

Writeup "Quizás"

En el contexto de los desafíos propuestos en la charla *Frida en +/- 45 Minutos* realizada el día 31 de Agosto en la conferencia de sombreros blancos, se presentan las opciones de resolución del desafío "Quizás".

Challenge

2 Solves

×

Quizás

500

La aplicación implementa métodos para muchas cosas distintas, incluso para algunas que podrían manejarse de otras formas.

Si le sabes a frida cierto?

Formato flag: `frida{.*}`

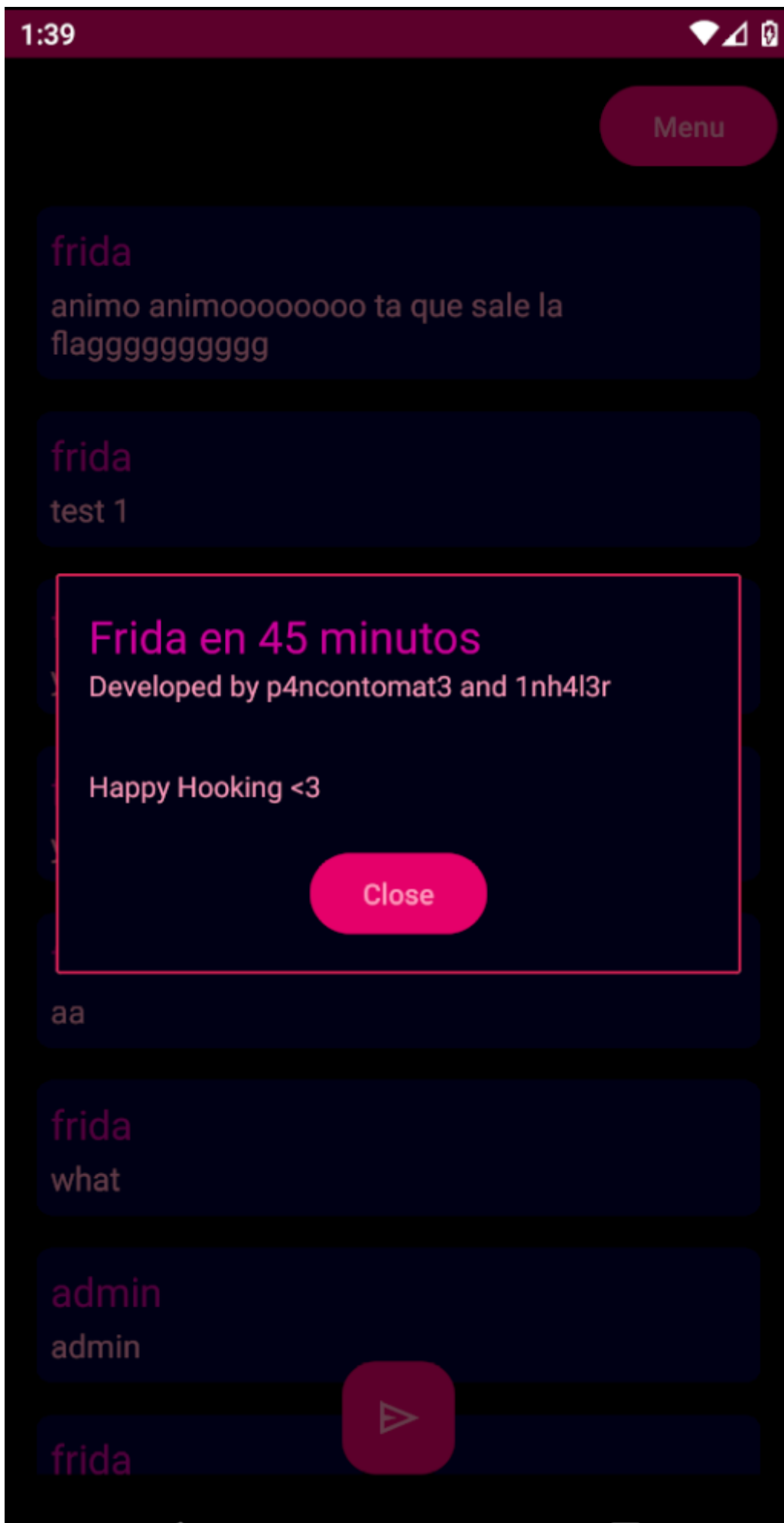
 fridaen45mi...

