CyberSecurity dan Forensik Digital e-ISSN: 2615-8442
Vol. 6, No. 2, November 2023, hlm. 39-47

Analisis Malware Hummingbad Dan Copycat Pada Android Menggunakan Metode Hybrid

Nurul Qomariah[1], Erick Irawadi Alwi[2], Muhammad Arfah Asis[3]

[1,2,3]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Muslim Indonesia
Email: [1]13020190342@student.umi.ac.id, [2]erick.alwi@umi.ac.id, [3]muh.arfah.asis@umi.ac.id

Abstrak

Perkembangan teknologi yang terus berlanjut dapat menjadi ancaman di dunia maya, terutama dalam hal kejahatan *cyber*. Kemudian tingkat popularitas *smartphone* serta jumlah penggunanya meningkat dari tahun ke tahun. Sementara itu, *smartphone* dengan *platform* android masih menduduki peringkat pertama dalam persentase pengguna tertinggi di dunia. Karena itu, jumlah *malware* dan serangan berbahaya di *platform* android semakin meningkat. Pengembang aplikasi penipuan ini mengeksploitasi kekurangan *platform* android dengan menyuntikkan *malware* sebagai kode sumber ke dalam aplikasi android dan menyebarkannya melalui *blog* atau pasar aplikasi android. Teknik yang digunakan dalam penelitian ini adalah teknik *hybrid* yang menggabungkan metode statis dan dinamis. Penelitian ini menggunakan contoh *malware HummingBad* dan *CopyCat*. Penelitian ini memiliki tujuan untuk melakukan identifikasi dan analisis pada *malware HummingBad* dan *CopyCat* menggunakan metode *hybrid* menggunakan *mobile security framework*. Analisis yang dilakukan pada sampel *malware HummingBad* dan *CopyCat* menunjukkan bahwa sampel *malware HummingBad* memiliki tingkat keamanan 27/100 sedangkan untuk sempel *malware CopyCat* memiliki tingkat keamanan 38/100.

Kata kunci: *android, static, dynamic, analysis hybrid, malware*

Analysis Of Hummingbad And Copycat Malware On Android Using Hybrid Methods

Abstract

The continued development of technology can become a threat in cyberspace, especially in terms of cyber crime. Then the level of popularity of smartphones and the number of users increases from year to year. Meanwhile, smartphones with the android platform are still ranked first in the highest percentage of users in the world. Because of this, the number of malware and malicious attacks on the android platform is increasing. Developers of these deceptive apps exploit the flaws of the android platform by injecting malware as source code into Android apps and spreading it via blogs or android app marketplaces. The technique used in this study is a hybrid technique that combines static and dynamic methods. This study uses the examples of HummingBad and CopyCat malware. This study aims to identify and analyze the hummingbad and copycat malware using a hybrid method using the mobile security framework. The analysis conducted on the HummingBad and CopyCat malware samples shows that the HummingBad malware samples have a security level of 27/100 while the CopyCat malware samples have a security level of 38/100.

***Keywords**: android, static, dynamic, analysis hybrid, malware*

1. PENDAHULUAN

Perkembangan teknologi yang terus berlanjut dapat menjadi ancaman di dunia maya, terutama dalam hal kejahatan cyber (Adam, Alwi and Asad, 2022). Dengan meningkatnya popularitas *smartphone* serta jumlah pengguna meningkat dari tahun ke tahun. Di saat yang sama, *smartphone* dengan *platform* android masih menempati posisi teratas dalam angka pengguna terbesar di dunia. Kemudian jumlah serangan *malware* di *platform* android meningkat. Pengembang aplikasi jahat ini mengeksploitasi celah di *platform* Android dengan menyuntikkan *malware* kode sumber ke dalam aplikasi android dan mempromosikannya melalui log web dan pasar aplikasi android (Tansen and Wahyu Nurdiarto, 2020).

Malware atau *Malicious Software* adalah perangkat lunak yang dapat menyusup ke sistem operasi sehingga dapat merusak sistem operasi, memanfaatkan sumber daya tanpa sepengetahuan pemilik perangkat, bahkan mengumpulkan informasi pribadi untuk dibagikan ke pihak ketiga tanpa persetujuan pengguna. *Malware-malware* baru terus bermunculan seiring dengan perkembangan teknologi, baik dari segi *platform* maupun sistem operasi, dengan memanfaatkan celah keamanan dan kelalaian pengguna. Kebutuhan akan mobilitas membuat masyarakat di seluruh belahan dunia marak menggunakan *gadget* seperti *tablet* dan *smartphone*

dengan berbagai macam sistem operasi(Rusdi, Widiyasono and Sulastri, 2019). Sistem operasi untuk perangkat *smartphone* yang paling banyak digunakan adalah *android* yakni sebanyak 74.82% (Widiyasono, Mubarok and Fatwa MF, 2022).

Malware android merupakan salah satu masalah yang paling banyak dibicarakan dikalangan pengguna saat ini. Pasalnya, orang menghabiskan sebagian besar waktunya untuk berinteraksi didunia maya. *Malware* yang awalnya ditemukan di sistem komputer mulai masuk keranah *smartphone*. *Malware* kini mulai hadir dalam berbagai jenis dan kegunaan (Kusuma, Akbi and Risqiwati, 2020).

Malware Copycat mempengaruhi sekitar 14 juta perangkat diseluruh dunia, melakukan *root* sekitar 8 juta perangkat serta mendapatkan penghasilan sekitar $1,5 juta bagi grup. sedangkan *malware Hummingbad* telah meretas perangkat android dengan aplikasi pihak ketiga yang dipasang diluar *Google Play Store*, yang memungkinkan *malware* dapat mengontrol ponsel dari jarak jauh. Yang mana merupakan salah satu seragan terbesar pada saat itu.

