

Android NDK 程式 (.so 檔) 逆向與防逆向

Jason Wang | 王羿廷

© Fourdesire. All Rights Reserved.

Outline

- Android NDK 程式簡介
- Hello World NDK
- 基本逆向分析與反逆向措施
- 進階逆向分析與反逆向措施
- 總結

Android NDK 程式簡介

Android NDK

- NDK = **Native Development Kit**, 原生開發套件
- 在 Android 平台進行 **C** 和 **C++** 程式開發的工具
- 編譯後會產出副檔名為 **so** 的檔案, 如 libndk.so
- 無法單獨執行, 需透過 Java 程式碼 `System.loadLibrary("ndk");` 載入
- 執行速度快, 常用於需大量運算的功能, 如 3D 繪圖、美肌、修圖等
- 逆向難度較高, 需熟悉基本 ARM 或 X86 組合語言指令集

Hello World NDK

Hello World NDK

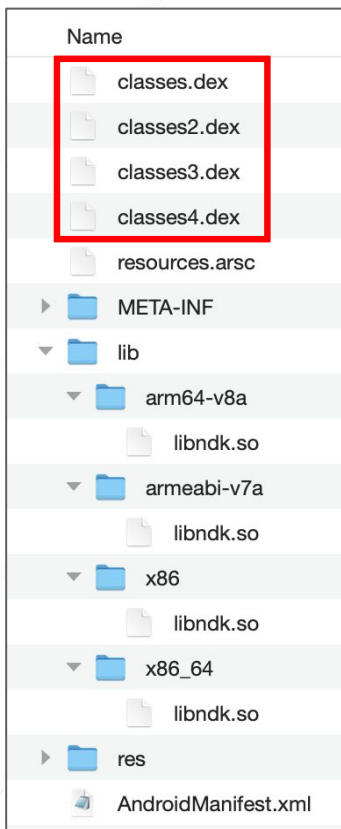
```
MainActivity.java x
8 public class MainActivity extends AppCompatActivity
9 {
10     public native String stringFromJNI();
11     static
12     {
13         System.loadLibrary("ndk");
14     }
15     @Override
16     protected void onCreate(Bundle savedInstanceState)
17     {
18         super.onCreate(savedInstanceState);
19         com.fourdesire.ndk.databinding.ActivityMainBinding
20         setContentView(binding.getRoot());
21
22         TextView tv = binding.sampleText;
23         tv.setText(stringFromJNI());
24     }
25 }

native-lib.cpp x
4 extern "C" JNIEXPORT jstring JNICALL
5 Java_com_fourdesire_ndk_MainActivity_stringFromJNI(
6     JNIEnv *env,
7     jobject MainActivity /* this */
8 )
9 {
10
11     std::string hello = "Hello from C++";
12     return env->NewStringUTF(hello.c_str());
13 }
```



基本逆向分析與反逆向措施

逆向分析 - Hello World Java 層

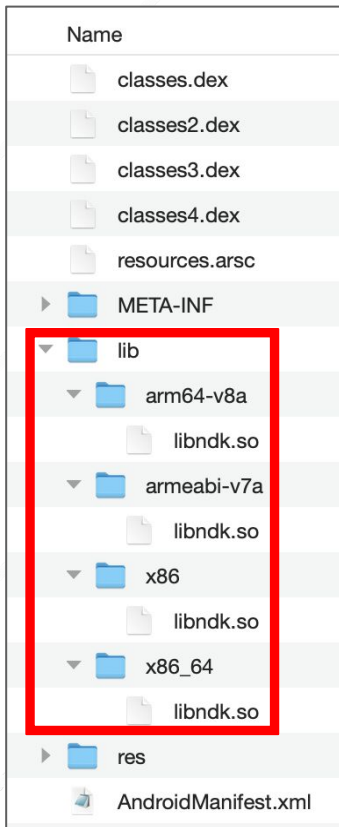


```
11 public class MainActivity extends AppCompatActivity {  
    public native String stringFromJNI();  
  
    static {  
13     System.loadLibrary("ndk");  
    }  
  
    /* access modifiers changed from: protected */  
    @Override // android.support.v7.app.AppCompatActivity  
17     public void onCreate(Bundle savedInstanceState) {  
18         super.onCreate(savedInstanceState);  
19         ActivityMainBinding binding = ActivityMainBinding  
20             setContentView(binding.getRoot());  
23         binding.sampleText.setText(stringFromJNI());  
    }  
}
```



<https://github.com/skylot/jadx>

逆向分析 - Hello World Native 層



```
1 __int64 __fastcall Java_com_fourdesire_ndk_MainActivity_stringFromJNI(_JNIEnv *a1)
2 {
3     __int64 v1; // ST30_8
4     __int64 result; // x0
5     __int64 v3; // [xsp+10h] [xbp-120h]
6     const char *v4; // [xsp+18h] [xbp-118h]
7     JNIEnv *v5; // [xsp+80h] [xbp-B0h]
8     const char *v6; // [xsp+F0h] [xbp-40h]
9     const char *v7; // [xsp+F8h] [xbp-38h]
10    char v8; // [xsp+100h] [xbp-30h]
11    __int64 v9; // [xsp+118h] [xbp-18h]
12
13    v9 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
14    v5 = a1;
15    std::_ndk1::__compressed_pair_elem<std::_ndk1::basic_string<char,std::_ndk1::char_traits<char>,std::_ndk1::allocator<char>>,1,true>::__compressed_pair_elem<std::_ndk1::basic_string<char,std::_ndk1::char_traits<char>,std::_ndk1::allocator<char>>,1,true>::__compressed_pair_elem($v8);
16    v1 = std::_ndk1::char_traits<char>::length("Hello from C++");
17    std::_ndk1::basic_string<char,std::_ndk1::char_traits<char>,std::_ndk1::allocator<char>>::__init(
18        $v8,
19        "Hello from C++",
20        v1);
21    if ( *(_BYTE *)std::_ndk1::__compressed_pair_elem<std::_ndk1::basic_string<char,std::_ndk1::char_traits<char>,std::_ndk1::allocator<char>>,1,true>::__compressed_pair_elem($v8);
22    {
23        v4 = *(const char **) (std::_ndk1::__compressed_pair_elem<std::_ndk1::basic_string<char,std::_ndk1::char_traits<char>,std::_ndk1::allocator<char>>,1,true>::__compressed_pair_elem($v8);
24        + 16);
25    }
26    else
27    {
28        v6 = (const char *) (std::_ndk1::__compressed_pair_elem<std::_ndk1::basic_string<char,std::_ndk1::char_traits<char>,std::_ndk1::allocator<char>>,1,true>::__compressed_pair_elem($v8);
29        + 1);
30        v4 = v6;
31    }
32    v7 = v4;
33    v3 = JNIEnv::NewStringUTF(v5, v4);
34    std::_ndk1::basic_string<char,std::_ndk1::char_traits<char>,std::_ndk1::allocator<char>>::__basic_string($v8);
35    result = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
36    if ( result == v9 )
37        result = v3;
38    return result;
39 }
40 }
```



<https://hex-rays.com/ida-pro/>

逆向分析 - 總結

- Java 層反編譯非常容易，原始碼還原度高
- Native 層反編譯後原始碼還原度低，但仍有弱點：
 - **函數名稱未隱藏**：可透過函數名稱猜出函數功能
 - **字串內容明碼顯示**：可透過字串內容猜出程式功能