Flag

Submit

1. Solución 1: Método con frida

Al revisar el about de la aplicación, vemos un mensaje que indica el nombre de los desarrolladores del laboratorio.



Al analizar el código fuente del about vemos que el *TextView aboutTextView* define el texto a mostrar en base a `nativeText(aboutText()) + "\n\n\nHappy Hooking <3"`. Si revisamos *aboutText()* vemos que este retorna el string "yes".

```

package com.nivel4.Dialogs;

import android.app.AlertDialog;
import android.app.Dialog;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import androidx.fragment.app.DialogFragment;
import com.nivel4.fridaen45minutos.R;

/* Loaded from: classes.dex */
public class About extends DialogFragment {
    public static String aboutText() {
        return "yes";
    }

    private native String nativeText(String str);

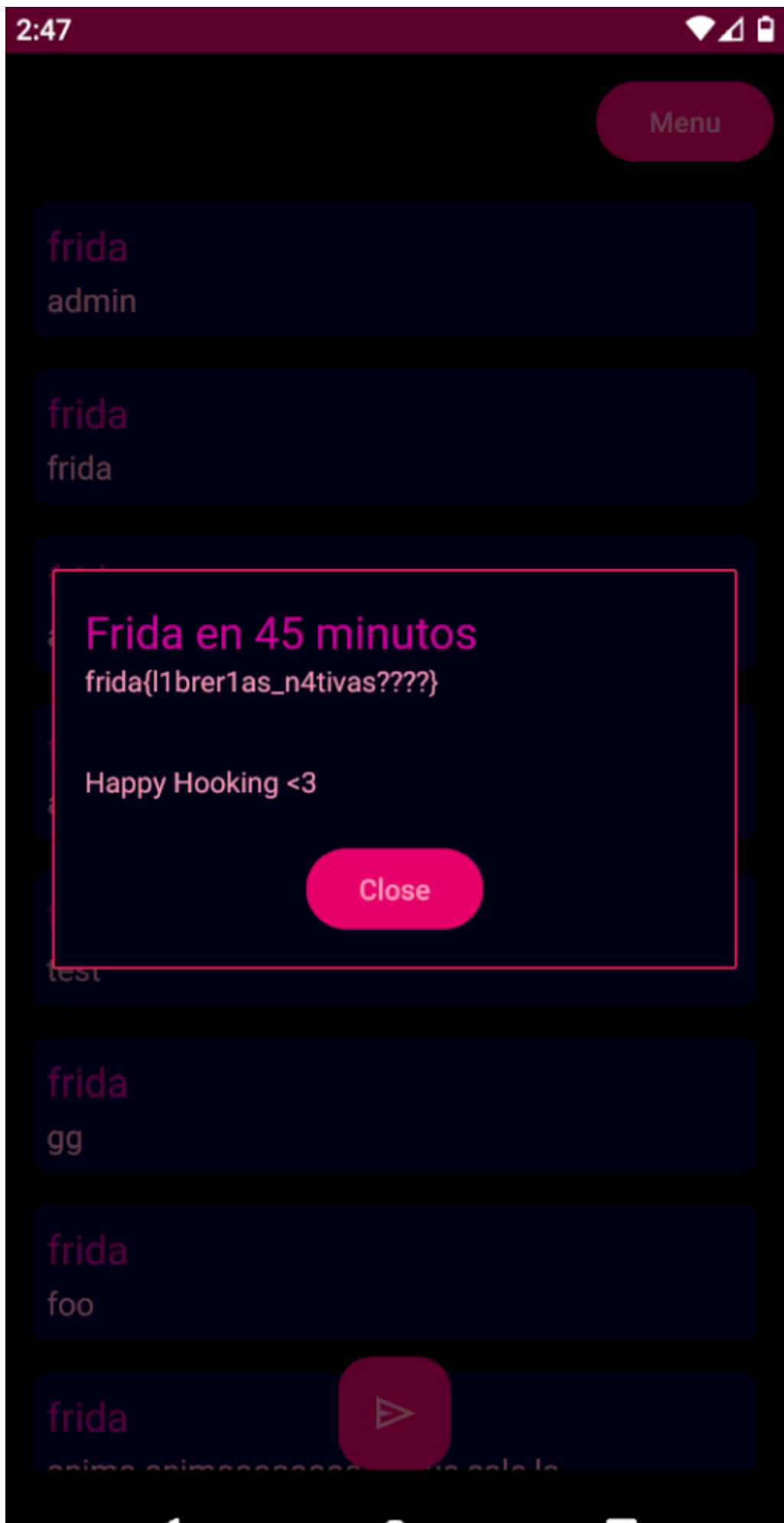
    static {
        System.loadLibrary("native-lib");
    }

    @Override // androidx.fragment.app.DialogFragment
    public Dialog onCreateDialog(Bundle bundle) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        View inflate = requireActivity().getLayoutInflater().inflate(R.layout.dialog_about, (ViewGroup) null);
        ((TextView) inflate.findViewById(R.id.labelTextView)).setText("Frida en 45 minutos");
        ((TextView) inflate.findViewById(R.id.aboutTextView)).setText(nativeText(aboutText()) + "\n\n\nHappy Hooking <3");
        ((Button) inflate.findViewById(R.id.closeButton)).setOnClickListener(new View.OnClickListener() { // from class: com.nivel4.Dialogs.About.1
            @Override // android.view.View.OnClickListener
            public void onClick(View view) {
                About.this.dismiss();
            }
        });
        builder.setView(inflate);
        return builder.create();
    }
}

