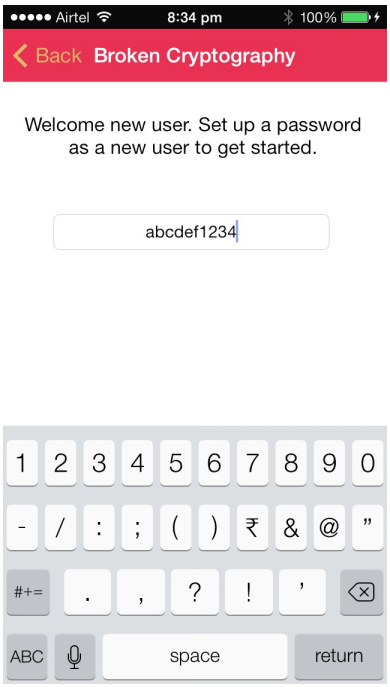
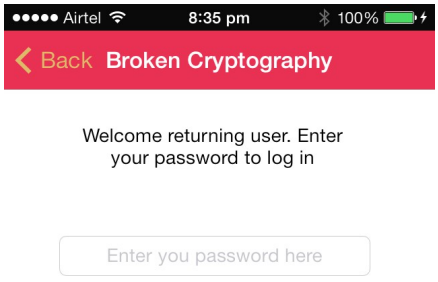
Damn Vulnerable IOS Application Solutions
http://damnvulnerableiosapp.com/

Broken Cryptography

Let's start by setting up a password. In this case, let's set the password to *abcdef1234*



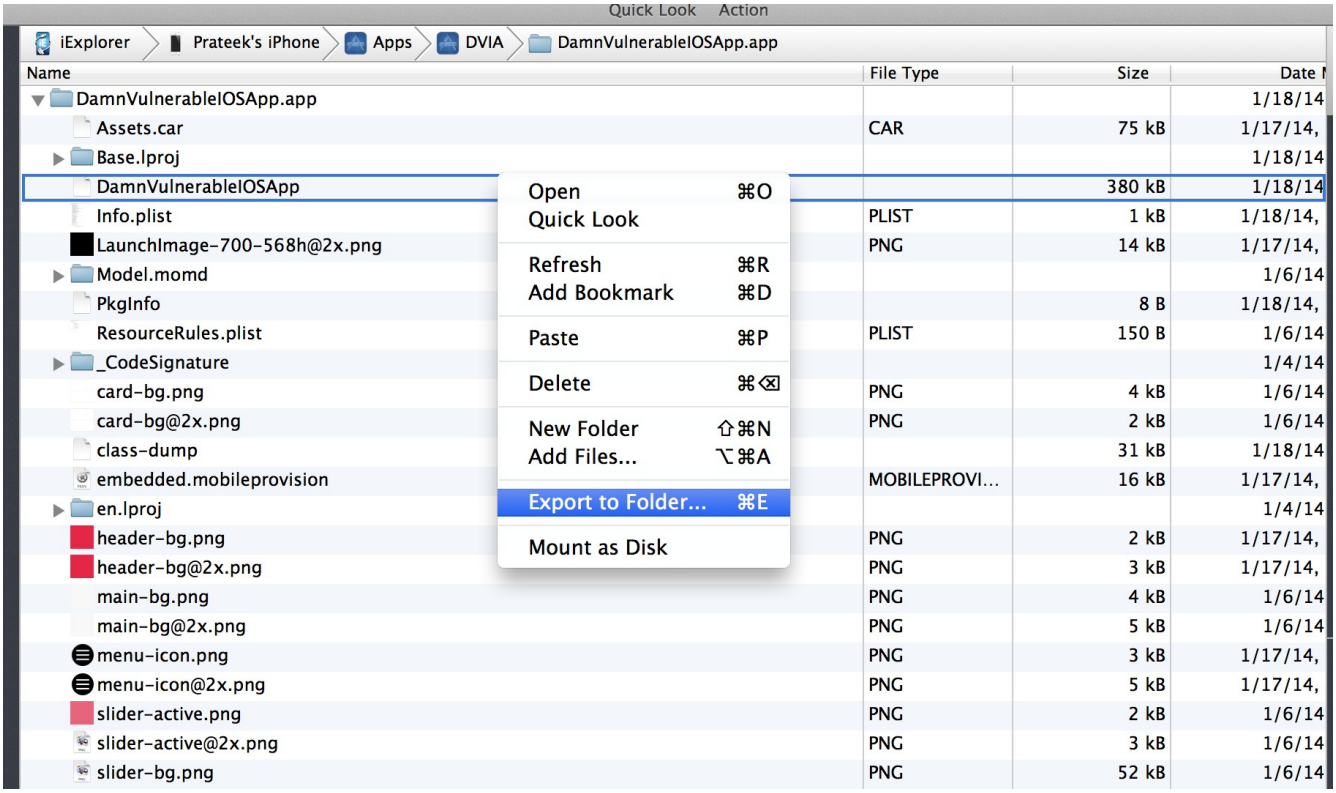
Now we are shown the returning user page. Let's assume that we don't know the password and want to know it somehow.



