MOBILE PENETRATION TESTING FINAL EXAM REPORT

Case Study 1 – Competitor Mail V2 APK (Exploit Features)

Nama : Daniel Rafael Ayorbaba

NIM : 254012023

Kelas : LB07

Checklists:

☒ Rooted Device and Emulator Detection: Bypass Detection

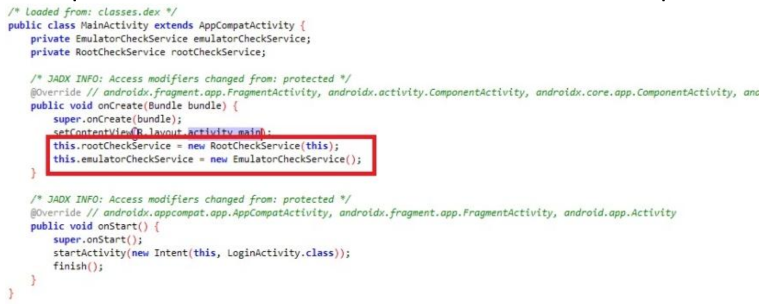
☒ Send E-mail Message: Duplicate each sent e-mail message and send to user@email.com

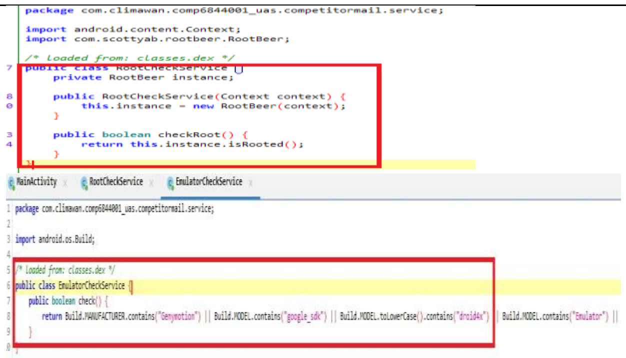
☒ Usage of Shared Preferences: Contents of Shared Preferences Are Dumped

☒ Inject a Forged Intent: Specific Activity Instantiated Without Authentication

Report:

Checklist Name	**Rooted Device and Emulator Detection**
Exploitable Status	Exploitable
Tools Used	Android Emulator, Frida, Jadx GUI, & VSCode
Information & How to Exploit	1. Buka JADX GUI lalu paste Competitor Mail 2 APK 2. Buka MainActivity 3. Click RootCheckService and EmulatorCheckService untuk lihat permission

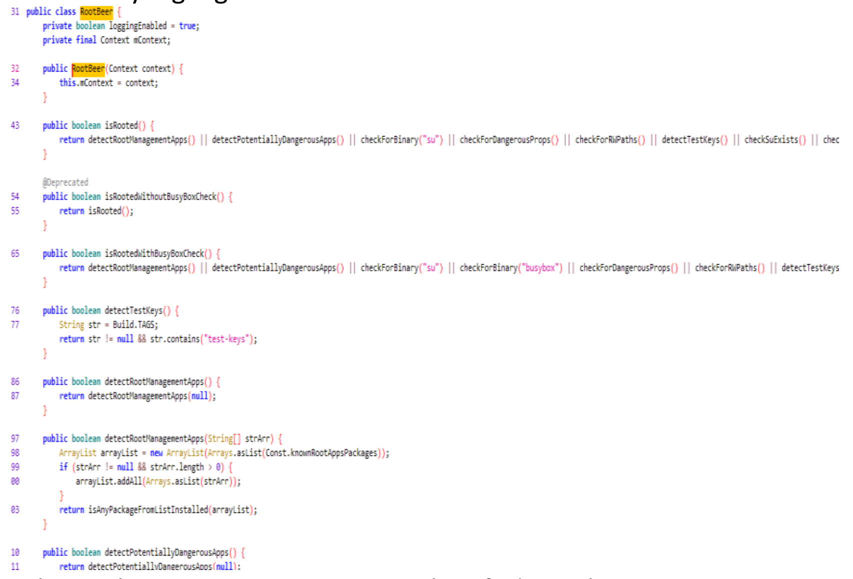
Evidence & Explanation	1. Terdapat RootCheckService and EmulatorCheckService pada MainActivity
	
	`/* loaded from: classes.dex */`
	`public class MainActivity extends AppCompatActivity {`
	`    private EmulatorCheckService emulatorCheckService;`
	`    private RootCheckService rootCheckService;`
	
