JEPIN
(Jurnal Edukasi dan Penelitian Informatika)
ISSN(e): 2548-9364 / ISSN(p) : 2460-0741

Vol. 8
No. 3
Desember
2022

Security Assessment Aplikasi Mobile E-Kinerja dengan Acuan OWASP Top 10 Mobile Risks

Dimas Febriyan Priambodo[#1], Muhammad Hasbi[*2], Mahar Surya Malacca[#3]

[#]*Rekayasa Keamanan Siber, Politeknik Siber dan Sandi Negara*
Jl. H.Usa Ciseeng, Bogor, Indonesia, 16120
[1]dimas.febriyan@poltekssn.ac.id
[3]mahar.surya@poltekssn.ac.id

[*]*STMIK Sinar Nusantara*
Jl. K.H Samanhudi No.84-86, Purwosari, Kec. Laweyan, Kota Surakarta, Jawa Tengah 57149
[2]m.hasbi@sinus.ac.id

Abstrak*— Mobile E-Kinerja XYZ adalah aplikasi yang digunakan untuk pelaporan kegiatan Aparatur Sipil Negara (ASN) Pemerintah Kabupaten XYZ. Aplikasi ini menunjang peraturan dari pemerintah pusat terkait Sistem Pemerintahan Berbasis Elektronik (SPBE). Mengingat jumlah ASN sebagai pengguna yang cukup banyak dan melibatkan data pribadi maka perlu dilakukan penilaian kerentanan aplikasi atau dikenal dengan *security assesment. Security assessment* yang dilakukan mencakup identifikasi kerentanan menggunakan MobSF dan MARA Framework diperkuat dengan analisis dinamis serta melakukan validasi mengacu pada OWASP Top Ten Mobile Risk 2016. Penilaian kerentanan juga menggunakan *Common Vulnerability Scoring System* (CVSS) 3.1. sekaligus rekomendasi keamanan terhadap kerentanan yang ditemukan mengacu pada *Common Weakness Enumeration* (CWE) sehingga dapat diperkirakan dampak yang terjadi dari kelemahan yang ditemukan dan rekomendasi perbaikannya. Aplikasi E---Kinerja mempunyai satu krentanan *high* (*Insecure Data Storage)*, tiga kerentanan *medium* (*Improper Platform Usage, Insufficient Cryptography, Reverse Engineering* dan satu kerentanan *low (Extraneous Functionality).

Kata kunci*— MARA framework; Mobile E-Kinerja; Mobile OWASP top 10; MobSF**; ***Security Assessment.

I. PENDAHULUAN

Pandemi COVID-19 mengharuskan pemerintah daerah untuk segera menyesuaikan layanan pemerintahan berbasis elektronik yang mana diatur dalam Peraturan Presiden No. 95 Tahun 2018 Tentang Sistem Pemerintahan Berbasis Elektronik (SPBE). Layanan disusun sesuai regulasi bidang kesehatan, kebencanaan, telekomunikasi, informatika, dan bidang lainnya untuk mencapai kerahasiaan privasi data pengguna pada layanan pemerintahan [1]. Sejalan dengan peraturan tersebut, Badan Siber dan Sandi Negara juga mengeluarkan Peraturan BSSN No.4 Tahun 2021 Tentang SPBE yang didalamnya terdapat standar untuk kebutuhan infrastruktur dan layanan Pemerintah Daerah [2]. Bupati Kabupaten XYZ mengeluarkan peraturan salah satunya Peraturan Daerah Nomor 4 Tahun 2018 Tentang Penyelenggaraan Sistem Pemerintah Berbasis Elektronik di Lingkungan Pemerintah Kabupaten XYZ yang mana penyelenggara pemerintahan memanfaatkan teknologi informasi dan komunikasi untuk memberikan layanan kepada pengguna SPBE [3]. Terkhusus pada kabupaten XYZ, Bupati melalui Peraturan Bupati Nomor 41 tahun 2021 Tentang Tata Kelola Sistem Pemerintahan Berbasis Elektronik digunakan sebagai kerangka kerja untuk memastikan terlaksananya peraturan, pengarahan dan pengendalian dalam penerapan SPBE secara terpadu [4].

Diskominfo Kabupaten XYZ selaku Dinas yang memiliki tugas dan fungsi di bidang informatika dan komunikasi membangun layanan aplikasi E-Letter, E--Presence, dan E-Kinerja XYZ berbasis website dan android. E-Kinerja XYZ merupakan aplikasi yang dikembangkan oleh Dinas Komunikasi dan Informatika Kabupaten XYZ untuk mendata aktivitas atau kinerja Pegawai Negeri Sipil (PNS) di lingkungan Pemerintah Kabupaten XYZ [5]. Aplikasi mobile E-Kinerja XYZ memiliki fitur *Login* dengan *input* yang harus diisi oleh pegawai Aparatur Sipil Negara (ASN) berupa *username* dan *password*, ketika sudah masuk maka pegawai ASN akan melakukan laporan kegiatan harian berupa data pegawai dan aktivitas pegawai [5]. Data tersebut merupakan kategori kerawanan *sensitive data exposure* yang tergolong dalam kerawanan *insecure data storage* pada OWASP Top 10 Mobile Risks [6].

OWASP TOP 10 Mobile Risks yang didalamnya terdapat 10 daftar kerawanan pada aplikasi mobile yang disusun oleh komunitas OWASP [7]. OWASP Mobile Top 10 2016 mengklasifikasikan risiko keamanan aplikasi seluler dan menawarkan solusi untuk mengurangi dampak dan kemungkinan eksploitasi [8]. Kerentanan yang sebelumnya ditemukan di aplikasi seluler dikumpulkan ke dalam *database* yang digunakan untuk mengidentifikasi

kerentanan secara umum sehingga dapat dikelompokkan berdasarkan cara kerja serangan dan dampaknya. Menurut [9], pendekatan yang digunakan CVSS didasarkan pada dampak atau karakteristik yang menyertai kerentanan. CVSS digunakan untuk memberikan analisis melalui peringkat keparahan dampak yang disebabkan oleh kerentanan.

Penelitian pendahuluan dari B. Yankson dkk [10] melakukan uji penetrasi, Penelitian H. Darvish dan M. Husain [11] penambahan analisis dinamis dan penelitian A. Mendoza dan G. Gu [12] melakukan *Application Programming Interface* (API) analisis. Berdasarkan alasan diatas dan beberapa penelitian pendahuluan perlu dilakukannya uji kerawanan atau *security assessment* pada aplikasi mobile E-Kinerja XYZ dengan validasi menggunakan *penetration testing* dan *scoring* menggunakan CVSS untuk mengetahui besaran dampaknya berdasarkan standar OWASP top 10 yang penggabungannya belum pernah dilakukan sebelumnya.

II. PENELITIAN TERKAIT

A. *Security Assessment for Zenbo Robot Using Drozer and mobSF Frameworks*

Penelitian B. Yankson dkk [10] Melakukan penilaian keamanan pada Zenbo Robot sebuah robot sosio-humanoid yang dikembangkan oleh Asus sebagai robot pendamping rumah yang menggunakan sistem operasi android. Penilaian dilakukan untuk membutikan keamanan dan jaminan dari Zenbo Robot karena memiliki fitur yang berdampak fatal jika terjadi serangan pada perangkat seperti roda yang dapat berpindah kemana saja di dalam rumah, kamera 3D RealSense untuk deteksi wajah dan objek, pengambilan foto dan video, deteksi tabrakan, dan perintah gestur. Uji penetrasi testing menggunakan aplikasi *Drozer*, Androbugs, dan MobSF. *Drozer* digunakan oleh peneliti sebagai alat audit komprehensif dan kerangka serangan untuk platform android. AndroBugs digunakan untuk melakukan scanning otomatis untuk menjalankan skema serangan eksploitasi kerawanan pada sistem aplikasi dalam waktu kurang dari dua menit. MobSF digunakan untuk pengujian statis dan dinamis pada aplikasi android untuk penilaian keamanan dan analisis malware pada aplikasi android.

