Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS 184853

AKUISISI DAN ANALISIS LIVE FORENSIC DARI DATA MEMORI VOLATILE PADA PERANGKAT MOBILE ANDROID

ACQUISITION AND ANALYSIS OF LIVE FORENSIC VOLATILE MEMORY DATA ON ANDROID MOBILE DEVICE

NEVADA VETERINO
NRP 05211640000096

Dosen Pembimbing :
Bekti Cahyo Hidayanto, S.Si., M.Kom.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS 184853

AKUISISI DAN ANALISIS LIVE FORENSIC DARI DATA MEMORI VOLATILE PADA PERANGKAT MOBILE ANDROID

NEVADA VETERINO
NRP 05211640000096

Dosen Pembimbing :
Bekti Cahyo Hidayanto, S.Si., M.Kom.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS 184853

ACQUISITION AND ANALYSIS OF LIVE FORENSIC VOLATILE MEMORY DATA ON ANDROID MOBILE DEVICE

NEVADA VETERINO
NRP 05211640000096

SUPERVISOR :
Bekti Cahyo Hidayanto, S.Si., M.Kom.

DEPARTMENT OF INFORMATION SYSTEMS
Faculty of Intelligence Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

AKUISISI DAN ANALISIS LIVE FORENSIC DARI DATA MEMORI VOLATILE PADA PERANGKAT MOBILE ANDROID

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer (S.Kom)
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas (ELECTICS)
Institut Teknologi Sepuluh Nopember

Oleh

Nevada Veterino
05211640000096

Surabaya, 14 Agustus 2020

Kepala Departemen Sistem Informasi

Dr. Mudjahidin, ST., MT.
NIP. 19701102003121001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

AKUISISI DAN ANALISIS LIVE FORENSIC DARI DATA MEMORI VOLATILE PADA PERANGKAT MOBILE ANDROID

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Oleh :

NEVADA VETERINO
NRP. 05211640000096

Disetujui Tim Penguji :Tanggal Ujian : 10 Juli 2020
Periode Wisuda : September 2020

Bekti Cahyo Hidayanto, S.Si., M.Kom.
(Pembimbing I)

Dr. Bambang Setiawan, S.Kom., M.T.
22 Jul 2020
(Penguji I)

Nisfu Asrul Sani, S.Kom., M.Sc.
(Penguji II)

Halaman ini sengaja dikosongkan

AKUISISI DAN ANALISIS LIVE FORENSIC DARI DATA MEMORI VOLATILE PADA PERANGKAT MOBILE ANDROID

Nama Mahasiswa : Nevada Veterino
NRP : 05211640000096
Jurusan : Sistem Informasi FTEIC-ITS
Pembimbing : Bekti Cahyo Hidayanto, S.Si., M.Kom.

ABSTRAK

Perkembangan teknologi informasi terutama perangkat mobile yang dampaknya diperoleh masyarakat mengakibatkan jumlah kepemilikan dan penggunaan smartphone di k,ehidupan sehari-hari meningkat. Di Indonesia sendiri fenomena ini sudah jelas dirasakan dampaknya di masyarakat, dengan jumlah smartphone melebihi jumlah penggunanya. Maraknya penggunaan smartphone dalam kehidupan sehari-hari menyebabkan kemungkinan sebuah kejahatan yang melibatkan smartphone akan selalu meningkat. Forensika perangkat mobile perlu dilakukan untuk mengambil informasi yang terdapat di memori perangkat supaya dapat dijadikan barang bukti yang sah di pengadilan. Pada umumnya forensika yang dilakukan pada perangkat mobile adalah forensika memori non-volatile, namun tidak semua informasi yang dibutuhkan ada. Pada kondisi ini diperlukan forensika terhadap memori volatile atau yang disebut live forensic. Metode yang digunakan dalam forensika memori volatile belum bisa menjadi andalan karena tidak ada cara pasti untuk melakukan hal ini. Tantangan forensika memori volatile yaitu sangat bergantung pada sistem operasi yang dipakai di perangkat tersebut. Dalam penelitian ini akan dilakukan percobaan forensika memori volatile pada perangkat mobile Android yang meliputi akuisisi data memori menggunakan metode memory dump dengan tools LiME (Linux Memory Extractor) dan AMExtractor (Android Memory Extractor) lalu analisis hasil

pengambilan memori menggunakan Volatility Framework dan ADB (Android Debug Bridge). Harapan hasil dari penelitian ini dapat menjadi rujukan untuk mengambil keputusan metode apa yang akan dipakai untuk pengambilan data secara live forensic dan saran tools yang akan digunakan.

Kata kunci: Live forensic, LiME, Volatility, ADB, Android, memori

ACQUISITION AND ANALYSIS OF LIVE FORENSIC VOLATILE MEMORY DATA ON ANDROID MOBILE DEVICE

Student Name : Nevada Veterino
NRP : 05211640000006
Department : Sistem Informasi FTEIC-ITS
Supervisor : Bekti Cahyo Hidayanto, S.Si., M.Kom.

ABSTRACT

The development of information technology, especially mobile devices whose impact is obtained by the society has resulted in an increase in the number of smartphone ownership and usage in daily life. In Indonesia, this phenomenon has clearly felt its impact on society, with the number of smartphones exceeding the number of users itself. The rise of smartphone usage in daily life will be causing the possibility of a crime involving a smartphone will always increase. As an evidence itself mobile device forensics needs to be done to retrieve information contained in the device's memory so that it can be used as legal evidence in court. In general, forensics performed on mobile devices are non-volatile memory forensics which examine the mobile device's storage, but not all the required information is available. In this condition the forensic of volatile memory or live forensic is needed to overcome what is impossible to do to the device's storage. The method used in volatile memory forensics cannot yet be a mainstay because there is no sure way to do this. The challenge of volatile memory forensics is that it really depends on the operating system used on the device. In this research a volatile memory forensics experiment will be conducted on an Android mobile device which includes memory data acquisition using the memory dump method with LiME (Linux Memory Extractor) and AMExtractor (Android Memory Extractor) tools and analysis of memory retrieval results using the Volatility Framework and ADB (Android Debug Brigde).

The hope of the results of this research is that it can be a reference for making decisions on what methods will be used for live forensic data collection and tool suggestions that will be used.
Key words: Live forensic, LiME, Volatility, ADB, Android, memory

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: Nevada Veterino
NRP	: 05211640000096
Tempat/Tanggal lahir	: Surabaya / 1 Agustus 1998
Fakultas/Departemen	: Fakultas Teknologi Elektro dan Informatika Cerdas / Sistem Informasi
Nomor Telp/Hp/email	: 082257818351 / nevadanero89@gmail.com

Dengan ini menyatakan dengan sesungguhnya bahwa penelitian/makalah/tugas akhir saya yang berjudul

AKUISISI DAN ANALISIS LIVE FORENSIC DARI DATA MEMORI VOLATILE PADA PERANGKAT MOBILE ANDROID

Bebas Dari Plagiarisme Dan Bukan Hasil Karya Orang Lain.

Apabila dikemudian hari ditemukan seluruh atau sebagian penelitian/makalah/tugas akhir tersebut terdapat indikasi plagiarisme, maka saya bersedia menerima sanksi sesuai peraturan dan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sesungguhnya dan untuk dipergunakan sebagaimana mestinya.

Surabaya , 10 Juli 2020

Nevada Veterino
NRP.05211640000096

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur penulis tuturkan ke hadirat Allah SWT, Tuhan Semesta Alam yang telah memberikan karunia dan hidayah-Nya kepada penulis sehingga penulis mendapatkan kelancaran dalam menyelesaikan tugas akhir dengan judul:

AKUISISI DAN ANALISIS LIVE FORENSIC DARI DATA MEMORI VOLATILE PADA PERANGKAT MOBILE ANDROID

Terima kasih penulis sampaikan kepada pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, dan bantuan baik berupa material maupun moril demi tercapainya tujuan pembuatan tugas akhir ini. Tugas akhir ini tidak akan pernah terwujud tanpa bantuan dan dukungan dari berbagai pihak yang sudah meluangkan waktu, tenaga dan pikirannya. Secara khusus penulis akan menyampaikan ucapan terima kasih yang sebanyak-banyaknya kepada :

1. Ibu Ernawati selaku orang tua serta Diandra Andromeda dan Bramada Affiandra selaku saudara kandung dari penulis yang selalu memberikan dukungan dan semangat.
2. Bapak Bekti Cahyo Hidayanto, S.Si., M.Kom., selaku dosen pembimbing dan sebagai narasumber yang senantiasa meluangkan waktu, memberikan ilmu dan petunjuk, serta memotivasi untuk kelancaran tugas akhir.
3. Bapak Dr. Bambang Setiawan, S.Kom., M.T. dan Bapak Nisfu Asrul Sani, S.Kom., M.Sc., selaku dosen penguji yang telah memberikan saran dan kritik untuk perbaikan tugas akhir.
4. Seluruh dosen Jurusan Sistem Informasi ITS yang telah memberikan ilmu yang bermanfaat kepada penulis.
5. Teman-teman Ave Rados yang telah memberikan bantuan, informasi, motivasi dan canda tawa sejak masa SMA hingga penulisan ini berlangsung.

6. Ludia Rosema Dewi yang telah membantu dan memberikan semangat kepada penulis setiap hari dalam penyelesaian tugas akhir.
7. Rekan-rekan Artemis yang telah memberikan banyak kenangan dan pengalaman selama kuliah
8. Berbagai pihak yang tidak bisa disebutkan satu persatu yang telah turut serta menyukseskan penulis dalam menyelesaikan tugas akhir.

Penyusunan laporan ini masih jauh dari kata sempurna sehingga penulis menerima adanya kritik maupun saran yang membangun untuk perbaikan di masa yang akan datang. Semoga buku tugas akhir ini dapat memberikan manfaat bagi pembaca.

Surabaya, 21 Juli 2020

(Penulis)

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

DAFTAR KODE

BAB I
PENDAHULUAN

Pada bagian ini akan diuraikan proses identifikasi masalah penelitian yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan relevansi terhadap pengerjaan tugas akhir. Berdasarkan uraian pada bagian ini, harapannya gambaran umum permasalahan dan pemecahan masalah pada tugas akhir ini dapat dipahami.

1.1. Latar Belakang

Perkembangan teknologi informasi saat ini sangat cepat. Masyarakat telah menggunakan teknologi informasi jauh lebih banyak dari satu dekade terakhir dalam kehidupan sehari-hari. Dalam melakukan kegiatan kita hampir tidak pernah terpisah dari penggunaan *smartphone*, yang seolah-olah sudah menjadi bagian dari hidup kita. Jumlah pengguna *smartphone* di seluruh dunia pada tahun 2018 mencapai 2,9 milyar, sedangkan di Indonesia sendiri pada tahun 2018 mencapai 111 juta dengan 80% *smartphone* memiliki OS Android [1] [2] [3]. Maraknya penggunaan *smartphone* dalam kehidupan sehari-hari menyebabkan kemungkinan sebuah kejahatan yang melibatkan *smartphone* akan selalu meningkat. Jika terjadi kasus kejahatan, pihak berwajib akan jauh lebih mungkin untuk menemukan tersangka dengan perangkat mobile daripada PC atau laptop sehingga angka permintaan untuk penyelidikan forensika perangkat *mobile* juga akan meningkat.

Forensika pada perangkat *mobile* berurusan dengan mengambil informasi yang bisa jadi merupakan barang bukti atau pendukung barang bukti dalam kasus kejahatan, yang informasi tersebut dapat ditemui di memori perangkat. Memori yang terdapat di perangkat mobile sama seperti dengan di komputer, yaitu memori non-volatile dan volatile. Penyelidikan forensika perangkat mobile pada umumnya dilakukan pada memori non-volatile, yaitu pada media penyimpanan internal

atau eksternal seperti SDCard, namun tidak semua informasi bisa didapatkan dari memori *non-volatile*. Memori *volatile* menyimpan informasi potensial tentang status sistem, aktivitas jaringan, *registry*, modul kernel yang dimuat, dan sebagainya. Melalui forensika memori *volatile* penyelidik dapat menghubungkan artefak potensial yang ditemukan dari analisis forensika konvensional seperti *disk*, *registry* sistem operasi, jaringan dan sistem file [4].

Pada penelitian sebelumnya yang dilakukan oleh Holger Macht pada tahun 2013 dengan judul "*Live Memory Forensics on Android with Volatility*" melakukan akuisisi memori menggunakan LiME (*Linux Memory Extractor*), tapi LiME sendiri memiliki kekurangan yaitu harus memuat modul ke dalam kernel untuk dijalankan. Untuk bisa memuat dan menjalankan modul di dalam kernel perangkat yang dituju, diperlukan *source code* kernel spesifik untuk suatu perangkat yang berasal dari pabrikan pembuat perangkat tersebut, dan yang menjadi masalah adalah sangat jarang pabrikan yang menyebarkan *source code* kernel mereka [5]. Penelitian lain yang dilakukan oleh Haiyu Yang dengan judul "*A Tool for Volatile Memory Acquisition from Android Devices*" dan Joseph T. Sylve dengan judul "*Android Memory Capture and Applications for Security and Privacy*" menggunakan tools open source yang dikembangkan sendiri [6] [7]. Mereka mencapai kesimpulan bahwa belum ada metode yang sempurna untuk melakukan akuisisi data memori ini, ditambah lagi dengan sangat banyaknya variasi perangkat mobile yang beredar di pasaran.

Oleh karena itu pada penelitian ini akan dilakukan percobaan untuk melakukan akuisisi data memori menggunakan metode dan *tools* yang sudah ada untuk dilakukan pada perangkat mobile yang sudah tersedia dalam skenario yang direncanakan sebagai barang bukti dari sebuah kasus kejahatan. Luaran yang diharapkan dari penelitian ini yaitu mengetahui apakah metode tertentu dalam melakukan forensika memori

dapat dilakukan pada suatu perangkat yang tidak diketahui probabilitas berhasilnya dilakukan forensika, dan memberi wawasan metode apa yang harus digunakan untuk menangani forensika perangkat mobile tertentu.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan sebelumnya, maka rumusan masalah pada tugas akhir ini adalah sebagai berikut:

1. Bagaimana melakukan akuisisi data dari memori *volatile* perangkat *mobile* Android?
2. Bagaimana melakukan analisis dari hasil akuisisi data memori *volatile*?
3. Apakah terjadi perubahan memori *volatile* pada saat proses akuisisi data yang diinginkan?

1.3. Batasan Permasalahan

Berdasarkan permasalahan yang telah diuraikan sebelumnya, adapun batasan masalah dalam tugas akhir ini adalah sebagai berikut:

1. Perangkat *mobile* yang digunakan dalam percobaan ini adalah Samsung Galaxy S3 yang memiliki sistem operasi Android versi 4.1.2 Jelly Bean.
2. *Tools* yang digunakan untuk akuisisi dan analisis memori adalah *tools* berbasis *open source*.
3. Aktivitas yang akan dianalisis pada bagian skenario adalah penggunaan aplikasi kamera dan aplikasi media sosial dan komunikasi yaitu WhatsApp.Proses penyelidikan dilakukan menggunakan *tools open source* dan terbatas pada kemampuan *tools* tersebut karena tidak terdapat akses ke *hardware* khusus penyelidikan forensik.
4. Proses penyelidikan dilakukan menggunakan *tools open source* dan terbatas pada kemampuan *tools* tersebut karena tidak terdapat akses ke *hardware* khusus penyelidikan forensik.

