ANALISIS KEAMANAN PADA APLIKASI HIMFO BERBASIS ANDROID MENGGUNAKAN MOBSF

Putri Rizkika, Didi Juardi, Agung Susilo Yuda Irawan
Informatika, Universitas Singaperbangsa Karawang
Jl. HS.Ronggo Waluyo, Puseurjaya, Telukjambe Timur, Karawang, Jawa Barat 41361
putririzkika807@gmail.com

ABSTRAK

HIMFO merupakan aplikasi yang dikembangkan oleh sebuah organisasi mahasiswa di Program Studi Informatika Fakultas Ilmu Komputer Universitas Singaperbangsa Karawang. Tujuan utama aplikasi ini dibuat adalah untuk membantu tugas sehari-hari dan meningkatkan efisiensi pengguna melalui beragam fitur yang telah disediakan. Namun, HIMFO juga mengandung data yang bersifat pribadi, sehingga diperlukan pengujian keamanan untuk mengidentifikasi dan menutup celah kerentanan yang dapat dieksploitasi oleh pihak yang memiliki motif pribadi. Kehadiran celah keamanan dapat membuka peluang bagi serangan yang berpotensi merugikan pengguna dengan memanfaatkan data yang telah diretas. Dalam penelitian ini, *Mobile Security Framework* (MobSF) digunakan untuk menguji keamanan aplikasi HIMFO berbasis Android. Penelitian ini memfokuskan pada lima parameter: *weak crypto, SSL bypass, root detection, dangerous permissions*, dan *domain malware check*, . Hasil penelitian menunjukkan bahwa aplikasi HIMFO memiliki *root detection, weak crypto*, SSL *bypass*, dan *domain malware check* dengan status *good*.

Kata kunci : *Android, Aplikasi HIMFO, Keamanan, MobSF*

1. PENDAHULUAN

Data pribadi yang dibuat, disimpan, dan ditransmisikan melalui komputer, perangkat *mobile*, *broadband*, situs internet, dan media telah meningkat secara eksponensial selama era digital. Hal ini terbukti dengan tingginya presentase jumlah pengguna aplikasi Android. Indonesia menempati urutan keempat dengan jumlah pengguna Android terbesar di dunia. Berdasarkan dari laporan data.ai, sepanjang tahun 2021 pengguna Indonesia mengunduh 7,31 miliar aplikasi, Sementara itu, total belanja di App Store dan Play Store mencapai US$532 juta, naik 38% dalam 2 tahun terakhir. Aplikasi Android berbasis *mobile* telah mengubah cara berkomunikasi, berbelanja, bekerja, dan bahkan mengakses informasi. Namun, dengan pertumbuhan ini, keamanan aplikasi *mobile* menjadi perhatian utama. Kontrol izin aplikasi Android dikritik karena tidak efektif, memungkinkan pengguna menerima semua permintaan izin tanpa pertimbangan. Ini dapat menyebabkan kehilangan privasi dan kebocoran data pengguna.

Penggunaan aplikasi Android merambah berbagai bidang, termasuk pendidikan. Universitas Singaperbangsa Karawang merilis aplikasi HIMFO untuk mahasiswa Informatika. Namun, aplikasi ini telah dicabut izin instalasinya oleh Google Play Store karena masalah pembaruan. Untuk mengantisipasi masalah keamanan, perlu dilakukan pengujian dan pemeriksaan menyeluruh sebelum aplikasi tersebut diluncurkan kembali, guna mencegah kemungkinan sistem teretas setelah rilis. Berdasarkan pemaparan di atas, pada penelitian ini akan dilakukan pengujian analisis statik dengan menggunakan *Mobile Security Framework* (MobSF) untuk dapat menemukan celah keamanan atau kerentanan pada sebuah aplikasi *Mobile* HIMFO berbasis Android dengan lima parameter yang harus di analisis.

2. TINJAUAN PUSTAKA

2.1. Kerentanan

Kerentanan adalah kelemahan atau celah dalam suatu sistem yang bisa dimanfaatkan oleh pihak tidak berwenang untuk mendapatkan akses ke sistem tersebut. Dalam konteks aplikasi *mobile*, kerentanan dapat memungkinkan *hacker* atau *cracker* untuk mencuri informasi pribadi pengguna, merusak integritas data, atau mengganggu layanan aplikasi [1]. Kerentanan dapat terjadi karena berbagai faktor seperti, kesalahan konfigurasi, kelemahan perangkat lunak, kekurangan dalam desain, dan sebagainya.

2.2. Android

Android adalah salah satu sistem operasi dari Linux untuk perangkat *Mobile* yang mencakup sistem operasi, *middleware*, dan aplikasi. Android merupakan sebuah platform berjenis *open source* (terbuka) sehingga para pengembang dapat menciptakan aplikasi. Selain itu, Android digunakan untuk pengelolaan sumber daya perangkat keras, baik untuk smartphone, PC tablet smart TV, dan lain sebagainya [2]. Hal ini selaras dengan Mansyuruddin bahwa Android adalah sebuah sistem operasi untuk *smartphone* dan tablet [3]. Sistem operasi dapat diilustrasikan sebagai jembatan antara piranti (*device*) dan penggunanya, sehingga pengguna dapat berintraksi dengan *device* nya dan menjalankan aplikasi-aplikasi yang tersedia pada *device*.

2.3. HIMFO

HIMFO merupakan aplikasi yang dikembangkan oleh sebuah organisasi mahasiswa di Program Studi Informatika Fakultas Ilmu Komputer Universitas Singaperbangsa Karawang. Tujuan utama aplikasi ini adalah untuk membantu tugas dan meningkatkan efisiensi pengguna melalui beragam fitur yang disediakannya. Aplikasi HIMFO ini merupakan aplikasi *mobile* berbasis Android yang penggunaannya lebih *flexible* lagi bagi pengguna. Oleh karena itu, aplikasi ini dapat membantu pengguna mempercepat dalam mengerjakan pekerjaannya dan mempermudah pengguna dalam mendapatkan sebuah informasi.