反逆向措施 1 - 隱藏函數名稱

```
extern "C" JNIEXPORT jstring JNICALL
Java_com_fourdesire_ndk_MainActivity_stringFromJNI(
    JNIEnv *env,
    jobject MainActivity
)
{
    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```



```
static
jstring nativeStringFromJNI(
    JNIEnv *env,
    jobject
)
{
    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

```
JNIEXPORT jint JNI_OnLoad(JavaVM *vm, void *reserved)
{
    JNIEnv *env;
    if (vm->GetEnv(reinterpret_cast<void **>(&env), JNI_VERSION_1_6) != JNI_OK) return JNI_ERR;

    jclass c = env->FindClass( name: "com/fourdesire/ndk/MainActivity");
    if (c == nullptr) return JNI_ERR;

    static const JNINativeMethod methods[] = {
        { .name: "stringFromJNI", .signature: "()Ljava/lang/String;", (void *) nativeStringFromJNI };
    };
    int rc = env->RegisterNatives(c, methods, nMethods: sizeof(methods) / sizeof(JNINativeMethod));
    if (rc != JNI_OK) return rc;

    return JNI_VERSION_1_6;
}
```

1. 函數宣告使用 **static** 關鍵字
2. 新增 **JNI_OnLoad** 函數
3. 在 **JNI_OnLoad** 函數內使用 **RegisterNatives** 動態註冊函數

參考資料

: <https://developer.android.com/training/articles/perf-jni#native-libraries>

反逆向措施 1 - 隱藏函數名稱效果

```
static
jstring nativeStringFromJNI(
    JNIEnv *env,
    jobject
)
{
    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```



```
int64 __fastcall sub_92CC(JNIEnv *a1)
{
    __int64 v1; // ST30_8
    __int64 result; // x0
    __int64 v3; // [xsp+10h] [xbp-120h]
    const char *v4; // [xsp+18h] [xbp-118h]
    JNIEnv *v5; // [xsp+80h] [xbp-B0h]
    const char *v6; // [xsp+F0h] [xbp-40h]
    const char *v7; // [xsp+F8h] [xbp-38h]
    char v8; // [xsp+100h] [xbp-30h]
    __int64 v9; // [xsp+118h] [xbp-18h]

    v9 = *(_QWORD *) (_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
    v5 = a1;
    std::_ndk1::_compressed_pair_elem<std::_ndk1::basic_string<char,std::_ndk1::char_traits<char>,std::allocator<char>,1,true>::__compressed_pair_elem(&v8);
    v1 = std::_ndk1::char_traits<char>::length("Hello from C++");
```

- 函數名稱 nativeStringFromJNI → 檔案位址 sub_92CC

反逆向措施 2 - 字串加密

1. 使用 C++ 巨集加密字串: <https://github.com/adamyaxley/Obfuscate>
2. 複製 obfuscate.h 到原始碼專案內, `#include "obfuscate.h"` 宣告該標頭檔
3. 替換要加密的字串: "My String" → `AY_OBFUSCATE("My String")`

```
static
jstring nativeStringFromJNI(
    JNIEnv *env,
    jobject
)
{
    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}

JNIEXPORT jint JNI_OnLoad(JavaVM *vm, void *reserved)
{
    JNIEnv *env;
    if (vm->GetEnv(reinterpret_cast<void **>(&env), JNI_VERSION_1_6) != JNI_OK) return JNI_ERR;

    jclass c = env->FindClass( name: "com/fourdesire/ndk/MainActivity");
    if (c == nullptr) return JNI_ERR;

    static const JNINativeMethod methods[] = {
        { name: "stringFromJNI", signature: "()Ljava/lang/String;", (void *) nativeStringFromJNI};
    };
    int rc = env->RegisterNatives(c, methods, nMethods: sizeof(methods) / sizeof(JNINativeMethod));
    if (rc != JNI_OK) return rc;

    return JNI_VERSION_1_6;
}
```



```
static
jstring nativeStringFromJNI(
    JNIEnv *env,
    jobject
)
{
    return env->NewStringUTF( bytes: AY_OBFUSCATE( data: "Hello from C++"));
}

JNIEXPORT jint JNI_OnLoad(JavaVM *vm, void *reserved)
{
    JNIEnv *env;
    if (vm->GetEnv(reinterpret_cast<void **>(&env), JNI_VERSION_1_6) != JNI_OK) return JNI_ERR;

    jclass c = env->FindClass( name: AY_OBFUSCATE( data: "com/fourdesire/ndk/MainActivity");
    if (c == nullptr) return JNI_ERR;

    static const JNINativeMethod methods[] = {
        { name: AY_OBFUSCATE( data: "stringFromJNI"), signature: AY_OBFUSCATE( data: "()Ljava/lang/String;");
    };
    int rc = env->RegisterNatives(c, methods, nMethods: sizeof(methods) / sizeof(JNINativeMethod));
    if (rc != JNI_OK) return rc;

    return JNI_VERSION_1_6;
}
```

反逆向措施 2 - 字串加密效果

```
static
jstring nativeStringFromJNI(
    JNIEnv *env,
    jobject
)
{
    return env->NewStringUTF( bytes: AY_OBFUSCATE( data: "Hello from C++" );
};
```

- 明碼字串已不再出現
- 所有字串皆使用 XOR 運算加密
- 每個字串使用的加密金鑰皆不同
- 逆向難度大幅上升

```
v1 = _ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2));
v11 = *(_QWORD *) (v1 + 40);
v2 = a1;
v3 = v1;
v4 = sub_89C8(&v10);
v8 = (const char *)ay::obfuscated_data<15ull,4171390554486297459ull>::operator char *(v4, v5, v6, v7);
result = JNIEnv::NewStringUTF(v2, v8);
*(_QWORD *) (v3 + 40);
return result;
```

```
ay * __fastcall ay::obfuscated_data<15ull,4171390554486297459ull>::decrypt(ay *result,
{
    ay *v4; // [xsp+8h] [xbp-18h]

    v4 = result;
    if ( *((_BYTE *)result + 15) & 1 )
    {
        result = (ay *)ay::cipher(result, &byte_9[6], 0x39E3C19BBB935373uLL);
        *((_BYTE *)v4 + 15) = 0;
    }
    return result;
}
```

```
int64 __fastcall ay::cipher(__int64 result, char *a2, unsigned __int64 a3)
{
    unsigned __int64 i; // [xsp+0h] [xbp-20h]

    for ( i = 0LL; i < (unsigned __int64)a2; ++i )
        *((_BYTE *) (result + i) ^= a3 >> 8 * (unsigned __int8) (i % 8));
    return result;
}
```


反逆向措施 - 總結

- 函數名稱未隱藏
 - 將函數宣告為 static 類型
 - 使用 RegisterNatives 動態註冊 JNI 函數
- 字串內容明碼顯示
 - 使用 C++ 巨集加密字串內容

進階逆向分析與反逆向措施

逆向神器 - Unicorn (unicorn-engine.org)

- 輕量化、多架構的 CPU 模擬器引擎
- 可模擬 ARM、ARM64、x86、x86_64 等指令集 CPU
- 已有許多使用該引擎的逆向工具，主要用於各平台 Binary 的逆向分析
- Android NDK 程式逆向分析工具，可模擬 ARM CPU 執行 so 檔的函數
 - unidbg (<https://github.com/zhkl0228/unidbg>)
 - AndroidNativeEmu (<https://github.com/AeonLucid/AndroidNativeEmu>)

OWASP - MSTG Crackmes

- 用於練習逆向的 App
- 由 OWASP 旗下的 MSTG (Mobile Security Testing Guide) 所提供
- Android 平台難度由簡單到困難分為 Level 1 ~ 4
- <https://github.com/OWASP/owasp-mstg/tree/master/Crackmes>

UnCrackable Mobile Apps



Welcome to the UnCrackable Apps for Android and iOS, a collection of mobile reverse engineering challenges. These challenges are used as examples throughout the Mobile Security Testing Guide. Of course, you can also solve them for fun.

Android

UnCrackable App for Android Level 2

- 目標: 找出隱藏在 App 裡的密碼字串
- 提示: 此 App 有 Root 及 Debugger 偵測機制

UnCrackable App for Android Level 2

This app holds a secret inside. May include traces of native code.

- Objective: A secret string is hidden somewhere in this app. Find a way to extract it.
- Author: [Bernhard Mueller](#).
- Special thanks to Michael Helwig for finding and fixing an oversight in the anti-tampering mechanism.
- Maintained by the OWASP MSTG leaders.

Installation

This app is compatible with Android 4.4 and up.

