!gnireenignE

A Reverse Engineering Primer by

Chris Davisson & Joe Grassl

Fundamentals

Reverse engineering is very useful for:

- Finding hardcoded or dynamically generated credentials (passwords)
- Vulnerability discovery and exploit development
- Modding, patching, and otherwise reading/modifying things that the creator of a program thinks or hopes you won’t be able to
- Keeping China’s economy afloat

Terminology

- **Crack** – To defeat a security mechanism
- **Patch** – To modify a program without changing the actual source code
- **Block** – A unit of linear code that ends in a conditional fork (true/false), a jump to another block, or a simple exit.
- **Obfuscation** – Hiding the true nature of the code through clever code mangling. Looks like gibberish. Not the same as encryption (but may include it).

Terminology – Part 2

- **Static Analysis** – The program is tested “at rest”. Involves looking at disassembled code and mentally modeling what the program should do.
- **Dynamic Analysis** – The program is tested live. Inputs are given to the running program and breakpoints are set to look at values in memory.
- **Debugger** – Allows you to do dynamic analysis. GDB, for example.
- **Disassembler** – Produces low-level output from a compiled program.
- **Decompiler** – Produces high-level output from a compiled program. Sometimes very close to the true source code itself!

Reverse Engineering in C/C++

radare2 & ghidra

Ghidra

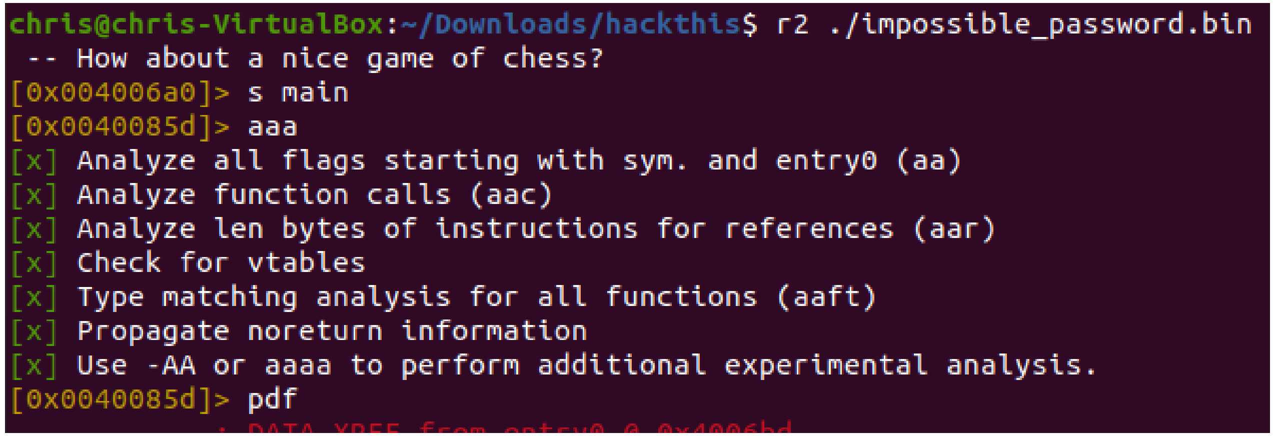
- An open-source reverse engineering tool
- Developed by the NSA in 2019
- First revealed in WikiLeaks in 2017
- Looks like it was made in the 90's but has some cool functionality.
    - Estimates the C code
    - Decompiles the header file

Radare2

- A Free framework for reverse-engineering and analyzing binaries
- Created in 2006 as an interface for editing hexadecimal an hard drive recovery.
- Created by Sergi Alvarez aka. pancake

Example Run

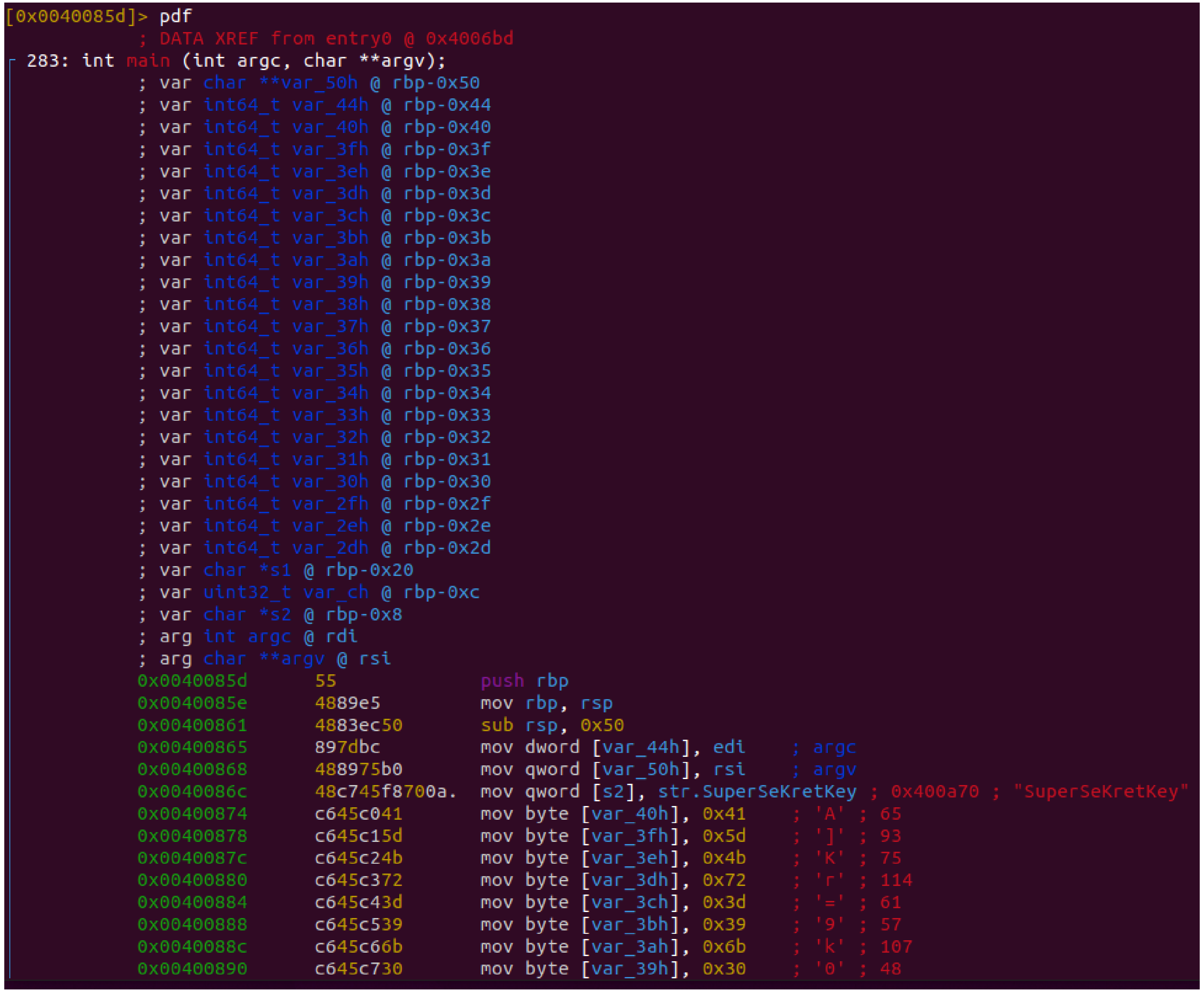
- Runs using the r2 keyword after compiling in command line
- S main seeks to the main
- Aaa is to tell it what to analyze



```
chris@chris-VirtualBox:~/Downloads/hackthis$ r2 ./impossible_password.bin
 -- How about a nice game of chess?
[0x004006a0]> s main
[0x0040085d]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[x] Analyze function calls (aac)
[x] Analyze len bytes of instructions for references (aar)
[x] Check for vtables
[x] Type matching analysis for all functions (aaft)
[x] Propagate noreturn information
[x] Use -AA or aaaa to perform additional experimental analysis.
[0x0040085d]> pdf
            ; DATA XREF from entry0 @ 0x4006bd
```

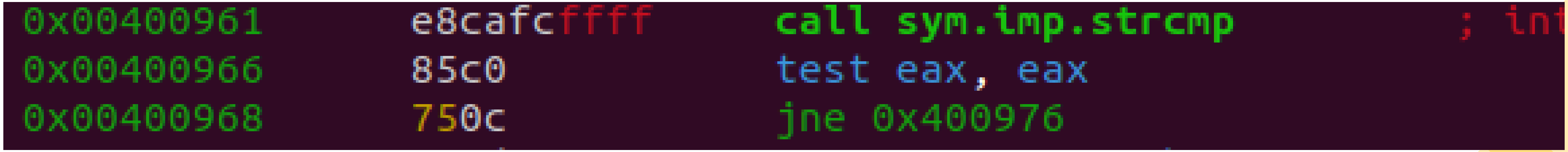
- Using the command: pdf
- Displays what the code is doing.

- Though it looks complex, we can see that a string is easily visible.
- "SuperSeKretKey"



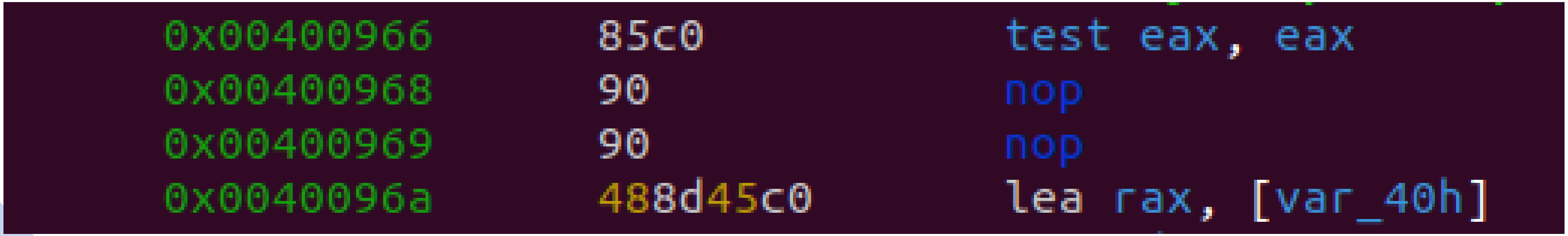
```
[0x0040085d]> pdf
            ; DATA XREF from entry0 @ 0x4006bd
┌ 283: int main (int argc, char **argv);
│           ; var char **var_50h @ rbp-0x50
│           ; var int64_t var_44h @ rbp-0x44
│           ; var int64_t var_40h @ rbp-0x40
│           ; var int64_t var_3fh @ rbp-0x3f
│           ; var int64_t var_3eh @ rbp-0x3e
│           ; var int64_t var_3dh @ rbp-0x3d
│           ; var int64_t var_3ch @ rbp-0x3c
│           ; var int64_t var_3bh @ rbp-0x3b
│           ; var int64_t var_3ah @ rbp-0x3a
│           ; var int64_t var_39h @ rbp-0x39
│           ; var int64_t var_38h @ rbp-0x38
│           ; var int64_t var_37h @ rbp-0x37
│           ; var int64_t var_36h @ rbp-0x36
│           ; var int64_t var_35h @ rbp-0x35
│           ; var int64_t var_34h @ rbp-0x34
│           ; var int64_t var_33h @ rbp-0x33
│           ; var int64_t var_32h @ rbp-0x32
│           ; var int64_t var_31h @ rbp-0x31
│           ; var int64_t var_30h @ rbp-0x30
│           ; var int64_t var_2fh @ rbp-0x2f
│           ; var int64_t var_2eh @ rbp-0x2e
│           ; var int64_t var_2dh @ rbp-0x2d
│           ; var char *s1 @ rbp-0x20
│           ; var uint32_t var_ch @ rbp-0xc
│           ; var char *s2 @ rbp-0x8
│           ; arg int argc @ rdi
│           ; arg char **argv @ rsi
│           0x0040085d      55             push rbp
│           0x0040085e      4889e5         mov rbp, rsp
│           0x00400861      4883ec50       sub rsp, 0x50
│           0x00400865      897dbc         mov dword [var_44h], edi    ; argc
│           0x00400868      488975b0       mov qword [var_50h], rsi    ; argv
│           0x0040086c      48c745f8700a.  mov qword [s2], str.SuperSeKretKey ; 0x400a70 ; "SuperSeKretKey"
│           0x00400874      c645c041       mov byte [var_40h], 0x41    ; 'A' ; 65
│           0x00400878      c645c15d       mov byte [var_3fh], 0x5d    ; ']' ; 93
│           0x0040087c      c645c24b       mov byte [var_3eh], 0x4b    ; 'K' ; 75
│           0x00400880      c645c372       mov byte [var_3dh], 0x72    ; 'r' ; 114
│           0x00400884      c645c43d       mov byte [var_3ch], 0x3d    ; '=' ; 61
│           0x00400888      c645c539       mov byte [var_3bh], 0x39    ; '9' ; 57
│           0x0040088c      c645c66b       mov byte [var_3ah], 0x6b    ; 'k' ; 107
│           0x00400890      c645c730       mov byte [var_39h], 0x30    ; '0' ; 48
```