Analisis menggunakan metode *hybrid* menggabungkan keunggulan dari analisis statis dan analisis dinamis dengan cara melakukan pengecekan pada setiap *signature malware* setelah itu memonitor perilaku kode.

Saat ini, ada banyak program jahat termasuk *virus*, *trojan*, *spyware*, dan lainnya. *Malware* ini sangat berbahaya karena dapat memengaruhi fungsi perangkat yang terinfeksi mulai dari ekstraksi data hingga Anda tidak dapat terus menggunakan perangkat tersebut(Sinambela, Pangestu and Feriyanto, 2020).

Penelitian berjudul "Analisis dan deteksi *malware* dengan metode *hybrid analysis* menggunakan *framework* MobSF" yang dilakukan oleh (Tansen and Wahyu Nurdiarto, 2020), dalam penelitiannya menggunakan kerangka kerja MobSF untuk mendeskripsikan properti dan perilaku menggunakan kombinasi analisis statis dan dinamis, atau yang kami sebut sebagai analisis *hybrid* dalam penelitian ini. Analisis yang dilakukan menunjukkan bahwa *Bouncing Golf* dapat mencuri informasi pribadi dan membajak perangkat android yang terinfeksi, sedangkan *Riltok Banking Trojan* dapat mencuri informasi *smartphone* dan mencuri informasi kartu kredit melalui *phishing*.

2. TINJAUAN PUSTAKA

2.1. Malware

Malicious Software Merupakan suatu perangkat lunak berbahaya yang dapat meyusup pada sistem orang lain beserta jaringannya tanpa izin pemilik, *malware* dapat mengganggu sistem orang lain serta mengumpulkan informasi rahasia dan curi file penting disistem tersebut (Tansen dan Wahyu Nurdiarto, 2020)(Pranoto, 2019).

Hal tersebut sangat merugikan bagi korban karena dampak negatif yang ditimbulkan dapat memperlambat kinerja sistem hingga merusak bahkan dapat menghancurkan data penting yang ada pada sistem(Manoppo, Lumenta and Karouw, 2020).

Perangkat lunak berbahaya untuk *platform* android ditiru melalui *google play store* atau diunduh dari internet dan diubah menjadi aplikasi resmi seperti aplikasi galeri dan game. Misalnya, *malware* yang ditemukan di *play store* termasuk *banking trojan*, *adware*, *RAT*, dan banyak lagi. *Adware* adalah *malware* yang secara aktif menyerang pengguna *play store*, naik 36% sejak 2016. Selain itu, *malware banking trojan* juga meningkat sebesar 12% dan *Crypto-Mining* meningkat sebesar 5% seiring dengan lonjakan nilai *Bitcoin* (Saputro, Alfitra and Oktaviaji, 2020).

2.2. CopyCat

Aplikasi yang terinfeksi oleh *malware copycat* melakukan *root* pada perangkat pengguna dan memberikan kendali atas perangkat pengguna kepada penyerang. *Copycat* adalah keluarga *adware* yang menyuntikkan kode ke proses *Zygote*(Suresh, 2018).

2.3. HummingBad

Malware ini sangat canggih karena bisa dihindari dengan membuat *rootkit* permanen. *Malware* ini secara diam-diam menginstal perangkat lunak pada perangkat android yang terinfeksi dan digunakan oleh penjahat dunia maya untuk mendapatkan keuntungan ilegal(Febriandiny, 2021).

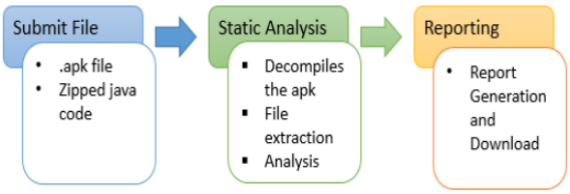
2.4. Sistem Operasi Android

Sistem android adalah sistem operasi berbasis Linux untuk perangkat seluler seperti *smartphone* dan tablet pribadi. Sistem android memiliki keunggulan seperti sistem operasi terbuka, *multitasking*, dan tampilan yang nyaman dari sejumlah aplikasi dan perangkat lunak yang tersedia di sistem android(Kunang dan ..., 2022).

2.5. Metode Statis

Analisis statis dilakukan tanpa benar-benar menjalankan aplikasi. Ini mirip dengan mencari tahu apa yang terjadi di dalam kode sumber dan mengidentifikasi dengan tepat kode berbahaya apa yang terkandung dalam aplikasi Anda.

Analisis statis adalah proses membongkar APK aplikasi ke dalam file Java dan XML yang sesuai. Pengurai statis harus memiliki kemampuan untuk memvalidasi XML secara akurat dan ringkas untuk melakukan analisis statis. File Java dapat diekstraksi menggunakan dekompiler DEXtoJar. Penganalisis persisten mengompilasi kode APK ke dalam format yang dapat dibaca manusia. sehingga penganalisa dapat membaca kode dan mengidentifikasi kelemahan dalam kode tersebut(Nurindahsari dan Parga Zen, 2022).