```

Utilizando frida, hacemos un hook para modificar el argumento de *nativeText*, esto puede hacerse hookeando *aboutText()* y forzando que el valor de retorno sea algo distinto. Si se sobrecarga de modo que el argumento de *nativeText* sea "maybe", es posible obtener la flag del desafío.

```
JS hooks.js x
JS hooks.js > Java.perform() callback > implementation
1  if(Java.available){
2    Java.perform(function(){
3      console.log("");
4
5      const newArray = Java.array('java.lang.String', ['/fake/fake', '/fake/fake', '/fake/fake',
6
7      const rootChecker = Java.use("com.nivel4.RootChecker.rootChecker");
8
9      rootChecker.packages.value = newArray;
10
11     rootChecker.checkSu.implementation = function(){
12       console.log("checkSu returned", this.checkSu());
13       return false;
14     };
15
16     rootChecker.testKeys.implementation = function(){
17       console.log("testKeys returned", this.testKeys());
18       return false;
19     };
20
21     rootChecker.checkPackages.implementation = function(){
22       console.log("checkPackages returned", this.checkPackages());
23       return this.checkPackages();
24     };
25
26     var About = Java.use('com.nivel4.Dialogs.About');
27     About.aboutText.implementation = function(){
28       var retval = "maybe";
29       console.log("nativeText(\"\" + retval + "\")");
30       return retval;
31     };
32
33   })
34 }
```