```
$ adb install UnCrackable-Level2.apk
```

Solutions

- [Solution using Frida and radare2](#) by c0dmtr1x.
- [Solution using Frida](#) by Eduardo Novella.
- [Solution using patches](#) by sh3llc0d3r.
- [Solution using RMS](#) by @mobilesecurity_ (video).
- [Solution using static analysis and Ghidra](#) by Eduardo Vasconcelos.
- [Solution using Ghidra and Frida](#) by Davide Cioccia



逆向分析 - Java 層

```
40 private native void init();
41
42 /* access modifiers changed from: protected */
43 @Override // android.support.v4.app.h, android.support.v4.app.z, android.support.v4.app.a
44 public void onCreate(Bundle bundle) {
45     init();
46     if (b.a() || b.b() || b.c()) {
47         a("Root detected!");
48     }
49     if (a.a(getApplicationContext()) {
50         a("App is debuggable!");
51     }
52     new AsyncTask<Void, String, String>() {
53         /* class sg.vantagepoint.uncrackable2.MainActivity.AnonymousClass1 */
54
55         /* access modifiers changed from: protected */
56         /* renamed from: a */
57         public String doInBackground(Void... voidArr) {
58             while (!Debug.isDebuggerConnected()) {
59                 SystemClock.sleep(100);
60             }
61             return null;
62         }
63     }
64
65     /* access modifiers changed from: protected */
66     /* renamed from: a */
67     public void onPostExecute(String str) {
68         MainActivity.this.a((MainActivity) "Debugger detected!");
69     }
70     }.execute(null, null, null);
71     this.m = new CodeCheck();
72     super.onCreate(bundle);
73     setContentView(R.layout.activity_main);
74 }
```

App開啟: OnCreate() → init()

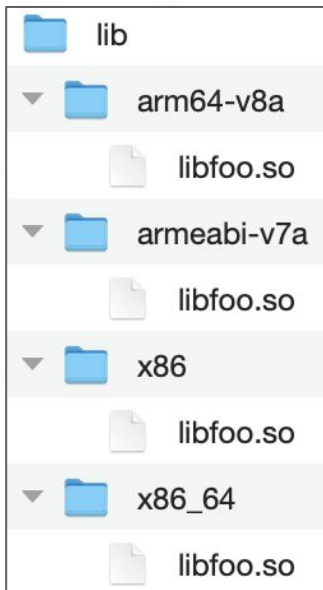
```
<?xml version="1.0" encoding="utf-8"?>
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:orientation="vertical" android:layout_width="match_parent" android:layout_height="match_parent">
2     <LinearLayout android:orientation="horizontal" android:layout_width="match_parent" android:layout_height="wrap_content">
3         <EditText android:id="@id/edit_text" android:layout_width="0dp" android:layout_height="wrap_content" android:hint="@string/edit_text" android:layout_weight="1" android:inputType="text" />
4         <Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="@string/button_verify" android:onClick="verify" />
5     </LinearLayout>
6     <LinearLayout>
7         <RelativeLayout android:padding="10dp" android:layout_width="match_parent" android:layout_height="match_parent">
8             <TextView android:layout_width="match_parent" android:layout_height="wrap_content" android:text="@string/thanks" android:layout_alignParentBottom="true" />
9         </RelativeLayout>
10 </LinearLayout>
```

```
75 public void verify(View view) {
76     String str;
77     String obj = ((EditText) findViewById(R.id.edit_text)).getText().toString();
78     AlertDialog create = new AlertDialog.Builder(this).create();
79     if (this.m.a(obj)) {
80         create.setTitle("Success!");
81         str = "This is the correct secret.";
82     } else {
83         create.setTitle("Nope..");
84         str = "That's not it. Try again.";
85     }
86     create.setMessage(str);
87     create.setButton(-3, "OK", new DialogInterface.OnClickListener() {
88         /* class sg.vantagepoint.uncrackable2.MainActivity.AnonymousClass3 */
89
90         public void onClick(DialogInterface dialogInterface, int i) {
91             dialogInterface.dismiss();
92         }
93     });
94     create.show();
95 }
```

```
3 public class CodeCheck {
4     private native boolean bar(byte[] bArr);
5
6     public boolean a(String str) {
7         return bar(str.getBytes());
8     }
9 }
```

按下確認按鈕: verify(View) → a(String) → bar(byte[])

逆向分析 - Native 層



- 字串長度為 23
- strcmp() 的第二參數
即為答案

	Java_sg_vantagepoint_uncrackable2_MainActivity_init	.text
	Java_sg_vantagepoint_uncrackable2_CodeCheck_bar	.text

```
1 bool __fastcall Java_sg_vantagepoint_uncrackable2_CodeCheck_bar(JNIEnv *a1, __int64 a2, __int64 a3)
2 {
3     __int64 v3; // x19
4     JNIEnv *v4; // x20
5     _BOOL8 result; // x0
6     __int64 v6; // x21
7     __int128 v7; // [xsp+0h] [xbp-40h]
8     __int64 v8; // [xsp+10h] [xbp-30h]
9     __int64 v9; // [xsp+18h] [xbp-28h]
10
11     v3 = a3;
12     v4 = a1;
13     result = 0LL;
14     v9 = *(QWORD *) (ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
15     if (byte 1300C == 1)
16     {
17         v8 = 0LL;
18         v7 = xmmword_EA0;
19         LODWORD(v8) = 1713399144;
20         WORD2(v8) = 29545;
21         BYTE6(v8) = 104;
22         v6 = ((int64 (__fastcall *) (JNIEnv *, int64, QWORD))(*v4)->GetByteArrayElements)(v4, a3, 0LL);
23         result = ((unsigned int (__fastcall *) (JNIEnv *, __int64))(*v4)->GetArrayLength)(v4, v3) == 23
24             && !(unsigned int)strcmp(v6, &v7, 23LL);
25     }
26     return result;
27 }
```

```
1 __int64 Java_sg_vantagepoint_uncrackable2_MainActivity_init()
2 {
3     __int64 result; // x0
4
5     result = sub_918();
6     byte 1300C = 1;
7     return result;
8 }
```


模擬執行 - unidbg

```
emulator.attach().addBreakPoint(module, offset: 0x820, (emulator, address) → {
    RegisterContext context = emulator.getContext();
    String s1 = context.getPointerArg( index: 0).getString( offset: 0);
    String s2 = context.getPointerArg( index: 1).getString( offset: 0);
    int n = context.getIntArg( index: 2);
    System.out.printf("UnCrackable2: strcmp() before, s1=\"%s\", s2=\"%s\", n=%d\n", s1, s2, n);
    emulator.attach().addBreakPoint(context.getLRPointer().peer, (emulator, address) → {
        RegisterContext context = emulator.getContext();
        System.out.printf("UnCrackable2: strcmp() after, return=0x%08x\n", context.getIntArg( index: 0));
        return true;
    });
    return true;
});
```

Hook strcmp()

