Category: Rev

Challenge: Not That Easy

Writeup:

This is an android reverse challenge. As usual, we are opening it with jadx-gui and installing it to the phone/ emulator. In this challenge there is no need to proxy connection but always establish it even if you do not need. You cannot know when you need that. I am using brupsuite as proxy server.

In jadx I am directly browsing to MainActivity class. We are lucky because the code is not obfuscated. After reading and analyzing the code I found the following.

There is a button on the main activity and if I click on it, it is decrypting a data and writing it to a file called flag. At that point I use one of the following solutions.

Solution 1 (using adb):

I can browse into /data/data/com.suctf.notthateasy/files . At that location there is a file named flag (as expected). If I read it with cat the flag is there:

```
root@vbox86p:/data/data/com.suctf.notthateasy/files # ls
flag
at flag
SUCTF{!_N3vEr_G1v3UP_!!}root@vbox86p:/data/data/com.suctf.notthateasy/files #
```

Solution 2 (decrypting data):

before writing calling writeToFile function, it is decrypting the flag with a function in the cip class named decrypt. If I analyze cip class, I can see it is making AES decryption.

The encrypted data: 3zrk8gCAkLeIOv9Vak30Oyj5xT7lAk5ZC66CMskgTOs=

AES key: SuCTF

As you can see the key size is too slow for online tools. They are not accepting it as key. So we can copy the cip class and use it to decrypt data

Main class:

```
public class Main
{
    public static void main(String []args){
        String flag =
cip.decrypt("3zrk8gCAkLeIOv9Vak30Oyj5xT7lAk5ZC66CMskgTOs=", "SuCTF");
        System.out.println(flag); //output: SUCTF{!_N3vEr_G1v3UP_!!}
    }
}
```

cip class:

```
import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;

import java.io.PrintStream;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class cip {
    private static byte[] key;
    private static SecretKeySpec secretKey;

    public static void setKey(String myKey) {
        try {
            key = myKey.getBytes("UTF-8");
            byte[] digest = MessageDigest.getInstance("SHA-1").digest(key);
            key = digest;
            key = Arrays.copyOf(digest, 16);
            secretKey = new SecretKeySpec(key, "AES");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e2) {
            e2.printStackTrace();
        }
    }

    public static String decrypt(String strToDecrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(2, secretKey);
            return new String(cipher.doFinal(Base64.decode(strToDecrypt)));
        } catch (Exception e) {
            PrintStream printStream = System.out;
            printStream.println("Error while decrypting: " + e.toString());
            return null;
        }
    }
}
```

Solution 3 (using Inspeckage):

During the flag generation process, inspeckage captures the decrypted data.

Solution 4 (using Frida):

By hooking decrypt function, you can get the flag.

Frida hook function

```
Java.perform(function(){
    var c1 = Java.use("com.suctf.notthateasy.cip");
    c1.decrypt.implementation = function(a,b){
        var flag = this.decrypt(a,b);
        console.log(flag);
        return flag;
    }
})
```

Frida result:

```
[Televole::com.suctf.notthateasy]-> SUCTF{!_N3vEr_G1v3UP_!!}
```