* yaitu BeNews.
b. Berhasil mengimplementasikan teknik akuisisi RAM pada perangkat android melalui koneksi TCP pada port 4444 dengan bantuan kabel USB. Hasil dari akuisisi RAM adalah file dengan nama ram-xperia-e3.lime yang kemudian dianalisis menggunakan Volatility.
c. Berhasil melakukan analisis pada RAM android menggunakan tools Volatility dengan memanfaatkan beberapa plugin yang tersedia yaitu linux_pslist, linux_psxview, linux_proc_maps, linux_arp, linux_ifconfig, dan linux_route_cache.
d. Hasil dari perbandingan analisis *malware* yang dieksekusi pada Cuckoodroid dan perangkat android adalah:
    1. Dari kedua tools, dapat mengetahui nama proses dari aplikasi BeNews yaitu "BeNews". pada Cuckoodroid, nama proses dapat dilihat pada sub bagian Android Application yaitu pada kolom package, sedangkan pada Volatility, nama proses dapat dilihat menggunakan plugin linux_pslist dan linux_psxview.
    2. Cuckoodroid mengkategorikan aplikasi BeNews sebagai *malware* karena memiliki *permissions* yang dianggap berbahaya, dan aplikasi BeNews ini sudah terdeteksi oleh beberapa antivirus lain dan dianggap sebagai *malware*.
    3. Sedangkan, hasil dari Volatility menunjukkan adanya proses mencurigakan dari aplikasi BeNews yang berjalan pada RAM. Proses mencurigakan ini terdeteksi oleh plugin linux_malfind. Proses mencurigakan tersebut berada pada file /dev/ashmem/dalvik-jit-code-cache. Proses mencurigakan tersebut dibuktikan apakah merupakan bagian dari proses BeNews dengan menggunakan plugin linux_proc_maps.

4. Pada analisis Cuckoo Sandbox ditemukan alamat IP 202.162.32.12 yang merupakan Network Time Protocol milik aplikasi BeNews dan pada analisis Volatility, ditemukan beberapa aktifitas komunikasi perangkat yang dideteksi oleh plugin linux_route_cache. Namun tidak terdapat komunikasi yang mencurigakan, dan perangkat tidak sedang dalam mode sniffing atau diawasi. Hal ini berarti aplikasi BeNews tidak melakukan hal mencurigakan melalui jaringan internet.

5.2 Saran

Penelitian ini masih terdapat banyak kekurangan, dan membutuhkan pemahaman lebih baik lagi untuk menghasilkan laporan analisis yang baik dan dapat dimengerti oleh orang awam. Oleh karena itu, penulis memberikan beberapa saran untuk penelitian kedepannya yaitu:

a. Melakukan eksplorasi dan pencarian terhadap *tools* lain untuk analisis *malware*.
b. Mempelajari lebih lanjut fungsi dan kegunaan Volatility, karena butuh pemahaman mendalam untuk melakukan analisis menggunakan *tools* Volatility karena mencakup bidang ilmu yang sangat luas.
c. Melakukan eksplorasi terhadap jenis-jenis *malware* yang berbeda karena perkembangan kejahatan siber menggunakan *malware* semakin canggih dan mutakhir.

DAFTAR PUSTAKA

504ENSICS Labs. (2012). Linux Memory Extractor Instructions v1.1, 1–8.

Alghamdi, R., Alfalqi, K., & Waqdan, M. (2015). Android Platform *Malware* Analysis. *(IJACSA) International Journal of Advanced Computer Science and Applications*, *6*(1), 140–146.

Alonso-Parrizas, A. (2015). InfoSec Reading Room Forensic Analysis On Android : A Practical Case.

Faheem, M., Le-Khac, N.-A., & Kechadi, T. (2014). Smartphone Forensic Analysis: A Case Study for Obtaining Root Access of an Android Samsung S3 Device and Analyse the Image without an Expensive Commercial Tool. *Journal of Information Security*, *05*(03), 83–90. https://doi.org/10.4236/jis.2014.53009

Grover, J. (2013). Android forensics: Automated data collection and reporting from a mobile device. *Digital Investigation*, *10*(SUPPL.), S12–S20. https://doi.org/10.1016/j.diin.2013.06.002

Hreirati, O., Iqbal, S., & Zulkarnine, M. (2018). An Adaptive Dataset for the Evaluation of Android *Malware* Detection Techniques. *4th International Conference on Software Security and Assurance (ICSSA), IEEE*.

Katz, M. (2013). Android Forensics.

Ligh, M. H., Case, A., Levy, J., & Walters, Aa. (2016). *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux and Mac Memory*. https://doi.org/10.1017/CBO9781107415324.004

Novrianda, R., Kunang, Y. N., & P.H. Shaksono. (2017). Analisis Forensik *Malware* pada Platform Android. *International Journal of Computer Science and Information Security*, *15*(1), 173–178. https://doi.org/10.13140/RG.2.1.2967.0003

Son, N., Lee, Y., Kim, D., James, J. I., Lee, S., Lee, K., & Lee, K. (2013). A Study of User Data Integrity During Acquisition of Android Devices By devices.