This String is the first password that was required. Though after, it called a function that would always give false.



```
0x00400961      e8cafcffff      call sym.imp.strcmp         ; int
0x00400966      85c0            test eax, eax
0x00400968      750c            jne 0x400976
```

The method is the line with jne 0x400976
And if we seek into that method, all we see is that it will always return false, forcing the user out of the program.
To combat this we can simply set the method to "nop" or no operation.



```
0x00400966      85c0            test eax, eax
0x00400968      90              nop
0x00400969      90              nop
0x0040096a      488d45c0        lea rax, [var_40h]
```

Explanation

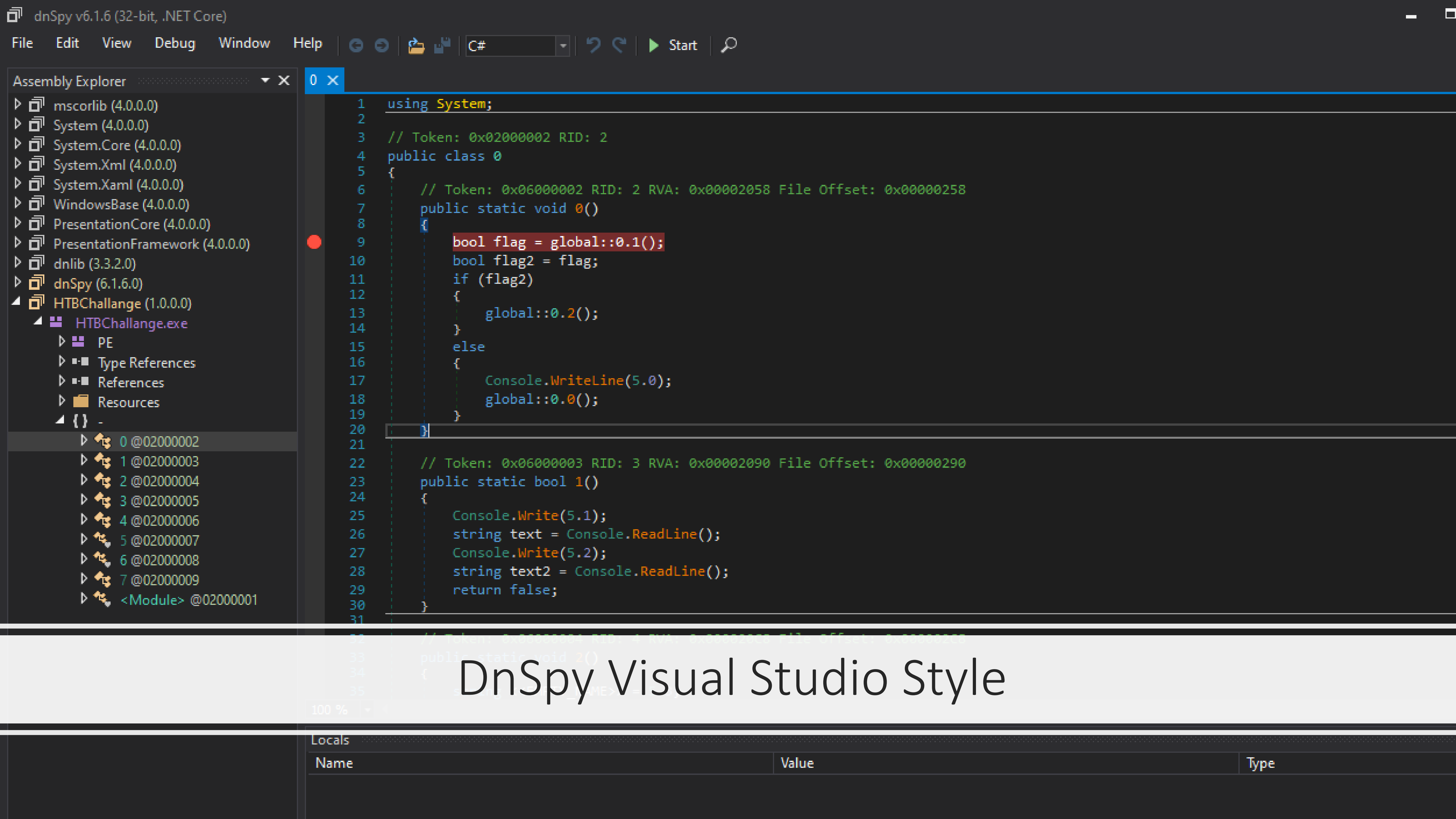
- The Radare2 software allows us to analyze what is happening inside the software. So if you have user input being compared to a string, it can see the string.
- If you have a gatekeeper method that is just changing a Boolean, it can alter the compiled code to just bypass it.
- I'm not a good hacker, and I was able to crack the security provided this program quickly. (Just think what someone good at it could do)

Reverse Engineering in C#

dnSpy & de4dot

dnSpy & de4dot

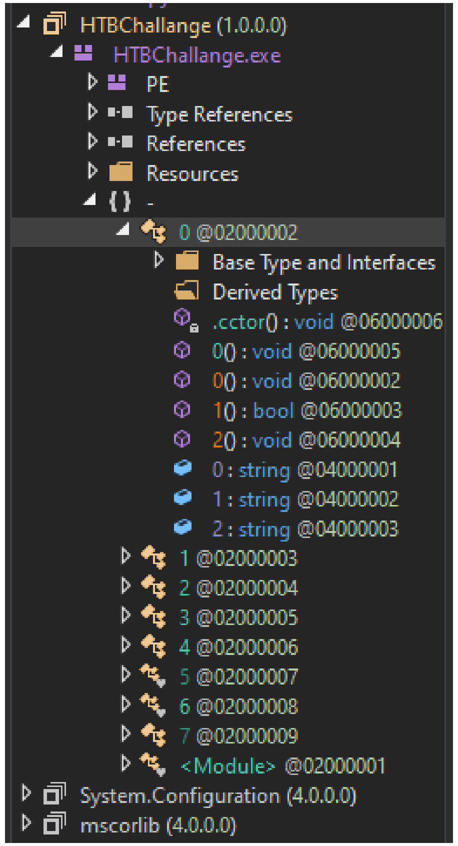
- Similar look and feel to Visual Studio
- Cool functionality
  - Allows you to alter values during runtime
- Some programs are obfuscated and very hard to read and edit
- de4dot can be used to remove well known obfuscation schemes in about three clicks



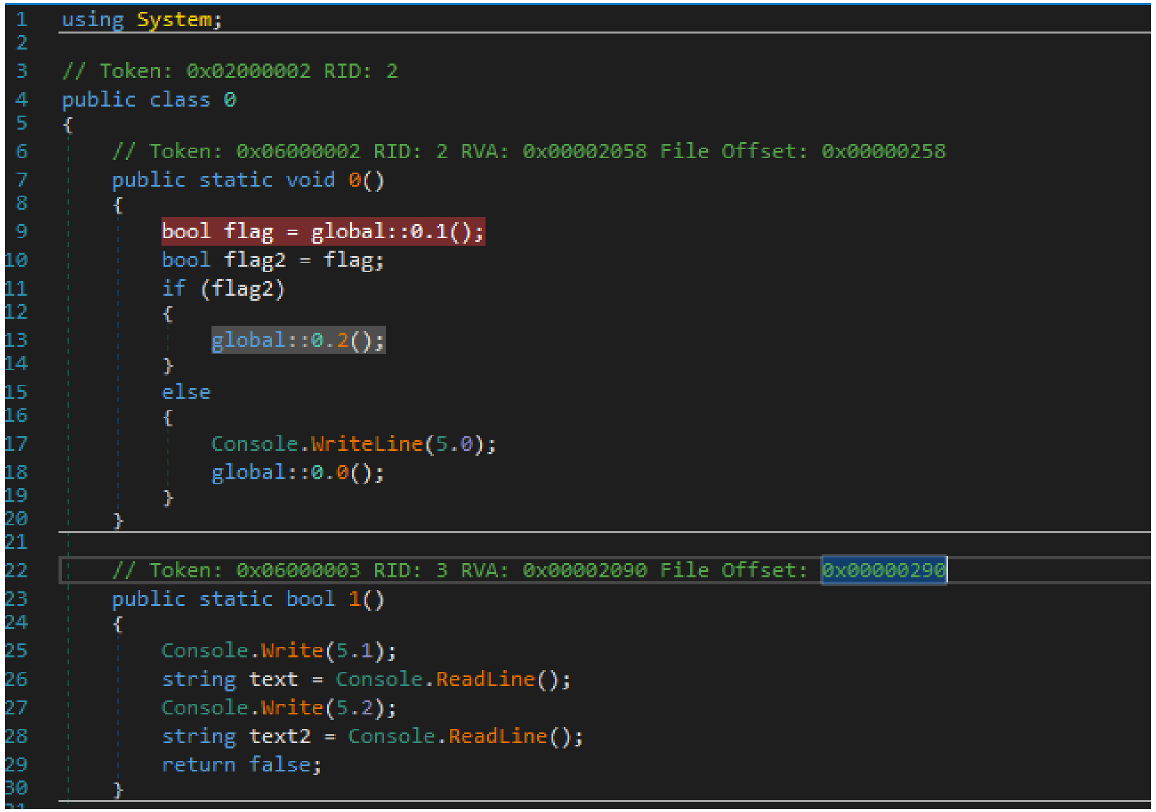
```
using System;

// Token: 0x02000002 RID: 2
public class 0
{
    // Token: 0x06000002 RID: 2 RVA: 0x00002058 File Offset: 0x00000258
    public static void 0()
    {
        bool flag = global::0.1();
        bool flag2 = flag;
        if (flag2)
        {
            global::0.2();
        }
        else
        {
            Console.WriteLine(5.0);
            global::0.0();
        }
    }

    // Token: 0x06000003 RID: 3 RVA: 0x00002090 File Offset: 0x00000290
    public static bool 1()
    {
        Console.Write(5.1);
        string text = Console.ReadLine();
        Console.Write(5.2);
        string text2 = Console.ReadLine();
        return false;
    }
```