2.4. *Mobile Security Framework*

Mobile Security Framework (MobSF) adalah framework yang digunakan untuk pengujian penetrasi teerhadap aplikasi seluler (Android/iOS/Windows) otomatis yang mampu melakukan analisis statis, analisis dinamis, dan *malware* [4]. MobSF memungkinkan analisis keamanan aplikasi seluler Android, iOS, dan Windows yang cepat dan efisien. Mendukung kedua binari (APK, IPA, dan APPX) dan kode sumber zip. Untuk aplikasi Android, MobSF memiliki kemampuan untuk melakukan pengujian aplikasi dinamis saat *runtime* dan didukung oleh CapFuzz, pemindai keamanan khusus Web API [5].

2.5. Analisis Statik

Dalam pengujian keamanan aplikasi *mobile*, analisis statik adalah teknik analisis yang dilakukan tanpa mengeksekusi program. Dalam hal keamanan aplikasi *mobile*, analisis statik melibatkan pemeriksaan kode sumber atau paket aplikasi Android untuk menemukan kerentanan keamanan potensial seperti *weak crypto, SSL bypass, root detection, domain malware check,* dan *dangerous permissions*. Analisis *weak crypto* akan melihat implementasi algoritma kriptografi yang lemah atau penggunaan algoritma kriptografi yang sudah usang atau dianggap tidak layak [4].

Kemudian, untuk melakukan analisis SSL *bypass*, seseorang harus memeriksa apakah layanan yang melibatkan protokol http benar-benar memerlukan penggunaan SSL sebagai persyaratan keamanan transaksi dalam protokol http, seperti mengizinkan http di manifest atau *string* berkonten http://, yang menunjukkan implementasi yang lemah [5].

Selain itu, aplikasi Android dirancang untuk melakukan berbagai tugas, beberapa di antaranya membutuhkan izin pengguna. Analisis izin pengguna dilakukan untuk mengetahui seberapa banyak *dangerous permissions* yang digunakan oleh aplikasi. Analisis *root detection* dilakukan dengan memeriksa apakah aplikasi memiliki kemampuan untuk mendeteksi akses *root* pada perangkat Android yang digunakan. Ini memberikan akses langsung ke sistem, termasuk data aplikasi. Analisis *domain malware check* mengevaluasi apakah domain dalam aplikasi terindikasi termasuk dalam kategori domain yang mengandung *malware* [4].

Dalam proses pengujiannya tidak perlu menjalankan aplikasi nya. Ini memungkinkan untuk menemukan kerentanan potensial sebelum aplikasi tersebut digunakan secara luas. Dalam penelitian yang bertujuan untuk menguji kerentanan keamanan aplikasi *mobile* berbasis Android, MobSF digunakan untuk melakukan analisis statik. Tahapan uji statik ini mencakup persiapan, memilih aplikasi yang akan diuji, dan menerapkan *software* MobSF [6].

2.6. *Common Vulnerability Scoring System* (CVSS)

CVSS adalah standar yang digunakan untuk mengevaluasi tingkat kerentanan perangkat lunak [7]. Suatu sistem skor kerentanan umum (CVSS) digunakan untuk menilai kerentanan sistem oleh individu dan lembaga. CVSS menilai kerentanan sistem berdasarkan 7 bagian besar: *attack vector, attack complexity, privilege required, user interaction, confidentiality, integrity,* dan *availability*. CVSS menilai kerentanan dari 0.0 hingga 10.0 berdasarkan 4 aspek, yaitu *Low* (0.0 hingga 3.9), *Medium* (4.0 hingga 6.9), *High* (7.0 hingga 8.9), dan *Critical* (9.0 hingga 10.0).

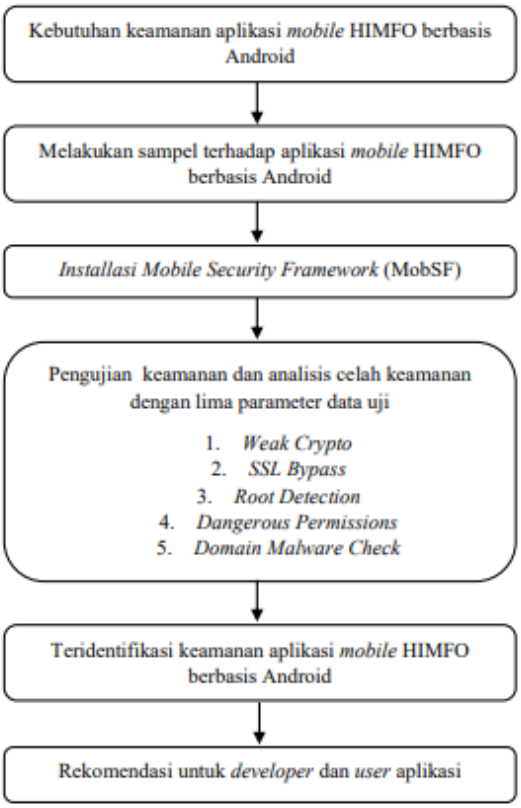
3. METODE PENELITIAN

Metode penelitian dalam penelitian ini menggunakan *penetration testing* dengan *tools Mobile Security Framework* (MobSF). *Mobile Security Framework* (MobSF) merupakan sebuah *tools* yang mampu melakukan analisis statik pada aplikasi *mobile* berbasis Android. Dalam pengujian, terdapat lima parameter yang harus di analisis yaitu *weak crypto,* SSL *bypass, root detection, dangerous permissions,* dan *domain malware check.* Selain itu, *Mobile Security Framework* (MobSF) juga akan mencari *security score, trackers detection*, dan nilai CVSS pada aplikasi tersebut.