Spreitzenbarth, M., Freiling, F., Echtler, F., Schreck, T., & Hoffmann, J. (2013). Mobile-sandbox: Having a Deeper Look into Android Applications. *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, 1808. https://doi.org/10.1145/2480362.2480701

Sylve, J., Case, A., Marziale, L., & Richard, G. G. (2012). Acquisition and Analysis of Volatile

Memory from Android Devices. *Digital Investigation*, *8*(3–4), 175–184. https://doi.org/10.1016/j.diin.2011.10.003

Wang, X., Yang, Y., & Zeng, Y. (2015). Accurate Mobile *Malware* Detection and Classification in the Cloud. *SpringerPlus*, *4*(1), 1–23. https://doi.org/10.1186/s40064-015-1356-1

Xenakis, C., & Ntantogian, C. (2013). Acquisition and Analysis of Android Memory.

Yang, S. J., Choi, J. H., Kim, K. B., Bhatia, R., Saltaformaggio, B., & Xu, D. (2017). Live Acquisition of Main Memory Data from Android Smartphones and Smartwatches. *Digital Investigation*, *23*, 50–62. https://doi.org/10.1016/j.diin.2017.09.003

Yang, S. J., Choi, J. H., Kim, K. B., & Chang, T. (2015). New Acquisition Method Based on Firmware Update Protocols for Android Smartphones. *Digital Investigation*, *14*(S1), S68–S76. https://doi.org/10.1016/j.diin.2015.05.008

LAMPIRAN

A. Instalasi Cuckoodroid

Instalasi tools diawali dengan menginstall Cuckoo Sandbox pada mesin Linux. Cuckoo Sandbox yang digunakan pada penelitian ini adalah Cuckoo Sandbox versi 1.2. Untuk menginstall Cuckoo Sandbox, penguji harus menginstall beberapa persyaratan.

Komponen host Cuckoo Sandbox dibangun menggunakan Python, oleh karena itu mesin harus memiliki python terinstal didalamnya. Python versi 2.7 lebih diutamakan. Untuk menginstal python, kita dapat menggunakan perintah dari terminal seperti pada Gambar 6.1.

```
laksa@laksa:~$ sudo apt-get install python
```

Gambar 6.1 Instalasi Python

Pada penelitian ini, penguji menggunakan MongoDB sebagai database untuk Cuckoo Sandbox. Oleh karena itu, mesin harus diinstall menggunakan perintah seperti pada Gambar 6.2.

```
laksa@laksa:~$ sudo apt-get install mongodb
```

Gambar 6.2 Instalasi mongodb

Agar fungsionalitas Cuckoo Sandbox dapat berjalan dengan baik, Cuckoo Sandbox membutuhkan SQLAlchemy dan Python BSON untuk diinstal. Instalasi SQLAlchemy dan Python BSON menggunakan perintah seperti pada Gambar 6.3.

```
laksa@laksa:~$ sudo apt-get install python-sqlalchemy python-bson
```

Gambar 6.3 Instalasi SQLAlchemy dan BSON

Terdapat dependensi opsional lainnya yang sebagian besar digunakan oleh modul dan utilitas. Dependensi berikut walaupun tidak harus diinstal pada mesin, tetapi sangat direkomendasikan yaitu:

a. Dpkt (sangat direkomendasikan) : untuk mengekstraksi informasi yang relevan dari file PCAP.
b. Jinja2 (Sangat direkomendasikan) : untuk render halaman HTML laporan dan antarmuka web.

c. Magic (opsional) : untuk mengidentifikasi format file.

d. Pydeep (opsional) : untuk menghitunh *ssdeep fuzzy hash* dari file.

e. Pymongo (opsional) : untuk menyimpan hasil analisis kedalam basis data MongoDB.

f. Yara dan Yara Python (opsional) : untuk mencocokkan tanda tangan Yara.

g. Libvirt (opsional) : untuk menggunakan mesin manajer KVM.

h. Bottlepy (opsional) : untuk menggunakan utilitas *api.py* dan *web.py*

i. Django (opsional) : untuk menggunakan antar muka web.

j. Pefile (opsional) : digunakan untuk analisis statis dari binary PE32.

k. Volatility (opsional) : digunakan untuk analisis forensik pada memori.

l. MAEC Python bindings (opsional) : digunakan untuk laporan MAEC.

m. Chardet (opsional) : untuk mendeteksi pengodean string.