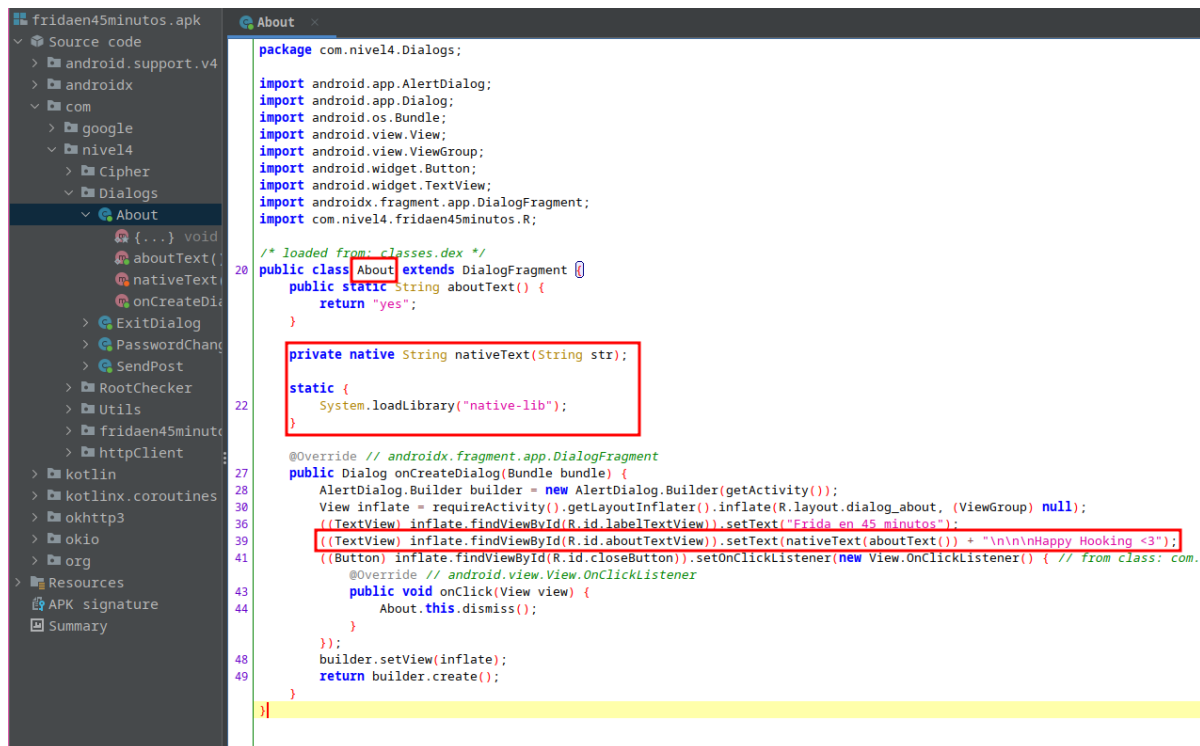
Cabe señalar que también es posible realizar este ataque de argument overload hookeando directamente a *nativeText*.

```
const about = Java.use("com.nivel4.Dialogs.About");  
  
about.nativeText.implementation = function(str) {  
  var result = this.nativeText("maybe");  
  console.log("nativeText('" + str + "') = " + result);  
  return result;  
}
```

2. Solución 2: método sin frida (legal?).

Tal como su nombre lo indica, hay una manera de resolver el desafío **sin utilizar frida**. Para ello, hay que tener en consideración ciertas cosas.

Luego, al revisar el código fuente de este dialog con JADX, vemos que tiene un método que declara el uso de una librería nativa.



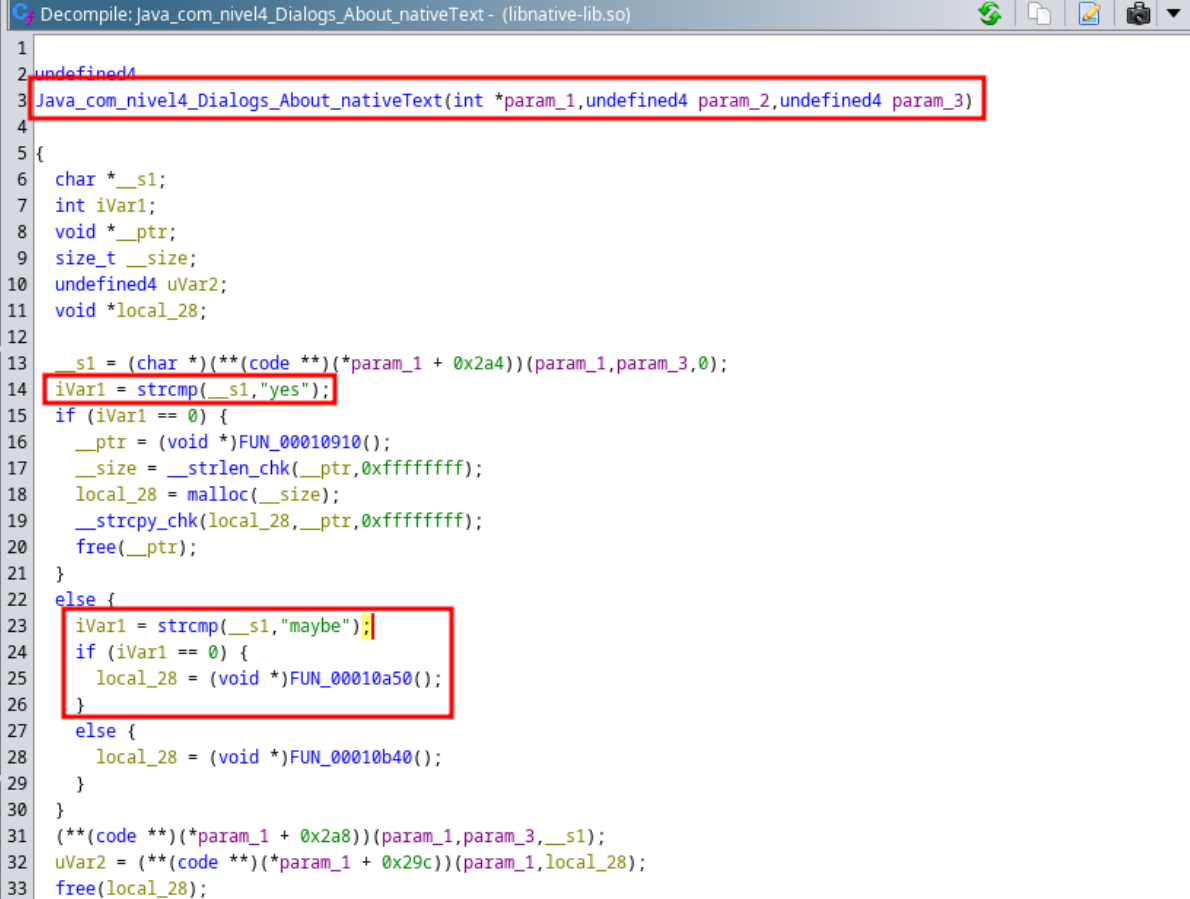
Decompilamos la aplicación utilizando **APKTOOL**.

```
randmcnally :: p4n/charla/lab » apktool d fridaen45minutos.apk
I: Using Apktool 2.7.0 on fridaen45minutos.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/p4n/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
I: Copying META-INF/services directory
randmcnally :: p4n/charla/lab » |
```