```
Pointer jniEnv = vm.getJNIEnv();
DvmObject<?> thiz = vm.resolveClass( className: "owasp.mstg.uncrackable2.MainActivity").newObject( value: null);
List<Object> args = new ArrayList<>();
args.add(jniEnv);
args.add(vm.addLocalObject(thiz));
module.callFunction(emulator, symbolName: "Java_sg_vantagepoint_uncrackable2_MainActivity_init", args.toArray());
```

模擬呼叫 init()

```
System.out.println("UnCrackable2: bar() before");
thiz = vm.resolveClass( className: "owasp.mstg.uncrackable2.CodeCheck").newObject( value: null);
args = new ArrayList<>();
args.add(jniEnv);
args.add(vm.addLocalObject(thiz));
String inputSecretString = "12345678901234567890123";
ByteArray byteArray = new ByteArray(vm, inputSecretString.getBytes());
args.add(vm.addLocalObject(byteArray));
int result = module.callFunction(emulator, symbolName: "Java_sg_vantagepoint_uncrackable2_CodeCheck_bar", args.toArray())[0].intValue();
System.out.println("UnCrackable2: bar() after, return=" + result);
```

傳入長度為 23 的任意字串，模擬呼叫 bar(byte[])

```
UnCrackable2: bar() before
JNIEnv->GetArrayLength([B@3ada9e37 => 23) was called from RX@0x40000e48[libfoo.so]0xe48
UnCrackable2: strcmp() before, s1="12345678901234567890123", s2="Thanks for all the fish", n=23
UnCrackable2: strcmp() after, return=0xffffffffba
UnCrackable2: bar() after, return=0
```

查看模擬結果，strcmp 參數 2 即為正確密碼

模擬執行 - AndroidNativeEmu

```
strncmp_symbol = 0
for module in emulator.modules:
    if 'libc.so' in module.filename:
        strncmp_symbol = module.find_symbol('strncmp')

try:
    function_hooker = FuncHooker(emulator)
    function_hooker.fun_hook(strncmp_symbol, 3, strncmp_before, strncmp_after)
```

Hook strncmp()

```
emulator.call_symbol(lib_module, 'Java_sg_vantagepoint_uncrackable2_MainActivity_init', emulator.java_vm.jni_env.address_ptr, 0x00)
```

模擬呼叫 init()

```
print("UnCrackable2: bar() before")
input_secret_string = b'12345678901234567890123'
java_byte_array = Array(bytearray(input_secret_string))
result = emulator.call_symbol(lib_module, 'Java_sg_vantagepoint_uncrackable2_CodeCheck_bar', emulator.java_vm.jni_env.address_ptr, 0x00, java_byte_array)
print("UnCrackable2: bar() after, return=%d\n" % result)
```

傳入長度為 23 的
任意字串，模擬呼
叫 bar(byte[])

```
except UcError as e:
    print("UnCrackable2: Exit at %x" % emulator.mu.reg_read(UC_ARM_REG_PC))
    raise
```

```
UnCrackable2: bar() before
```

```
DEBUG:androidemu.java.jni_env:JNIEnv->GetByteArrayElements(bytearray(b'12345678901234567890123')(1), 0) was called
```

```
DEBUG:androidemu.java.jni_env:JNIEnv->GetArrayLength(1) was called, return value = 23
```

```
DEBUG:root:trigger hook on 0xCBC45B1C
```

```
UnCrackable2: strncmp() before, s1="12345678901234567890123", s2="Thanks for all the fish", n=23
```

```
UnCrackable2: strncmp() after, return=0xFFFFFFFFFFFFFFD0
```

```
UnCrackable2: bar() after, return=0
```

查看模擬結果
， strncmp 參數 2
即為正確密碼

UnCrackable App for Android Level 3

- 目標: 找出隱藏在 App 裡的密碼字串
- 提示: 此 App 有 Root、Debugger 偵測及完整性檢查機制

UnCrackable App for Android Level 3

The crackme from hell!

- Objective: A secret string is hidden somewhere in this app. Find a way to extract it.
- Author: [Bernhard Mueller](#).
- Special thanks to Eduardo Novella for testing, feedback and pointing out flaws in the initial build(s).
- Maintained by the OWASP MSTG leaders.

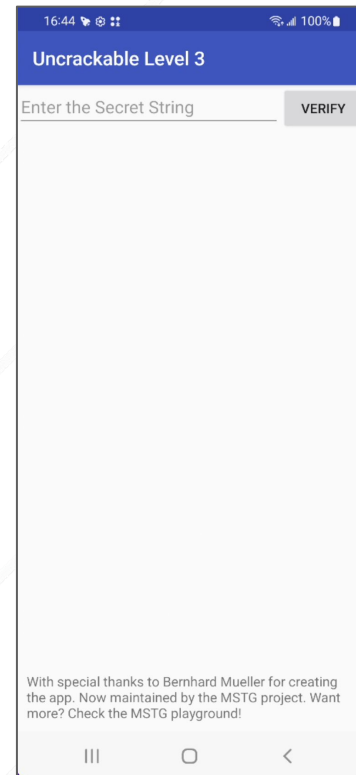
Installation

This app is compatible with Android 4.4 and up.

```
$ adb install UnCrackable-Level3.apk
```

Solutions

- [Solution using Frida by Eduardo Novella](#).
- [Solution using patches by sh3llc0d3r](#).
- [Solution using Ghidra and Frida by Davide Cioccia](#)



逆向分析 - Java 層

App開啟: onCreate() → init(byte[])

```
103 public void onCreate(Bundle bundle) {
104     verifyIbs();
106     init(xorkey.getBytes());
124     new AsyncTask<Void, String, String>() {
        /* Class sg.vantagepoint.uncrackable3.MainActivity.AnonymousClass2 */

        /* access modifiers changed from: protected */
112     public String doInBackground(Void... voidArr) {
113         while (!Debug.isDebuggerConnected()) {
114             SystemClock.sleep(100);
114         }
        return null;
    }

    /* access modifiers changed from: protected */
120     public void onPostExecute(String str) {
121         MainActivity.this.showDialog("Debugger detected!");
122         System.exit(0);
    }
    }.execute(null, null, null);
126 if (RootDetection.checkRoot1() || RootDetection.checkRoot2() || RootDetection.checkRoot3() || IntegrityCheck.isDebuggable(getApplicationContext()) || tampered != 0) {
127     showDialog("Rooting or tampering detected.");
    }
130 this.check = new CodeCheck();
132 super.onCreate(bundle);
133 setContentView(R.layout.activity_main);
}
```

```
36 public class MainActivity extends AppCompatActivity {
    private static final String TAG = "UnCrackable3";
    static int tampered = 0;
    private static final String xorkey = "pizzapizzapizzapizzapizz";
    private CodeCheck check;
    Map<String, Long> crc;

    private native long baz();



    private native void init(byte[] bArr);
}
```

按下確認按鈕: verify(View) → check_code(String) → bar(byte[])

```
137 public void verify(View view) {
138     String obj = ((EditText) findViewById(R.id.edit_text)).getText().toString();
139     AlertDialog create = new AlertDialog.Builder(this).create();
141     if (this.check.check_code(obj)) {
142         create.setTitle("Success!");
143         create.setMessage("This is the correct secret.");
    } else {
145         create.setTitle("Nope...");
146         create.setMessage("That's not it. Try again.");
    }
}
```