Beberapa dari dependensi diatas sudah tersedia dalam bentuk paket pada Debian / Ubuntu dan dapat diinstal dengan perintah pada Gambar 6.4:

```
laksa@laksa:~$ sudo apt-get install python-dpkt python-jinja2 python-magic python-pymongo python-gridfs python-libvirt
python-bottle python-pefile python-chardet
```

Gambar 6.4 Instalasi paket opsional

Langkah selanjutnya adalah menginstall Cuckoo Sandbox dan Cuckoodroid. Langkah untuk mengunduh dan menginstall Cuckoo dan Cuckoodroid dengan melalui terminal seperti pada Gambar 6.5.

```
git config –global user.email “rizkylaksamana@gmail.com”
git config –global user.name “Laksa”
git clone –depth=1 https://github.com/Cuckoobox/Cuckoo.git Cuckoo -b 1.2
cd Cuckoo
git remote add droid https://github.com /idanr1986/Cuckoo-droid
git pull –allow-unrelated-histories –no-edit -s recursive -X theirs droid master
cat conf-extra/processing.conf >> conf/processing.conf
cat conf-extra/reporting.conf >> conf/reporting.conf
rm -r conf-extra
echo “protobuff” >> requirements.txt
```

Gambar 6.5 Instalasi Cuckoo dan Cuckoodroid

Setelah proses unduh dan instalasi selesai, langkah selanjutnya adalah melakukan konfigurasi file dari Cuckoo agar bisa dijalankan. Beberapa file yang harus di konfigurasi adalah:

a. Konfigurasi skrip Cuckoo.conf seperti pada Gambar 6.6.

```
machinery = avd

[resultserver]
ip = 127.0.0.1
```

Gambar 6.6 Konfigurasi skrip Cuckoo.conf

b. Konfigurasi skrip avd.conf seperti pada Gambar 6.7.

```
[avd]
# Path to the local installation of the android emulator
emulator_path = /home/laksa/sdk/tools/emulator

# Path to the local installation of adb - android debug bridge utility
adb_path = /home/laksa/sdk/platform-tools/adb

# Path to the emulator machine files is located
avd_path = /root/.android/avd

# name of the reference machine that is used to duplicate
refereance_machine = aosx

# Specify a comma-separated list of available machines to be used. For each
# specified ID you have to define a dedicated section containing the details
# on the respective machine. (E.g. aosx_1,aosx_2,aosx_3)
# currently supports only 1 machine for network limitations
machines = aosx_1

[aosx_1]
# Specify the label name of the current machine as specified in your
# aosx_1 configuration.
label = aosx_1

# Specify the operating system platform used by current machine
platform = android

# Specify the IP address of the current virtual machine. Make sure that the
# IP address is valid and that the host machine is able to reach it. If not,
# the analysis will fail.
# its always 127.0.0.1 because android emulator netowrking configuration this the
   loopback of the host machine
ip = 127.0.0.1
```

Gambar 6.7 Konfigurasi skrip avd.conf

c. konfigurasi skrip auxiliary.conf seperti pada Gambar 6.8.

```
[sniffer]
# Enable or disable the use of an external sniffer (tcpdump) (yes/no).
enabled = no
```

Gambar 6.8 Konfigurasi skrip auxiliary.conf

d. Konfigurasi skrip processing.conf seperti pada Gambar 6.9

```
[droidmon]
Enabled = yes

[googleplay]
Enabled = yes
Android_id = <add android_id>
Google_login = <add google_login>
Google_password = <add google_password>

[apkinfo]
Enabled = yes
```

Gambar 6.9 Konfigurasi skrip processing.conf

e. Konfigurasi skrip reporting.conf seperti pada gambar Gambar 6.10.

```
[reporthtml]
Enabled = no

[reportandroidhtml]
Enabled = yes
```

Gambar 6.10 Konfigurasi skrip reporting.conf

Cuckoo Sandbox merupakan perangkat lunak manajemen pusat untuk eksekusi dan analisis sampel. Setiap analisis dijalankan dalam mesin virtual baru dan terisolasi. Infrastruktur Cuckoo terdiri dari mesin host (perangkat lunak manajemen) dan mesin guest (mesin virtual yang melakukan analisis). Mesin host menjalankan komponen inti dari sandbox yang mengelola seluruh proses analisis, sementara mesin guest adalah lingkungan yang terisolasi dimana sampel *malware* dijalankan dengan aman dan kemudian dianalisis.

Setiap mesin guest terdiri dari emulator yang dikendalikan oleh modul mesin *avd.py*. Komponen tambahan yang dipasang di dalam emulator untuk mendukung proses analisis adalah:

a. Python 2.7 dikompilasi menjadi arm.
b. Skrip *Agent.py* yang dimodifikasi untuk emulator android
c. Komponen analisis android yang dikirim ke mesin guest pada awal analisis
d. Xposed, yaitu sebuah *framework* untuk modul yang dapat mengubah perilaku sistem dan aplikasi tanpa mempengaruhi APK apapun. Adapun dua modul tambahan yang dibutuhkan dalam *framework* ini adalah:
    1. Droidmon, yaitu modul pemantauan panggilan API Dalvik.
    2. Anti-deteksi Emulator, yaitu koleksi teknik anti-deteksi yang dikenal untuk menyembunyikan emulator android.
e. Aplikasi Superuser, untuk memberi dan mengelola hak superuser untuk emulator
f. Generator Konten, untuk menghasilkan daftar kontak acak untuk tampilan yang lebih realistis.
g. AAPT Arm, yaitu alat pengemas aset android untuk mengekstraksi aktivitas utama dan nama paket dari APK.