1.4. Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah yang telah disebutkan sebelumnya, adapun tujuan dari penelitian tugas akhir ini adalah sebagai berikut:

1. Mengetahui metode apa yang akan berhasil untuk melakukan akuisisi data memori *volatile* dari perangkat *mobile* Android.
2. Mengetahui metode apa yang akan berhasil untuk melakukan analisis data memori *volatile* dari perangkat *mobile* Android.
3. Mengetahui perubahan memori *volatile* pada saat proses akuisisi data yang diinginkan

1.5. Manfaat Penelitian

Manfaat yang diharapkan dapat diperoleh dari penelitian tugas akhir ini adalah memudahkan penyelidik forensika untuk melakukan investigasi kasus kejahatan yang melibatkan *smartphone* Android sebagai barang bukti yang disita, dan membantu pengambilan keputusan metode apa yang harus digunakan untuk akuisisi dan analisa memori *volatile* pada perangkat *mobile* Android.

1.6. Relevansi

Tugas akhir ini disusun untuk memenuhi salah satu syarat kelulusan sebagai Sarjana Komputer di Departemen Sistem Informasi Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC) Institut Teknologi Sepuluh Nopember (ITS). Tugas akhir ini sesuai dengan *roadmap* isu strategis dari laboratorium Infrastruktur dan Keamanan Teknologi Informasi (IKTI) yaitu Infrastruktur Teknologi Informasi, dan mengarah pada Pengembangan Implementasi Forensika Digital. Selain itu tugas akhir ini sesuai dengan penerapan mata kuliah Forensika Digital. Penelitian ini melakukan akuisisi dan analisis memori *volatile* pada perangkat *mobile* Android supaya bisa didapatkan barang bukti yang sah di pengadilan.

BAB II
TINJAUAN PUSTAKA

Pada bagian ini akan menjelaskan mengenai studi literatur terhadap penelitian sebelumnya dan dasar teori yang dijadikan acuan atau landasan dalam pengerjaan tugas akhir ini.

2.1 Penelitian Sebelumnya

Dalam proses pengerjaan penelitian tugas akhir ini, dilakukan pencarian penelitian-penelitian yang sudah dilakukan untuk dijadikan sebagai referensi dalam pengerjaan penelitian tugas akhir ini.

Tabel 2.1 Studi Sebelumnya

1. Live Memory Forensics on Android with Volatility [5]
Penulis; Tahun Holger Macht; 2013
Pembahasan Penelitian ini memberikan gambaran umum tentang *live memory forensic* pada perangkat Android dengan menyediakan tumpukan perangkat lunak yang memberi kemudahan penyelidik forensika untuk melakukan analisis aplikasi. Setelah mendapatkan data yang diperlukan menggunakan LiME akan dilakukan analisis dari tiga sudut pandang yaitu Linux Kernel, Android DalvikVM, dan Android Application.
Keterkaitan Penelitian tugas akhir yang dilakukan berkaitan dengan akuisisi memori menggunakan LiME dan analisis memori menggunakan Volatility.

2. Practical Infeasibility of Android Smartphone Live Forensics Applicability Constraints of LiME and Volatility [8]
Penulis; Tahun Philipp Wächter; 2015
Pembahasan Penelitian ini mencoba melakukan perbandingan untuk mengakuisisi dan menganalisa memori *volatile* pada berbagai perangkat dengan *tools* yang digunakan yaitu LiME dan Volatility. Dengan perbandingan tersebut dan mencari batasan apa saja yang mempengaruhi keberhasilan percobaan dapat disimpulkan kepraktisan dalam melakukan *live forensic* pada memori *volatile* perangkat Android.
Keterkaitan Penelitian tugas akhir yang dilakukan berkaitan dengan akuisisi memori menggunakan LiME dan analisis memori menggunakan Volatility.
3. Android Memory Capture and Applications for Security and Privacy
Penulis; Tahun Joseph T. Sylve; 2011
Pembahasan Penelitian ini membuat sebuah *tools* bernama DMD (Droid Memory Dumper) yang memungkinkan melakukan *live forensic* pada perangkat Android dengan mengakuisisi memori *volatile* dan melakukan analisis dari hasil akuisisi tersebut.

Keterkaitan Penelitian tugas akhir ini berkaitan dengan *tools* yang digunakan yaitu DMD yang merupakan versi lama dari LiME.
4. Discovering Authentication Credentials in Volatile Memory of Android Mobile Devices
Penulis; Tahun Dimitris Apostolopoulos, Giannis Marinakis, Christoforos Ntantogian, Christos Xenakis; 2013
Pembahasan Penelitian ini melakukan *dump* pada memori perangkat Android dengan mengggunakan *tool Open Source* Dalvik Debug Monitor Server (DDMS) yang membolehkan untuk memeriksa proses yang sedang berjalan. Aplikasi dikategorikan menjadi empat berdasarkan kategorinya, yaitu aplikasi *m-banking*, aplikasi *e-shopping*/finansial, aplikasi jejaring sosial dan komunikasi, dan aplikasi pengelola *password* dan enkripsi.
Keterkaitan Penelitian tugas akhir ini berkaitan dengan pemeriksaan aplikasi yang sedang berjalan yang empat kategori aplikasi tersebut dapat digunakan oleh pelaku kejahatan dalam melakukan tindak kriminal, di mana terdapat kemungkinan besar dapat ditemukan barang bukti di aplikasi tersebut.
5. A Tool for Volatile Memory Acquisition from Android Devices
Penulis; Tahun Haiyu Yang, Jianwei Zhuge, Huiming Liu, Wei Liu; 2016
Pembahasan Penelitian ini mengembangkan metode baru untuk melakukan akuisisi memori dengan menggunakan AMExtractor (Android Memory Extractor) yang

memanfaatkan /dev/kmem untuk mengeksekusi kode di kernel perangkat, setelah itu diekspor ke Volatility Framework untuk analisis lebih lanjut
Keterkaitan Penelitian tugas akhir ini berkaitan dengan akuisisi memori *volatile* yang akan dianalisis menggunakan Volatility

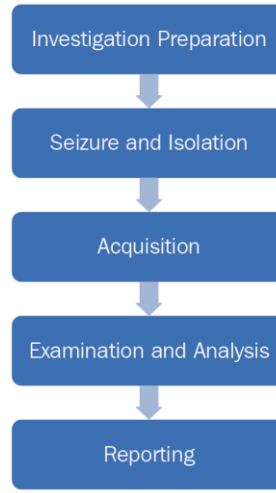
2.2 Dasar Teori

Dasar teori berisi teori-teori yang digunakan dalam pengerjaan penelitian tugas akhir. Dalam dasar teori, acuan yang digunakan adalah berdasarkan penelitian dan buku.

2.2.1 Forensika Digital

Menurut Dr. H. B. Wolfe (2008), forensika digital adalah serangkaian metode dari teknik dan prosedur untuk mengumpulkan barang bukti, dari peralatan komputer dan berbagai perangkat penyimpanan dan media digital, yang dapat disajikan di pengadilan dalam format yang koheren dan bermakna. Langkah dalam melaksanakan penyelidikan forensika dari buku yang ditulis oleh Oleg Skulkin (2008) [9] adalah pada Gambar 2.1.

Penyelidikan akan dimulai ketika terdapat permintaan untuk pemeriksaan, biasanya meliputi persiapan dari berkas-berkas dan formulir untuk mendokumentasikan *chain of custody*, informasi kepemilikan, informasi yang dicari oleh pemohon, dan seterusnya. Langkah selanjutnya yaitu penyitaan dan pemisahan barang bukti, di mana penanganan barang bukti saat proses penyitaan adalah salah satu langkah penting saat melaksanakan analisis forensika. Setelah itu adalah tahap akuisisi yang mengacu pada ekstraksi data dari perangkat. Metode ekstraksi data dapat dikelompokkan menjadi metode menggunakan perangkat keras dan menggunakan perangkat lunak [7].



Gambar 2.1. Tahapan Forensik

Lalu pada tahap pemeriksaan dan analisis berbagai *tools* digunakan karena tidak ada satu *tools* yang dapat digunakan dalam semua kasus. Tahap terakhir adalah pelaporan, namun dokumentasi pemeriksaan harus dilakukan selama proses berlangsung dengan mencatat apa saja yang dilakukan pada setiap fase. Data yang sudah diambil harus dipresentasikan dengan jelas pada pendengar supaya arti dari sebuah data tersampaikan. Penting bagi penyelidik untuk memahami kebutuhan dari model langkah penyelidikan forensika digital yang bertujuan untuk memperbaiki prosedur yang diikuti dalam bidang ini [10].

2.2.2 Forensika Perangkat *Mobile*

Forensika perangkat *mobile* adalah cabang forensika digital yang berkaitan dengan mengekstraksi, memulihkan, dan menganalisis bukti atau data digital dari perangkat *mobile* dalam kondisi yang memenuhi metode forensika. Hal ini berkaitan dengan mengakses data yang disimpan pada perangkat, yang meliputi SMS, kontak, catatan panggilan, foto, video, dokumen, *file* aplikasi, riwayat penelusuran, dan sebagainya, dan

memulihkan data yang dihapus dari perangkat menggunakan berbagai teknik forensika. Penting bahwa proses memulihkan atau mengakses data dari perangkat harus memenuhi metode forensika jika barang bukti diterima di pengadilan dan untuk menjaga integritas barang bukti. Jika barang bukti itu diterima di pengadilan maka penting untuk hanya bekerja pada *image file* dari perangkat tersebut dan bukan pada perangkat asli itu sendiri [9].

2.2.3 Live Forensic

Live forensic adalah salah satu bagian dari forensika komputer yang merupakan cabang ilmu forensika digital yang berkaitan dengan barang bukti yang ditemukan pada sebuah perangkat komputer. *Live forensic* mengikuti tujuan dari forensika digital tetapi hanya berfokus pada sistem komputer yang sedang menyala. Tujuan utamanya adalah untuk memperoleh data *volatile* yang jika tidak akan hilang ketika sistem komputer dimatikan atau akan ditimpa jika sistem komputer tetap dihidupkan untuk periode yang lebih lama [11].

2.2.4 Forensika Memori

Forensika memori adalah sebuah teknik penyelidikan forensika yang digunakan untuk mengekstrak artefak potensial dari memori *volatile* sebuah sistem. Pada peralatan komputer RAM (*Random Access Memory*) digunakan sebagai memori *volatile*. Forensika memori meliputi dua tahap, yaitu penangkapan memori dan analisis terhadap hasil memori yang ditangkap. Penangkapan RAM adalah proses pembuatan *image* dari memori fisik dan menyimpannya sebagai *file* pada media penyimpanan eksternal. Analisis memori melibatkan penguraian struktur pohon data dari *file* memori yang diambil, mencari proses yang berjalan ketika memori diambil serta data lain seperti kata sandi, *file* yang diunduh, Sertifikat SSL, URL, dll [4].

2.2.5 Barang Bukti Digital

Barang bukti digital didefinisikan sebagai informasi apa pun yang bernilai nilai yang disimpan atau dikirim dalam bentuk digital [12]. Informasi digital dapat dikumpulkan saat memeriksa media penyimpanan digital, memantau lalu lintas jaringan, atau membuat salinan duplikat dari data digital yang ditemukan selama penyelidikan forensika. Barang bukti digital harus mempunyai beberapa karakteristik untuk bisa dibawa ke pengadilan, yaitu:

- Jelas dan dapat dipahami oleh hakim.
- Tidak boleh ada keraguan terhadap keaslian atau kebenarannya.
- Membuktikan pelaku bersalah atau tidak bersalah.
- Terkait dengan fakta yang akan dibuktikan.
- Nyata dan terkait dengan kejadian kasus dengan cara yang benar.

Adanya indikasi terdapat barang bukti digital pada sebuah memori *volatile* bergantung pada aplikasi apa saja yang terdapat pada perangkat, karena setiap aplikasi bisa saja memiliki petunjuk yang disimpan pada memori *volatile*. Jejak yang ditinggalkan oleh aplikasi tersebut berada pada DalvikVM milik setiap proses yang sedang berjalan. Pada *virtual machine* tersebut memiliki *object* bernama DvmGlobals yang memiliki isi seperti daftar dari semua *system class* yang dimuat dan informasi spesifik dari sebuah *class* seperti *fields*, *variable*, dan *method.*

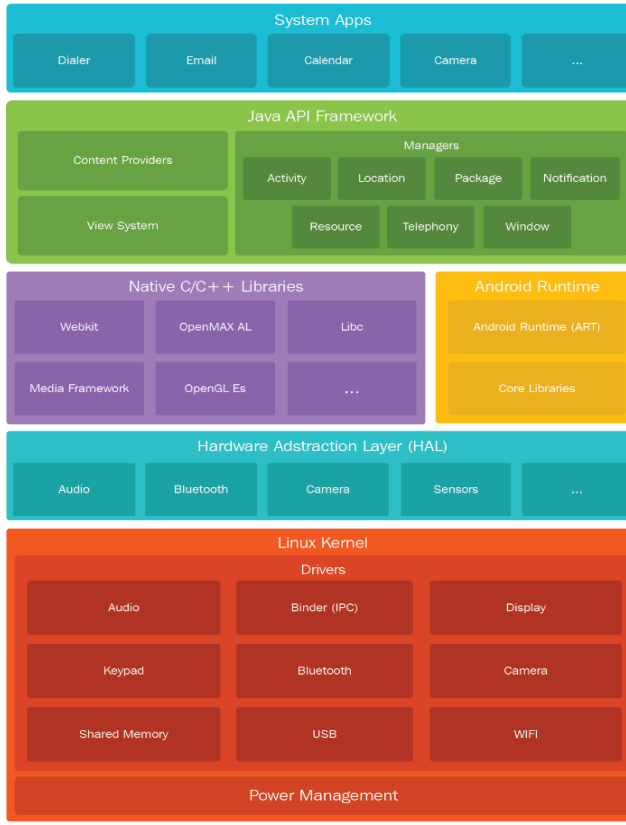
2.2.6 Android

Android adalah platform perangkat lunak dan sistem operasi untuk perangkat seluler, berdasarkan kernel Linux, dan dikembangkan oleh Google dan kemudian Open Handset Alliance. Ini memungkinkan pengembang untuk menulis kode terkelola dalam bahasa Java, mengendalikan perangkat melalui Java *library* yang dikembangkan Google. Android tersedia sebagai *Open Source*. Android adalah tumpukan perangkat lunak *Open Source* yang dapat diunduh secara bebas untuk

perangkat seluler yang mencakup sistem operasi, *middleware*, dan aplikasi utama berbasis Linux dan Java. [13].

2.2.7 Kernel Linux

Kernel Linux adalah bagian yang sangat penting dari perangkat lunak di hampir setiap perangkat Android [14]. Kernel Linux dasar diatur langsung di sekitar *services* utama yang disediakannya, yaitu penanganan proses, manajemen memori, manajemen sistem *file*, akses jaringan dan *driver* untuk perangkat keras. Area-area ini masing-masing sesuai dengan direktori kernel, *mm*, *fs*, *net* dan *driver*. Selain itu, area-area ini sering dibagi lagi menjadi layanan-layanan khusus [15].



Gambar 2.2. Arsitektur Android

2.2.8 Linux Memory Extractor

Linux Memory Extractor merupakan kepanjangan dari LiME yang sebelumnya dikenal sebagai DMD (Droid Memory Dumper) adalah sebuah Loadable Kernel Module (LKM) yang memungkinkan akuisisi memori *volatile* dari Linux dan perangkat berbasis Linux, termasuk yang berbasis Android. LiME dapat melakukan akuisisi memori dan menyimpannya baik ke sistem perangkat atau melalui jaringan. Hal ini membuat LiME unik karena merupakan *tools* pertama yang memungkinkan untuk mengakuisisi memori secara penuh pada perangkat Android. Hal ini juga meminimalkan interaksi antara ruang proses *user* dan kernel selama akuisisi, maka memungkinkan untuk menghasilkan tangkapan memori yang lebih baik secara forensik daripada *tools* lain yang dirancang untuk akuisisi memori Linux.

