Frida Framework Implementation for Workflow Manipulation in Android Applications
[Implementasi Frida Framework untuk Manipulasi Alur Kerja pada Aplikasi Android]

Aldo Reghan Ramadhan[1)], Arif Senja Fitriani [*,2)] , Mochamad Alfan Rosid[*,3)], Cindy Taurusta [*,4)]

[1,2,3,4)]Program Studi Teknik Informatika, Universitas Muhammadiyah Sidoarjo, Indonesia

*Email Penulis Korespondensi : asfjim@umsida.ac.id

Abstract. *Enhancing security on Android devices has posed challenges for security researchers. Root bypass is a commonly employed method to evade detection by security mechanisms. In this research, the author explains the utilization of Frida, a dynamic instrumentation framework, for performing root bypass on Android devices. By leveraging Frida's capabilities for runtime code interception and modification, the author can alter the behavior of applications attempting to detect root presence. A series of experiments were conducted using Frida, successfully bypassing common root detection mechanisms. The results demonstrate Frida's potential as an effective tool for root bypass and security testing on Android devices. This research provides further insights into the use of Frida in the context of Android device security.*

Keywords - *Root, Android devices, Frida Framework, Dynamic Instrumentation Framework, Android Device Security*

Abstrak. *Peningkatan keamanan pada perangkat Android telah menjadi tantangan bagi para peneliti keamanan. Bypass root adalah salah satu metode yang sering digunakan untuk menghindari deteksi oleh mekanisme keamanan. Dalam penelitian ini, menjelaskan penggunaan Frida, sebuah framework dynamic instrumentation, untuk melakukan bypass root pada perangkat Android. Dengan memanfaatkan kemampuan Frida untuk melakukan intersepsi dan modifikasi kode pada saat runtime, dapat mengubah perilaku aplikasi yang mencoba mendeteksi keberadaan root. Penulis melakukan serangkaian percobaan menggunakan Frida dan berhasil melewati mekanisme deteksi root yang umum digunakan. Hasil penelitian ini menunjukkan potensi Frida sebagai alat yang efektif dalam melakukan bypass root dan serangkaian pengujian keamanan pada perangkat Android. Penelitian ini memberikan pemahaman lebih lanjut tentang penggunaan Frida dalam konteks keamanan perangkat Android.*

Kata Kunci - *Root, Android devices, Frida Framework, Dynamic Instrumentation Framework, Android Device Security*

I. PENDAHULUAN

Perangkat mobile selular adalah suatu hal sudah menjadi kebutuhan bagi banyak masyarakat, yang berawal dari hanya sebatas melakukan telepon dan mengirim pesan antar perangkat hingga melakukan transaksi jual beli, dan seluruh aktivitas itu dikerjakan hanya dengan sebuah perangkat smartphone.

Android merupakan salah satu sistem operasi yang paling banyak digunakan di perangkat smartphone, berdasarkan data dari businessofapps.com data terakhir android global market share sebesar 72,1% dan release Asosiasi Pengguna Jasa Internet Indonesia (APJII) 2020 mengungkapkan kepemilikan smartphone dan tablet pribadi lebih banyak daripada pengguna laptop atau PC. Dari pembuktian ini bahwa, android menjadi sistem operasi seluler yang sangat populer melebihi Iphone Operating System (IOS). Pada faktor dominan yang terdapat di android, sisi keamanan juga menjadi issue yang menjadi perhatian. Pada topik mobile hacking adalah suatu aktivitas peretasan yang menargetkan pada perangkat seluler. Menurut RSA Security, 60% dari semua serangan siber di dunia dilakukan melalui perangkat seluler. Sekitar 80% menyerang melalui aplikasi seluler. Aplikasi memberi akses penuh terhadap perangkat yang telah diretas.

Dalam sebuah aplikasi android terdapat sebuah celah keamanan yang dimana itu dari bahasa pemrograman yang digunakan, bahasa pemrograman yang populer adalah kotlin dan java, kedua bahasa pemrograman ini memiliki beberapa celah keamanan antara lain : Common Weakness Enumeration (CWE-20): Improper input validation, CWE-269: Improper privilege management.