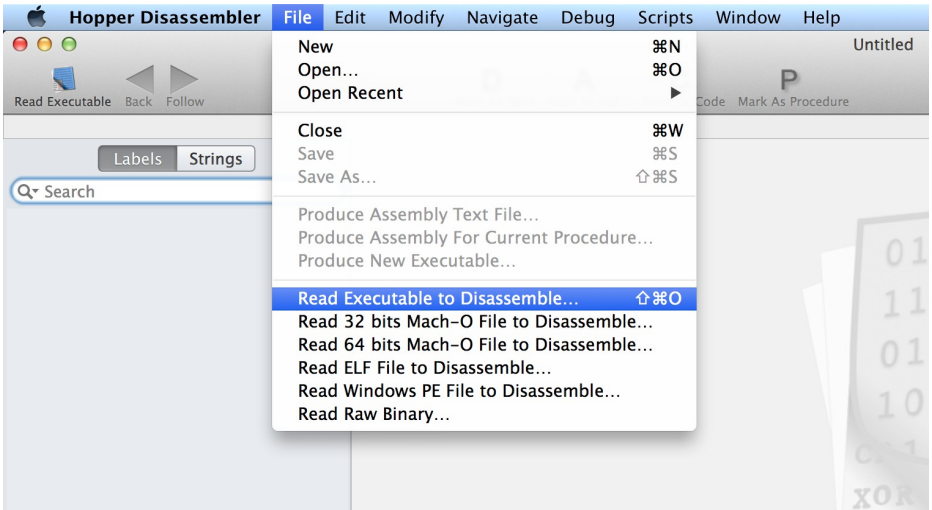
Lets export the application binary onto our system so we can analyze it using Hopper. If you don't know what is Hopper, I would recommend you to have a look here

http://highaltitudehacks.com/2014/01/17/ios-application-security-part-28-patching-ios-application-with-hopper

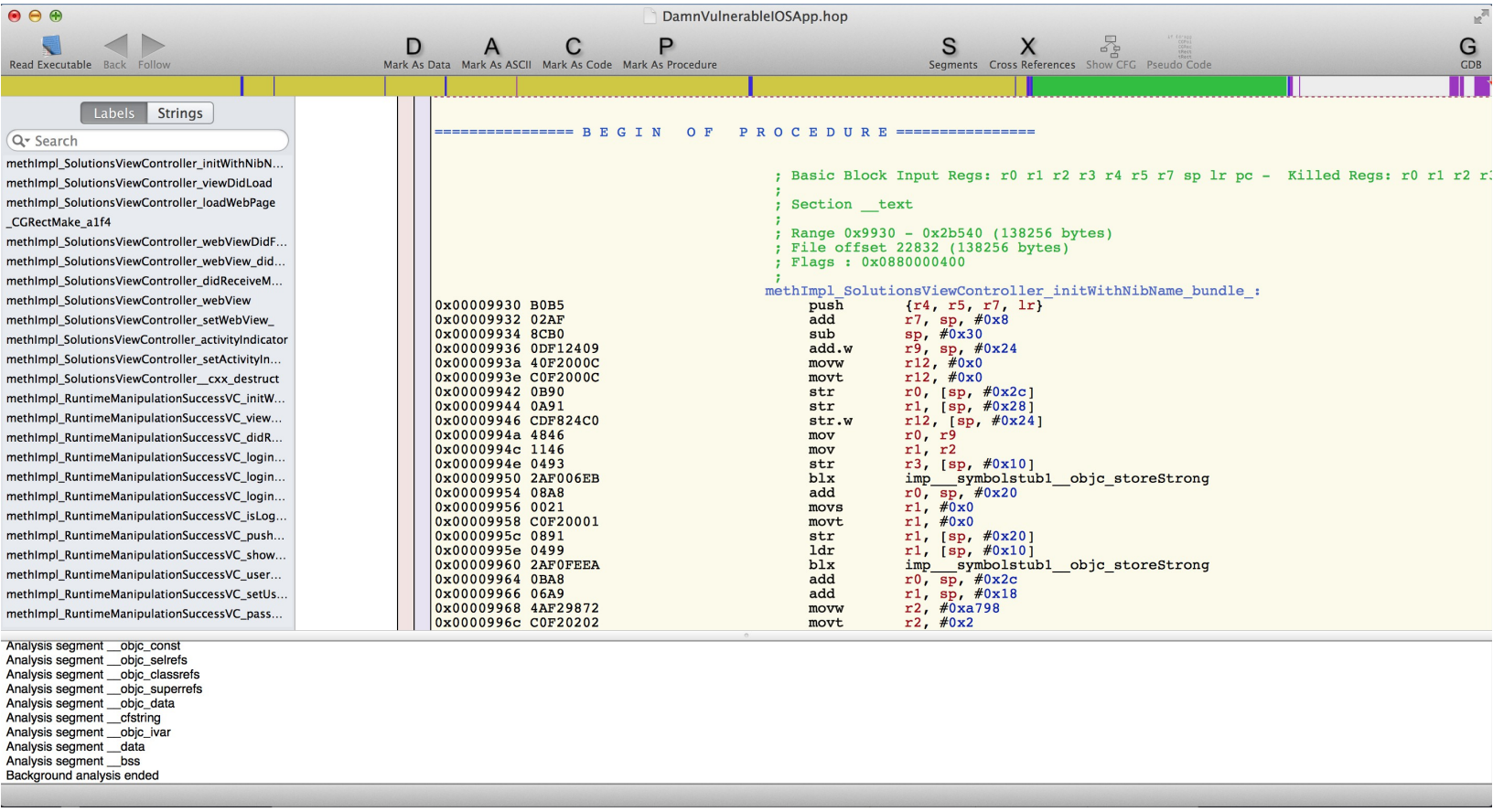
Go to the application folder using iExplorer, then right-click on the application binary and select the option *Export To Folder* , and save it somewhere on your computer.



