Android Mobile Pentest 101

Lecture 7.5 – Lab: Practice Frida

Goal: Improve reversing & hooking skills

Task

- Install the challenge2_release.apk (This is challenge from h1-702 2018 CTF): https://github.com/tsug0d/AndroidMobilePentest101/blob/master/lab/frida_lab/challenge2_release.apk
- Try to reverse and use frida to crack the pin, then get the flag

Lecture End Here

Next slide is about hints to solve the challenge, please try hard first ☺

(No Pain, No Gain!)

Hint

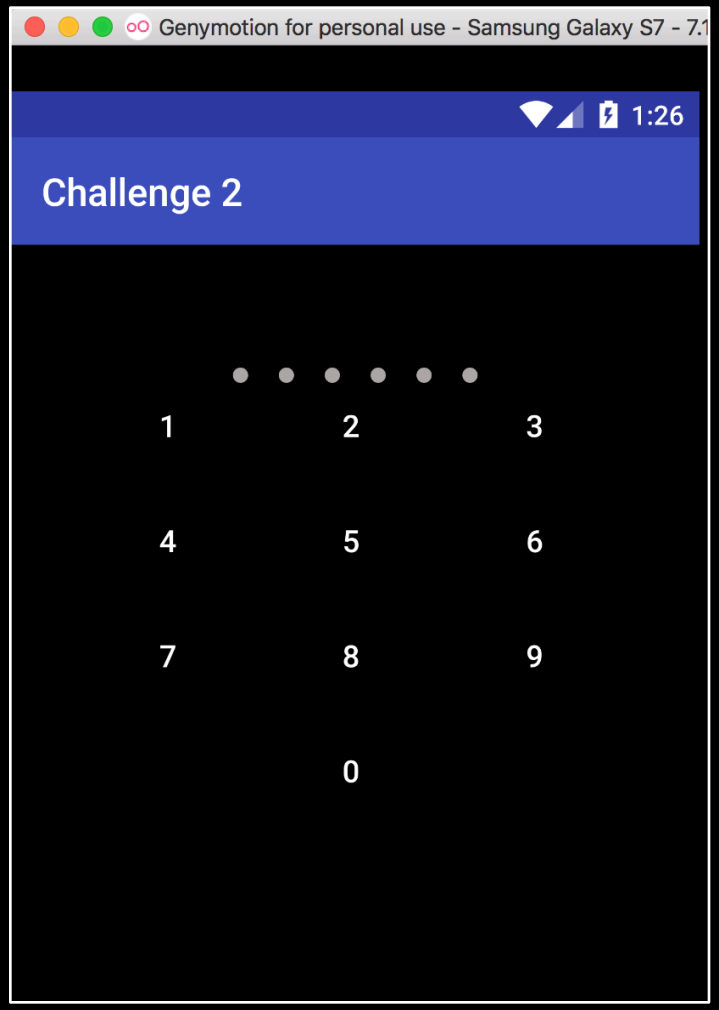
- Brute-force pin
- public native void resetCoolDown();

Stuck? Here the solution…

- Full script:
https://github.com/tsug0d/AndroidMobilePentest101/blob/master/lab/frida_lab/lab_frida.py
- Explanation:
(below slide)