Gambar 1. Alur analisis statis

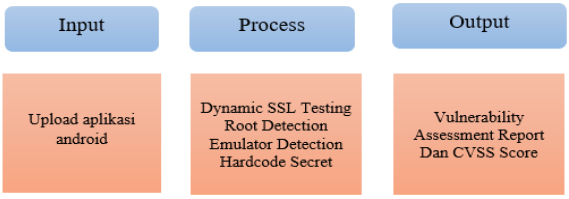
Pada gambar 1 menunjukkan alur *reverse engineering* pada sampel *malware* menggunakan kerangka kerja MobSF. Tahap pertama dalam analisis statis adalah dengan melakukan *reverse engineering* pada sampel *malware* dengan mengubah ekstensi file dari *.apk menjadi *.zip menggunakan kerangka kerja MobSF. Hasil dari proses ini adalah ekstraksi file sampel *malware* menjadi data yang dapat dianalisis seperti yang terlihat pada alur proses di gambar 6. Hasil ekstraksi file dari kerangka kerja MobSF berupa report.pdf atau laporan yang selanjutnya akan dianalisis untuk mengetahui perizinan apa yang diminta oleh aplikasi dari sampel malware terhadap sistem, serta kode sumber dari file aplikasi yang telah diubah melalui proses *decompiler*.

2.6. Metode Dinamis

Analisis dinamis dilakukan dengan cara *running* sampel *malware* dalam jangkauan kontrol dan pemantauan selama dilakukan proses *running*. Pada beberapa kasus analisis statis tidak menampilkan informasi yang banyak karena *obfuscation* dan *packing*. Analisis dinamis adalah cara terbaik untuk dapat mengidentifikasi fungsionalitas *malware* pada kasus seperti ini.

Metode analisis ini terutama menggunakan perangkat lunak seperti *VirtualBox* untuk memastikan bahwa *malware* yang berjalan tidak merusak sistem utama, meskipun *malware* yang berjalan merusak sistem(Adenansi and Novarina, 2017).

Itu aktif di lingkungan yang aman dalam sistem fisik atau bentuk virtual yang disediakan sebagai *lab malware* (mesin virtual). Kemudian dapat mengumpulkan informasi tentang dampak file *malware* saat menjalankan prosesnya(Cahyanto, Wahanggara and Ramadana, 2017).



Gambar 2. Alur analisis dinamis

Gambar 2 menunjukkan alur kerja yang dilakukan dengan menggunakan analisis dinamis yang dipecah menjadi tiga fase: *input*, *process*, dan *output*. Setiap level memiliki peran dan fungsinya masing-masing. Pada tahap *input*, aplikasi android diunggah ke MobSF. Langkah *process* MobSF menguji aspek keamanan dari semua aktivitas aplikasi. Hasil proses mengungkap kerentanan keamanan berdasarkan aktivitas yang diuji(Alviansyah and Ramadhani, 2021).

2.7. Metode Hybrid

Analisis menggunakan pendekatan hybrid, yaitu kombinasi antara analisis statis dan dinamis. Hybrid menggabungkan manfaat statis dan dinamis. Artinya, saat kode dipindai, ia memeriksa semua tanda malware dan memantau perilaku kode.

Analisis *hybrid* mengumpulkan informasi tentang *malware* dari analisis statis dan analisis dinamis. Dengan menggunakan *hybrid* analisis, peneliti keamanan mendapatkan manfaat dari keduanya analisis statis dan dinamis. Oleh karena itu, meningkatkan kemampuan mendeteksi *malware* dengan benar. Kedua analisis ini memiliki kelebihan dan keterbatasannya masing-masing. Analisis statis lebih murah, cepat dan lebih aman dibandingkan dengan analisis dinamis(Sihwail, Omar and Ariffin, 2018).

2.8. Mobile Security Framework

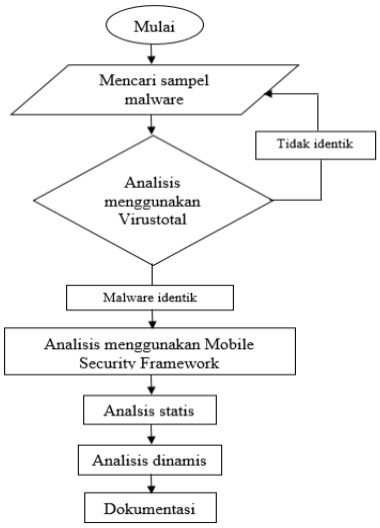
Mobile Security Framework (MobSF) adalah kerangka kerja untuk pengujian penetrasi seluler (Android/iOS/Windows) otomatis yang mampu melakukan analisis statis, dinamis. MobSF digunakan untuk analisis keamanan aplikasi seluler yang efisien dan cepat dari aplikasi seluler dan mendukung *Android Package Kit* (APK), aplikasi iPhone (IPA), format file Windows Store dan kode sumber zip, binari APPX (Yuniati, Tambunan and Setyoko, 2022).

2.9. VirusTotal

Virus total adalah alat untuk memindai *malware* yang tersedia secara online. Alat ini memiliki lebih dari 70 mesin pemindai *anti-malware* dan juga menyediakan laporan analisis serta *metadata* yang kaya dengan informasi tentang *malware* yang dianalisis. *Virus total* bekerja sama dengan 68 vendor keamanan lainnya untuk membangun *database* gabungan (*Virus total Database*) untuk menyediakan informasi yang sangat relevan tentang *file*, web URL atau *hash* yang dianalisis kedalam alat ini(David, 2021).