Hasil dari penelitian [10] ditemukan banyak informasi seperti email, gambar, kalender, dan sebagainya yang tersinkronisasi dengan perangkat Zenbo Robot. Hal ini berdampak buruk karena pada uji penetrasi ditemukan kerawanan yang menunjukkan bahwa penyerang dapat mengeksploitasi OS Android atau aplikasi android untuk mendapatkan informasi data pengguna. Zenbo Robot belum memiliki sistem pendeteksi aplikasi berbahaya pada sistemnya, sehingga menimbulkan kemungkinan sistem Zenbo Robot mudah diretas. Dan terakhir ditemukan bahwa informasi Personally identifiable information (PII) dapat dengan mudah diakses dan diambil.

B. *Security Analysis of Mobile Money Application on Android*

Penelitian H. Darvish dan M. Husain [11] melakukan analisis statis otomatis, statis manual dan analisis dinamis pada *mobile money application*. Analisis statis otomatis menggunakan tool pengujian AndroBugs yang digunakan untuk mendapatkan informasi kerawanan yang ada. Analisis statis manual dilakukan dengan *reverse engineering* dengan mengubah aplikasi menjadi dengan mengubah aplikasi menjadi bytecode Smali dan menghasilkan file AndroidManifest.XML yang mencakup semua perizinan yang digunakan aplikasi. Untuk analisis dinamis menggunakan *network intercepting* dengan bantuan *tools* Charles Proxy dan Burp Suite untuk apakah aplikasi rentan terhadap serangan *man-in-the-middle* (MITM). Hasil yang didapatkan dari penelitian ini yaitu dari 26 aplikasi yang dilakukan uji coba, terdapat 4 aplikasi keuangan yang aman diantaranya Android Pay dan Google Wallet.

C. *Mobile Application Web API Reconnaissance: Web-to-Mobile Inconsistencies & Vulnerabilities*

Melakukan deteksi inkonsistensi dalam validasi input antara aplikasi mobile dan layanan API web. Penelitian ini menggunakan analisis statis otomatis dengan menggunakan pendekatan WARDroid untuk analisis komunikasi aplikasi mobile ke web. Dasar penelitian ini yakni layanan API berbasi HTTP yang digunakan aplikasi *mobile* untuk keperluan validasi *input* agar dapat berinteraksi dengan API web. Namun ketika terjadi kegagalan dalam validasi, akan menimbulkan inkonsistensi yang dapat membahayakan privasi pengguna aplikasi. Hasil dari penelitian A. Mendoza dan G. Gu [12] didapati jutaan pengguna berpotensi terdampak dari masalah keamanan ini.

D. *Security vulnerabilities in android applications*

Penelitian P. R. Chernenko dan M. M. Orlova [13] menggunakan ostorlab untuk memindai kerentanan dan melaporkan hasilnya. Penelitian kami menerapkan multi alat MobSF dan MARA dan penilaian dengan kalkulator CWSS. Berlainan dengan penelitian tersebut *security assesment* yang dilakukan menggunakan penetration testing untuk validasi dan *scoring* menggunakan CVSS calculator.

E. *A Study on the Mobile Application Security Threats and Vulnerability Analysis Cases*

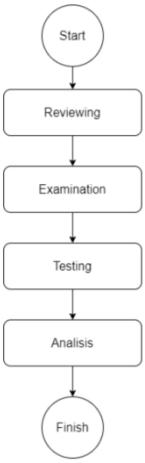
Penelitian H. W. Kim [14] menggunakan aplikasi publik untuk studi kasus, kerentanan dikorelasikan dengan CWE sama seperti penelitian ini. Hasil dan kesimpulan tidak divalidasi dan tidak ada penilaian.

Kelima penelitian terkait diatas memiliki bagian yang diambil untuk dikembangkan dalam penelitian ini seperti merujuk pada penelitian [10] maka memakai tools yang sama yaitu MobSf dengan menambahkan MARA frameworrk dengan objek yang berbeda dan adanya penambahan scoring, validasi serta rekomendasi. Merujuk

penelitian [11] memperkaya penelitian ini dengan analisis statis pada kode program dan sekali lagi memberikan novelty berupa scoring, validasi dan rekomendasi kerentanan. Merujuk penelitian [12] analisis terhadap Web API yang merupakan pencetus penggunaan OWASP top 10 sebagai standar acuannya. Sedangkan dari penelitian [13] memberikan kelengkapan dalam *security assesment* ini melalui penggunaan CVSS sehingga dapat diberikan penilaian kerentanan. Sebagai penutup penelitian [14] memberikan masukkan berkenaan dengan penggunaan CWE untuk dapat dikorelasikan dengan perbaikan yang direkomendasikan. Sehingga penggabungan semua aspek pengujian, paramter dan standar menjadi novelty dalam penelitian ini

III. METODOLOGI PENELITIAN

Metodologi *security assessment* yang diambil dari dokumen SANS Institute terkait *Scoping Security Assessment - A Project Management Approach* [15], yang kemudian dilanjutkan dengan memberikan rekomendasi perbaikan kerawanan pada aplikasi mobile E-Kinerja XYZ. Pada tahapan ini security assessment dibagi menjadi empat tahap seperti pada Gambar 1.



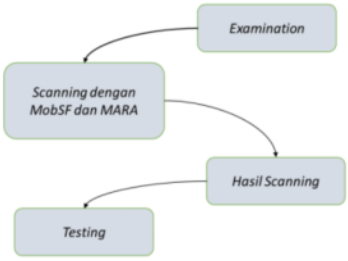
Gambar. 1 Tahapan penelitan

A. Reviewing

Merupakan tahapan yang digunakan untuk mengumpulkan data dan informasi. Tahap ini membutuhkan data primer dan sekunder. Data primer didapatkan dari hasil wawancara yang dilakukan melalui percakapan dengan pihak lokus dan data sekunder diperoleh dari hasil telaah kepustakaan. Berdasarkan hasil wawancara diperoleh data primer berupa dokumen kebijakan Bupati terkait SPBE, dokumentasi penggunaan aplikasi E-Kinerja, serta data pendukung terkait informasi aplikasi mobile E-Kinerja XYZ. Data sekunder diperoleh dari telaah kepustakaan pada sumber-sumber bacaan terkait dengan aplikasi android, kerawanan aplikasi seluler, *security assessment*, OWASP Top Ten Mobile Risk 2016.

B. Examination

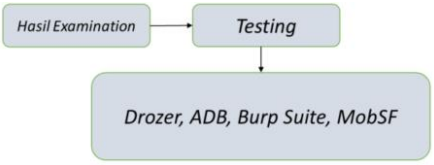
Examination merupakan tahap untuk mengidentifikasi kerawanan yang ada pada aplikasi *mobile*. Tahap ini disebut juga analisis statis karena dilakukan dengan pemindaian atau *scanning* kerawanan menggunakan aplikasi *scanner*. Pada tahap ini dilakukan analisis statis otomatis yaitu *scanning* seperti pada Gambar 2.



Gambar. 2 Tahap *examination*

C. Testing

Testing merupakan tahap pengujian kerawanan yang teridentifikasi dari tahap examination untuk membuktikan bahwa kerawanan yang ditemukan valid. Tahap ini disebut juga dengan dengan analisis dinamis karena pada tahap ini dilakukan pengujian dengan menjalankan aplikasi pada perangkat yang sudah terkondisikan [16]. Alur tahapan testing ditunjukan pada Gambar 3.



Gambar. 3 Tahap Testing

D. Analisis

Tahap ini mempelajari hasil yang didapatkan dari tahap testing dimana kerawanan yang teridentifikasi sudah tervalidasi. Setelah kerawanan telah divalidasi maka dapat diketahui apakah kerawanan tersebut memberikan dampak pada aspek *confidentiality*, *integrity*, *dan availability*, sehingga dapat digunakan untuk menentukan nilai kerawanan dengan menggunakan CVSS 3.1 dan memberikan rekomendasi keamanan yang ditunjukkan pada Gambar 4.

