BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Sebelumnya

Pada subbab ini berisi tentang keterkaitan penelitian-penelitian sebelumnya dengan penelitian yang dilakukan oleh peneliti, baik tentang tema yang berkaitan atau metode yang digunakan. Berikut tabel 2.1 menunjukkan data penelitian sebelumnya.

Tabel 2.1 Data Penelitian Sebelumnya

No.	Judul Penelitian	Nama Peneliti (Tahun)	Masalah, Metode	Hasil
1.	Analisis dan Deteksi *Malware* dengan Metode *Hybrid Analysis* Menggunakan *Framework* MobSF	Edward Tansen dan Deris Wahyu Nurdiarto (2020)	Mencari hal-hal yang janggal pada malware yang menginfeksi perangkat android, dengan metode analisis hybrid menggunakan *Mobile Security Framework* (MobSF).	Mendapatkan beberapa karakteristik dari *malware Bouncing Golf* dan *Riltok*. Kedua *malware* memiliki karakteristik yang sama, yaitu data akan dikirimkan ke server C&C (*Command*& *Control Server*).
2.	Analisis Statis Menggunakan *Mobile Security Framework* Untuk Pengujian Keamanan Aplikasi *Mobile E-Commerce* Berbasis Android	Cholis Hanifurohman dan Deanna Durbin Hutagalung (2020)	Mencari celah-celah keamanan pada aplikasi *mobile e-commerce*, dengan metode analisis statik menggunakan *Mobile Security Framework* (MobSF).	SP, TP, BL, LZ dan SR yang merupakan lima besar *mobile e-commerce* berbasis android paling populer di Indonesia menunjukkan bahwa beberapa celah keamanan masih ada dengan tingkat kemanan yang relatif sama.

No.	Judul Penelitian	Nama Peneliti (Tahun)	Masalah, Metode	Hasil
3.	*A Comprehensive Analysis of the Android Permissions System*	Iman m. Almomani dan Aala Al Khayer (2020)	Mencari masalah keamanan dan cara aplikasi menangani informasi dan sumber daya berisiko tinggi dari sistem izin android.	Ditemukan beberapa ancaman keamanan yang dapat membuka OS untuk berbagai serangan, seperti serangan tabrakan, eskalasi, TOCTOU, dan ransomware.
4.	Membangun Sistem Pengujian Keamanan Aplikasi Android Menggunakan MobSF	Aan Kartono, Anang Sularsa, Setia Juli Irzal Ismail (2019)	Banyaknya aplikasi android ada beberapa aplikasi yang terdeteksi didalamnya terdapat malware berbahaya, menggunakan *Mobile Security Framework* (MobSF).	Dari sampel ditemukan bahwa *Blackmart* yaitu sebuah *malware* bertipe *adware*. Krep.itmtd.ywtjexf yaitu sebuah *malware* bertipe trojan. Earthquake bukan sebuah malware.
5.	Implementasi Digital Rights Management pada MediaStreaming sebagai Pelindung Data Digital	Surya Michrandi Nasution, R, Rumani M, Agus Virgono (2016)	Banyaknya Pembajakan pada konten digital, menggunakan *Digital Rights Management* (DRM).	Hasil dekripsi dari video terenkripsi bernilai 2,59 yang berarti hasil pengembalian masih mirip atau serupa dengan video aslinya.

2.2 Dasar Teori

2.2.1 Aplikasi *Video Streaming*

Streaming adalah suatu teknologi untuk menjalankan file audio atau video secara langsung menggunakan jaringan internet maupun lokal [13].

File audio atau video yang terletak pada sebuah *server* dapat secara langsung dijalankan pada komputer *client* sesaat setelah ada permintaan dari pengguna sehingga proses *download* file tersebut yang membutuhkan waktu cukup lama dapat dihindari. Saat file tersebut diputar maka akan terbentuk sebuah *buffer* di komputer *client* dan data audio atau video tersebut akan mulai di-*download* ke dalam *buffer* yang telah terbentuk pada mesin *client*. Setelah *buffer* terisi dalam waktu beberapa detik, maka secara otomatis file video ataupun audio akan di jalankan oleh sistem. Sistem akan membaca informasi dari *buffer* sambil tetap melakukan proses *download* file sehingga proses *streaming* tetap berlangsung ke mesin *client*.

