Android Security Project
Walk-Through
To-do-list Vulnerable Application

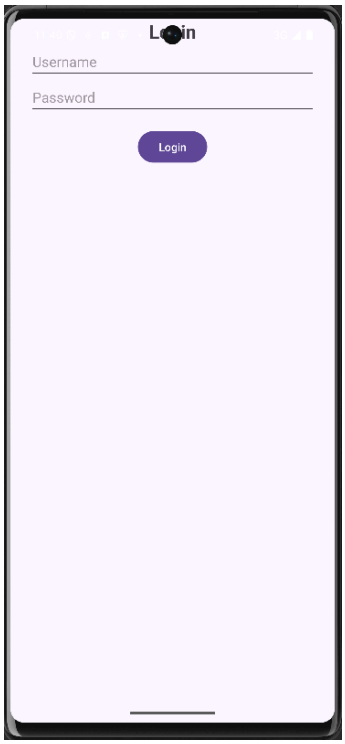
21pc07 - A Bhadri Narayanan
21pc08 - A Dharsann
21pc33 - A Shanju Shree

Task:
This APK file consists of several flags and vulnerabilities embedded inside the To-Do list application. The task is to exploit these vulnerabilities.

APK - https://drive.google.com/file/d/1p_iF1bjAWFnQVDu3wBjhij-OF3vRlv_s

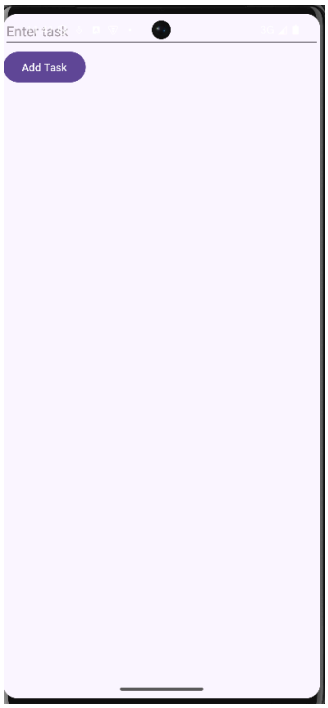
Walk-through:

This is the login page of the application , where the username and the password is hardcoded inside the **MainActivity.kt.**



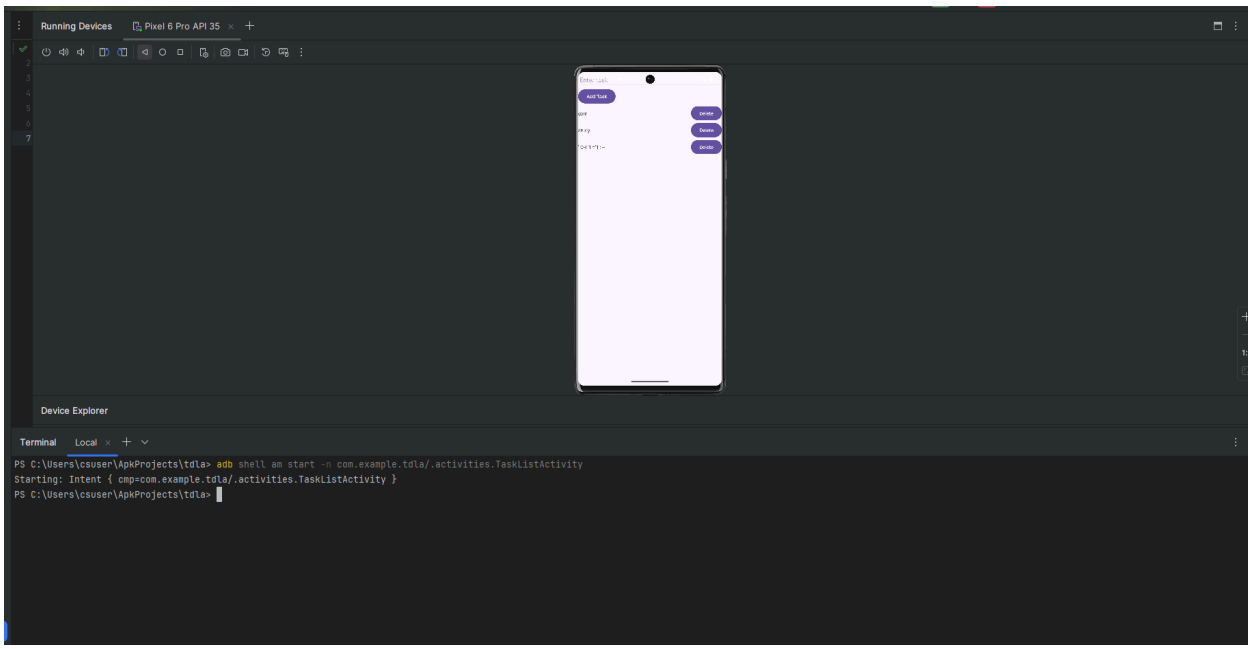
```
    }

    private final boolean validateLogin(String username, String password) {
        return Intrinsics.areEqual(username, "admin") && Intrinsics.areEqual(password, "12345");
    }
```