En el directorio *fridaen45minutos/lib/x86* vemos el shared-object *libnative-lib.so* que es utilizado en la clase del *About*.


```
randmcnally :: p4n/charla/lab » cd fridaen45minutos/lib/x86
randmcnally :: fridaen45minutos/lib/x86 » ls -la
-rw-r--r-- 4.3k p4n  1 Sep 13:44 Δ libnative-lib.so
randmcnally :: fridaen45minutos/lib/x86 » |
```

Utilizando Ghidra para analizar la librería, vemos en el código de *nativeText* que esta valida si el string proporcionado es "yes" o "maybe" y según ello gatilla un comportamiento distinto.



```
Decompile: Java_com_nivel4_Dialogs_About_nativeText - (libnative-lib.so)
1
2 undefined4
3 Java_com_nivel4_Dialogs_About_nativeText(int *param_1,undefined4 param_2,undefined4 param_3)
4
5 {
6   char *__s1;
7   int iVar1;
8   void *__ptr;
9   size_t __size;
10  undefined4 uVar2;
11  void *local_28;
12
13  __s1 = (char *) (**(code **)(*param_1 + 0x2a4))(param_1,param_3,0);
14  iVar1 = strcmp(__s1,"yes");
15  if (iVar1 == 0) {
16    __ptr = (void *)FUN_00010910();
17    __size = __strlen_chk(__ptr,0xffffffff);
18    local_28 = malloc(__size);
19    __strcpy_chk(local_28,__ptr,0xffffffff);
20    free(__ptr);
21  }
22  else {
23    iVar1 = strcmp(__s1,"maybe");
24    if (iVar1 == 0) {
25      local_28 = (void *)FUN_00010a50();
26    }
27    else {
28      local_28 = (void *)FUN_00010b40();
29    }
30  }
31  (**(code **)(*param_1 + 0x2a8))(param_1,param_3,__s1);
32  uVar2 = (**(code **)(*param_1 + 0x29c))(param_1,local_28);
33  free(local_28);
```

Si revisamos el código de la función ejecutada, al evaluar el string "maybe" vemos que se declara un array con valores en hexadecimal. Entre todas las opciones que tenemos, podemos directamente convertirlo a texto y obtener la flag del desafío.