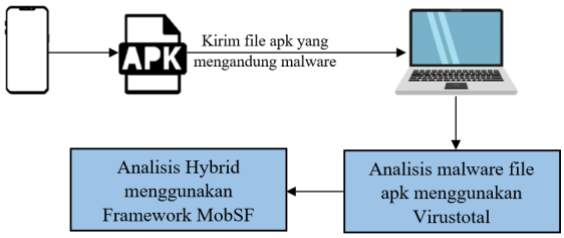
3. METODOLOGI

Metode yang digunakan dalam penelitian ini adalah metode kuantitatif *pre experimental design* dengan menggunakan *one shoot studi case*. Dalam penelitian ini tujuan dari studi kasus tunggal adalah agar kelompok kontrol tidak digunakan sebagai pembanding dengan kelompok eksperimen(Tansen and Wahyu Nurdiarto, 2020).



Gambar 3. Alur penelitian analisis malware

Pada gambar 3 dilakukan pencarian sampel *malware* untuk mendapatkan sebuah aplikasi yang mengandung *malware* untuk dilakukan analisa berikutnya. Selanjutnya, sampel *malware* yang telah ditemukan akan dilakukan analisa menggunakan *virus total* untuk mengetahui jenis *malware* apa yang terdapat dalam aplikasi tersebut selanjutnya dilakukan analisis dengan *framework* MobSF dengan metode analisis statis dan analisis dinamis.



Gambar 4. Skenario analisis malware

Pada gambar 4 tipe file apk yang dijadikan sebagai sampel malware yang berada pada *smartphone* akan dikirim ke leptop/komputer menggunakan aplikasi yang dapat digunakan untuk mentransfer file tipe aplikasi kemudian setelah proses *transfer* file dilakukan selanjutnya akan dilakukan analisis menggunakan *Virus total* untuk mengetahui jenis malware apa saja yang terkandung didalam aplikasi tersebut karena pada sebuah aplikasi bisa jadi bukan hanya satu jenis *malware* saja yang terkandung didalamnya, setelah dipastikan bahwa apliaksi tersebut mengandung jenis *malware* yang dipilih sebagai sampel selanjutnya dilakukan analisis statis dan analisis dinamis menggunakan *Mobile Security Framework.*

4. HASIL DAN PEMBAHASAN

Malware yang diteliti adalah malware hummingbad dan malware copycat. Lihat di bawah untuk detail tentang malware hummingbad dan copycat yang digunakan:

Tabel 1. Informasi Malware Hummingbad Pada Virus Total

No	Malware Hummingbad	
1	SHA256	c27886eb1da732acc144e88b db25607c72d85a0fbcdf82be d6b75dc3548baf07
2	Rasio Deteksi	9/64
3	Waktu Analisis	05-07-2023 20:28:47 WITA
4	Ukuran File	4.02MB
5	Tipe File	APK

Pada tabel 1 terdapat informasi dari *virus total* pada sampel *malware hummingbad* ada 9 *antimalware* dari 64 *antimalware* yang dapat mengidentifikasi bahwa dalam aplikasi *clean doctor* memiliki *malware* didalamnya. *Malware hummingbad* memiliki nilai *checksum* SHA256 atau *Source Hash Algorithm* c27886eb1da732acc144e 88bdb25607e72d85a0fbcdi32bed6075dc3548baf07. File sampel *malware* ini bertipe file APK yang memiliki ukuran 4.02MB.

Tabel 2. Informasi Malware Copycat Pada Virus Total

No	Malware Copycat	
1	SHA256	d1eb05c04fc022024a27d470 060c058f0bbb56adf72467c2 936b85a4ffd76cca
2	Rasio Deteksi	23/64
3	Waktu Analisis	05-07-2023 21:04:36
4	Ukuran File	12.05MB
5	Tipe File	APK

Pada tabel 2 terdapat informasi dari *virus total* pada sampel *malware copycat* ada 23 *antimalware* dari 64 *antimalware* yang dapat mengidentifikasi bahwa dalam aplikasi *clean doctor* memiliki *malware* didalamnya. *Malware copycat* memiliki nilai *checksum* SHA256 atau *Source Hash Algorithm* d1eb05c04fc022024a27d470060c058f0bbb56adf724 67c2936b85a4ffd76cca. File sampel *malware* ini bertipe file APK yang memiliki ukuran 12.05MB.

Data integrity atau keaslian dari kedua sampel tersebut dapat dilakukan pemeriksaan nilai *chechsum* yang diperoleh dari *virus total* dan *mobile security framework.*

Gambar 5. Checksum malware hummingbad pada MobSF

Pada gambar 5 menunjukkan nilai checksum SHA256 dari malware hummingbad. Yang dimana nilai SHA256 pada framework MobSF sama dengan nilai SHA yang didapatkan pada virus to*tal.*

Gambar 6. Checksum malware copycat pada MobSF

Pada gambar 6 menunjukkan nilai checksum SHA256 dari malware copycat menggunakan framework MobSF. Yang dimana nilai SHA256 pada framework MobSF sama dengan nilai SHA yang didapatkan pada *virus total.*

Tabel 3. Nilai Perbandingan Checksum Pada Virus Total Dan Mobsf

No	Malware	Nilai Checksum		Definisi
		Virustotal	MobSF	
1	Hummingbad	c27886eb 1da732ac c144e88b db25607c 72d85a0f bcdf82be d6b75dc3 548baf07	c27886e b1da732 acc144e 88bdb25 607c72d 85a0fbc df82bed 6b75dc3 548baf07	Identik
2	Copycat	d1eb05c0 4fc02202 4a27d470 060c058f 0bbb56ad f72467c29 36b85a4ff d76 cca	d1eb05c 04fc022 024a27d 470060c 058f0bb b56adf7 2467c29 36b85a4 ffd76cca	Identik

4.1. Analisis Statis Malware Hummingbad

4.1.1. Application Permission Analysis

Application permission analysis merupakan hasil dari *reverse engineering* menggunakan *mobile security framework* yang dimana didalamnya mengandung keseluruhan informasi dari suatu aplikasi, seperti informasi file, informasi aplikasi, komponen aplikasi, sertificate informasi, perizinan aplikasi, APKID analisis, analisis manifest, analisis kode, domain pemeriksaan malware, basis data firebase, email dan pelacak.

