这个app有root检测 有以下几种方式解决

- **通过反编译找到检测root的java代码 进而hook java**
- **通过frida反root框架FridaAntiRootDetection github搜就行**
- **通过定制aosp 隐藏或修改su文件**
- **通过面具隐藏root 面具+Shamiko模块方案+隐藏magisk应用**

这里通过FridaAntiRootDetection和hook java分别实现

FridaAntiRootDetection直接github搜 不解释

java搜提示的文字 找到如下图片

```java
private void isRootPhone() {
    if (!MiscUtil.isSimulator(this) && !MiscUtil.isRooted()) {
        initVariables();
        if (NetUtil.getSharedPreferences(SettingsConfig.KEY_USER_PERMISSION, false)) {
            ScreenAdLoader.pullSSPAd(true);
            return;
        }
        return;
    }
    DialogFactory dialogFactory = new DialogFactory((Context) this, "温馨提示", "运行大姨妈在Root设备或模拟器上,将威胁您的数据安全请您在正常设备上安装使用~",
        @Override // com.yoloho.controller.dialog.customdialog.DialogCallBack
        public void negativeOnClickListener() {
        }

        @Override // com.yoloho.controller.dialog.customdialog.DialogCallBack
        public void positiveOnClickListener() {
            Launcher.this.finish();
        }

        @Override // com.yoloho.controller.dialog.customdialog.DialogCallBack
        public void titleRightOnClickListener() {
        }
    });
    dialogFactory.setCancel(false);
    dialogFactory.setNamePositiveButton("退出");
```

发现这里检测模拟器和root 因从我们把这两个函数置成false

```java
Java.perform(function () {
    var MiscUtil = Java.use("com.yoloho.libcore.util.MiscUtil");

    MiscUtil.isRooted.implementation = function () {
        return false;
    }

    MiscUtil.isSimulator.implementation = function (ctx) {
        return false;
    }


    var SystemManager = Java.use("com.mobile.auth.gatewayauth.manager.SystemManager");

    SystemManager.checkEnvSafe.implementation = function () {
        return null;
    }
});
```

解决完这个问题 开始抓包

Overview Contents Summary Chart Notes

Name	Value
username	13849857524
password	u1uNlVbP2uH3/BOU8sNszw==
sign	a63dd19e35a910332da573376d376506
androidid	ba76afc0e1370865
mac	02:00:00:00:00:00
imei	
density	2.75
brand	Redmi

device	b9bfeb09bf69b23a47ddb7cd2806cfc05b55e5bd
ver	630
screen_width	1080
screen_height	2240
model	23049RAD8C
sdkver	33
platform	android
releasever	13
channel	360
latt	0
lngt	0
networkType	0
token	
userStatus	0

需要破解的参数

url中device勉强算 device不变

请求头中没有

请求体中

- **password**
- **sign**

经过多次发请求分析,发现sign应该是由password和其他字段的来,password不变 sign不变

现在先搜索请求url看看 有两个结果

节点	
com.yoloho.kangseed.presenter.entrance.LoginByAccountPresenter.AnonymousClass3.call(Object) void	sb.append("user/login");
com.yoloho.kangseed.view.activity.entance.LoginBaseActivity.getLoginData(String, String) JSONObject	sb.append("user/login");

```java
public void loginByAccount(final String str, final String str2) {
    getLoginLoading().show();
    Observable.create(new Observable.OnSubscribe<JSONObject>() { // from class: com.yoloho.kangseed.present
        @Override // rx.functions.Action1
        public void call(Subscriber<? super JSONObject> subscriber) {
            JSONObject jSONObject;
            ArrayList arrayList = new ArrayList();
            arrayList.add(new BasicNameValuePair("username", str));
            String privateStrHandle = DayimaPrivateUtil.privateStrHandle(str2, str);
            arrayList.add(new BasicNameValuePair("password", privateStrHandle));
            StringBuilder sb = new StringBuilder();
            sb.append(PeriodAPIV2.getInstance().getDeviceCode());
            sb.append("user/login");
            sb.append(str);
            sb.append(privateStrHandle);
            arrayList.add(new BasicNameValuePair("sign", Crypt.encrypt_data(0L, sb.toString(), sb.length()))
            try {
                jSONObject = PeriodAPIV2.getInstance().api("user", "login", arrayList);
            } catch (ServiceException e2) {
                e2.printStackTrace();
                jSONObject = null;
            }
            subscriber.onNext(jSONObject);
        }
    }).subscribeOn(Schedulers.newThread()).observeOn(AndroidSchedulers.mainThread()).subscribe(new Observer
        @Override // rx.Observer
        public void onCompleted() {
        }

        @Override // rx.Observer
        public void onError(Throwable th) {
            LoginByAccountPresenter.this.getLoginLoading().dismiss();
        }


private JSONObject getLoginData(String str, String str2) {
    Message obtainMessage = this.handler.obtainMessage();
    obtainMessage.what = 5;
    this.handler.sendMessage(obtainMessage);
    ArrayList arrayList = new ArrayList();
    arrayList.add(new BasicNameValuePair("username", str));
    arrayList.add(new BasicNameValuePair("password", str2));
    StringBuilder sb = new StringBuilder();
    sb.append(PeriodAPIV2.getInstance().getDeviceCode());
    sb.append("user/login");
    sb.append(str);
    sb.append(str2);
    arrayList.add(new BasicNameValuePair("sign", Crypt.encrypt_data(0L, sb.toString(), sb.length())));
    try {
        return PeriodAPIV2.getInstance().api("user", "login", arrayList);
    } catch (ServiceException e2) {
        e2.printStackTrace();
        return null;
    }
}
```

