猿人学–frida检测与反反调试

检测案例

1. 多种特征检测 Frida

https://bbs.pediy.com/thread-217482.htm

https://github.com/muellerberndt/frida-detection

https://github.com/qtfreet00/AntiFrida

检测点

- 进程检测
- 端口检测
- dbus 通信检测
- maps 特征检测
- 内存特征检测
- svc 实现 openat/read 等
- 自实现字符串比较：memcmp

2. DetectFrida

https://github.com/kumar-rahul/detectfridalib

检测点

- 线程检测
- fd 检测
- 被 hook so 特征检测
- inline + 自实现 str 相关操作函数

3. 从 inlinehook 角度检测 frida

https://bbs.pediy.com/thread-269862.htm

检测点

- 特定 native hook 特征
- 目标 java hook 特征

4. 关于 frida 检测的一个新思路

https://bbs.pediy.com/thread-268586.htm

检测点

- maps 段 rwxp 权限特征
- 特定 native hook 特征

5. Anti Frida

https://github.com/TUGOhost/anti_Android/blob/main/app/src/main/cpp/anti_frida.c

检测点

- 针对 frida 注入特性
- 只有 attach 对抗

6. FridaCheck.apk

检测点

端口检测

```
__int64 __fastcall sub_12D0(_JNIEnv *a1)
{
  unsigned __int64 v1; // x23
  _JNIEnv *v2; // x19
  unsigned int v3; // w20
  __int64 v4; // x0
  unsigned int v5; // w21
  jclass v6; // x21
  jmethodID v7; // x22
  jstring v8; // x0
  __int64 result; // x0
  __int128 v10; // [xsp+0h] [xbp-B0h]
  __int128 v11; // [xsp+10h] [xbp-A0h]
  __int128 v12; // [xsp+20h] [xbp-90h]
  __int128 v13; // [xsp+30h] [xbp-80h]
  __int128 v14; // [xsp+40h] [xbp-70h]
  __int128 v15; // [xsp+50h] [xbp-60h]
  int v16; // [xsp+60h] [xbp-50h]
  __int16 v17; // [xsp+68h] [xbp-48h]
  __int16 v18; // [xsp+6Ah] [xbp-46h]
  char v19[12]; // [xsp+6Ch] [xbp-44h]
  __int64 v20; // [xsp+78h] [xbp-38h]

  v1 = _ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2));
  v2 = a1;
  v20 = *(_QWORD *)(v1 + 40);
  *(_QWORD *)&v19[4] = 0LL;
  v17 = 2;
  *(_QWORD *)v19 = (unsigned int)inet_addr("127.0.0.1");
  v16 = 0;
  v18 = 41577;
  v14 = 0u;
  v15 = 0u;
  v12 = 0u;
  v13 = 0u;
  v10 = 0u;
  v11 = 0u;
  v3 = 1;
  v4 = socket(2LL, 1LL, 0LL);
  v5 = v4;
  if ( !(unsigned int)connect(v4, &v17, 16LL) )
  {
    v10 = xmmword_2BF2;
    *(_QWORD *)((char *)&v10 + 13) = 49690116226672299LL;
    close(v5);
    v3 = 0;
  }
  v6 = v2->functions->FindClass(&v2->functions, "com/yimian/envcheck/CheckResult");
  v7 = v2->functions->GetMethodID(&v2->functions, v6, "<init>", "(ZLjava/lang/String;)V");
  v8 = v2->functions->NewStringUTF(&v2->functions, (const char *)&v10);
  result = _JNIEnv::NewObject(v2, v6, v7, v3, v8);
  *(_QWORD *)(v1 + 40);
  return result;
}
```