3.1. Alur Penelitian

Alur penelitian ini mengidentifikasi suatu masalah atau fenomena yang harus diteliti, yaitu penelitian analisis statik keamanan pada aplikasi *mobile* HIMFO berbasis Android menggunakan *tools Mobile Security Framework* (MobSF). Adapun alur penelitiannya adalah sebagai berikut:

Peneliti menggunakan *Mobile Security Framework* (MobSF) untuk menemukan masalah keamanan pada aplikasi HIMFO berbasis android. Kemudian, akan dilakukan analisis statik dan menyarankan solusi untuk masalah tersebut sehingga pengguna dapat mengetahui tingkat keamanan aplikasi HIMFO berbasis Android yang telah diteliti.

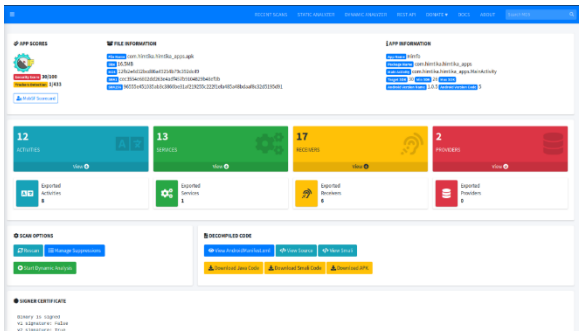


Gambar 1. Alur penelitian

4. HASIL DAN PEMBAHASAN

4.1. Hasil Analisis

Pada tahap awal, file aplikasi HIMFO dengan format APK diunduh. Setelah proses selesai, hasil analisis di *upload* ke *Mobile Security Framework* (MobSF). Saat proses telah selesai, maka akan muncul hasil analisis file tersebut.



Gambar 2. Hasil Analisis Aplikasi HIMFO pada MobSF

Berdasarkan Gambar 2 hasil analisis yang dilakukan oleh *Mobile Secuirty Framework* (MobSF) salah satunya memberikan informasi mengenai komponen-kompenen Android di antaranya *Activity*, *Service*, *Broadcast Receiver*, dan *Content Provider*. Pada aplikasi *mobile* Android HIMFO terdapat 12 *Activities*, 13 *Services*, 17 *Receivers*, dan 2 *Providers*. Analisis statis dari *Mobile Security Framework* (MobSF) menemukan isu komponen-komponen yang terekspos, di antaranya terdapat 8 *exported activities* dari 12 *activities*, 1 *exported services* dari 13 *services*, dan 6 *exported receivers* dari 17 *receivers*. Pada hasil analisis tersebut juga menampilkan nilai *security score* nya yaitu 30/100 yang berarti aplikasi *mobile* Android HIMFO memiliki tingkat keamanan yang rendah dan berbahaya. Selain itu, menampilkan juga nilai *trackers detection* sebanyak 1/433 menunjukkan bahwa satu *tracker* telah terdeteksi dari total 433 yang diawasi atau dipantau.

Selain itu, terdapat juga sebuah informasi aplikasi HIMFO dari hasil analisis *Mobile Security Framework* (MobSF) yang terdapat pada Tabel 1. Informasi Aplikasi HIMFO sebagai berikut:

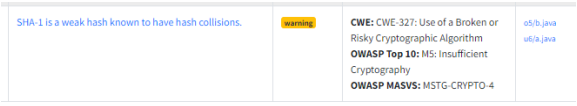
Tabel 1. Informasi Aplikasi Berdasarkan

Info	Keterangan
Nama Aplikasi	HIMFO
Nama *Package*	com.himtika.himtika_apps
Main Activity	com.himtika.himtika_apps.MainActivity
Versi Aplikasi	1.0.5
Android Version Code	5
Minimal SDK	26
Target SDK	30
Developer	Sukacode Dev | Himpunan Mahasiswa Informatika
Certificate/ Key	MD5, SHA1, dan SHA256

Berdasarkan hasil *scanning* otomatis dengan MobSF pada Tabel 1. di atas terlihat bahwa file APK yang menjadi target untuk di analisis merupakan versi aplikasi 1.0.5 dengan *Android Version Code* 5. Aplikasi *Mobile* Android HIMFO juga dapat beroperasi atau berjalan dengan minimal API (*Application Programming Interface*) pada level 26 atau Android 8 dengan target SDK (*Software Development Kit*) pada API level 30 atau Android 11. Keaslian dari file APK aplikasi *Mobile* Android HIMFO dapat diperiksa menggunakan algoritma kriptografi SHA256 (*Secure Hasing Algorithm*) dengan hash MD5 (*Message Digest*) dan SHA1.

4.1.1. *Weak Crypto*

Pada aplikasi *mobile* Android HIMFO memiliki 3 *weak crypto* yang terdiri dari 1 *high severity* dan 2 *warning severity*. *Weak crypto* yang memiliki *warning* severity terjadi pada SHA-1 dan *Random Number Generator*. Sedangkan *weak crypto* yang memiliki *high severity* terjadi pada *The Encryption Mode* CBC *with* PKCS5/PKCS7 *Padding*.



Gambar 3. Hasil Analisis *Weak Crypto* SHA1 Aplikasi HIMFO oleh MobSF

Berdasarkan Gambar 3 bahwa terdapat sebuah *weak crypto* SHA-1 dengan *warning severity* yang

terdapat pada dua lokasi file yaitu pada o5/b.java dan u6/a.java. SHA-1 atau *Security Hash Algorithm* adalah *hash* lemah yang diketahui memiliki tabrakan *hash*, yaitu dua atau lebih teks yang menghasilkan nilai hash yang sama. Serangan *brute-force* adalah serangan yang mencoba semua kombinasi huruf, angka, dan simbol untuk mendapatkan *plaintext* dari *hash*, yang dapat dilakukan dengan cepat dengan SHA-1 yang digunakan untuk *password*.

