WriteUp - TAG Mobile - Processo Seletivo GRIS 21.1

David Rodrigues Albuquerque[1]

[1]davidra@dcc.ufrj.br - 120047390

Abril, 2021

Sumário

1 Instalação das Ferramentas e da Aplicação

Primeiramente, foi necessário instalar as ferramentas necessárias para a instalação do app e análise do mesmo. Foi utilizado para instalação do aplicativo o ADB. Foram utilizados em conjunto o apktool e o Jadx para realizar a análise do apk. Por último, foi instalado o frida-tools para escrever o script de bypass. Todos foram instalados através do gerenciador de pacotes Chocolatey e /hrefhttps://pypi.org/project/pip/pip.

A instalação foi realizada no Powershell através dos comandos:

```
choco install adb
choco install apktool
choco install jadx
pip install frida-tools
```

A seguir, foi necessário ligar a depuração USB no dispositivo móvel em que foi instalado o apk nas configurações de desenvolvedor do aparelho. Após conectar o dispositivo com a permissão de depuração, foi utilizado o ADB para checar se o dispositivo estava conectado e instalar a aplicação.

```
adb devices
adb install UnCrackable-Level1.apk
```

2 Decompilação do Código

Primeiramente, foi utilizado o apktool para decompilar a aplicação e visualizar o AndroidManifest.xml afim de encontrar a Activity que realiza a checagem de root. Foi utilizado o comando

```
apktool d UnCrackable-Level1.apk
```

O resultado obtido nos permite visualizar que a primeira Activity a ser executada ao inicializar o app é a MainActivity, que poderá nos levar as funções que realizam a checagem se o aparelho está rootado.

```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android" package="owasp.mstg.uncrackable1">
    <application android:allowBackup="true" android:icon="@mipmap/ic_launcher" android:label="@string/app_name" android:theme="@style/AppTheme">
        <activity android:label="@string/app_name" android:name="sg.vantagepoint.uncrackable1.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Figure 1: AndroidManifest.xml

3 Análise Estática

Após analisar o AndroidManifest, foi utilizado o Jadx para fazer a análise estática dos arquivos da apk. Como esperado, navegado para a MainActivity, foram encontradas as verificações realizadas para checar se o usuário está rootado.

```java
public class MainActivity extends Activity {
    private void a(String str) {
        AlertDialog create = new AlertDialog.Builder(this).create();
        create.setTitle(str);
        create.setMessage("This is unacceptable. The app is now going to exit.");
        create.setButton(-3, "OK", new DialogInterface.OnClickListener() {
            /* class sg.vantagepoint.uncrackable1.MainActivity.AnonymousClass1 */

            public void onClick(DialogInterface dialogInterface, int i) {
                System.exit(0);
            }
        });
        create.setCancelable(false);
        create.show();
    }

    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        if (c.a() || c.b() || c.c()) {
            a("Root detected!");
        }
        if (b.a(getApplicationContext())) {
            a("App is debuggable!");
        }
        super.onCreate(bundle);
        setContentView(R.layout.activity_main);
    }
```

Figure 2: sg.vantagepoint.uncrackable1.MainActivity

Como a aplicação não está em modo debug, precisamos apenas do bypass das funções c.a(), c.b() e c.c(). Cada uma destas podem ser encontradas na class a do pacote sg.vantagepoint.a

```java
public class c {
    public static boolean a() {
        for (String str : System.getenv("PATH").split(":")) {
            if (new File(str, "su").exists()) {
                return true;
            }
        }
        return false;
    }

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

    public static boolean c() {
        for (String str : new String[]{"/system/app/Superuser.apk", "/system/xbin/daemonsu", "
            if (new File(str).exists()) {
                return true;
            }
        }
        return false;
    }
}
```

Figure 3: sg.vantagepoint.a

4 Root Bypass

Para realizar o bypass, foram sobrescritas as implementações dos métodos contains (java.lang.String) e exists (java.io.File) utilizando um frida script. Com isto, foi feito com que, ao detectar as Strings que detectariam que o aparelho está rootado (caminhos de arquivos, extensões, etc.) ou ao detectar que a Build TAG "test-keys" está presente, ao invés de retornar true, retorna falso, indicando que as mesmas não existem. Desta forma, os métodos a(), b() e c() vão sempre retornar falso, possibilitando que a verificação seja bypassada.

```
Java.perform(function() {

    var NativeFile = Java.use('java.io.File');
    var RootBinaries = ["su", "busybox", "supersu", "Superuser.apk", "KingoUser.apk", "SuperSu.apk", "/system/app/Superuser.apk",
                        "/system/xbin/daemonsu", "/system/etc/init.d/99SuperSUDaemon", "/system/bin/.ext/.su",
                        "/system/etc/.has_su_daemon","/system/etc/.installed_su_daemon", "/dev/com.koushikdutta.superuser.daemon/"];
    var String = Java.use('java.lang.String');

    String.contains.implementation = function(name) {
        if (name == "test-keys") {
            send("Bypass test-keys check");
            return false;
        }
        return this.contains.call(this, name);
    };

    NativeFile.exists.implementation = function() {
        var name = NativeFile.getName.call(this);
        var shouldFakeReturn = (RootBinaries.indexOf(name) > -1);
        if (shouldFakeReturn) {
            send("Bypass return value for binary: " + name);
            return false;
        } else {
            return this.exists.call(this);
        }
    };
});
```

Figure 4: Código disponível em: https://github.com/Albuquerque-David/gris_treinamento/blob/main/Mobile/bypass.js

5 Referências Bibliográficas

https://codeshare.frida.re/@dzonerzy/fridantiroot/ https://book.hacktricks.xyz/mobile-apps-pentesting/android-app-pentesting/frida-tutorial