线程检测

```
121     {
122       v21 = v13;
123       v22 = 1;
124       do
125       {
126         v24 = *(unsigned __int8 *)(v21 + 19);
127         v23 = v21 + 19;
128         if ( (unsigned int)(v24 - 48) <= 9 )
129         {
130           v25 = getpid();
131           sub_2758(&v50, 200LL, 200LL, "/proc/%d/task/%s/status", v25, v23);
132           v26 = fopen(&v50, "r");
133           if ( v26 )
134           {
135             v27 = v26;
136             while ( fgets(&v50, 200LL, v27) )
137             {
138               if ( strstr(&v50, "Name:\tgmain") )
139               {
140                 v28 = "found gmain thread: %s";
141                 goto LABEL_21;
142               }
143               if ( strstr(&v50, "Name:\tgdbus") )
144               {
145                 v28 = "found gdbus thread: %s";
146                 goto LABEL_21;
147               }
148               if ( strstr(&v50, "Name:\tpool-frida") )
149               {
150                 v28 = "found pool-frida thread: %s";
151                 goto LABEL_21;
152               }
153               if ( strstr(&v50, "Name:\tgum-js-loop") )
154               {
155                 v28 = "found gum-js-loop thread: %s";
156                 goto LABEL_21;
157               }
158               if ( strstr(&v50, "SigBlk:\tfffffffe0fffbfaff") )
159               {
160                 v28 = "found exceptional sigblk thread: %s";
161 LABEL_21:
162                 sub_2758(&v33, 200LL, 200LL, v28, &v50);
163                 v22 = 0;
164                 break;
165               }
166             }
167             fclose(v27);
168           }
169           else
170           {
171             __android_log_print(6LL, "envcheck-native", "checkThreads failed to open file %s", &v50);
172           }
173         }
174         v21 = readdir(
175                 v12,
176                 v14,
177                 v15,
178                 v16,
179                 v17,
```

maps 检测

```
  v19 = 0u;
  v16 = 0u;
  v17 = 0u;
  v14 = 0u;
  v15 = 0u;
  v2 = getpid();
  sub_2758(&v27, 200LL, 200LL, "/proc/%d/maps", v2);
  v3 = fopen(&v27, "r");
  if ( v3 )
  {
    v4 = v3;
    v5 = "found maps record with agent: %s";
    while ( fgets(&v27, 200LL, v4) )
    {
      if ( strstr(&v27, "agent") )
        goto LABEL_7;
      if ( strstr(&v27, "/data/local/tmp") )
      {
        v5 = "found maps record with /data/local/tmp path: %s";
LABEL_7:
        sub_2758(&v14, 200LL, 200LL, v5, &v27);
        fclose(v4);
LABEL_8:
        v6 = 0;
        goto LABEL_12;
      }
    }
    fclose(v4);
    sub_2758(&v27, 200LL, 200LL, "/data/local/tmp/re.frida.server/frida-agent-32.so");
    if ( !(unsigned int)stat(&v27, &v13) )
    {
      sub_2758(&v14, 200LL, 200LL, (const char *)&unk_2E24);
      goto LABEL_8;
    }
    v6 = 1;
LABEL_12:
    v10 = v1->functions->FindClass(&v1->functions, "com/yimian/envcheck/CheckResult");
    v11 = v1->functions->GetMethodID(&v1->functions, v10, "<init>", "(ZLjava/lang/String;)V");
    v12 = v1->functions->NewStringUTF(&v1->functions, (const char *)&v14);
    result = _JNIEnv::NewObject(v1, v10, v11, v6, v12);
  }
  else
  {
    v7 = (unsigned int *)__errno();
    v8 = strerror(*v7);
    __android_log_print(6LL, "envcheck-native", "failed to open maps, error: %s", v8);
    result = 0LL;
  }
  return result;
}
```


fd 检测

```
          v22 = getpid();
          sub_2758(&v62, 200LL, 200LL, "/proc/%d/fd/%s", v22, v14 + 19);
          v23 = readlink(&v62, &v49, 199LL);
          if ( v23 & 0x8000000000000000LL )
          {
            v29 = (unsigned int *)__errno();
            v27 = (__int128 *)strerror(*v29);
            v26 = "checkFd failed to readlink, error: %s";
            goto LABEL_15;
          }
          *((_BYTE *)&v49 + v23) = 0;
          if ( strstr(&v49, "linjector") )
          {
            v24 = "found fd path with linjector: %s";
LABEL_19:
            sub_2758(&v49, 200LL, 200LL, v24, v14 + 19);
            closedir(v12);
LABEL_20:
            v25 = 0;
            goto LABEL_21;
          }
          if ( strstr(&v49, "/data/local/tmp") )
          {
            v24 = "found fd path with /data/local/tmp: %s";
            goto LABEL_19;
          }
        }
        v14 = readdir(
                v12,
                v15,
                v16,
                v17,
                v18,
                v19,
                v20,
                v21,
                v33,
                v34,
                v35,
                v36,
                v37,
                v38,
                v39,
                v40,
                v41,
                v42,
                v43,
                v44,
                v45,
                v46,
                v47,
                v48,
                v49);
      }
      while ( v14 );
    }
    closedir(v12);
    sub_2758(&v62, 200LL, 200LL, "/data/local/tmp/re.frida.server");
    if ( !(unsigned int)stat(&v62, &v33) )
    {
      sub_2758(&v49, 200LL, 200LL, "found /data/local/tmp/re.frida.server dir");
      goto LABEL_20;
    }
    v25 = 1;
```