发现loginByAccount是请求的入口

```
JSONObject jSONObject;
ArrayList arrayList = new ArrayList();
arrayList.add(new BasicNameValuePair("username", str));
String privateStrHandle = DayimaPrivateUtil.privateStrHandle(str2, str);
arrayList.add(new BasicNameValuePair("password", privateStrHandle));
StringBuilder sb = new StringBuilder();
```

password加密方式也看出来了

str2是密码 str是账号

```
public static String privateStrHandle(String str, String str2) {
    String encrypt = AESUtil.encrypt(str, MD5Util.getMD5(str2).substring(0, 16).toLowerCase(), "yoloho_dayima!%_");
    return TextUtils.isEmpty(encrypt) ? str : encrypt;
}
```

显然是把账号得到md5字符串取前16位 在经过aes加密

```
/* loaded from: classes4.dex */
public class AESUtil {
    public static String encrypt(String str, String str2, String str3) {
        try {
            return Base64.encodeBytes(genAESCipher(str2, str3, 1).doFinal(str.getBytes("utf-8")));
        } catch (Exception unused) {
            return "";
        }
    }

    public static Cipher genAESCipher(String str, String str2, int i2) {
        SecretKeySpec secretKeySpec = new SecretKeySpec(str.getBytes(), "AES");
        IvParameterSpec ivParameterSpec = new IvParameterSpec(str2.getBytes());
        try {
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
            cipher.init(i2, secretKeySpec, ivParameterSpec);
            return cipher;
        } catch (InvalidAlgorithmParameterException e2) {
            throw new RuntimeException(e2);
        } catch (InvalidKeyException e3) {
            throw new RuntimeException(e3);
        } catch (NoSuchAlgorithmException e4) {
            throw new RuntimeException(e4);
        } catch (NoSuchPaddingException e5) {
            throw new RuntimeException(e5);
        }
    }
}
```

aes是个cbc模式的加密 encrypt第一个参数是密码 第二个参数是对账号进行md5加密取前16位 第三个参数是固定字符串"yoloho_dayima!%_"

```
AESUtil.encrypt(str, MD5Util.getMD5(str2).substring(0, 16).toLowerCase(),
"yoloho_dayima!%_");
```

```java
/* loaded from: classes4.dex */
public class AESUtil {
    public static String encrypt(String str, String str2, String str3) {
        try {
            return Base64.encodeBytes(genAESCipher(str2, str3, 1).doFinal(str.getBytes("utf-8")));
        } catch (Exception unused) {
            return "";
        }
    }

    public static Cipher genAESCipher(String str, String str2, int i2) {
        SecretKeySpec secretKeySpec = new SecretKeySpec(str.getBytes(), "AES");
        IvParameterSpec ivParameterSpec = new IvParameterSpec(str2.getBytes());
        try {
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
            cipher.init(i2, secretKeySpec, ivParameterSpec);
            return cipher;
        } catch (InvalidAlgorithmParameterException e2) {
            throw new RuntimeException(e2);
        } catch (InvalidKeyException e3) {
            throw new RuntimeException(e3);
        } catch (NoSuchAlgorithmException e4) {
            throw new RuntimeException(e4);
        } catch (NoSuchPaddingException e5) {
            throw new RuntimeException(e5);
        }
    }
}
```