2.2.9 Android Memory Extractor

Android Memory Extractor merupakan kepanjangan dari AMExtractor adalah sebuah *tools* untuk melakukan *dumping* memori *volatile* dari sebuah perangkat *mobile* Android tanpa membutuhkan *source code* kernel. LiME dan *tools* serupa yang menggunakan LKM tidak dapat digunakan tanpa adanya *source code* kernel atau dukungan LKM dari perangkat, daripada menggunakan LKM, AMExtractor menggunakan */dev/kmem* untuk menjalankan kode pada kernel.

2.2.10 Volatility Framework

Volatility Framework adalah kumpulan *tools* yang sepenuhnya terbuka menggunakan implementasi Python di bawah GNU General Public License untuk melakukan ektraksi artefak digital dari sampel memori *volatile* (RAM). Teknik ektrasi yang dilakukan sepenuhnya tidak berkaitan dengan sistem yang sedang diselidiki tetapi menawarkan transparansi *runtime* dari sistem. *Framework* ini bertujuan untuk memperkenalkan orang-orang dengan teknik dan kompleksitas yang terkait dengan penggalian artefak digital dari sampel

memori *volatile* dan menyediakan *platform* untuk pekerjaan lebih lanjut.

2.2.11 Android Debug Bridge

Android Debug Bridge (ADB) adalah *command-line tool* serbaguna yang memungkinkan komputer berkomunikasi dengan perangkat Android. Perintah ADB dapat melakukan berbagai tindakan kepada perangkat seperti menginstal, melakukan *debugging*, dan memberikan akses *shell* untuk menjalankan berbagai perintah pada perangkat.

BAB III
METODOLOGI PENELITIAN

Pada bagian metodologi akan menjelaskan bagaimana langkah yang akan dilakukan dalam pengerjaan tugas akhir ini yaitu studi literatur, pembuatan skenario kejahatan, penyelidikan forensika, dan dokumentasi tugas akhir. Penjelasan proses tahapan pelaksanaan tugas akhir ini adalah sebagai berikut:

Gambar 3.1. Alur pengerjaan tugas akhir

3.1 Studi Literatur

Tahap pertama pengerjaan tugas akhir yaitu studi literatur. Pada tahap ini dilakukan pengumpulan dan pengkajian literatur yang memiliki keterkaitan dengan topik tugas akhir yang dikerjakan. Hasil dari studi literatur adalah konsep, acuan, *tools*, dan metode yang akan digunakan untuk menyelesaikan tugas akhir. Selain itu tahap ini juga mencari informasi yang

dapat menjadi referensi macam-macam tindak kejahatan, tingkah, dan perilaku yang dapat digunakan dalam tahap pembuatan skenario.

3.2 Pembuatan Skenario

Pada tahap ini dilakukan pembuatan skenario kejahatan yang selanjutnya akan dilakukan pada perangkat *mobile*. Skenario ini perlu dilakukan untuk menjadi pengganti barang bukti karena tidak mungkin untuk mendapatkan barang bukti asli dari pihak kepolisian. Tindakan yang akan dilakukan menggunakan perangkat *mobile* ini diharapkan dapat menyimulasikan apa yang dilakukan pelaku kejahatan dalam menggunakan perangkat *mobile*-nya selama pelaku melakukan tindak kejahatan yang sebenarnya, maupun aktivitas sehari-hari yang sekiranya berhubungan dengan tindak kejahatan tersebut.

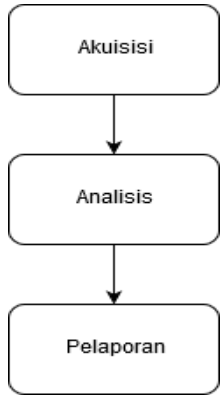
Penggunaan perangkat *mobile* dalam skenario akan berupa pembuatan *file* pada perangkat, seperti memfoto gambar, merekam video, atau merekam suara. Selain itu juga melibatkan aktivitas jejaring sosial, seperti berkomunikasi menggunakan WhatsApp, komunikasi ini juga dapat melibatkan mengirim *file* yang tadi dibuat. Jika pada skenario yang dibuat ternyata perangkat yang ditemukan belum dilakukan *rooting*, maka terdapat prosedur untuk tetap dapat melakukan akuisisi memori *volatile*.

Prosedur yang diikuti mirip dengna kasus jika barang bukti harddisk tidak dapat dibaca datanya, dengan asumsi terjadi masalah pada *port* konektor pada harddisk tersebut, maka harddisk perlu dibongkar untuk diambil piringan magnetiknya lalu dipindahkan ke tempat lain yaitu *casing* harddisk yang identik. Setelah diambil data yang diperlukan piringan tersebut dikembalikan lagi ke *casing* semula dan dirakit kembali supaya menjaga integritas barang bukti. Pada kasus perangkat Android yang tidak dilakukan *rooting* maka prosedurnya yaitu melakukan proses *rooting* sementara yang tidak melibatkan *reboot* pada perangkat, lalu setelah dilakukan akuisisi memori *volatile* dilakukan proses *unroot* untuk mengembalikan kondisi

barang bukti seperti semula sehingga barang bukti dapat dianggap sah. Proses *rooting* dan *unrooting* ini tidak merubah atau menghapus memori *volatile* selama metode yang digunakan sudah dipastikan berhasil dan tidak melibatkan *reboot* pada perangkat.

3.3 Penyelidikan Forensik

Pada tahap ini dilakukan penyelidikan forensik dengan perangkat *mobile* Android, namun dalam penelitian ini tidak meliputi tahapan persiapan penyelidikan dan penyitaan barang bukti seperti yang telah disebutkan pada dasar teori karena asumsi barang bukti sudah tersedia dan siap untuk dilakukan forensik. Penjelasan proses penyelidikan forensika dalam tugas akhir ini adalah sebagai berikut:



Gambar 3.2. Tahapan forensik

3.3.1 Akuisisi

Pada tahap ini dilakukan akuisisi data memori *volatile* yaitu RAM dari perangkat *mobile* yang menjadi barang bukti. Untuk bisa mendapatkan data dari RAM digunakan *tools open source* yang berfungsi untuk melakukan *memory dump*. Proses *memory dump* ini yaitu menyalin data dari memori dengan mengeksekusi kode pada kernel perangkat, lalu dampak dari pengeksekusian kode itu akan menghasilkan *file image* salinan memori, setelah itu *file image* hasil dari *dump* akan dipindahkan ke komputer.

3.3.2 Analisis

Pada tahap ini dilakukan analisis dari data memori yang sudah didapatkan dari tahapan akuisisi. *File image* yang sudah didapatkan dilakukan analisis lebih lanjut supaya data tersebut dapat diambil informasi yang berarti supaya bisa menjadi barang bukti kasus kejahatan. Tahapan analisis ini akan menggunakan *tools open source* untuk membaca *file image* dari hasil akuisisi. Pada tahapan analisis *image* ini yaitu mengetahui apa saja yang dilakukan oleh pengguna pada perangkatnya, maka data yang dicari berupa hal yang berhubungan dengan aktifitas, contohnya adalah *process list* yang berisi *process* yang sedang berjalan saat dilakukan akuisisi memori *volatile* maupun *process tree* yang menunjukkan hubungan antara *process*, selain itu *connection history* juga bisa digunakan, yang isinya adalah daftar IP mana saja yang dituju oleh perangkat, dan juga bisa saja menggunakan ARP *Table*. Harapan dari hasil analisis ini dapat mengungkap informasi kejadian atau aktivitas yang dilakukan oleh pelaku kejahatan seperti riwayat penggunaan aplikasi, log penggunaan perangkat, aktivitas jaringan, *file* yang berada di memori, informasi dari Dalvik VM, dsb.

3.3.3 Pelaporan

Pada tahap ini dilakukan pelaporan dari semua temuan yang didapatkan selama proses tahapan akuisisi dan analisis dan membuat sebuah dokumentasi penyelidikan. Data dan informasi yang menjadi isi dari dokumen adalah:

- Tanggal dan waktu pemeriksaan dimulai.
- Kondisi fisik dari perangkat.
- Keadaan perangkat saat diterima.
- Pembuat, model, dan sistem operasi dari perangkat.
- Foto dari perangkat dan komponen individu lainnya.
- *Tools* apa saja yang digunakan dalam penyelidikan.
- Data yang diperoleh dari proses pemeriksaan.

3.4 Dokumen Tugas Akhir

Dokumen tugas akhir merupakan tahap akhir dari pengerjaan tugas akhir dari keseluruhan tahapan penelitian yang telah dilakukan sesuai metodologi. Seluruh temuan dan hasil dari penelitian dituliskan dalam dokumen ini, serta kesimpulan dan saran untuk penelitian kedepannya.

Halaman ini sengaja dikosongkan

BAB IV
PERANCANGAN

Pada bab ini akan dijelaskan mengenai rancangan dari langkah-langkah penelitian tugas akhir ini. Perancangan yang dibuat berupa persiapan workstation, persiapan perangkat *mobile*, dan perisapan beberapa metode yang akan digunakan untuk proses akuisisi dan analisis.

4.1 Perisapan Workstation

Dalam pengerjaan penelitian ini dibutuhkan komputer *workstation* untuk melakukan persiapan, proses akuisisi dan analisis hasil. Selain itu juga dibutuhkan VMware untuk keperluan virtualisasi yang dijalankan pada *workstation*. Rincian dari *workstation* yang digunakan sebagai berikut di tabel 4.1 dan tabel 4.2.

Tabel 4.1. Spesifikasi Workstation

Nama Perangkat	Laptop MSI GL62M 7RDX
Processor	Intel® Core™ i7-7700HQ @ 2.8 GHz
Memory	16.00 GB DDR4 2133 MHz
GPU	Nvidia GeForce GTX 1050 2 GB GDDR5
Storage	Hitachi 2.5’ 1 TB HDD 5400 RPM + ADATA SX8200 240 GB NVMe SSD
Sistem Operasi	Windows 10 Education 1909 64-bit

Tabel 4.2. Spesifikasi *Virtual Machine*

Perangkat Lunak	VMware Workstation 15
Processor	4 Core
Memory	4 GB
Storage	50 GB HDD VMDK
Sistem Operasi	Linux Ubuntu 18.04.4 64-bit

4.2 Persiapan Perangkat *Mobile*

Perangkat *Mobile* yang digunakan dalam penelitian ini yaitu *handphone* Samsung Galaxy S3 dengan deskripsi perangkat dapat ditemui pada *Setting>About device*:

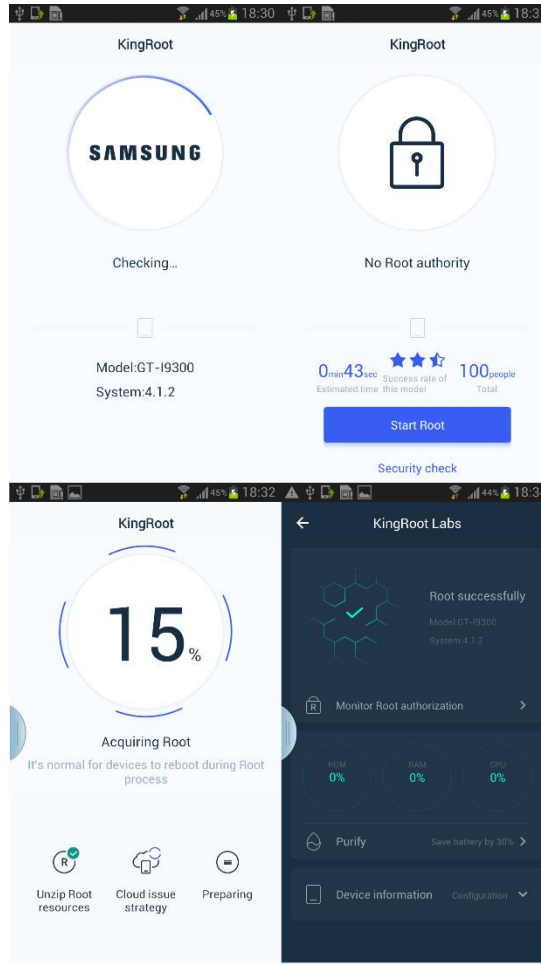
- Model number: GT-I9300
- Android version: 4.1.2
- Baseband version: I9300XXEMG4
- Kernel version: 3.0.31-1314436
- Build number: JZO54K.I9300XXEMG4

Gambar 4.1. Samsung Galaxy S3

Untuk dapat melakukan tindakan forensik pada perangkat harus membuka menu *Developer Options* dan menyalakan fitur USB *Debugging*. Langkah yang dilakukan yaitu pergi ke *Setting>About device* dan menekan pilihan *Build number* sebanyak tujuh kali, lalu menu *Developer Options* akan terbuka dan aktikfkan fitur USB *Debugging*.

Selanjutnya dilakukan proses *rooting* untuk mendapatkan akses *root* menggunakan aplikasi KingRoot, hal ini perlu dilakukan karena akses *user* terhadap *hardware* dan sistem sangat terbatas dan hanya akses *root* yang dapat melakukan interaksi lebih jauh. Metode *rooting* yang dipilih menggunakan aplikasi KingRoot karena dengan metode ini dapat memperoleh hak akses *root* tanpa melakukan *restart* pada

perangkat yang mengakibatkan kehilangan data dari memori *volatile*. Aplikasi KingRoot dapat diunduh pada *kingroot.en.uptodown.com/android* dan dipasang pada perangkat dengan menyalakan fitur membolehkan instalasi aplikasi dengan sumber selain dari Play Store pada *Setting>Security>Unknown sources*. Setelah aplikasi terpasang jalankan perintah "Try Root" untuk mendapatkan akses *root*.



Gambar 4.2. Langkah *rooting* menggunakan KingRoot

Setelah menyalakan fitur USB *Debugging* dan mendapatkan hak akses *root* dilakukan skenario penggunaan perangkat yang akan dibuktikan melalui data memori *volatile*. Penggunaan perangkat yaitu tersambung ke jaringan, membuka aplikasi kamera lalu mengambil gambar, dan membuka aplikasi WhatsApp.

4.3 Persiapan Linux Memory Extractor

Pada tahap ini dilakukan persiapan untuk dapat mengimplementasikan *tools* Linux Memory Extractor (LiME) dengan sukses. Beberapa hal yang dilakukan dalam persiapan yaitu menyiapkan *environment*, *toolchain*, dan *cross compile* dengan kernel.

4.3.1 Persiapan Environment dan Toolchain

Pada tahap ini dilakukan persiapan *environment* untuk membuat sebuah modul yang akan dijalankan pada sistem target. Beberapa hal yang diperlukan dalam tahap ini yaitu *source code* kernel sistem target, *toolchain compiler*, dan adb *tools* yang sudah terinstal di *workstation*. Dalam studi kasus ini ARM merupakan arsitektur yang digunakan pada perangkat Android, lalu untuk menjalankan modul pada sistem target tersebut modul harus di-*compile* pada sebuah sistem *host* untuk target spesifik terlebih dahulu. Hal ini disebut sebagai *cross compilation* dan *cross compiler* yang berupa *toolchain* dapat membuat modul yang tepat untuk sistem target tersedia secara *standalone* maupun dari Android NDK.