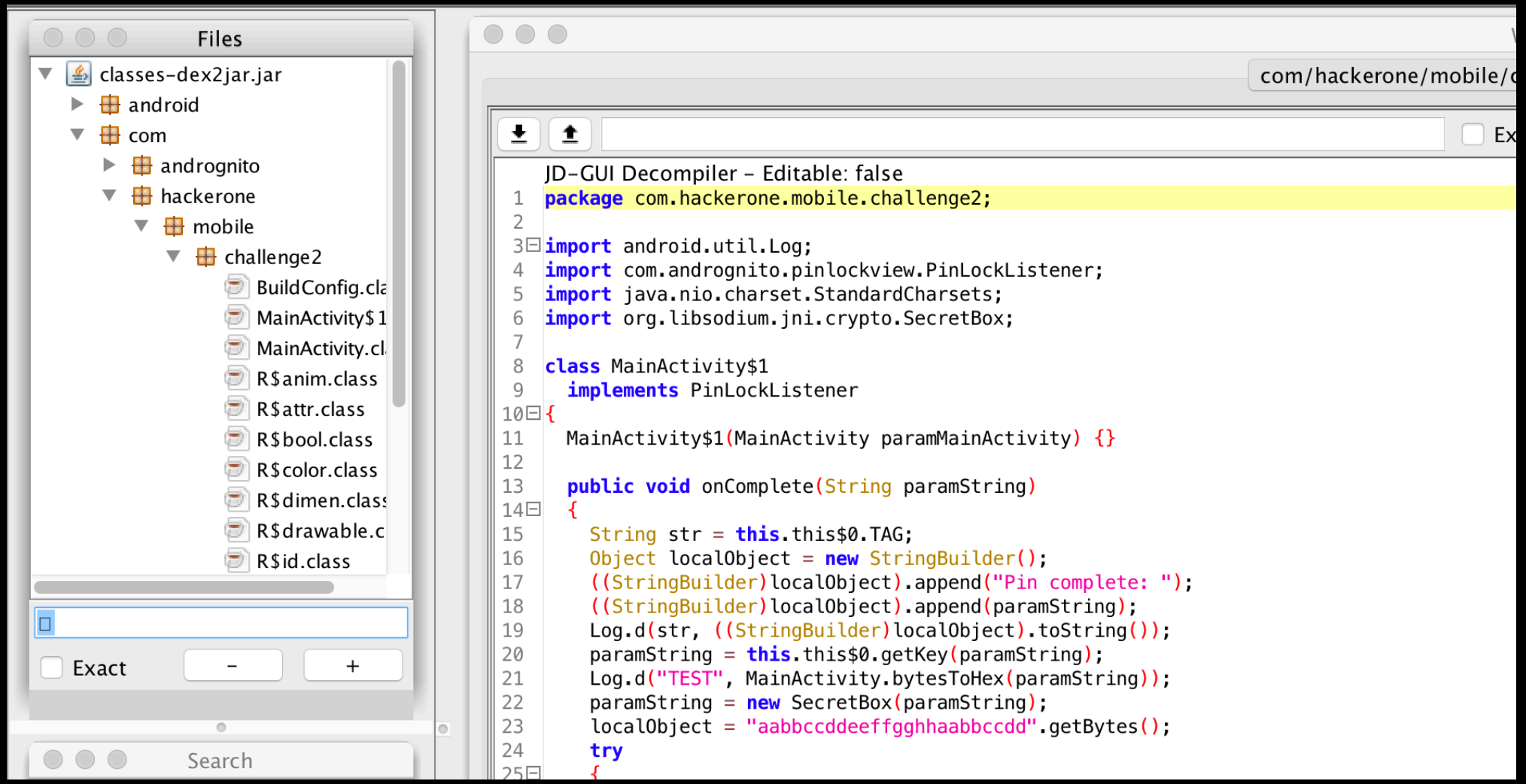
Solution

- The app requires modern android device, so we cannot use our virtual galaxy s6 – 5.1.0 – api 22 as usual
- I am going to use genymotion virtual Samsung Galaxy S7 – 7.1.0 – API 25 instead.
- So install, then open it, look like we have to input the correct pin to get flag:



Solution

- No clue, we have to reverse the apk file to read code
- Unzip it, use dex2jar to convert classes.dex to jar file, drag and drop the jar to Bytecode Viewer, we got the code:



Solution

- We see a lot of Log.d command, so first turn on logcat, then use the app to got an overview idea
- First pin number (1):

```
09-27 13:48:42.704  7691  7691 D PinLock : Pin changed, new length 1 with intermediate pin 1
```

- Second pin number (2):

```
09-27 13:48:42.704  7691  7691 D PinLock : Pin changed, new length 1 with intermediate pin 1
09-27 13:48:55.153  7691  7691 D PinLock : Pin changed, new length 2 with intermediate pin 12
09-27 13:48:55.483   265   570 D AudioFlinger: mixer(0xf0e03980) throttle end: throttle time(11)
```

- Third, Fourth, Fifth pin number (3…4…5):

```
09-27 13:48:42.704  7691  7691 D PinLock : Pin changed, new length 1 with intermediate pin 1
09-27 13:48:55.153  7691  7691 D PinLock : Pin changed, new length 2 with intermediate pin 12
09-27 13:48:55.483   265   570 D AudioFlinger: mixer(0xf0e03980) throttle end: throttle time(11)
09-27 13:49:11.286  7691  7691 D PinLock : Pin changed, new length 3 with intermediate pin 123
09-27 13:49:11.595   265   570 D AudioFlinger: mixer(0xf0e03980) throttle end: throttle time(11)
09-27 13:49:14.703  7691  7691 D PinLock : Pin changed, new length 4 with intermediate pin 1234
09-27 13:49:16.353  7691  7691 D PinLock : Pin changed, new length 5 with intermediate pin 12345
```

Solution

- Sixth pin number (6):

```
09-27 13:48:42.704  7691  7691 D PinLock : Pin changed, new length 1 with intermediate pin 1
09-27 13:48:55.153  7691  7691 D PinLock : Pin changed, new length 2 with intermediate pin 12
09-27 13:48:55.483   265   570 D AudioFlinger: mixer(0xf0e03980) throttle end: throttle time(11)
09-27 13:49:11.286  7691  7691 D PinLock : Pin changed, new length 3 with intermediate pin 123
09-27 13:49:11.595   265   570 D AudioFlinger: mixer(0xf0e03980) throttle end: throttle time(11)
09-27 13:49:14.703  7691  7691 D PinLock : Pin changed, new length 4 with intermediate pin 1234
09-27 13:49:16.353  7691  7691 D PinLock : Pin changed, new length 5 with intermediate pin 12345
09-27 13:49:24.337  7691  7691 D PinLock : Pin complete: 123456
09-27 13:49:24.337  7691  7691 D TEST    : 6C034108003A2DF4DC02C4C2D8D63CF2ACA689FE7D70E3891CA70C34A59CF28F
09-27 13:49:24.337  7691  7691 I org.libsodium.jni.NaCl: librarypath=/system/lib:/vendor/lib
09-27 13:49:24.338  7691  7691 D PROBLEM : Unable to decrypt text
09-27 13:49:24.338  7691  7691 W System.err: java.lang.RuntimeException: Decryption failed. Ciphertext failed verification
09-27 13:49:24.339  7691  7691 W System.err:    at org.libsodium.jni.crypto.Util.isValid(Util.java:47)
09-27 13:49:24.339  7691  7691 W System.err:    at org.libsodium.jni.crypto.SecretBox.decrypt(SecretBox.java:56)
09-27 13:49:24.339  7691  7691 W System.err:    at com.hackerone.mobile.challenge2.MainActivity$1.onComplete(MainActivity.java:42)
09-27 13:49:24.339  7691  7691 W System.err:    at com.andrognito.pinlockview.PinLockView$1.onNumberClicked(PinLockView.java:56)
09-27 13:49:24.339  7691  7691 W System.err:    at com.andrognito.pinlockview.PinLockAdapter$NumberViewHolder$1.onClick(PinLockAdapter.java:191)
09-27 13:49:24.339  7691  7691 W System.err:    at android.view.View.performClick(View.java:5637)
09-27 13:49:24.339  7691  7691 W System.err:    at android.view.View$PerformClick.run(View.java:22429)
09-27 13:49:24.339  7691  7691 W System.err:    at android.os.Handler.handleCallback(Handler.java:751)
09-27 13:49:24.339  7691  7691 W System.err:    at android.os.Handler.dispatchMessage(Handler.java:95)
09-27 13:49:24.339  7691  7691 W System.err:    at android.os.Looper.loop(Looper.java:154)
09-27 13:49:24.339  7691  7691 W System.err:    at android.app.ActivityThread.main(ActivityThread.java:6119)
09-27 13:49:24.339  7691  7691 W System.err:    at java.lang.reflect.Method.invoke(Native Method)
09-27 13:49:24.339  7691  7691 W System.err:    at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:886)
09-27 13:49:24.339  7691  7691 W System.err:    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:776)
```