Tabel 4. Application Perssion Malware Hummingbad

No	PERMISSION	STATUS	INFO
1	android.permission.CALL_PHONE	Berbahaya	langsung menghubungi nomor telepon
2	android.permission.GET_TASKS	Berbahaya	mengambil aplikasi yang sedang berjalan
3	android.permission.MOUNT_UNMOUNT_FILESYSTEMS	Berbahaya	Meningkatkan dan tidak meningkatkan system file
4	android.permission.PROCESS_OUTGOING_CALLS	Berbahaya	mencegat panggilan keluar
5	android.permission.READ_EXTERNAL_STORAGE	Berbahaya	membaca isi penyimpanan eksternal
6	android.permission.READ_LOGS	Berbahaya	membaca data log sensitif
7	android.permission.READ_PHONE_STATE	Berbahaya	membaca status dan identitas ponsel
8	android.permission.READ_SMS	Berbahaya	membaca SMS atau MMS
9	android.permission.RECEIVE_SMS	Berbahaya	Menerima SMS
10	android.permission.SYSTEM_ALERT_WINDOW	Berbahaya	Menampilkan peringatan tingkat sistem
11	android.permission.WRITE_EXTERNAL_STORAGE	Berbahaya	Baca/modifikasi/hapus konten penyimpanan eksternal
12	android.permission.WRITE_SETTINGS	Berbahaya	Memodifikasi pengaturan sistem global

4.1.2. Code Review

Setelah semua perizinan oleh aplikasi sampel malware terhadap sistem ditemukan maka dilakukan analisis yang lebih lanjut dalam bentuk java source code.

```
public void onReceive(final Context context, Intent intent) {
    final int i;
    final int i2;
    final int i3;
    final int i4;
    final int i5;
    final int i6;
    final String action = intent.getAction();
    if (intent.getAction().equals("android.intent.action.NEW_OUTGOING_CALL")) {
        com.boost.clean.doctor.without.lock.a.a(false);
    } else if ("android.intent.action.PHONE_STATE".equals(action)) {
        switch (((TelephonyManager) context.getSystemService("phone")).getCallState()) {
            case 0:
                com.boost.clean.doctor.without.lock.a.a(true);
                return;
```

Gambar 7. Analisis fungsi source code call phone malware hummingbad

Gambar 7 adalah analisis statis dari kode sumber malware hummingbad. Malware hummingbad diketahui memiliki kemampuan untuk merekam dan memantau panggilan telepon di perangkat pengguna. Data yang dicuri dikumpulkan menggunakan operasi XOR sederhana, dienkripsi, dan dikirim ke server perintah dan kontrol menggunakan metode HTTP POST.

4.2. Analisis Statis Malware Copycat

4.2.1. Application Permission

Proses yang dijalankan adalah proses yang sama yang dijalankan oleh sampel *malware Hummingbad.* Metode rekayasa balik (*reverse engineering*) dilakukan dengan menyusun file *malware* yang disimulasikan dengan MobSF. Analisis ini berfokus pada izin aplikasi sehingga kami dapat menemukan izin yang diminta oleh salinan malware sampel.

Tabel 5. Application Permission Malware Copycat

No	PERMISSION	STATUS	INFO
1	android.permission.ACCESS_COARSE_LOCATION	Berbahaya	*coarse (network-based) location*
2	android.permission.ACCESS_FINE_LOCATION	Berbahaya	*fine (GPS) location*
3	android.permission.GET_ACCOUNTS	Berbahaya	*list accounts*
4	android.permission.READ_PHONE_STATE	Berbahaya	*read phone state and identity*
5	android.permission.SET_DEBUG_APP	Berbahaya	*enable application debugging*
6	android.permission.WRITE_EXTERNAL_STORAGE	Berbahaya	*read/modify/delete external storage contents*

4.2.2. Code Review

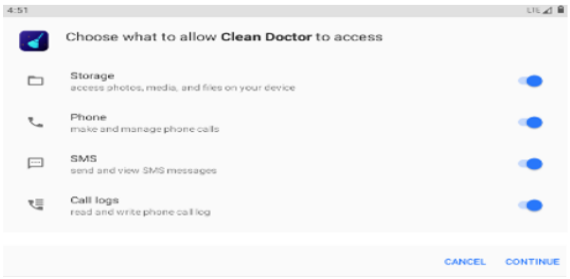
Setelah menemukan semua izin istimewa pada sistem, program malware sampel melakukan analisis lebih lanjut dalam bentuk kode sumber java.

```
private static final String a = em.class.getSimpleName();
private static byte[] b;

public static byte[] a() {
    if (b != null) {
        return b;
    }
    if (eg.a().b().checkCallingOrSelfPermission("android.permission.READ_PHONE_STATE") != 0) {
        return null;
    }
    b();
    return b;
}

private static void b() {
    String deviceId;
    TelephonyManager telephonyManager = (TelephonyManager) eg.a().b().getSystemService("phone");
    if (telephonyManager != null && (deviceId = telephonyManager.getDeviceId()) != null && deviceId.trim().length() > 0) {
        try {
            byte[] d = fh.d(deviceId);
            if (d != null && d.length == 20) {
                b = d;
            } else {
                ex.a(6, a, "shal is not 20 bytes long: " + Arrays.toString(d));
            }
        } catch (Exception e) {
            ex.a(6, a, "Exception in generateHashedImei()");
```