```
3 public class CodeCheck {
4     private static final String TAG = "CodeCheck";
5
6     private native boolean bar(byte[] bArr);
7
8     public boolean check_code(String str) {
9         return bar(str.getBytes());
10    }
11 }
```

逆向分析 - Native 層

	Java_sg_vantagepoint_uncrackable3_MainActivity_init	.text
	Java_sg_vantagepoint_uncrackable3_MainActivity_baz	.text
	Java_sg_vantagepoint_uncrackable3_CodeCheck_bar	.text

```
1 __int64 __fastcall Java_sg_vantagepoint_uncrackable3_MainActivity_init(JNIEnv *env, __int64 a2, __int64 xorkey)
2 {
3     __int64 v3; // x19
4     JNIEnv *v4; // x20
5     __int64 v5; // x21
6     __int64 result; // x0
7
8     v3 = xorkey;
9     v4 = env;
10    sub_323C();
11    v5 = ((__int64 (__fastcall *) (JNIEnv *, __int64, _QWORD))(*v4)->GetByteArrayElements)(v4, v3, 0LL);
12    strncpy(&qword_15038, v5, 24LL);
13    result = ((__int64 (__fastcall *) (JNIEnv *, __int64, __int64, signed __int64))(*v4)->ReleaseByteArrayElements)(
14        v4,
15        v3,
16        v5,
17        2LL);
18    ++dword_15054;
19    return result;
20 }
```

- 字串長度為 24
- 輸入字元: *((uint8 *) (v7+v8)) → byte [X21+X8] → W12
- 密碼字元: *((uint8 *) (&v9+v8)) XOR qword_15038[v8]
→ byte [X9+X8] XOR byte [X23+X8]
→ W11 XOR W10 → W10
- v8 → X8 → 0,1,2,3...23
- 關鍵點為 0x3450: CMP W12, W10, 密碼字元在 W10

```
1 signed __int64 __fastcall Java_sg_vantagepoint_uncrackable3_CodeCheck_bar(JNIEnv *env, __int64 a2, __int64 a3)
2 {
3     unsigned __int64 v3; // x22
4     __int64 v4; // x19
5     JNIEnv *v5; // x20
6     signed __int64 result; // x0
7     __int64 v7; // x21
8     __int64 v8; // x8
9     __int64 v9; // [xsp+8h] [xbp-58h]
10    __int64 v10; // [xsp+10h] [xbp-50h]
11    __int64 v11; // [xsp+18h] [xbp-48h]
12    char v12; // [xsp+20h] [xbp-40h]
13    __int64 v13; // [xsp+28h] [xbp-38h]
14
15    v3 = ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2));
16    v4 = a3;
17    v5 = env;
18    result = 0LL;
19    v13 = *( _QWORD *) (v3 + 40);
20    v9 = 0LL;
21    v10 = 0LL;
22    v12 = 0;
23    v11 = 0LL;
24    if ( dword_15054 == 2 )
25    {
26        sub_10E0(&v9);
27        v7 = ((__int64 (__fastcall *) (JNIEnv *, __int64, _QWORD))(*v5)->GetByteArrayElements)(v5, v4, 0LL);
28        if ( ((unsigned int ( __fastcall *) (JNIEnv *, __int64))(*v5)->GetArrayLength)(v5, v4) == 24 )
29        {
30            v8 = 0LL;
31            while ( *(unsigned __int8 *) (v7 + v8) == (qword_15038[v8] ^ *(unsigned __int8 *) (&v9 + v8)) )
32            {
33                if ( (unsigned __int64) ++v8 >= 24 )
34                {
35                    if ( (_DWORD) v8 != 24 )
36                    {
37                        break;
38                        result = 1LL;
39                        goto LABEL_9;
40                    }
41                }
42                result = 0LL;
43            }
44        LABEL_9:
45            *( _QWORD *) (v3 + 40);
46            return result;
47        }
48    }
```

33F4	ADD	X23, SP, #0x60+var_58
33F8	BL	sub_10E0
33FC	LDR	X8, [X20]
3400	MOV	X0, X20
3404	MOV	X1, X19
3408	MOV	X2, XZR
340C	LDR	X8, [X8, #0x5C0]
3410	BLR	X8
3414	LDR	X8, [X20]
3418	MOV	X21, X0
341C	MOV	X0, X20
3420	MOV	X1, X19
3424	LDR	X8, [X8, #0x558]
3428	BLR	X8
342C	CMP	W0, #0x18
3430	B.NE	loc_3474
3434	ADRP	X9, #qword_15038@PAGE
3438	MOV	X8, XZR
343C	ADD	X9, X9, #qword_15038@PAGEOFF
3440		
3440	loc_3440	; CODE XREF: Java_sg_
3440	LDRB	W10, [X23, X8]
3444	LDRB	W11, [X9, X8]
3448	LDRB	W12, [X21, X8]
344C	EOR	W10, W11, W10
3450	CMP	W12, W10

模擬執行 - unidbg

```
emulator.attach().addBreakPoint(module, offset: 0x3450, (emulator, address) → {
    RegisterContext context = emulator.getContext();
    int w10 = context.getIntByReg(Arm64Const.UC_ARM64_REG_W10);
    int w12 = context.getIntByReg(Arm64Const.UC_ARM64_REG_W12);
    System.out.printf("UnCrackable3: w10=0x%X(\\\"%s\\\")", w12=0x%X(\\\"%s\\\")\n", w10, (char) w10, w12, (char) w12);
    emulator.getBackend().reg_write(Arm64Const.UC_ARM64_REG_W12, w10);
    return true;
});
```

指令級 Hook, 顯示密碼字元 W10
並修改 W12 為 W10, 讓字元比對
能繼續下去

```
Pointer jniEnv = vm.getJNIEnv();
DvmObject<?> thiz = vm.resolveClass(className: "owasp.mstg.uncrackable3.MainActivity").newObject(value: null);
List<Object> args = new ArrayList<>();
args.add(jniEnv);
args.add(vm.addLocalObject(thiz));
String xorkey = "pizzapizzapizzapizzapizz";
args.add(vm.addLocalObject(new ByteArray(vm, xorkey.getBytes())));
module.callFunction(emulator, symbolName: "Java_sg_vantagepoint_uncrackable3_MainActivity_init", args.toArray());
```

模擬呼叫 init(byte[])

```
System.out.println("UnCrackable3: bar() before");
thiz = vm.resolveClass(className: "owasp.mstg.uncrackable3.CodeCheck").newObject(value: null);
args = new ArrayList<>();
args.add(jniEnv);
args.add(vm.addLocalObject(thiz));
String inputSecret = "123456789012345678901234";
args.add(vm.addLocalObject(new ByteArray(vm, inputSecret.getBytes())));
int result = module.callFunction(emulator, symbolName: "Java_sg_vantagepoint_uncrackable3_CodeCheck_bar", args.toArray()[0].intValue());
System.out.printf("UnCrackable3: bar() after, return=%d\n", result);
```

傳入長度為 24 的任意字串, 模
擬呼叫 bar(byte[])