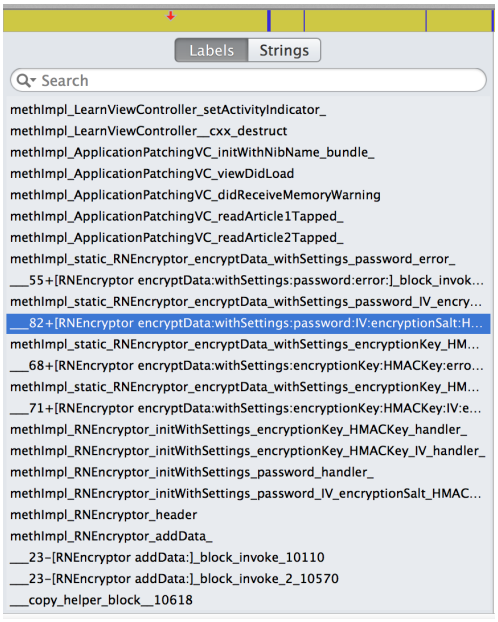
Now open Hopper and select the option *File->Read Executable To Disassemble*



Give the Binary that we just exported to disassemble to Hopper. Hopper will start disassembling the binary and produce an output like this...

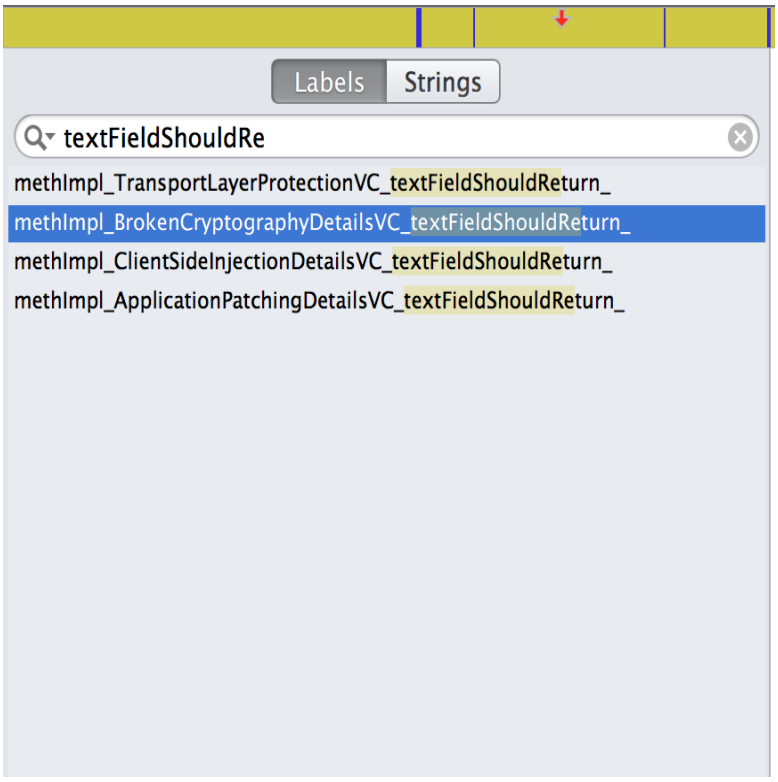


On the left side, you can see stuff like RNEncryptor etc. Looks like this class is being used to encrypt data.