Source code kernel sistem target dapat diunduh dari situs *opensource.samsung.com* di mana Samsung yang memproduksi perangkat Android-nya. Penting untuk diketahui bahwa untuk penelitian ini diperlukan *source code* kernel yang tepat dan sesuai dengan sistem perangkat, jika *source code* kernel tidak sesuai maka modul yang dibuat juga tidak akan bisa dijalankan pada sistem target. Sudah diketahui dari sub bab sebelumnya *model number* dari perangkat yaitu GT-I9300, dengan mencari menggunakan parameter *model number* akan menampilkan hasil lalu memastikan mengunduh versi yang

sesuai dengan *baseband version* perangkat yaitu I9300XXEMG4 seperti yang bisa dilihat pada gambar 4.3.

Setelah mendapatkan *source code* kernel yaitu mengunduh *toolchain* yang sesuai dengan kernel sistem perangkat. Hal ini dapat diketahui dengan mengekstrak *source code* kernel tersebut lalu membuka *file* README_Kernel.txt dan dapat diketahui bahwa *toolchain* yang dibutuhkan yaitu arm-eabi-4.4.3. Mengunduh *toolchain* arm-eabi-4.4.3 dapat melalui *android.googlesource.com* untuk mencari *toolchain* yang sudah diarsipkan.

Mobile Phone	GT-I9300	XXUGMJ9	GT-I9300_JB_Opensource_Update12.zip
Mobile Phone	GT-I9300		GT-I9300_JB_Opensource_Update12.zip
Mobile Phone	GT-I9300	XXEMG4	GT-I9300_JB_Opensource_Update11.zip
Mobile Phone	GT-I9300	XXEMC2	GT-I9300_JB_Opensource_Update10.zip

Gambar 4.3. Hasil pencarian source code kernel

```
1. How to Build
        - get Toolchain
                From android git server , codesourcery and
etc ..
                 - arm-eabi-4.4.3
        - edit Makefile
                edit  "CROSS_COMPILE"  to  right  toolchain
path(You downloaded).
                  EX)    CROSS_COMPILE=  $(android  platform
directory       you       download)/android/prebuilt/linux-
x86/toolchain/arm-eabi-4.4.3/bin/arm-eabi-
          Ex)   CROSS_COMPILE=/usr/local/toolchain/arm-eabi-
4.4.3/bin/arm-eabi-        // check the location of toolchain

                $ make arch=arm m0_00_defconfig
                $ make
```

Kode 4.1. README_Kernel.txt

Sebelum *source code* kernel bisa di-*compile* menggunakan *toolchain* yang sudah ada, terdapat dua variabel yang perlu diperhatikan yaitu ARCH dan CROSS_COMPILE. Dua variabel tersebut dapat diubah menggunakan perintah *export* pada terminal Linux atau melalui *file* Makefile pada folder Kernel.

```
ARCH            ?= arm
CROSS_COMPILE   = /home/nero/android-ndk-r8e/toolchains/arm-
linux-androideabi-4.4.3/prebuilt/linux-x86_64/bin/arm-linux-
androideabi-
```

Kode 4.2. Menentukan direktori *toolchain* sebagai variabel

Terdapat hal yang perlu diperhatikan ketika ingin meng-*compile source code* kernel yaitu tahun dibuatnya *source code* tersebut. Untuk mengetahui pada tahun berapa *source code* kernel tersebut dibuat dapat diidentifikasi melalui tahun keluaran *firmware* yang bersangkutan, dalam kasus ini *firmware* yang digunakan yaitu versi I9300XXEMG4 yang rilis sekitar tahun 2013, berarti sudah sekitar tujuh tahun lalu. Hal ini dapat menjadi masalah ketika terdapat potongan kode yang sudah *deprecated* dan tidak digunakan lagi atau sudah diganti. Versi Ubuntu yang digunakan pada *virtual machine* ini yaitu 18.04.4 yang dirilis sekitar bulan Februari 2020.

Pada Ubuntu versi 18.04.4 terdapat bahasa pemrograman bernama Perl yang terdapat sedikit perubahan dengan tidak menggunakan sebuah fungsi bernama *defined* pada versi terbarunya, sedangkan *source code* kernel yang dibuat pada tahun 2013 masih menggunakan Perl dengan versi lebih lama masih menggunakan fungsi *defined* pada baris kodenya. Penggunaan fungsi *defined* ini dapat ditemukan pada *file* bernama *timeconst.pl* pada baris 373 seperti pada Kode 4.3 menunjukkan masih menggunakan fungsi *defined*, lalu untuk menangani terdapat dua cara yaitu men-*downgrade* versi Perl dengan menginstal Ubuntu versi yang lebih lama, atau dengan menghapus fungsi *defined* tersebut karena sudah tidak digunakan lagi dalam pemrograman bahasa Perl. Cara yang

dipilih tentu yang tidak menggunakan fungsi *defined* lagi dengan menghapus dari *source code*, maka setelah itu proses *compile* dapat dijalankan.

```
if ($hz < 1) {
                die "Usage: $0 HZ\n";
        }

        @val = @{$canned_values{$hz}};
        if (!defined(@val)) {
                @val = compute_values($hz);
        }
        output($hz, @val);
```

Kode 4.3. Fungsi *defined* pada *timeconts.pl* yang sudah *deprecated*

Sesuai dengan isi dari README_Kernel.txt terlebih dahulu membuat konfigurasi kernel dengan preset bernama *m0_00_defconfig* dengan pilihan arsitektur ARM namun tidak perlu dituliskan pada perintah di terminal karena sudah didefinisikan sebagai variabel ARCH pada Makefile. Setelah membuat konfigurasi kernel yaitu meng-*compile source code* kernel tersebut dengan *toolchain* yang akan menghasilkan kernel *image* dan berbagai *file* yang akan digunakan untuk tahap selanjutnya.

```
nero@ubuntu:~/XXEMG4/Kernel$ make m0_00_defconfig
nero@ubuntu:~/XXEMG4/Kernel$ make
```

Kode 4.4. *Compile source code* kernel

4.3.2 *Compile* Kernel dengan LiME

Tools yang akan digunakan untuk melakukan akuisisi data memori *volatile* adalah LiME (*Linux Memory Extractor*), namun sebelum bisa digunakan LiME harus di-*compile* berdasarkan kernel sistem target. Persiapan yang dilakukan sebelum meng-*compile* LiME yaitu menyesuaikan variabel direktori kernel dan *toolchain* pada *file* Makefile. Tujuan dari proses *compile* ini yaitu membuat modul berjenis *kernel object* (.ko) yang akan dimuat pada sistem perangkat. Pada saat memasukkan perintah *make* pada terminal perlu ditambahkan *flag* berupa *EXTRA_CFLAGS=-fno-pic* untuk menghindari

masalah berkaitan dengan *error "Unknown symbol _GLOBAL_OFFSET_TABLE_"* pada proses akuisisi nantinya.

```
nero@ubuntu:~/LiME/src$ make EXTRA_CFLAGS=-fno-pic
```

Kode 4.5. Meng-*compile* modul LiME

```
obj-m := lime.o
lime-objs := tcp.o disk.o main.o delfate.o hash.0

KDIR_GOLD := /home/nero/XXEMG4/Kernel

KVER := $(shell uname -r)

PWD := $(shell pwd)
CCPATH  :=  /home/nero/android-ndk-r8e/toolchains/arm-linux-
androideabi-4.4.3/prebuilt/linux-x86_64/bin
default:
        $(MAKE) ARCH=arm CROSS_COMPILE=$(CCPATH)/arm-linux-
androideabi- -C $(KDIR_GOLD) M="$(PWD)" modules
        $(CCPATH)/arm-linux-androideabi-strip       --strip-
unneeded lime.ko
        mv lime.ko lime-ready.ko

        $(MAKE) tidy

tidy:
        rm  -f  *.o  *.mod.c  Module.symvers  Module.markers
modules.order \.*.o.cmd \.*.ko.cmd \.*.o.d
        rm -rf \.tmp_versions

clean:
        $(MAKE) tidy
        rm -f *.ko
```

Kode 4.6. Makefile LiME

4.4 Persiapan Android Memory Extractor

Pada tahap ini dilakukan persiapan untuk dapat mengimplementasikan *tools* Android Memory Extractor (AMExtractor) dengan sukses. Beberapa hal dalam persiapan yang dilakukan yaitu mendapatkan konfigurasi perangkat.

4.4.1 Mendapatkan Konfigurasi Perangkat

AMExtractor berbeda dengan *tools* yang menggunakan LKM yang sudah di-*compile* dengan konfigurasi kernel perangkat yang mana *function address* dari sistem perangkat sudah diketahui, maka AMExtractor memerlukan cara lain untuk mendapatkan *function address* tersebut melalui */proc/kalsyms* untuk menentukan *address* yang diperlukan. Setelah itu memperhatikan *address* pada *vm_normal_page* untuk dijadikan acuan menemukan *function address* pada kernel *image*.

```
nero@ubuntu:~$ adb shell cat /proc/kallsyms
00000000 T stext
00000000 T _sinittext
00000000 T _stext
00000000 T __init_begin
00000000 t __create_page_tables
00000000 t __enable_mmu_loc
00000000 t __fixup_pv_table
00000000 t __vet_atags
00000000 t __fixup_smp
00000000 t __fixup_smp_on_up
.
.
.
```

Kode 4.7. Pencarian *address* untuk AMExtractor

Kode 4.7 menunjukkan bahwa keluaran perintah *cat /proc/kallsyms* memberikan umpan balik yang tidak diharapkan, karena setiap *address* yang ditampilkan tidak memiliki nilai. Dari hasil yang ditemukan ini tidak dapat dilanjutkan untuk melakukan akuisisi menggunakan AMExtractor.

4.5 Pembuatan *Profile* pada Volatility

Proses analisis data akuisisi memori *volatile* menggunakan Volatility Framework memerlukan *profile* yang dibuat berdasarkan sistem target. Hal ini dibutuhkan karena setiap sistem memiliki struktur pemataan memori yang berbeda, maka diperlukan rancangan pemetaan memori yang berasal dari sistem target. Pemetaan itu tersimpan dalam *file* bernama System.map yang tersimpan dalam folder kernel yang sudah di-

compile. Selain itu juga diperlukan *file* bernama module.dwarf yang di-*compile* dengan kernel dan *toolchain* terlebih dahulu, Makefile untuk proses *compile* terdapat pada direktori */tools/linux* dan layaknya pada LiME perlu merubah variabel direktori kernel dan *toolchain*.

```
obj-m += module.o
KDIR := /home/nero/XXEMG4/Kernel
CCPATH  :=  /home/nero/android-ndk-r8e/toolchains/arm-linux-
androideabi-4.4.3/prebuilt/linux-x86_64/bin
DWARFDUMP := dwarfdump

-include version.mk

all: dwarf

dwarf: module.c
        $(MAKE)  ARCH=arm  CROSS_COMPILE=$(CCPATH)/arm-linux-
androideabi- -C $(KDIR) CONFIG_DEBUG_INFO=y M=$(PWD) modules
        $(DWARFDUMP) -di module.ko > module.dwarf

clean:
        $(MAKE) -C $(KDIR) M="$(PWD)" clean
        rm -f module.dwarf
```

Kode 4.8. Makefile Volatility

Kedua *file* module.dwarf dan System.map ini perlu dimasukkan ke dalam .zip dan menempatkan .zip tersebut pada direktori */volatility/plugins/overlays/linux* untuk menjadikan sebuah *profile*. *Profile* yang sudah jadi dan siap dipakai akan muncul pada daftar *profile* ketika Volatility dijalankan.

```
nero@ubuntu:~/volatility$ python vol.py --info | grep Linux
Volatility Foundation Volatility Framework 2.6.1
LinuxLocalx64         - A Profile for Linux Local x64
LinuxS3_XXEMG4ARM     - A Profile for Linux S3_XXEMG4 ARM
LinuxAMD64PagedMemory            - Linux-specific AMD 64-bit
address space.
linux_aslr_shift            - Automatically detect the Linux
ASLR shift
linux_banner             - Prints the Linux banner information
linux_yarascan             - A shell in the Linux memory image
```

Kode 4.9. Volatility *profile listing*

BAB V
IMPLEMENTASI DAN HASIL

Pada bab ini, akan dijelaskan mengenai implementasi dari perancangan yang telah dilakukan sesuai dengan metode penelitian yang dibuat. Bagian implementasi akan menjelaskan mengenai penggunaan *tools* pada akusisi data memori *volatile* dan analisis hasil akusisi.

5.1 Akuisisi Data Memori *Volatile*

Pada proses akusisi ini akan dijalankan dengan metode yang menggunakan LiME. *Module* LiME akan dikirimkan kepada perangkat melalui ADB. Modul akan ditempatkan pada media penyimpanan internal perangkat pada direktori /sdcard/lime.ko menggunakan perintah *adb push*. ADB memanfaatkan fitur USB *Debugging* dan koneksi kabel USB melalui workstation.

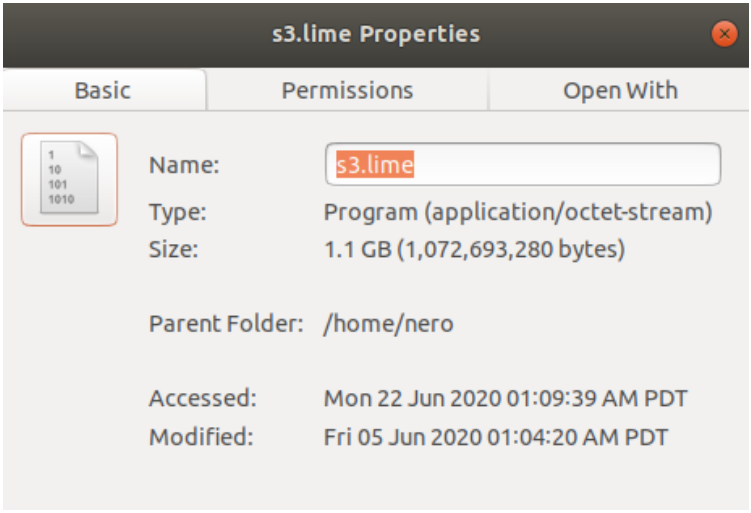
Setelah modul LiME sudah berada pada perangkat proses akuisisi dapat dimulai. Untuk melakukan *dump* pada memori digunakan jalur TCP yang menghubungkan perangkat dengan workstation. Proses akuisisi dimulai dengan menjalankan perintah *insmod* pada *shell* perangkat android, parameter yang digunakan yaitu *path* dengan nilai tcp:4444 menandakan *file* hasil *dump* akan dikirim melalui TCP dengan port 4444, lalu format yang digunakan yaitu *lime*. Pada terminal yang berbeda dijalankan perintah *nc* untuk menerima hasil *dump* memori dengan parameter jalur *localhost* dan port 4444.

```
nero@ubuntu:~/LiME/src$ adb push lime-ready.ko
/sdcard/lime.ko
lime-ready.ko: 1 file pushed. 0.6 MB/s (13408 bytes in 0.022s)
nero@ubuntu:~/LiME/src$ adb forward tcp:4444 tcp:4444
nero@ubuntu:~/LiME/src$ adb shell
shell@android:/ $ su
root@android:/ # insmod /sdcard/lime.ko "path=tcp:4444
format=lime"
```

Kode 5.1. Akuisisi memori menggunakan proses *dump* melalui TCP

```
nero@ubuntu:~$ nc localhost 4444 > s3.lime
```

