[原创]小红书新版shield逆向

hczhong 5

极客

9小时前 197

小红书新版shield逆向

@

目录

1、 通过Jni_load 找到函数偏移

定位到jni_load 函数，跟踪函数sub_A654。如图

```
int __fastcall sub_A654(int a1)
{
  int v1; // r4
  _DWORD *v2; // r0
  _DWORD *v3; // r0
  _DWORD *v4; // r0
  _DWORD *v5; // r0
  int result; // r0
  int v7; // [sp+4h] [bp-Ch]

  v1 = a1;
  v2 = sub_A2BC();
  sub_A47C((int)v2, v1);
  v3 = sub_A2BC();
  sub_73604((int)v3);
  v4 = sub_A2BC();
  sub_736B0((int)v4);
  v5 = sub_A2BC();
  sub_7339C((int)v5);
  sub_2E1F4();
  result = _stack_chk_guard - v7;
  if ( _stack_chk_guard == v7 )
    result = 65542;
  return result;
}
```

其中sub_2e1f4 是对app的签名进行验证，直接nop，sub_736B0 是通过jni调用java的okhttp类的一些方法。sub_7306 是动态注册的函数。

```
1  int __fastcall sub_73604(int a1)
2  {
3    int v1; // r11
4    int v2; // r0
5    JNIEnv *env; // r4
6    jclass v4; // r6
7    JNIEnv *v6; // [sp+0h] [bp-18h]
8    int v7; // [sp+4h] [bp-14h]
9    int v8; // [sp+8h] [bp-10h]
10
11   v8 = v1;
12   v2 = (*(int (**)(void))(**(_DWORD **)(a1 + 76) + 24))();
13   env = v6;
14   if ( v2 < 0 )
15     env = 0;
16   v4 = (*env)->FindClass(env, "com/xingin/shield/http/XhsHttpInterceptor");
17   if ( (*env)->RegisterNatives(env, v4, (const JNINativeMethod *)off_8E0D0, 4) >= 0 )
18     (*env)->DeleteLocalRef(env, v4);
19   return _stack_chk_guard - v7;
20 }
```

找到地址off_8E0D0，各个函数地址如图所示。

```
1:0008E0D0 off_8E0D0       DCD aInitializenati     ; DATA XREF: sub_73604+44↑o
1:0008E0D0                                         ; .ppp.ttl:off_736A8↑o
1:0008E0D0                                         ; "initializeNative"
1:0008E0D4                 DCD aV                  ; "()V"
1:0008E0D8                 DCD sub_740E4+1
1:0008E0DC                 DCD aIntercept          ; "intercept"
1:0008E0E0                 DCD aLokhttp3Interc     ; "(Lokhttp3/Interceptor$Chain;J)Lokhttp3/"...
1:0008E0E4                 DCD sub_73B78+1
1:0008E0E8                 DCD aInitialize         ; "initialize"
1:0008E0EC                 DCD aLjavaLangStrin_12  ; "(Ljava/lang/String;)J"
1:0008E0F0                 DCD sub_73960+1
1:0008E0F4                 DCD aDestroy            ; "destroy"
1:0008E0F8                 DCD aJV                 ; "(J)V"
1:0008E0FC                 DCD sub_73B24+1
```

2、分析各个函数的作用

2.1 initializeNative函数

```
40 v3 = sub_A2BC();
41 sub_A61C((int)&v16, (int)v3);
42 m_getSharedPreferences = (int)(*_env)->GetMethodID(
43                                 _env,
44                                 v16,
45                                 "getSharedPreferences",
46                                 "(Ljava/lang/String;I)Landroid/content/SharedPreferences;");
47 m_edit = (int)(*_env)->GetMethodID(_env, v17, "edit", "()Landroid/content/SharedPreferences$Editor;");
48 m_getString = (int)(*_env)->GetMethodID(
49                       _env,
50                       v17,
51                       "getString",
52                       "(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;");
53 m_putString = (int)(*_env)->GetMethodID(
54                       _env,
55                       v18,
56                       "putString",
57                       "(Ljava/lang/String;Ljava/lang/String;)Landroid/content/SharedPreferences$Editor;");
58 m_commit = (int)(*_env)->GetMethodID(_env, v18, "commit", (const char *)&dword_746D0);
59 m_request = (int)(*_env)->GetMethodID(_env, v27, "request", "()Lokhttp3/Request;");
60 m_proceed = (int)(*_env)->GetMethodID(_env, v27, "proceed", "(Lokhttp3/Request;)Lokhttp3/Response;");
61 dword_8F154 = ((int (__fastcall *)(JNIEnv *, void *, int *, const char *))(*_env)->GetMethodID)(
62                 _env,
63                 v19,
64                 &dword_746F4,
65                 "()Lokhttp3/HttpUrl;");
```