```
UnCrackable3: w10=0x6D("m"), w12=0x31("1") UnCrackable3: w10=0x77("w"), w12=0x39("9") UnCrackable3: w10=0x61("a"), w12=0x37("7")
UnCrackable3: w10=0x61("a"), w12=0x32("2") UnCrackable3: w10=0x61("a"), w12=0x38("8") UnCrackable3: w10=0x74("t"), w12=0x38("8")
UnCrackable3: w10=0x68("k"), w12=0x33("3") UnCrackable3: w10=0x73("s"), w12=0x31("1") UnCrackable3: w10=0x20(" "), w12=0x39("9")
UnCrackable3: w10=0x69("i"), w12=0x34("4") UnCrackable3: w10=0x70("p"), w12=0x32("2") UnCrackable3: w10=0x61("a"), w12=0x38("8")
UnCrackable3: w10=0x6E("n"), w12=0x35("5") UnCrackable3: w10=0x20(" "), w12=0x33("3") UnCrackable3: w10=0x67("g"), w12=0x31("1")
UnCrackable3: w10=0x67("g"), w12=0x36("6") UnCrackable3: w10=0x67("g"), w12=0x34("4") UnCrackable3: w10=0x61("a"), w12=0x32("2")
UnCrackable3: w10=0x20(" "), w12=0x37("7") UnCrackable3: w10=0x72("r"), w12=0x35("5") UnCrackable3: w10=0x69("i"), w12=0x33("3")
UnCrackable3: w10=0x6F("o"), w12=0x38("8") UnCrackable3: w10=0x65("e"), w12=0x36("6") UnCrackable3: w10=0x6E("n"), w12=0x34("4")
```

查看模擬結果, W10為正確的字
元, 正確的密碼字串為
“making owasp great again”

模擬執行 - AndroidNativeEmu

```
def hook_code(uc_engine, address, size, user_data):
    if address in instructions_list:
        w10 = uc_engine.reg_read(UC_ARM64_REG_W10)
        w12 = uc_engine.reg_read(UC_ARM64_REG_W12)
        print('UnCrackable3: w10=0x%X("%s"), w12=0x%X("%s")' % (w10, chr(w10), w12, chr(w12)))
        uc_engine.reg_write(UC_ARM64_REG_W12, w10)

emulator = Emulator(vfs_root=posixpath.join(posixpath.dirname(__file__), "vfs"), arch=emu_const.ARCH_ARM64)
libfoo3_module = emulator.load_library("tests/bin64/libfoo3.so")
instructions_list = [libfoo3_module.base + 0x3450]
emulator.mu.hook_add(UC_HOOK_CODE, hook_code, emulator, libfoo3_module.base, libfoo3_module.base + libfoo3_module.size)

try:
    xorkey = b'pizzapizzapizzapizzapizz'
    emulator.call_symbol(libfoo3_module, 'Java_sg_vantagepoint_unCrackable3_MainActivity_init', emulator.java_vm.jni_env.address_ptr, 0x00, Array(bytearray(xorkey)))

    print("UnCrackable3: bar() before")
    input_secret = b'123456789012345678901234'
    result = emulator.call_symbol(libfoo3_module, 'Java_sg_vantagepoint_unCrackable3_CodeCheck_bar', emulator.java_vm.jni_env.address_ptr, 0x00, Array(bytearray(input_secret)))
    print("UnCrackable3: bar() after, return=%d" % result)
except UcError as e:
    print("UnCrackable3: Exit at %x" % emulator.mu.reg_read(UC_ARM64_REG_PC))
    raise
```

指令級 Hook, 顯示密碼字元 W10 並修改 W12 為 W10, 讓字元比對能繼續下去

模擬呼叫 init(byte[])

傳入長度為 24 的任意字串, 模擬呼叫 bar(byte[])

```
UnCrackable3: w10=0x6D("m"), w12=0x31("1") UnCrackable3: w10=0x77("w"), w12=0x39("9") UnCrackable3: w10=0x61("a"), w12=0x37("7")
UnCrackable3: w10=0x61("a"), w12=0x32("2") UnCrackable3: w10=0x61("a"), w12=0x30("0") UnCrackable3: w10=0x74("t"), w12=0x38("8")
UnCrackable3: w10=0x6B("k"), w12=0x33("3") UnCrackable3: w10=0x73("s"), w12=0x31("1") UnCrackable3: w10=0x20(" "), w12=0x39("9")
UnCrackable3: w10=0x69("i"), w12=0x34("4") UnCrackable3: w10=0x70("p"), w12=0x32("2") UnCrackable3: w10=0x61("a"), w12=0x30("0")
UnCrackable3: w10=0x6E("n"), w12=0x35("5") UnCrackable3: w10=0x20(" "), w12=0x33("3") UnCrackable3: w10=0x67("g"), w12=0x31("1")
UnCrackable3: w10=0x67("g"), w12=0x36("6") UnCrackable3: w10=0x67("g"), w12=0x34("4") UnCrackable3: w10=0x61("a"), w12=0x32("2")
UnCrackable3: w10=0x20(" "), w12=0x37("7") UnCrackable3: w10=0x72("r"), w12=0x35("5") UnCrackable3: w10=0x69("i"), w12=0x33("3")
UnCrackable3: w10=0x6F("o"), w12=0x38("8") UnCrackable3: w10=0x65("e"), w12=0x36("6") UnCrackable3: w10=0x6E("n"), w12=0x34("4")
```

查看模擬結果, 將 W10 的字元組合起來即為密碼字串

逆向分析 - 總結

- 善用模擬執行工具可事半功倍，無需手機或模擬器也能逆向分析
- 監控敏感函數的參數或回傳值可獲得許多有用資訊
 - 字串相關: strcmp()、strncmp()、strstr()、strlen()...
 - 檔案相關: open()、read()、write()、mmap()...
- 對於沒有呼叫敏感函數的程式，需要多點耐心去分析

反逆向措施 1 - 自行實作敏感函數

1. 自行實作 libc.so 常用的敏感函數並將函數宣告為 static
2. 將原先呼叫 libc.so 敏感函數的地方改為呼叫自行實作的函數, 不再依賴 libc.so
3. 可避免逆向人員使用逆向工具對 libc.so 敏感函數進行 Hook, 監控參數及回傳值
4. 實作範例: <https://github.com/darvincisec/DetectFrida/tree/master/app/src/main/c>

strncmp() → my_strncmp()

```
static inline int
my_strncmp(const char *s1, const char *s2, size_t n)
{
    if (n == 0)
        return (0);
    do {
        if (*s1 != *s2++)
            return (*(unsigned char *)s1 - *(unsigned char *)--s2);
        if (*s1++ == 0)
            break;
    } while (--n != 0);
    return (0);
}
```

read() → my_read()

```
static inline ssize_t my_read(int __fd, void* __buf, size_t __count){
    return __syscall3(__NR_read, __fd, (long)__buf, (long)__count);
}

static inline long __syscall3(long n, long a, long b, long c)
{
    register long x8 __asm__("x8") = n;
    register long x0 __asm__("x0") = a;
    register long x1 __asm__("x1") = b;
    register long x2 __asm__("x2") = c;
    __asm_syscall("r"(x8), "0"(x0), "r"(x1), "r"(x2));
}

#define __asm_syscall(...) do { \
    __asm__ __volatile__ ( "svc 0" \
    : "=r"(x0) : __VA_ARGS__ : "memory", "cc"); \
    return x0; \
} while (0)
```

反逆向措施 1 - 自行實作敏感函數效果