Obfuscation code adalah sebuah teknik untuk menyamarkan sebuah code namun masih menjaga fungsionalitas nya yang bertujuan untuk menyulitkan pemahaman oleh manusia. Teknik obfuscation yang kompleks mengalahkan model PetaDroid deteksi malware Android, yang mengakibatkan deteksi palsu[1]. Obfuscation adalah

metode untuk mengubah kode sedemikian rupa sehingga menyembunyikan niat dari pemrogram namun tetap memiliki ekivalensi semantik dengan basis kode asli[2]. Pengembang aplikasi Android sering menggunakan teknik obfuscation untuk melindungi logika bisnis dan algoritma inti dalam aplikasi mereka dari serangan reverse engineering[3].

Reverse engineering adalah sebuah proses untuk menganalisis, memahami, dan mencari tahu cara aplikasi tersebut berjalan tanpa mengetahui source code aslinya. Reverse engineering menyediakan kode sumber aplikasi, pandangan wawasan terhadap arsitektur, dan ketergantungan pihak ketiga[4]. Reverse engineering dalam perangkat lunak memungkinkan untuk mengubah file biner yang dapat dibaca oleh mesin menjadi file yang dapat dibaca oleh manusia, seperti yang terjadi pada file DEX[5]. Melakukan reverse engineering pada aplikasi Android dan mengekstrak fitur serta melakukan analisis statis dari mereka tanpa harus menjalankannya. Metode ini melibatkan pemeriksaan isi dua file: AndroidManifest.xml dan classes.dex serta bekerja pada file dengan ekstensi .apk[6].

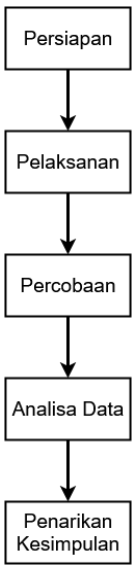
Root akses adalah sebuah hak akses penuh padda sebuah sistem operasi berbasis UNIX, hak akses ini memungkinkan kita untuk mengakses dan memodifikasi file system sistem operasi tersebut. Rooting adalah proses mendapatkan akses root pada perangkat Android. Untuk dapat mengakses dan menjelajahi sistem Android secara bebas, serta memanfaatkan fungsionalitas penuh Android, pengguna bersedia melakukan rooting pada perangkat mereka[7].

Dynamic Instrumentation Application Testing adalah sebuah proses pengujian perangkat lunak yang bertujuan untuk melakukan analisis dan memeriksa alur aplikasi secara realtime selama runtime. DAST dapat melakukan analisa saat aplikasi sedang dijalankan, dengan melakukan injeksi pada aplikasi, seperti memasukan input berbahaya untuk mengidentifikasi apakah aplikasi akan menampilkan kesalahan sesuai dengan input yang dilakukan.[8]. DAST mengimplementasikan black box testing terhadap perilaku runtime sambil menjalankannya dari luar ke dalam[9].

Frida adalah sebuah framework open-source yang digunakan untuk melakukan dynamic instrumentation (pemantauan dan modifikasi pada runtime) pada aplikasi di berbagai platform seperti Android, iOS, Windows, macOS, dan Linux. Tujuan utama dari Frida adalah memberikan fleksibilitas dan kontrol yang tinggi kepada pengembang dan peneliti dalam menganalisis dan memodifikasi perilaku aplikasi pada level kode yang lebih dalam. Pada artikel ini akan berfokus pada manipulasi alur kerja library anti root dan function yang memproses sebuah nama dan harga, untuk dapat melakukan transaksi sebuah item pada aplikasi android.

II. METODE

Pada metode ini memberikan pemaparan tentang alur kerja penelitan dengan penjelasan singkat didalamnya.Alur metode penelitian diawali dengan menentukan versi frida dan identifikasi masalah yang terjadi, kemudian dilanjutkan dengan studi literatur, Analsis Kebutuhan, Analisis source code aplikasi , Perancangan exploit, Implementasi, Pengujian dan analisis, dan kesimpulan. Alur, metode penelitian digambarkan pada gambar 1 dan disertai penjelasan singkat pada tiap segmen alur.