initializeNative函数是对jni调用java方法的一些类进行初始化操作，建议对变量进行改名，类用c开头，方法用*m*开头便于后续分析。

2.2 initialize 函数

```
_env = env;
_input = (void *)input;
ptr = operator new(0x2D4u);
_aeabi_memclr8(ptr, 724);
*(_BYTE *)(ptr + 650) = 1;
input_str = (*_env)->GetStringUTFChars(_env, _input, 0);
input_str_len = (*_env)->GetStringUTFLength(_env, _input);
strcpy((char *)ptr, input_str);
_ptr = ptr + 400;
v23 = input_str;
_aeabi_memcpy(ptr + 400, input_str, input_str_len);
*(_DWORD *)(_ptr + input_str_len) = 0x616D685F;// _hmac
*(_BYTE *)(_ptr + input_str_len + 4) = 0x63;  // input_str = main   main_hmac
v9 = sub_A2BC();
v10 = sub_A378((int)v9);
v11 = m_getSharedPreferences;
_s_xml = (*_env)->NewStringUTF(_env, (const char *)&s_xml_1);
v13 = CallObjectMethodV(_env, v10, v11, (int)_s_xml);
v14 = m_getString;
(*_env)->NewStringUTF(_env, (const char *)&unk_82B7A);
v24 = _input;
v15 = CallObjectMethodV(_env, (int)v13, v14, (int)_input);
v16 = m_getString;
v17 = (*_env)->NewStringUTF(_env, (const char *)(ptr + 400));
(*_env)->NewStringUTF(_env, (const char *)&unk_82B7A);
main_hac_value = CallObjectMethodV(_env, (int)v13, v16, (int)v17);
_main_hac_value = (int)main_hac_value;
if ( main_hac_value && (*_env)->GetStringUTFLength(_env, main_hac_value) )
```

initialize 函数是从s.xml文件中读取key为main_hmac的值value。

```
if ( sub_AAAC(_env, _main_hac_value, ptr) )
{
  v20 = 1;
}
else
{
  if ( !(*_env)->GetStringUTFLength(_env, v15) )
    goto LABEL_9;
  v20 = 0;
}
*(_BYTE *)(ptr + 650) = v20;
LABEL_9:
```

把读取的value使用sub_AAAC函数进行传参， sub_AAAC 函数的主要功能是对value和device_id 进行aes得到一个key，把key存入ptr + 0x28C 处， 如果sub_AAAC返回值为1，则使用新版的shield算法，反之则使用旧版的s1-s12算法。

2.3 intercept函数

intercept 是shield算法的逻辑部分，

```
if ( ptr[650] )
{
  v53 = v67;
  sub_ABB8(v67, v29, (int)ptr, v52);
}
else
{
  v53 = v67;
  sub_AD14(v67, v29, device_id, ptr);
}
```

通过ptr+650的值来判断使用哪种算法，sub_ABB8为新版，sub_AD14为旧版。

```
20  _env = env;
21  _ptr = ptr;
22  v6 = CallObjectMethodV(env, a2, clone, a4);
23  ByteArray = (char *)(*_env)->NewByteArray(_env, 4096);
24  v8 = sub_1FB28();
25  v9 = (int)v8;
26  v10 = sub_1FBB0((int)v8, *(_DWORD *)(_ptr + 652), _ptr + 656, 64);
27  v19 = 0;
28  v17 = 0;
29  v18 = 0;
30  v15 = 0;
31  byte_shield = 0;
32  if ( v10 )
33  {
34    for ( len = CallIntMethodV(_env, (int)v6, m_read, ByteArray);
35          len != -1;
36          len = CallIntMethodV(_env, (int)v6, m_read, ByteArray) )
37    {
38      byte_array = (*_env)->GetByteArrayElements(_env, ByteArray, 0);
39      sub_1FC52(v9, byte_array, len);
40      (*_env)->ReleaseByteArrayElements(_env, ByteArray, byte_array, 0);
41    }
42    ((void (__fastcall *)(JNIEnv *, char *))(*_env)->DeleteLocalRef)(_env, ByteArray);
43    sub_1FC7E(v9, (int)&byte_shield, (int)&v15);
44    sub_1FB88(v9);
45  }
```

sub_1fbb0函数对sub_AAAC 函数的key进行异或0x36和0x5c，这里大胆猜测shield使用的是hmacmd5算法，sub_1fbb0是对key进行初始化，sub_1fc52是对url进行md5，sub_1fc7e是对前面两步进行收尾工作计算出真正的shield。

```
1 int __fastcall sub_2C258(int a1)
2 {
3   return (*(int (**)(void))(a1 + 12))();
4 }
```