Solution

- As you see, we can guess that if our pin length < 6, it print current pin
- If pin length = 6, its generate something (look like the key) and fire the problem

```
D PROBLEM : Unable to decrypt text
```

- Then print stack trace:

```
09-27 13:49:24.338  7691  7691 W System.err: java.lang.RuntimeException: Decryption failed. Ciphertext failed verification
09-27 13:49:24.339  7691  7691 W System.err:    at org.libsodium.jni.crypto.Util.isValid(Util.java:47)
09-27 13:49:24.339  7691  7691 W System.err:    at org.libsodium.jni.crypto.SecretBox.decrypt(SecretBox.java:56)
09-27 13:49:24.339  7691  7691 W System.err:    at com.hackerone.mobile.challenge2.MainActivity$1.onComplete(MainActivity.java:42)
09-27 13:49:24.339  7691  7691 W System.err:    at com.andrognito.pinlockview.PinLockView$1.onNumberClicked(PinLockView.java:56)
09-27 13:49:24.339  7691  7691 W System.err:    at com.andrognito.pinlockview.PinLockAdapter$NumberViewHolder$1.onClick(PinLockAdapter.java:191)
09-27 13:49:24.339  7691  7691 W System.err:    at android.view.View.performClick(View.java:5637)
09-27 13:49:24.339  7691  7691 W System.err:    at android.view.View$PerformClick.run(View.java:22429)
09-27 13:49:24.339  7691  7691 W System.err:    at android.os.Handler.handleCallback(Handler.java:751)
09-27 13:49:24.339  7691  7691 W System.err:    at android.os.Handler.dispatchMessage(Handler.java:95)
09-27 13:49:24.339  7691  7691 W System.err:    at android.os.Looper.loop(Looper.java:154)
09-27 13:49:24.339  7691  7691 W System.err:    at android.app.ActivityThread.main(ActivityThread.java:6119)
09-27 13:49:24.339  7691  7691 W System.err:    at java.lang.reflect.Method.invoke(Native Method)
09-27 13:49:24.339  7691  7691 W System.err:    at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:886)
09-27 13:49:24.339  7691  7691 W System.err:    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:776)
```

- Maybe “123456” is not the correct pin => “Ciphertext failed verification” => Error!

Solution

- Now we are going to review the code to know what's going on here
- The MainActivity.class:

```
10  public class MainActivity
11    extends AppCompatActivity
12  {
13    private static final char[] hexArray = "0123456789ABCDEF".toCharArray();
14    String TAG = "PinLock";
15    private byte[] cipherText;
16    IndicatorDots mIndicatorDots;
17    private PinLockListener mPinLockListener = new MainActivity.1(this);
18    PinLockView mPinLockView;
19
20    static
21    {
22      System.loadLibrary("native-lib");
23    }
24
25    public static String bytesToHex(byte[] paramArrayOfByte)
26    {
37
38    public native byte[] getKey(String paramString);
39
40    protected void onCreate(Bundle paramBundle)
41    {
42      super.onCreate(paramBundle);
43      setContentView(2131296283);
44      this.cipherText = new Hex().decode("9646D13EC8F8617D1CEA1CF4334940824C700ADF6A7A3236163CA2C9604B9BE4BDE770AD698C02070F571A0B612BBD3572D81F99")
45      this.mPinLockView = ((PinLockView)findViewById(2131165263));
46      this.mPinLockView.setPinLockListener(this.mPinLockListener);
47      this.mIndicatorDots = ((IndicatorDots)findViewById(2131165241));
48      this.mPinLockView.attachIndicatorDots(this.mIndicatorDots);
49    }
50
51    public native void resetCoolDown();
52  }
53
```

Solution

- onCreate method will set the view and initialize the variable:

```
protected void onCreate(Bundle paramBundle)
{
  super.onCreate(paramBundle);
  setContentView(2131296283);
  this.cipherText = new Hex().decode("9646D13EC8F8617D1CEA1CF4334940824C700ADF6A7A3236163CA2C9604B9BE4BDE770AD698C02070F571A0B612BBD3572D81F99");
  this.mPinLockView = ((PinLockView)findViewById(2131165263));
  this.mPinLockView.setPinLockListener(this.mPinLockListener);
  this.mIndicatorDots = ((IndicatorDots)findViewById(2131165241));
  this.mPinLockView.attachIndicatorDots(this.mIndicatorDots);
}
```