Gambar 1. Alur metode penelitian

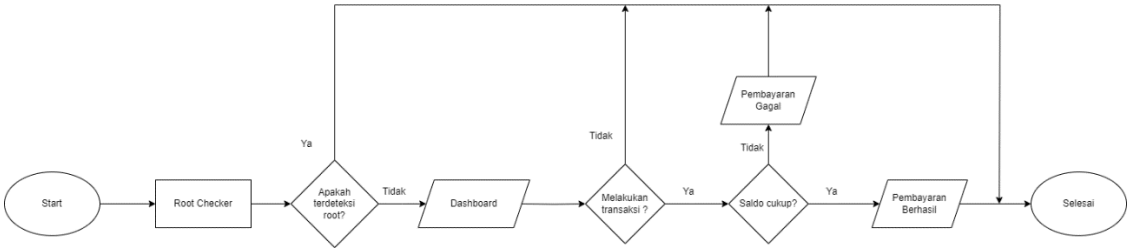
A. Tahapan persiapan

Pada tahap ini menentukan versi frida yang sesuai dengan os android serta mencari berbagai sumber referensi yang nantinya akan digunakan sebagai acuan untuk melakukan penelitian ini. Juga pada tahapan ini mengumpulkan berbagai jenis artikel yang berhubungan dengan reverse engineering pada android dan penggunaan frida framework yang beredar di internet.

B. Tahapan Pelaksanaan

Pada tahap ini peneliti mulai menganalisis sebuah aplikasi mobile yang akan di eksploitasi. Dan dilanjutkan dengan langkah-langkah sebagai berikut :

1. Menganalisis normal flow pada aplikasi



Gambar 2. Flowchart Normal Flow Aplikasi

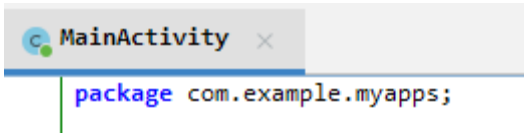
Berikut penjelasan normal flow dari aplikasi yang digambarkan oleh flowchart diatas :

1. Program melakukan pengecekan menggunakan root checker
2. Jika program mendeteksi perangkat dalam kondisi root maka program akan berhenti.
3. Jika program tidak mendeteksi root makan akan langsung diarahkan ke dashboard
4. Ketika masuk ke dalam dashboard, user akan diberikan sebuah pilihan yaitu melakukan transaki atau tidak.
5. Jika user melakukan transaksi program akan melakukan pengecekan saldo, apakah saldo cukup atau tidak.
6. Jika saldo tidak mencukupi akan memberikan respon “Pembayaran Gagal” jika mencukupi “Pembayaran Berhasil”.

2. Menganalisis source code hasil reverse engineering dari aplikasi yang dibuat

Pada tahapan berikut penulis melakukan beberapa hal yang nantinya akan mendukung penulis untuk melakukan tahapan percobaan seperti :

a) Identifikasi package



Gambar 3. package dari aplikasi

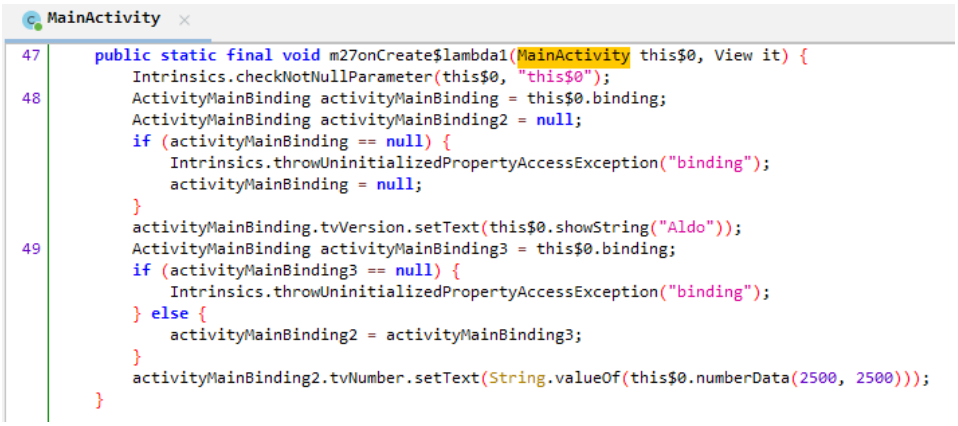
Pada tahapan ini penulis mengidentifikasi nama package di *MainActivity* dari source code hasil *reverse engineering*.