Untuk membuat Cuckoo dan android emulator berjalan dengan baik, beberapa dependensi dan *library* perlu diinstal pada mesin host Cuckoo yang disajikan pada gambar Gambar 6.11.

```
$ sudo apt-get install openjdk-8-jre libstdc++6:i386 libgcc1:i386 zlib1g:i386
libncurses5:i386
```

Gambar 6.11 Instalasi dependensi dan *library*

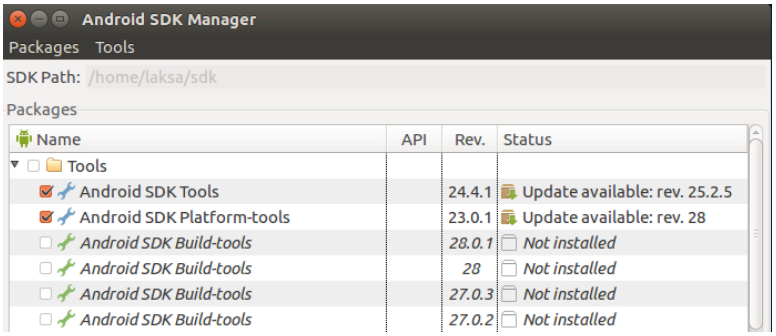
Android SDK adalah syarat wajib yang dibutuhkan untuk komponen Cuckoo *android_on_linux guest (analyzer)* agar dapat berjalan dengan baik. Android SDK yang digunakan adalah *android-sdk_r24.0.2-linux.tgz.* Unduh dan ekstrak file *android-sdk_r24.0.2-linux.tgz* pada direktori tertentu dengan perintah pada gambar Gambar 6.12:

```
$ tar -xvf android-sdk_r24.0.2-linux.tgz
$ mv android-sdk-linux sdk
$ cd sdk/
$ cd tools/
```

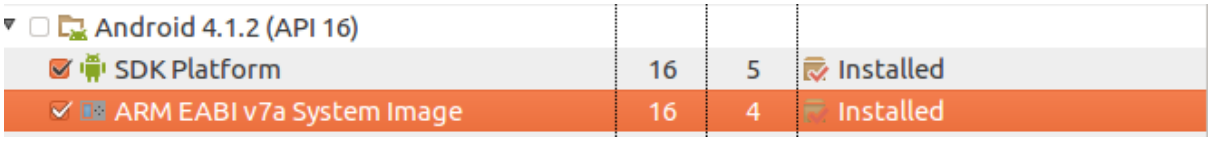
Gambar 6.12 Ekstraksi android SDK

Buka aplikasi Android SDK Manager yang berada pada direktori tools dengan perintah "python android", tampilan jendela dari Android SDK Manager yang terbuka akan terlihat seperti pada Gambar 6.13. Lalu install komponen berikut:

a. Tools
    3. Android SDK Tools
    4. Android Platform Tools
    5. Android Build Tools
b. Android 4.1.2 (API 16) (Gambar 6.14)
    6. SDK Platform
    7. ARM EABI v7a System Image