- We can see something interesting here, the cipherText is take the value from the result of Hex().decode, keep it in mind ☺
- Then it use PinLockListener to perform the pin check ui
- In MainActivity$1.class, we can see:

```
public void onPinChange(int paramInt, String paramString)
{
  String str = this.this$0.TAG;
  StringBuilder localStringBuilder = new StringBuilder();
  localStringBuilder.append("Pin changed, new length ");
  localStringBuilder.append(paramInt);
  localStringBuilder.append(" with intermediate pin ");
  localStringBuilder.append(paramString);
  Log.d(str, localStringBuilder.toString());
}
```

Solution

- And:

```
public void onComplete(String paramString)
{
  String str = this.this$0.TAG;
  Object localObject = new StringBuilder();
  ((StringBuilder)localObject).append("Pin complete: ");
  ((StringBuilder)localObject).append(paramString);
  Log.d(str, ((StringBuilder)localObject).toString());
  paramString = this.this$0.getKey(paramString);
  Log.d("TEST", MainActivity.bytesToHex(paramString));
  paramString = new SecretBox(paramString);
  localObject = "aabbccddeeffgghhaabbccdd".getBytes();
  try
  {
    localObject = paramString.decrypt((byte[])localObject, MainActivity.access
    paramString = new java/lang/String;
    paramString.<init>((byte[])localObject, StandardCharsets.UTF_8);
    Log.d("DECRYPTED", paramString);
  }
  catch (RuntimeException paramString)
  {
    Log.d("PROBLEM", "Unable to decrypt text");
    paramString.printStackTrace();
  }
}
```

- onPinChange and onComplete act like the idea we guess based on logcat, we focus on onComplete to see what happen when we input 6-length pin

Solution

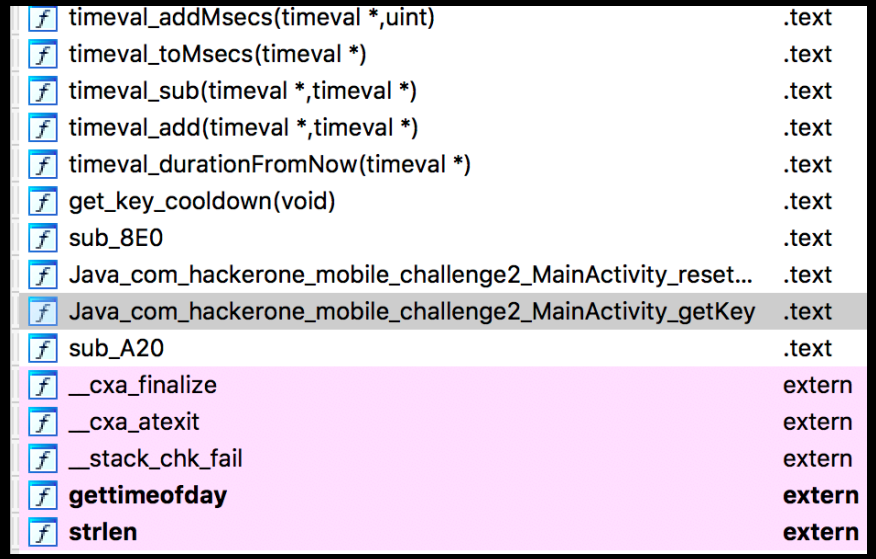
- After Log "Pin complete: XXXXXX", the pin is passed to getKey(string) function, which is a native function:

```
public native byte[] getKey(String paramString);
```

- Android provides developers to create C/C++ binaries and load the functions from it if present inside the jniLibs/ directory. Hence, it loads getKey() native function to our application
- Come to lib/x86/ folder, we found the lib:

```
 ~/Desktop/mobile/h1702-2018/lib/x86/ ls
libnative-lib.so
```

- Open it in IDA:



Solution

- Reverse the getKey function:

```
1  int __cdecl Java_com_hackerone_mobile_challenge2_MainActivity_getKey(int a1, int a2, int a3, __int16 a4, int a5,
2  {
3    const char *v12; // esi
4    struct timeval tv; // [esp+10h] [ebp-4Ch]
5    int savedregs; // [esp+5Ch] [ebp+0h]
6
7    v12 = (const char *)(*(int (__cdecl **)(int, int, _DWORD))(*(_DWORD *)a1 + 676))(a1, a3, 0);
8    gettimeofday(&tv, 0);
9    tv.tv_usec %= 1000000;
10   JUMPOUT(dword_2004, 51, &loc_A79);
11   return sub_A20((int)&savedregs, tv.tv_usec, v12, a1, a2, (int *)a3, a4, a5, a6, a7, a8, a9, a10, a11, a12);
12 }
```

- This function seems calculate some value based on gettimeofday(), then pass it and our 6 digits pin to sub_A20, so we check it.

- sub_A20 is very long, and I gave up in reverse it, but since it performed many calculate code, we know it will get back to us the key of 6 digits pin input, the program will use the key & nonce to decrypt the cipherText

```
try
{
  localObject = paramString.decrypt((byte[])localObject, MainActivity.access$000(this.this$0));
  paramString = new java/lang/String;
  paramString.<init>((byte[])localObject, StandardCharsets.UTF_8);
  Log.d("DECRYPTED", paramString);
}
```

Solution

- What we got so far:

```
1  cipherText='9646D13EC8F8617D1CEA1CF4334940824C700ADF6A7A3236163CA2C9604B9BE4BDE770AD698C02070F571A0B612BBD3572D81F99'.decode('hex')
2  nonce='aabbccddeeffgghhaabbccdd'
3  key=getKey(string)
```

- So the idea now is, use frida to hook the getKey(pin), then let it do its job, pin is brute force from 999999 -> 000000