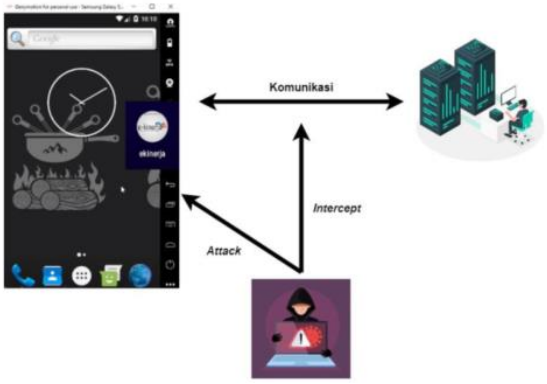
Gambar. 4 Tahap Analisis

Security assessment dilakukan pada kerawanan yang telah tervalidasi. Penilaian menggunakan *base metric* pada CVSS 3.1. *Base metric* ini digunakan karena penilaian dilakukan oleh peneliti berdasarkan kemudahan dalam melakukan serangan (*exploitability metric*) dan dampak dari kerawanan yang dieksploitasi (*impact metric*). *Temporal metric* dan *environmental metric* tidak digunakan dalam penelitian ini karena pada *temporal metric* berkaitan dengan ketersediaan referensi kerawanan dan *environmental metric* berkaitan dengan subjektifitas

pemilik aset. Kemudian setelah dilakukan *security assessment* diberikan rekomendasi perbaikan keamanan untuk mengurangi atau mencegah dampak yang ditimbulkan dari kerawanan yang ada.

IV. HASIL DAN PEMBAHASAN

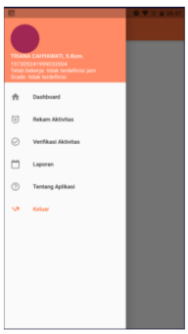
A. *REVIEWING*



Gambar. 5 Skenario pengujian serangan

Gambar 5 menunjukkan skema skenario pengujian *security assessment* yang dilakukan dalam tahapan testing. Pada tahapan testing menggunakan emulator Nox Player untuk menguji kerentanan dari tahap examination karena aplikasi E-Kinerja XYZ tidak mendukung emulator yang berjalan pada sebuah VM. Didalam emulator sudah disesuaikan pengaturan debugging aktif, dalam kondisi root, ter-*install* aplikasi *drozer* agent, dan terinstal sertifikat Burp Suite.

Selanjutnya dicoba untuk menjalankan aplikasi dengan memasukkan *username* dan *password* dan mencoba fitur-fitur yang ada di dalam aplikasi E-Kinerja XYZ. Setelah aplikasi beroperasi maka dilakukan pengujian untuk mencoba mengubungkan dari emulator ke *command line* dengan menggunakan *tools adb* dan *drozer* untuk dilakukan testing kerentanan yang ditemukan pada langkah *examination*. Selain *adb* dan *drozer* juga menggunakan *tools* tambahan lainnya yaitu *Burp Suite* yang digunakan untuk melakukan serangan MITM pada proses komunikasi aplikasi dengan server E-Kinerja XYZ.



Gambar. 6 Tampilan Aplikasi E-Kinerja

Aplikasi ini dibuat dengan framework Flutter dan bahasa pemrograman Dart dengan minimal SDK 16. Berdasarkan Gambar 6 Aplikasi E-Kinerja XYZ memiliki beberapa fitur diantaranya:

a. Login, merupakan tampilan pertama ketika aplikasi dibuka, halaman ini menyajikan dua kolom input berupa username dan password. Sistem login dari mobile E-Kinerja XYZ sendiri dikelompokkan menjadi beberapa yaitu: admin sistem, Admin BKPPD, Admin Bag. Organisasi, Admin BPKAD, Admin OPD, dan Pegawai.
b. Dashboard, berisikan informasi dari user atau pengguna yang melakukan *login* ke aplikasi, selain itu dicantumkan juga Data Atasan dari pegawai yang melakukan login. Atasan ini yang akan memberikan persetujuan terhadap laporan aktivitas dari pegawai di bawahnya.
c. Rekam Aktivitas, fitur ini menyajikan tampilan aktivitas belum diverifikasi, sudah diverifikasi, dan membuat laporan aktivitas yang dilakukan oleh pegawai.
d. Verifikasi Aktivitas, fitur ini akan muncul ketika user memiliki privilege yang sudah diatur diawal, atau user memiliki jabatan tertentu. Fitur ini digunakan untuk menyetujui laporan aktivitas dari pegawai yang berada dibawah naungan user tersebut.
e. Laporan, merupakan fitur yang menampilkan rekapitulasi laporan yang telah dilakukan dengan pembagian waktu Bulan ini, Kemarin, Lusa, dan 1 Tahun.
f. Tentang Aplikasi, menampilkan informasi terkait versi dari aplikasi E-Kinerja, nama pengembang aplikasi, dan disediakan manual E-Kinerja Android.
g. Keluar, merupakan fitur yang digunakan untuk melakukan *log out* dari *session* aplikasi.

B. *EXAMINATION*

1) Mobile Security Framework (MobSF): Dalam perhitungannya *MobSF* memiliki aturan yaitu akan diberikan nilai 100 yang mana dianggap dalam kondisi ideal pada aplikasi yang akan diuji. Pada setiap temuan akan memberikan nilai tambahan atau pengurangan berdasarkan temuan kerentanan, jika kerentanan tidak berbahaya atau dalam kategori *good* maka nilai akan diberi tambahan 5 untuk setiap kerentanannya, untuk kerentanan dengan kategori *high* dan *warning* akan diberikan pengurangan masing-masing poin sebesar 15 dan 10. Sehingga akan dihitung hasil akhirnya melalui seluruh temuan kerentanan pada analisis kode. Jika nilai akhirnya melebihi 100 maka akan dibulatkan menjadi nilai 100 dan jika nilai kurang dari 0 maka nilai keamanannya akan dibulatkan menjadi 10. Nilai keamanan dari pengujian *MobSF* ini dibagi menjadi empat kategori rentang nilai kerentanan untuk menunjukkan kondisi aplikasi yang diuji yang ditunjukkan pada Tabel I. Hasil *scaning mobile* E-Kinerja ditunjukkan dalam Tabel II.

TABEL I
KERENTANAN DAN NILAI MOBSF

Kategori kerentanan	Rentang nilai kerentanan
Critical	0-15
High	16-40
Medium	41-70
Low	71-100

TABEL II
HASIL PEMINDAIAN ISU KERENTANAN MOBSF

Severity	Owasp Top 10	Issue
High	*M5: Insufficient Cryptography*	Aplikasi menggunakan mode enkripsi CBC dengan PKCS5/PKCS7 *padding*. Ini konfigurasi rentan terhadap *padding* serangan *oracel*.
High	*M2: Insecure Data Storage*	Aplikasi dapat membaca/menulis ke *Eksternal* Penyimpanan. Aplikasi apa pun dapat membaca data ditulis ke penyimpanan *Eksternal*
Warning	*M7: Client Code Quality*	Aplikasi menggunakan Database SQLite dan jalankan kueri SQL mentah. Tidak terpercaya masukan pengguna dalam kueri SQL mentah dapat menyebabkan SQL Injection. Juga sensitive informasi harus dienkripsi dan ditulis ke *database*.

2) MARA Framework: Scanning menggunakan MARA *Framework* mengidentifikasi kerentanan dengan mengacu pada OWASP Top 10 Mobile Risk dan CVE. Pada aplikasi E-Kinerja terdapat 10 isu kerentanan. Kemudian isu kerentanan tersebut dikategorikan menjadi *critical*, *high*, *medium*, *low* atau *information* lebih detail ditampilkan dalam Tabel III. Berdasarkan *scanning* dari MARA *Framework* diidentifikasi kerentanan yang mengacu OWASP Mobile *Top Ten* 2016 yaitu *Insecure Data Storage* [M2] dan *Client Code Quality* [M7].

TABEL III
HASIL PEMINDAIAN ISU KERENTANAN MARA FRAMEWORK

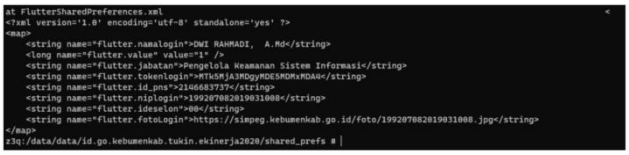
Kategori	Jumlah	Deskripsi
Warning	3	*Insecure Data Storage: External Storage Accessing*
		Insecure Data Storage: Sensitive_Information (Getting ANDROID_ID)
		WebView Local File Access Attacks (webview)
Notice	7	*AndroidManifest Adb Backup Checking*
		<Database><#CVE-2011-3901#> Android SQLite Databases Vulnerability
		File Unsafe Delete
		Client Code Quality : <Debug><Hacker> Codes for Checking Android Debug Mode
		<Signature><Hacker> Getting Signature Code
		Native Library Loading
		AndroidManifest Exported Components

C. *TESTING*

TABEL IV
HASIL PEMINDAIAN MOBSF DAN MARA FRAMEWORK

OWASP Top 10 Mobile Risk	MObSF	MARA
Improper Platform Usage	x	x
Insecure Data Storage	√	√
Insecure Communication	x	x
Insecure Authentication	x	x
Insufficient Cryptography	√	x
Insecure Authorization	x	x
Client Code Quality	√	√
Code Tampering	x	x
Reverse Engineering	x	x
Extraneous Functionality	x	x

1) M1: Improper Platform Usage: Uji coba serangan untuk membuktikan ada atau tidak ada kerentanan *improper* platform *usage* yang mana merupakan penyalahgunaan fitur atau kegagalan kontrol keamanan yang dapat menyebabkan pengguna yang tidak berhak dapat mengakses fitur tertentu. Pengujian menggunakan *tools adb* dan menggunakan bantuan emulator *Nox Player* untuk menjalankan *mobile* E-Kinerja XYZ. Untuk mengakses folder *shared_prefs* perangkat harus dalam keadaan *root*. Kemudian melalui *tools adb* dilakukan akses pada *Nox Player* dan mengakses *file* .xml pada folder *shared_prefs*.