	`    /* JADX INFO: Access modifiers changed from: protected */`
	`    @Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app.ComponentActivity, anc`
	`    public void onCreate(Bundle bundle) {`
	`        super.onCreate(bundle);`
	`        setContentView(R.layout.activity_main);`
	`        this.rootCheckService = new RootCheckService(this);`
	`        this.emulatorCheckService = new EmulatorCheckService();`
	`    }`
	
	`    /* JADX INFO: Access modifiers changed from: protected */`
	`    @Override // androidx.appcompat.app.AppCompatActivity, androidx.fragment.app.FragmentActivity, android.app.Activity`
	`    public void onStart() {`
	`        super.onStart();`
	`        startActivity(new Intent(this, LoginActivity.class));`
	`        finish();`
	`    }`
	`}`
	2. Pada RootCheckService and EmulatorCheckService seperti berikut



```java
package com.climawan.comp6844001_uas.competitormail.service;

import android.content.Context;
import com.scottyab.rootbeer.RootBeer;

/* loaded from: classes.dex */
public class RootCheckService {
    private RootBeer instance;

    public RootCheckService(Context context) {
        this.instance = new RootBeer(context);
    }

    public boolean checkRoot() {
        return this.instance.isRooted();
    }
}
```

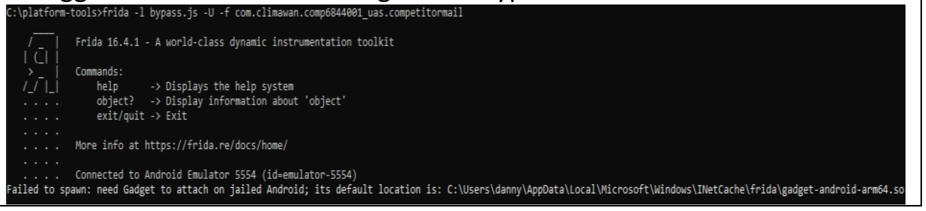
```java
package com.climawan.comp6844001_uas.competitormail.service;

import android.os.Build;

/* loaded from: classes.dex */
public class EmulatorCheckService {
    public boolean check() {
        return Build.MANUFACTURER.contains("Genymotion") || Build.MODEL.contains("google_sdk") || Build.MODEL.toLowerCase().contains("droid4x") || Build.MODEL.contains("Emulator") ||
    }
}
```

3. Gambar pada point 2 di atas menunjukkan bahwa RootCheckService menggunakan library RootBeer untuk mengecheck apakah device yang kita gunakan telah di-root atau tidak, berikut ini merupakan screenshot dari RootBeer yang digunakan:



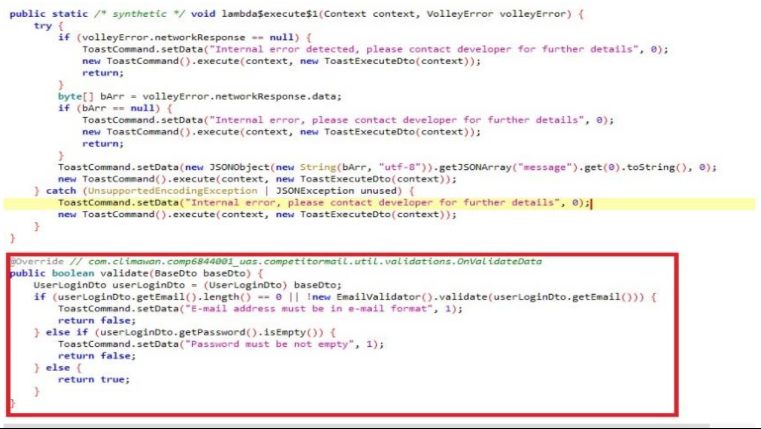
```java
public class RootBeer {
    private boolean loggingEnabled = true;
    private final Context mContext;

    public RootBeer(Context context) {
        this.mContext = context;
    }

    public boolean isRooted() {
        return detectRootManagementApps() || detectPotentiallyDangerousApps() || checkForBinary("su") || checkForDangerousProps() || checkForRWPaths() || detectTestKeys() || checkSuExists() || chec
    }

    @Deprecated
    public boolean isRootedWithoutBusyBoxCheck() {
        return isRooted();
    }

    public boolean isRootedWithBusyBoxCheck() {
        return detectRootManagementApps() || detectPotentiallyDangerousApps() || checkForBinary("su") || checkForBinary("busybox") || checkForDangerousProps() || checkForRWPaths() || detectTestKeys
    }

    public boolean detectTestKeys() {
        String str = Build.TAGS;
        return str != null && str.contains("test-keys");
    }

    public boolean detectRootManagementApps() {
        return detectRootManagementApps(null);
    }

    public boolean detectRootManagementApps(String[] strArr) {
        ArrayList arrayList = new ArrayList(Arrays.asList(Const.knownRootAppsPackages));
        if (strArr != null && strArr.length > 0) {
            arrayList.addAll(Arrays.asList(strArr));
        }
        return isAnyPackageFromListInstalled(arrayList);
    }

    public boolean detectPotentiallyDangerousApps() {
        return detectPotentiallyDangerousApps(null);
```