Solution

- Here the code:

```
hook_script="""
function rpad(width, string, padding)
{
    return (width <= string.length) ? string : rpad(width, padding + string, padding)
}

function genPin(pin)
{
    return rpad(6, pin.toString(), '0')
}

Java.perform
(
    function ()
    {
        console.log("inside hook script");
        nonce = Java.array('byte',[ 97, 97, 98, 98, 99, 99, 100, 100, 101, 101, 102, 102, 103, 103, 104, 104, 97, 97, 98, 98, 99, 99, 100, 100]);
        cipherText = Java.array('byte',[ 150, 70, 209, 62, 200, 248, 97, 125, 28, 234, 28, 244, 51, 73, 64, 130, 76, 112, 10, 223, 106, 122, 50, 54,
        Java.choose("com.hackerone.mobile.challenge2.MainActivity",{
            "onMatch":function(instance)
            {
                    console.log("[*] Instace found: "+instance);
                    counter = 0;
                    for(var i = 999999; i >= 0; i--)
                    {
                        pin = genPin(i.toString());
                        key = instance.getKey(pin);
                        console.log("Pin: " + pin + "- Key: " + instance.bytesToHex(key));
                    }
            },
            "onComplete":function()
            {
                console.log("Done");
            }
        });
    }
);
"""
```

Solution

- Let check:

```
Pin: 999955- Key: 00000000000000000B1D4D900B1D4D9ACA689FEACA689FEAC175D27AC175D27
Pin: 999954- Key: 00000000000000000B1D4D909B1825CACA689FEA5A6DF7BAC175D27AC175D27
Pin: 999953- Key: 00000000000000000B1D4D9DCB3101BACA689FE70A44D3CAC175D27AC175D27
Pin: 999952- Key: 00000000000000000B1D4D9098BAFA8ACA689FEA59CF28FAC175D27AC175D27
Pin: 999951- Key: 00000000000000000B1D4D9B0B05113ACA689FE1CA70C34AC175D27AC175D27
Pin: 999950- Key: 00000000000000000B1D4D900B1D4D9ACA689FEACA689FEAC175D27AC175D27
Pin: 999949- Key: 00000000000000009B1825C00000000A5A6DF7BAC175D27AC175D27AC175D27
```

- It sleep at 999949, then run next, and for some reason, the program timeout kill us

```
Pin: 999855- Key: 00000000499B77D800B1D4D9492AA301ACA689FEACA689FEAC175D27AC175D27
Pin: 999854- Key: 00000000499B77D800B1D4D9402AF584ACA689FEA5A6DF7BAC175D27AC175D27
Pin: 999853- Key: 00000000499B77D800B1D4D9952867C3ACA689FE70A44D3CAC175D27AC175D27
Pin: 999852- Key: 00000000499B77D800B1D4D94010D870ACA689FEA59CF28FAC175D27AC175D27
Pin: 999851- Key: 00000000499B77D800B1D4D9F92B26CBACA689FE1CA70C34AC175D27AC175D27
Pin: 999850- Key: 00000000499B77D800B1D4D9492AA301ACA689FEACA689FEAC175D27AC175D27
Pin: 999849- Key: 00000000499B77D809B1825C499B77D8A5A6DF7BAC175D27AC175D27AC175D27
Pin: 999848- Key: 00000000499B77D809B1825C00000000A5A6DF7BE58C2AFFAC175D27AC175D27
Pin: 999847- Key: 00000000499B77D809B1825C2DE2E6E5A5A6DF7BC86ECC1AAC175D27AC175D27
Pin: 999846- Key: 00000000499B77D809B1825C98FCC976A5A6DF7B7D70E389AC175D27AC175D27
Pin: 999845- Key: 00000000499B77D809B1825C492AA301A5A6DF7BACA689FEAC175D27AC175D27
Traceback (most recent call last):
  File "/Users/tsug0d/Desktop/mobile/lab1_frida.py", line 55, in <module>
    script.load()
  File "/usr/local/lib/python2.7/site-packages/frida/core.py", line 192, in load
    self._impl.load()
frida.TransportError: timeout was reached
```

Solution

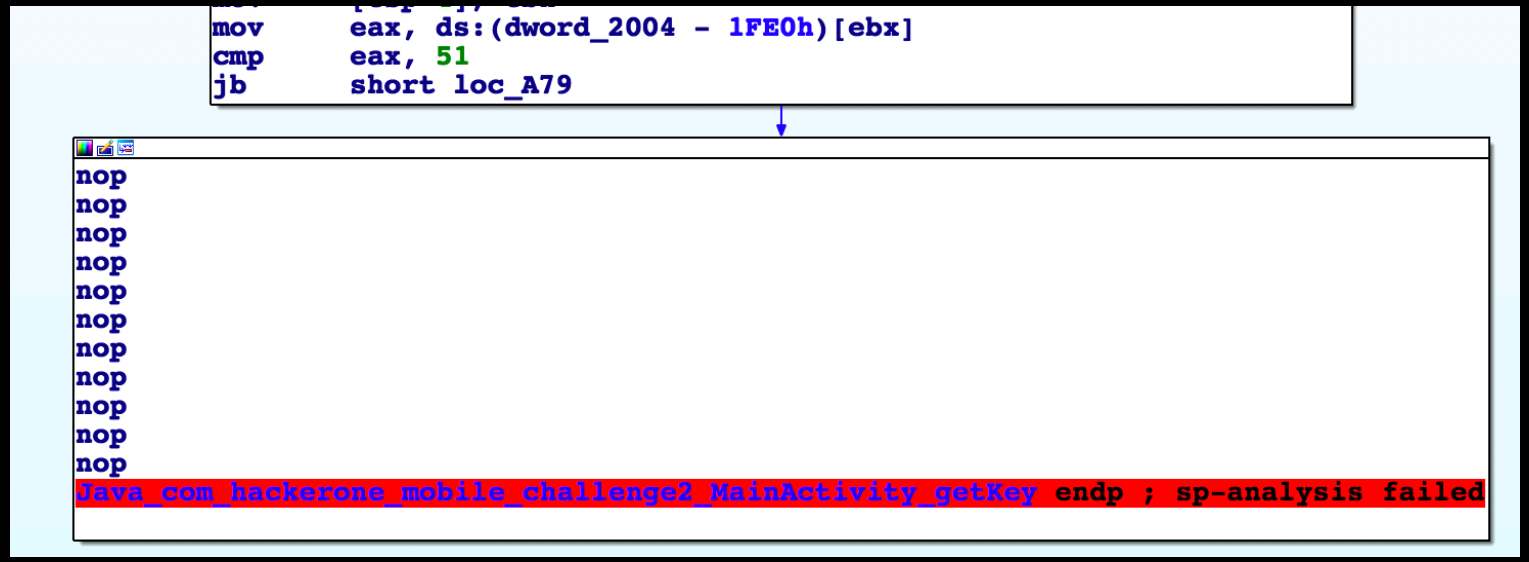
- This is frida timeout, to avoid it, we wrap our hook script with setTimeout() function

```
setTimeout(function  ()
{
        function x();
        function y();
        Java.perform(function()
                {
                //blah blah;
                }
        )
}, 0);
```