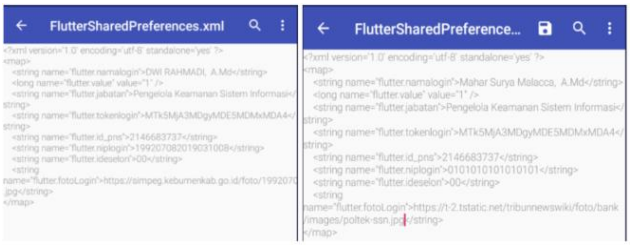


Gambar. 7 Data FlutterSharedPreferences.xml

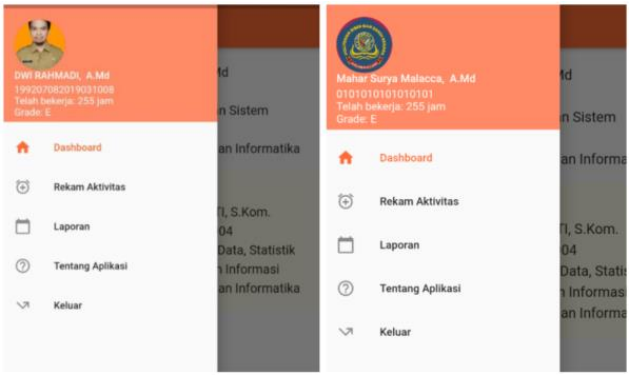
Gambar 7 menunjukkan dengan melakukan ini didapatkan data pengguna berupa nama pengguna, jabatan pengguna, *token login*, *id user*, NIP, dan foto pengguna. Hasil temuan didapatkan pada lokasi data/data/id.go.xxxxx.tukin.ekinerja2020. Data sensitif tersebut didapatkan dalam *file* FlutterSharedPreferences.xml didalam folder *shared_prefs* yang mana merupakan salah satu lokasi penyimpanan dengan format *file* xml.

2) M2: Insecure Data Storage: Pembuktian kerentanan *insecure* data *storage* juga berkaitan dengan kerentanan *improper* platform *usage*. Pengujian yang

dilakukan menggunakan *tools text* editor bawaan dari *Nox Player* ketika membuka *file* .xml seperti yang ditunjukkan pada Gambar 8.



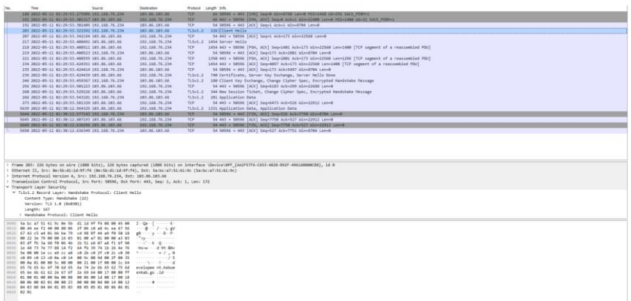
Gambar. 8 Modifikasi FlutterSharedPreferences.xml



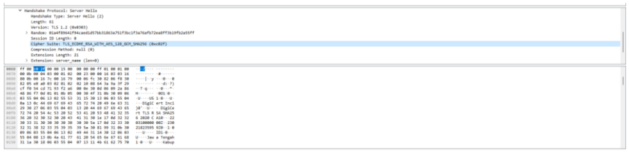
Gambar. 9 Tampilan Sebelum dan Sesudah Modifikasi

Perubahan yang dilakukan pada file FlutterSharedPreferences.xml meliputi modifikasi pada nama pengguna, NIP, dan foto pengguna yang ditunjukkan pada Gambar 9. *File* temp dapat dimanfaatkan untuk melakukan perubahan data terkait informasi pribadi pengguna.

3) M3: Insecure Communication: Kerentanan *insecure communication* memunculkan isu bahwa aplikasi rawan terhadap serangan MITM. Pada hasil pemindaian *mobile* E-Kinerja tidak teridentifikasi bahwa aplikasi terdapat kerawanan *insecure communication*. Untuk pengujian kerentanan ini menggunakan *tools wireshark* dan *burp suite* untuk menangkap komunikasi aplikasi terkait *request* dan *response* yang diberikan serta data yang dikirimkan terenkripsi atau tidak. Komunikasi data yang dikirimkan *mobile* E-Kinerja telah terenkripsi yang ditunjukkan pada Gambar 10 dan Gambar 11 menunjukkan bahwa server memberikan *response* terenkripsi dengan generator dan *verifikator* yang digunakan yaitu RSA *With* AES 128-SHA256.

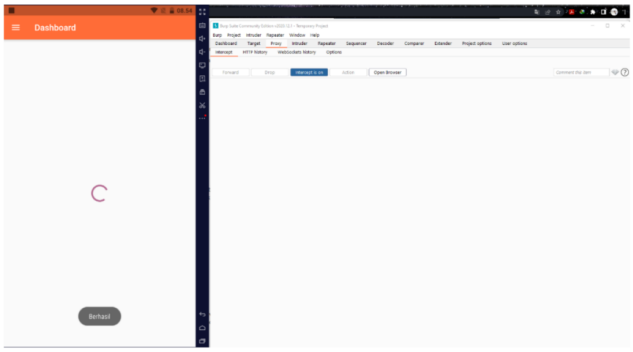


Gambar. 10 Hasil penangkapan komunikasi data E-Kinerja



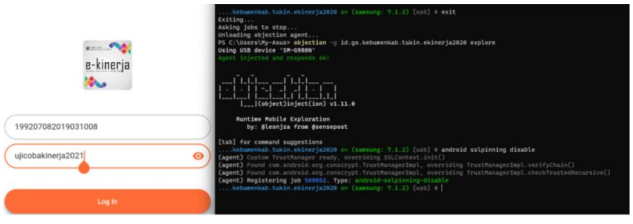
Gambar. 11 Response server terhadap client

4) M4: Insecure Authentication: Eksploitasi *otentikasi* pada kerentanan *insecure authentication* dilakukan oleh penyerang menggunakan serangan otomatis yang disediakan oleh *tools* yang sudah tersedia seperti *burp suite*. Untuk membuktikan dilakukan percobaan serangan *brute force* menggunakan bantuan aplikasi *burp suite* untuk menangkap *request* yang dikirimkan oleh aplikasi ke server. Kemudian dilakukan perubahan pada *username* dan *password* untuk dilakukan *brute force*.



Gambar. 12 Aplikasi E-Kinerja tidak tertangkap *burpsuite*

Gambar 12 menunjukkan bahwa *burp suite* tidak dapat melakukan *intercept* data komunikasi pada *mobile* E-Kinerja XYZ karena sudah menggunakan protokol komunikasi HTTPS. Kemudian dilakukan percobaan *bypass SSL Pinning* untuk mengelabuhi SSL aplikasi agar dapat terhubung dengan burp suite. Namun aplikasi E-Kinerja tetap tidak dapat terhubung dengan burp suite untuk ditangkap response yang dikirimkan dari aplikasi ke server seperti yang ditunjukkan pada Gambar 13. Percobaan *SSL Pinning* sudah dilakukan dan hasilnya tetap sama bahwa aplikasi tidak dapat dilakukan *intercept* data menggunakan proxy burp suite dan juga *SSL Pinning*.