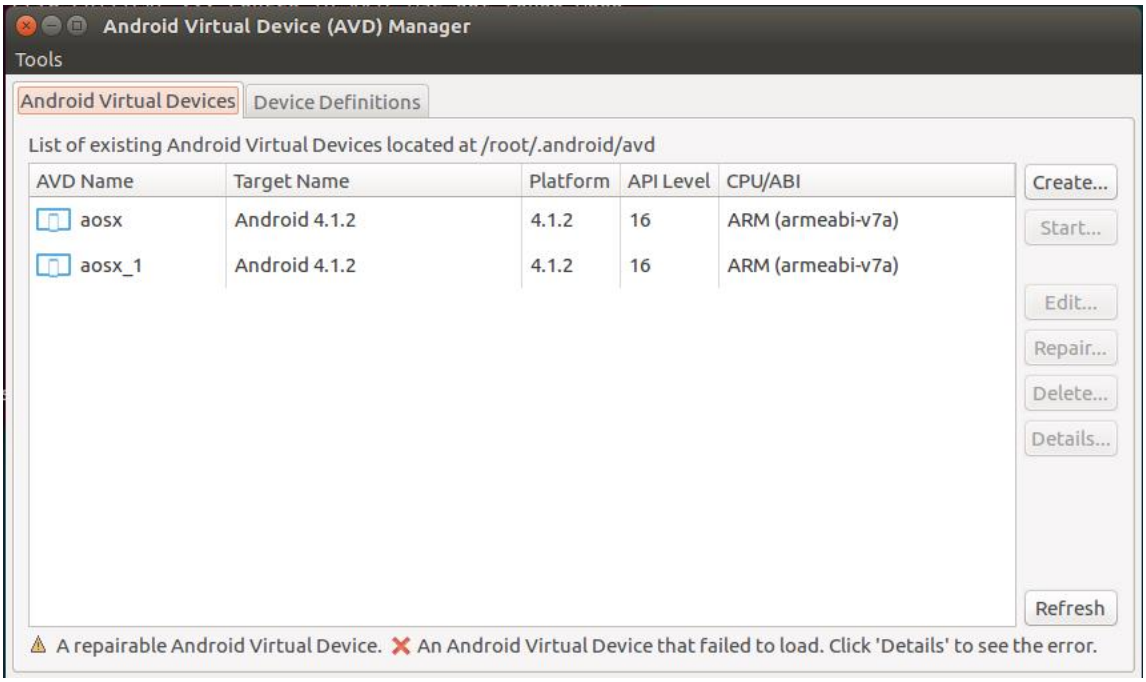


Gambar 6.13 Tampilan jendela Android SDK Manager



Gambar 6.14 API Android SDK Manager

Setelah semua tools dan API terinstall, selanjutnya adalah membuat emulator android. Jalankan *Android Virtual Device (AVD) Manager* dengan perintah "python android avd" pada direktori /tools. Jendela baru akan terbuka seperti pada Gambar 6.15.

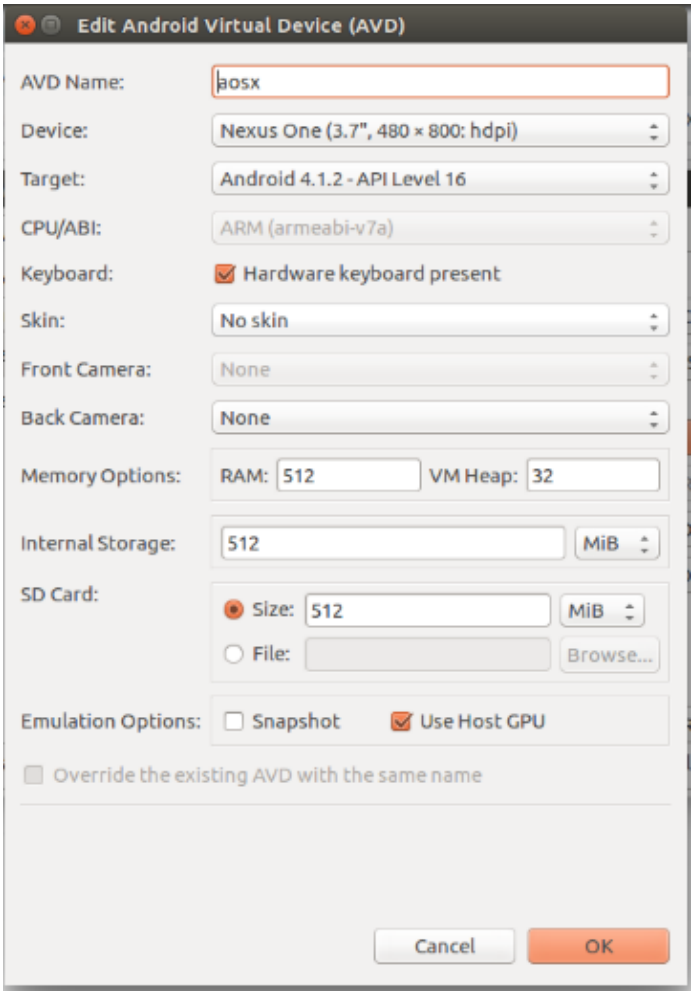


Gambar 6.15 Android Virtual Device (AVD) Manager

Untuk membuat emulator baru, jalankan "Create...". agar emulator dapat berjalan dengan baik, kita harus melakukan beberapa konfigurasi seperti pada Gambar 6.17.

```
AVD Name - aosx
Device - Nexus One
Target - android 4.1.2
Cpu/Abi - arm
Ram - 512mb
Vm Heap - 32
Internal Storage - 512mb
Sdcard size - 512 mib
Emulation options - use host GPU
```