Kode 5.2. Menyimpan hasil *dump* menggunakan *nc*



Gambar 5.1. Hasil *dump* memori

Setelah berhasil mendapatkan hasil *dump* memori, terdapat cara untuk memastikan apakah hasil *dump* tersebut benar atau tidak. Cara yang dapat dilakukan yaitu melihat *header* dari *file dump* memori tersebut dan memcocokkan nilai yang tertera dengan *address* pada memori yang dialokasikan untuk *System RAM*. Untuk mengetahui *address* apa yang dialokasikan untuk *System RAM* dapat dilihat menggunakan ADB dengan perintah *cat /proc/iomem* sesuai pada Kode 5.3.

```
nero@ubuntu:~$ adb shell cat /proc/iomem
.
.
.
13930000-139300ff : s3c64xx-spi.1
  13930000-139300ff : s3c64xx-spi
40000000-7fefffff : System RAM
  40057000-408f7fff : Kernel text
  408f8000-40ef5da7 : Kernel data
```

Kode 5.3. Menunjukkan *address* yang dialokasikan untuk memori

Pada hasil keluaran perintah pada Kode 5.3 dapat diketahui bahwa *address* yang dialokasikan untuk *System RAM* yaitu *40000000* hingga *7fefffff*. Langkah selanjutnya yaitu membuka *header* dari hasil *dump* menggunakan Hexdump untuk melihat cakupan *address* yang dicakup oleh proses akuisisi, pada Kode 5.4 diketahui bahwa *address* yang dicakup oleh proses akuisisi yaitu *00 00 00 40* hingga *ff ff ef 7f* yang cocok dengan keluaran pada Kode 5.3 namun dengan penulisan urutan yang sedikit terbalik.

```
nero@ubuntu:~$ hexdump -C -n 40 ~/s3.lime
00000000  45 4d 69 4c 01 00 00 00  00 00 00 40 00 00 00 00
|EMiL.......@....|
00000010  ff ff ef 7f 00 00 00 00  00 00 00 00 00 00 00 00
|................|
00000020        64    25    26    66    d8    69    ab    74
|d%&f.i.t|
00000028
```

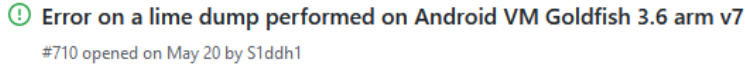
Kode 5.4. *Header* hasil *dump* memori

5.2 Analisis dengan Volatility

Pada proses analisis ini hasil akuisisi akan dibaca menggunakan Volatility Framework. *Profile* yang sudah dibuat pada sub bab 4.5 seharusnya sudah siap digunakan karena dibuat dengan rancangan pemetaan memori yang sama dengan sistem target. Volatility akan dijalankan menggunakan python, lebih tepatnya python2 lalu dengan parameter profil yang akan digunakan, direktori *file* hasil *dump*, dan pilihan *plugin* yang akan diterapkan. Plugin yang tersedia untuk menganalisa sistem Linux dapat diketahui menggunakan perintah --*info*. Plugin *linux_pslist* menyebutkan semua proses yang sedang berjalan di sistem target, perintah ini mirip dengan *ps* pada Linux.

Pada saat Volatility dijalankan sistem memberi umpan balik berupa *No suitable address space mapping found*. Masalah seperti ini tidak terjadi pada penelitian [5] dan [8] yang mana menjadikan penyebabnya juga tidak diketahui. Hal ini masih menjadi permasalahan bagi *developer* Volatility di mana terdapat *issue* pada halaman situs Github Volatility yang masih

berstatus *Open* dan belum terdapat solusi untuk menyelesaikan masalah tersebut, maka untuk saat ini belum bisa dilakukan analisis menggunakan Volatility [16].



Gambar 5.2. *Open Issue* pada situs GitHub Volatility

```
nero@ubuntu:~/volatility$ python vol.py --info
Volatility Foundation Volatility Framework 2.6.1
Plugins
-------
linux_apihooks             - Checks for userland apihooks
linux_arp                  - Print the ARP table
linux_aslr_shift            - Automatically detect the Linux
ASLR shift
linux_banner            - Prints the Linux banner information
linux_bash                  - Recover bash history from bash
process memory
linux_bash_env                 - Recover a process' dynamic
environment variables
linux_bash_hash              - Recover bash hash table from
bash process memory
linux_check_afinfo         - Verifies the operation function
pointers of network protocols
linux_check_creds             - Checks if any processes are
sharing credential structures
linux_check_evt_arm       - Checks the Exception Vector Table
to look for syscall table hooking
linux_check_fop            - Check file operation structures
for rootkit modifications
```

Kode 5.5. Volatility Linux *plugin*

```
nero@ubuntu:~/volatility$ python2 vol.py --
profile=LinuxS3_XXEMG4ARM -f ~/s3.lime linux_pslist
Volatility Foundation Volatility Framework 2.6.1
Offset      Name                    Pid                 PPid
Uid             Gid    DTB        Start Time
---------- -------------------- --------------- ------------
--- --------------- ------ ---------- ----------
No suitable address space mapping found
```

Kode 5.6. Penggunaan *plugin* Volatility pada hasil *dump* memori

5.3 Analisis dengan Android Debug Bridge

Terdapat alternatif untuk melakukan analisis memori *volatile* perangkat yaitu menggunakan ADB. ADB menyediakan perintah *ps* yang mirip dengan plugin *linux_pslist* pada Volatility.

```
shell@android:/ $ ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
root      12730 2     0      0     ffffffff 00000000 S flush-
0:17
root       12848 2       0        0       ffffffff 00000000 S
migration/3
root       12850 2       0        0       ffffffff 00000000 S
kworker/3:0
root       12851 2       0        0       ffffffff 00000000 S
ksoftirqd/3
root       12852 2       0        0       ffffffff 00000000 S
watchdog/3
root       12854 2       0        0       ffffffff 00000000 S
kworker/3:1
u0_a89      12876  1908   491444  45336  ffffffff  00000000  S
com.sec.android.app.camera
root       12997 2       0        0       ffffffff 00000000 S
migration/2
root       12998 2       0        0       ffffffff 00000000 S
kworker/2:0
root       13000 2       0        0       ffffffff 00000000 S
ksoftirqd/2
root       13002 2       0        0       ffffffff 00000000 S
watchdog/2
root       13003 2       0        0       ffffffff 00000000 S
kworker/2:1
u0_a0       13162  1908   475780  29836  ffffffff  00000000  S
com.sec.android.daemonapp.ap.accuweather
u0_a143     13529  1908   591416  72404  ffffffff  00000000  S
com.whatsapp
root       13925 2       0        0       ffffffff 00000000 S
kworker/3:2
shell       13948  2026   872     332     c00614c4 40037500 S
/system/bin/sh
shell     13950 13948 1088   232   00000000 400b495c R ps
```

Kode 5.7. Hasil keluaran perintah *ps* pada ADB

Pada hasil keluaran perintah *ps* dari Kode 5.7 dapat diketahui bahwa terdapat proses *com.sec.android.app.camera* dan *com.whatsapp* terdaftar pada *list* proses. Hal ini dapat terjadi karena aplikasi tersebut pernah dibuka dan belum dibersihkan dari *Recents Task* mengakibatkan aplikasi masih menempati *space* pada memori.

5.3.1 Pengujian Perubahan Memori akibat *Rooting*

Akses *root* yang didapatkan melalui aplikasi KingRoot juga dapat terlihat pada proses *com.kingroot.kinguser:service* menandakan aplikasi tersebut berjalan sebagai *service* pada *background* dan menempati memori *virtual* dengan ukuran sebesar 542684 untuk menjalankan */data/data/com.kingroot.kinguser/applib/ktools* yang berfungsi untuk mendapatkan akses *root*. Berdasarkan hasil pada Kode 5.8 dapat diketahui bahwa aplikasi KingRoot berjalan sebagai *service* di *background* dan menempati *space* pada memori.

```
shell@android:/ $ ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
u0_a142     3268   1908   547948  38776  ffffffff  00000000  S
com.kingroot.kinguser:service
u0_a9       3421   1908   541000  28956  ffffffff  00000000  S
android.process.acore
u0_a32      3557   1908   492648  30012  ffffffff  00000000  S
android.process.media
u0_a19      3617   1908   565804  59112  ffffffff  00000000  S
com.google.android.gms.persistent
root        3758   1       13196   1048   ffffffff  00000000  S
kworker/271
root        3765   1       2956    436    ffffffff  00000000  S
kworker/271
u0_a142    3781   3268   2956    204    ffffffff  00000000  S
k_worker/41:10142
root        3786   1       5004    276    ffffffff  00000000  S
k_worker/41:0
root      3787  3786  892    292   ffffffff 00000000 S sh
root        3877   1       332     28     ffffffff  00000000  S
/data/data/com.kingroot.kinguser/applib/ktools.
```

Kode 5.8. Aplikasi KingRoot menempati memori

Hal yang perlu diperhatikan oleh penyelidik forensik dalam melakukan *rooting* perangkat yang belum mempunya akses root yaitu seberapa jauh perubahan pada memori setelah proses *rooting* dilakukan. Untuk mengetahui perubahan memori sebelum dan sesudah perangkat dilakukan proses *rooting* dapat menggunakan skenario perangkat yang belum di-*root* melakukan aktivitas tersambung ke jaringan, membuka aplikasi kamera, dan membuka aplikasi WhatsApp, kemudian pada kondisi perangkat saat ini dilakukan analisis proses apa saja yang sedang berjalan pada memori. Setelah itu proses *rooting* dilakukan sesuai dengan sub bab 4.2 bagian *rooting* perangkat. Analisis dilakukan kembali setelah proses *rooting* selesai lalu hasil sebelum dan sesudah *rooting* dibandingkan untuk mengetahui apa saja perubahan yang terjadi pada memori.

```
shell@android:/ $ ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
u0_a89      17478  1908   509212  39868  ffffffff  00000000  S
com.sec.android.app.camera
u0_a22      17504  1908   481072  29732  ffffffff  00000000  S
com.sec.android.app.clockpackage
u0_a137     17574  1908   476348  27944  ffffffff  00000000  S
com.sec.android.sCloudBackupApp
u0_a138     17617  1908   476132  28236  ffffffff  00000000  S
com.sec.android.sCloudBackupProvider
u0_a1        17637  1908   476672  27996  ffffffff  00000000  S
com.sec.android.widgetapp.favoriteswidget
u0_a0        17858  1908   475684  27388  ffffffff  00000000  S
com.sec.android.daemonapp.ap.accuweather
u0_a78      17956  1908   578864  65672  ffffffff  00000000  S
com.android.vending
u0_a17      18071  1908   496804  30400  ffffffff  00000000  S
com.sec.chaton
u0_a85      18138  1908   482444  29724  ffffffff  00000000  S
com.sec.spp.push:RemoteDlcProcess
u0_a78      18152  1908   513168  41088  ffffffff  00000000  S
com.android.vending:download_service
u0_a143     18474  1908   586988  67908  ffffffff  00000000  S
com.whatsapp
```

Kode 5.9. Memori sebelum *rooting*

Pada Kode 5.9 menunjukkan keadaan memori sebelum dilakukan proses *rooting*, diketahui bahwa proses *com.sec.android.app.camera* memiliki PID 17478 dan proses *com.whatsapp* memiliki PID 18474. Lalu pada Kode 5.10 memunjukkan keadaan memori setelah dilakuakn proses *rooting*, diketahui bahwa proses *com.sec.android.app.camera* yang sebelumnya memiliki PID 17478 dan proses *com.whatsapp* yang memiliki PID 18474 hilang dari memori. Setelah *rooting* ini dilakukan mengakibatkan aplikasi KingRoot dan */data/data/com.kingroot.kinguser/applib/ktools* menempati memori dengan PID 19708 dan 20329.

```
shell@android:/ $ ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
root        14114 2       0        0       ffffffff 00000000 S
kworker/1:1
u0_a17      18071  1908   496804  25108  ffffffff  00000000  S
com.sec.chaton
u0_a107     18979  1908   497624  32500  ffffffff  00000000  S
com.sec.android.app.myfiles
u0_a142     19708  1908   594736  64604  ffffffff  00000000  S
com.kingroot.kinguser
u0_a142   19745 19708 880    260   ffffffff 00000000 S sh
u0_a47      19962  1908   526864  35228  ffffffff  00000000  S
com.google.android.gm
u0_a142    20329 1       328      24     ffffffff 00000000 S
/data/data/com.kingroot.kinguser/applib/ktools
.
.
.
```

Kode 5.10. Memori setelah *rooting*

Tabel 5.1. Perubahan memori akibat *rooting*

Waktu	Aplikasi	PID	VSIZE	RSS
Sebelum Rooting	Kamera	17478	509212	39868
	WA	18474	586988	67908
	KingRoot	-	-	-
	root lib	-	-	-

Setelah Rooting	Kamera	-	-	-
	WA	-	-	-
	KingRoot	19708	594736	64604
	root lib	20329	328	24

Untuk mengetahui konsistensi dari hasil yang didapatkan perlu dilakukan pengujian lagi untuk membuktikan konsistensi dari hasil yang didapatkan. Tabel 5.2 hingga 5.6 menunjukkan hasil pengujian perubahan memori akibat *rooting* dengan skenario setelah dilakukan *rooting* aplikasi kamera dan WhatsApp dibuka kembali untuk mengetahui perubahan PID.

Tabel 5.2. Pengujian perubahan memori nomor 1

Waktu	Aplikasi	PID	VSIZE	RSS
Sebelum Rooting	Kamera	8892	491504	39512
	WA	10855	592328	76728
	KingRoot	-	-	-
	root lib	-	-	-
Setelah Rooting	Kamera	-	-	-
	WA	-	-	-
	KingRoot	12247	576748	62796
	root lib	13007	328	24
Buka ulang aplikasi	Kamera	21753	492668	39940
	WA	22253	593432	76812
	KingRoot	12279	548680	42468
	root lib	13007	328	24

Tabel 5.3. Pengujian perubahan memori nomor 2

Waktu	Aplikasi	PID	VSIZE	RSS
Sebelum Rooting	Kamera	8198	491972	45076
	WA	8919	593240	77244
	KingRoot	-	-	-

	root lib	-	-	-
Setelah Rooting	Kamera	-	-	-
	WA	-	-	-
	KingRoot	10140	581676	58188
	root lib	10824	328	24
Buka ulang aplikasi	Kamera	20172	510676	41032
	WA	20492	584864	73400
	KingRoot	10166	552840	40496
	root lib	10824	328	24

Tabel 5.4. Pengujian perubahan memori nomor 3

Waktu	Aplikasi	PID	VSIZE	RSS
Sebelum Rooting	Kamera	6807	491208	42820
	WA	8066	592340	66484
	KingRoot	-	-	-
	root lib	-	-	-
Setelah Rooting	Kamera	-	-	-
	WA	-	-	-
	KingRoot	10190	594724	66984
	root lib	10960	328	24
Buka ulang aplikasi	Kamera	19157	511140	39936
	WA	19516	589780	73024
	KingRoot	10190	578732	60656
	root lib	10960	328	24

Tabel 5.5. Pengujian perubahan memori nomor 4

Waktu	Aplikasi	PID	VSIZE	RSS
Sebelum Rooting	Kamera	7673	493228	43660
	WA	7156	596572	73892
	KingRoot	-	-	-
	root lib	-	-	-

Setelah Rooting	Kamera	-	-	-
	WA	-	-	-
	KingRoot	10490	592544	62820
	root lib	10945	328	24
Buka ulang aplikasi	Kamera	19894	510640	39936
	WA	20292	591260	74528
	KingRoot	10490	578732	59552
	root lib	10945	328	24

Tabel 5.6. Pengujian perubahan memori nomor 5

Waktu	Aplikasi	PID	VSIZE	RSS
Sebelum Rooting	Kamera	11053	491608	41716
	WA	11547	584880	73896
	KingRoot	-	-	-
	root lib	-	-	-
Setelah Rooting	Kamera	-	-	-
	WA	-	-	-
	KingRoot	13060	599036	62708
	root lib	13629	328	24
Buka ulang aplikasi	Kamera	22637	494056	40184
	WA	24112	588164	75460
	KingRoot	13060	581304	58712
	root lib	13629	328	24

Dari hasil lima pengujian perubahan memori dapat disimpulkan bahwa proses *com.sec.android.app.camera* dan *com.whatsapp* selalu terhapus dari memori ketika dilakukan proses *rooting*, lalu ketika aplikasi tersebut dibuka kembali akan mendapatkan PID baru yang sudah berbeda dengan PID pada awalnya. VSIZE dari kedua aplikasi juga mengalami perubahan dengan rata-rata untuk aplikasi kamera sebesar 11932 dan WhatsApp sebesar 2372. RSS juga mengalami perubahan dengan rata-rara untuk aplikasi kamera sebesar 2351 dan

WhatsApp sebesar 996. Belum diketahui mengapa terjadi perubahan terhadap VSIZE dan RSS aplikasi setelah terjadi proses *rooting* pada perangkat.