Pada Gambar 3 memberikan informasi bahwa SHA-1 memiliki uji standar CWE (*Common Weakness Enumeration*) terdeteksi CWE-327:penggunaan algoritma kriptografi rusak atau berisiko, *Open Web Application Security Project* (OWASP) Top 10 terdeteksi M5:kriptografi tidak memadai, *The Mobile Application Security Verification Standard* terdeteksi MSTG-CRYPTO-4 artinya aplikasi tidak menggunakan protokol atau algoritme kriptografi yang secara luas dianggap tidak digunakan lagi untuk tujuan keamanan.

```
private static String c(PublicKey publicKey) {
    try {
        byte[] digest = MessageDigest.getInstance("SHA1").digest(publicKey.getEncoded());
        digest[0] = (byte) (((digest[0] & 15) + R.styleable.AppCompatTheme_toolbarNavigati
        return Base64.encodeToString(digest, 0, 8, 11);
    } catch (NoSuchAlgorithmException unused) {
        Log.w("ContentValues", "Unexpected error, device missing required algorithms");
        return null;
    }
```

Gambar 4. Kode pada o5/b.java

Pada Gambar 4 terdapat sebuah kode dari file o5/b.java yang menunjukkan adanya kode Message. Digest. getInstance (“SHA-1”) yang menandakan bahwa SHA-1 akan mengenkripsi sebuah string. *Message digest* adalah sebuah nilai yang dikenal sebagai *cryptography checksum* atau *secure* hash. *Message digest* digunakan untuk meningkatkan keamanan dalam mentranformasikan sebuah data dan juga kelas yang dipakai dalam sebuah aplikasi yang membutuhkan autentikasi *user* melalui kata sandi atau password. Selain itu, pada kode di Gambar 4 nilai *hash* (*digest*) yang diarahkan yaitu ke publicKey.getEncoded() dimana hal ini merupakan kunci publik yang sering kali digunakan dalam kriptografi untuk enkripsi dan tanda tangan digital

Gambar 5. Hasil Analaisis *Weak Crypto Random Number Generator* Aplikasi HIMFO oleh MobSF

Berdasarkan Gambar 5 bahwa terdapat sebuah *weak crypto* pada *Random Number Generator* atau Generator Angka Acak dengan *warning severity* dan memiliki 6 file lokasi yaitu c3/n0.java, com/onesignal/OSUtils.java, f3/b.java, h8/a.java, h8/b.java, i8/a.java. *Random Number Generator* adalah alat atau algoritma yang menghasilkan urutan angka yang secara statistik independen dan tidak dapat ditebak. Kerentanan pada *Random Number Generator* memiliki standar pengujian yaitu CWE (*Common Weakness Enumeation*) terdeteksi di tingkat CWE-330 yang menunjukkan bahwa penggunan *random number generator* yang tidak sesuai dengan porprosinya atau tidak memadai dalam konteks keamanan yang bergantung pada angka yang tidak dapat diprediksi. Selain itu, terdapat juga dua standar pengujian lainnya yaitu *Open Web Application Security Project* (OWASP) Top 10 terdeteksi M5:kriptografi tidak memadai. *The Mobile Application Security Verification Standard* terdeteksi MSTG-CRYPTO-6 yang berarti semua nilai acak dihasilkan menggunakan generator nomor acak yang cukup aman.

```
private a(int i10, Random random) {
    this(a(i10, random), random);
}

private a(int[] iArr, Random random) {
    this.f4348b = iArr;
    this.f4347a = random;
    this.f4349c = new int[iArr.length];
    for (int i10 = 0; i10 < iArr.length; i10++) {
        this.f4349c[iArr[i10]] = i10;
    }
}

private static int[] a(int i10, Random random) {
    int[] iArr = new int[i10];
    int i11 = 0;
    while (i11 < i10) {
        int i12 = i11 + 1;
        int nextInt = random.nextInt(i12);
        iArr[i11] = iArr[nextInt];
        iArr[nextInt] = i11;
        i11 = i12;
    }
    return iArr;
```

Gambar 6. Kode pada c3/n0.java

Pada Gambar 6. menunjukkan bahwa terdapat kelas java yaitu java.util.Random yang berfungsi untuk membuat *random number* pada sebuah *access modifiers* berfungsi untuk memberikan suatu hak kepada *user* berupa akses. Akses tersebut bersifat terbatas dan tidak semua data yang berada di dalam suatu kelas atau turunannya dapat diakses oleh *user*. Salah satu akses yang bersifat *private* yang pengaksessan suatu *class* yang hanya dapat diakses oleh *class* dimana *class* ini dibuat. *Access Modifiers* termasuk suatu data yang bersifat penting yang keberadaannya harus dilindungi secara aman dan menggunakan kunci yang tidak dibuat dengan *random number generator* yang tidak cocok dan tidak memadai sehingga kecil kemungkinan mendapatkan suatu serangan.

Gambar 7. Hasil Analisis *Weak Crypto The Encryption Mode* CBC *with* PKCS5/PKCS7 *Padding* Aplikasi HIMFO oleh MobSF

Berdasarkan Gambar 7 terdapat sebuah *weak crypto* dengan *high severity* pada *The Encryption Mode* CBC *with* PKCS5/PKCS7 *Padding* yang terdapat pada satu file lokasi saja yaitu h3/a.java. Dalam hal ini, aplikasi *mobile* Android HIMFO

menggunakan *mode encryption* CBC dengan *padding* PKCS5/PKCS7 yang lemah atau rentan terhadap suatu serangan *oracle padding*. *Advanced Encryption Standard* (AES) adalah algoritma kriptografi yang menjadi standar algoritma enkripsi kunci simetris pada saat ini. *Cipher Block Chaining* (CBC) adalah salah satu mode operasi dalam enkripsi blok yang digunakan pada algoritma kriptografi dan suatu mode dimana blok yang satu dengan blok lain saling terkait (*chained*) dan juga memastikan setiap blok *plaintext* dienkripsi dengan memperhitungkan blok *cipherteks* sebelumnya. *Public Key Cryptographic Standard* (PKCS) adalah penambahan data pada awal, tengah, atau akhir sebelum enkripsi, nilai tiap bit yang ditambahkan adalah banyak bit yang ditambahkan.