After entering the credentials we get a flag - **FLAG{HARDCODED_CREDS}**

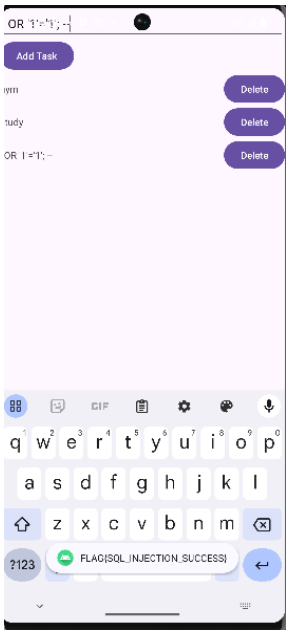


There is a hidden vulnerability using which we can login without username and password. This is through giving the “am”(Activity Manager) command in the shell. The command is as follows:

adb shell am start -n com.example.tdla/.activities.TaskListActivity



Next is the SQL-injection vulnerability , where the payload to be given is **‘ OR ‘1’=’1’;** – We exploit this vulnerability to get the flag. The flag is - **FLAG{SQL_INJECTION_SUCCESS}**

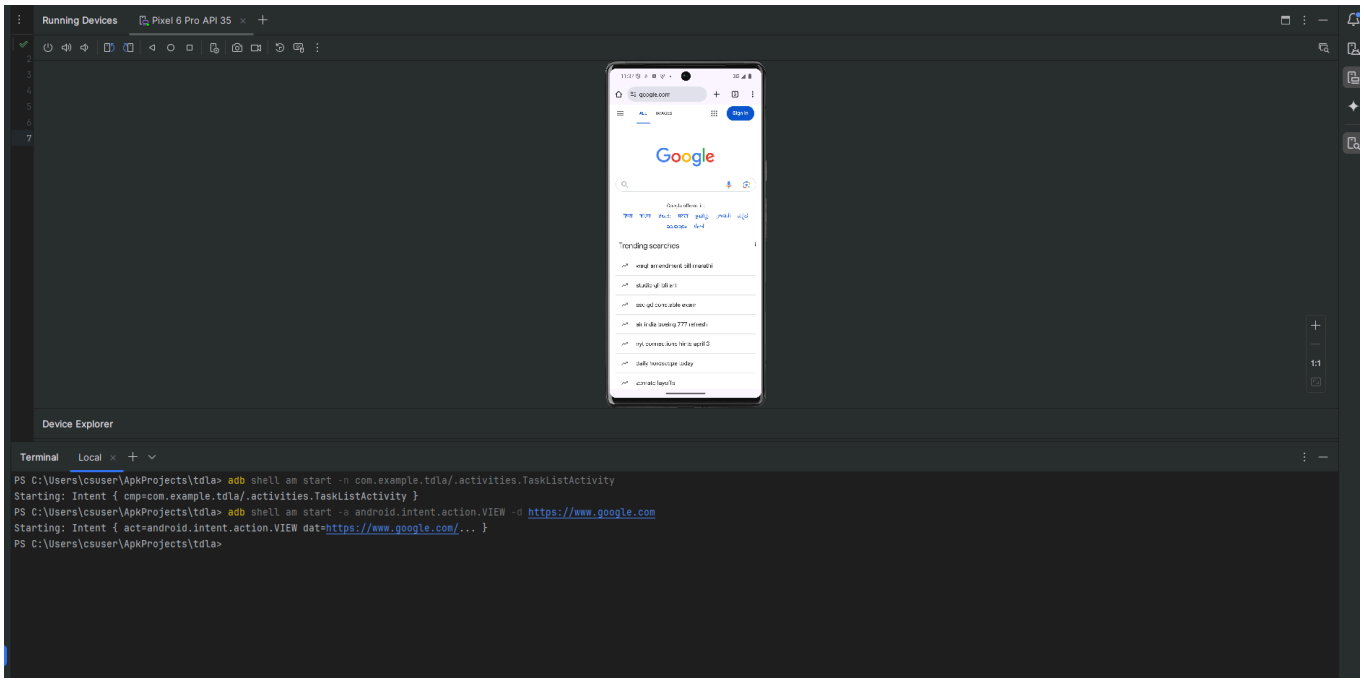


Next we explore the strings.xml file in the res/values directory to obtain the next flag - **FLAG{HIDDEN_IN_STRINGS_XML}**