Gambar 6.16 Konfigurasi emulator android

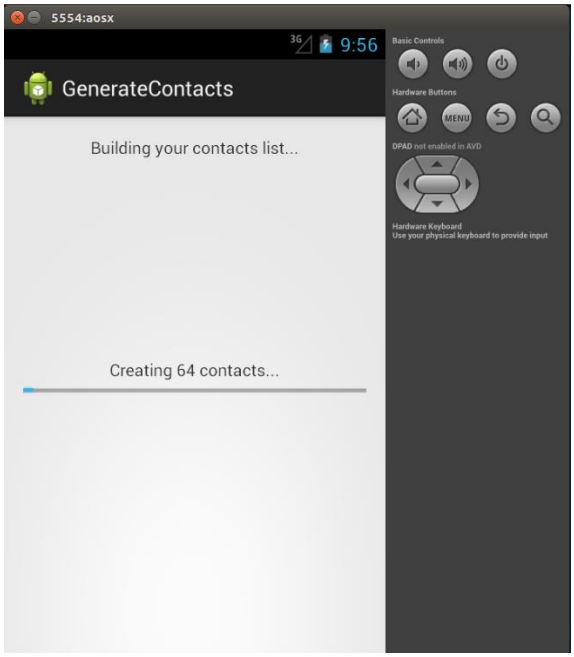


Gambar 6.17 Implementasi konfigurasi emulator android

Setelah emulator selesai dibuat, kita dapat menjalankan emulator dengan perintah "python emulator -avd aosx" pada terminal. Setelah emulator berhasil dijalankan, jalankan juga skrip *create_guest_avd.sh* yang terdapat pada direktori */Cuckoo/utils/ android_emulator_creator/*. Tujuan menjalankan skrip ini adalah menginstal beberapa komponen penting yaitu Superuser, Xposed, Droidmon, Emulator Anti-Detection, dan Generator Konten.

Setelah instalasi skrip *create_guest_avd.sh* berhasil, lakukan beberapa konfigurasi pada emulator android yang tengah berjalan. Konfigurasi yang dilakukan sebagai berikut:

a. Jalankan *Settings* – *Security* – *Screenlock* – Ubah menjadi *None*
b. Jalankan *Setting* – *Display* – *Sleep* – Ubah menjadi 30 Menit
c. Jalankan aplikasi Generate Contacts (Gambar 6.18)

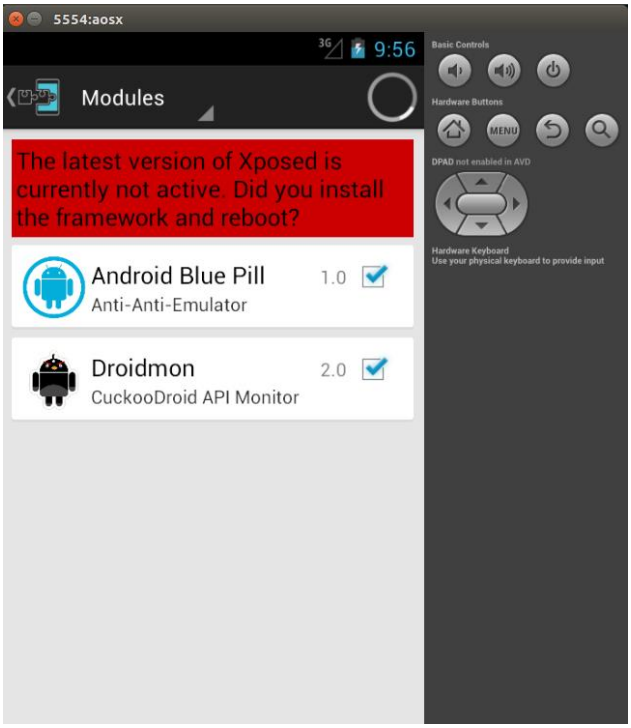


Gambar 6.18 Aplikasi Generate Contacts

d. Jalankan aplikasi Superuser

e. Jalankan aplikasi Xposed Installer

f. Pada bagian Module, centang kedua packages yaitu Droidmon dan Android Blue Pill (Gambar 6.19)



Gambar 6.19 Modul Xposed Installer

g. Klik *Framework - Install – Cancel – Soft Reboot*. Tunggu hingga proses *Soft Reboot* selesai.

Agar mesin emulator dapat melakukan tangkapan layar, mesin guest harus diinstall CuckooAgent.apk. Aplikasi ini dapat ditemukan pada direktori *Cuckoo/agent/android/ java_agent/*. Untuk menginstall CuckooAgent.apk, jalankan perintah pada Gambar 6.20 melalui *command-line* terminal:

```
$ cd /Cuckoo/agent/android/java_agent/
$ adb install CuckooAgent.apk
```

Gambar 6.20 Instalasi CuckooAgent.apk

Setelah rangkaian proses instalasi Cuckoo, Cuckoodroid dan Emulator Android selesai dilakukan, maka penguji dapat menjalankan Cuckoodroid melalui mesin host dengan perintah "python Cuckoo.py -d" pada direktori *Cuckoo/* . Sedangkan untuk menjalankan antarmukan web dari Cuckoodroid, penguji dapat menjalankan perintah "python manage.py runserver 8080" pada direktori /*Cuckoo/web_android/*.