Kerentanan pada *Mode Encyrption* CBC *with* PKCS5/PKCS7 *Padding* memiliki standar pengujian yaitu *Common Weakness Enumeration* (CWE) terdeteksi CWE-649:ketergantungan pada kebingungan atau enkripsi input keamanan yang relevan tanpa pemeriksaan integritas, *Open Web Application Security Project* (OWASP) Top 10 terdeteksi M5:kriptografi tidak memadai, OWASP *The Mobile Application Security Verification Standard* (MASVS) terdeteksi MSTG-CRYPTO-3 artinya aplikasi menggunakan kriptografi kuno yang sesuai untuk tujuan tertentu dan dikonfigurasi dengan parameter yang sesuai dengan praktik industri terbaik.

```
protected Cipher s() {
    return Cipher.getInstance("AES/CBC/PKCS7Padding");
}
```

Gambar 8. Kode pada h3/a.java

Pada Gambar 8. terdapat sebuah kode yang berarti membuat sebuah *cipherinstance* dengan cara memanggil getInstance() sebagai metodenya dan memasukkan parameter untuk algoritma enkripsi yang digunakan, yaitu AES (*Advanced Encryption Standard*) dengan mode operasi CBC *padding* PKCS5. Untuk membuat *cipher*, pertama-tama masukkan kelas Java, javax.crypto.Cipher. PKCS5 dan PKCS7 identik, tetapi PKCS5 hanya digunakan untuk penyandian blok berukuran 64 bit. Keduanya dapat diterapkan secara praktis.

4.1.2. SSL *Bypass*

Pada aplikasi *mobile* Android HIMFO terdapat URL dengan protokol jaringan *Hypertext Transfer Protocol* (HTTP) dan *Hypertext Transfer Protocol Secure* (HTTPS). HTTPS (*Hypertext Transfer Protocol Secure*) adalah versi aman dari protokol HTTP yang menggunakan SSL/TLS untuk enkripsi dan otentikasi. SSL (*Secure Sockets Layer*) adalah sebuah protokol keamanan internet yang digunakan untuk melindungi data yang dikirimkan melalui internet.

Gambar 9. Hasil Analaisis SSL *Bypass* Aplikasi HIMFO oleh MobSF

Pada Gambar 9 terdapat URL http://dashif.org/guidelines/last-segment-number data:cs:audiopurposecs:2007 yang menggunakan protokol jaringan HTTP. URL tersebut mengandung sebuah *segment* metadata yang sangat dianjurkan untuk menggunakan HTTPS yang terdapat sebuah SSL karena untuk video atau audio *streaming* metadata yang berfungsi untuk memberikan informasi tentang file audio atau video. Metadata membutuhkan sebuah perlindungan dari *Secure Socket Layer* (SSL) agar pihak ketiga tidak bisa mengaksesnya. Adapun kode metadata pada aplikasi *mobile* Android HIMFO dapat dilihat pada Gambar 10. di bawah ini.

```
protected int c0(List<e> list) {
    int q02;
    int i10 = 0;
    for (int i11 = 0; i11 < list.size(); i11++) {
        e eVar = list.get(i11);
        if (v4.b.a("urn:mpeg:dash:role:2011", eVar.f7469a)) {
            q02 = d0(eVar.f7470b);
        } else if (v4.b.a("urn:tva:metadata:cs:AudioPurposeCS:2007", eVar.f7469a)) {
            q02 = q0(eVar.f7470b);
        }
        i10 |= q02;
    }
    return i10;
}
```

Gambar 10 Kode pada g3/d.java

4.1.3. *Dangerous Permissions*

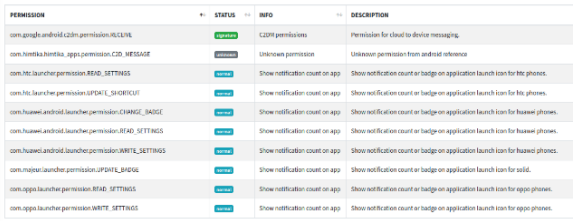
Pada aplikasi *mobile* Android HIMFO terdapat 22 status perizinan yang bersifat normal, 1 status perizinan yang bersifat *signature*, dan 3 status perizinan yang bersifat *unknown*. Perizininan yang berstatus normal berarti izin ini merupakan bagian dari grup izin yang secara umum diberikan kepada aplikasi tanpa meminta izin secara khusus dari pengguna.

Selain itu, terdapat 1 status perzininan yang bersifat *signature* yaitu "com.google.android.c2dm.permission.RECEIVE" yang merujuk pada izin yang diperlukan oleh aplikasi Android untuk menerima pesan *cloud* melalui layanan pesan push dari Google (Google Cloud Messaging atau GCM). Izin ini digunakan oleh aplikasi untuk menerima pemberitahuan, pembaruan, atau pesan lainnya dari server melalui layanan push yang disediakan oleh Google.