b) Identifikasi library anti root

Gambar 4. Function yang memproses anti root

Pada tahapan ini penulis melakukan identifikasi *library anti root* pada MainActivity : line 22-24 (memanggil object pada library rootbeer), line 25-28(membuat sebuah alert dialog). Pembahasan utama terdapat pada line 29-35 dimana dilakukan implementasi anti root pada aplikasi android.

c) Identifikasi function



```
47    public static final void m27onCreate$lambda1(MainActivity this$0, View it) {
          Intrinsics.checkNotNullParameter(this$0, "this$0");
48        ActivityMainBinding activityMainBinding = this$0.binding;
          ActivityMainBinding activityMainBinding2 = null;
          if (activityMainBinding == null) {
              Intrinsics.throwUninitializedPropertyAccessException("binding");
              activityMainBinding = null;
          }
          activityMainBinding.tvVersion.setText(this$0.showString("Aldo"));
49        ActivityMainBinding activityMainBinding3 = this$0.binding;
          if (activityMainBinding3 == null) {
              Intrinsics.throwUninitializedPropertyAccessException("binding");
          } else {
              activityMainBinding2 = activityMainBinding3;
          }
          activityMainBinding2.tvNumber.setText(String.valueOf(this$0.numberData(2500, 2500)));
      }
```

Gambar 5. Function yang memproses nama dan harga

Pada tahapan ini penulis melakukan identifikasi function yang memproses nama(Aldo) dan harga(5000), yang diterapkan pada line 48-49.

III. HASIL DAN PEMBAHASAN

Pada tahap hasil dan pembahasan ini berfokus pada pembuatan dan pengujian *exploit* yang telah dibuat berdasarkan hasil analisis dari tahapan sebelumnya :

a) Flowchart alur manipulasi

b) Menjalankan Frida server yang terdapat pada emulator

```
E:\> adb root
E:\> adb shell
```

```
E:\>adb root
adbd is already running as root

E:\>adb shell
generic_x86:/ # cd /data/local/tmp
generic_x86:/data/local/tmp # ./frida-server
```

Gambar 6. Command untuk mengakses emulator

Pada command *adb root* dan *adb shell* Gambar 5, dimana untuk adb root digunakan untuk menjalankan akses root pada emulator dengan ditandai status “adbd is already running as root”. Pada command adb shell untuk mengakses root shell pada emulator dengan terakses nya generic_x86 yang merupakan arsitektur pada emulator.

c) Melihat list aplikasi

```
E:\>frida-ps -U
```

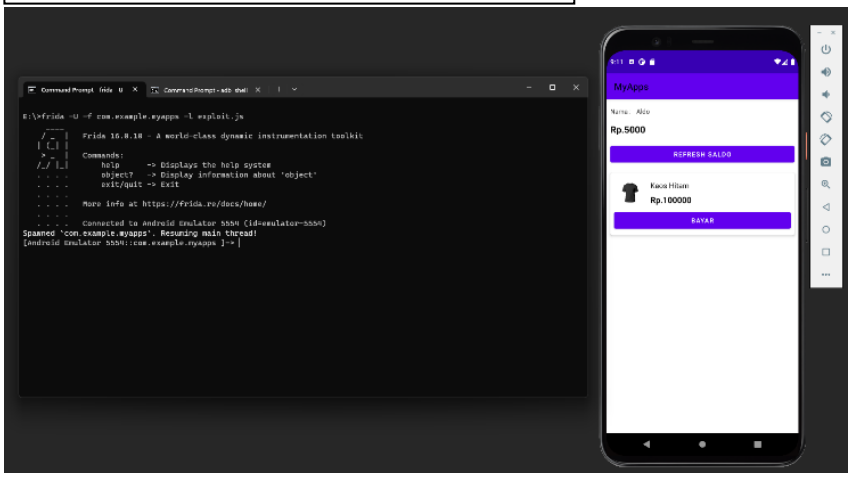
```
E:\>frida-ps -U
 PID  Name
----  ------------------------------------------------------
8339  Calendar
8387  Clock
2303  Gallery
8205  MyApps
1865  Phone
```