内存特征检测

```
  v43 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
  v2 = fopen("/proc/self/maps", "r");
  if ( v2 )
  {
    v3 = v2;
    while ( 1 )
    {
      do
      {
LABEL_3:
        if ( !fgets(&v42, 512LL, v3) )
        {
          fclose(v3);
          v14 = v1->functions->FindClass(&v1->functions, "com/yimian/envcheck/Che
          v15 = v1->functions->GetMethodID(&v1->functions, v14, "<init>", "(ZLjav
          v16 = v1->functions->NewStringUTF(&v1->functions, "");
          v17 = 1LL;
          v18 = v1;
          v19 = (__int64)v14;
          v20 = (__int64)v15;
          goto LABEL_90;
        }
      }
      while ( (unsigned int)sscanf() == '\x03'
            || v29 == 's'
            || v28 != 'r'
            || v41 != '/'
            || !(unsigned int)memcmp(&v41, "/dev/", '\x05') );
      v4 = v27;
      v39 = 810260286956381450 2LL;
      v40 = 99;
      v5 = v26 - 10;
      if ( v26 - 10 > v27 )
        break;
LABEL_23:
      v38 = 101;
      v37 = xmmword_2A10;
      if ( v26 - 18 > v27 )
      {
        v7 = (_BYTE *)v27;
        while ( *v7 != 'F'
              || v7[1] != 'r'
              || v7[2] != 'i'
              || v7[3] != 'd'
              || v7[4] != 'a'
              || v7[5] != 'S'
              || v7[6] != 'c'
              || v7[7] != 'r'
              || v7[8] != 'i'
              || v7[9] != 'p'
              || v7[10] != 't'
              || v7[11] != 'E'
              || v7[12] != 'n'
              || v7[13] != 'g'
              || v7[14] != 'i'
              || v7[15] != 'n'
              || v7[16] != 'e'
              || v7[17] )
        {
          if ( (_BYTE *)(v26 - 18) == ++v7 )
            goto LABEL_45;
```

```
  if ( v26 - 9 > v27 )
  {
    v8 = (_BYTE *)v27;
    while ( *v8 != 'G'
         || v8[1] != 'L'
         || v8[2] != 'i'
         || v8[3] != 'b'
         || v8[4] != '-'
         || v8[5] != 'G'
         || v8[6] != 'I'
         || v8[7] != 'O'
         || v8[8] )
    {
      if ( (_BYTE *)(v26 - 9) == ++v8 )
        goto LABEL_58;
    }
    __android_log_print(5LL, "envcheck-native", "found %s in memory at addr
    v9 = &v35;
    goto LABEL_87;
  }
LABEL_58:
  v32 = 803056954237375392 7LL;
  v33 = 31096;
  v34 = 0;
  if ( v26 - 11 > v27 )
  {
    v10 = (_BYTE *)v27;
    while ( *v10 != 'G'
         || v10[1] != 'D'
         || v10[2] != 'B'
         || v10[3] != 'u'
         || v10[4] != 's'
         || v10[5] != 'P'
         || v10[6] != 'r'
         || v10[7] != 'o'
         || v10[8] != 'x'
         || v10[9] != 'y'
         || v10[10] )
    {
      if ( (_BYTE *)(v26 - 11) == ++v10 )
        goto LABEL_73;
    }
    __android_log_print(5LL, "envcheck-native", "found %s in memory at addr
    v9 = &v32;
    goto LABEL_87;
  }
LABEL_73:
  v30 = 810013117572955885 5LL;
  v31 = 116;
  if ( v5 > v27 )
  {
    while ( *(_BYTE *)v4 != 'G'
         || *(_BYTE *)(v4 + 1) != 'u'
         || *(_BYTE *)(v4 + 2) != 'm'
         || *(_BYTE *)(v4 + 3) != 'S'
         || *(_BYTE *)(v4 + 4) != 'c'
         || *(_BYTE *)(v4 + 5) != 'r'
         || *(_BYTE *)(v4 + 6) != 'i'
         || *(_BYTE *)(v4 + 7) != 'p'
         || *(_BYTE *)(v4 + 8) != 't'
         || *(_BYTE *)(v4 + 9) )
    {
      if ( v5 == ++v4 )
        goto LABEL_3;
```

```
          goto LABEL_3;
        }
        __android_log_print(5LL, "envcheck-native", "found %s in memory at address %llx");
        v9 = &v30;
LABEL_87:
        sub_2758(&v42, 512LL, 512LL, "found %s in memory", v9);
        v11 = v1->functions->FindClass(&v1->functions, "com/yimian/envcheck/CheckResult");
        v12 = v1->functions->GetMethodID(&v1->functions, v11, "<init>", "(ZLjava/lang/String;)V");
        v13 = v1->functions->NewStringUTF(&v1->functions, &v42);
        _JNIEnv::NewObject(v1, v11, v12, 0LL, v13);
        return;
      }
    }
    v6 = (_BYTE *)v27;
    while ( *v6 != 'f'
         || v6[1] != 'r'
         || v6[2] != 'i'
         || v6[3] != 'd'
         || v6[4] != 'a'
         || v6[5] != ':'
         || v6[6] != 'r'
         || v6[7] != 'p'
         || v6[8] != 'c'
         || v6[9] )
    {
      if ( (_BYTE *)v5 == ++v6 )
        goto LABEL_23;
    }
    __android_log_print(5LL, "envcheck-native", "found %s in memory at address %llx", v6);
    v23 = &v39;
LABEL_93:
    sub_2758(&v42, 512LL, 512LL, "found %s in memory", v23);
    v24 = ((__int64 (__fastcall *)(_JNIEnv *, const char *))v1->functions->FindClass)(
            v1,
            "com/yimian/envcheck/CheckResult");
    v25 = ((__int64 (__fastcall *)(_JNIEnv *, __int64, const char *, const char *))v1->functions->GetMethodII
            v1,
            v24,
            "<init>",
            "(ZLjava/lang/String;)V");
    v16 = (jstring)((__int64 (__fastcall *)(_JNIEnv *, char *))v1->functions->NewStringUTF)(v1, &v42);
    v18 = v1;
    v19 = v24;
    v20 = v25;
    v17 = 0LL;
LABEL_90:
    _JNIEnv::NewObject(v18, v19, v20, v17, v16);
  }
  else
  {
    v21 = (unsigned int *)__errno();
    v22 = strerror(*v21);
    __android_log_print(6LL, "envcheck-native", "checkMemory failed to open maps", v22);
```