res/values/strings.xml | MainActivity | TaskListActivity | WebViewActivity | CryptoUtils

```
<string name="bottomsheet_action_expand_halfway">Expand halfway</string>
<string name="bottomsheet_drag_handle_clicked">Drag handle double-tapped</string>
<string name="bottomsheet_drag_handle_content_description">Drag handle</string>
<string name="call_notification_answer_action">Answer</string>
<string name="call_notification_answer_video_action">Video</string>
<string name="call_notification_decline_action">Decline</string>
<string name="call_notification_hang_up_action">Hang Up</string>
<string name="call_notification_incoming_text">Incoming call</string>
<string name="call_notification_ongoing_text">Ongoing call</string>
<string name="call_notification_screening_text">Screening an incoming call</string>
<string name="character_counter_content_description">Characters entered %1$d of %2$d</string>
<string name="character_counter_overflowed_content_description">Character limit exceeded %1$d of %2$d</string>
<string name="character_counter_pattern">%1$d/%2$d</string>
<string name="clear_text_end_icon_content_description">Clear text</string>
<string name="error_a11y_label">Error: invalid</string>
<string name="error_icon_content_description">Error</string>
<string name="exposed_dropdown_menu_content_description">Show dropdown menu</string>
<string name="fab_transformation_scrim_behavior">com.google.android.material.transformation.FabTransformationScrimBehavior</string>
<string name="fab_transformation_sheet_behavior">com.google.android.material.transformation.FabTransformationSheetBehavior</string>
<string name="hidden_flag">FLAG{HIDDEN_IN_STRINGS_XML}</string>
<string name="hide_bottom_view_on_scroll_behavior">com.google.android.material.behavior.HideBottomViewOnScrollBehavior</string>
<string name="icon_content_description">Dialog Icon</string>
<string name="item_view_role_description">Tab</string>
<string name="m3_exceed_max_badge_text_suffix">%1$s%2$s</string>
<string name="m3_ref_typeface_brand_medium">sans-serif-medium</string>
<string name="m3_ref_typeface_brand_regular">sans-serif</string>
<string name="m3_ref_typeface_plain_medium">sans-serif-medium</string>
<string name="m3_ref_typeface_plain_regular">sans-serif</string>
<string name="m3_sys_motion_easing_emphasized">path(M 0,0 C 0.05, 0, 0.133333, 0.06, 0.166666, 0.4 C 0.208333, 0.82, 0.25, 1, 1, 1)
<string name="m3_sys_motion_easing_emphasized_accelerate">cubic-bezier(0.3, 0, 0.8, 0.2)</string>
<string name="m3_sys_motion_easing_emphasized_decelerate">cubic-bezier(0.1, 0.7, 0.1, 1)</string>
<string name="m3_sys_motion_easing_emphasized_path_data">M 0,0 C 0.05, 0, 0.133333, 0.06, 0.166666, 0.4 C 0.208333, 0.82, 0.25, 1,
<string name="m3_sys_motion_easing_legacy">cubic-bezier(0.4, 0, 0.2, 1)</string>
<string name="m3_sys_motion_easing_legacy_accelerate">cubic-bezier(0.4, 0, 1, 1)</string>
<string name="m3_sys_motion_easing_legacy_decelerate">cubic-bezier(0, 0, 0.2, 1)</string>
<string name="m3_sys_motion_easing_linear">cubic-bezier(0, 0, 1, 1)</string>
<string name="m3_sys_motion_easing_standard">cubic-bezier(0.2, 0, 0, 1)</string>
<string name="m3_sys_motion_easing_standard_accelerate">cubic-bezier(0.3, 0, 1, 1)</string>
```

There is another vulnerability which can be prompted in the shell which allows us to open any URL such as a phishing link from a remote attacker directly.