- But, if we pause for each 51 times, the brute-force process is very slow, and it's not good to hack the program like this ☺

Solution

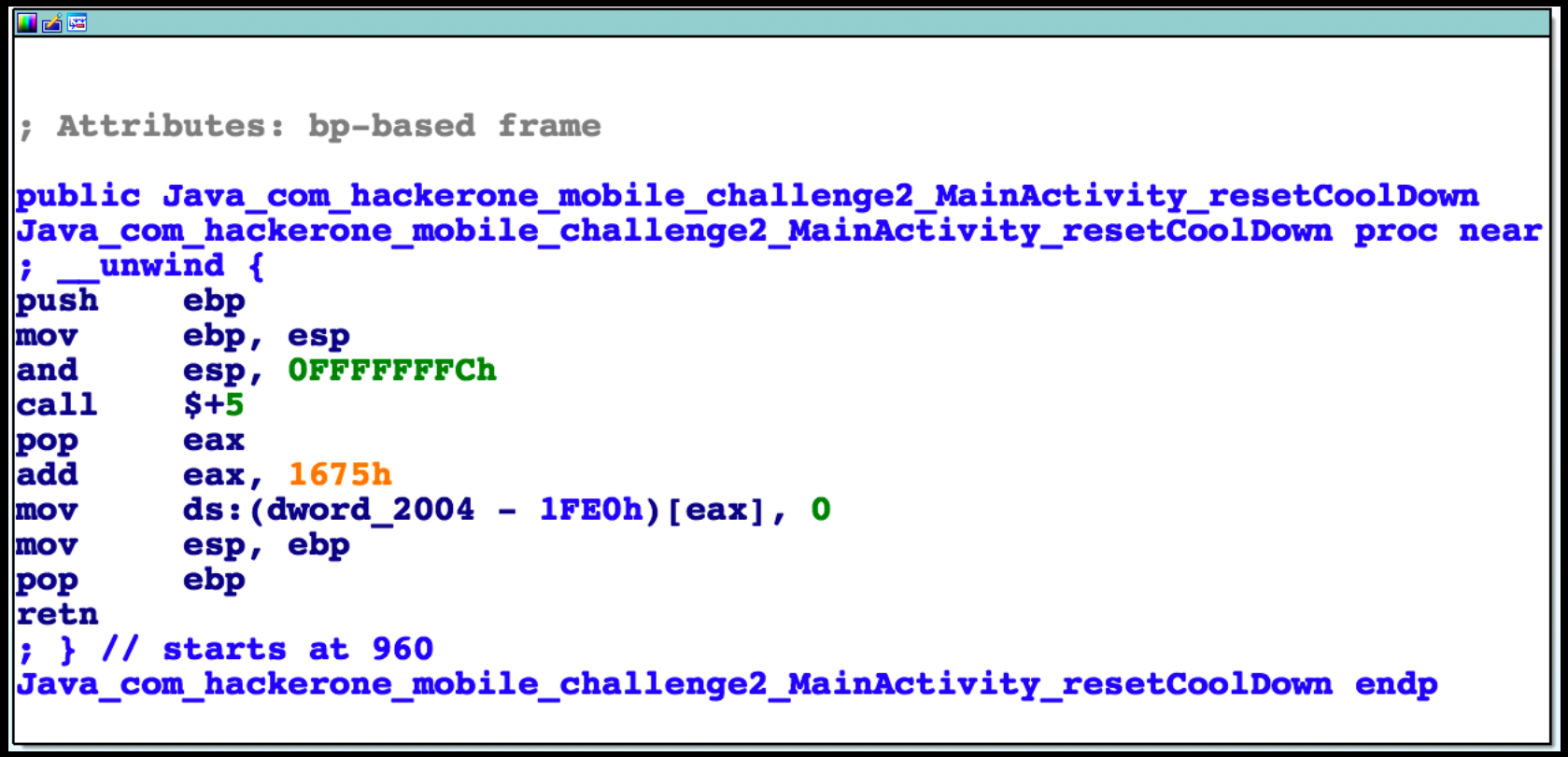
- We can figure it in Java_com_hackerone_mobile_challenge2_MainActivity_getKey



- It move the count value into eax, then compare it with 51, if below, jump to loc_A79, else fail, so that why it stop at 999949
- It obviously prevent us to bruteforce the pin, so we stuck? There is one function that we haven't looked yet!