Gambar 7. Command untuk melihat aplikasi yang berjalan

Pada tahap ini penulis mengidentifikasi aplikasi “MyApps” apakah sudah terinstall pada Gambar 6, dengan command frida-ps -U.

d) Menjalankan exploit

```
E:\> frida -U com.example.myapps -l exploit.js
```



Gambar 8. Script exploit berhasil di jalankan

Pada Gambar 7 secara realtime ditampilkan aplikasi dan terminal untuk membuktikan status exploit berhasil dijalankan. Aplikasi akan berubah alur nya, pada Gambar 3 dimana menunjukan kondisi awal anti root yang akan menghentikan aplikasi jika perangkat terdeteksi root. Pada terminal menjalankan frida dengan command *frida -U com.example.myapps -l exploit.js* dimana hasil command anti root ter-bypass dan berhasil masuk ke dalam dashboard aplikasi, sehingga penulis dapat melakukan manipulasi nama dan harga pada Gambar 5.

VII. SIMPULAN

Artikel ini menginvestigasi penggunaan Frida Framework dalam memanipulasi alur kerja pada aplikasi Android. Hasil penelitian menunjukkan potensi besar dari Frida dalam mengubah alur kerja aplikasi secara runtime, yang memberikan fleksibilitas dalam memodifikasi perilaku aplikasi tanpa perlu melakukan perubahan pada kode sumber asli. Dalam skenario pengujian, penelitian ini berhasil memodifikasi alur kerja aplikasi dengan mengintersep fungsi-fungsi kunci dan mengganti perilaku mereka. Selain itu, jurnal ini menyoroti keefektifan Frida dalam mengatasi mekanisme keamanan yang ada, seperti anti-debugging dan enkripsi kode.

Pembahasan Artikel ini menekankan pentingnya pengujian lebih lanjut pada berbagai jenis aplikasi untuk mengukur ketersediaan dan kehandalan Frida Framework. Selain itu, keterbatasan potensi risiko dalam penggunaan Frida juga dibahas, termasuk potensi penyalahgunaan dan dampaknya pada pengalaman pengguna akhir

Dari hasil penelitian dan pembahasan tentang "Implementasi Frida Framework untuk Manipulasi Alur Kerja pada Aplikasi Android", maka dapat diambil kesimpulan bahwa pada Frida Framework merupakan alat yang efektif untuk melakukan dynamic testing, ini dibuktikan hasil uji coba dapat memanipulasi alur kerja seperti input, output dan perilaku dari aplikasi android, versi android terbaru tidak menjamin bahwa aplikasi aman terhadap Frida Framework, karena Frida juga mengikuti perkembangan OS Android, untuk developer penerapan library pada aplikasi yang dibangun harus menggunakan versi latest atau yang terbaru.

UCAPAN TERIMA KASIH

Alhamdulillah, puji dan syukur peneliti panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya. Ucapan terima kasih dari peneliti diberikan kepada pihak-pihak

a. Allah SWT yang selalu melindungi dan memudahkan hamba dalam penyusunan karya ilmiah ini hingga dapat terselesaikan dengan baik.
b. Kedua Orang tua tercinta yang telah memberikan dukungan sehingga penelitian ini dapat berjalan dengan lancar hingga akhir penelitian.
c. Arif Senja Fitriani., M.Kom., selaku Dosen Pembimbing atas bimbingan, saran dan motivasi yang diberikan dalam penyelesaian karya ilmiah ini.
d. Seluruh dosen pengajar, staf dan karyawan di Fakultas Sains Dan Teknologi Universitas Muhammadiyah Sidoarjo yang telah banyak memberikan ilmu, wawasan dan pengalaman kepada penulis.