DnSpy Visual Studio Style

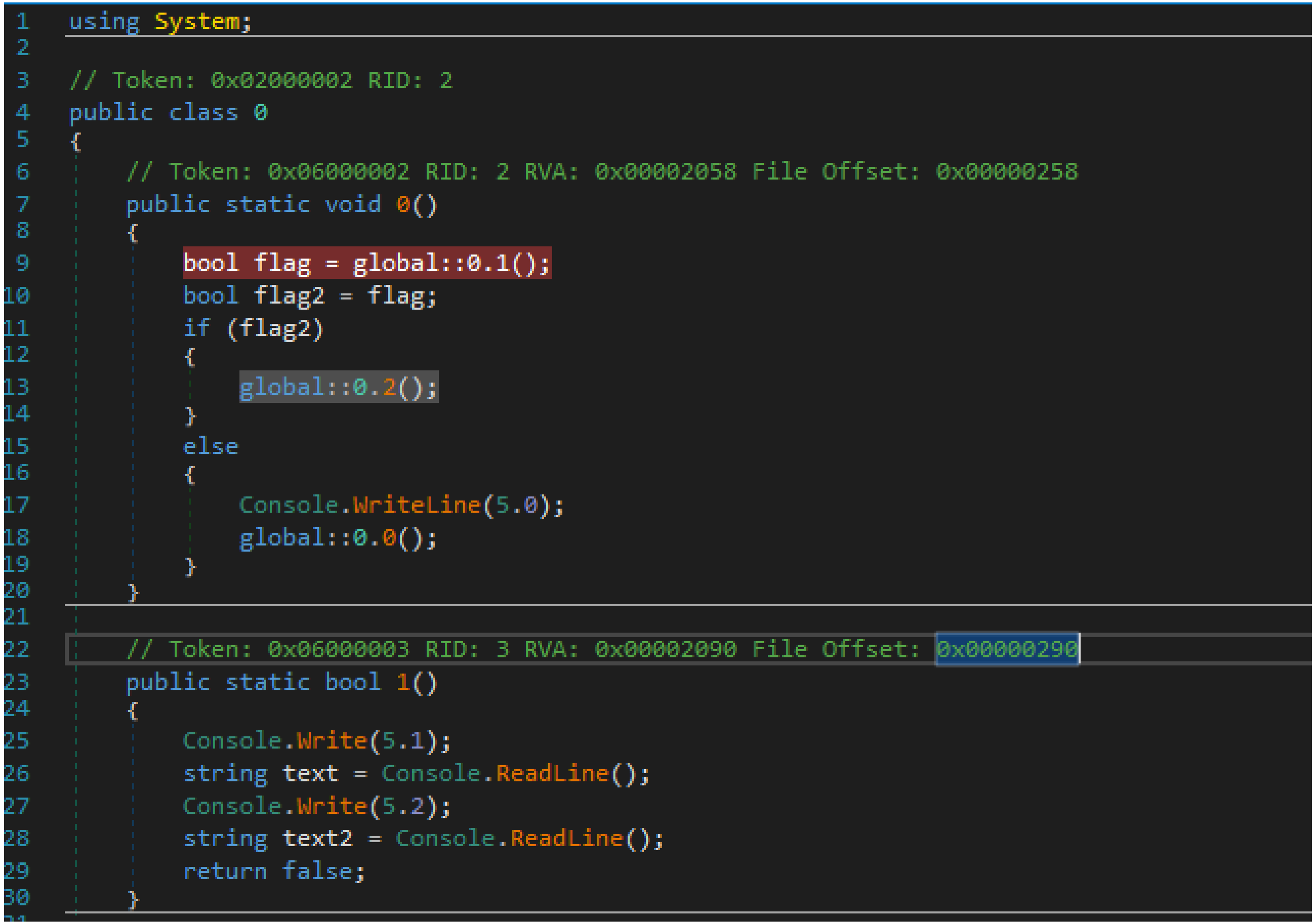
Lists of methods and variables



```
◢ HTBChallange (1.0.0.0)
   ◢ HTBChallange.exe
      ▷ PE
      ▷ Type References
      ▷ References
      ▷ Resources
      ◢ { } -
         ◢ 0 @02000002
            ▷ Base Type and Interfaces
              Derived Types
              .cctor() : void @06000006
              0() : void @06000005
              0() : void @06000002
              1() : bool @06000003
              2() : void @06000004
              0 : string @04000001
              1 : string @04000002
              2 : string @04000003
         ▷ 1 @02000003
         ▷ 2 @02000004
         ▷ 3 @02000005
         ▷ 4 @02000006
         ▷ 5 @02000007
         ▷ 6 @02000008
         ▷ 7 @02000009
         ▷ <Module> @02000001
▷ System.Configuration (4.0.0.0)
▷ mscorlib (4.0.0.0)
```



```
using System;

// Token: 0x02000002 RID: 2
public class 0
{
    // Token: 0x06000002 RID: 2 RVA: 0x00002058 File Offset: 0x00000258
    public static void 0()
    {
        bool flag = global::0.1();
        bool flag2 = flag;
        if (flag2)
        {
            global::0.2();
        }
        else
        {
            Console.WriteLine(5.0);
            global::0.0();
        }
    }

    // Token: 0x06000003 RID: 3 RVA: 0x00002090 File Offset: 0x00000290
    public static bool 1()
    {
        Console.Write(5.1);
        string text = Console.ReadLine();
        Console.Write(5.2);
        string text2 = Console.ReadLine();
        return false;
    }
```



```
using System;

// Token: 0x02000002 RID: 2
public class 0
{
    // Token: 0x06000002 RID: 2 RVA: 0x00002058 File Offset: 0x00000258
    public static void 0()
    {
        bool flag = global::0.1();
        bool flag2 = flag;
        if (flag2)
        {
            global::0.2();
        }
        else
        {
            Console.WriteLine(5.0);
            global::0.0();
        }
    }

    // Token: 0x06000003 RID: 3 RVA: 0x00002090 File Offset: 0x00000290
    public static bool 1()
    {
        Console.Write(5.1);
        string text = Console.ReadLine();
        Console.Write(5.2);
        string text2 = Console.ReadLine();
        return false;
    }
```

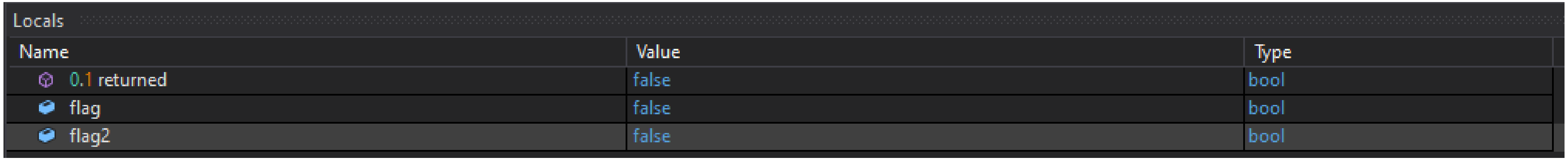
We can see that the main aka "0" method first creates a bool called flag, then assigns flag2 to it too.

If flag2 is false, then it just repeats.

The method it calls to is method 1, that always returns a false. User entry doesn't even matter.

You could enter anything or nothing, it don't care. All answers are wrong

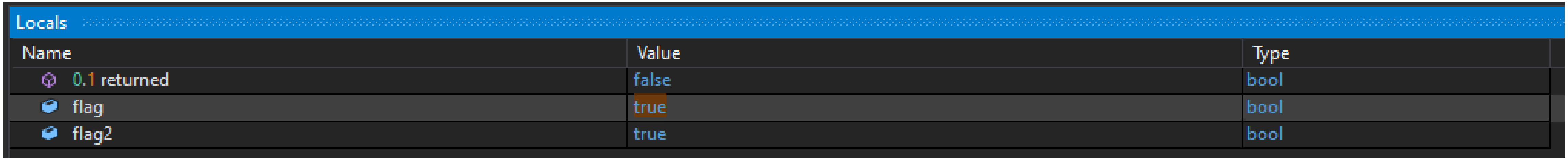
Bool values before changing



Locals

Name	Value	Type
0.1 returned	false	bool
flag	false	bool
flag2	false	bool

Bool values after changing



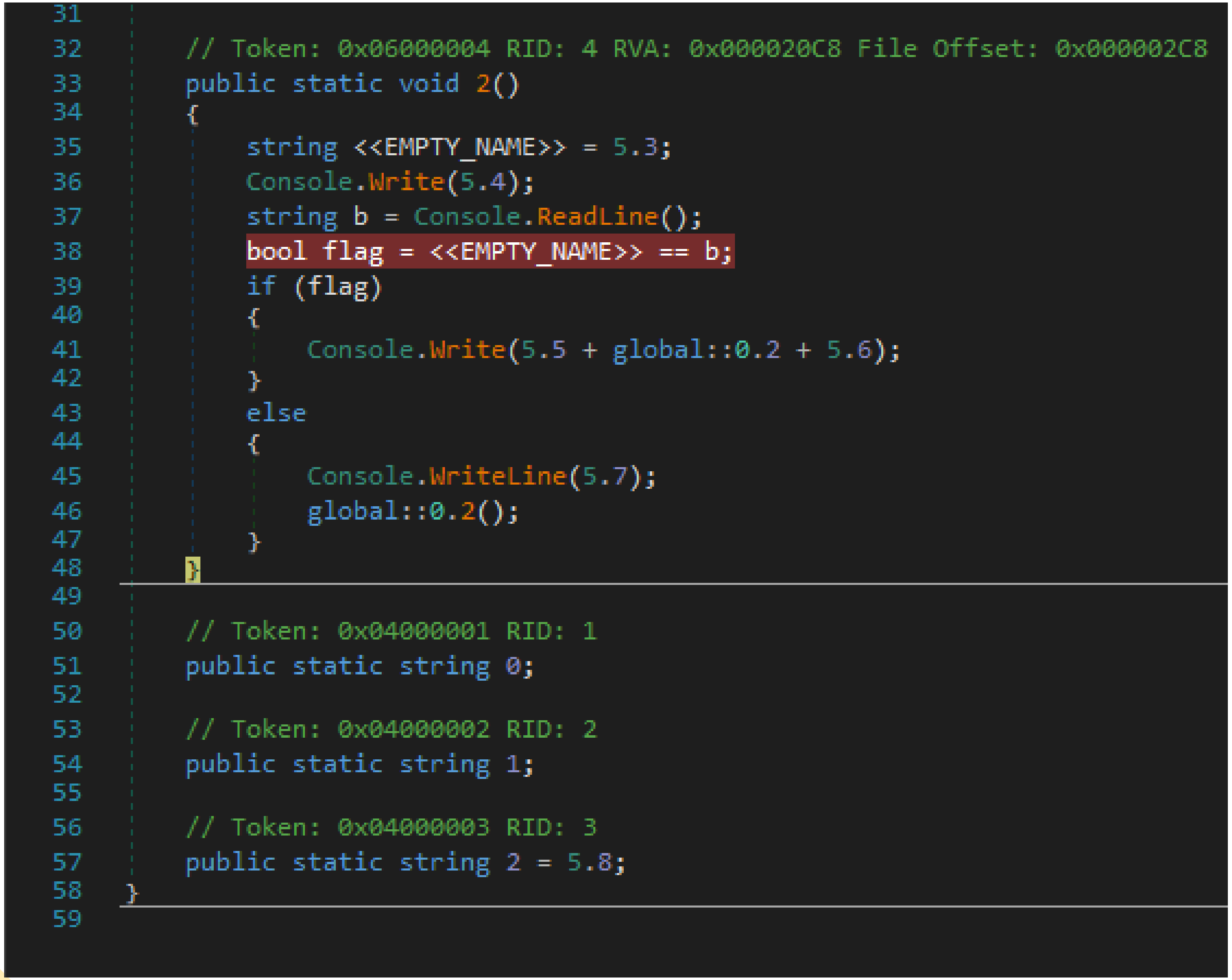
Locals

Name	Value	Type
0.1 returned	false	bool
flag	true	bool
flag2	true	bool

After passing that check, it takes you to method 2.

