Hardcoding Issues

Intdroduction

- Hard coding refers to the practice of embedding data directly into the source code of an application.
- Hard coding can lead to security vulnerabilities and maintenance challenges.
- Attackers can reverse engineer the applications and get hardly coded data and use them in attack vectors.

Practical Demonstration

Hard Coding Issues - Part 1

At first glance, hard coding task asks for vendor password but there is no way to get the password in a logical way.

Try to write anything in box

when you write anything in the text box, the app will give you an error that says **password is incorrect**.

So, lets reverse engineer the application with **JADX** to see what happens in the corresponding class (**jakhar.aseem.diva.HardCodeActivity**).

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hardcode);
    }

    public void access(View view) {
        EditText hckey = (EditText) findViewById(R.id.hcKey);
        if (hckey.getText().toString().equals("vendorsecretkey")) {
            Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
        } else {
            Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
        }
    }
}
```

we can see that the vendor key is hardcoded, thus we can copy the "**vendorsecretkey**" value and pop it into our app.

Hard Coding Issues - Part 2

By exmine **Hardcode2Activity** in **JADX-GUI** and open it up. We can immediately identify the DivaJni class that is being referenced, as the rest of the code depends on this class to be able to validate the access key.

1. **Identify Where It's Hardcoded:** In JADX-GUI, navigate to the `Hardcode2Activity` and open it. You can identify the `DivaJni` class, which is used to validate the access key.

```
/* loaded from: classes.dex */
public class Hardcode2Activity extends AppCompatActivity {
    private DivaJni djni;

    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, a
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hardcode2);
        this.djni = new DivaJni();
    }

    public void access(View view) {
        EditText hckey = (EditText) findViewById(R.id.hc2Key);
        if (this.djni.access(hckey.getText().toString()) != 0) {
            Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
        } else {
            Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
        }
    }
}
```

2. **Explore DivaJni Class:** Open the `DivaJni` class and observe the code. You initially notice the string `soName = "divajni"`, but it's not the hardcoded value.

```
/* loaded from: classes.dex */
public class DivaJni {
    private static final String soName = "divajni";

    public native int access(String str);

    public native int initiateLaunchSequence(String str);

    static {
        System.loadLibrary(soName);
    }
}
```

3. **Investigate the Static Function:** Further inspection reveals that the static function loads the native library with the name `divajni`. This library is loaded dynamically during Android runtime.
4. **Pull apk in system:** Use the following ADB command to find the path to the Diva APK file on your device:

`adb shell pm list packages | grep diva`

`adb shell pm path jakhar.aseem.diva`

Once you have the package name, use the following ADB command to pull the APK file to your computer:

```
adb pull /data/app/jakhar.aseem.diva_jKQ4XlxNqDbFiZUUvhJ2A==/base.apk
```

5 Access the Library File: Within the extracted APK, go to the `lib` folder and find the file `libdivajni.so`

`cd lib`

`ls`

`cd arm64-v8a`

`ls`

here we canfind libdivajni.so file

6 View the Library Text: To view the content of the library file, use the `strings` command in the terminal. Open your terminal, navigate to the folder containing the `libdivajni.so` file, and run

`strings libdivajni.so`.