Gambar 8. Analisis fungsi source code call phone malware copycat

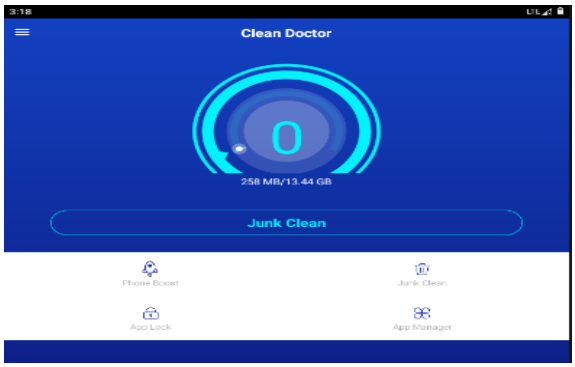
Gambar 8 menunjukkan hasil analisis statis dari kode sumber malware, yang diketahui mengontrol panggilan terminal pengguna. Data yang dicuri dikumpulkan menggunakan operasi XOR sederhana, dienkripsi dan dikirim ke server perintah dan kontrol menggunakan metode HTTP POST.

4.3. Analisis Dinamis Malware Hummingbad



Gambar 9. Perizinan aplikasi

Pada gambar 9 menunjukkan perizinan yang akan diminta setelah menginstal aplikasi clean doctor ada 4 perizinan yang diminta yaitu storage, phone, SMS dan call logs. Dimana jika kita mengizinkan perizinan-perizinan tersebut maka pihak clean doctor dapat mengakses penyimpanan, telepon, SMS dan log panggilan pada smartphone kita.



Gambar 10. Menu utama aplikasi clean doctor

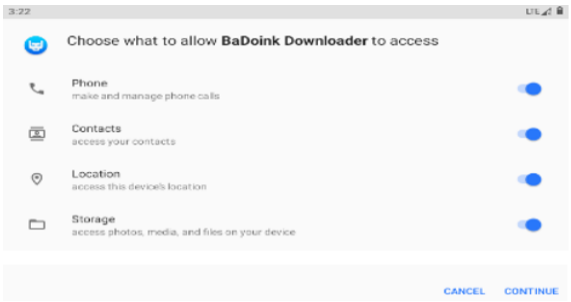
Pada gambar 10 yaitu tampilan utama dari aplikasi *clean doctor* dimana aplikasi ini berfungsi sebagai pembersih *cache* yang berada pada *smartphone* serta *clean doctor* dapat digunakan untuk mengelola, menginstal aplikasi dan penyimpanan pada *smartphone.*

Gambar 11. Monitoring traffic malware HummingBad dengan HTTP tools MobSF

Gambar 11 menunjukkan pengawasan yang dilakukan menggunakan alat HTTP pada kerangka kerja MobSF. Hasil pengawasan ini menemukan respon ke alamat situs web GET http://ads.mopub.com/m/ad?id=d4dcab0a6f0245d49

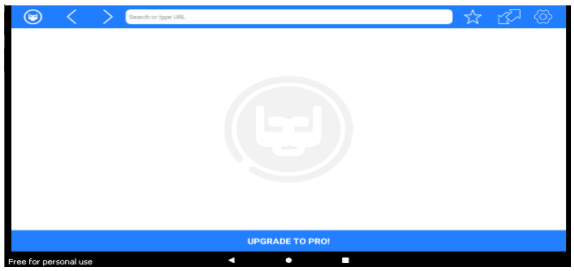
3c3585ad39e0675&nsv=4.13.0 dengan metode POST. Respon sebelumnya adalah 404, yang berarti situs web sementara dialihkan. Hal ini mungkin disebabkan oleh gangguan pada *server* nama domain (DNS) yang mengarah ke situs web tersebut.

4.4. Analisis Dinamis Malware Copycat



Gambar 12. Perizinan aplikasi

Pada gambar 12 menunjukkan perizinan yang akan diminta setelah menginstal aplikasi *badoink* ada 4 perizinan yang diminta yaitu *phone*, *contacts*, *location* dan *storage*. Dimana jika kita mengizinkan perizinan-perizinan tersebut maka pihak *badoink* dapat mengakses telepon, kontak, lokasi dan penyimpanan pada *smartphone* kita.



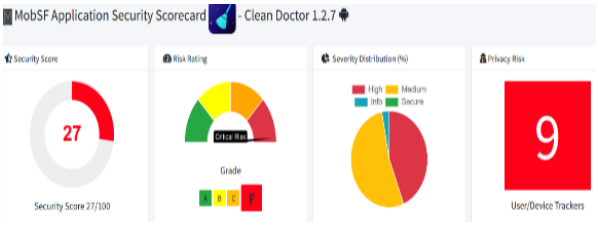
Gambar 13. Menu utama pada aplikasi badoink

Pada gambar 13 merupakan tampilan utama dari aplikasi *badoink* dimana aplikasi ini berfungsi untuk memungkinkan pengguna mendapatkan akses gratis ke semua fitur premium. Aplikasi ini memungkinkan pengguna untuk meonton video online, baik di situs *video streaming* atau situs lainnya tanpa harus membayar sejumlah uang.