Gambar. 13 Percobaan serangan SSL *Pinning*

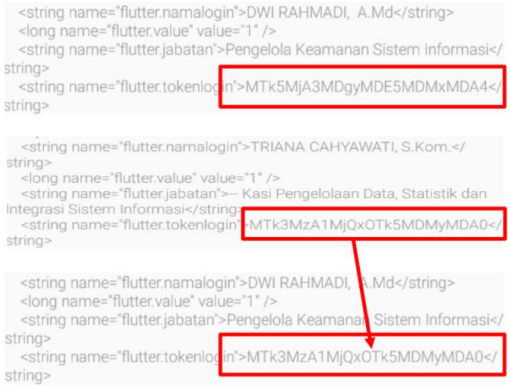
5) M5: Insufficient Cryptography: Berdasarkan hasil pemindaian teridentifikasi adanya isu kerentanan *insufficient cryptography* yang mana *mobile* E-Kinerja XYZ menggunakan mode enkripsi CBC dengan *padding* PKC55/PKC57. Hal ini dibuktikan dengan ditemukannya

penggunaan PKCS7Padding pada *source code* aplikasi E-Kinerja XYZ yang ditunjukkan pada Gambar 14.

Gambar. 14 Penggunaan PKCS7Padding dan SHA 256 pada *source code*

PKCS7Padding yang digunakan merupakan peningkatan dari PKCS5Padding. Perbedaan mekanisme keduanya adalah ukuran blok, PKCS5Padding didefinisikan untuk blok berukuran 8 *byte* dan PKCS7Padding didefinisikan untuk ukuran blok apa pun dari 1 hingga 255 *byte*. Sehingga pada dasarnya PKCS5Padding adalah sub set dari PKCS7Padding untuk blok berukuran 8 *byte*. Oleh karena itu, tidak dapat digunakan untuk AES. PKCS5Padding adalah standar untuk Enkripsi Berbasis Kata Sandi atau PBE, dan PKCS7Padding mendefinisikan *Cryptographic Message Syntax* (CMS). Kriptografi ini lemah dalam hal enkripsi sehingga berpengaruh terhadap status pengguna, status sistem, atau beberapa keputusan yang dibuat di server.

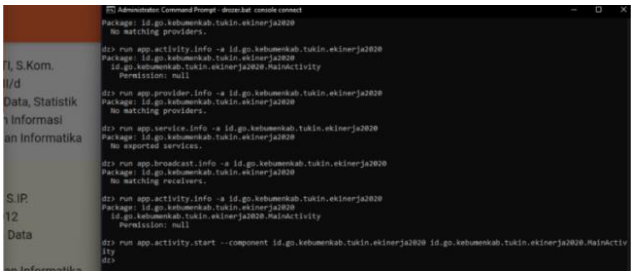
6) M6: Insecure Authorization: Kerentanan *insecure authorization* memungkinkan adanya ancaman terhadap eksploitasi kerentanan otorisasi. Dari kerentanan *insecure data storage* ditemukan adanya *flutter.tokenlogin* yang ditunjukkan pada Gambar 15 *tokenlogin* ini berbeda di setiap akun yang melakukan *login* dan tersimpan pada FlutterSharedPreferences.xml yang mana memiliki kerentanan untuk diubah oleh penyerang. Dua *tokenlogin* yang didapatkan berupa *tokenlogin* atasan MTk3MzA1MjQxOTk5MDMyMDA0 dan *tokenlogin* staf MTk5MjA3MDgyMDE5MDMxMDA4.



Gambar. 15 Modifikasi tokenlogin

Pengujian pada tahap ini akan dilakukan modifikasi *tokenlogin* yang didapatkan dari dua akun pengguna *mobile* E-Kinerja XYZ. Ketika *tokenlogin* milik atasan dimasukkan ke pegawai maka akan muncul notifikasi selesai dan pada aplikasi tidak menunjukkan adanya perubahan tampilan maupun penambahan fitur. Namun kerentanan disini yaitu tokenlogin yang diberikan oleh aplikasi tidak berubah disetiap sesi login yang dilakukan oleh pengguna. Sehingga menyebabkan kemungkinan untuk dilakukannya serangan insecure authorization oleh penyerang terhadap aplikasi.

7) M7: Client Code Quality: Hasil pemindaian aplikasi diidentifikasi adanya isu kerentanan *client code quality* pada *SQLite Database*. Untuk pengujian kerentanan ini menggunakan *drozer* dan manual *payload*. Percobaan serangan menggunakan *drozer* dilakukan berdasarkan temuan *activity* pada *package* id.go.xxxxx.tukin.ekinerja2020.

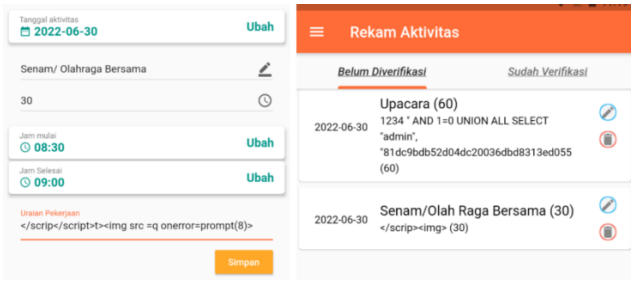


Gambar. 16 Percobaan *Activity Attack*

Perintah app.activity.info digunakan untuk mencari *activity* yang ada pada mobile E-Kinerja XYZ. Kemudian aktifitas yang ditemukan dilakukan percobaan serangan dengan menjalankannya menggunakan *drozer* melalui perintah *app.activity.start*. Selanjutnya untuk pengujian manual *sql injection* dengan memasukkan *payload* '1 OR '1'='1' dan beberapa *payload* lainnya pada *username* dan *password*, namun tidak menunjukkan adanya *error* maupun keterkaitan pada kerentanan *sql injection*.

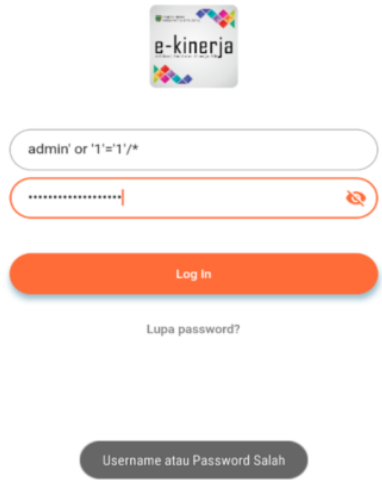
Berdasarkan pengujian yang dilakukan pada isu kerentanan *client code quality* yang terbukti yaitu kerentanan pada serangan *activity*. Namun kerentanan ini tidak memberikan dampak pada *confidentiality*, *integrity*, *availability* (CIA). Serangan *activity* akan memberikan dampak pada *activity* yang berisikan database yang mana akan memberikan data pengguna *mobile* E-Kinerja XYZ kepada penyerang.

8) M8: Code Tampering: Kerentanan *code tampering* tidak teridentifikasi pada hasil pemindaian *mobile* E-Kinerja XYZ. Salah satu pengujian untuk membuktikan adanya kerentanan *code tampering* adalah dengan serangan *Cross-Site Scripting* (XSS) dengan memanfaatkan fitur keterangan yang berguna untuk memasukkan teks.



Gambar. 17 Percobaan *Cross-Site Scripting* (XSS)

Percobaan XSS yang ditunjukkan pada Gambar 17 menunjukkan bahwa aplikasi tetap merespon semua masukkan yang dituliskan pada fitur Uraian Pekerjaan. Namun aplikasi menghapus strings dan parameter yang merupakan *query* untuk serangan XSS. Kemudian dari Rekaman Aktivitas yang dikirimkan terdapat *payload* XSS dan dicoba untuk disetujui menggunakan akun atasan, ditemukan bahwa Rekaman Aktivitas tersebut tidak dapat disetujui yang mana ditunjukkan dengan tidak adanya respon ataupun perubahan ketika dilakukan fitur persetujuan aktivitas. Sehingga perlu adanya penghapusan pada Rekaman Aktifitas yang menggunakan *payload* XSS agar tidak tersimpan didalam sistem dan menjadi kerentanan XSS Stored.