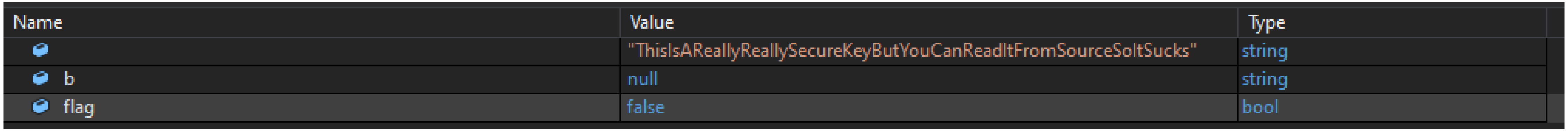
This accepts user input and compares it against a string. Unlike java, the == is a valid way to compare strings

So we have two options, change the flag variable or find the value or <<Empty_Name>>



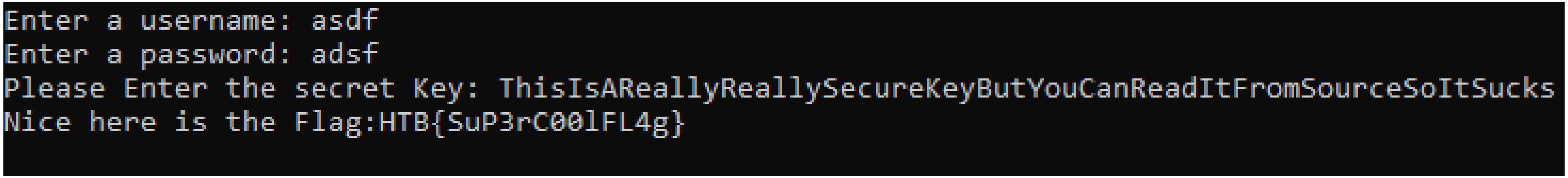
```

    // Token: 0x06000004 RID: 4 RVA: 0x000020C8 File Offset: 0x000002C8
    public static void 2()
    {
        string <<EMPTY_NAME>> = 5.3;
        Console.Write(5.4);
        string b = Console.ReadLine();
        bool flag = <<EMPTY_NAME>> == b;
        if (flag)
        {
            Console.Write(5.5 + global::0.2 + 5.6);
        }
        else
        {
            Console.WriteLine(5.7);
            global::0.2();
        }
    }

    // Token: 0x04000001 RID: 1
    public static string 0;

    // Token: 0x04000002 RID: 2
    public static string 1;

    // Token: 0x04000003 RID: 3
    public static string 2 = 5.8;
}

```

Using breakpoints we can see the value of <<Empty_Name>>



Name	Value	Type
	"ThisIsAReallyReallySecureKeyButYouCanReadItFromSourceSoItSucks"	string
b	null	string
flag	false	bool

Final run after changing bool value and finding secret Key



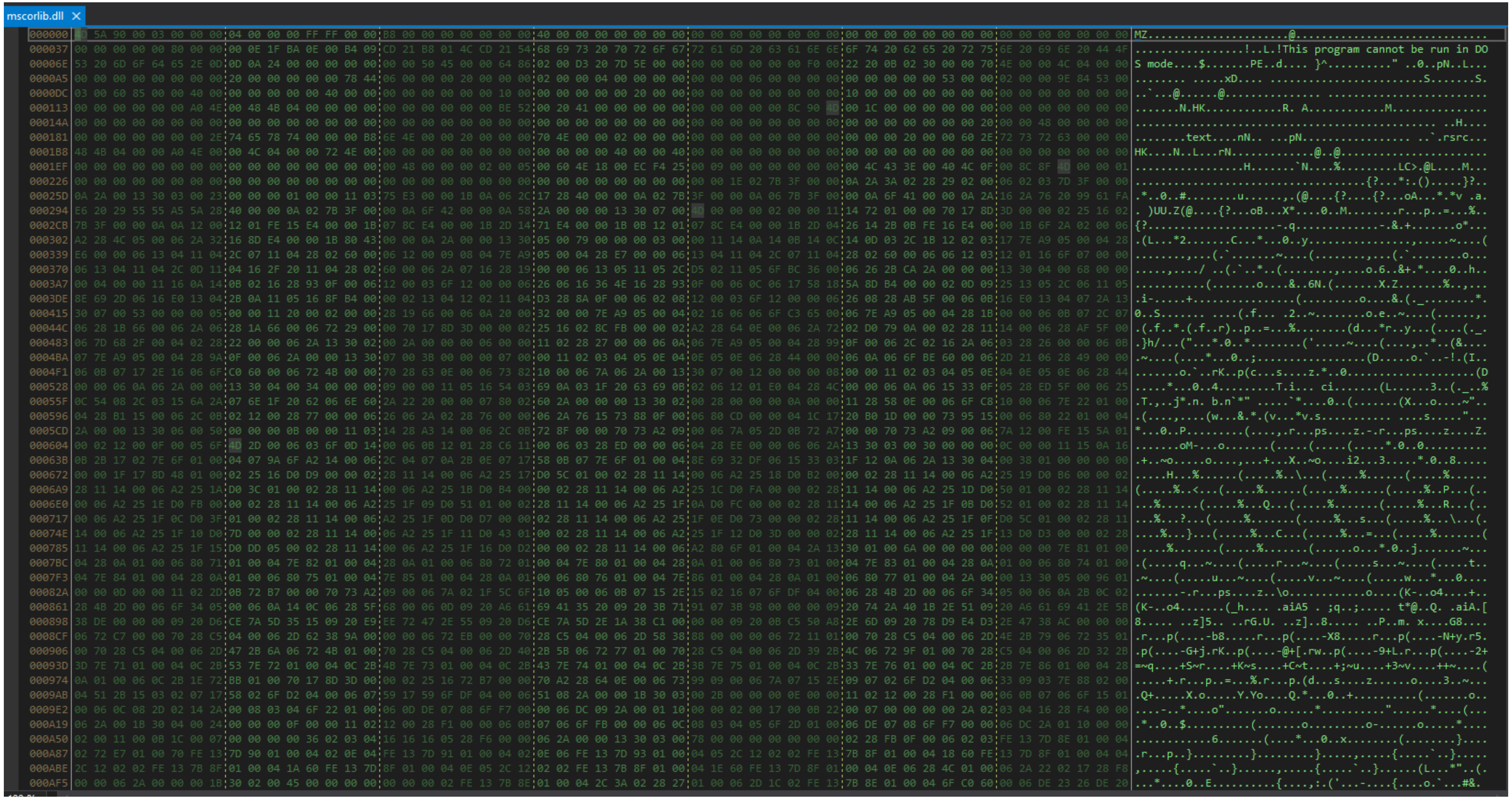
```
Enter a username: asdf
Enter a password: adsf
Please Enter the secret Key: ThisIsAReallyReallySecureKeyButYouCanReadItFromSourceSoItSucks
Nice here is the Flag:HTB{SuP3rC00lFL4g}
```

Hacked!

Final Thoughts

- The dnSpy software provides many tools that make looking at decompiled code a lot less daunting
  - The gui makes all the information easy to see and understand
- Changing the value of variables is too powerful