第二个参数是key 第三个参数是iv 第一个参数是加密字符 加密之后转base64

即key是对账号进行md5加密取前16位 iv是yoloho_dayima!%_取字节 对密码加密后转base64

用python实现

```python
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import base64
from hashlib import md5

# 加密过程
# key必须是16 24 或者32位 分别对应AES-128 AES-192 AES-256
md5_encoder = md5()
md5_encoder.update("13849857524".encode('utf-8'))
key = md5_encoder.hexdigest()[:16].encode('utf-8')
aes = AES.new(key=key, mode=AES.MODE_CBC, iv=b"yoloho_dayima!%_")
s = "123456"
s_pad = pad(s.encode("utf-8"), 16)
s_jiami = aes.encrypt(s_pad)
s_b64 = base64.b64encode(s_jiami).decode("utf-8")
print(s_b64)
```

```
u1uNlVbP2uH3/BOU8sNszw==
```

username	13849857524
password	u1uNlVbP2uH3/BOU8sNszw==
sign	a63dd19e35a910332da573376d376506
androidid	ba76afc0e1370865
mac	02:00:00:00:00:00
imei	
density	2.75
brand	Redmi

发现跟我们分析的完全一致 这就说明我们的分析完全没错 到此password分析完成 看看sign

```
public void call(Subscriber<? super JSONObject> subscriber) {
    JSONObject jSONObject;
    ArrayList arrayList = new ArrayList();
    arrayList.add(new BasicNameValuePair("username", str));
    String privateStrHandle = DayimaPrivateUtil.privateStrHandle(str2, str);
    arrayList.add(new BasicNameValuePair("password", privateStrHandle));
    StringBuilder sb = new StringBuilder();
    sb.append(PeriodAPIV2.getInstance().getDeviceCode());
    sb.append("user/login");
    sb.append(str);
    sb.append(privateStrHandle);
    arrayList.add(new BasicNameValuePair("sign", Crypt.encrypt_data(0L, sb.toString(), sb.length())));
    try {
        jSONObject = PeriodAPIV2.getInstance().api("user", "login", arrayList);
    } catch (ServiceException e2) {
        e2.printStackTrace();
        jSONObject = null;
    }
    subscriber.onNext(jSONObject);
}
```

sign是几个参数拼接得来的 第一个参数是固定值 b9bfeb09bf69b23a47ddb7cd2806cfc05b55e5bd 每个手机不一样 具体可以看这个方法 这里不看了 之后都已经知道了

看看加密函数

```
Crypt.encrypt_data is called: j2=0, str=b9bfeb09bf69b23a47ddb7cd2806cfc05b55e5bduser/login13849857524u1uNlVbF
Crypt.encrypt_data result=a63dd19e35a910332da573376d376506
```

第一个参数是0 第二个参数是个字符串 第三个参数是这个字符串的长度

```
/* loaded from: classes4.dex */
public class Crypt {
    static {
        System.loadLibrary("Crypt");
    }

    public static native String encrypt_data(long j2, String str, long j3);
}
```

显然这个函数在so文件中

反编译so文件之后 看到如下

```
2 {
3   jsize v7; // w22
4   const char *v8; // x0
5   char v10[36]; // [xsp+4h] [xbp-5Ch] BYREF
6   __int64 v11; // [xsp+28h] [xbp-38h]
7
8   v11 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
9   v7 = a1->functions->GetStringUTFLength((JNIEnv *)a1, (jstring)a4);
10  v8 = a1->functions->GetStringUTFChars(a1, a4, 0LL);
11  sub_1DA0(a3, v8, v7, v10);
12  return a1->functions->NewStringUTF(a1, v10);
13 }
```

```
1 __int64 __fastcall sub_1DA0(__int64 a1, char *a2, __int64 a3, char *a4)
2 {
3   unsigned int v7; // w9
4   bool v8; // cf
5   int v9; // w8
6   __int64 v10; // x26
7   __int64 v11; // x23
8   unsigned int v12; // w23
9   bool v13; // cc
10  __int64 v14; // x21
11  __int64 v15; // x22
12  __int64 v16; // x8
13  __int128 v17; // q0
14  __int128 v18; // q2
15  __int128 v19; // q3
16  unsigned int v20; // w24
17  unsigned int v21; // w8
18  __int128 v23; // [xsp+20h] [xbp-C0h] BYREF
19  __int64 v24; // [xsp+30h] [xbp-B0h]
20  _OWORD v25[4]; // [xsp+38h] [xbp-A8h] BYREF
21  __int128 v26; // [xsp+78h] [xbp-68h] BYREF
22  char v27[16]; // [xsp+88h] [xbp-58h] BYREF
23  __int64 v28; // [xsp+98h] [xbp-48h]
24
25  v28 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
26  *(_OWORD *)a4 = 0u;
27  *((_OWORD *)a4 + 1) = 0u;
28  a4[32] = 0;
29  v24 = 0LL;
30  v23 = xmmword_24D0;
31  v26 = xmmword_24E0;
32  sub_8D0(&v23, a1);
33  v7 = v24;
34  v8 = __CFADD__((_DWORD)v24, 8 * a3);
00001DA0 sub_1DA0:1 (1DA0)
```