5.3.2 Pengujian Urutan PID

Hasil keluaran perintah *ps* dari Kode 5.7 dapat diketahui bahwa proses *com.sec.android.app.camera* memiliki PID atau *process ID* 12876 dan proses *com.whatsapp* memiliki PID 13529. Proses *com.sec.android.app.camera* memiliki PID yang angkanya lebih kecil dibandingkan *com.whatsapp*, hal ini menunjukkan proses *com.sec.android.app.camera* dibuka terlebih dahulu kemudian dibuka proses *com.whatsapp*, namun untuk membuktikan apakah benar penomoran PID berkaitan dengan urutan pembukaan aplikasi dapat dilakukan dengan menguji berbagai skenario membuka aplikasi.

Skenario pertama yaitu melakukan seperti pada rencana utama, dilakukan dengan membuka aplikasi kamera terlebih dahulu, menutupnya dengan tombol *back*, membuka aplikasi WhatsApp, lalu menutup denvgan tombol *back*. Sesuai dengan hasil luaran Kode 5.11 proses *com.sec.android.app.camera* memiliki PID 29010 yang lebih kecil dari PID proses *com.whatsapp* yaitu 29979.

```
nero@ubuntu:~$ adb shell ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
u0_a89      29010  1908   525492  42644  ffffffff  00000000  S
com.sec.android.app.camera
root        29166 2      0        0      ffffffff 00000000 S
migration/2
u0_a0        29276  1908   475660  28636  ffffffff  00000000  S
com.sec.android.daemonapp.ap.accuweather
u0_a2        29301  1908   478108  30680  ffffffff  00000000  S
com.sec.android.widgetapp.ap.hero.accuweather
u0_a1        29336  1908   476648  29584  ffffffff  00000000  S
com.sec.android.widgetapp.favoriteswidget
u0_a143     29979  1908   585912  70788  ffffffff  00000000  S
com.whatsapp
```

Kode 5.11. Skenario pertama pengujian urutan PID

Skenario kedua yaitu membuka aplikasi WhatsApp terlebih dahulu, menutup aplikasi dengan tombol *back*, kemudian membuka aplikasi kamera, lalu menutupnya dengan tombol *back.* Hasil keluaran perintah *ps* kali ini pada Kode 5.12 menunjukkan bahwa proses *com.whatsapp* memiliki PID 11172 yang lebih kecil dari PID proses *com.sec.android.app.camera* yaitu 11504.

```
nero@ubuntu:~$ adb shell ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
u0_a143     11172  1908   577636  69348  ffffffff  00000000  S
com.whatsapp
u0_a2        11328  1908   479152  32264  ffffffff  00000000  S
com.sec.android.widgetapp.ap.hero.accuweather
u0_a1        11345  1908   476708  30928  ffffffff  00000000  S
com.sec.android.widgetapp.favoriteswidget
u0_a89       11504  1908   491556  41320  ffffffff  00000000  S
com.sec.android.app.camera
root        11653  2       0        0       ffffffff  00000000  S
migration/2
root        11654  2       0        0       ffffffff  00000000  S
kworker/2:0
root        11655  2       0        0       ffffffff  00000000  S
ksoftirqd/2
root        11656  2       0        0       ffffffff  00000000  S
watchdog/2
root        11712  2       0        0       ffffffff  00000000  S
kworker/2:1
u0_a0        11745  1908   475720  30236  ffffffff  00000000  S
com.sec.android.daemonapp.ap.accuweather
u0_a82       11921  1908   484548  34052  ffffffff  00000000  S
com.samsung.rcs:service
u0_a17       11979  1908   497116  37536  ffffffff  00000000  S
com.sec.chaton
u0_a88       12008  1908   479668  32232  ffffffff  00000000  S
com.sec.android.app.samsungapps.una2
shell        12034  1978   872      332    c00614c4  400ea500  S
/system/bin/sh
shell     12036 12034 1088   232   00000000 4007695c R ps
```

Kode 5.12. Skenario kedua pengujian urutan PID

Skenario ketiga yaitu membuka aplikasi WhatsApp terlebih dahulu, menutup aplikasi dengan tombol *back*,

kemudian membuka aplikasi kamera, menutupnya dengan tombol *back*, lalu aplikasi WhatsApp ditutup melalui *Recent Apps* dengan menggeser hingga hilang, dan membuka aplikasi WhatsApp kembali. Hasil keluaran perintah *ps* yang dilakukan saat aplikasi WhatsApp dan kamera ditutup, pada Kode 5.13 menunjukkan bahwa proses *com.whatsapp* memiliki PID 19576 yang lebih kecil dari PID proses *com.sec.android.app.camera* yaitu 19961.

```
nero@ubuntu:~$ adb shell ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
u0_a143     19576  1908   585096  73244  ffffffff  00000000  S
com.whatsapp
root        19656 2       0        0       ffffffff 00000000 S
migration/1
root        19657 2       0        0       ffffffff 00000000 S
kworker/1:0
root        19658 2       0        0       ffffffff 00000000 S
ksoftirqd/1
root        19659 2       0        0       ffffffff 00000000 S
watchdog/1
root        19660 2       0        0       ffffffff 00000000 S
kworker/1:1
u0_a89      19961  1908   513304  45524  ffffffff  00000000  S
com.sec.android.app.camera
.
.
.
```

Kode 5.13. Skenario ketiga pengujian urutan PID

Lalu setelah itu aplikasi WhatsApp ditutup melalui *Recent Apps* dan membukanya kembali, pada Kode 5.14 menunjukkan bahwa proses *com.sec.android.app.camera* memiliki PID 19961 yang lebih kecil dari PID proses *com.whatsapp* yaitu 20931. Pada skenario ketiga ini dapat diketahui bahwa proses *com.sec.android.app.camera* yang masih berada pada *Recent Apps* memiliki PID yang tetap yaitu 19961, sedangkan untuk proses *com.whatsapp* pada awalnya memiliki PID 19576 lalu setelah ditutup melalui *Recent Apps* dan dibuka kembali PID-nya berubah menjadi 20931.

```
nero@ubuntu:~$ adb shell ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
u0_a89      19961 1908   513304 45524 ffffffff 00000000 S
com.sec.android.app.camera
root        20686 2      0        0      ffffffff 00000000 S
migration/3
root        20687 2      0        0      ffffffff 00000000 S
kworker/3:0
root        20688 2      0        0      ffffffff 00000000 S
ksoftirqd/3
root        20689 2      0        0      ffffffff 00000000 S
watchdog/3
root        20694 2      0        0      ffffffff 00000000 S
kworker/3:1
u0_a143     20931 1908   585040 72832 ffffffff 00000000 S
com.whatsapp
.
.
.
```

Kode 5.14. Skenario ketiga pengujian urutan PID setelah WhatsApp dibuka kembali

Tabel 5.7. Rekapitulasi hasil pengujian urutan PID

Aplikasi	PID			
	Skenario 1	Skenario 2	Skenario 3	Skenario 3+
Kamera	29010	11504	19961	19961
WA	29979	11172	19576	20931

Hal ini dimungkinkan karena pada saat menutup aplikasi WhatApp melalui *Recent Apps* berdampak seperti meng-*kill* proses tersebut dan menghapus dari memori, lalu pada saat membuka aplikasi WhatsApp Kembali proses *com.whatsapp* mendapatkan PID baru yang angkanya berada di atas proses *com.sec.android.app.camera*. Berdasarkan tiga skenario yang dilakukan untuk menguji urutan PID dapat disimpulkan bahwa benar urutan penomoran PID berkaitan dengan urutan dibukanya aplikasi.

5.3.3 Pengujian Ketahanan Memori

Pada penelitian ini percobaan yang dilakukan diharapkan menyerupai kejadian di lapangan, di mana terdapat pelaku kejahatan yang menggunakan perangkat Android untuk mendukung kejahatannya, lalu ketika pelaku tersebut ditangkap dan perangkatnya menjadi barang bukti perangkat tersebut akan dilakukan penyelidikan forensik untuk membuktikan apakah benar pelaku menggunakan perangkat tersebut untuk membantu kejahatannya. Namun pada kenyataannya akan terdapat selang waktu dari pelaku menggunakan perangkatnya hingga pelaku ditangkap, dan hingga barang bukti dilakukan proses forensik, maka untuk mengetahui seberapa lama sebuah aktivitas yang terekam pada memori *volatile* dapat bertahan sebelum akhirnya terhapus dari memori dan tidak bisa menjadi barang bukti.

Untuk Skenario yang dilakukan yaitu aktivitas akan dijalankan pada perangkat lalu akan dianalisis sesegera mungkin, kemudian untuk analisis berikutnya akan dilakukan setiap sekitar 30 menit hingga sudah tidak terjadi perubahan lagi terhadap proses aplikasi kamera dan WhatsApp. Hasil keluaran dari Kode 5.15 yang dilakukan pada pukul 12.52 menunjukkan bahwa proses *com.sec.android.app.camera* memiliki PID 15482, VSIZE sebesar 491668, dan RSS sebesar 40036, untuk proses *com.whatsapp* memiliki PID 16196, VSIZE sebesar 587132, dan RSS sebesar 71172.

```
nero@ubuntu:~$ adb shell ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
u0_a89      15482 1908   491668 40036 ffffffff 00000000 S
com.sec.android.app.camera
u0_a6       15736 1908   477844 32972 ffffffff 00000000 S
com.sec.android.allshare.framework
u0_a5       15750 1908   476304 29376 ffffffff 00000000 S
com.sec.android.nearby.mediaserver
u0_a84      15796 1908   479116 30536 ffffffff 00000000 S
com.sec.android.app.sns3
u0_a143     16196 1908   587132 71172 ffffffff 00000000 S
com.whatsapp.
```

Kode 5.15. Pengujian ketahanan memori tahap 1

Selama waktu tunggu untuk analisis berikutnya, perangkat terkoneksi ke jaringan dan tidak dilakukan aktivitas lain. Pada pukul 13.35 dilakukan analisis berikutnya dan berdasarkan hasil keluaran dari Kode 5.16 menunjukkan bahwa proses *com.sec.android.app.camera* memiliki PID dan VSIZE yang tetap yaitu PID 15482 dan VSIZE 491668, namun untuk RSS berkurang menjadi 36540. Untuk proses *com.whatsapp* memiliki PID yang sama yaitu 16196, namun VSIZE dan RSS berkurang menjadi VSIZE 585404 dan RSS 56596.

```
nero@ubuntu:~$ adb shell ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
u0_a89      15482  1908   491668  36540  ffffffff  00000000  S
com.sec.android.app.camera
u0_a6       15736  1908   477844  29756  ffffffff  00000000  S
com.sec.android.allshare.framework
u0_a5       15750  1908   476304  27792  ffffffff  00000000  S
com.sec.android.nearby.mediaserver
u0_a84      15796  1908   479116  28260  ffffffff  00000000  S
com.sec.android.app.sns3
u0_a143     16196  1908   585404  56596  ffffffff  00000000  S
com.whatsapp
.
.
.
```

Kode 5.16. Pengujian ketahanan memori tahap 2

Pada pukul 14.01 dilakukan analisis lagi dan berdasarkan hasil keluaran pada Kode 5.17 menunjukkan bahwa proses *com.sec.android.app.camera* sudah terhapus dari memori, namun proses *com.whatsapp* masih bertahan dan dengan PID dan VSIZE yang sama yaitu PID 16196 dan VSIZE 585404, melainkan RSS terdapat perubahan menjadi 57148. Proses *com.sec.android.app.camera* teridentifikasi tidak bertahan pada analisis setelah satu jam berlalu, maka proses tersebut hilang dari memori setelah sekitar 30 hingga 60 menit ketika tidak ada aktivitas dari aplikasi.

```
nero@ubuntu:~$ adb shell ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
u0_a103     10678  1908   523596  50452  ffffffff  00000000  S
com.sec.android.gallery3d
u0_a22      13800  1908   482108  30496  ffffffff  00000000  S
com.sec.android.app.clockpackage
u0_a1        14003  1908   476648  28616  ffffffff  00000000  S
com.sec.android.widgetapp.favoriteswidget
u0_a11      15136  1908   477648  28764  ffffffff  00000000  S
com.sec.android.provider.badge
u0_a55      15149  1908   478332  28700  ffffffff  00000000  S
com.samsung.helphub
u0_a2        15212  1908   478112  29188  ffffffff  00000000  S
com.sec.android.widgetapp.ap.hero.accuweather
u0_a143     16196  1908   585404  57148  ffffffff  00000000  S
com.whatsapp
.
.
.
```

Kode 5.17. Pengujian ketahanan memori tahap 3

Pada pukul 14.30 dilakukan analisis kembali dan berdasarkan hasil keluaran pada Kode 5.18 menunjukkan bahwa proses *com.whatsapp* sudah terhapus dari memori, maka proses *com.whatsapp* hanya dapat bertahan pada memori dalam jangka waktu sekitar 60 hingga 90 menit ketika tidak ada aktivitas aplikasi, Tabel 5.8 menunjukkan rekapitulasi hasil pengujian.

Tabel 5.8. Keadaan aplikasi berdasarkan waktu

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	15482	491668	40036
	WA	16196	587132	71172
30 menit	kamera	15482	491668	36540
	WA	16196	585404	56596
60 menit	kamera	-	-	-
	WA	16196	585404	57148
90 menit	kamera	-	-	-
	WA	-	-	-

```
nero@ubuntu:~$ adb shell ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
.
.
.
u0_a22      13800 1908   482112  30480 ffffffff 00000000 S
com.sec.android.app.clockpackage
u0_a1        14003 1908   476648 28756 ffffffff 00000000 S
com.sec.android.widgetapp.favoriteswidget
u0_a11      15136 1908   478688 28784 ffffffff 00000000 S
com.sec.android.provider.badge
u0_a55      15149 1908   478332 28684 ffffffff 00000000 S
com.samsung.helphub
u0_a2        15212 1908   479152 29176 ffffffff 00000000 S
com.sec.android.widgetapp.ap.hero.accuweather
system      17476 1908   477544 30568 ffffffff 00000000 S
com.wssyncmldm
root        17802 2      0        0      ffffffff 00000000 S
kworker/u:0
root        17810 2      0        0      ffffffff 00000000 S
kworker/0:0
u0_a67      17952 1908   484260 32716 ffffffff 00000000 S
com.samsung.music
u0_a123     17966 1908   481668 30720 ffffffff 00000000 S
com.sec.android.app.videoplayer
.
.
.
```

Kode 5.18. Pengujian ketahanan memori tahap 4

Untuk mendapatkan hasil yang lebih akurat dilakukan pengujian sebanyak 5 kali dan dengan skenario yang berbeda untuk mengetahui ketahanan memori menyimpan jejak dalam berbagai kondisi. Skenario pertama yaitu sama dengan pengujian awal yaitu membuka aplikasi kamera dahulu lalu membuka aplikasi WhatsApp. Tabel 5.9 hingga 5.13 merupakan hasil pengujian skenario pertama. Kolom pada tabel yang berwarna merah menandakan aplikasi tersebut sudah hilang dari memori, lalu kolom yang berwarna kuning menandakan terjadi perubahan PID dari kondisi awal. Hasil skenario 1 nomor 1 pada Tabel 5.9 menunjukkan hasil yang sama dengan pengujian di awal yaitu aplikasi kamera bertahan 30 hingga 60 menit, lalu aplikasi WhatsApp bertahan 60 hingga 90 menit.