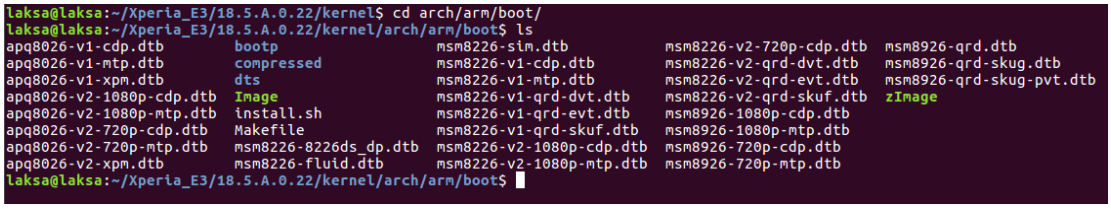
B. Proses Akuisisi RAM

Setelah seluruh kebutuhan telah siap, hal pertama yang dilakukan adalah mengkompilasi kernel dari *opensource code* yang telah dipersiapkan sebelumnya. *Opensource code* yang digunakan adalah kernel untuk Telepon Selular Sony Xperia E3 D2202 dengan tipe "18.5.A.0.22". Kode nama dari kernel untuk Xperia E3 ini adalah "arima_8226ds_dp_defconfig". Cara mengkompilasi kernel ini adalah dengan melakukan export arsitektur yang akan digunakan yaitu ARCH, kemudian melakukan export *cross-compiler* yang digunakan agar dikenali oleh mesin, kemudian jalankan perintah seperti pada Gambar 6.21.

```
laksa@laksa:~/Xperia_E3/18.5.A.0.22/kernel$ export ARCH=arm
laksa@laksa:~/Xperia_E3/18.5.A.0.22/kernel$ export CROSS_COMPILE=/home/laksa/arm-eabi-4.6/bin/arm-eabi-
laksa@laksa:~/Xperia_E3/18.5.A.0.22/kernel$ make arima_8226ds_dp_defconfig
```

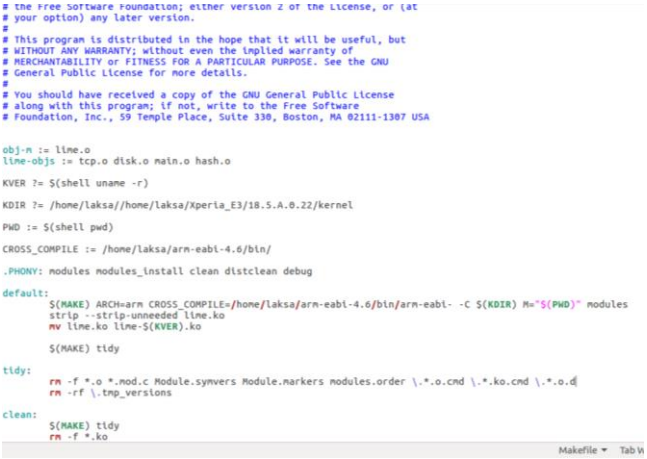
Gambar 6.21 Membangun Kernel

Hasil dari eksekusi perintah pada **Error! Reference source not found.** ini adalah file bernama "zImage". File zImage ini merupakan komponen utama dari *boot image* yang akan kita modifikasi. File zImage ini dapat ditemukan didalam direktori /*arch/arm/boot/* yang ditunjukkan oleh Gambar 6.22.



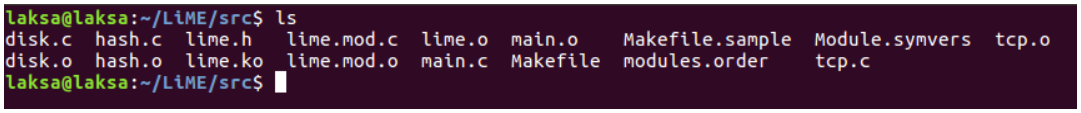
Gambar 6.22 Folder zImage

Tujuan dari memodifikasi *boot image* milik perangkat android target adalah agar perangkat android target dapat mengenali modul LiME yang akan kita buat dan kita install kedalam perangkat sebagai jembatan aktifitas akuisisi RAM dari perangkat android target ke komputer host. Langkah selanjutnya adalah membuat modul lime.ko. Pembuatan module lime ini dilakukan pada direktori LiME/src/ dengan cara menjalankan perintah make. Namun sebelum perintah make dijalankan, kita harus mengedit skrip pada file Makefile terlebih dahulu. Skrip Makefile yang telah diedit ditunjukkan pada Gambar 6.23.



Gambar 6.23 Skrip Makefile

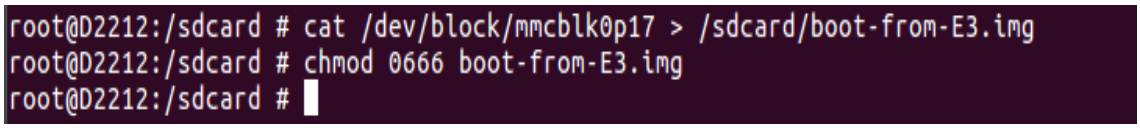
Setelah mengedit skrip Makefile, selanjutnya penguji menjalankan perintah make. Hasil dari perintah make ini adalah file bernama “lime.ko”. File ini merupakan modul yang akan kita instal pada perangkat nantinya. Folder tempat modul lime.ko berada terihat pada Gambar 6.24.