4. Tahap selanjutnya yaitu menggunakan frida tools serta script js yang telah dibuat namun saya terus mendapatkan error sehingga tidak dapat menggunakan Frida untuk mengecheck bypass detection

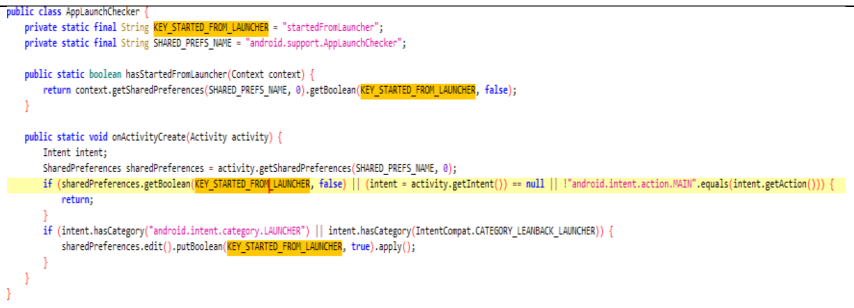


```
C:\platform-tools>frida -l bypass.js -U -f com.climawan.comp6844001_uas.competitormail
     ____
    / _  |   Frida 16.4.1 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Android Emulator 5554 (id=emulator-5554)
Failed to spawn: need Gadget to attach on jailed Android; its default location is: C:\Users\danny\AppData\Local\Microsoft\Windows\INetCache\frida\gadget-android-arm64.so
```

Checklist Name	**Send E-mail Message**
Exploitable Status	Not exploitable
Tools Used	JADX GUI
Information & How to Exploit	Tidak ada cara untuk exploit email message

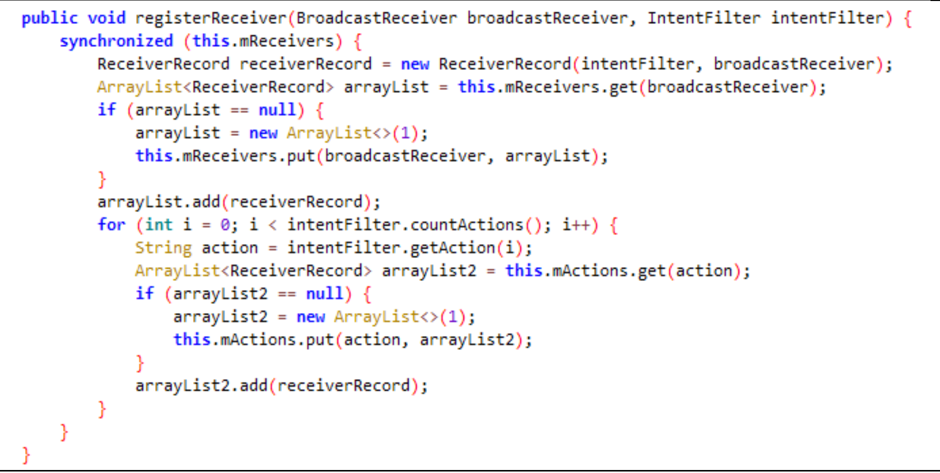
Evidence & Explanation	1. Pada LoginCommand terdapat informasi terkait email, di sini mengatakan bahwa email harus sesuai dengan formatnya serta password tidak boleh kosong. 

```java
public static /* synthetic */ void lambda$execute$1(Context context, VolleyError volleyError) {
    try {
        if (volleyError.networkResponse == null) {
            ToastCommand.setData("Internal error detected, please contact developer for further details", 0);
            new ToastCommand().execute(context, new ToastExecuteDto(context));
            return;
        }
        byte[] bArr = volleyError.networkResponse.data;
        if (bArr == null) {
            ToastCommand.setData("Internal error, please contact developer for further details", 0);
            new ToastCommand().execute(context, new ToastExecuteDto(context));
            return;
        }
        ToastCommand.setData(new JSONObject(new String(bArr, "utf-8")).getJSONArray("message").get(0).toString(), 0);
        new ToastCommand().execute(context, new ToastExecuteDto(context));
    } catch (UnsupportedEncodingException | JSONException unused) {
        ToastCommand.setData("Internal error, please contact developer for further details", 0);
        new ToastCommand().execute(context, new ToastExecuteDto(context));
    }
}

@Override // com.climawan.comp6844001_uas.competitormail.util.validations.OnValidateData
public boolean validate(BaseDto baseDto) {
    UserLoginDto userLoginDto = (UserLoginDto) baseDto;
    if (userLoginDto.getEmail().length() == 0 || !new EmailValidator().validate(userLoginDto.getEmail())) {
        ToastCommand.setData("E-mail address must be in e-mail format", 1);
        return false;
    } else if (userLoginDto.getPassword().isEmpty()) {
        ToastCommand.setData("Password must be not empty", 1);
        return false;
    } else {
        return true;
    }
}
```