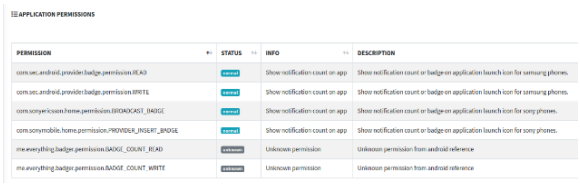
Perizinan yang bersifat *unknown* disebabkan oleh istem Android mungkin tidak memiliki informasi yang cukup untuk menentukan status yang tepat untuk izin tersebut, mungkin karena konfigurasi khusus atau situasi yang tidak biasa. Terdapat 3 status perizinan yang bersifat *unknown* yaitu: com.himtika.himtika_apps.permission.C2D_MESSAGE;
me.everything.badger.permission.BADGE_COUNT_READ;

me.everything.badger.permission.BADGE_COUNT_WRITE

Gambar 11. Hasil Analisis *Dangerous Permissions Slide* 1 Aplikasi HIMFO oleh MobSF



Gambar 12. Hasil Analisis *Dangerous Permissions Slide* 2 Aplikasi HIMFO oleh MobSF



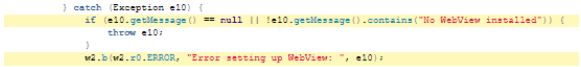
Gambar 13. Hasil Analisis *Dangerous Permissions Slide* 3 Aplikasi HIMFO oleh MobSF

4.1.4. *Root Detection*

Gambar 14. Hasil Analisis *Root Detection* Aplikasi HIMFO oleh MobSF

Berdasarkan Gambar 14. terdapat sebuah *root detection* dengan *high severity* pada *Remote WebView Debugging is Enabled* yang terdapat pada satu file lokasi yaitu com/onesignal/i4.java. *Remote WebView Debugging* adalah proses di mana pengembang dapat melakukan debug atau pemecahan masalah (*troubleshooting*) pada *WebView* di aplikasi Android dari jarak jauh, biasanya dari komputer pengembang. Ini memungkinkan pengembang untuk mengamati, menginspeksi, dan menguji tampilan dan perilaku konten web yang dimuat di dalam *WebView* aplikasi Android tanpa harus mengakses perangkat fisik secara langsung.

Kerentanan pada *Remote WebView Debugging is Enabled* memiliki standar pengujian CWE (*Common Weakness Enumeration*) di level CWE-919: *Weakness in Mobile Applications* CWE-919 menunjukkan ketidaksempurnaan dalam cara aplikasi memproses atau mengonstruksi URL yang digunakan dalam permintaan ke server web. Hal ini dapat dimanfaatkan oleh penyerang untuk menyusun URL dengan karakter khusus yang tidak diizinkan, seperti karakter *encoding* yang tidak valid, karakter kontrol, atau karakter yang dapat digunakan untuk serangan injeksi, seperti tanda kutip (') atau karakter delimiter lainnya seperti yang terdapat pada Gambar 15 yang menunjukkan bahwa terjadi *error* pada installasi *webview* nya.



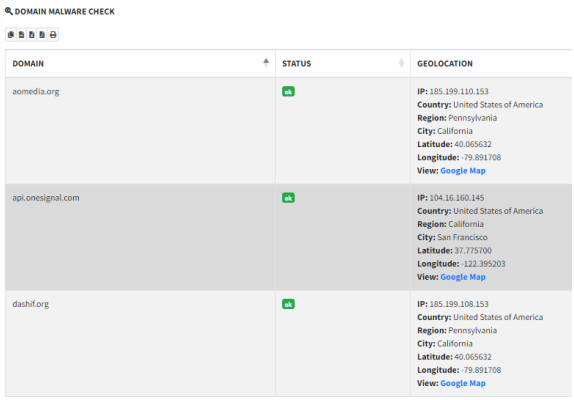
```
} catch (Exception e10) {
    if (e10.getMessage() == null || !e10.getMessage().contains("No WebView installed")) {
        throw e10;
    }
    w2.b(w2.r0.ERROR, "Error setting up WebView: ", e10);
}
```

Gambar 15. Kode pada com/onesignal/i4.java

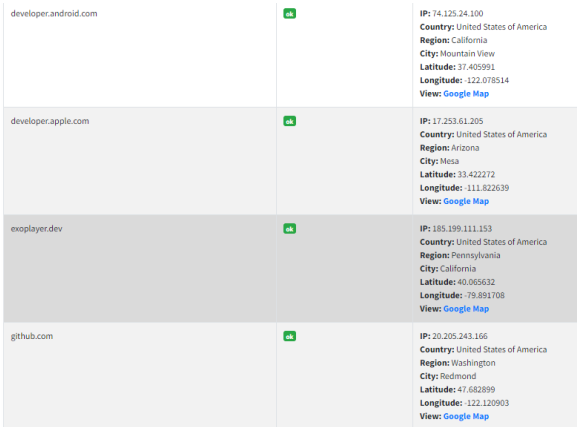
Selain itu, standar pengujian lainnya menampilkan informasi bahwa terdapat OWASP Top 10 M1: *Improper Platform Usage* dan juga terdapat dalam OWASP MASVS: MSTDG *Resilience* 2 merujuk pada salah satu dari banyak persyaratan keamanan yang terdapat dalam MASVS yang berkaitan dengan ketangguhan (*resilience*) aplikasi *mobile*.

4.1.5. *Domain Malware Check*

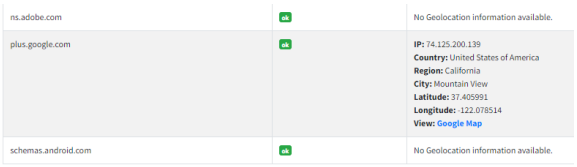
Pada aplikasi *mobile* Android HIMFO tidak terdapat *domain malware* karena semua domain berstatus OK atau *Good* yang berarti domain pada aplikasi *mobile* Android HIMFO tidak terindikasi *malware* seperti yang tertera pada 4 gambar di bawah.