On googlng *RNEncryptor*, we can note that it is a open source Github class used for encrypting data. Anyways, our task is to find a weakness in the encryption being used to find the actual password. So let's search for the method that gets called to perform the authentication check. Let's go back to the app. We notice that there is no login button to tap on the authentication page in the app. The login check happens once you tap on the *return* button in the text field.

And if you are a bit familiar with IOS development you should know that if the current view controller is a delegate of this text field, then the method that will be called is *-(BOOL)textFieldShouldReturn: (UITextField *)textField*. So lets search for this method in the labels section.



Of course, there are more implementations of the same method but we are interested in the one for the view controller BrokenCryptographyDetailsVC which is shown highlighted in the image above. Let's select on it and see its disassembly on the right side. However, another cool feature is checking out the pseudo code for this function. Let's click on the *Pseudo Code* button on the top right to check out the Pseudo code for this function.

DamnVulnerableIOSApp.hop

```
function methImpl_BrokenCryptographyDetailsVC_textFieldShouldReturn_ {
    r7 = &var_12;
    var_m4 = r8;
    r13 = r13 - 0x1f8;
    var_496 = r0;
    var_492 = r1;
    var_488 = 0x0;
    objc_storeStrong(&var_488, r2);
    NSSearchPathForDirectoriesInDomains(0x9, 0x1, SIGN_EXTEND(0x1));
    r7 = r7;
    r0 = objc_retainAutoreleasedReturnValue();
    var_440 = r0;
    var_436 = *imp___nl_symbol_ptr__objc_msgSend;
    (*0x39618)(r0, *0x39618, 0x0, var_436);
    r7 = r7;
    r0 = objc_retainAutoreleasedReturnValue();
    var_432 = r0;
    var_428 = @"/secret-data";
    var_424 = *imp___nl_symbol_ptr__objc_msgSend;
    (var_424)();
    r7 = r7;
    r0 = objc_retainAutoreleasedReturnValue();
    var_484 = r0;
    [var_432 release];
    [var_440 release];
    r2 = var_488;
    var_420 = *imp___nl_symbol_ptr__objc_msgSend;
    var_416 = r2;
    (*0x3954c)(var_496, *0x39470, r2, var_420);
    r7 = r7;
    r0 = objc_retainAutoreleasedReturnValue();
    var_412 = r0;
    [r0 release];
    if (var_416 != var_412) goto loc_14990;
    goto loc_14560;

loc_14990:
    r2 = var_488;
    var_220 = *imp___nl_symbol_ptr__objc_msgSend;
    var_216 = r2;
    (var_220)(var_496, *0x39670, r2, var_220);
    r7 = r7;
    r0 = objc_retainAutoreleasedReturnValue();
    var_212 = r0;
    [r0 release];
    if (var_216 != var_212) goto loc_14d7e;
```

Well, this pretty much gives everything away. And this is why i love Hopper so much. You can find the exported pdf file containing the Pseudo code in the same folder.

Some things that we can interpret from this Pseudo code.

a)
```
objc_storeStrong(&var_488, r2);
NSSearchPathForDirectoriesInDomains(0x9, 0x1, SIGN_EXTEND(0x1));
r7 = r7;
r0 = objc_retainAutoreleasedReturnValue();
var_440 = r0;
var_436 = *imp___nl_symbol_ptr__objc_msgSend;
(*0x39618)(r0, *0x39618, 0x0, var_436);
r7 = r7;
r0 = objc_retainAutoreleasedReturnValue();
var_432 = r0;
var_428 = @"/secret-data";
var_424 = *imp___nl_symbol_ptr__objc_msgSend;
(var_424)();
```