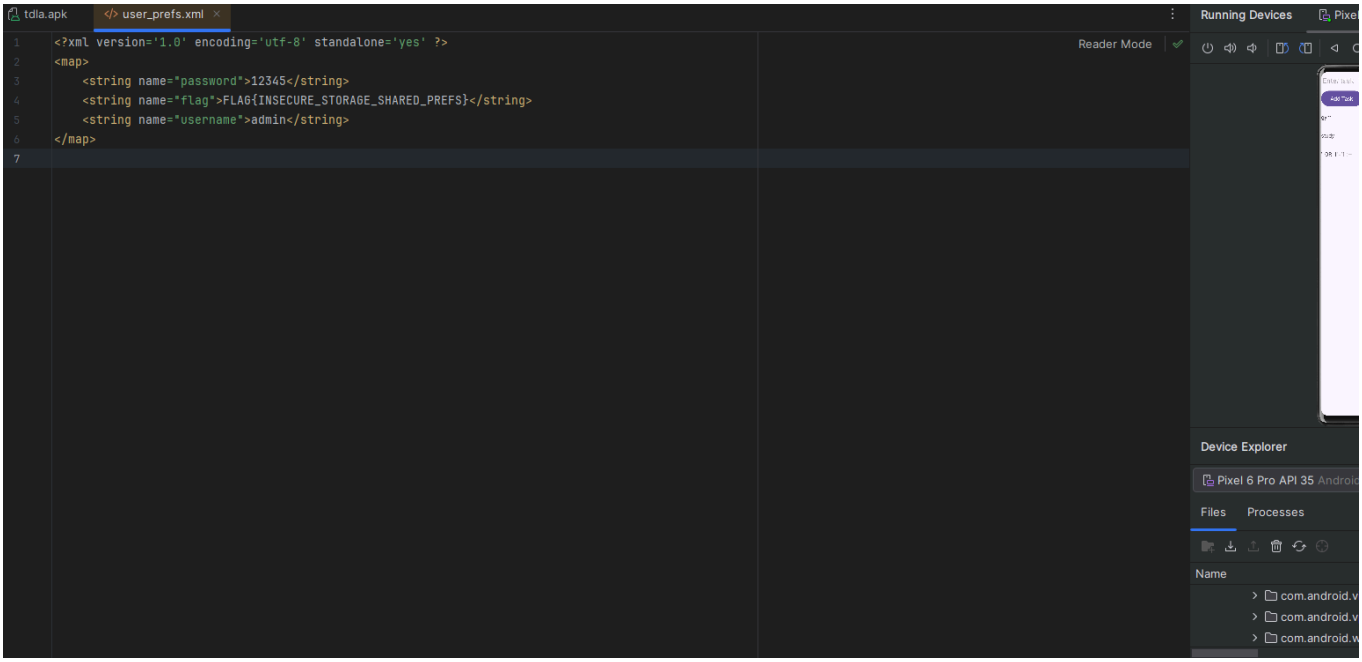


6) Go to **View -> Device Explorer** and go to this directory **/data/data/com.example.tdla/shared_prefs/user_prefs.xml.**

We can find the flag - **FLAG{INSECURE_STORAGE_SHARED_PREFS}**

```
    private final void saveToSharedPrefs(String username, String password) {
        SharedPreferences sharedPreferences = this.sharedPreferences;
        if (sharedPreferences == null) {
            Intrinsics.throwUninitializedPropertyAccessException("sharedPreferences");
            sharedPreferences = null;
        }
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString("username", username);
        editor.putString("password", password);
        editor.putString("flag", "FLAG{INSECURE_STORAGE_SHARED_PREFS}");
        editor.apply();
    }
}
```