Checklist Name	**Usage of Shared Preferences**
Exploitable Status	Not Exploitable
Tools Used	JADX GUI
Information & How to Exploit	Tidak ada vulnerability dan tidak dapat di-exploit
Evidence & Explanation	 Berdasarkan codingan di atas, SharedPreferences tidak dapat di-dumped dan data sensitive dari user tidak ditemukan.

```java
public class AppLaunchChecker {
    private static final String KEY_STARTED_FROM_LAUNCHER = "startedFromLauncher";
    private static final String SHARED_PREFS_NAME = "android.support.AppLaunchChecker";

    public static boolean hasStartedFromLauncher(Context context) {
        return context.getSharedPreferences(SHARED_PREFS_NAME, 0).getBoolean(KEY_STARTED_FROM_LAUNCHER, false);
    }

    public static void onActivityCreate(Activity activity) {
        Intent intent;
        SharedPreferences sharedPreferences = activity.getSharedPreferences(SHARED_PREFS_NAME, 0);
        if (sharedPreferences.getBoolean(KEY_STARTED_FROM_LAUNCHER, false) || (intent = activity.getIntent()) == null || !"android.intent.action.MAIN".equals(intent.getAction())) {
            return;
        }
        if (intent.hasCategory("android.intent.category.LAUNCHER") || intent.hasCategory(IntentCompat.CATEGORY_LEANBACK_LAUNCHER)) {
            sharedPreferences.edit().putBoolean(KEY_STARTED_FROM_LAUNCHER, true).apply();
        }
    }
}
```

Checklist Name	**Inject a Forged Intent**
Exploitable Status	Not Exploitable
Tools Used	JADX GUI
Information & How to Exploit	Tidak ada vulnerability dan tidak dapat di-exploit
Evidence & Explanation	Saya ambil salah satu contoh dari LocalBroadcastManager, dari codingan yang dibuat, kita tidak bisa menambahkan intent tambahan. Berikut ini merupakan screenshot dari codingan tersebut:

```java
public void registerReceiver(BroadcastReceiver broadcastReceiver, IntentFilter intentFilter) {
    synchronized (this.mReceivers) {
        ReceiverRecord receiverRecord = new ReceiverRecord(intentFilter, broadcastReceiver);
        ArrayList<ReceiverRecord> arrayList = this.mReceivers.get(broadcastReceiver);
        if (arrayList == null) {
            arrayList = new ArrayList<>(1);
            this.mReceivers.put(broadcastReceiver, arrayList);
        }
        arrayList.add(receiverRecord);
        for (int i = 0; i < intentFilter.countActions(); i++) {
            String action = intentFilter.getAction(i);
            ArrayList<ReceiverRecord> arrayList2 = this.mActions.get(action);
            if (arrayList2 == null) {
                arrayList2 = new ArrayList<>(1);
                this.mActions.put(action, arrayList2);
            }
            arrayList2.add(receiverRecord);
        }
    }
}
```