Solution

- It is Java_com_hackerone_mobile_challenge2_MainActivity_resetCoolDown()



```
; Attributes: bp-based frame

public Java_com_hackerone_mobile_challenge2_MainActivity_resetCoolDown
Java_com_hackerone_mobile_challenge2_MainActivity_resetCoolDown proc near
; __unwind {
push    ebp
mov     ebp, esp
and     esp, 0FFFFFFFCh
call    $+5
pop     eax
add     eax, 1675h
mov     ds:(dword_2004 - 1FE0h)[eax], 0
mov     esp, ebp
pop     ebp
retn
; } // starts at 960
Java_com_hackerone_mobile_challenge2_MainActivity_resetCoolDown endp
```

- You can see it move 0 to the counter, so it reset the value of counter (or just look at its name ☺), by using it, we never reach the branch greater than 51

Solution

- So we define counter=0, increase it each loop, if it equal to 50, call resetCoolDown() to reset, we avoid!

```
32          Java.choose("com.hackerone.mobile.challenge2.MainActivity",{
33              "onMatch":function(instance)
34              {
35                      console.log("[*] Instace found: "+instance);
36                      counter = 0;
37                      for(var i = 999999; i >= 0; i--)
38                      {
39                          pin = genPin(i.toString());
40                          key = instance.getKey(pin);
41                          console.log("Pin: " + pin + "- Key: " + instance.bytesToHex(key));
42                          counter+=1;
43                          if(counter==50)
44                              {
45                              instance.resetCoolDown();
46                              counter=0;
47                              }
48                      }
49              },
50              "onComplete":function()
51              {
52                  console.log("Done");
53              }
54          });
```

Solution

- Run a script, we are facing new problem, the result is stop at 949284 (look at the image)

```
Pin: 949288 - Key: 00000000003A2DF4499B77D84010D870E58C2AFFE58C2AFFAC175D27A5A6DF7B
Pin: 949287 - Key: 00000000003A2DF4499B77D86DF23E95E58C2AFFC86ECC1AAC175D27A5A6DF7B
Pin: 949286 - Key: 00000000003A2DF4499B77D8D8EC1106E58C2AFF7D70E389AC175D27A5A6DF7B
Pin: 949285 - Key: 00000000003A2DF4499B77D8093A7B71E58C2AFFACA689FEAC175D27A5A6DF7B
Pin: 949284 - Key: 00000000003A2DF4499B77D8003A2DF4E58C2AFFA5A6DF7BAC175D27A5A6DF7B
```

- In logcat, we found:

```
JNI ERROR (app bug): global reference table overflow (max=51200)
global reference table dump:
  Last 10 entries (of 51200):
    51199: 0x130aa0a0 byte[] (32 elements)
    51198: 0x130aa070 byte[] (32 elements)
    51197: 0x1309afd0 byte[] (32 elements)
    51196: 0x1309afa0 byte[] (32 elements)
    51195: 0x1309af70 byte[] (32 elements)
    51194: 0x1309af40 byte[] (32 elements)
    51193: 0x1309af10 byte[] (32 elements)
    51192: 0x1309aee0 byte[] (32 elements)
    51191: 0x1309aeb0 byte[] (32 elements)
    51190: 0x1309ae80 byte[] (32 elements)
```

- Maybe using garbage collection solve this, but I am unable to do => We only can brute-force around ~50000 pin, so in a very bad case (assume that the pin in 000000->050000), we have to run script 1000000/50000=20 times

Solution

- We are going to define the signal to stop our script if the pin is correct
- The decryption phase is as below:

```
paramString = this.this$0.getKey(paramString);
```

```
paramString = new SecretBox(paramString);
```

```
localObject = paramString.decrypt((byte[])localObject, MainActivity.access$000(this.this$0));
```

- It's using SecretBox class of libsodium, so we use frida to create this class instance, then call the decrypt method

```
secretBoxClass = Java.use("org.libsodium.jni.crypto.SecretBox");
decrypt_result = secretBoxClass.$new(key).decrypt(nonce,cipherText);
```

- Note that if the decryption fail, the script is stop, we have to include it in try – catch – finally statement
- The Idea is, we init the variable flag = false, when the decryption success, we turn it to true and break the loop

Solution

- Here the code:

```
flag = false;
secretBoxClass = Java.use("org.libsodium.jni.crypto.SecretBox");
Java.choose("com.hackerone.mobile.challenge2.MainActivity",{
    "onMatch":function(instance)
    {
            console.log("[*] Instace found: "+instance);
            counter = 0;
            for(var i = 930000; i >= 0; i--)
            {
                pin = genPin(i.toString());
                key = instance.getKey(pin);
                console.log("Pin: " + pin + " - Key: " + instance.bytesToHex(key));

                try
                {
                    decrypt_result = secretBoxClass.$new(key).decrypt(nonce,cipherText);
                    flag = true;
                }
                catch(err)
                {
                    //Do nothing
                }
                finally
                {
                    if ( flag == true )
                    {
                        console.log("Found");
                        console.log("Pin: "+ pin );
                        break;
                    }
                }
```