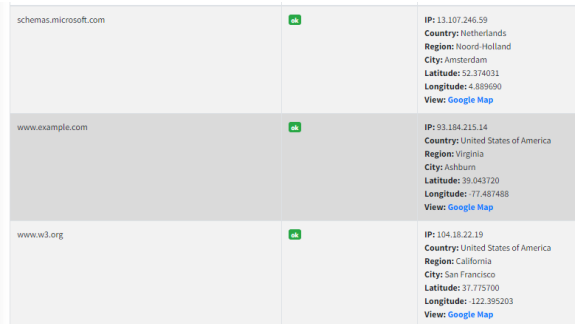
Gambar 16. Hasil Analisis *Domain Malware Check Slide* 1 Aplikasi HIMFO oleh MobSF



Gambar 17. Hasil Analisis *Domain Malware Check Slide* 2 Aplikasi HIMFO oleh MobSF



Gambar 18. Hasil Analisis *Domain Malware Check Slide* 3 Aplikasi HIMFO oleh MobSF

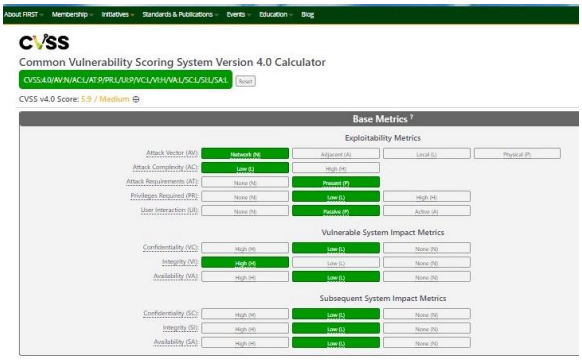


Gambar 20. Hasil Analisis *Domain Malware Check Slide* 4 Aplikasi HIMFO oleh MobSF

4.1.6. *Common Vulnerability Scoring System* (CVSS)

Setelah melakukan pengujian keamanan menggunakan *tools Mobile Security Framework* (MobSF), selanjutnya adalah menghitung nilai kerentanan menggunakan CVSS (*Common Vulnerability Scoring System*). Pada aplikasi Mobile Android HIMFO memiliki nilai kerentanan CVSS 5.9 atau dalam tingkat medium seperti yang tertera pada Gambar 4.33 di bawah. Hal tersebut hasil dari kalkulasi berdasarkan 7 bagian besar dalam CVSS, yaitu:

a. *Attack Vector* (AV): Mendeskripsikan bagaimana serangan dapat diinisiasi, apakah melalui jaringan (*network*), melalui jarak jauh (*adjacent network*), atau lokal (*local*).
b. *Attack Complexity* (AC): Mendeskripsikan seberapa sulitnya meng-exploitasi kerentanan, apakah memerlukan kondisi khusus atau tidak.
c. *Privileges Required* (PR): Mendeskripsikan tingkat akses yang diperlukan oleh penyerang untuk mengeksploitasi kerentanan.
d. *User Interaction* (UI): Mendeskripsikan apakah interaksi pengguna diperlukan untuk mengeksploitasi kerentanan.
e. *Confidentiality* (C), *Integrity* (I), dan *Availability* (A) *Impact*: Mendeskripsikan dampak terhadap kerahasiaan, integritas, dan ketersediaan data saat kerentanan dieksploitasi.



Gambar 21. Nilai Kerentanan Aplikasi HIMFO oleh CVSS

4.2. Rekomendasi Perbaikan

Peneliti membuat rekomendasi tentang masalah penelitian berdasarkan temuan penelitian. Hasil penelitian ini dapat digunakan oleh pengembang dan pengguna aplikasi HIMFO untuk mempertimbangkan analisis statik *Mobile Security Framework* (MobSF).

4.2.1. *Weak Crypto*

Rekomendasi perbaikan untuk pengembang aplikasi *mobile* Android HIMFO untuk menggunakan generator angka acak yang aman (*SecureRandom*), yang dapat menginisialisasi kunci kriptografis yang dibuat oleh *KeyGenerator*. Penggunaan kunci yang tidak dibuat dengan generator angka acak yang aman untuk data penting dapat melemahkan algoritma dan memungkinkan serangan. Dibandingkan dengan SHA-1, SHA-3 lebih tahan terhadap serangan *brute-force* karena membutuhkan waktu lebih lama untuk mendapatkan karakter *plaintext* 8, 9 dan 10 [8].

4.2.2. SSL *Bypass*

Rekomendasi perbaikan pengembang aplikasi *mobile* Android HIMFO untuk menghindari menggunakan protokol jaringan *Hypertext Transfer Protocol* (HTTP) untuk data pribadi seperti video yang dikhususkan untuk pelanggan berbayar atau premium, data *login*, dan profil pengguna. Hal ini disebabkan oleh *Secure Socket Layer* (SSL) yang sangat terenkripsi sehingga pihak ketiga tidak dapat mengaksesnya [9].

4.2.3. *Dangerous Permissions*

Rekomendasi bagi pengguna aplikasi untuk memilih opsi hanya kali ini agar perizininan pada aplikasi tersebut tidak selalu hidup walaupun tidak sedang digunakan. Selain itu, rekomendasi lain bagi pengguna yaitu aplikasi untuk izin akses penyimpanan dan daftar akun, apabila pengguna merasa izin ini tidak begitu diperlukan, sebaiknya pengguna menonaktifkan izin tersebut agar kemungkinan data pada penyimpanan dan data pribadi lainnya disalahgunakan tidak terjadi.

