BAB 2
LANDASAN TEORI

2.1 Android

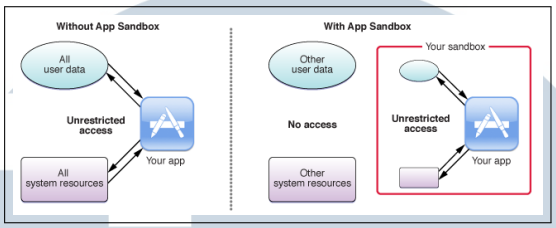
Android merupakan sistem operasi dan perangkat lunak untuk perangkat seluler yang diluncurkan oleh Google pada tahun 2007 dan kemudian diakui oleh Open Handset Alliance. Platform berbasis pada kernel Linux ini memungkinkan pengembang membuat kode Java terkelola dan menggunakan pustaka Java yang dikembangkan oleh Google untuk mengontrol perangkat. Android terdiri dari sistem operasi, middleware, dan aplikasi utama berbasis Linux dan Java, dan dirilis sebagai sumber terbuka di bawah lisensi Apache. Android memiliki banyak pengembang yang merancang sebuah aplikasi dari berbagai penjuru, kemudian aplikasi tersebut dapat diunduh dari situs web pihak ketiga [10]

Sejak dirilis, Android terus melakukan perbaruan dengan berfokus pada perbaikan *bug* dan penambahan fitur baru untuk menciptakan lingkungan yang lebih baik [11].

2.1.1 Android versi 14

Android versi 14 dirilis pada awal Oktober 2023, bersamaan dengan peluncuran Pixel 8 dan Pixel 8 Pro. Android versi ini memperkenalkan sistem baru untuk menarik dan melepas teks di antara aplikasi serta beberapa opsi penyesuaian asli untuk layar kunci Android, serta terus meningkatkan privasi dan keamanan penggunanya. Selain itu, Android versi 14 juga menambah sistem informasi yang lebih mendalam dan membutuhkan konteks untuk mengetahui alasan aplikasi ingin mengakses lokasi pengguna [12].

2.2 *Sandbox*



Gambar 2.1. Tampilan Sandbox

Sandbox merupakan bagian terpisah dari sistem komputer yang tidak membahayakan keseluruhan sistem apabila pengujian terhadap perangkat lunak dilakukan [13]. *Sandbox* mengisolasi aplikasi untuk mencegah pihak ketiga untuk mencuri data pengguna atau pun merusak sistem utama dalam perangkat, serta menjamin stabilitas, keamanan, dan privasi sistem utamanya, sehingga *Sandbox* dapat dikatakan memiliki fungsi sebagai lingkungan yang aman bagi program yang dijalankan.

Terdapat dua prinsip utama *Sandbox*, yaitu virtualisasi dan isolasi. Virtualisasi memungkinkan aplikasi untuk berjalan secara terpisah dari sistem utama dan membuat lingkungan tersebut terisolasi di dalam sistem. Sedangkan isolasi tersebut memastikan aplikasi yang berjalan melalui *Sandbox* tidak dapat mengakses data pengguna yang bersifat sensitif di luar *Sandbox*. Terdapat tiga jenis *Sandbox* yang biasanya digunakan, yaitu sebagai berikut [14].

1. *Sandbox* berbasis Aplikasi
   Sandbox jenis ini membuat lingkungan khusus untuk aplikasi tertentu yang memungkinkan aplikasi tersebut berjalan tanpa mengganggu sistem secara keseluruhan. Salah satu contohnya adalah *browser* yang dapat menjalankan *tab* dalam *Sandbox* terpisah.

2. *Sandbox* berbasis *Hardware*
   Sandbox jenis ini menggunakan fitur *hardware* khusus untuk membuat lingkungan terisolasi, misalnya pembuatan *hardware* seperti AMD-V yang memungkinkan penggunanya untuk menggunakan *Sandbox* tanpa harus menggunakan *software* lain.

3. *Sandbox berbasis Sistem Operasi*
   Sandbox ini biasanya digunakan untuk menjalankan aplikasi yang

memerlukan keamanan tingkat tinggi, seperti pengujian dan pengisolasian terhadap aplikasi yang mencurigakan ataupun *malware*.