This code indicates a file with the name *secret-data*.

```
b) r0 = [var_460 isEqualToData:var_448];
   if (SIGN_EXTEND(r0) != 0x0) {
         var_164 = *imp___nl_symbol_ptr__objc_msgSend;
         (var_164)(var_496, *0x39680, var_164);
         r0 = objc_retainAutoreleasedReturnValue();
         var_160 = r0;
         var_156 = 0x0;
         var_152 = *imp___nl_symbol_ptr__objc_msgSend;
         (var_152)(r0, *0x39428, SIGN_EXTEND(var_156), var_156);
         [var_160 release];
         var_148 = *imp___nl_symbol_ptr__objc_msgSend;
         [var_496 returningUserTextField];
         r0 = objc_retainAutoreleasedReturnValue();
         var_144 = r0;
         var_140 = 0x1;
         var_136 = *imp___nl_symbol_ptr__objc_msgSend;
         (var_136)(r0, *0x39428, SIGN_EXTEND(var_140), var_140);
         [var_144 release];
         var_132 = *imp___nl_symbol_ptr__objc_msgSend;
         [var_496 returningUserLabel];
         r0 = objc_retainAutoreleasedReturnValue();
         var_128 = r0;
         var_124 = 0x1;
         var_120 = *imp___nl_symbol_ptr__objc_msgSend;
         (var_120)(r0, *0x39428, SIGN_EXTEND(var_124), var_124);
         [var_128 release];
         var_480 = 0x0;
   }
   else {
         r0 = *imp___nl_symbol_ptr__objc_msgSend;
         r2 = 0x39450;
         r3 = 0x3a162;
         var_116 = r0;
         var_112 = r0;
         var_108 = 0x3944a;
         var_104 = r0;
         var_100 = r2;
         var_96 = r3;
         var_92 = @"Password is incorrect";
         var_88 = 0x0;
         var_84 = 0x3a3c2;
         (*0x39418)(*bind__OBJC_CLASS_$_UIAlertView, *0x39418, r2, r3);
         r9 = var_88;
         *r13 = r9;
         var_4 = var_84;
         var_8 = r9;
         r0 = (var_104)();
         var_80 = r0;
         (var_112)(r0, *var_100, var_112);
         [var_80 release];
         var_500 = 0x0;
         var_480 = 0x1;
```

This code indicates that two NSData values are being compared and if they are equal, then the user is

logged in, otherwise the user is shown an alert stating that the password is incorrect.

c)
```
var_300 = *_kRNCryptorAES256Settings;
var_296 = r9;
do {
        r1 = var_308;
        r2 = var_296;
        *r2 = *r1;
        r0 = var_304 - 0x4;
        var_304 = r0;
        var_308 = r1;
        var_296 = r2;
} while (r0 != 0x0);
[var_312 encryptData:r2 withSettings:STK3 password:STK31 error:STK30];
```

The encryption being used is AES256 and the method implementation can be found by the last line of this code

d) Let's also have a look at the disassembly. We can see a section like this in the disassembly of this method

```
0x000146b0 4FF64C20          movw    r0, #0xfa4c                 ; XREF=0x145e0
0x000146b4 C0F20100          movt    r0, #0x1
0x000146b8 7844              add     r0, pc                      ; 0x34108
0x000146ba 0068              ldr     r0, [r0]                    ; @imp___nl_symbol_ptr__objc_msgSend
0x000146bc 44F6A851          movw    r1, #0x4da8
0x000146c0 C0F20201          movt    r1, #0x2
0x000146c4 7944              add     r1, pc                      ; 0x39470
0x000146c6 7C9A              ldr     r2, [sp, #0x1f0]
0x000146c8 0968              ldr     r1, [r1]                    ; @selector(passwordTextField)
0x000146ca 5590              str     r0, [sp, #0x154]
0x000146cc 1046              mov     r0, r2
0x000146ce 559A              ldr     r2, [sp, #0x154]
0x000146d0 9047              blx     r2
0x000146d2 3F46              mov     r7, r7
0x000146d4 1FF03EEC          blx     imp___symbolstub1__objc_retainAutoreleasedReturnValue
0x000146d8 4FF62421          movw    r1, #0xfa24
0x000146dc C0F20101          movt    r1, #0x1
0x000146e0 7944              add     r1, pc                      ; 0x34108
0x000146e2 0968              ldr     r1, [r1]                    ; @imp___nl_symbol_ptr__objc_msgSend
0x000146e4 44F67852          movw    r2, #0x4d78
0x000146e8 C0F20202          movt    r2, #0x2
0x000146ec 7A44              add     r2, pc                      ; 0x39468
0x000146ee 1268              ldr     r2, [r2]                    ; @selector(text)
0x000146f0 0346              mov     r3, r0
0x000146f2 5490              str     r0, [sp, #0x150]
0x000146f4 1846              mov     r0, r3
0x000146f6 5391              str     r1, [sp, #0x14c]
0x000146f8 1146              mov     r1, r2
0x000146fa 539A              ldr     r2, [sp, #0x14c]
0x000146fc 9047              blx     r2
0x000146fe 3F46              mov     r7, r7
0x00014700 1FF028EC          blx     imp___symbolstub1__objc_retainAutoreleasedReturnValue
0x00014704 0422              movs    r2, #0x4
0x00014706 C0F20002          movt    r2, #0x0
0x0001470a 4FF6F211          movw    r1, #0xf9f2
0x0001470e C0F20101          movt    r1, #0x1
0x00014712 7944              add     r1, pc                      ; 0x34108
0x00014714 0968              ldr     r1, [r1]                    ; @imp___nl_symbol_ptr__objc_msgSend
0x00014716 44F69A53          movw    r3, #0x4d9a
0x0001471a C0F20203          movt    r3, #0x2
0x0001471e 7B44              add     r3, pc                      ; 0x394bc
0x00014720 1B68              ldr     r3, [r3]                    ; @selector(dataUsingEncoding:)
0x00014722 8146              mov     r9, r0
0x00014724 5290              str     r0, [sp, #0x148]
0x00014726 4846              mov     r0, r9
0x00014728 5191              str     r1, [sp, #0x144]
```