Solution

- Run the script, we found the correct pin:

```
Pin: 918278 - Key: 499B77D8B93BFEBB2DE2E6E54010D870C86ECC1AE58C2AFFAC175D271CA70C34
Pin: 918277 - Key: 499B77D8B93BFEBB2DE2E6E56DF23E95C86ECC1AC86ECC1AAC175D271CA70C34
Pin: 918276 - Key: 499B77D8B93BFEBB2DE2E6E5D8EC1106C86ECC1A7D70E389AC175D271CA70C34
Pin: 918275 - Key: 499B77D8B93BFEBB2DE2E6E5093A7B71C86ECC1AACA689FEAC175D271CA70C34
Pin: 918274 - Key: 499B77D8B93BFEBB2DE2E6E5003A2DF4C86ECC1AA5A6DF7BAC175D271CA70C34
Pin: 918273 - Key: 499B77D8B93BFEBB2DE2E6E5D538BFB3C86ECC1A70A44D3CAC175D271CA70C34
Pin: 918272 - Key: 499B77D8B93BFEBB2DE2E6E500000000C86ECC1AA59CF28FAC175D271CA70C34
Pin: 918271 - Key: 499B77D8B93BFEBB2DE2E6E5B93BFEBBC86ECC1A1CA70C34AC175D271CA70C34
Pin: 918270 - Key: 499B77D8B93BFEBB2DE2E6E56DF23E95C86ECC1AC86ECC1AAC175D271CA70C34
Pin: 918269 - Key: 499B77D8B93BFEBB98FCC976098BAFA87D70E389AC175D27AC175D271CA70C34
Pin: 918268 - Key: 499B77D8B93BFEBB98FCC9764010D8707D70E389E58C2AFFAC175D271CA70C34
Pin: 918267 - Key: 499B77D8B93BFEBB98FCC9766DF23E957D70E389C86ECC1AAC175D271CA70C34
Pin: 918266 - Key: 499B77D8B93BFEBB98FCC976D8EC11067D70E3897D70E389AC175D271CA70C34
Pin: 918265 - Key: 499B77D8B93BFEBB98FCC976093A7B717D70E389ACA689FEAC175D271CA70C34
Pin: 918264 - Key: 499B77D8B93BFEBB98FCC976003A2DF47D70E389A5A6DF7BAC175D271CA70C34
Found
Pin: 918264
```

- Turn on logcat, input the pin found, we got flag:

```
09-27 20:03:30.902 11681 11681 D PinLock : Pin complete: 918264
09-27 20:03:30.903 11681 11681 D TEST    : 499B77D8B93BFEBB98FCC976003A2DF47D70E389A5A6DF7BAC175D271CA70C34
09-27 20:03:30.903 11681 11681 I org.libsodium.jni.NaCl: librarypath=/system/lib:/vendor/lib
09-27 20:03:30.904 11681 11681 D DECRYPTED: flag{wow_yall_called_a_lot_of_func$}
```

Bonus Solution

- Remember the resetCoolDown() function? Why the hell on earth got this ?? :D ??, I think it should not been in the code ☺
- Assume that if we fail, app exit, so the resetCoolDown() is not necessary here ☺
- We are going to solve this challenge without resetCoolDown() !

Bonus Solution

- As I said above, if the counter reach 51, the program is pausing, so the task is obvious: Make counter not reach 51 (of course without resetCoolDown() function)
- Frida provide the method to change the memory dynamically (more like enter god-mode ☺)
- So the first things is get the base address of the loaded libnative-lib library using Module.findBaseAddress

```
libnative = Module.findBaseAddress("libnative-lib.so");
```

- `Module.findBaseAddress(name)`: returns the base address of the `name` module, or `null` if the module isn't loaded

Bonus Solution

- Locate the variable relative to the libraries base address, so now, we have to find address of the counter
- In IDA, from Java_com_hackerone_mobile_challenge2_MainActivity_getKey:

```
int __cdecl Java_com_hackerone_mobile_challenge2_MainActivity_getKey(int a1, int a2, int a3, __int16 a4, int a
{
  const char *v12; // esi
  struct timeval tv; // [esp+10h] [ebp-4Ch]
  int savedregs; // [esp+5Ch] [ebp+0h]

  v12 = (const char *)(*(int (__cdecl **)(int, int, _DWORD))(*(_DWORD *)a1 + 676))(a1, a3, 0);
  gettimeofday(&tv, 0);
  tv.tv_usec %= 1000000;
  JUMPOUT(dword_2004, 51, &loc_A79);
  return sub_A20((int)&savedregs, tv.tv_usec, v12, a1, a2, (int *)a3, a4, a5, a6, a7, a8, a9, a10, a11, a12);
}
```

- We can see that the JUMPOUT compare value at dword_2004 with 51, so we know that is our variable we need, let check:

```
.bss:00002004 dword_2004      dd ?                    ; DATA XREF: LOAD:00000248↑o
.bss:00002004                                         ; LOAD:00000268↑o ...
.bss:00002004 _bss            ends
.bss:00002004
```

Bonus Solution

- virtual address = base address + offset, we calculate it:

```
counter_address = libnative.add(ptr("0x00002004"));
```

- Now, set the variable to 1 (which is below 51)

```
Memory.writeInt(counter_address,1);
```

- `Memory.writeS8(address, value)`, `Memory.writeU8(address, value)`, `Memory.writeS16(address, value)`, `Memory.writeU16(address, value)`, `Memory.writeS32(address, value)`, `Memory.writeU32(address, value)`, `Memory.writeShort(address, value)`, `Memory.writeUShort(address, value)`, `Memory.writeInt(address, value)`, `Memory.writeUInt(address, value)`, `Memory.writeFloat(address, value)`, `Memory.writeDouble(address, value)`: write the number `value` to the signed or unsigned 8/16/32/etc. or float/double value at `address`. A JavaScript exception will be thrown if `address` isn't writable.

Bonus Solution

- Putting it all together, here the full PoC script:
https://github.com/tsug0d/AndroidMobilePentest101/blob/master/lab/frida_lab/lab_frida_bonus.py