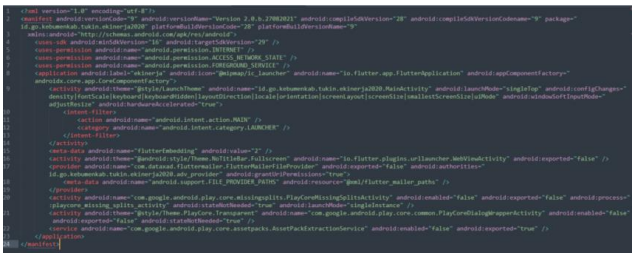
Gambar. 18 Serangan *SQL Injection Login*

Selain pengujian XSS dilakukan juga serangan SQL Injection dengan beberapa *payload* yang dapat dilihat pada Tabel V. Gambar 18 menunjukkan bahwa aplikasi E-Kinerja XYZ tidak berhasil di bypass menggunakan *payload* yang digunakan. Sehingga dapat diketahui bahwa aplikasi tidak dapat dilakukan serangan SQL Injection.

TABEL V
PAYLOAD SQL INJECTION LOGIN

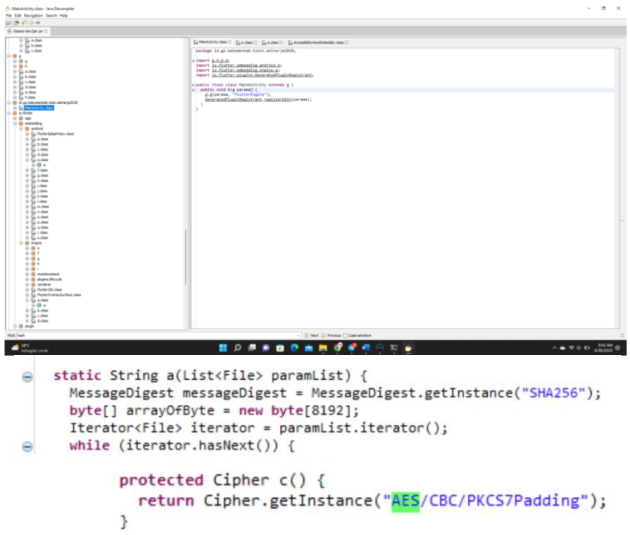
Payload SQL Injection Login	' OR 1 -- -
	" OR "" = "
	1' ORDER BY 1--+
	1' ORDER BY 1,2,3--+
	1-false
	AS INJECTX WHERE 1=1 AND 1=1
	admin' or '1'='1'/*
	1234 " AND 1=0 UNION ALL SELECT "admin", "81dc9bdb52d04dc20036dbd8313ed055

9) M9: Reverse Engineering: Kerentanan *reverse engineering* merupakan kerentanan untuk mendapatkan informasi penting yang ada dalam aplikasi melalui *source code* yang sudah dilakukan *decompile*. Untuk prosesnya menggunakan bantuan *tools* apktool dan MARA *Framework*. Pada aplikasi ini diperoleh *folder* dan *file* assets, kotlin, lib, META-INF, res, AndroidManifest.xml, classes.dex, resources.arsc.



Gambar. 19 Tampilan AndroidManifest.xml

Terdapat informasi yang didapatkan pada file AndroidManifest.xml seperti version, package, android:minSdkVersion, android:targetSdkVersion, permission yang ditunjukkan pada Gambar 19 namun tidak ditemukan adanya informasi berbahaya yang dapat dimanfaatkan oleh penyerang untuk mengambil data pribadi yang ada pada database aplikasi.

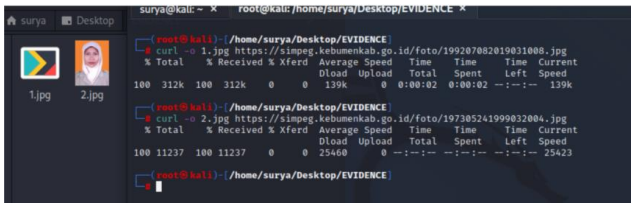


Gambar. 20 Reverse Engineering menggunakan JD-GUI

Gambar 20 menggunakan aplikasi JD-GUI untuk membuka file classes.dex2jar.jar yang merupakan hasil *compile* menggunakan aplikasi dex2jar untuk. Dapat dilihat bahwa hasil *reverse* kode sumber mobile E-Kinerja XYZ masih dapat dibaca oleh penyerang, namun untuk hirarki dari lokasi path tidak urut atau acak dan terdapat beberapa library yang tidak ter-*decompile* dengan baik sehingga tidak dapat terbaca. Hasil *reverse* engineering yang dilakukan ditemukan beberapa informasi seperti kriptograsi yang digunakan aplikasi dalam enkripsi komunikasinya menggukanan AES PKCS7Padding, untuk penggunaan hash menggunakan SHA256, selain itu juga diketahui bahwa aplikasi melakukan pembuatan file sharedpreferences yang mana file tersebut berisikan informasi dengan konten sensitif didalamnya.

10) M10: Extraneous Functionality: Biasanya dalam serangan *extraneous functionality* penyerang akan memahami fungsi yang ada dalam aplikasi hingga fungsi

tersembunyi didalam sistem server. Skema serangan yang dilakukan berupa penyerang akan mengunduh dan memeriksa file yang didapatkan. Pengujian dapat menggunakan tools curl untuk menjalankan fungsi administratif back-end. Pada pengujian ini ditemukan dua informasi berupa https://simpeg.xxxx.go.id/foto/199207082019031008.jpg dan https://simpeg.xxxx.go.id/foto/197305241999032004.jpg. Dari kedua informasi tersebut dilakukan pengujian curl.



Gambar. 21 Percobaan *Curl*

Gambar 21 menunjukkan bahwa informasi yang didapatkan dapat diakses dan diunduh menggunakan bantuan tools curl. Jika informasi yang didapatkan oleh penyerang merupakan sebuah basis data yang berisikan informasi pribadi pengguna aplikasi, maka penyerang akan dengan mudah mengunduh informasi sensitif tersebut.

D. Analisis

TABEL VI
HASIL VALIDASI TERHADAP AUTOMATED SCAN

OWASP Top 10 Mobile Risk	Validasi
Improper Platform Usage	√
Insecure Data Storage	√
Insecure Communication	x
Insecure Authentication	x
Insufficient Cryptography	√
Insecure Authorization	x
Client Code Quality	√
Code Tampering	x
Reverse Engineering	√
Extraneous Functionality	√

Pemindaian kerentanan menggunakan MobSF dan MARA terdapat 3 kerentanan yang terdeteksi. Sebagai validasi dilakukan 10 pengujian dengan hasil pada Tabel VI dan ditemukan 6 kerentanan terbukti yaitu *Improper Platform Usage* dengan score kerentanan dalam Tabel VII, *Insecure Data Storage* dengan *score* dalam Tabel VIII, *Insufficient Cryptography* dengan *score* dalam Tabel IX, *Client Code Quality* dengan *score* dalam Tabel X, *Extraneous Functionality* dengan *score* dalam Tabel XI dan *Reverse Engineering* dengan *score* dalam Tabel XII. Kemudian terdapat 4 kerentanan yang tidak terbukti yaitu *Insecure Communication, Insecure Authentication, Insecure Authorization, dan Code Tampering*.

TABEL VII
PENILAIAN KERENTANAN *IMPROPER PLATFORM USAGE*

Expoitability Metric		*Impact Metric*		Nilai CVSS
Aspek	Nilai Metrik	Aspek	Nilai Metrik	
AV	L	C	L	5,1 (Medium)
AC	L	I	L	
PR	N	A	N	
UI	N			
S	U			

TABEL VIII
PENILAIAN KERENTANAN *INSECURE DATA STORAGE*

Expoitability Metric		*Impact Metric*		Nilai CVSS
Aspek	Nilai Metrik	Aspek	Nilai Metrik	
AV	L	C	H	7,1 (High)
AC	L	I	N	
PR	N	A	N	
UI	N			
S	C			

TABEL IX
PENILAIAN KERENTANAN INSUFFICIENT CRYPTOGRAPHY

Expoitability Metric		*Impact Metric*		Nilai CVSS
Aspek	Nilai Metrik	Aspek	Nilai Metrik	
AV	N	C	L	6.5 (Medium)
AC	L	I	L	
PR	N	A	N	
UI	N			
S	U			

TABEL X
PENILAIAN KERENTANAN *CLIENT CODE QUALITY*

Expoitability Metric		*Impact Metric*		Nilai CVSS
Aspek	Nilai Metrik	Aspek	Nilai Metrik	
AV	L	C	N	0,0 (None)
AC	L	I	N	
PR	N	A	N	
UI	R			
S	U			