From the labels on the right, we can figure out that the *passwordtextfield* is being accessed and its text is being found out. Then this text is converted into NSData using the method dataUsingEncoding. The argument to this method would be contained in the r2 register . If we look a bit above this we see the assembly instruction *movs r2, #0x4* which indicates the argument value is 4 which stands NSUTF8StringEncoding.

```
e) 0x00014812 CDF8D4E1          str.w     lr, [sp, #0x1d4]
0x00014816 759C                 ldr       r4, [sp, #0x1d4]
0x00014818 799D                 ldr       r5, [sp, #0x1e4]
0x0001481a DCF800C0             ldr.w     r12, [r12]                ; @selector(writeToFile:atomically:)
0x0001481e 4890                 str       r0, [sp, #0x120]
0x00014820 2046                 mov       r0, r4
0x00014822 4791                 str       r1, [sp, #0x11c]
```

This code indicates something is being written to the file *secret-data*.

```
f)
0x000149d0 4FF22C70             movw      r0, #0xf72c
0x000149d4 C0F20100             movt      r0, #0x1
0x000149d8 7844                 add       r0, pc                    ; 0x34108
0x000149da 0068                 ldr       r0, [r0]                  ; @imp___nl_symbol_ptr__objc_msgSend
0x000149dc 44F68841             movw      r1, #0x4c88
0x000149e0 C0F20201             movt      r1, #0x2
0x000149e4 7944                 add       r1, pc                    ; 0x39670
0x000149e6 7C9A                 ldr       r2, [sp, #0x1f0]
0x000149e8 0968                 ldr       r1, [r1]                  ; @selector(returningUserTextField)
0x000149ea 3490                 str       r0, [sp, #0xd0]
0x000149ec 1046                 mov       r0, r2
0x000149ee 349A                 ldr       r2, [sp, #0xd0]
0x000149f0 9047                 blx       r2
0x000149f2 3F46                 mov       r7, r7
0x000149f4 1FF0AEEA             blx       imp___symbolstub1__objc_retainAutoreleasedReturnValue
0x000149f8 4FF20471             movw      r1, #0xf704
0x000149fc C0F20101             movt      r1, #0x1
0x00014a00 7944                 add       r1, pc                    ; 0x34108
0x00014a02 0968                 ldr       r1, [r1]                  ; @imp___nl_symbol_ptr__objc_msgSend
0x00014a04 44F65822             movw      r2, #0x4a58
0x00014a08 C0F20202             movt      r2, #0x2
0x00014a0c 7A44                 add       r2, pc                    ; 0x39468
0x00014a0e 1268                 ldr       r2, [r2]                  ; @selector(text)
0x00014a10 0346                 mov       r3, r0
0x00014a12 3390                 str       r0, [sp, #0xcc]
0x00014a14 1846                 mov       r0, r3
0x00014a16 3291                 str       r1, [sp, #0xc8]
0x00014a18 1146                 mov       r1, r2
0x00014a1a 329A                 ldr       r2, [sp, #0xc8]
0x00014a1c 9047                 blx       r2
0x00014a1e 3F46                 mov       r7, r7
0x00014a20 1FF098EA             blx       imp___symbolstub1__objc_retainAutoreleasedReturnValue
0x00014a24 0422                 movs      r2, #0x4
0x00014a26 C0F20002             movt      r2, #0x0
0x00014a2a 4FF2D261             movw      r1, #0xf6d2
0x00014a2e C0F20101             movt      r1, #0x1
0x00014a32 7944                 add       r1, pc                    ; 0x34108
0x00014a34 0968                 ldr       r1, [r1]                  ; @imp___nl_symbol_ptr__objc_msgSend
0x00014a36 44F67A23             movw      r3, #0x4a7a
0x00014a3a C0F20203             movt      r3, #0x2
0x00014a3e 7B44                 add       r3, pc                    ; 0x394bc
0x00014a40 1B68                 ldr       r3, [r3]                  ; @selector(dataUsingEncoding:)
0x00014a42 8146                 mov       r9, r0
0x00014a44 3190                 str       r0, [sp, #0xc4]
```

```
0x00014a46 4846                 mov     r0, r9
0x00014a48 3091                 str     r1, [sp, #0xc0]
0x00014a4a 1946                 mov     r1, r3
0x00014a4c 309B                 ldr     r3, [sp, #0xc0]
0x00014a4e 9847                 blx     r3
0x00014a50 3F46                 mov     r7, r7
0x00014a52 1FF080EA                 blx     imp___symbolstub1__objc_retainAutoreleasedReturnValue
0x00014a56 7390                 str     r0, [sp, #0x1cc]
0x00014a58 3198                 ldr     r0, [sp, #0xc4]
0x00014a5a 1FF074EA                 blx     imp___symbolstub1__objc_release
0x00014a5e 3398                 ldr     r0, [sp, #0xcc]
0x00014a60 1FF070EA                 blx     imp___symbolstub1__objc_release