Tabel 5.9. Hasil pengujian ketahanan memori skenario 1 nomor 1

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	19773	493504	41760
	WA	20676	593076	74604
30 menit	kamera	19773	493504	40640
	WA	20676	589956	65256
60 menit	kamera	-	-	-
	WA	20676	590996	59200
90 menit	kamera			
	WA			

Hasil skenario 1 nomor 2 pada Tabel 5.10 menunjukkan hasil yang berbeda yaitu aplikasi kamera bertahan 60 hingga 90 menit, lalu aplikasi WhatsApp bertahan 150 hingga 180 menit.

Tabel 5.10. Hasil pengujian ketahanan memori skenario 1 nomor 2

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	27077	491428	42972
	WA	27854	591316	75556
30 menit	kamera	27077	491428	39004
	WA	27854	590276	55984
60 menit	kamera	27077	491428	38004
	WA	27854	590276	54496
90 menit	kamera	-	-	-
	WA	27854	590276	54352
120 menit	kamera	-	-	-
	WA	27854	590276	52856
150 menit	kamera	-	-	-
	WA	27854	590276	52404
180 menit	kamera			
	WA			

Hasil skenario 1 nomor 3 pada Tabel 5.11 menunjukkan hasil yaitu aplikasi kamera bertahan 0 hingga 30 menit begitu juga dengan aplikasi WhatsApp, namun aplikasi WhatsApp terlacak kembali dengan PID yang berbeda.

Tabel 5.11. Hasil pengujian ketahanan memori skenario 1 nomor 3

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	9235	490936	40876
	WA	10064	593376	74060
30 menit	kamera	-	-	-
	WA	-	-	-
60 menit	kamera	-	-	-
	WA	12538	566636	49724
90 menit	kamera	-	-	-
	WA	12538	566636	49888
120 menit	kamera	-	-	-
	WA	14480	569800	52868
150 menit	kamera			
	WA			

Hasil skenario 1 nomor 4 pada Tabel 5.12 menunjukkan hasil yaitu aplikasi kamera bertahan 0 hingga 30 menit, lalu aplikasi WhatsApp bertahan hingga 180 hingga 210 menit.

Tabel 5.12. Hasil pengujian ketahanan memori skenario 1 nomor 4

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	16437	493760	43300
	WA	17261	592240	76840
30 menit	kamera	-	-	-
	WA	17261	590160	63148
60 menit	kamera	-	-	-
	WA	17261	591200	62776
	kamera	-	-	-

90 menit	WA	17261	591200	62424
120 menit	kamera	-	-	-
	WA	17261	591200	50784
150 menit	kamera	-	-	-
	WA	17261	591200	50688
180 menit	kamera	-	-	-
	WA	17261	591200	51344

Hasil skenario 1 nomor 5 pada Tabel 5.13 menunjukkan hasil yaitu aplikasi kamera bertahan 30 hingga 60 menit, lalu aplikasi WhatsApp bertahan hingga 300 hingga 330 menit.

Tabel 5.13. Hasil pengujian ketahanan memori skenario 1 nomor 5

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	23834	492720	42100
	WA	25049	592268	76688
30 menit	kamera	23834	492720	41412
	WA	25049	590188	64556
60 menit	kamera	-	-	-
	WA	25049	591228	52532
90 menit	kamera	-	-	-
	WA	25049	591228	51760
120 menit	kamera	-	-	-
	WA	25049	591228	50764
150 menit	kamera	-	-	-
	WA	25049	591228	50464
180 menit	kamera	-	-	-
	WA	25049	591228	50512
210 menit	kamera	-	-	-
	WA	25049	591228	50604
	kamera	-	-	-

240 menit	WA	25049	591228	50684
270 menit	kamera	-	-	-
	WA	25049	591228	50468
300 menit	kamera	-	-	-
	WA	25049	591228	50604

Skenario kedua yaitu membalik urutan membuka aplikasi, maka membuka aplikasi WhatsApp dahulu kemudian aplikasi kamera. Tabel 5.14 hingga 5.18 merupakan hasil pengujian skenario kedua. Hasil skenario 2 nomor 1 pada Tabel 5.14 menunjukkan hasil aplikasi kamera bertahan 120 hingga 150 menit, lalu aplikasi WhatsApp bertahan 60 hingga 90 menit namun kembali terlacak pada memori dengan PID yang berbeda.

Tabel 5.14. Hasil pengujian ketahanan memori skenario 2 nomor 1

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	14588	525268	40476
	WA	10297	616924	72760
30 menit	kamera	14588	525268	37300
	WA	10297	609636	71336
60 menit	kamera	14588	526308	31836
	WA	17466	571872	56800
90 menit	kamera	14588	526308	31940
	WA			
120 menit	kamera	14588	526308	31620
	WA	21350	571856	51096
150 menit	kamera			
	WA			

Hasil skenario 2 nomor 2 pada Tabel 5.15 menunjukkan hasil aplikasi kamera bertahan 0 hingga 30 menit, lalu aplikasi WhatsApp bertahan 150 hingga 180 menit.

Tabel 5.15. Hasil pengujian ketahanan memori skenario 2 nomor 2

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	22039	508068	44768
	WA	21361	598644	74056
30 menit	kamera	-	-	-
	WA	21361	596724	64032
60 menit	kamera	-	-	-
	WA	21361	596724	56144
90 menit	kamera	-	-	-
	WA	21361	596724	54540
120 menit	kamera	-	-	-
	WA	21361	596724	53596
150 menit	kamera	-	-	-
	WA	21361	596724	52640
180 menit	kamera			
	WA			

Hasil skenario 2 nomor 3 pada Tabel 5.16 menunjukkan hasil aplikasi kamera bertahan 30 hingga 60 menit, lalu aplikasi WhatsApp bertahan 60 hingga 90 menit.

Tabel 5.16. Hasil pengujian ketahanan memori skenario 2 nomor 3

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	12214	491732	44484
	WA	11752	597396	68136
30 menit	kamera	12214	491700	40968
	WA	11752	596576	63036
60 menit	kamera	-	-	-
	WA	11752	596576	62640

90 menit	kamera			
	WA			

Hasil skenario 2 nomor 4 pada Tabel 5.17 menunjukkan hasil aplikasi kamera bertahan 60 hingga 90 menit, lalu aplikasi WhatsApp mengalami perubahan PID pada waktu antara 30 dan 60 menit.

Tabel 5.17. Hasil pengujian ketahanan memori skenario 2 nomor 4

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	16843	491988	41732
	WA	16431	592180	60948
30 menit	kamera	16843	491956	39584
	WA	16431	589060	61736
60 menit	kamera	16843	491956	39584
	WA	19019	566504	60856
90 menit	kamera	-	-	-
	WA	19019	566504	61068
120 menit	kamera	-	-	-
	WA	19019	567544	58792
150 menit	kamera	-	-	-
	WA	19019	567544	54548
180 menit	kamera	-	-	-
	WA	19019	567544	54536
210 menit	kamera	-	-	-
	WA	19019	567544	51624
240 menit	kamera			
	WA			

Hasil skenario 2 nomor 5 pada Tabel 5.18 menunjukkan hasil aplikasi kamera dan WhatsApp bertahan 30 hingga 60 menit.

Tabel 5.18. Hasil pengujian ketahanan memori skenario 2 nomor 5

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	25675	491240	45456
	WA	25269	595988	74308
30 menit	kamera	25675	491208	38688
	WA	25269	594948	58660
60 menit	kamera			
	WA			

Skenario ketiga yaitu membuka aplikasi kamera dan WhatsApp bersamaan dengan tambahan aplikasi lain, dipilih aplikasi YouTube dan browser. Tabel 5.19 hingga 5.23 merupakan hasil pengujian skenario ketiga. Hasil skenario 3 nomor 1 pada Tabel 5.19 menunjukkan hasil aplikasi kamera bertahan 0 hingga 30 menit, lalu apliasi WhatsApp bertahan 150 hingga 180 menit.

Tabel 5.19. Hasil pengujian ketahanan memori skenario 3 nomor 1

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	30739	492480	38084
	WA	31540	593260	61356
	youtube	32400	612668	71752
	browser	29549	556372	64404
30 menit	kamera	-	-	-
	WA	31540	591180	58532
	youtube	32400	600224	59904
	browser	29549	554164	63140
60 menit	kamera	-	-	-
	WA	31540	591180	53828
	youtube	-	-	-
	browser	29549	554164	57728
	kamera	-	-	-

90 menit	WA	31540	591180	50472
	youtube	-	-	-
	browser	29549	554164	57728
120 menit	kamera	-	-	-
	WA	31540	591180	50704
	youtube	-	-	-
	browser	29549	554164	57728
150 menit	kamera	-	-	-
	WA	31540	591180	48908
	youtube	-	-	-
	browser	29549	554164	55936
180 menit	kamera	-	-	-
	WA	-	-	-
	youtube	-	-	-
	browser	29549	554164	55996

Hasil skenario 3 nomor 2 pada Tabel 5.20 menunjukkan hasil aplikasi kamera bertahan 30 hingga 60 menit, lalu aplikasi WhatsApp mengalami perubahan PID pada waktu antara 30 dan 60 menit.

Tabel 5.20. Hasil pengujian ketahanan memori skenario 3 nomor 2

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	8560	490672	41256
	WA	9352	593140	68984
	youtube	10056	609532	68716
	browser	9766	523168	48692
30 menit	kamera	8560	490672	40416
	WA	9352	590020	62096
	youtube	10056	599152	59776
	browser	9766	519920	47912
	kamera	-	-	-

60 menit	WA	12446	567708	60948
	youtube	10056	599152	58656
	browser	9766	519920	45488
90 menit	kamera	-	-	-
	WA	12446	567708	58964
	youtube	10056	599152	58560
	browser	9766	519920	45488
120 menit	kamera	-	-	-
	WA	12446	567708	55284
	youtube	10056	599224	59600
	browser	9766	519920	45488
150 menit	kamera	-	-	-
	WA	12446	567708	55012
	youtube	10056	599152	58552
	browser	9766	519920	45488
180 menit	kamera	-	-	-
	WA	12446	567708	567708
	youtube	10056	599152	59408
	browser	9766	519920	45488
210 menit	kamera	-	-	-
	WA	12446	575640	52956
	youtube	-	-	-
	browser	9766	519920	44164

Hasil skenario 3 nomor 3 pada Tabel 5.21 menunjukkan hasil aplikasi kamera dan WhatsApp bertahan 0 hingga 30 menit, namun aplikasi WhatsApp terlacak dengan PID baru pada waktu antara 30 dan 60 menit, dan 90 dan 120 menit.

Tabel 5.21. Hasil pengujian ketahanan memori skenario 3 nomor 3

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	22608	506712	42904

	WA	23545	593320	62832
	youtube	24195	612748	73628
	browser	23952	522096	46940
30 menit	kamera	-	-	-
	WA	-	-	-
	youtube	24195	599272	49896
	browser	23952	518848	41196
60 menit	kamera	-	-	-
	WA	26058	566660	53720
	youtube	24195	599176	48756
	browser	23952	518848	40704
90 menit	kamera	-	-	-
	WA	-	-	-
	youtube	-	-	-
	browser	23952	518848	37832
120 menit	kamera	-	-	-
	WA	28319	566668	58976
	youtube	27999	588536	48576
	browser	23952	518848	37832
150 menit	kamera			
	WA			
	youtube			
	browser			

Hasil skenario 3 nomor 4 pada Tabel 5.22 menunjukkan hasil aplikasi kamera dan WhatsApp bertahan 0 hingga 30 menit, namun aplikasi WhatsApp terlacak dengan PID baru pada waktu antara 30 dan 60 menit.

Tabel 5.22. Hasil pengujian ketahanan memori skenario 3 nomor 4

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	29383	490920	40756

	WA	30169	593292	61540
	youtube	30763	614808	70444
	browser	30552	520280	47184
30 menit	kamera	29383	490920	38212
	WA	30169	590172	57092
	youtube	30763	600420	55996
	browser	30552	519112	42660
60 menit	kamera	-	-	-
	WA	32465	566664	59748
	youtube	-	-	-
	browser	30552	519112	40672
90 menit	kamera	-	-	-
	WA	32465	566664	58588
	youtube	-	-	-
	browser	30552	519112	40672
120 menit	kamera	-	-	-
	WA	32465	567704	53624
	youtube	-	-	-
	browser	30552	519112	40560
150 menit	kamera	-	-	-
	WA	32465	567704	53628
	youtube	-	-	-
	browser	30552	519112	40256
180 menit	kamera	-	-	-
	WA	32465	567832	50648
	youtube	-	-	-
	browser	30552	519112	40300
210 menit	kamera	-	-	-
	WA	-	-	-
	youtube	5408	588572	62224

	browser	30552	519112	40348

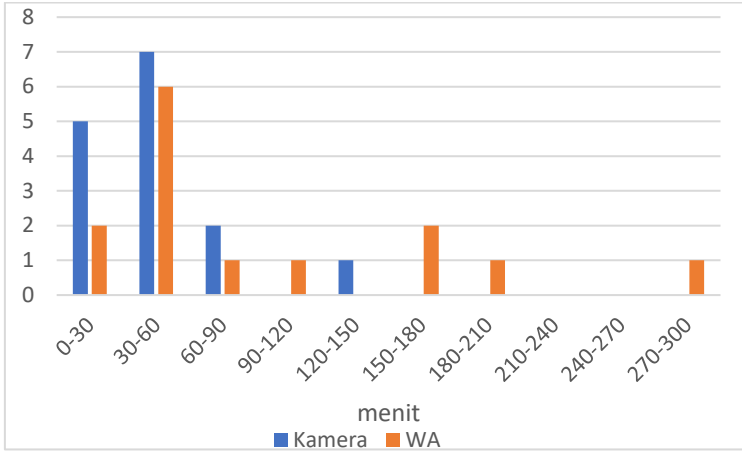
Hasil skenario 3 nomor 5 pada Tabel 5.23 menunjukkan hasil aplikasi kamera bertahan 30 hingga 60 menit, lalu aplikasi WhatsApp bertahan 90 hingga 120 menit dan mengalami perubahan PID