Berdasarkan prinsip utama tersebut, *Sandbox* memiliki beberapa cara untuk melindungi sistem dari sebuah ancaman, yaitu sebagai berikut.

1. Mencegah penyebaran *malware* ke seluruh sistem dengan cara menyingkirkan aplikasi yang memiliki potensi bahaya dari sistem utama.
2. Melakukan pengawasan terkait aktivitas aplikasi di dalam *Sandbox*, kemudian membatasi akses terhadap sumber daya yang diperlukan untuk meminimalisir kemungkinan terjadinya infeksi pada aplikasi.
3. Melakukan pengujian kode dan aplikasi tanpa merusak sistem utama di lingkungan yang aman.
4. Memberikan lapisan perlindungan tambahan bagi aplikasi yang berjalan di lingkungan yang terisolasi dan sulit diakses oleh orang lain, dengan tujuan melindungi pengguna dari eksploitasi dan serangan yang ditargetkan pihak luar.

Sandbox memiliki beberapa keuntungan, yaitu sebagai berikut [15].

1. *Security*
   Sandbox mampu melindungi sistem dari *malware* dan ancaman *cyber* lainnya.
2. *Reliability*
   Sandbox dapat meningkatkan keandalan sistem dengan cara mencegah aplikasi dan program yang dapat mengganggu sistem utama.
3. *Privacy*
   Sandbox dapat mencegah aplikasi dan program lain mengakses data pribadi pengguna, sehingga data pengguna dapat terlindungi.
4. *Testing*
   Sandbox dapat digunakan untuk menguji perangkat lunak baru atau pun menjalankan kode yang tidak dipercaya di lingkungan yang aman.
5. *Education*
   Sandbox dapat digunakan dalam lingkup pendidikan untuk menciptakan keamanan pada lingkungan di mana siswa dapat melakukan eksperimen terhadap perangkat lunak atau teknologi baru.

2.3 *Package Manager*

Pengembang Android menggunakan *Package Manager* untuk membantu mengelola instalasi dan pembaruan aplikasi pada perangkat Android.*Package Manager* juga membantu *developer* dalam menghapus aplikasi[16].

2.4 Kotlin

Kotlin adalah bahasa pemrograman tingkat tinggi, berjenis statis, dan berorientasi objek yang dirancang oleh JetBrains. Kotlin dirancang untuk berinteroperasi sepenuhnya dengan Java. Kotlin memiliki sintaks yang lebih ringkas dibandingkan Java, yang memungkinkan kode menjadi lebih efektif. Kotlin pertama kali diperkenalkan oleh JetBrains pada tahun 2011 [17]

2.5 *Mobile Application*

Mobile application atau aplikasi seluler adalah program atau kumpulan perangkat lunak yang dijalankan pada perangkat seluler. Aplikasi ini memudahkan pengguna untuk melakukan berbagai tugas seperti berkomunikasi, mengelola file, membuat dokumen, hiburan, dan lain-lain. Aplikasi seluler memiliki dampak global yang positif, memfasilitasi kehidupan individu dan bisnis, serta membantu masyarakat berkembang dengan infrastruktur TI yang baru.[18]

2.6 Metode *Waterfall*

Metode *waterfall* merupakan pendekatan pengembangan yang menekankan langkah-langkah sistematis. Oleh karena itu, proses penciptaan sistem harus dilakukan secara berurutan, dimulai dari pengidentifikasian kebutuhan hingga tahap perawatan Langkah-langkah tersebut digambarkan sebagai air terjun dari atas ke bawah [9].

Herbert D. Benington memaparkan setiap tahap metode *waterfall* pada 29 Juni 1956 di *Symposium on Advanced Programming Method for Digital Computers* sebagai berikut [9].

1. *Requirement Analysis*

   Menganalisis kebutuhan merupakan tahapan pertama dalam metode *waterfall*. Melalui tahap ini, pengembang diminta untuk mencari tahu

kebutuhan pengguna sistem yang akan dirancang sebagai referensi ketika menentukan layanan ataupun fitur yang harus dibuat. Untuk memperoleh sejumlah informasi dan wawasan, pengembang dapat melakukan observasi, wawancara, ataupun mengikuti diskusi forum sesuai dengan target yang ingin dicapai.