TABEL XI
PENILAIAN KERENTANAN *REVERSE ENGINEERING*

Expoitability Metric		*Impact Metric*		Nilai CVSS
Aspek	Nilai Metrik	Aspek	Nilai Metrik	
AV	P	C	H	5,3 (Medium)
AC	L	I	N	
PR	N	A	N	
UI	N			
S	C			

TABEL XII
PENILAIAN KERENTANAN *REVERSE ENGINEERING*

Expoitability Metric		*Impact Metric*		Nilai CVSS
Aspek	Nilai Metrik	Aspek	Nilai Metrik	
AV	N	C	L	3,7 (Low)
AC	H	I	N	
PR	N	A	N	
UI	N			
S	U			

E. Rekomendasi

1) Improper Platform Usage: Pengujian kerentanan *improper platform usage* didapatkan sebuah *file temp* atau sebuah lokasi penyimpanan yang mengandung informasi sensitif pengguna yang berada pada *folder package* aplikasi id.go.xxxx.tukin.ekinerja2020 yang mana termasuk dalam kerentanan CWE-276 *Incorrect Default Permission* dimana ketika penginstalan aplikasi, izin *file* yang diinstal diatur untuk mengizinkan siapa saja memodifikasi *file* tersebut[20]. Kerentanan ini memiliki

dampak berupa pencurian data identitas pengguna, pelanggaranan privasi, dan dapat digunakan untuk penipuan yang dapat merugikan pengguna. mitigasi pada fase arsitektur dan desain aplikasi dapat menggunakan strategi pemisahan hak istimewa [17]. Selain itu dapat menggunakan tambahan kode untuk menghapus *file shared preference* langsung menggunakan kode java pada Gambar 22.

```
public static boolean deleteSharedPreferences(Context
context, String name) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        return context.deleteSharedPreferences(name);
    } else {
        context.getSharedPreferences(name,
MODE_PRIVATE).edit().clear().commit();
        File dir = new
File(context.getApplicationInfo().dataDir,
"shared_prefs");
        return new File(dir, name + ".xml").delete();
    }
}
```

Gambar. 22 modifikasi kode java

2) Insecure Data Storage: Kerentanan *insecure data storage* terjadi ketika *file* .xml yang berisikan informasi data pribadi terkait dengan aplikasi E-Kinerja XYZ dan ditemukan oleh penyerang dan kemudian dapat dilakukan modifikasi pada *file* tersebut. Kerentanan ini termasuk ke dalam CWE-922: *Insecure Storage of Sensitive Information* karena perangkat lunak atau aplikasi menyimpan informasi sensitif tanpa membatasi akses baca atau tulis dengan benar, sehingga menyebabkan penyerang dapat melakukan modifikasi pada informasi yang didapatkan [18]. Penanggulangan dapat dilakukan dengan melakukan enkripsi pada data yang ada sehingga ketika infomasi dapat diakses oleh siapapun namun informasi tidak dapat dibaca oleh pihak yang tidak berwenang. Selain itu dapat menghapus *file* .xml menggunakan kode yang dijelaskan pada anak subbab xxx agar file temp tidak ditampilkan.

3) Insufficient Cryptography: Kerentanan ini terdaftar pada CWE-649: *Reliance on Obfuscation or Encryption of SecurityRelevant Inputs without Integrity Checking*. Aplikasi menggunakan enkripsi input yang tidak boleh diubah oleh pihak eksternal, tetapi perangkat lunak tidak menggunakan validasi integritas untuk mendeteksi apakah input tersebut dimodifikasi atau tidak.

Saat aplikasi bergantung pada kebingungan atau penerapan yang salah/enkripsi yang lemah untuk melindungi token atau parameter yang dapat dikontrol klien, hal itu mungkin berdampak pada status pengguna, status sistem, atau beberapa keputusan yang dibuat di server. Tanpa melindungi token/parameter untuk integritas, aplikasi rentan terhadap serangan di mana musuh melintasi ruang nilai yang mungkin dari token/parameter tersebut untuk mencoba mendapatkan keuntungan. Tujuan penyerang adalah untuk menemukan nilai lain yang dapat diterima yang entah bagaimana akan meningkatkan hak istimewa mereka dalam sistem, mengungkapkan informasi atau mengubah perilaku sistem dalam beberapa cara yang bermanfaat bagi penyerang. Jika aplikasi tidak melindungi token/parameter penting ini untuk integritas, aplikasi tidak akan dapat menentukan bahwa nilai-nilai ini telah dirusak.

CWE-649 dapat dimitigasi menggunakan tiga cara yaitu [19]:

- Lindungi token/parameter penting yang dapat dikontrol klien untuk integritas menggunakan metode PKI (yaitu tanda tangan digital) atau cara lain, dan periksa integritas di sisi server.
- Permintaan berulang dari pengguna tertentu yang menyertakan nilai token/parameter yang tidak valid (yang tidak boleh diubah secara manual oleh pengguna) akan mengakibatkan penguncian akun pengguna.
- Token/parameter sisi klien tidak boleh sedemikian rupa sehingga mudah/dapat diprediksi untuk menebak status valid lainnya.

4) Client Code Quality: Kerentanan ini termasuk kedalam *manipulatable activity* yang menyebabkan penyerang dapat menjalankan aplikasi pada tampilan pertama aplikasi yaitu halaman *login*, ketika pengguna terlupa untuk melakukan *logout* pada aplikasi maka akan menyebabkan penyerang masuk ke halamam terakhir yang digunakan oleh pengguna. Kerentanan ini dapat dicegah dengan mengatur *ekspor activity* id.go.xxxx.tukin.ekinerja2020.MainActivity yang awalnya true diubah menjadi false.

5) Reverse Engineering: Hasil pengujian terhadap kerentanan *reverse engineering* menunjukkan bahwa mobile E-Kinerja XYZ belum menggunakan *obfuscation* sehingga beberapa informasi *sensitive* pada *file* string dapat dibaca oleh penyerang secara mudah. Kerentanan ini tercatat dalam CWE-312 Cleartext *storage of sensitive information* [20], yang mana dapat menyebabkan pencurian informasi dan sumber daya hingga modifikasi sumber kode dari aplikasi yang ditargetkan. *Framework* flutter menyediakan pengamanan *obfuscating* dengan menambahkan perintah flutter build apk --obfuscate --split-debuginfo=/<project-name>/<directory> pada kode sumbernya. Sehingga ketika aplikasi dilakukan *reverse engineering* maka kode sumber dan *path* dari aplikasi akan diacak agar penyerang tidak dapat melakukan analisis pada kode sumber yang didapatkan.

6) Extraneous Functionality: Kerentanan extraneous functionality ditemukan dengan pengujian yang dilakukan menggunakan bantuan tools curl yang merupakan kerentanan CWE-319: *Cleartext Transmission of Sensitive Information* dimana perangkat lunak mentransmisikan data sensitif dalam sebuah teks yang jelas yang dapat dilihat oleh penyerang.

CWE-319 dapat dimitigasi dengan tiga cara yaitu [21]:

- Fase Arsitektur dan Desain, melakukan enkripsi data dengan skema enkripsi yang terjamin sebelum data dikirim.

- Fase Implementasi, saat menggunakan aplikasi web dengan SSL, gunakan SSL untuk seluruh sesi dari login hingga logout, bukan hanya untuk halaman login awal.
- Fase Operasi, lakukan konfigurasi server menggunakan saluran terenkripsi untuk komunikasi yang mungkin menyertakan SSL atau protokol aman lainnya.

Selain itu pada CVE-2018-1000007 terdapat rekomendasi berupa melakukan pembaharuan pada curl yang digunakan, terapkan pembaharuan ke aplikasi dan lakukan *rebuild*, dan jangan aktifkan CURLOPT_FOLLOWLOCATION agar penyerang tidak dapat menggunakan curl untuk melakukan GET maupun POST pada server [21].