Decompile: FUN_00010a50 - (libnative-lib.so)

```
1
2 Undefined * FUN_00010a50(void)
3
4 {
5     undefined *puVar1;
6
7     puVar1 = (undefined *)malloc(0x1c);
8     *puVar1 = 0x66;
9     puVar1[1] = 0x72;
10    puVar1[2] = 0x69;
11    puVar1[3] = 100;
12    puVar1[4] = 0x61;
13    puVar1[5] = 0x7b;
14    puVar1[6] = 0x6c;
15    puVar1[7] = 0x31;
16    puVar1[8] = 0x62;
17    puVar1[9] = 0x72;
18    puVar1[10] = 0x65;
19    puVar1[0xb] = 0x72;
20    puVar1[0xc] = 0x31;
21    puVar1[0xd] = 0x61;
22    puVar1[0xe] = 0x73;
23    puVar1[0xf] = 0x5f;
24    puVar1[0x10] = 0x6e;
25    puVar1[0x11] = 0x34;
26    puVar1[0x12] = 0x74;
27    puVar1[0x13] = 0x69;
28    puVar1[0x14] = 0x76;
29    puVar1[0x15] = 0x61;
30    puVar1[0x16] = 0x73;
31    puVar1[0x17] = 0x3f;
32    puVar1[0x18] = 0x3f;
33    puVar1[0x19] = 0x3f;
34    puVar1[0x1a] = 0x3f;
35    puVar1[0x1b] = 0x7d;
36    return puVar1;
37 }
38
```

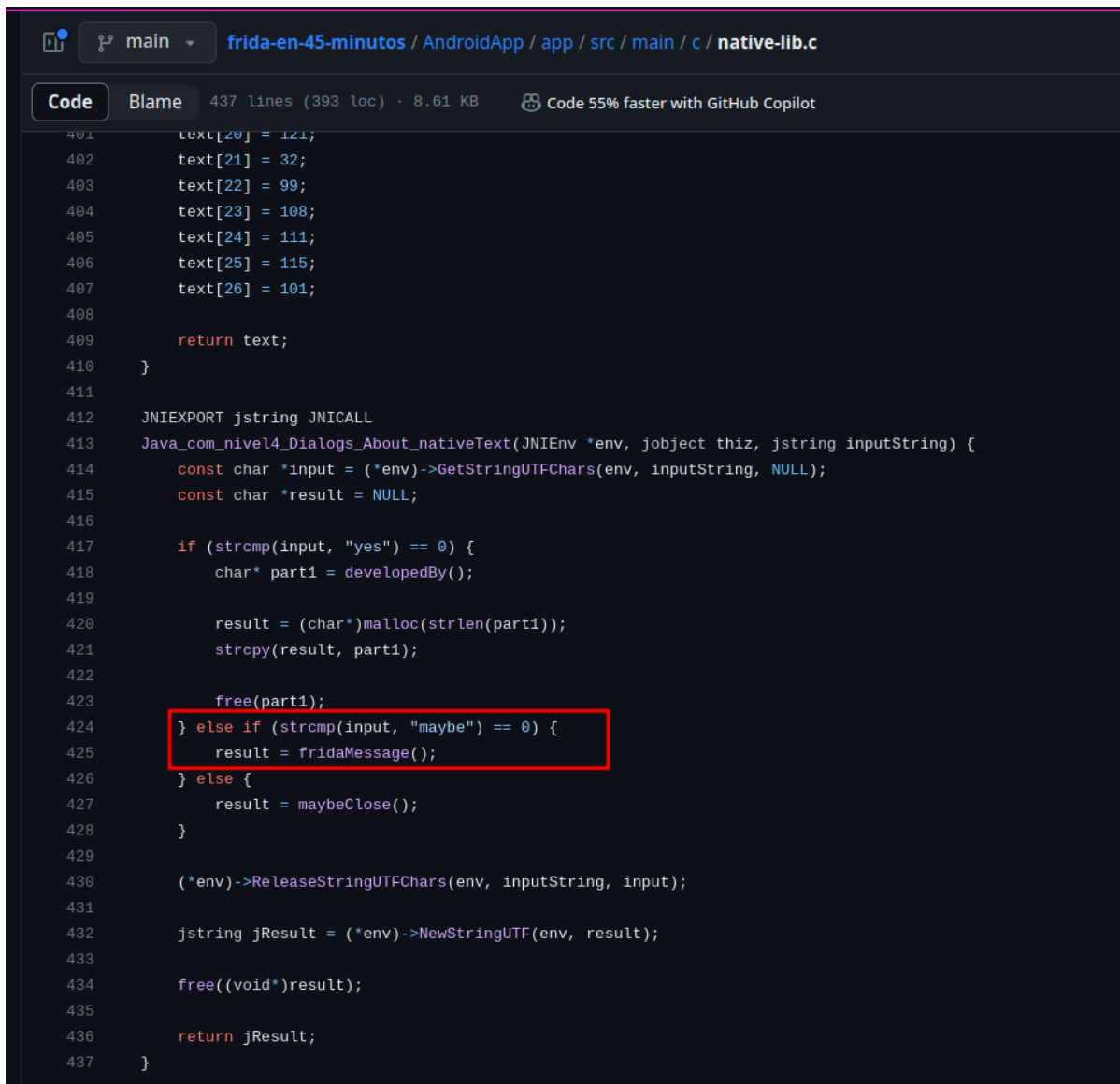
Input	
0x66	
0x72	
0x69	
0x64	
0x61	
0x7b	
0x6c	
0x31	
0x62	
0x72	
0x65	
0x72	
0x31	
0x61	
0x73	
0x5f	
0x6e	
0x34	
0x74	
0x69	
REC 139	28
Output	
frida{librerias_n4tivas????}	