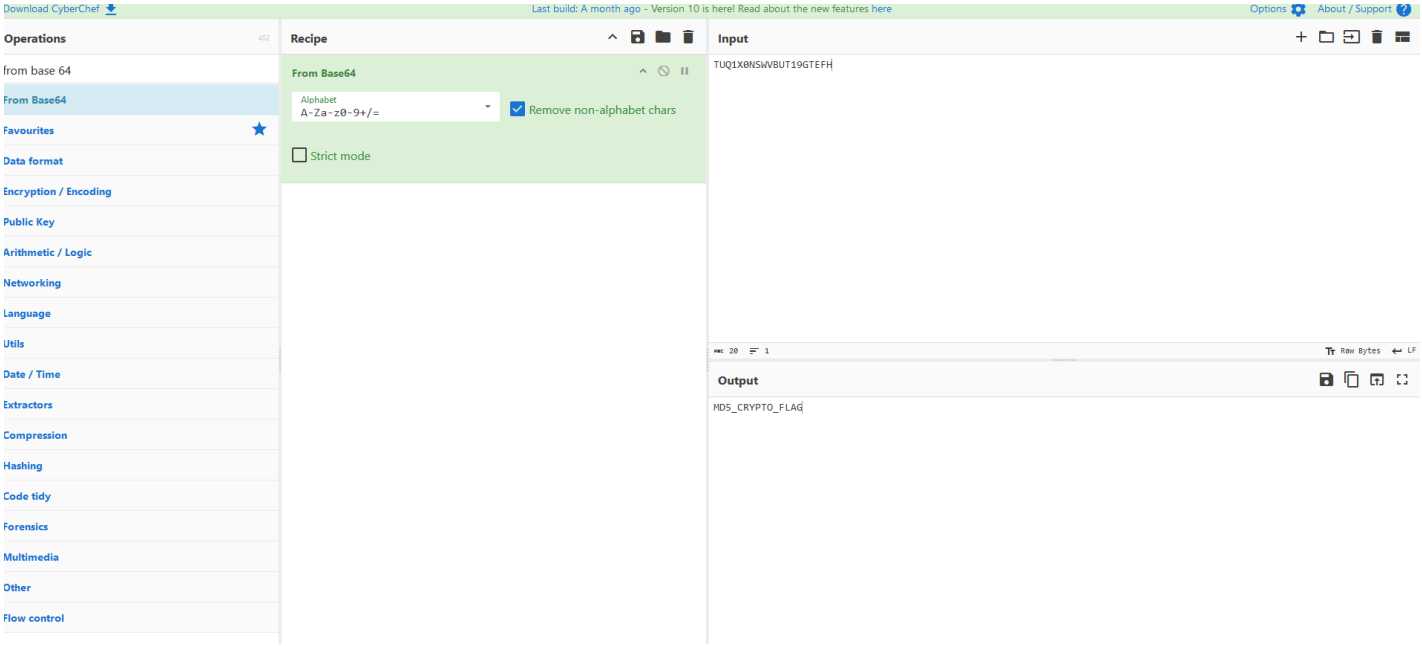
-



7) Next we have a cryptographic vulnerability inside the **CryptoUtils.kt** under utils directory in which we can discover our next flag. This will be in the encoded format which can be obtained through CyberChef (Recipe - to base64).

res/values/strings.xml | MainActivity | TaskListActivity | WebViewActivity | CryptoUtils

```
package com.example.tdla.utils;

import android.util.Base64;
import androidx.constraintlayout.widget.ConstraintLayout;
import java.security.MessageDigest;
import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import kotlin.text.Charsets;

/* compiled from: CryptoUtils.kt */
@Metadata(d1 = {"\u0000\u001c\n\u0002\u0018\u0002\n\u0002\u0010\u0000\n\u0002\b\u0002\n\u0002\u0010\u000e\n\u0002\b\u0002\n\u0002\u0010\u0012\n\u0002\b\u0002\b€\u0002\u0018\u00002\u00020\u0001
/* loaded from: classes4.dex */
public final class CryptoUtils {
    public static final CryptoUtils INSTANCE = new CryptoUtils();

    private CryptoUtils() {
    }

    public final byte[] md5HashRaw(String input) {
        Intrinsics.checkNotNullParameter(input, "input");
        MessageDigest messageDigest = MessageDigest.getInstance("MD5");
        byte[] bytes = input.getBytes(Charsets.UTF_8);
        Intrinsics.checkNotNullExpressionValue(bytes, "getBytes(...)");
        byte[] digest = messageDigest.digest(bytes);
        Intrinsics.checkNotNullExpressionValue(digest, "digest(...)");
        return digest;
    }

    private final String getSecretValue() {
        byte[] decode = Base64.decode("TUQ1X0NSWVBUT19GTEFH", 2);
        Intrinsics.checkNotNullExpressionValue(decode, "decode(...)");
        return new String(decode, Charsets.UTF_8);
    }

    public final String getWeakCryptoFlag() {
        String secretValue = getSecretValue();
        byte[] rawHash = md5HashRaw(secretValue + "s3cur3s@lt");
        String encodedFlag = Base64.encodeToString(rawHash, 2);
        return "FLAG{" + encodedFlag + "}";
    }
}
```



The flag is : **MD5_CRYPTO_FLAG**

The List of Flags and Vulnerabilities in TDLA:

1. **FLAG{HARDCODED_CREDS}**
2. **FLAG{SQL_INJECTION_SUCCESS}**
3. **FLAG{HIDDEN_IN_STRINGS_XML}**
4. **FLAG{INSECURE_STORAGE_SHARED_PREFS}**
5. **MD5_CRYPTO_FLAG**
6. Hardcoded Credentials
7. Open an Activity without Login using adb
8. Open any Url using adb