检测总结

只针对上文检测案例

- 进程检测（自编译 frida 解决）
- 端口检测（自定义端口解决）
- dbus 通信检测
- maps 特征检测（自编译 frida 解决一部分）
  - maps 段 rwxp 权限特征
- 遍历 so 内存字符串特征检测（字符串 patch 解决一部分，自定义内核过滤 maps 解决大部分，但 so 链表也能遍历到每个加载的 so）
- 线程检测（自编译 frida 解决一部分）
- fd 检测（自编译 frida 解决）
- svc 实现 openat/read 等（参考看雪 frida-seccomp）
- 针对 frida 注入特性（不处理，spawn > attach ）
  - 只有 attach 对抗
- hook 特征（暂不处理）
  - 特定 native hook 特征
  - 目标 java hook 特征

    - 被 hook so 特征检测
- inline + 自实现 str 相关操作函数（暂不处理）

反反调试实现

```
编译可参考：
fridaserver去特征检测以及编译
https://bbs.pediy.com/thread-272536.htm
(patch逻辑有lief操作，记得pip3 install lief，然后一遍过)

frida-20220715
1，https://github.com/AAAA-Project/Patchs.git
2，去除uuid特征
3，去frida-helper特征

frida-20220716
1，去除线程特征 + pool-frida(没效果) (gdbus改了报错(霜哥正常)，Unable to get frontmost application on localhost:6666: process w
2，去tmp特征
3，去maps权限特征
4，memstring特征（FridaScriptEngine，GLib-GIO，GDBusProxy，GumScript）（这里永远改不全，要想彻底bypass搭配自定义内核隐藏maps食用更佳

自编译得frida会发到猿人学网盘，后续会持续更新ing，公众号[妄为写代码]回复[frida]获取(自愿)

日常操作：
1，深入hook strcmp，strstr，pthread_create
2，  自定义端口

内核日志:
adb logcat -b kernel |grep HEXL

日常操作带上，实在不行只能去分析patch了
```