3. Solución 3: método sin frida (inesperado?)

Como vimos en la parte anterior, al revisar el código fuente del dialog de about con JADX, vemos que tiene un método que declara el uso de una librería nativa.

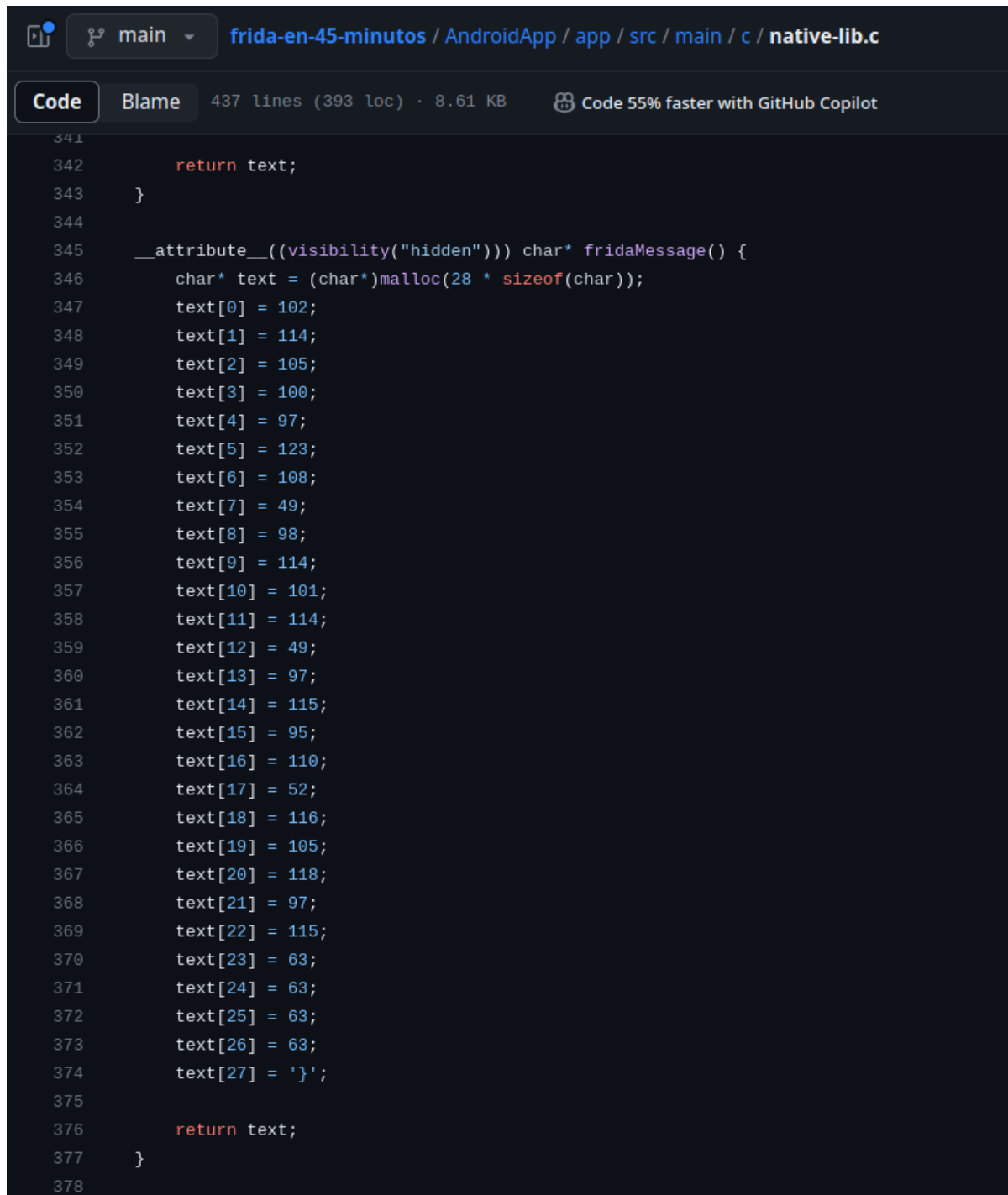


Si revisamos en github el código fuente de esta librería vemos que el export JNI ejecuta la función *fridaMessage()* si el argumento de ejecución de *nativeText* es "maybe".



```
401     text[20] = 121;
402     text[21] = 32;
403     text[22] = 99;
404     text[23] = 108;
405     text[24] = 111;
406     text[25] = 115;
407     text[26] = 101;
408
409     return text;
410 }
411
412 JNIEXPORT jstring JNICALL
413 Java_com_nivel4_Dialogs_About_nativeText(JNIEnv *env, jobject thiz, jstring inputString) {
414     const char *input = (*env)->GetStringUTFChars(env, inputString, NULL);
415     const char *result = NULL;
416
417     if (strcmp(input, "yes") == 0) {
418         char* part1 = developedBy();
419
420         result = (char*)malloc(strlen(part1));
421         strcpy(result, part1);
422
423         free(part1);
424     } else if (strcmp(input, "maybe") == 0) {
425         result = fridaMessage();
426     } else {
427         result = maybeClose();
428     }
429
430     (*env)->ReleaseStringUTFChars(env, inputString, input);
431
432     jstring jResult = (*env)->NewStringUTF(env, result);
433
434     free((void*)result);
435
436     return jResult;
437 }
```

En el código fuente buscamos la función *fridaMessage()* y la utilizamos para compilarlo en local agregando un *printf*.



```
341
342     return text;
343 }
344
345 __attribute__((visibility("hidden"))) char* fridaMessage() {
346     char* text = (char*)malloc(28 * sizeof(char));