Also... I found the matrix source code in the Hex Editor



```
000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  MZ......................@..............................
000037 00 00 00 00 00 80 00 00 00 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F  .................!..L.!This program cannot be run in DO
00006E 53 20 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 50 45 00 00 64 86 02 00 D3 20 7D 5E 00 00 00 00 00 00 00 00 F0 00 22 20 0B 02 30 00 00 70 4E 00 00 4C 04 00 00  S mode....$.......PE..d.... }^..........." ..0..pN..L...
0000A5 00 00 00 00 00 00 00 00 20 00 00 00 00 00 78 44 06 00 00 00 20 00 00 00 02 00 00 04 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 00 00 53 00 00 02 00 00 9E 84 53 00  ........ .....xD.... ........................S.......S.
0000DC 03 00 60 85 00 00 40 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 20 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..`...@......@............... .........................
000113 00 00 00 00 00 00 A0 4E 00 48 4B 04 00 00 00 00 00 00 00 00 00 00 BE 52 00 20 41 00 00 00 00 00 00 00 00 00 00 8C 90 4D 00 1C 00 00 00 00 00 00 00 00 00 00 00 00 00  .......N.HK............R. A............M...............
00014A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 48 00 00 00 00  ............................................... ..H....
000181 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00 B8 6E 4E 00 00 20 00 00 00 70 4E 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60 2E 72 73 72 63 00 00 00  ........text....nN.. ...pN................. ..`.rsrc...
0001B8 48 4B 04 00 00 A0 4E 00 00 4C 04 00 00 72 4E 00 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  HK....N..L...rN.............@..@.......................
0001EF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 48 00 00 00 02 00 05 00 60 4E 18 00 EC F4 25 00 09 00 00 00 00 00 00 00 4C 43 3E 00 40 4C 0F 00 8C 8F 4D 00 00 01  .................H.......`N....%.........LC>.@L....M...
000226 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1E 02 7B 3F 00 00 0A 2A 3A 02 28 29 02 00 06 02 03 7D 3F 00 00  ....................................{?...*:.().....}?..
00025D 0A 2A 00 13 30 03 00 23 00 00 00 01 00 00 11 03 75 E3 00 00 1B 0A 06 2C 17 28 40 00 00 0A 02 7B 3F 00 00 0A 06 7B 3F 00 00 0A 6F 41 00 00 0A 2A 16 2A 76 20 99 61 FA  .*..0..#........u......,.(@....{?....{?...oA...*.*v .a.
000294 E6 20 29 55 55 A5 5A 28 40 00 00 0A 02 7B 3F 00 00 0A 6F 42 00 00 0A 58 2A 00 00 00 13 30 07 00 4D 00 00 00 02 00 00 11 14 72 01 00 00 70 17 8D 3D 00 00 02 25 16 02  . )UU.Z(@....{?...oB...X*....0..M........r...p..=...%..
0002CB 7B 3F 00 00 0A 0A 12 00 12 01 FE 15 E4 00 00 1B 07 8C E4 00 00 1B 2D 14 71 E4 00 00 1B 0B 12 01 07 8C E4 00 00 1B 2D 04 26 14 2B 0B FE 16 E4 00 00 1B 6F 2A 02 00 06  {?......................-.q.............-.&.+.......o*...
000302 A2 28 4C 05 00 06 2A 32 16 8D E4 00 00 1B 80 43 00 00 0A 2A 00 00 13 30 05 00 79 00 00 00 03 00 00 11 14 0A 14 0B 14 0C 14 0D 03 2C 1B 12 02 03 17 7E A9 05 00 04 28  .(L...*2.......C...*...0..y...................,.....~....(
000339 E6 00 00 06 13 04 11 04 2C 07 11 04 28 02 60 00 06 12 00 09 08 04 7E A9 05 00 04 28 E7 00 00 06 13 04 11 04 2C 07 11 04 28 02 60 00 06 06 12 03 12 01 16 6F 07 00 00  ........,...(.`.......~....(.......,...(.`........o...
000370 06 13 04 11 04 2C 0D 11 04 16 2F 20 11 04 28 02 60 00 06 2A 07 16 28 19 00 00 06 13 05 11 05 2C D5 02 11 05 6F BC 36 00 06 26 2B CA 2A 00 00 00 13 30 04 00 68 00 00  .....,..../ ..(.`..*..(........,....o.6..&+.*....0..h..
0003A7 00 04 00 00 11 16 0A 14 0B 02 16 28 93 0F 00 06 12 00 03 6F 12 00 00 06 26 06 16 36 4E 16 28 93 0F 00 06 0C 06 17 58 18 5A 8D B4 00 00 02 0D 09 25 13 05 2C 06 11 05  ...........(.......o....&..6N.(.......X.Z.......%..,...
0003DE 8E 69 2D 06 16 E0 13 04 2B 0A 11 05 16 8F B4 00 00 02 13 04 12 02 11 04 D3 28 8A 0F 00 06 02 08 12 00 03 6F 12 00 00 06 26 08 28 AB 5F 00 06 0B 16 E0 13 04 07 2A 13  .i-.....+.................(.........o....&.(._........*.
000415 30 07 00 53 00 00 00 05 00 00 11 20 00 02 00 00 28 19 66 00 06 0A 20 00 32 00 00 7E A9 05 00 04 02 16 06 06 6F C3 65 00 06 7E A9 05 00 04 28 1B 00 00 06 0B 07 2C 07  0..S....... ....(.f... .2..~........o.e..~....(......,.
00044C 06 28 1B 66 00 06 2A 06 28 1A 66 00 06 72 29 00 00 70 17 8D 3D 00 00 02 25 16 02 8C FB 00 00 02 A2 28 64 0E 00 06 2A 72 02 D0 79 0A 00 02 28 11 14 00 06 28 AF 5F 00  .(.f..*.(.f..r)..p..=...%........(d...*r..y...(....(._.
000483 06 7D 68 2F 00 04 02 28 22 00 00 06 2A 13 30 02 00 2A 00 00 00 06 00 00 11 02 28 27 00 00 06 0A 06 7E A9 05 00 04 28 99 0F 00 06 2C 02 16 2A 06 03 28 26 00 00 06 0B  .}h/...("...*.0..*........('.....~....(....,..*..(&....
0004BA 07 7E A9 05 00 04 28 9A 0F 00 06 2A 00 00 13 30 07 00 3B 00 00 00 07 00 00 11 02 03 04 05 0E 04 0E 05 0E 06 28 44 00 00 06 0A 06 6F BE 60 00 06 2D 21 06 28 49 00 00  .~....(....*...0..;.....................(D.....o.`..-!.(I..
0004F1 06 0B 07 17 2E 16 06 6F C0 60 00 06 72 4B 00 00 70 28 63 0E 00 06 73 82 10 00 06 7A 06 2A 00 13 30 07 00 12 00 00 00 08 00 00 11 02 03 04 05 0E 04 0E 05 0E 06 28 44  .......o.`..rK..p(c...s....z.*..0....................(D
000528 00 00 06 0A 06 2A 00 00 13 30 04 00 34 00 00 00 09 00 00 11 05 16 54 03 69 0A 03 1F 20 63 69 0B 02 06 12 01 E0 04 28 4C 00 00 06 0A 06 15 33 0F 05 28 ED 5F 00 06 25  .....*...0..4.........T.i... ci.......(L......3..(._..%
00055F 0C 54 08 2C 03 15 6A 2A 07 6E 1F 20 62 06 6E 60 2A 22 20 00 00 07 80 02 60 2A 00 00 00 13 30 02 00 28 00 00 00 0A 00 00 11 28 58 0E 00 06 6F C8 10 00 06 7E 22 01 00  .T.,..j*.n. b.n`*" .....`*....0..(.......(X...o....~"..
000596 04 28 B1 15 00 06 2C 0B 02 12 00 28 77 00 00 06 26 06 2A 02 28 76 00 00 06 2A 76 15 73 88 0F 00 06 80 CD 00 00 04 1C 17 20 B0 1D 00 00 73 95 15 00 06 80 22 01 00 04  .(....,....(w...&.*.(v...*v.s........... ....s....."...
0005CD 2A 00 00 13 30 06 00 50 00 00 00 0B 00 00 11 03 14 28 A3 14 00 06 2C 0B 72 8F 00 00 70 73 A2 09 00 06 7A 05 2D 0B 72 A7 00 00 70 73 A2 09 00 06 7A 12 00 FE 15 5A 01  *...0..P.........(....,.r...ps....z.-.r...ps....z....Z.
000604 00 02 12 00 0F 00 05 6F 4D 2D 00 06 03 6F 0D 14 00 06 0B 12 01 28 C6 11 00 06 03 28 ED 00 00 06 04 28 EE 00 00 06 06 2A 13 30 03 00 30 00 00 00 0C 00 00 11 15 0A 16  .......oM-...o.......(.....(.....(.....*.0..0..........
00063B 0B 2B 17 02 7E 6F 01 00 04 07 9A 6F A2 14 00 06 2C 04 07 0A 2B 0E 07 17 58 0B 07 7E 6F 01 00 04 8E 69 32 DF 06 15 33 03 1F 12 0A 06 2A 13 30 04 00 38 01 00 00 00 00  .+..~o.....o....,...+...X..~o....i2...3.....*.0..8.....
000672 00 00 1F 17 8D 48 01 00 02 25 16 D0 D9 00 00 02 28 11 14 00 06 A2 25 17 D0 5C 01 00 02 28 11 14 00 06 A2 25 18 D0 B2 00 00 02 28 11 14 00 06 A2 25 19 D0 B6 00 00 02  .....H...%......(.....%..\...(......%.....(.....%......
0006A9 28 11 14 00 06 A2 25 1A D0 3C 01 00 02 28 11 14 00 06 A2 25 1B D0 B4 00 00 02 28 11 14 00 06 A2 25 1C D0 FA 00 00 02 28 11 14 00 06 A2 25 1D D0 50 01 00 02 28 11 14  (.....%..<...(.....%......(.....%......(.....%..P...(..
0006E0 00 06 A2 25 1E D0 FB 00 00 02 28 11 14 00 06 A2 25 1F 09 D0 51 01 00 02 28 11 14 00 06 A2 25 1F 0A D0 FC 00 00 02 28 11 14 00 06 A2 25 1F 0B D0 52 01 00 02 28 11 14  ...%......(.....%...Q...(.....%.......(.....%...R...(..
000717 00 06 A2 25 1F 0C D0 3F 01 00 02 28 11 14 00 06 A2 25 1F 0D D0 D7 00 00 02 28 11 14 00 06 A2 25 1F 0E D0 73 00 00 02 28 11 14 00 06 A2 25 1F 0F D0 5C 01 00 02 28 11  ...%...?...(.....%.......(.....%...s...(.....%...\...(.
00074E 14 00 06 A2 25 1F 10 D0 7D 00 00 02 28 11 14 00 06 A2 25 1F 11 D0 43 01 00 02 28 11 14 00 06 A2 25 1F 12 D0 3D 00 00 02 28 11 14 00 06 A2 25 1F 13 D0 D3 00 00 02 28  ....%...}...(.....%...C...(.....%...=...(.....%.......(
000785 11 14 00 06 A2 25 1F 15 D0 DD 05 00 02 28 11 14 00 06 A2 25 1F 16 D0 D2 00 00 02 28 11 14 00 06 A2 80 6F 01 00 04 2A 13 30 01 00 6A 00 00 00 00 00 00 00 7E 81 01 00  .....%.......(.....%.......(......o...*.0..j.......~...
0007BC 04 28 0A 01 00 06 80 71 01 00 04 7E 82 01 00 04 28 0A 01 00 06 80 72 01 00 04 7E 80 01 00 04 28 0A 01 00 06 80 73 01 00 04 7E 83 01 00 04 28 0A 01 00 06 80 74 01 00  .(.....q...~....(.....r...~....(......s...~....(.....t..
0007F3 04 7E 84 01 00 04 28 0A 01 00 06 80 75 01 00 04 7E 85 01 00 04 28 0A 01 00 06 80 76 01 00 04 7E 86 01 00 04 28 0A 01 00 06 80 77 01 00 04 2A 00 00 13 30 05 00 96 01  .~....(.....u...~....(.....v...~....(.....w...*...0....
00082A 00 00 0D 00 00 11 02 2D 0B 72 B7 00 00 70 73 A2 09 00 06 7A 02 1F 5C 6F 10 05 00 06 0B 07 15 2E 15 02 16 07 6F DF 04 00 06 28 4B 2D 00 06 6F 34 05 00 06 0A 2B 0C 02  .......-.r...ps....z..\o.............o....(K-..o4....+..
000861 28 4B 2D 00 06 6F 34 05 00 06 0A 14 0C 06 28 5F 68 00 06 0D 09 20 A6 61 69 41 35 20 09 20 3B 71 91 07 3B 98 00 00 00 09 20 74 2A 40 1B 2E 51 09 20 A6 61 69 41 2E 5B  (K-..o4.......(_h.... .aiA5 . ;q..;..... t*@..Q. .aiA.[
000898 38 DE 00 00 00 09 20 D6 CE 7A 5D 35 15 09 20 E9 EE 72 47 2E 55 09 20 D6 CE 7A 5D 2E 1A 38 C1 00 00 00 09 20 0B C5 50 A8 2E 6D 09 20 78 D9 E4 D3 2E 47 38 AC 00 00 00  8..... ..z]5.. ..rG.U. ..z]..8..... ..P..m. x....G8....
0008CF 06 72 C7 00 00 70 28 C5 04 00 06 2D 62 38 9A 00 00 00 06 72 EB 00 00 70 28 C5 04 00 06 2D 58 38 88 00 00 00 06 72 11 01 00 70 28 C5 04 00 06 2D 4E 2B 79 06 72 35 01  .r...p(....-b8.....r...p(....-X8.....r...p(....-N+y.r5.
000906 00 70 28 C5 04 00 06 2D 47 2B 6A 06 72 4B 01 00 70 28 C5 04 00 06 2D 40 2B 5B 06 72 77 01 00 70 28 C5 04 00 06 2D 39 2B 4C 06 72 9F 01 00 70 28 C5 04 00 06 2D 32 2B  .p(....-G+j.rK..p(....-@+[.rw..p(....-9+L.r...p(....-2+
00093D 3D 7E 71 01 00 04 0C 2B 53 7E 72 01 00 04 0C 2B 4B 7E 73 01 00 04 0C 2B 43 7E 74 01 00 04 0C 2B 3B 7E 75 01 00 04 0C 2B 33 7E 76 01 00 04 0C 2B 2B 7E 86 01 00 04 28  =~q....+S~r....+K~s....+C~t....+;~u....+3~v....++~....(
000974 0A 01 00 06 0C 2B 1E 72 BB 01 00 70 17 8D 3D 00 00 02 25 16 72 B7 00 00 70 A2 28 64 0E 00 06 73 99 09 00 06 7A 07 15 2E 09 07 02 6F D2 04 00 06 33 09 03 7E 88 02 00  .....+.r...p..=...%.r...p.(d...s....z......o....3..~...
0009AB 04 51 2B 15 03 02 07 17 58 02 6F D2 04 00 06 07 59 17 59 6F DF 04 00 06 51 08 2A 00 00 1B 30 03 00 2B 00 00 00 0E 00 00 11 02 12 00 28 F1 00 00 06 0B 07 06 6F 15 01  .Q+.....X.o.....Y.Yo....Q.*...0..+..........(.......o..
0009E2 00 06 0C 08 2D 02 14 2A 00 08 03 04 6F 22 01 00 06 0D DE 07 08 6F F7 00 00 06 DC 09 2A 00 01 10 00 00 02 00 17 00 0B 22 00 07 00 00 00 00 2A 02 03 04 16 28 F4 00 00  ....-..*....o"........o......*.........."......*....(...
000A19 06 2A 00 1B 30 04 00 24 00 00 00 0F 00 00 11 02 12 00 28 F1 00 00 06 0B 07 06 6F FB 00 00 06 0C 08 03 04 05 6F 2D 01 00 06 DE 07 08 6F F7 00 00 06 DC 2A 01 10 00 00  .*..0..$..........(.......o.........o-......o.....*....
000A50 02 00 11 00 0B 1C 00 07 00 00 00 00 36 02 03 04 16 16 16 05 28 F6 00 00 06 2A 00 00 13 30 03 00 78 00 00 00 00 00 00 00 02 28 FB 0F 00 06 02 03 FE 13 7D 8E 01 00 04  ............6.......(....*...0..x........(........}....
000A87 02 72 E7 01 00 70 FE 13 7D 90 01 00 04 02 0E 04 FE 13 7D 91 01 00 04 02 0E 06 FE 13 7D 93 01 00 04 05 2C 12 02 02 FE 13 7B 8F 01 00 04 18 60 FE 13 7D 8F 01 00 04 04  .r...p..}.........}..........}.....,.......{.....`..}.....
000ABE 2C 12 02 02 FE 13 7B 8F 01 00 04 1A 60 FE 13 7D 8F 01 00 04 0E 05 2C 12 02 02 FE 13 7B 8F 01 00 04 1E 60 FE 13 7D 8F 01 00 04 0E 06 28 4C 01 00 06 2A 22 02 17 28 F8  ,.....{.....`..}......,......{....`..}......(L...*"..(.
000AF5 00 00 06 2A 00 00 00 1B 30 02 00 45 00 00 00 00 00 00 00 02 FE 13 7B 8E 01 00 04 2C 3A 02 28 27 01 00 06 2D 1C 02 FE 13 7B 8E 01 00 04 6F C0 60 00 06 DE 23 26 DE 20  ...*....0..E..........{....,:.('...-....{....o.`...#&. 
```