0x00014a64 4FF29860                 movw    r0, #0xf698
0x00014a68 C0F20100                 movt    r0, #0x1
0x00014a6c 7844                 add     r0, pc                  ; 0x34108
0x00014a6e 0068                 ldr     r0, [r0]                ; @imp___nl_symbol_ptr__objc_msgSend
0x00014a70 44F6F831                 movw    r1, #0x4bf8
0x00014a74 C0F20201                 movt    r1, #0x2
0x00014a78 7944                 add     r1, pc                  ; 0x39674
0x00014a7a 44F6CE72                 movw    r2, #0x4fce
0x00014a7e C0F20202                 movt    r2, #0x2
0x00014a82 7A44                 add     r2, pc                  ; 0x39a54
0x00014a84 0023                 movs    r3, #0x0
0x00014a86 C0F20003                 movt    r3, #0x0
0x00014a8a 7293                 str     r3, [sp, #0x1c8]
0x00014a8c 1268                 ldr     r2, [r2]                ; @bind__OBJC_CLASS_$_NSData
0x00014a8e 799B                 ldr     r3, [sp, #0x1e4]
0x00014a90 0968                 ldr     r1, [r1]                ; @selector(dataWithContentsOfFile:)
0x00014a92 2F90                 str     r0, [sp, #0xbc]
0x00014a94 1046                 mov     r0, r2
0x00014a96 1A46                 mov     r2, r3
0x00014a98 2F9B                 ldr     r3, [sp, #0xbc]
0x00014a9a 9847                 blx     r3
0x00014a9c 3F46                 mov     r7, r7
0x00014a9e 1FF05AEA                 blx     imp___symbolstub1__objc_retainAutoreleasedReturnValue
0x00014aa2 45F62611                 movw    r1, #0x5926
0x00014aa6 C0F20201                 movt    r1, #0x2
0x00014aaa 7944                 add     r1, pc                  ; 0x3a3d4
0x00014aac 6FAA                 add     r2, sp, #0x1bc
0x00014aae 4FF24E63                 movw    r3, #0xf64e
0x00014ab2 C0F20103                 movt    r3, #0x1
0x00014ab6 7B44                 add     r3, pc                  ; 0x34108
0x00014ab8 1B68                 ldr     r3, [r3]                ; @imp___nl_symbol_ptr__objc_msgSend
0x00014aba 44F6B239                 movw    r9, #0x4bb2
0x00014abe C0F20209                 movt    r9, #0x2
0x00014ac2 F944                 add     r9, pc                  ; 0x39678
0x00014ac4 44F6C47C                 movw    r12, #0x4fc4
0x00014ac8 C0F2020C                 movt    r12, #0x2
0x00014acc FC44                 add     r12, pc                 ; 0x39a94
0x00014ace 7190                 str     r0, [sp, #0x1c4]
0x00014ad0 DCF80000                 ldr.w   r0, [r12]               ; @0x39a94
0x00014ad4 DDF8C4C1                 ldr.w   r12, [sp, #0x1c4]
0x00014ad8 DDF8C8E1                 ldr.w   lr, [sp, #0x1c8]
0x00014adc CDF8BCE1                 str.w   lr, [sp, #0x1bc]
0x00014ae0 D9F80090                 ldr.w   r9, [r9]                ;
@selector(decryptData:withPassword:error:)
0x00014ae4 2E91                 str     r1, [sp, #0xb8]
0x00014ae6 4946                 mov     r1, r9
0x00014ae8 2D92                 str     r2, [sp, #0xb4]
0x00014aea 6246                 mov     r2, r12
0x00014aec DDF8B890                 ldr.w   r9, [sp, #0xb8]
0x00014af0 2C93                 str     r3, [sp, #0xb0]
0x00014af2 4B46                 mov     r3, r9
0x00014af4 DDF8B4C0                 ldr.w   r12, [sp, #0xb4]
0x00014af8 CDF800C0                 str.w   r12, [sp]
0x00014afc DDF8B0E0                 ldr.w   lr, [sp, #0xb0]
0x00014b00 F047                 blx     lr
```

```
0x00014b02 3F46          mov       r7, r7
0x00014b04 1FF026EA          blx       imp___symbolstub1__objc_retainAutoreleasedReturnValue
0x00014b08 72A9          add       r1, sp, #0x1c8
0x00014b0a 6F9A          ldr       r2, [sp, #0x1bc]
0x00014b0c 2B90          str       r0, [sp, #0xac]
0x00014b0e 0846          mov       r0, r1
0x00014b10 1146          mov       r1, r2
0x00014b12 1FF026EA          blx       imp___symbolstub1__objc_storeStrong
0x00014b16 4FF2E650          movw      r0, #0xf5e6
0x00014b1a C0F20100          movt      r0, #0x1
0x00014b1e 7844          add       r0, pc                  ; 0x34108
0x00014b20 0068          ldr       r0, [r0]                ; @imp___nl_symbol_ptr__objc_msgSend
0x00014b22 44F64E31          movw      r1, #0x4b4e
0x00014b26 C0F20201          movt      r1, #0x2
0x00014b2a 7944          add       r1, pc                  ; 0x3967c
0x00014b2c 2B9A          ldr       r2, [sp, #0xac]
0x00014b2e 7092          str       r2, [sp, #0x1c0]
0x00014b30 739B          ldr       r3, [sp, #0x1cc]
0x00014b32 709A          ldr       r2, [sp, #0x1c0]
0x00014b34 0968          ldr       r1, [r1]                ; @selector(isEqualToData:)
```