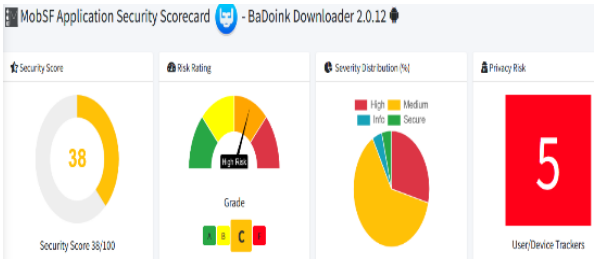
Gambar 14. Monitoring traffic malware CopyCat dengan HTTP tools MobSF

Gambar 14 menunjukkan lalu lintas HTTP yang dilakukan oleh *malware Copycat* ke alamat http://ads.mopub.com/m/ad?v=6&id=34c7be951721 4aba813b350bdf2329dc&nv=4.13.0 dengan metode POST. Respon sebelumnya adalah 404, yang berarti situs web sementara dialihkan. Hal ini mungkin disebabkan oleh gangguan pada *server* nama domain (DNS) yang mengarah ke situs web tersebut.

4.5. Pembahasan



Gambar 15. Tingkat keamanan malware hummingbad



Gambar 16. Tingkat keamanan malware copycat

Pada gambar 15 dan 16 merupakan tingkat keamanan dari kedua sampel yang digunakan dimana *malware hummingbad* memiliki tingkat keamanan 27/100 sedangkan *malware copycat* memiliki tingkat keamanan 38/100. Meskipun kedua sempel *malware* sama-sama beresiko tinggi akan tetapi tingkat keamanan pada *malware hummingbad* lebih rendah dari pada sampel *malware copycat* karena *device trecker* atau pelacak perangkat pada sampel *malware hummingbad* yang ditemukan oleh *framework* MobSF lebih banyak dibandingkan dengan *malware copycat*.

Risk reting atau reting resiko yang dimiliki oleh kedua malware ini berdasarkan pada analisis statis yang dilakukan pada MobSF yaitu sampel *malware hummingbad* mendapatkan reting *critical risk* sedangkan sampel *malware copycat* mendapatkan reting *high risk*. Sampel *malware copycat* mendapatkan reting yang lebih rendah dari pada sampel *malware hummingbad* karena tinggkat keamanan *malware copycat* masih lebih tinggi dari pada tingkat keamanan yang didapatkan oleh *malware hummingbad.*

Privasi risk atau resiko privasi dari sampel *malware hummingbad* yang ditemukan oleh *framework* MobSF pada sampel *malware* yaitu 9 *device trecker* sedangkan sampel dari *malware copycat* hanya ditemukan 5 *devise trecker*.

Pada tabel 6 merupakan informasi dari *malware hummingbad* dan *copycat* yang didapatkan oleh *framework* MobSF dimana *malware hummingbad* memiliki nilai *chechsum* SHA256 c27886eb1da732 acc144e88bdb25607c72d85a0fbcdf82bed6b75dc354 8baf07 dengan target SDK 19, *min SDK* 11, *android version name* 1.2.7 dan *android version code* 28.

Sedangkan *malware copycat* memiliki nilai *checksum* d1eb05c04fc022024a27d470060c058f0bbb56adf72467c2936b85a4ffd76cca dengan target SDK 19, *min SDK* 7, *android version name* 2.0.12 dan *android version code* 12.

Tabel 6. Informasi Malware Pada Mobsf

No		Mobile Security Framework	
		Humming Bad	**Copycat**
1	SHA256	c27886eb 1da732ac 144e88bd 25607c72 d85a0fbc df82bed6 b75dc354 8baf07	d1eb05c04 fc022024a 27d47006 0c058f0bb b56adf724 67c2936b 85a4ffd76 cca
2	Target SDK	19	19
3	Min SDK	11	7
4	Android Version Name	1.2.7	2.0.12
5	Android Version Code	28	12

5. KESIMPULAN DAN SARAN

Malware yang digunakan sebagai objek penelitian yaitu *malware hummingbad* dan *malware copycat*. Penelitian ini akan menggunakan virus total untuk mengidentifikasikan jenis malware apa yang berada dalam aplikasi yang akan digunakan. Berdasarkan hasil dari virus total *malware hummingbad* memiliki SHA c27886eb1da732acc144e88bdb25607e72d85a0fbcdi32bed6075dc3548baf07 dan *malware copycat* memiliki SHA d1eb05c04fc022024a27d470060c058f0bbb56adf72467c2936b85a4ffd76cca.

Kemudian dilanjutkan dengan melakukan analisis statis dan analisis dinamis menggunakan mobile security framework yang dimana pada analisis statis sampel malware hummingbad memiliki tingkat keamanan yang sangat rendah yaitu 27/100 yang dimana sampel malware yang digunakan memiliki ukuran 4.02MB. pada saat sampel malware hummingbad dijalankan aplikasi ini akan meminta beberapa perizinan aplikasi seperti perizinan penyimpanan, telepon, SMS dan log panggilan. Pada analisis statis sampel malware copycat tingkat keamanan yang didapatkan yaitu 38/100 walaupun tingkat keamanan yang didapatkan lebih tinggi dibandingkan dengan sampel malware hummingbad tapi persentase ini juga sudah termaksud beresiko tinggi. Sampel malware copycat bertipe file APK dengan ukuran file 12.05MB. Pada saat sampel malware copycat dijalankan aplikasi ini akan meminta berizinan-perizinan seperti perizinan telepon, kontak, lokasi dan perizinan penyimpanan. Baik dalam tingkat keamanan, reting resiko maupun resiko privasi berdasarkan hasil analisis yang didapatkan sampel malware hummingbad lebih berbahaya dari pada sampel malware copycat meskipun sampel malware copycat tidak lebih berbahaya dari sampel malware hummingbad ini tidak menutupi bahwa berdasarkan hasil yang didapatkan sampel malware copycat juga berbahaya jika pengguna menginstal aplikasi dari sampel malware tersebut.