像这种a1+12 是一个函数指针我是通过动态调试得到函数地址。使用的是魔改的md5。

3、算法还原

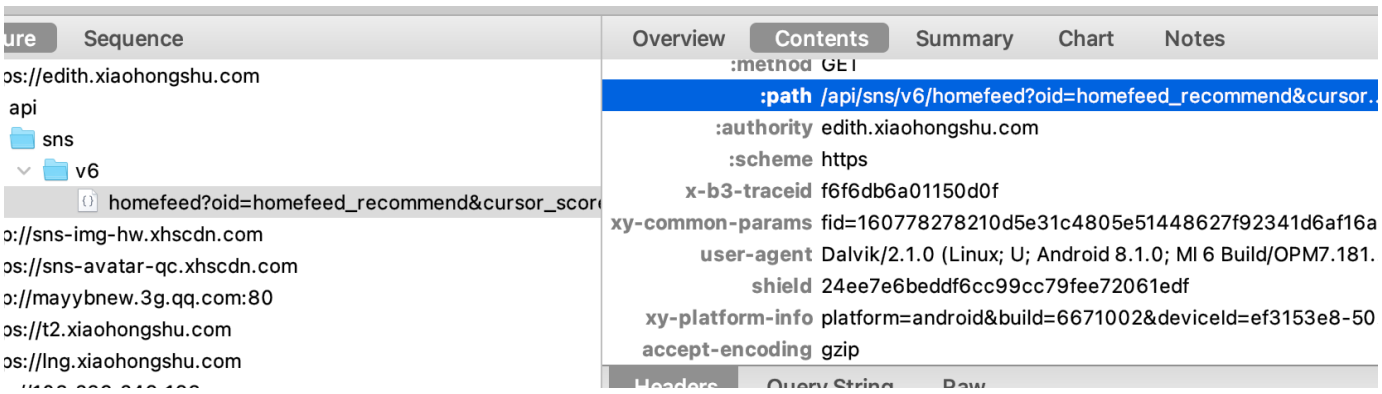
3.1 算法流程

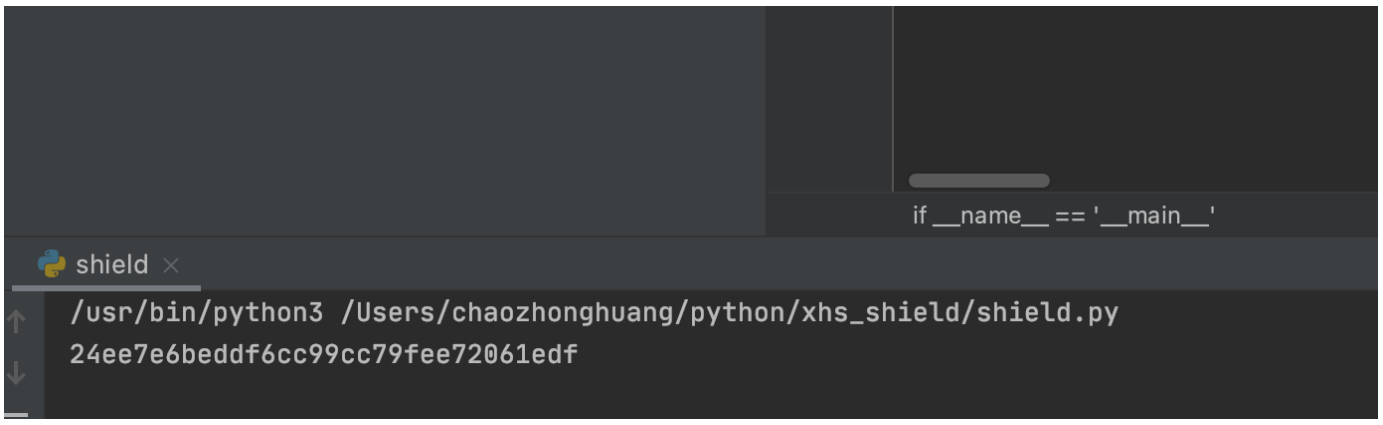
1、result = md5((key ^ 0x36) + param)

2、shield = md5((key ^ 0x5c) + result)

aes部分的算法暂时没进行还原，md5的部分使用c语言进行了还原，本来想移植到python的，但是python没有无符号等数据类型，每一次操作都要进行&0xFFFFFFFF，心态有点爆炸，所以编译了一个shield.so供python调用，最终完成图如下：

md5的参数为(urlpath+xy-common-params+xy-platform-info) (ps:urlpath 需要去掉? 比如urlpath为： /api/sns/v6/homefeed?oid=homefeed_recommend，传入的为/api/sns/v6/homefeedoid=homefeed_recommend)





2020 KCTF秋季赛【攻击篇】正在火热进行中！

收藏 · 2　点赞　打赏　分享

最新回复 (1)

Bk_Humor 8小时前　2 楼　0

极客

游客

登录 | 注册 方可回帖

回帖　表情　高级回复

返回