Reverse Engineering in Android

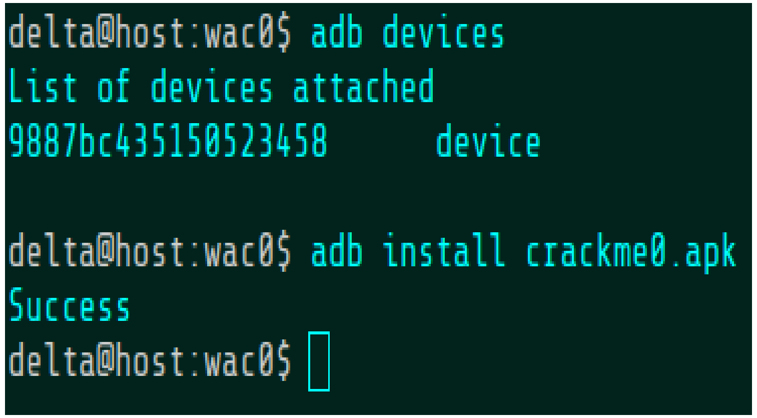
apktool, jadx, & mitmproxy

Android Debug Bridge

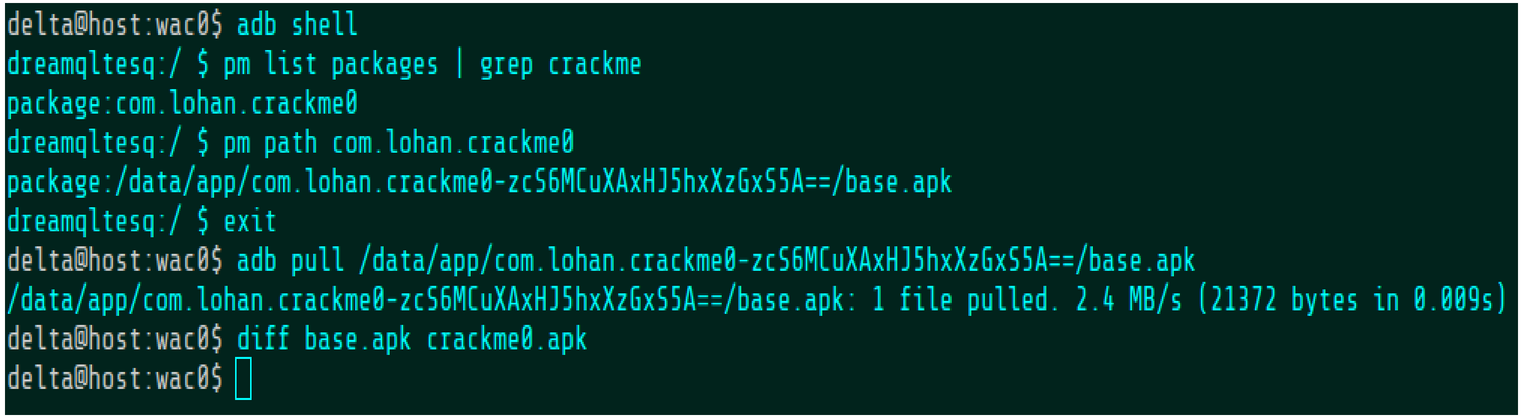
ADB is the Android hacker’s bread and butter.

- Powerful command shell
- Connects your laptop to the Android file system
- Lots of special features

Let’s install an app and take a copy of it off the phone!



```
delta@host:wac0$ adb devices
List of devices attached
9887bc435150523458      device

delta@host:wac0$ adb install crackme0.apk
Success
delta@host:wac0$
```



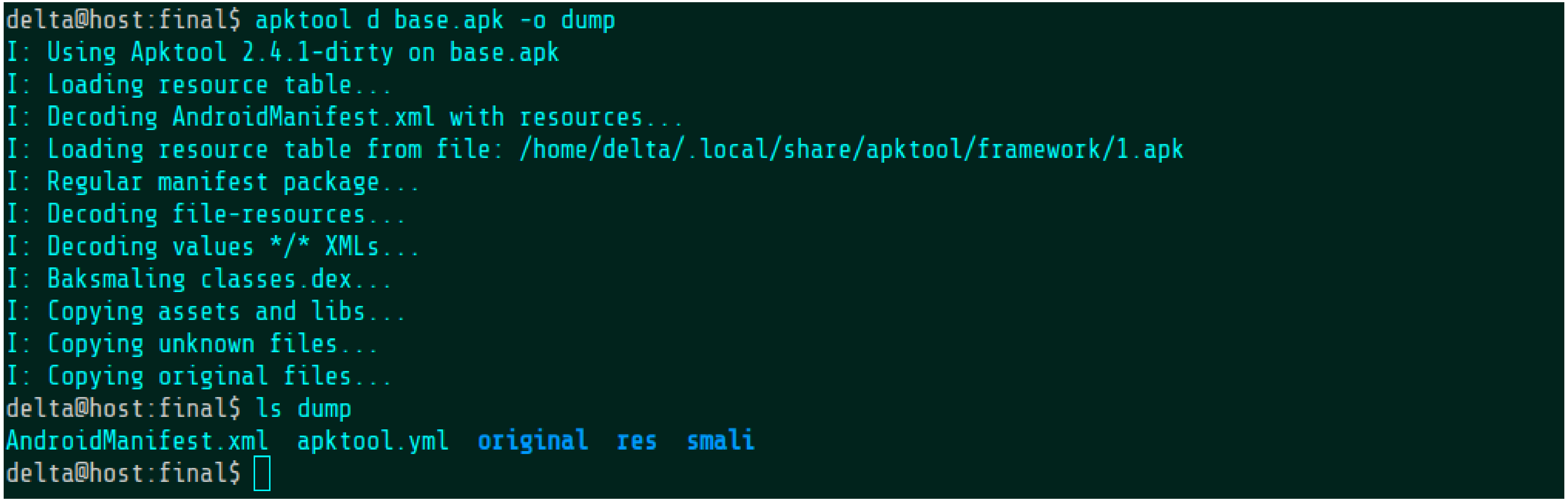
```
delta@host:wac0$ adb shell
dreamqltesq:/ $ pm list packages | grep crackme
package:com.lohan.crackme0
dreamqltesq:/ $ pm path com.lohan.crackme0
package:/data/app/com.lohan.crackme0-zcS6MCuXAxHJ5hxXzGxS5A==/base.apk
dreamqltesq:/ $ exit
delta@host:wac0$ adb pull /data/app/com.lohan.crackme0-zcS6MCuXAxHJ5hxXzGxS5A==/base.apk
/data/app/com.lohan.crackme0-zcS6MCuXAxHJ5hxXzGxS5A==/base.apk: 1 file pulled. 2.4 MB/s (21372 bytes in 0.009s)
delta@host:wac0$ diff base.apk crackme0.apk
delta@host:wac0$
```

apktool

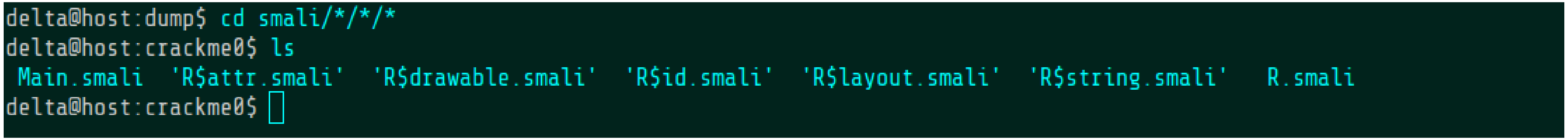
apktool is a disassembler. It lets you read, edit, and repack Android apps. Here’s how it works:

- Android apps are packaged as .apk (Android Package) files.
- These are basically just zip archives.
- Most of the code is stored in a file called classes.dex.
- Dex (Dalvik Executable) is a format similar to .class but made to run more efficiently on mobile platforms.
- apktool converts the .dex code into .smali files.
- Smali is basically the Android version of assembly language. It’s human-readable bytecode.

apktool in action!



```
delta@host:final$ apktool d base.apk -o dump
I: Using Apktool 2.4.1-dirty on base.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/delta/.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...
delta@host:final$ ls dump
AndroidManifest.xml  apktool.yml  original  res  smali
delta@host:final$ 
```

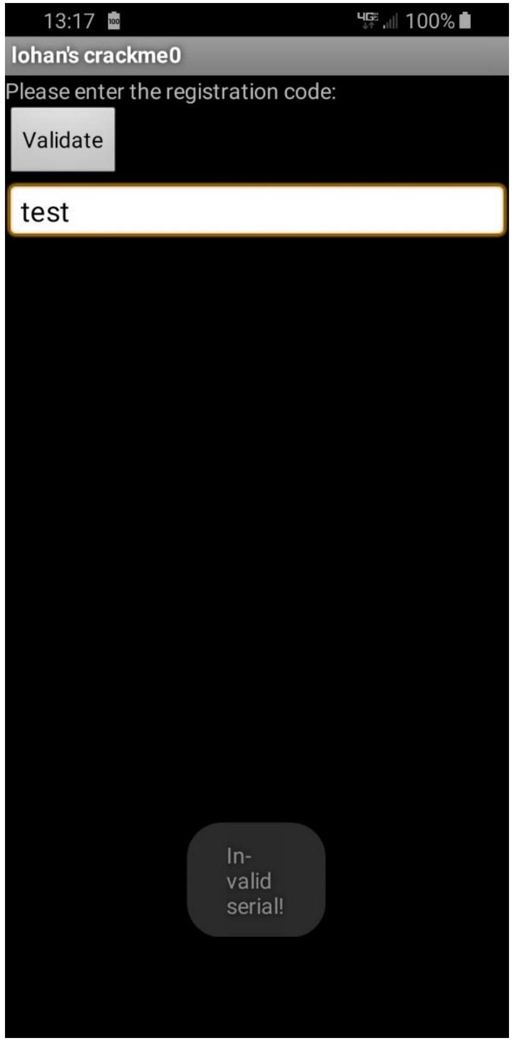


```
delta@host:dump$ cd smali/*/*/*
delta@host:crackme0$ ls
 Main.smali  'R$attr.smali'  'R$drawable.smali'  'R$id.smali'  'R$layout.smali'  'R$string.smali'   R.smali
delta@host:crackme0$ 
```

The App

Here’s the example app being run.