Secara garis besar, konsep *video streaming* dibagi ke dalam tiga tahap, antara lain:

1. Mempartisi atau membagi data video yang telah terkompresi ke dalam paket - paket data.
2. Pengiriman paket - paket data video.
3. Pihak penerima (*client*) mulai men-*decode* dan menjalankan video walaupun paket data yang lain masih dalam proses pengiriman ke PC *client* [14].

Video streaming yaitu layanan transmisi video dan audio melalui internet. Layanan ini di-*broadcast* kepada banyak pengguna yang mengakses suatu situs video *online*. Banyak situs yang menyediakan fasilitas *video streaming*, sehingga pengguna dapat dengan mudah menonton video yang diinginkan secara *online*. Ada tiga tipe *video streaming* menurut bentuk layanannya, yaitu:

1. *Video on Demand* adalah layanan yang mengizinkan pengguna untuk dapat melakukan proses *rewind*, *pause* dan *fast forward*.
2. *Live Streaming* adalah aplikasi yang mengizinkan pengguna untuk menerima siaran radio dan televisi secara langsung.

3. *Real Time Streaming* adalah aplikasi yang mengizinkan pengguna untuk berkomunikasi video dan audio dalam *real time* [15].

Menurut jurnal [16] komponen-komponen dalam media *streaming* yaitu:

1. *Media Source*.

Media Source yaitu sumber yang akan menampilkan suatu konten presentasi. *Media source* dapat berupa sumber yang sifatnya *live*, seperti *microphone* dan kamera video.

2. *Encoder*.

Encoder yaitu program yang digunakan untuk mengubah *media source* ke format yang sesuai untuk *streaming*. *Encoder* memiliki kompresi yang cukup tinggi untuk mengatasi keterbatasan bandwidth jaringan.

3. Media *Server*.

Media *server* digunakan untuk mendistribusikan *video streaming* ke *client* dan bertanggung jawab untuk mencatat semua aktivitas *streaming*, yang nantinya digunakan untuk statistik.

4. *Player*.

Player digunakan untuk menampilkan atau mempresentasikan konten multimedia (*data stream*) yang diterima dari multi *server*. File-file khusus yang disebut metafile digunakan untuk mengaktifkan *player* dari halaman web. Metafile berisi keterangan dari konten multimedia. Browser web men-*download* dan meneruskan ke *player* yang tepat untuk mempresentasikannya serta berfungsi untuk melakukan dekompresi.

2.2.2 Mobile Security Framework (MobSF)

Mobile Security Framework (MobSF) adalah *framework* yang digunakan untuk pengujian penetrasi terhadap aplikasi seluler (android/iOS/windows) otomatis yang mampu melakukan analisis statis, dinamis, dan *malware*. MobSF digunakan untuk analisis keamanan yang efektif dan cepat dari aplikasi seluler dan mendukung kedua binari (*Android Package Kit* (APK), *iPhone Application* (IPA) & APPX yang merupakan format file *windows store*) dan kode sumber zip. MobSF

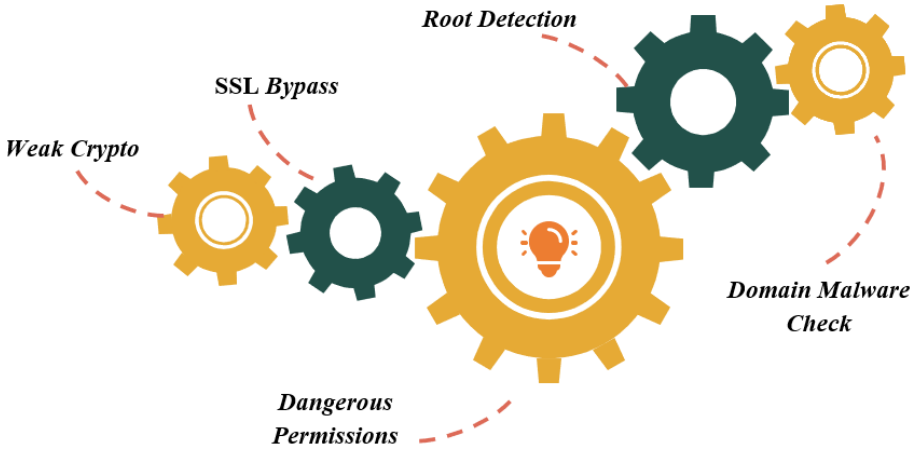
dapat melakukan pengujian aplikasi dinamis saat *runtime* untuk aplikasi Android dan memiliki kemampuan *fuzzing* API Web yang didukung oleh CapFuzz, pemindai keamanan khusus Web API. *Fuzzing* merupakan suatu metode mencari kesalahan piranti lunak dengan menyediakan input yang tidak diduga lalu mengamati hasilnya [17]. MobSF dirancang untuk membuat integrasi *Continuous Integration*/ *Continuous Delivery* (CI/CD) atau *Development, Security, and Operations* (DevSecOps) secara bagus [18]. CI/CD merupakan metode untuk mengirimkan aplikasi ke pelanggan secara rutin dengan memperkenalkan otomatisasi ke dalam tahapan pengembangan aplikasi [19]. DevSecOps adalah kerangka kerja kolaborasi yang memperluas dampak *Development and Operations* (DevOps) dengan menambahkan praktik keamanan ke proses pengembangan dan pengiriman perangkat lunak [20].