347     text[0] = 102;
348     text[1] = 114;
349     text[2] = 105;
350     text[3] = 100;
351     text[4] = 97;
352     text[5] = 123;
353     text[6] = 108;
354     text[7] = 49;
355     text[8] = 98;
356     text[9] = 114;
357     text[10] = 101;
358     text[11] = 114;
359     text[12] = 49;
360     text[13] = 97;
361     text[14] = 115;
362     text[15] = 95;
363     text[16] = 110;
364     text[17] = 52;
365     text[18] = 116;
366     text[19] = 105;
367     text[20] = 118;
368     text[21] = 97;
369     text[22] = 115;
370     text[23] = 63;
371     text[24] = 63;
372     text[25] = 63;
373     text[26] = 63;
374     text[27] = '}' ;
375
376     return text;
377 }
378
```

```
randmcnally :: ~/p4n » cat frida_msg.c
#include <string.h>
#include <stdlib.h>

__attribute__((visibility("hidden"))) char* fridaMessage() {
    char* text = (char*)malloc(28 * sizeof(char));
    text[0] = 102;
    text[1] = 114;
    text[2] = 105;
    text[3] = 100;
    text[4] = 97;
    text[5] = 123;
    text[6] = 108;
    text[7] = 49;
    text[8] = 98;
    text[9] = 114;
    text[10] = 101;
    text[11] = 114;
    text[12] = 49;
    text[13] = 97;
    text[14] = 115;
    text[15] = 95;
    text[16] = 110;
    text[17] = 52;
    text[18] = 116;
    text[19] = 105;
    text[20] = 118;
    text[21] = 97;
    text[22] = 115;
    text[23] = 63;
    text[24] = 63;
    text[25] = 63;
    text[26] = 63;
    text[27] = '}' ;

    return text;
}

int main(){
    const char *result = NULL;

    result = fridaMessage();
    printf(result);
}
randmcnally :: ~/p4n » |
```

```
randmcnally :: ~/p4n » ./frida_msg  
frida{librerias_n4tivas????}%  
randmcnally :: ~/p4n » |
```