REFERENSI

[1] W. F. Elsersy, A. Feizollah, and N. B. Anuar, "The rise of obfuscated Android malware and impacts on detection methods," *PeerJ Comput. Sci.*, vol. 8, no. September 2018, 2022, doi: 10.7717/PEERJ-CS.907.

[2] A. You, M. Be, and I. In, "Java Code Obfuscator to Prevent Reverse Engineering," vol. 020004, no. June, 2023.

[3] G. You, G. Kim, S. Han, M. Park, and S. J. Cho, "Deoptfuscator: Defeating Advanced Control-Flow Obfuscation Using Android Runtime (ART)," *IEEE Access*, vol. 10, pp. 61426–61440, 2022, doi: 10.1109/ACCESS.2022.3181373.

[4] S. W. Asher, S. Jan, G. Tsaramirsis, F. Q. Khan, A. Khalil, and M. Obaidullah, "Reverse engineering of mobile banking applications," *Comput. Syst. Sci. Eng.*, vol. 38, no. 3, pp. 265–278, 2021, doi: 10.32604/CSSE.2021.016787.

[5] M. Ziadia, J. Fattahi, M. Mejri, and E. Pricop, "Smali+: An operational semantics for low-level code generated from reverse engineering android applications," *Inf.*, vol. 11, no. 3, 2020, doi: 10.3390/info11030130.

[6] B. Urooj, M. A. Shah, C. Maple, M. K. Abbasi, and S. Riasat, "Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms," *IEEE Access*, vol. 10, no. December 2021, pp. 89031–89050, 2022, doi: 10.1109/ACCESS.2022.3149053.

[7] B. Soewito and A. Suwandaru, "Android sensitive data leakage prevention with rooting detection using Java function hooking," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 5, pp. 1950–1957, 2022, doi: 10.1016/j.jksuci.2020.07.006.

[8] F. A. Alviansyah and E. Ramadhani, "Implementasi Dynamic Application Security Testing pada Aplikasi Berbasis Android," *Automata*, vol. 2, no. 1, pp. 1–6, 2021, [Online]. Available: https://journal.uii.ac.id/AUTOMATA/article/view/17387

[9] J. Li, “Vulnerabilities mapping based on OWASP-SANS: A survey for static application security testing (SAST),” *Ann. Emerg. Technol. Comput.*, vol. 4, no. 3, pp. 1–8, 2020, doi: 10.33166/AETiC.2020.03.001.
[10] M. Sharma, “Review of the Benefits of DAST (Dynamic Application Security Testing) Versus SAST SAST Integration and DAST Reporting,” no. May, pp. 5–8, 2021.
[11] F. O. Sonmez and B. G. Kilic, “Holistic Web Application Security Visualization for Multi-Project and Multi-Phase Dynamic Application Security Test Results,” *IEEE Access*, vol. 9, pp. 25858–25884, 2021, doi: 10.1109/ACCESS.2021.3057044.
[12] Y. Pan, “Interactive application security testing,” *Proc. - 2019 Int. Conf. Smart Grid Electr. Autom. ICSGEA 2019*, vol. 1, pp. 558–561, 2019, doi: 10.1109/ICSGEA.2019.00131.
[13] I. U. Haq and T. A. Khan, “Penetration Frameworks and Development Issues in Secure Mobile Application Development: A Systematic Literature Review,” *IEEE Access*, vol. 9, no. 1, pp. 87806–87825, 2021, doi: 10.1109/ACCESS.2021.3088229.
[14] M. Aydos, Ç. Aldan, E. Coşkun, and A. Soydan, “Security testing of web applications: A systematic mapping of the literature,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 9, pp. 6775–6792, 2022, doi: 10.1016/j.jksuci.2021.09.018.
[15] L. Ardito, R. Coppola, S. Leonardi, M. Morisio, and U. Buy, “Automated Test Selection for Android Apps Based on APK and Activity Classification,” *IEEE Access*, vol. 8, pp. 187648–187670, 2020, doi: 10.1109/ACCESS.2020.3029735.

Conflict of Interest Statement:
The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.