DAFTAR PUSTAKA

Adam, M., Alwi, E.I. and As'ad, I. (2022) Analisis Forensik Terhadap Serangan Ddos Ping of Death Pada Server, *Cyber Security dan Forensik Digital*, 5(1), pp. 23–31. Available at: https://doi.org/10.14421/csecurity.2022.5.1.3423.

Adenansi, R. and Novarina, L.A. (2017) Malware dynamic, *Jurnal of Education and Information Communication Technology*, 1(1), pp. 37–43.

Alviansyah, F.A. and Ramadhani, E. (2021) Implementasi Dynamic Application Security Testing pada Aplikasi Berbasis Android, *Automata*, 2(1), pp. 1–6.

Cahyanto, T.A., Wahanggara, V. and Ramadana, D. (2017) Analisis dan Deteksi Malware Menggunakan Metode Malware Analisis Dinamis dan Malware Analisis Statis, *Jurnal Sistem & Teknologi Informasi Indonesia*, 2(1), pp. 19–30.

David, T.T. (2021) Analisis Malware Backdoor Menggunakan Metode Analisis Statis Dan Dinamis Pada Malware Analysis Lab Serta Rancang Bangun Aplikasi Untuk Menghapus Malware, *Repository Politeknik Negeri Jakarta* [Preprint].

Febriandiny, F.A. (2021) *Analisis Malware "Hummingbad" Pada Perangkat Smartphone Menggunakan Metode Hybrid*, *Repositori Universitas Siliwangi*.

Kunang, Y.N.K.Y.N. and ... (2022) Analisis Forensik Malware Pada Platform Android., *Analisis Forensik …*, pp. 2–10.

Kusuma, S.D., Akbi, D.R. and Risqiwati, D. (2020) Analisis Malware Berdasarkan Tujuan Pembuatan Dengan Menggunakan Metode Hybrid Pada Android, *Jurnal Repositor*, 2(8), pp. 1163–1173. Available at: https://doi.org/10.22219/repositor.v2i8.764.

Manoppo, V.A., Lumenta, A.S.. and Karouw, S.D.. (2020) Analisa Malware Menggunakan Metode Dynamic Analysis Pada Jaringan Universitas Sam Ratulangi, *Jurnal Teknik Elektro Dan Komputer*, 9(3), pp. 181–188.

Nurindahsari, F. and Parga Zen, B. (2022) Analisis

Statik Keamanan Aplikasi Video Streaming Berbasis Android Menggunakan Mobile Security Framework (Mobsf), *Cyber Security dan Forensik Digital*, 4(2), pp. 63–80. Available at: https://doi.org/10.14421/csecurity.2021.4.2.3373.

Pranoto, W. (2019) Malicious Software Analysis, *Cyber Security dan Forensik Digital*, 1(2), pp. 62–66. Available at: https://doi.org/10.14421/csecurity.2018.1.2.1374.

Rusdi, A.S., Widiyasono, N. and Sulastri, H. (2019) Analisis Infeksi Malware Pada Perangkat Android Dengan Metode Hybrid Analysis, *Universitas Putera Batam*, 46115(24), p. 107.

Saputro, B.A., Alfitra, L.I. and Oktaviaji, R.B. (2020) Analisis Malware Android Menggunakan Metode Reverse Engineering, *Jurnal Repositor*, 2(10), pp. 1331–1337. Available at: https://doi.org/10.22219/repositor.v2i10.1061.

Sihwail, R., Omar, K. and Ariffin, K.A.Z. (2018) A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis, *International Journal on Advanced Science, Engineering and Information Technology*, 8(4–2), pp. 1662–1671. Available at: https://doi.org/10.18517/ijaseit.8.4-2.6827.

Sinambela, S., Pangestu, A.R. and Feriyanto, R. (2020) Analisis Aplikasi Malware pada Android dengan Metode Statik, *Jurnal Ilmiah ILKOMINFO - Ilmu Komputer & Informatika*, 3(2). Available at: https://doi.org/10.47324/ilkominfo.v3i2.101.

Suresh, S. (2018) Analyzing Android Adware, *Masters Projects*, p. 621.

Tansen, E. and Wahyu Nurdiarto, D. (2020) Analisis dan Deteksi Malware Dengan Metode Hybrid Analysis Menggunakan Framework MobSF, *Jurnal Teknologi Informasi*, 4(2). Available at: https://doi.org/10.36294/jurti.v4i2.1338.

Widiyasono, N., Mubarok, H. and Fatwa MF, A. (2022) Analisis Malware Ahmyth pada Platform Android Menggunakan Metode Reverse Engineering, *Generation Journal*, 6(2), pp. 73–82. Available at: https://doi.org/10.29407/gj.v6i2.17749.

Yuniati, T., Tambunan, A.R. and Setyoko, Y.A. (2022) Implementasi Static Analysis Dan Background Process Untuk Mendeteksi Malware Pada Aplikasi Android Dengan Mobile Security Framework, *LEDGER : Journal Informatic and Information Technology*, 1(2), pp. 24–28. Available at: https://doi.org/10.20895/ledger.v1i2.848.