```
__int64 __fastcall sub_8B58(_JNIEnv *a1)
{
    unsigned __int64 v1; // x8
    _JNIEnv *v2; // ST58_8
    unsigned __int64 v3; // ST38_8
    __int64 v4; // x0
    __int64 v5; // ST48_8
    __int64 v6; // x0
    __int64 v7; // ST40_8
    __int64 v8; // x0
    __int64 v9; // ST30_8
    unsigned int v10; // w0
    __int64 result; // x0
    char v12; // [xsp+60h] [xbp-230h]
    char v13; // [xsp+68h] [xbp-228h]
    char v14; // [xsp+70h] [xbp-220h]
    char v15; // [xsp+78h] [xbp-218h]
    __int64 v16; // [xsp+278h] [xbp-18h]

    v1 = _ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2));
    v16 = *(_QWORD *) (v1 + 40);
    v2 = a1;
    v3 = v1;
    v4 = sub_8FB8(&v14);
    v5 = ay::obfuscated_data<15ull,8432754782978560477ull>::operator char *(v4);
    v6 = sub_90AC(&v13);
    v7 = ay::obfuscated_data<15ull,5164447508740687751ull>::operator char *(v6);
    v8 = sub_91A0(&v12);
    v9 = ay::obfuscated_data<27ull,7305246811387187981ull>::operator char *(v8);
    v10 = strcmp(v5, v7, 14LL);
    sprintf(&v15, v9, v10);
    result = _JNIEnv::NewStringUTF(v2, &v15);
    *(_QWORD *) (v3 + 40);
    return result;
}
```

```
__int64 __fastcall sub_8B18(_JNIEnv *a1)
{
    unsigned __int64 v1; // x8
    _JNIEnv *v2; // ST58_8
    unsigned __int64 v3; // ST38_8
    __int64 v4; // x0
    unsigned __int8 *v5; // ST48_8
    __int64 v6; // x0
    unsigned __int8 *v7; // ST40_8
    __int64 v8; // x0
    __int64 v9; // ST30_8
    unsigned int v10; // ST2C_4
    __int64 result; // x0
    char v12; // [xsp+60h] [xbp-230h]
    char v13; // [xsp+68h] [xbp-228h]
    char v14; // [xsp+70h] [xbp-220h]
    char v15; // [xsp+78h] [xbp-218h]
    __int64 v16; // [xsp+278h] [xbp-18h]

    v1 = _ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2));
    v16 = *(_QWORD *) (v1 + 40);
    v2 = a1;
    v3 = v1;
    v4 = sub_8F78(&v14);
    v5 = (unsigned __int8 *)ay::obfuscated_data<15ull,1122904582700321319ull>::operator char *(v4);
    v6 = sub_906C(&v13);
    v7 = (unsigned __int8 *)ay::obfuscated_data<15ull,4171390554486297459ull>::operator char *(v6);
    v8 = sub_9160(&v12);
    v9 = ay::obfuscated_data<27ull,14375339529603991449ull>::operator char *(v8);
    v10 = sub_9254(v5, v7, 14LL);
    sprintf(&v15, v9, v10);
    result = _JNIEnv::NewStringUTF(v2, &v15);
    *(_QWORD *) (v3 + 40);
    return result;
}
```

```
v8 = a1;
v7 = a2;
v6 = a3;
if ( a3 )
{
    do
    {
        v3 = v7++;
        if ( *v8 != *v3 )
            return *v8 - (unsigned int)*(v7 - 1);
        v4 = v8++;
        if ( !*v4 )
            break;
        --v6;
    }
    while ( v6 );
    v9 = 0;
}
else
{
    v9 = 0;
}
return v9;
```

- strcmp() → my_strcmp()
- 函數名稱 my_strcmp 被隱藏, 變成檔案位址 sub_9254
- sub_9254 函數內容不易被辨識為 strcmp()

反逆向措施 2 - 程式碼混淆

- 使用帶有混淆功能的 C/C++ 編譯器 O-LLVM 進行程式碼混淆：
 1. FLA: Control Flow Flattening, 控制流程平坦化
 2. SUB: Instructions Substitution, 指令替換
 3. BCF: Bogus Control Flow, 虛假控制流程
- 在 CMakeLists.txt 裡面新增 O-LLVM 相關編譯設定即可啟用
- O-LLVM: <https://github.com/darvincisec/o-llvm-binary>

```
set(OLLVM_PATH ${CMAKE_HOME_DIRECTORY}/../../../../../build/bin)
set(OLLVM_C_COMPILER ${OLLVM_PATH}/clang)
set(OLLVM_CXX_COMPILER ${OLLVM_PATH}/clang++)
```

```
set(OLLVM_C_FLAGS "-mllvm -fla -mllvm -sub -mllvm -bcf")
```

```
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} ${OLLVM_C_FLAGS}")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} ${OLLVM_C_FLAGS}")
set(CMAKE_C_COMPILER ${OLLVM_C_COMPILER})
set(CMAKE_CXX_COMPILER ${OLLVM_CXX_COMPILER})
```


反逆向措施 2 - 程式碼混淆效果

原始碼

```
static size_t my_strlen(const char *s)
{
    size_t len = 0;
    while (*s++)
    {
        len++;
    }
    return len;
}
```

正常編譯後反編譯

```
int64 __fastcall sub_9160(_BYTE *a1)
{
    _BYTE *v1; // x10
    int64 i; // [xsp+0h] [xbp-10h]
    _BYTE *v4; // [xsp+8h] [xbp-8h]

    v4 = a1;
    for ( i = 0LL; ; ++i )
    {
        v1 = v4++;
        if ( !*v1 )
            break;
    }
    return i;
}
```

經 O-LLVM 編譯後反編譯

```
int64 __fastcall sub_95DC(_BYTE *a1)
{
    _BYTE *v1; // x12
    signed int v2; // w8
    signed int v4; // [xsp+Ch] [xbp-14h]
    int64 v5; // [xsp+10h] [xbp-10h]
    _BYTE *v6; // [xsp+18h] [xbp-8h]

    v6 = a1;
    v5 = 0LL;
    v4 = 1217852730;
    do
    {
        while ( 1 )
        {
            while ( v4 == 1217852730 )
            {
                v1 = v6++;
                if ( *v1 )
                    v2 = 791389876;
                else
                    v2 = 1334314678;
                v4 = v2;
            }
            if ( v4 != 791389876 )
                break;
            ++v5;
            v4 = 1217852730;
        }
    }
    while ( v4 != 1334314678 );
    return v5;
}
```

- 正常編譯後進行反編譯，結果與原始碼差異不大
- 經 O-LLVM 編譯後反編譯，出現許多 while 及 if else
- 混淆後逆向難度大幅上升

反逆向措施 - 總結

- 模擬執行工具可監控任意函數及指令執行時的 CPU 暫存器數值
 - 使用 O-LLVM 進程式碼混淆, 避免監控點輕易被找到
- 外部敏感函數呼叫易被監控
 - 自行實作敏感函數, 不呼叫外部敏感函數
- 自行實作的敏感函數可搭配程式碼混淆, 反逆向效果更佳

總結

如何增加 NDK 程式的逆向難度

1. 將函數宣告為 static 類型，避免暴露函數名稱
2. JNI 函數使用 RegisterNatives 進行動態註冊
3. 所有字串使用字串加密巨集進行加密
4. 敏感函數自行實作，不呼叫系統函示庫的敏感函數
5. 所有程式碼使用 O-LLVM 進行混淆
6. 使用商用 App 安全防護產品加密整個 .so 檔

反逆向措施 - 加密 .so 檔

加密前