4.2.4. *Root Detection*

Rekomendasi untuk pengembang aplikasi *mobile* Android HIMFO: Matikan *remote debugging* pada *WebView* secara *default* di lingkungan produksi, konfigurasi untuk tidak mengizinkan *remote debugging* kecuali dalam mode pengembangan atau uji coba. Validasi dan sanitasi *input* URL *WebView* harus dilakukan, hindari URL dengan karakter khusus yang dapat dimanfaatkan oleh penyerang. Batasi akses pengguna untuk mengaktifkan atau mengonfigurasi *Remote WebView Debugging* hanya kepada pengguna berwenang. Lakukan pemantauan terhadap aktivitas *Remote WebView Debugging* yang diaktifkan, termasuk pencatatan akses dan penggunaan fitur untuk mendeteksi potensi penyalahgunaan atau aktivitas mencurigakan.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian dengan menggunakan *Mobile Security Framework* (MobSF) pada Aplikasi HIMFO ditemukan 3 kerentanan pada *Weak Crypto* dengan 1 *high severity* dan 2 *warning severity*. Pada SSL *Bypass* ditemukan 1 URL yang tidak menggunakan SSL sedangkan URL metadata tersebut harus menggunakan HTTPS SSL agar dapat terlindungi dengan aman. Pada *Dangerous Permissions* terdapat 22 status perizinan normal, 1 status perizinan *signature*, dan 3 status perizinan unknown. Pada *Root Detection* terdapat 1 kerentanan dengan *high severity*. Pada *Domain Malware Check* semuanya dalam kondisi baik. Ditemukan juga *Security Score* nya yaitu 30/100 dan *Trackers Detectionnya* yaitu 1/433. Selain itu, berdasarkan perhitungan yang dilakukan oleh CVSS (*Common Vulnerability Scoring System*) aplikasi *mobile* Android HIMFO memiliki tingkat keamanan medium atau 5.9.

Saran yang diberikan untuk penelitian selanjutnya dan untuk pengetahuan adalah sebagai berikut: Pada penelitian ini hanya menggunakan *tools Mobile Security Framework* dengan analisis statik. Diharapkan untuk penelitian selanjutnya dapat menggunakan OWASP *Mobile Top* 10 atau OWASP MASV MSTG agar pengujian keamanannya dapat lebih mendalam. Pengujian yang dilakukan hanya mencakup pembuatan dokumen hasil pengujian evaluasi keamanan dan rekomendasi untuk perbaikan sehingga penelitian ini tidak melanjutkan atau memeriksa hasil rekomendasi perbaikan tersebut, diharapkan pada penelitian selanjutnya dapat langsung memonitor rekomendasi perbaikan tersebut. Untuk mencegah kesalahpahaman antara kedua belah pihak selama proses pengujian, penguji dan pihak yang melakukan pengujian harus mencapai perjanjian yang disepakati bersama.

DAFTAR PUSTAKA

[1] R. Abdillah, A. A. Trinoto, and I. Himawan, "Ciptaan disebarluaskan di bawah Lisensi Creative Commons Atribusi 4.0 Internasional. STATIC ANALYSIS USING MOBILE SECURITY FRAMEWORK FOR SMART HOME APPLIANCES," *J. Inf. Syst. Applied, Manag. Account. Res.*, vol. 7, no. 3, pp. 760–765, 2023, doi: 10.52362/jisamar.v7i3.1161.

[2] A. D. Pratama and Amiruddin, "Uji Keamanan Aplikasi ABC Milik Instansi XYZ Menggunakan OWASP Mobile Security Testing Guide," *Info Kripto*, vol. 15, no. 3, pp. 113–121, 2021, doi: 10.56706/ik.v15i3.26.

[3] M. Mansyuruddin, I. Astuti, and E. Enawaty, "Penggunaan Media Pembelajaran Berbasis Android pada Mata Pelajaran Perawatan Sistem Injeksi Sepeda Motor Kelas XI di SMK Negeri 9 Pontianak," *JIIP - J. Ilm. Ilmu Pendidik.*, vol. 6, no. 4, pp. 2183–2187, 2023, doi: 10.54371/jiip.v6i4.1734.

[4] F. Nurindahsari and B. Parga Zen, "Analisis Statik Keamanan Aplikasi Video Streaming Berbasis Android Menggunakan Mobile Security Framework (Mobsf)," *Cyber Secur. dan Forensik Digit.*, vol. 4, no. 2, pp. 63–80, 2022, doi: 10.14421/csecurity.2021.4.2.3373.

[5] T. Yuniati, A. R. Tambunan, and Y. A. Setyoko, "Implementasi Static Analysis Dan Background Process Untuk Mendeteksi Malware Pada Aplikasi Android Dengan Mobile Security Framework," *LEDGER J. Inform. Inf. Technol.*, vol. 1, no. 2, pp. 24–28, 2022, doi: 10.20895/ledger.v1i2.848.

[6] I. K. A. O. Ardita, I. G. N. Anom Cahyadi Putra, M. R. Kustiadie, G. N. M. Dika Varuna, and M. Y. Eka Prananda, "Analisis Keamanan Aplikasi Android Dengan Metode Vulnerability Assessment," *JELIKU (Jurnal Elektron. Ilmu Komput. Udayana)*, vol. 10, no. 3, p. 279, 2022, doi: 10.24843/jlk.2022.v10.i03.p04.

[7] Candra Kurniawan and N. Trianto, "Security Assessment Pada Aplikasi Mobile Android XYZ Dengan Mengacu Pada Kerentanan OWASP Mobile Top Ten 2016," *Info Kripto*, vol. 15, no. 1, pp. 11–18, 2021, doi: 10.56706/ik.v15i1.2.

[8] F. Kurniawan, A. Kusyanti, and H. Nurwarsito, "Analisis dan Implementasi Algoritma SHA-1 dan SHA-3 pada Sistem Autentikasi Garuda Training Cost," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 1, no. 9, pp. 803–812, 2017, [Online]. Available: http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/247

[9] D. Prayama, Yuhefizar, and Amelia Yolanda, "Protokol HTTPS, Apakah Benar-benar Aman?," *J. Appl. Comput. Sci. Technol.*, vol. 2, no. 1, pp. 7–11, 2021, doi: 10.52158/jacost.v2i1.118.