V. KESIMPULAN

Berdasarkan serangkaian pengujian kerentanan berdasarkan Top 10 Mobile *risk* yang tervalidasi menggunakan *penetration* testing dan dicocokkan dengan CWE sehingga diperoleh rekomendasi, maka Aplikasi E-Kinerja mempunyai satu *kerentanan high (Insecure Data Storage)*, tiga kerentanan *medium (Improper Platform Usage*, *Insufficient Cryptography*, *Reverse Engineering*) dan satu kerentanan *low (Extraneous Functionality)* dengan detail sebagai berikut:

A. Improper *Platform* Usage

Berupa temuan sebuah *file temp* atau sebuah lokasi penyimpanan yang mengandung informasi sensitif pengguna yang mana merupakan kesalahan dari dan tergolong dalam kategori medium. Dampak dari kerentanan ini berupa adanya izin untuk siapa saja mengakses dan memodifikasi *file* tersebut. Rekomendasi keamanan ini berupa mengatur ulang pembagian akses pada proses instalasi aplikasi pada perangkat dan juga menambahkan perintah deleteSharedPreferences pada kode sumber untuk menghapus *file temp*.

B. Insecure Data Storage

Pada *file temp* dapat dilakukan modifikasi yang mana hal ini merupakan kerentanan dengan kategori *high* karena penyerang dapat mengubah informasi hingga ditampilkan pada aplikasi langsung. Hal ini terjadi karena perangkat lunak atau aplikasi menyimpan informasi sensitif tanpa membatasi akses baca atau tulis dengan benar, sehingga menyebabkan penyerang dapat melakukan modifikasi pada informasi yang didapatkan. Rekomendasi perbaikan dari kerentanan ini berupa melakukan enkripsi data dan menghapus *file temp* agar penyerang tidak dapat mengakses informasi pribadi pengguna yang tersimpan pada penyimpanan perangkat.

C. Insufficient Cryptography

Kerentanan ini termasuk kategori kerentanan medium karena serangan menargetkan pada kriptografi *input* yang digunakan oleh aplikasi. Sehingga penyerang menemukan nilai lain yang ada pada sistem untuk mengungkap informasi atau mengubah sistem agar bermanfaat bagi penyerang. Rekomendasi perbaikannya pada bagian arsitektur dan desain dengan melindungi *token*/parameter yang penting dan menambahkan sistem validasi *input*.

D. Client Code Quality

Kerentanan ini tergolong dalam kategori none karena tidak berdampak pada aspek CIA (kerahasiaan, integritas dan ketersediaan). Namun tetap saja ini merupakan sebuah celah, untuk menanggulanginya dengan cara mengatur *ekspor activity* id.go.xxxx.tukin.ekinerja2020.MainActivity yang awalnya *true* diubah menjadi *false*.

E. Reverse Engineering

Kerentanan ini berdampak pada aspek kerahasiaan karena informasi pada sumber kode aplikasi berupa *cryptography* yang digunakan, informasi terkait *permission*, target SDK, min SDK pada *file manifest* dapat dibaca dengan mudah. Sehingga membuat kerentanan ini tergolong dalam kategori medium. Rekomendasi perbaikannya dapat menggunakan teknik obfuscation pada kode sumber sehingga ketika dilakukan serangan ini maka kode sumber tidak dapat dibaca oleh penyerang.

F. Extraneous Functionality

Kerentanan ini tergolong dalam kategori *low*. Karena perangkat lunak mentransmisikan data sensitif dalam sebuah teks yang jelas yang dapat dilihat oleh penyerang. Rekomendasi perbaikan dari kerentanan ini berupa pembaharuan pada *curl* yang digunakan, terapkan pembaharuan ke aplikasi dan lakukan *rebuild*, dan jangan aktifkan CURLOPT_FOLLOWLOCATION agar penyerang tidak dapat menggunakan *curl* untuk melakukan GET maupun POST pada server.

REFERENSI

[1] K. S. RI, "Perpres," *Menteri Huk. Dan Hak Asasi Mns. Republik Indones.*, p. 110, 2018.

[2] Badan Siber dan Sandi Negara, "Peraturan Badan Siber Dan Sandi Negara Nomor 4 Tahun 2021 Tentang Pedoman Manajemen Keamanan Informasi Sistem Pemerintahan Berbasis Elektronik Dan Standar Teknis Dan Prosedur Keamanan Sistem Pemerintahan Berbasis Elektronik," *Bssn.Go.Id*, 2021.

[3] Pemerintah Kabupaten Kebumen, "Peraturan Daerah Kabupaten Kebumen Nomor 4 Tahun 2018 Tentang Penyelenggaraan Sistem Pemerintahan Berbasis Elektronik Di Lingkungan Kabupaten Kebumen," 2018.

[4] Pemerintah Kabupaten Kebumen, "TATA KELOLA SISTEM PEMERINTAHAN BERBASIS ELEKTRONIK DI LINGKUNGAN PEMERINTAH KABUPATEN KEBUMEN," p. 6, 2021.

[5] D. Kebumen, "APLIKASI E-KINERJA," *Kabupaten Kebumen*, vol. 1, p. 5, 2018.

[6] A. Elanda and R. L. Buana, "Analisis Kualitas Keamanan Sistem Informasi E-Office Berbasis Website Pada Stmik Rosma Dengan Menggunakan Owasp Top 10," *CESS (Journal Comput. Eng. Syst. Sci.*, vol. 6, no. 2, pp. 37–43, 2021.

[7] The OWASP®Foundation, "OWASP Mobile Top 10," *OWASP Mobile Top 10*. 2011, [Online]. Available: https://owasp.org/www-project-mobile-top-10/.

[8] Umasankar, "Analysis of latest vulnerabilities in android," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1236–1241, doi: 10.1109/ICACCI.2017.8126011.

[9] V. R. Kebande, I. Kigwana, H. S. Venter, N. M. Karie, and R. D. Wario, "CVSS Metric-Based Analysis, Classification and Assessment of Computer Network Threats and Vulnerabilities," in *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, 2018, pp. 1–10, doi: 10.1109/ICABCD.2018.8465420.

[10] B. Yankson, K. Javed Vali, P. C. K. Hung, F. Iqbal, and L. Ali, "Security Assessment for Zenbo Robot Using *Drozer* and mobSF Frameworks," *2021 11th IFIP Int. Conf. New Technol. Mobil. Secur. NTMS 2021*, 2021, doi: 10.1109/NTMS49979.2021.9432666.

[11] H. Darvish and M. Husain, "Security Analysis of Mobile Money Applications on Android," *Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018*, pp. 3072–3078, 2019, doi: 10.1109/BigData.2018.8622115.

[12] A. Mendoza and G. Gu, "Mobile Application Web API Reconnaissance: Web-to-Mobile Inconsistencies & Vulnerabilities," *Proc. - IEEE Symp. Secur. Priv.*, vol. 2018-May, pp. 756–769, 2018, doi: 10.1109/SP.2018.00039.

[13] P. R. Chernenko and M. M. Orlova, "Security Vulnerabilities in Android OS Applications," *Visnyk Vinnytsia Politech. Inst.*, vol. 150, no. 3, pp. 43–50, 2020, doi: 10.31649/1997-9266-2020-150-3-43-50.

[14] H. W. Kim, "A Study on the Mobile Application Security Threats and Vulnerability Analysis Cases," *Int. J. Internet, Broadcast. Commun.*, vol. 12, no. 4, pp. 180–187, 2020.

[15] A. Abdel-Aziz, "Scoping Security Assesment- A Project Management Approach," *SANS Inst.*, 2011, [Online]. Available: https://www.sans.org/white-papers/33673/.

[16] J. Wi. Bernhard Mueller, Sven Schleier, "Mobile Security Testing Guide," *Area*, pp. 1–8, 2001.

[17] CERT C Secure Coding, "CWE-276 - Basis Data Keamanan." MITRE, 2009, [Online]. Available: https://www.security-database.com/cwe.php?name=CWE-276.

[18] MITRE, "CWE - CWE-922: Insecure Storage of Sensitive Information (2.11)." 2020, [Online]. Available: https://cwe.mitre.org/data/definitions/922.html.

[19] Plover, "CWE - CWE-312: Cleartext Storage of Sensitive Information (4.5)." pp. 7–19, 2006, [Online]. Available: https://cwe.mitre.org/data/definitions/312.html.

[20] PLOVER, "CWE-319: Transmisi Cleartext dari Informasi Sensitif." 2006, [Online]. Available: https://cwe.mitre.org/data/definitions/319.html.

[21] D. S. Craig de Stigter, "HTTP authentication leak in redirects - CVE-2018-1000007." 2018, [Online]. Available: https://curl.se/docs/CVE-2018-1000007.html.