Note the “Invalid serial!” message.



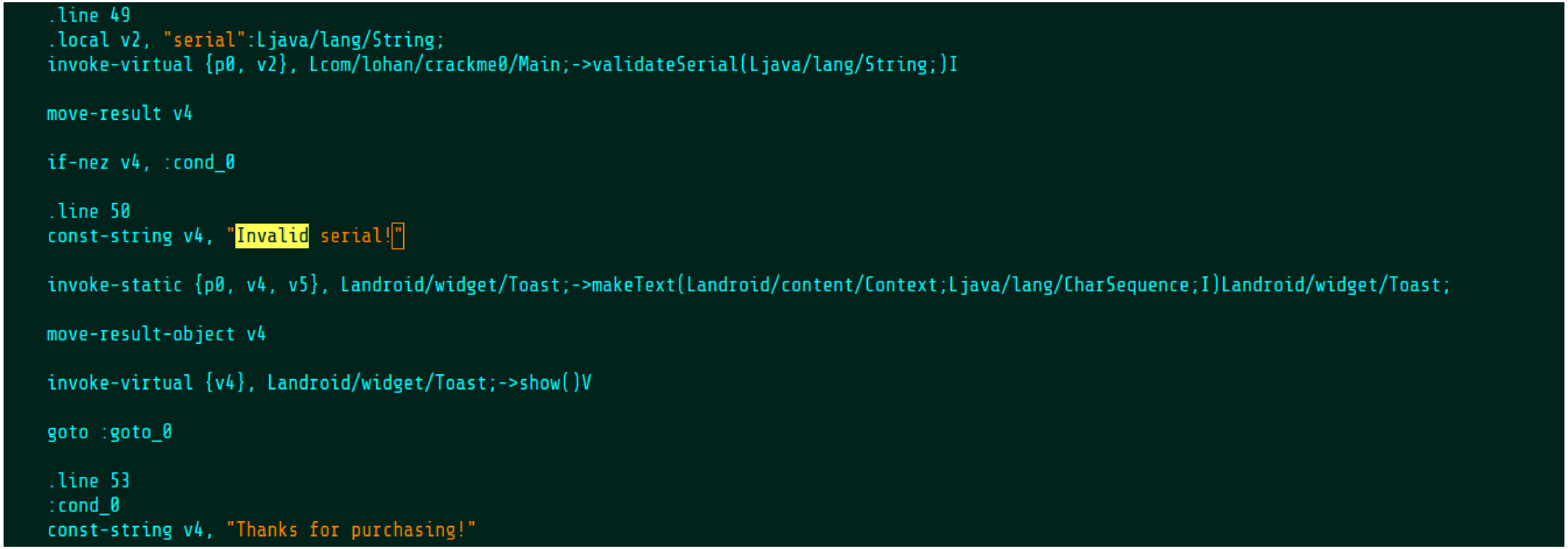
Please enter the registration code:

test

In-
valid
serial!

It’s a Smali world after all!

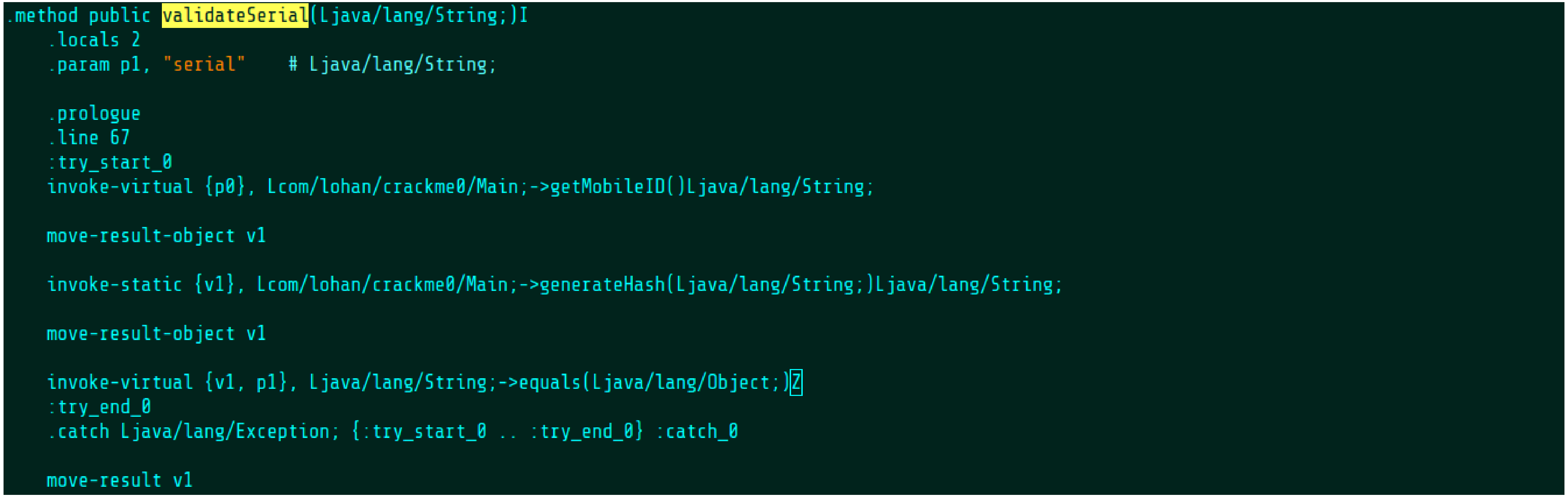
In Main.smali, you can see the serial check by searching for the error message. Note the validateSerial(String) function and the “if-nez”.



```
.line 49
.local v2, "serial":Ljava/lang/String;
invoke-virtual {p0, v2}, Lcom/lohan/crackme0/Main;->validateSerial(Ljava/lang/String;)I

move-result v4

if-nez v4, :cond_0

.line 50
const-string v4, "Invalid serial!"

invoke-static {p0, v4, v5}, Landroid/widget/Toast;->makeText(Landroid/content/Context;Ljava/lang/CharSequence;I)Landroid/widget/Toast;

move-result-object v4

invoke-virtual {v4}, Landroid/widget/Toast;->show()V

goto :goto_0

.line 53
:cond_0
const-string v4, "Thanks for purchasing!"
```

Making Smali talk

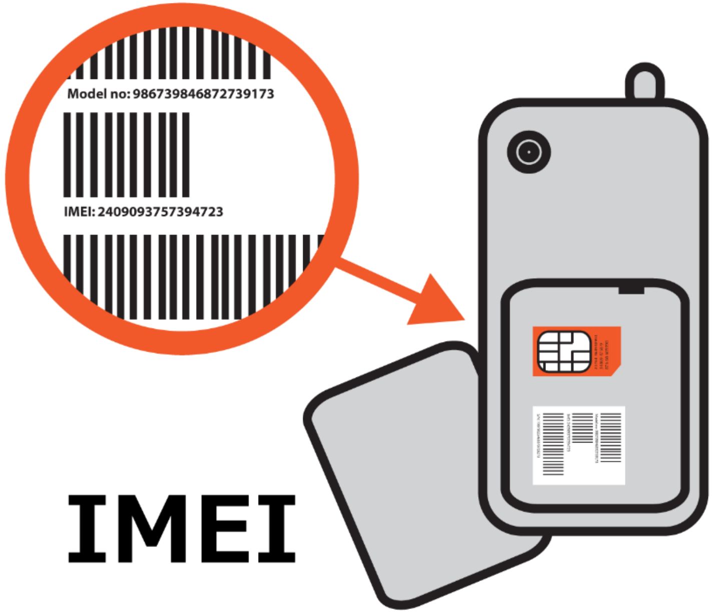
validateSerial() is very simple. It just gets the phone’s IMEI number performs a message digest on it.



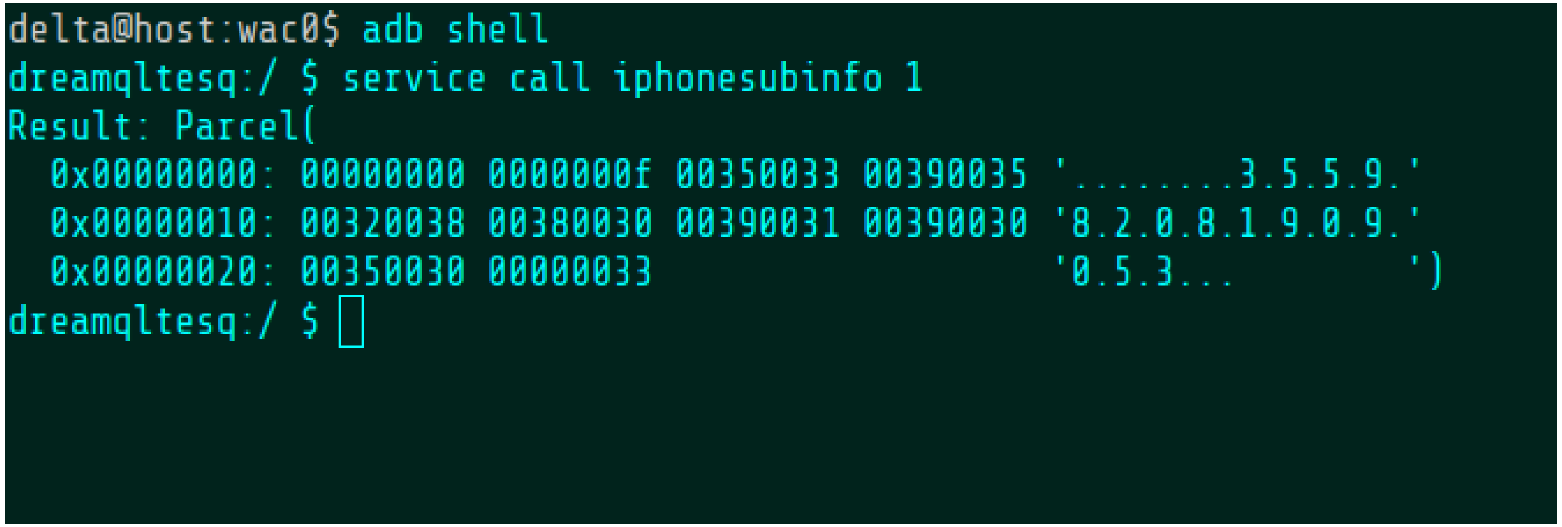
```
.method public validateSerial(Ljava/lang/String;)I
    .locals 2
    .param p1, "serial"    # Ljava/lang/String;

    .prologue
    .line 67
    :try_start_0
    invoke-virtual {p0}, Lcom/lohan/crackme0/Main;->getMobileID()Ljava/lang/String;

    move-result-object v1

    invoke-static {v1}, Lcom/lohan/crackme0/Main;->generateHash(Ljava/lang/String;)Ljava/lang/String;

    move-result-object v1

    invoke-virtual {v1, p1}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z
    :try_end_0
    .catch Ljava/lang/Exception; {:try_start_0 .. :try_end_0} :catch_0

    move-result v1
```