看到sub函数 发现太复杂 不是常见的加密 没有字符串出现

直接使用frida_rpc调用

```
import frida

rdev = frida.get_remote_device()
session = rdev.attach("大姨妈")

scr = """
rpc.exports = {
    xx:function(j2,str,j3){
        var res;
        Java.perform(function () {
            // 包.类
            var Crypt = Java.use("com.yoloho.libcore.util.Crypt");
            // 类中的方法
            res = Crypt.encrypt_data(j2,str,j3);
```

```
        });

        return res;
    }
}
"""
script = session.create_script(scr)
script.load()

# python 调用
sign = script.exports_sync.xx(0,
"b9bfeb09bf69b23a47ddb7cd2806cfc05b55e5bduser/login13849857524u1uNlVbP2uH3/BOU8sNszw==", 85)
print(sign)
```

发现结果和抓包相同 所以总的代码如下:

```
import frida
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import base64
from hashlib import md5
import requests

# 加密过程
# key必须是16 24 或者32位 分别对应AES-128 AES-192 AES-256
# username = str(input("输入手机号"))
username = "13849857524"
md5_encoder = md5()
md5_encoder.update(username.encode('utf-8'))
key = md5_encoder.hexdigest()[:16].encode('utf-8')
print(key)
aes = AES.new(key=key, mode=AES.MODE_CBC, iv=b"yoloho_dayima!%_")
# s = str(input("输入密码"))
s = "123456"
s_pad = pad(s.encode("utf-8"), 16)
s_jiami = aes.encrypt(s_pad)
password = base64.b64encode(s_jiami).decode("utf-8")

# 调用rpc
rdev = frida.get_remote_device()
session = rdev.attach("大姨妈")

scr = """
rpc.exports = {
    xx:function(j2,str,j3){
        var res;
        Java.perform(function () {
           // 包.类
           var Crypt = Java.use("com.yoloho.libcore.util.Crypt");
           // 类中的方法
           res = Crypt.encrypt_data(j2,str,j3);
        });

        return res;
    }
}
"""
script = session.create_script(scr)
script.load()
sign_str = "b9bfeb09bf69b23a47ddb7cd2806cfc05b55e5bduser/login" + username + password
```

```python
print(sign_str)
# python 调用
sign = script.exports_sync.xx(0, sign_str, 85)
print(sign)
query_string = {
    "device": "b9bfeb09bf69b23a47ddb7cd2806cfc05b55e5bd",
    "ver": "630",
    "screen_width": "1080",
    "screen_height": "2240",
    "model": "23049RAD8C",
    "sdkver": "33",
    "platform": "android",
    "releasever": "13",
    "channel": "360",
    "latt": "0",
    "lngt": "0",
    "networkType": "0",
    "token": "",
    "userStatus": "0"
}
basic_url = "https://uicapi.yoloho.com/user/login?"
for key in query_string:
    basic_url += key + "=" + query_string[key] + "&"
total_url = basic_url[:-1]
data = {
    "username": "13849857524",
    "password": "u1uNlVbP2uH3/BOU8sNszw==",
    "sign": "a63dd19e35a910332da573376d376506",
    "androidid": "ba76afc0e1370865",
    "mac": "02:00:00:00:00:00",
    "imei": "",
    "density": "2.75",
    "brand": "Redmi"
}
headers = {
    "Accept-Encoding": "gzip",
    "User-Agent": "Mozilla/5.0 (Linux; Android 13; 23049RAD8C Build/TQ3C.230901.001.B1; wv)
AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/120.0.6099.43 Mobile
Safari/537.36",

    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "uicapi.yoloho.com",
    "Connection": "Keep-Alive"
}
resp = requests.post(total_url, data=data, headers=headers)
print(resp.text)
```