Again by looking at the assembly above, it can be found out that the text from *returninguserTextField* is being accessed, converted into NSData and then compared against some other NSData. If both the data values match, user is logged in. There is also some decryption of data *(@selector(decryptData:withPassword:error:))* and a method call (*@selector(dataWithContentsOfFile:)*. It looks like the data being compared is actually fetched from the file secret-data, decrypted and then compared with the data that is obtained after using the method *datausingencoding:* on the input text being entered. So it looks like we can just find the password by decrypting the data from the file secret-data. However, we don't know the password that is used in the method *decryptData:withPassword:error:* . If we can find the password being used for encryption, we can decrypt the pasword from the data stored in the file *secret-data.*

g) After searching for a bit more, we can see this snippet of code in the pseudo code.

```
var_100 = 0x0;
 r6 = 0x44;
 r5 = r13;
 var_68 = @"Secret-Key";
 var_72 = &var_100;
 do {
        r6 = r6 - 0x4;
        *r5 = *_kRNCryptorAES256Settings;
 } while (r6 != 0x0);
 r1 = *0x29604;
```

So maybe the password is *Secret-Key.* Well, it is pretty much clear that we can find the password by decrypting the data from the file *secret-data* and converting it into a string with the encoding NSUTF8StringEncoding. Let's write a simple IOS Application to decrypt the data. For this, you will need to copy the file *secret-data* from the application sandbox (for the DVIA application) and paste it into the documents folder of the application's sandbox of this new application. You can also download the complete code from https://github.com/prateek147/InsecureCryptographyDecryptor. This project code is also included in the same folder as this pdf file.

We add this method in the new project. Note that we are using the same key (Secret-Key) to decrypt the password.

```
NSString *dataPath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES) objectAtIndex:0] stringByAppendingPathComponent:@"/secret-data"];
    NSError *error;
    NSData *encryptedData = [NSData dataWithContentsOfFile:dataPath];
    NSData *decryptedData = [RNDecryptor decryptData:encryptedData
                                        withPassword:@"Secret-Key"
                                               error:&error];
   NSString *password = [[NSString alloc] initWithData:decryptedData
                                               encoding:NSUTF8StringEncoding];


  UILabel *newLabel = [[UILabel alloc] initWithFrame:CGRectMake(140.0, 160.0, 100.0, 100.0)];
  [self.view addSubview:newLabel];
  [newLabel setText:password];
```

As you can clearly note, this method decrypts the data using the hardcoded password (Secret-Key), and shows the value in a label that is displayed on the view. We must run this application first without the *secret-data* file on the device. This will create an application folder for the application. Then we can copy the *secret-data* file into its application sandbox in the same location, i.e inside the Documents folder.

After running this application for the 2nd time, we can easily see the decrypted password.