Gambar 6.24 Folder modul lime.ko

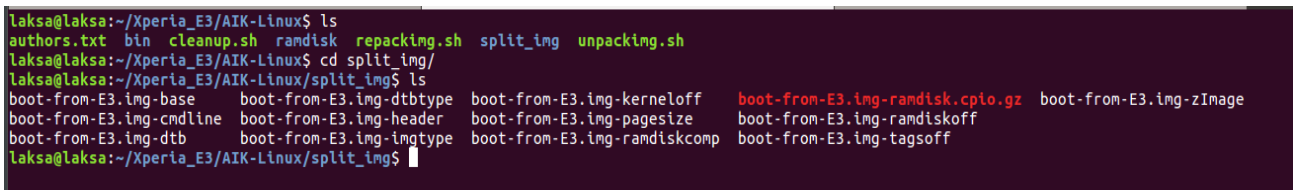
Agar perangkat android target mengenali modul lime.ko, penguji harus memodifikasi *boot image* milik perangkat android target dengan cara memasukkan file zImage yang telah

kita kompilasi sebelumnya kedalam *boot image* milik perangkat android target. Penguji mengambil boot.img dari perangkat android target untuk di bongkar terlebih dahulu. Tujuannya adalah untuk mengambil file bernama "ramdisk.cpio.gz" dari dalam perangkat. Cara mengambil boot.img dari dalam perangkat seperti pada Gambar 6.25.



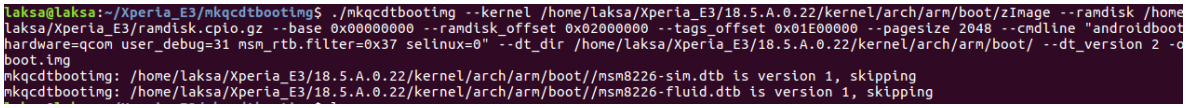
Gambar 6.25 Mengambill boot.img dari perangkat

Setelah berhasil mengambil boot.img dari perangkat, langkah selanjutnya adalah melakukan unpack atau membongkar ulang boot.img tersebut. Cara melakukan unpack adalah dengan tools Android Image Kitchen dengan menjalankan skrip unpackimg.sh. Hasil dari unpacking ini terlihat pada Gambar 6.26.



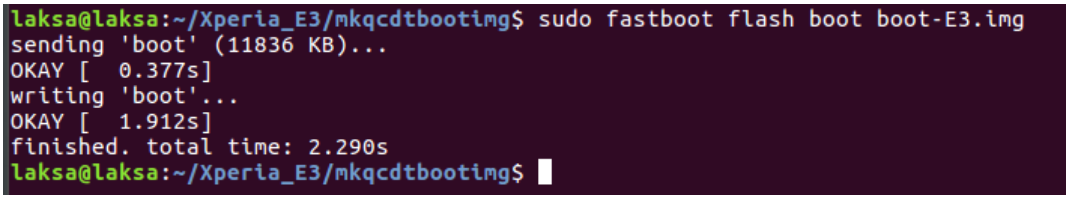
Gambar 6.26 Unpack boot.img

Selanjutnya, ramdisk akan digabungkan dengan file zImage dari kernel tadi menggunakan tools "mkqcdtbootimg" untuk membangun ulang boot.img. perintah untuk menjalankan tools mkqcdbootimg dapat dilihat pada Gambar 6.27.



Gambar 6.27 Membangun ulang boot.img

Setelah boot.img yang baru terbentuk, lakukan flashing kedalam perangkat melalui tools fastboot seperti pada Gambar 6.28.



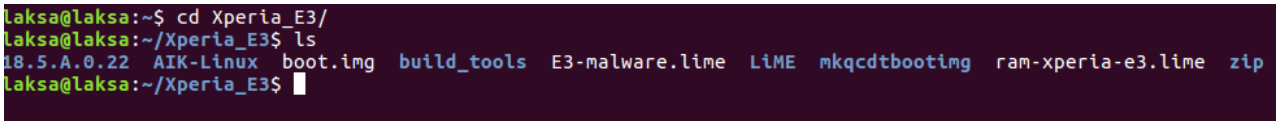
Gambar 6.28 Flash boot.img

Setalah boot.img yang baru telah terinstall, kita akan melakukan insert module lime.ko kedalam perangkat yang akan diakuisisi dengan menjalankan perintah pada Gambar 6.29:

```
$ sudo adb push lime.ko /sdcard/lime.ko
$ adb shell
$ su
$ insmod lime.ko ("path=tcp:4444 format=lime")
#terminal kedua
$ adb forward tcp:4444
$ nc 127.0.0.1 4444 > E3-malware.lime
```

Gambar 6.29 Insert modul dan Akuisisi RAM

Setelah proses akuisisi berhasil, akan ada file ram dengan ukuran 1GB yang berhasil diakuisisi bernama "E3-*malware*.lime" seperti pada Gambar 6.30.



Gambar 6.30 RAM hasil akuisisi