2. *Design*
   Proses pengembangan dan perancangan yang didasari oleh kebutuhan pengguna merupakan tahap kedua dari metode *waterfall*.Tahap ini dilakukan untuk memperoleh gambaran mendetail terkait tampilan sistem, sehingga dapat mempermudah proses pengerjaan.

3. *Implementation and Unit Testing*
   Ketika *developer* menerapkan tahapan ketiga metode *waterfall*, maka akan terjadi proses *coding*. Setiap modul yang dibuat di dalamnya harus diperiksa dengan baik hingga modul tersebut dapat memenuhi fungsi yang telah ditetapkan dan sesuai dengan standar.

4. *Integration and System Testing*
   Tahap keempat merupakan tahap pengintegrasian setiap modul yang dibuat, kemudian melakukan uji coba terkait apa yang telah dibuat dengan tujuan memastikan sistem yang ada berfungsi sebagaimana mestinya, sekaligus menemukan beberapa kesalahan yang dapat diperbaiki.

5. *Maintenance*
   Pada tahap akhir, hal yang dapat dilakukan adalah melakukan perawatan sistem setelah sistem tersebut didistribusikan dan digunakan *user*. Tujuannya adalah untuk meningkatkan pelayanan dari sistem yang dibuat agar sistem tersebut dapat berjalan sesuai dengan fungsinya tanpa kendala. Tahap ini mencakup perbaikan implementasi unit sistem, perbaikan error yang baru saja ditemukan ataupun sisanya' serta melakukan peningkatan kenirja sistem berdasarkan kebutuhan penggunanya.

2.7 OWASP *Mobile Security Testing Guide* (MSTG)

OWASP *Mobile Security Testing Guide* adalah panduan menyeluruh untuk pengujian keamanan aplikasi seluler. Pengujian fungsional dari MSTG mengevaluasi kemampuan *Sandbox* yang dirancang untuk membatasi akses aplikasi

tertentu terhadap fitur dan data sistem sesuai dengan izin tertentu. Pengujian fungsional MSTG terbagi menjadi beberapa jenis, yaitu sebagai berikut[19].

1. Pengujian Izin Aplikasi
   Pengujian jenis ini memiliki tujuan untuk memastikan aplikasi yang sedang diuji memiliki izin yang terbatas, menyesuaikan dengan keperluan utama dan tidak dapat mengakses data pengguna yang sensitif. Pengujian ini dapat dilakukan melalui emulator *mobile* maupun *manual testing*. Pengujian dilakukan dengan cara melakukan instalasi aplikasi yang ingin diuji terlebih dahulu dengan *Sandbox* yang telah diaktifkan dan dikonfigurasikan pada perangkat tersebut; lalu melakukan verifikasi akses fitur dari segi kontak, lokasi, dan penyimpanan; dan diakhiri dengan proses dokumentasi terkait hasil akhir uji coba aplikasi yang telah dibatasi.

2. Pengujian Akses Penyimpanan
   Tujuan dari pengujian akses penyimpanan adalah untuk memastikan aplikasi yang diuji tidak dapat mengakses penyimpanan yang telah dibatasi sebelumnya. Pengujian ini dapat dilakukan secara manual ataupun dapat dilakukan melalui *File Manager* yang ada dalam *device* atau *emulator*. Cara kerjanya menyerupai pengujian sebelumnya, namun berfokus pada bagian penyimpanan data pada perangkat ataupun aplikasi lainnya.

3. Pengujian Respon Terhadap Pembatasan
   Tujuan dari pengujian ini adalah untuk mengecek respon dari aplikasi yang diuji coba pada saat mencoba untuk mengakses data dan sejenisnya yang telah dibatasi oleh *Sandbox*. Pengujian ini dapat dilakukan melalui perangkat atau *emulator mobile* dengan cara melakukan simulasi aksi yang telah dibatasi melalui *Sandbox*, lalu mengamati respon dari aplikasi yang diuji coba ketika gagal mengakses data yang dibatasi, kemudian diakhiri dengan dokumentasi hasil akhirnya.