```
Function name
start
sub_800C
JNI_OnLoad
_JavaVM::GetEnv(void**)
_JNIEnv::FindClass(char const*)
sub_829C
ay:obfuscated_data<32ull,182331>
sub_8390
ay:obfuscated_data<14ull,802254>
sub_8484
ay:obfuscated_data<21ull,802254>
sub_8578
_JNIEnv::RegisterNatives(_jclass*,
ay:obfuscated_data<32ull,182331>
ay:obfuscated_data<32ull,182331>
ay:obfuscated_data<32ull,1823317168>
ay:obfuscated_data<14ull,802254>
ay:obfuscated_data<14ull,802254>
ay:obfuscated_data<14ull,80225443531>
ay:obfuscated_data<21ull,802254>
ay:obfuscated_data<21ull,80225443531>
_JNIEnv::NewStringUTF(char const
sub_89C8
ay:obfuscated_data<15ull,417139C
```

```
LOAD:0000000000000000 ; Format : ELF64 for ARM64 (Shared object)
LOAD:0000000000000000 ; Needed Library 'liblog.so'
LOAD:0000000000000000 ; Needed Library 'libm.so'
LOAD:0000000000000000 ; Needed Library 'libdl.so'
LOAD:0000000000000000 ; Needed Library 'libc.so'
LOAD:0000000000000000 ; Shared Name 'libndk.so'
LOAD:0000000000000000 ;
LOAD:0000000000000000 ; Processor : ARM
LOAD:0000000000000000 ; ARM architecture: metaarm
LOAD:0000000000000000 ; Target assembler: Generic assembler for ARM
LOAD:0000000000000000 ; Byte sex : Little endian
LOAD:0000000000000000 ;
LOAD:0000000000000000 ; Segment type: Pure code
LOAD:0000000000000000 AREA LOAD, CODE, ALIGN=0
LOAD:0000000000000000 CODIE64
LOAD:0000000000000000 dword_0 DCD 0x464C457E ; DATA XREF: LOAD:0000000000000000EFOj.o
LOAD:0000000000000000 ; LOAD:000000000000000010A0j.o ...
LOAD:0000000000000000 ; File format: \x7FELF
LOAD:0000000000000004 DCB 2 ; File class: 64-bit
LOAD:0000000000000005 DCB 1 ; Data encoding: little-endian
LOAD:0000000000000006 DCB 1 ; File version
LOAD:0000000000000007 DCB 0 ; OS/ABI: UNIX System V ABI
LOAD:0000000000000008 DCB 0 ; ABI Version
LOAD:0000000000000009 DCB 0, 0, 0, 0, 0, 0 ; DATA XREF: ay:obfuscated_data<14ull,802254>
LOAD:0000000000000009 ; ay:obfuscated_data<15ull,417139055448
LOAD:0000000000000009 ; Padding
LOAD:0000000000000009 ; DATA XREF: sub_34858<214j
LOAD:0000000000000010 word_10 DCD 3 ; File type: Shared object
LOAD:0000000000000012 DCDW 0xB7 ; Machine: ARM64
LOAD:0000000000000014 DCD 1 ; DATA XREF: ay:obfuscated_data<21ull,80225443531>
LOAD:0000000000000014 ; File version
LOAD:0000000000000018 DCQ start ; Entry point
LOAD:0000000000000020 DCQ 0x40 ; DATA XREF: ay:obfuscated_data<32ull,1823317168>
LOAD:0000000000000020 ; PHF file offset
```

有效的 .so 檔, 可進行反組譯分析

加密後

```
Function name
qword_8
db 73h a
7Ah, 8Dh, 0BFh, 5Fh, 0B3h, 8Bh, 0E1h
dQ 0FA4365CF15A6BA6Ch, 0CA194EFC0A51B80Fh, 0ECD46EB696D6168h
dQ 92A49BDEA4160504h, 85A34151ABFC039h, 0E53B124A260E67Ch
dQ 0AD167C20CFF6767h, 8CA01D140042E91Ah, 59A2803AB0CC2E7h
dQ 0F43E718A00C72B5h, 0B93A953E4613E616h, 0DA87F557E2AC92Ch
dQ 987440BB5652D8222h, 1679A37700B9734Ch, 4502A2F8678A21h
dQ 1892A49405DC17Fh, 0CEDA6E24416A9A04h, 0AF98970417A988C9h
dQ 0A69D05A07A8A8384h, 394364486D05D1DCh, 0EE2737722AAB42Eh
dQ 34DDFF1221F7C1ABh, 80EACE7F1211921h, 782423E2C6921ECh
dQ 6531183A2C1AC886h, 9406394D143E326h, 2371FC0E0233090h
dQ 79C9F68CC4598Bh, 957A5FE7467C891h, 2A2E5BC31A8E742h
dQ 54D41923E0A05AFCh, 0B8A184002C78766h, 431E931956B4A8h
dQ 8B437277BBF3A1Bh, 5A1C230E34825F0h, 0CFF5958212A00377h
dQ 79220357A5F86Ch, 93636B81A806CAh, 0E961F6E24338ECh
dQ 9ED493DAB886EBBh, 94F1D05684F9B9Fh, 38E85FF0E20180Fh
dQ 0F474FC3E2B4657Bh, 0AD64FA916B17A85h, 0C92FC88415EDBAEh
dQ 0A13B41177A1F0BAh, 0E05835A813728A8h, 7B4CB1076FF4ADCh
dQ 0A8F89A8E8B81B2h, 2670778C4230A56h, 0C7C61028C53C0Dh
dQ 694F4ADE0473C98h, 0D853B814B127E92h, 0F0B126C91667E8Eh
dQ 15379FA062E6E71h, 7AA96E610405B0A8h, 7E441B034278E658h
dQ 0D259418AB285E34Bh, 0CDB2DEDF209C98Ch, 8BF769C0E82658h
dQ 832C8078E48EB70h, 0D25732138D3C5FF4h, 329AB33C3770288h
dQ 0DEE779336822F6h, 33F979A052C09D2h, 1563D963035A55A0h
dQ C3E074685316409h, 0CE2E4471E2C040h, 3661890EB95A44F8h
dQ 2AD2D6857CC67078h, 0BDC158CF65D0D7Eh, 8C4D8E978725E7Eh
dQ 0BC3EB8D5A0F1068h, 2F8BDFC781693D9h, 5B925E8E2CA706A8h
dQ 5P233C97CDF926Eh, 82DF50CCE8BA48Fh, 59A8A044FD912370h
dQ 0ADD7F1FEF3E3074h, 8341FF401CDAF4ABh, 6201569BDD14798Bh
dQ 2D81013BE16C4F0h, 1FDBE144C4A91C8Eh, 0AB86CBFF9E9C5DACH
dQ 351282576D2B6FE1h, 0F2C8C8E76C4C051h, 0C9090E7850A870Eh
dQ 0CA6959A313F2EACCh, 76C73A48E7D6C32Bh, 0AC72F38E2C84744h
dQ 6ED298BF62DEBD9h, 625E8A9B70CFC5ACh, 0CB315016A8FCE8Fh
dQ 0A7BD241A102CFCh, 1DA57721253D6F95h, 05C8A22484C2C83h
dQ 967BEAACC70EE77h, 0FDEBF40B2D04FF7h, 0E8C3A9A76490D1Fh
dQ 90FC1B92932505FDh, 789A8D045F739A8h, 5BC6C4452A3F55h
dQ 125B17B0E9A8C797h, 0AD16E881A3CC9A78h, 8E230F5561E74E4Fh
dQ 0F41A1373B20998Dh, 26E678E6E10813Dh, 96C77849E98C508h
```

無效的 .so 檔, 都是亂碼無法分析

Thank you :)

王羿廷 | Jason Wang

github.com/jasonwang1018