Tabel 5.23. Hasil pengujian ketahanan memori skenario 3 nomor 5

waktu	aplikasi	PID	VSIZE	RSS
0 menit	kamera	6057	491192	40400
	WA	6845	591260	62456
	youtube	7519	599148	68160
	browser	7236	518508	49796
30 menit	kamera	6057	491192	40400
	WA	6845	591260	62456
	youtube	7519	599148	68160
	browser	7236	518508	49796
60 menit	kamera	-	-	-
	WA	6845	590220	60148
	youtube	7519	599152	64056
	browser	7236	518380	48720
90 menit	kamera	-	-	-
	WA	6845	590220	56256
	youtube	-	-	-
	browser	7236	518380	41152
120 menit	kamera	-	-	-
	WA	11170	567712	60256
	youtube	-	-	-
	browser	7236	518380	41120
150 menit	kamera	-	-	-
	WA	11170	567712	59912
	youtube	-	-	-

	browser	7236	518380	41132
180 menit	kamera	-	-	-
	WA	11170	567712	51276
	youtube	-	-	-
	browser	7236	518380	41096
210 menit	kamera	-	-	-
	WA	11170	567712	51000
	youtube	-	-	-
	browser	7236	518396	41016
240 menit	kamera	-	-	-
	WA	11170	567840	45784
	youtube	-	-	-
	browser	7236	518396	38188
270 menit	kamera	-	-	-
	WA	11170	567840	46812
	youtube	-	-	-
	browser	7236	518396	38184

Dari hasil pengujian ketahanan memori ini dapat diketahui bahwa proses *com.sec.android.app.camera* terhapus dari memori dengan waktu tanpa aktivitas selama 0 hingga 30 menit sebanyak lima kali, 30 hingga 60 menit sebanyak tujuh kali, 60 hingga 90 menit sebanyak dua kali, dan 120 hingga 150 menit sebanyak satu kali. Lalu untuk proses *com.whatsapp* terhapus dari memori dengan waktu tanpa aktivitas selama 0 hingga 30 menit sebanyak dua kali, 30 hingga 60 menit sebanyak enam kali, 60 hingga 90 menit sebanyak satu kali, 90 hingga 120 menit sebanyak satu kali, 150 hingga 180 menit sebanyak dua kali, 180 hingga 210 menit sebanyak satu kali, dan 270 hingga 300 menit sebanyak satu kali. Gambar 5.2 menunjukkan rekapitulasi hasil dari ketiga skenario pengujian yang dilakukan



Gambar 5.3. Rekapitulasi hasil pengujian ketahanan memori

Hal ini menunjukkan bahwa setiap aplikasi diperlakukan berbeda oleh sistem operasi Android dalam mengalokasikan memori untuk proses yang sedang berjalan. Beberapa faktor lain bisa saja berupa ukuran memori, versi aplikasi, versi sistem operasi Android, dan aktivitas aplikasi lain yang sedang berjalan.

5.3.4 Analisis Aktivitas Jaringan

Aktivitas jaringan bisa saja terekam pada memori *volatile* seperti sedang tersambung ke alamat IP mana saja dan koneksi apa saja yang sedang terjadi, namun pada proses penyelidikan forensik barang bukti tidak boleh terhubung ke jaringan manapun. Hal ini menjadi halangan karena fitur yang tersedia tidak dapat menampilkan aktivitas jaringan yang terjadi di masa lampau, pada perintah *netstat* seharusnya menampilkan koneksi keluar jaringan yang sedang terjadi, namun karena tidak ada aktivitas jaringan yang terjadi pada saat analisis perintah *netstat* tidak menampilkan apa-apa. Hasil yang serupa juga didapatkan dari perintah *cat* untuk menampilkan ARP *Table* dan *routing cache* pada *file /proc/net/arp* dan */proc/net/route* dikarenakan perangkat sudah tidak tersambung ke jaringan apapun.

```
shell@android:/ $ netstat
Proto Recv-Q Send-Q Local Address           Foreign Address
State
```

Kode 5.19. Hasil keluaran perintah *netstat*

```
shell@android:/ $ cat /proc/net/arp
IP address      HW type    Flags      HW address          Mask
Device
```

Kode 5.20. Hasil keluaran ARP *table*

```
shell@android:/ $ cat /proc/net/route
Iface   Destination     Gateway Flags   RefCnt  Use
        Metric  Mask            MTU     Window  IRTT
```

Kode 5.21. Hasil keluaran *routing cache*

5.3.5 Analisis Aktivitas Aplikasi

Memori *volatile* menyimpan sesuatu yang tidak disimpan pada media penyimpanan perangkat, yaitu aktivitas apa saja yang terjadi pada perangkat. Aktivitas ini terekam pada *log* yang menunjukkan keadaan sistem yang terekam pada luaran perintah *dumpsys*. Perintah ini sebenarnya digunakan untuk tujuan *debugging*, namun karena juga menyajikan informasi tentang apa saja yang dilakukan pada perangkat dapat dijadikan sebagai metode analisis memori.

```
nero@ubuntu:~$ adb shell dumpsys
Currently running services:
  CustomFrequencyManagerService
  DirEncryptService
  FMPlayer
  SYSSCOPE
  SecTVOutService
  SurfaceFlinger
  TvoutService_C
  accessibility
  account
  activity
.
.
.
```

Kode 5.22. Hasil keluaran perintah *dumpsys*

Dumpsys mengeluarkan daftar *services* apa saja yang sedang berjalan pada sistem perangkat beserta isi dari *services* tersebut. Untuk mengetahui aktivitas atau pemakaian aplikasi pada perangkat, analisis ini berfokus pada salah satu *service* yaitu *usagestats*. *Service* ini menyimpan data penggunaan aplikasi dari beberapa hari ke belakang, Kode 5.23 menunjukkan penggunaan aplikasi WhatsApp pada tanggal 16 Juli 2020, di mana aplikasi WhatsApp dibuka 20 kali dengan total durasi waktu 629008 ms atau 10 menit 29 detik.

```
Date: 20200716
.
.
.
  com.whatsapp: 20 times, 629008 ms
    com.whatsapp.registration.RegisterPhone: 12 starts, 0-
250ms=1,  250-500ms=8,  500-750ms=1,  750-1000ms=1,  1500-
2000ms=1
    com.whatsapp.gallerypicker.MediaPreviewActivity:       1
starts, 250-500ms=1
    com.whatsapp.Main: 16 starts
    com.whatsapp.backup.google.RestoreFromBackupActivity:  1
starts, 750-1000ms=1
    com.whatsapp.registration.VerifySms:  1  starts,  1500-
2000ms=1
    com.whatsapp.Conversation: 4 starts, 750-1000ms=1, 1000-
1500ms=1
    com.whatsapp.registration.EULA: 16 starts, 250-500ms=1,
500-750ms=2,  750-1000ms=1,  1000-1500ms=1,  1500-2000ms=1,
2000-3000ms=5, 3000-4000ms=3, >=5000ms=1
    com.whatsapp.HomeActivity: 6 starts, 500-750ms=1, 2000-
3000ms=1, 3000-4000ms=1
    com.whatsapp.ContactPicker: 6 starts, 500-750ms=1, 1500-
2000ms=1, 2000-3000ms=1
    com.whatsapp.mediaview.MediaViewActivity: 1 starts, 0-
250ms=1
    com.whatsapp.registration.RegisterName:    2    starts,
>=5000ms=1
.
.
.
```

Kode 5.23. Aktivitas aplikasi WhatsApp tanggal 16 Juli 2020

Hasil rekaman aktivitas yang didapatkan pada aplikasi WhatsApp beragam mulai dari registrasi hingga percakapan,

Tabel 5.24 menunjukkan aktivitas apa saja yang dapat dilacak pada penggunaan aplikasi WhatsApp pada tanggal 16 Juli 2020.

Tabel 5.24. Aktivitas WhatsApp

Aktivitas	Deskripsi
registration.RegisterPhone	Pendaftaran nomor telepon
gallerypicker.MediaPreview Activity	Memilih media yang berasal dari *gallery*
Main	Menu utama
backup.google.RestoreFrom BackupActivity	*Restore backup* dari Google Drive
registration.VerifySms	Verifikasi pendaftaran melalui SMS
Conversation	Percakapan
registration.EULA	Persetujuan pendaftaran
HomeActivity	Menu *home*
ContactPicker	Memilih kontak
mediaview.MediaViewActi vity	Membuka media pada percakapan
registration.RegisterName	Pendaftaran nama

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dibahas mengenai kesimpulan dari semua proses yang telah dilakukan dan saran yang dapat diberikan untuk pengembangan yang lebih baik.

6.1. Kesimpulan

Berdasarkan hasil akuisisi dan analisis *live forensic* dari data memori *volatile* perangkat *mobile* Android, berikut adalah beberapa kesimpulan yang didapatkan dari Tugas Akhir ini :

1. Metode akusisi memori *volatile* menggunakan *tools* LiME berhasil diimplementasikan pada perangkat target Galaxy S3, sedangkan metode menggunakan AMExtractor tidak bisa diimplementasikan karena masalah persiapan *tools*.
2. Untuk menjalankan modul LiME pada sistem perangkat yang sedang berjalan dibutuhkan hak akses *root*. Proses *rooting* harus menggunakan metode yang tidak melibatkan *reboot* dan membuka *bootloader*, yang mana tidak selalu tersedia untuk setiap jenis perangkat.
3. Proses untuk menganalisa memori *volatile* menggunakan Volatility tidak berjalan dengan lancar. Pada penelitian ini terdapat masalah yang masih menjadi *Open Issue* pada halaman Github Volatility yang mana para pengembang dari *framework* ini belum dapat menyelesaikannya.
4. Proses analisa memori *volatile* menggunakan ADB dibuat menyerupai penggunaan *plugin* pada Volatility untuk menggambarkan hasil seperti apa yang diharapkan. Melalui analisis didapatkan informasi bahwa perangkat tersebut digunakan yaitu membuka aplikasi kamera dan WhatsApp, dengan urutan membuka kamera terlebih dahulu lalu membuka WhatsApp, dan terlacak pemakaian WhatsApp digunakan untuk apa saja.
5. Perubahan memori akibat proses *rooting* yaitu menghilangkan jejak aktivitas penggunaan aplikasi kamera dan WhatsApp pada perintah *ps* dikarenakan memori membutuhkan *space* lebih saat menjalankan proses *rooting*

sehingga menghapus jejak aktivitas penggunaan aplikasi kamera dan WhatsApp untuk dialokasikan ke aplikasi lain. Setelah aplikasi kamera dan WhatsApp dibuka kembali terdapat perubahan VSIZE rata-rata sebesar 11932 untuk kamera dan 2372 untuk WhatsApp, dan perubahan RSS rata-rata sebesar 2351 untuk kamera dan 996 untuk WhatsApp.

6.2. Saran

Berdasarkan hasil akuisisi dan analisis *live forensic* dari data memori *volatile* perangkat *mobile* Android, berikut beberapa saran untuk menyempurnakan penelitian yang akan datang supaya didapatkan hasil yang lebih baik. Saran yang dapat disampaikan untuk penelitian yang akan datang yaitu:

1. Perangkat yang digunakan sebagai barang bukti bisa menggunakan perangkat Android merk lain yang mempublikasikan *source code* kernel produknya, seperti Sony dan HTC.
2. Proses analisis yang dilakukan pada memori dapat diperluas seperti membuat *plugin* sendiri untuk menganalisa hal yang lebih spesifik seperti analisis DalvikVM.
3. Skenario yang dilakukan dapat diberi improvisasi aktivitas yang lebih kompleks, maka dapat memberikan gambaran dan tantangan untuk menemukan petunjuk dan menyelesaikan teka-teki penyelidikan.

DAFTAR PUSTAKA

[1] "Smartphone ownership in advanced economies higher than in emerging | Pew Research Center," Pew Research Center, 5 2 2019. [Online]. Available: https://www.pewresearch.org/global/2019/02/05/smartphone-ownership-is-growing-rapidly-around-the-world-but-not-always-equally/pg_global-technology-use-2018_2019-02-05_0-01/. [Accessed 30 10 2019].

[2] "Number of smartphone users worldwide 2014-2020," Statista, 2019. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Accessed 30 10 2019].

[3] "Hampir 80 Persen Smartphone di Indonesia adalah Android. Sisanya Jadi Rebutan 8 Vendor," 12 2016. [Online]. Available: https://www.et.co.id/2016/12/total-pengguna-android-indonesia.html. [Accessed 30 10 2019].

[4] N. R. Mistry and M. S. Dahiya, "Signature based volatile memory forensics: a detection based approach for analyzing sophisticated cyber attacks," 2018.

[5] H. Macht, "Live Memory Forensics on Android with Volatility," 2013.

[6] J. T. Sylve, "Android Memory Capture and Applications for Security and Privacy," 2011.

[7] H. Yang, Z. Jianwei, H. Liu and W. Liu, "A Tool for Volatile Memory Acquisition from Android Devices," 2018.

[8] P. Wächter, "Practical Infeasibility of Android Smartphone Live Forensics Applicability Constraints of LiME and Volatility," 2015.

[9] O. Skulkin, D. Tindall and R. Tamma, Learning Android Forensics - Second Edition, 2018.

[10] A. Agarwal, M. Gupta, S. Gupta and S. C. Gupta, "Systematic Digital Forensic Investigation Model,"

International Journal of Computer Science and Security, vol. 5, no. 1, 2011.

[11] "Live Data Forensics - or - Why volatile data can be crucial for your cases," Council of Europe, 2013. [Online]. Available: https://www.coe.int/en/web/octopus/blog/-/blogs/live-data-forensics-or-why-volatile-data-can-be-crucial-for-your-cases. [Accessed 28 11 2019].

[12] EC-Council, " Module 4 Digital Evidence," *Computer Hacking Forensics Investigator*, vol. 4, 2006.

[13] B. Kirthika, S. Prabhu and S. Visalakhsi, "Android Operating System: A Review," *International Journal of Trend in Research and Development*, vol. 2, 2015.

[14] "Kernel | Android Open Source Project," Google, [Online]. Available: https://source.android.com/devices/architecture/kernel. [Accessed 29 10 2019].

[15] L. Torvalds, "Linux: a Portable Operating System," 1997.

[16] A. Hoog, Android Forensics: Investigation, Analysis and Mobile Security for Google Android, Elvesier, 2011.

BIODATA PENULIS

Penulis lahir di Surabaya pada tanggal 01 Agustus 1998. Merupakan anak ketiga dari tiga bersaudara. Penulis telah menempuh beberapa pendidikan formal yaitu; SDN Dr. Soetomo V Surabaya, SMPN 4 Surabaya, dan SMAN 2 Surabaya.

Pada Tahun 2016 pasca kelulusan SMA, penulis melanjutkan pendidikan di Jurusan Sistem Informasi Fakultas Teknologi Elektro dan Informatika Cerdas – Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 05211641000096. Selama menjadi mahasiswa, penulis mengikuti berbagai kegiatan kemahasiswaan seperti pada tahun kedua pernah menjabat sebagai Staff Biro Technology Development HMSI FTIK ITS dan pada tahun ketiga menjabat sebagai Staff Ahli Biro Technology Development HMSI FTIK ITS. Di bidang akademik, penulis aktif menjadi asisten praktikum pada mata kuliah Pengantar Sistem Operasi, Desain dan Manajemen Jaringan Komputer, dan Proteksi Aset Informasi. Selain itu, penulis juga mengikuti kegiatan IT Club yaitu Information System Geek Club (ISGC) yang berfokus pada bidang keamanan, jaringan, dan *open source*.

Pada tahun keempat, karena penulis memiliki ketertarikan di bidang forensika digital, maka penulis mengambil laboratorium bidang minat Infrastruktur dan Keamanan Teknologi Informasi (IKTI). Penulis dapat dihubungi melalui *email* di nevadanero89@gmail.com.