Mobile Security Framework adalah kerangka kerja gabungan yang melakukan analisis statis dan dinamis dari sebuah APK. MobSF dikembangkan berdasarkan Python. Untuk menghasilkan laporan MobSF menggunakan html. *Mobile Security Framework* (MobSF) menjalankan server lokal melalui baris perintah di komputer *host* [21]. Semua alat analisis statis dan dinamis yang ada melakukan analisis menggunakan *DroidMon-Dalvik Monitoring Framework* [22] dan *Xposed Module Repository*. Xposed adalah kerangka kerja untuk modul yang dapat mengubah perilaku sistem dan aplikasi tanpa menyentuh APK apa pun. Cara kerja Xposed bagus karena itu berarti modul dapat bekerja untuk versi yang berbeda dan bahkan ROM tanpa perubahan apa pun (selama kode aslinya tidak terlalu banyak diubah) [23].

2.2.3 Analisis Statik

Metode statis analis dilakukan tanpa benar-benar menjalankan programnya dan lebih seperti menyelidiki apa yang terjadi pada *source code* dengan tujuan utama yaitu untuk mengetahui kode berbahaya seperti apa yang tertanam dalam aplikasi tersebut [24].

Analisis statis mengacu pada dekompilasi APK aplikasi ke file Java dan *Extensible Markup Language* (XML) yang sesuai. Untuk melakukan analisis statis, penganalisis statis harus memiliki fitur untuk memeriksa XML dengan benar dan presisi. File Java dapat diekstraksi dengan dekompiler DEXtoJar [25]. Alat analisis Statis mendekompilasi kode APK ke format yang dapat dibaca manusia. Sehingga penganalisa dapat membaca kode dan mengidentifikasi kerentanan yang mungkin dimiliki aplikasi [21]. Dari hasil analisis statis maka didapat hasil daftar *malwords* yang sering muncul sebagai daftar string berbahaya dengan tingkat resiko masing-masing, setelah mendapatkan daftar malwords, peneliti akan menganalisisnya [26]. Gambar 2.1 menunjukkan parameter pada analisis statik.



Gambar 2.1 Parameter pada Analisis Statik

1. *Weak Crypto*

Analisis *weak crypto* dilakukan dengan acuan ada atau tidaknya implementasi algoritma kriptografi yang lemah atau penggunaan algoritma kriptografi yang sudah usang atau sudah dianggap tidak layak.

2. SSL *bypass*

Analisis SSL *bypass* dilakukan dengan melakukan cek ada atau tidaknya *service* yang melibatkan protokol http yang tidak mewajibkan penggunaan SSL sebagai persyaratan keamanan transaksi dalam protokol http menggunakan SSL seperti mengizinkan http di manifest, atau terdapat string berkonten http:// yaitu *weak implementation*.

3. *Dangerous permissions*

Aplikasi Android dibangun untuk melakukan serangkaian tindakan, beberapa di antaranya memerlukan izin dari pengguna. Analisis terhadap *permission* dilakukan dengan melihat pada seberapa banyak *dangerous permissions* yang digunakan oleh aplikasi.

4. *Root Detection*

Analisis *root detection* dilakukan dengan melakukan cek ada atau tidaknya aplikasi mempunyai fungsi untuk melakukan deteksi akses *root* terhadap perangkat android yang digunakan. Dimana akses memungkinkan untuk akses langsung ke dalam sistem termasuk data-data yang dimiliki aplikasi.

5. *Domain Malware Check*

Analisis *domain malware check* dilakukan dengan melakukan cek ada atau tidaknya domain-domain yang terdapat dalam aplikasi terindikasi dalam kategori domain yang mengandung *malware* atau tidak [27].

2.2.4 Keamanan dalam Android

Sistem operasi android memiliki teknik pengamanan multilayer untuk melindungi data dari pengguna. Terdapat beberapa tujuan dari pengamanan tersebut:

1. Melindungi data dari pengguna.
2. Melindungi *system resource*.
3. Menjamin satu aplikasi tidak akan mengakses data milik aplikasi lainnya [28].

Kejadian *malwords* sering terjadi di aplikasi Android. Selain itu, penyerang dapat menggunakannya untuk tujuan distorsi. Misalnya, izin READ_CONTACTS memungkinkan aplikasi untuk membaca semua kontak yang tersimpan di telepon, tetapi pemrogram yang buruk dapat mengambilnya untuk mencuri data pengguna. Selain itu, untuk izin ACCESS_FINE_LOCATION, izin ini memungkinkan pengaksesan sumber lokasi yang baik, seperti Sistem Pemosisian Global. Lokasi pengguna dapat diketahui [29]. Table 2.2 menunjukkan daftar *malwords*.

Table 2.2 Daftar *Malwords*[29]

No.	String Berbahaya	Tingkat Resiko
1	READ_SMS	Moderat-Tinggi
2	SEND_SMS	Tinggi
3	RECEIVE_SMS	Tinggi
4	WRITE_SMS	Tinggi
5	PROCESS_OUTGOING_CALLS	Sangat Tinggi
6	MOUNT_UNMOUNT_FILESYSTEMS	Moderat
7	READ_HISTORY_BOOKMARKS	Medium-Tinggi
8	WRITE_HISTORY_BOOKMARKS	Moderat-Tinggi
9	READ_LOGS	Sangat Tinggi
10	INSTALL_PACKAGES	Sangat Tinggi
11	READ_PHONE_STATE	Moderat-Tinggi
12	READ_CONTACTS	Medium-Tinggi
13	ACCESS_FINE_LOCATION	Moderat-Tinggi

Permission Model

Pada prinsipnya, setiap aplikasi memiliki *User ID* (UID) nya masing-masing dan setiap UID tidak dapat mengakses UID lainnya sepanjang aplikasi tersebut tidak memberikan *permissions* kepada UID tersebut [28].

Arsitektur keamanan android memiliki desain inti yaitu tidak ada aplikasi yang dapat melakukan operasi apa pun yang dapat berdampak buruk bagi pengguna, aplikasi lain, atau sistem operasi itu sendiri. Oleh karena itu, permintaan aplikasi untuk mengakses komponen, data sensitif, dan fitur sistem tertentu diatur oleh sistem izin Android. Pada setiap versi android memiliki total *permission* yang berbeda, ini menunjukkan evolusi sistem perizinan Android [30]. Tabel 2.3 menunjukkan data rilis resmi platform android.

Tabel 2.3 Data Rilis Resmi Platform Android

No.	Nama Kode Rilis	Versi Android	Level API	Berbahaya	Normal	Signature	Signature or System	Total Permission
1.	Android 11 Beta	11	30	30	47	48	42	167
2.	Q	10	29	30	45	42	41	158
3.	Pie	9	28	27	42	38	41	148
4.	Oreo	8.0 – 8.1	26–27	26	39	38	41	144
5.	Nougat	7.0 – 7.1	24 – 25	24	35	35	41	135
6.	Marshmallow	6	23	24	35	32	40	131

Sampai dengan Android 6.0 pengguna harus memberikan izin pada saat proses instalasi dengan menerima seluruhnya atau membatalkan instalasi. Tetapi sejak Android 6.0, pemberian izin dilakukan ketika aplikasi sedang berjalan. Sistem *permissions* yang baru ini memberikan pengendalian yang lebih baik atas fungsionalitas aplikasi dengan memberikan opsi pada pengguna untuk memilih *permissions*. Pengguna dapat mencabut *permissions* tersebut setiap saat, dengan memasuki *tab setting* dari satu aplikasi.

Dengan kata lain, jika terdapat aktivitas *malicious* yang disebabkan oleh *malware*, maka kondisi tersebut disebabkan pengguna memberikan *permission* atas malware untuk mengakses *system resource* dan data [28].