Side note: IMEI

- International Mobile Equipment Identity
- Uniquely identifies a mobile device at the hardware level regardless of the assigned phone number or SIM card
- Used by ISPs to track stolen phones (or just phones the government is interested in)



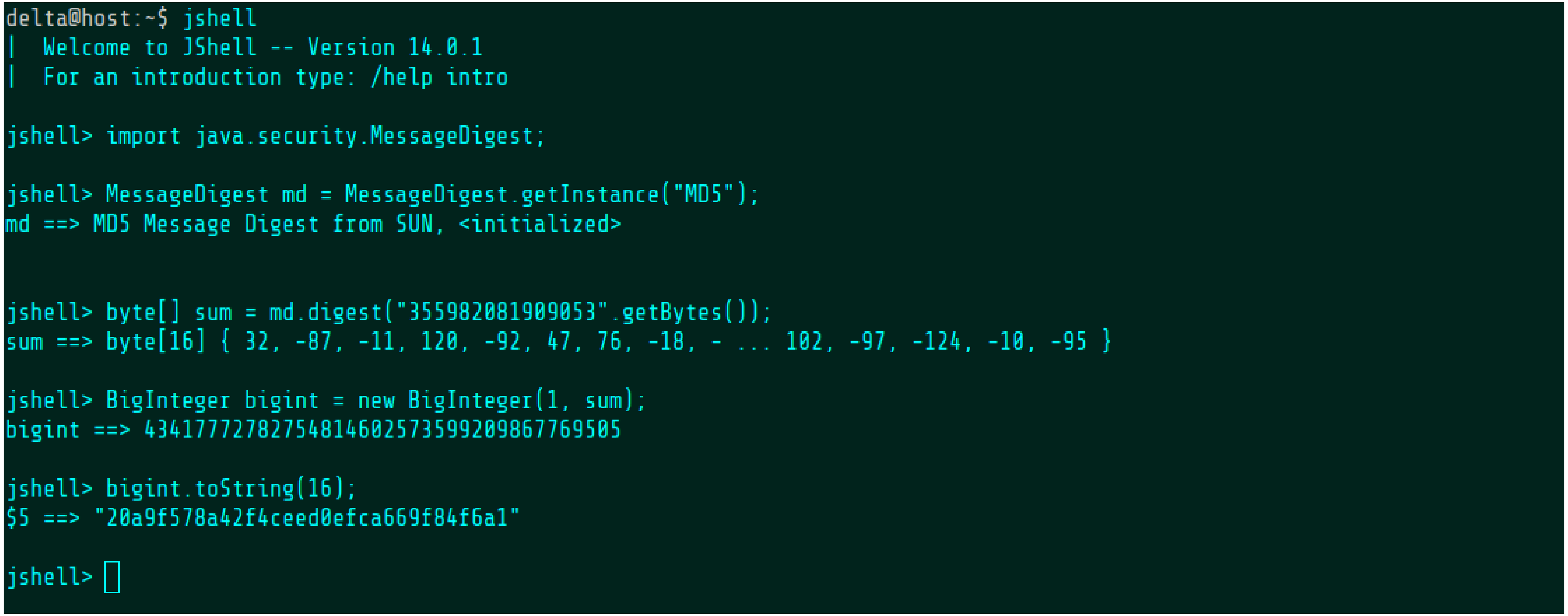
Getting the IMEI via ADB



```
delta@host:wac0$ adb shell
dreamqltesq:/ $ service call iphonesubinfo 1
Result: Parcel(
  0x00000000: 00000000 0000000f 00350033 00390035 '........3.5.5.9.'
  0x00000010: 00320038 00380030 00390031 00390030 '8.2.0.8.1.9.0.9.'
  0x00000020: 00350030 00000033                   '0.5.3...        ')
dreamqltesq:/ $ 
```

Doing a Keygen the jshell Way

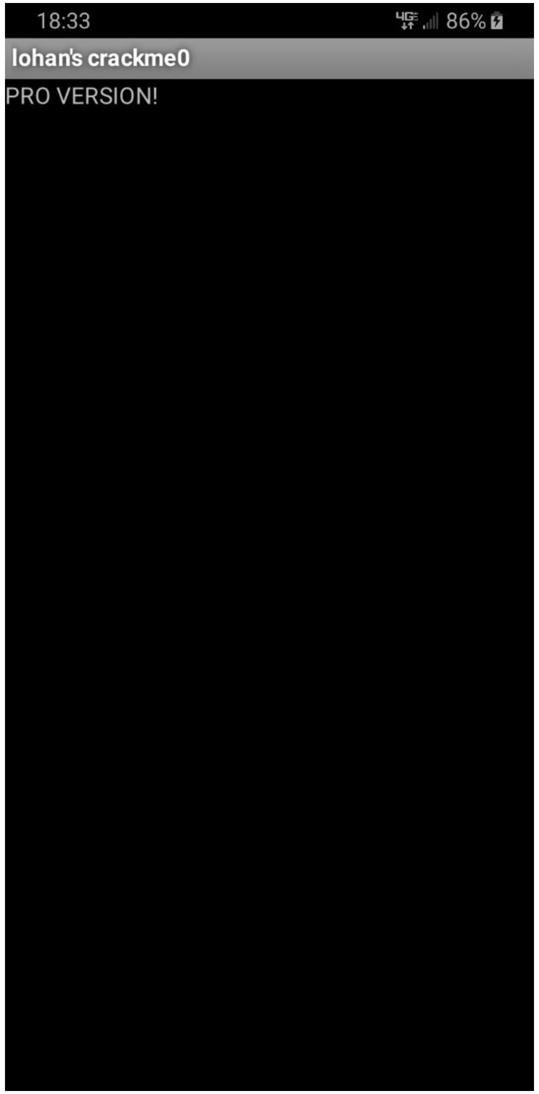
This is basically the same code seen in the Smali.



```
delta@host:~$ jshell
|  Welcome to JShell -- Version 14.0.1
|  For an introduction type: /help intro

jshell> import java.security.MessageDigest;

jshell> MessageDigest md = MessageDigest.getInstance("MD5");
md ==> MD5 Message Digest from SUN, <initialized>


jshell> byte[] sum = md.digest("355982081909053".getBytes());
sum ==> byte[16] { 32, -87, -11, 120, -92, 47, 76, -18, - ... 102, -97, -124, -10, -95 }

jshell> BigInteger bigint = new BigInteger(1, sum);
bigint ==> 43417772782754814602573599209867769505

jshell> bigint.toString(16);
$5 ==> "20a9f578a42f4ceed0efca669f84f6a1"

jshell> 
```

Success!

Houston, we have pwnage!



18:33

86%

Iohan's crackme0

PRO VERSION!

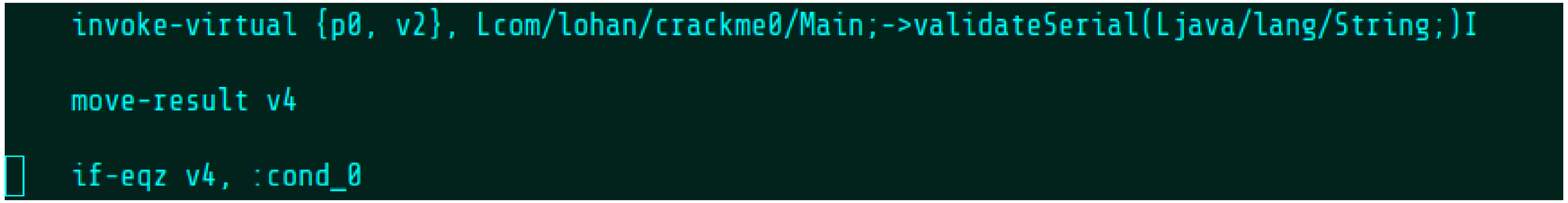
BUT WAIT
THERE'S MORE

Additional Techniques

- Patching
- Decompilation
- Traffic capture (man-in-the-middle)

Patching Android Apps

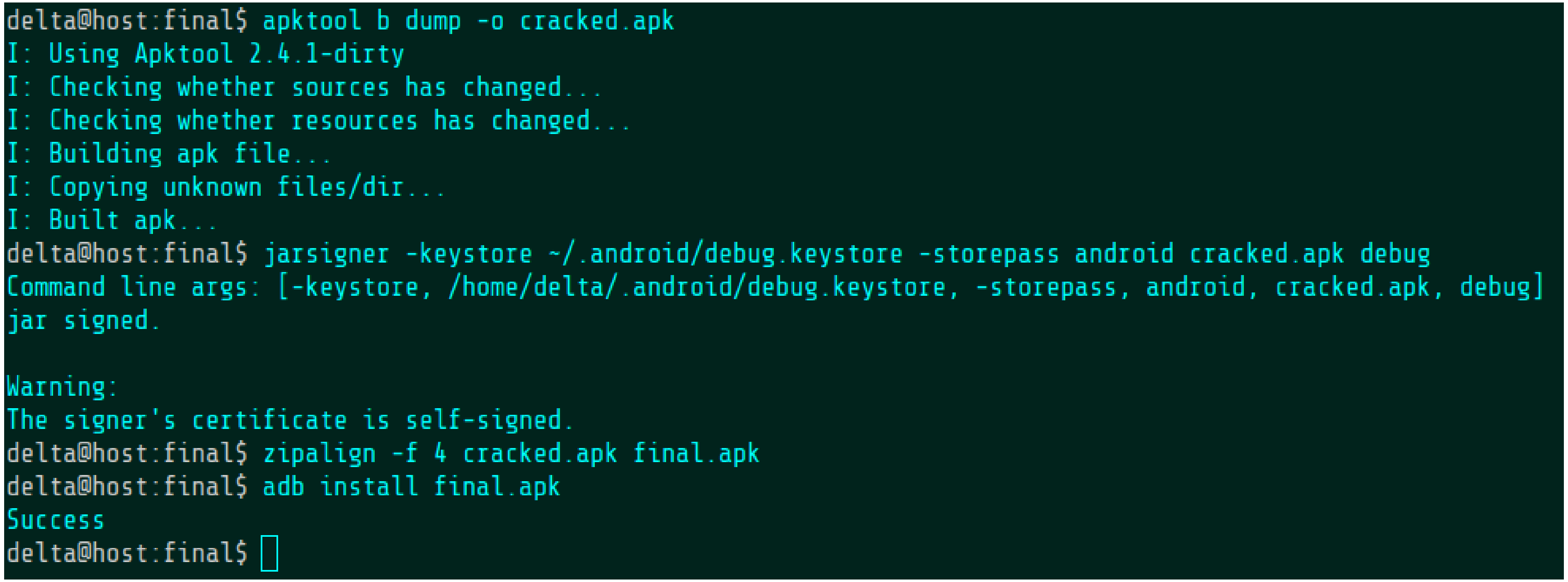
Remember that “if-nez” line back in the Smali code? Let’s reverse it – literally!



```
    invoke-virtual {p0, v2}, Lcom/lohan/crackme0/Main;->validateSerial(Ljava/lang/String;)I

    move-result v4

    if-eqz v4, :cond_0
```

Patching Android Apps

Now we just need to repack, resign, and zip-align.



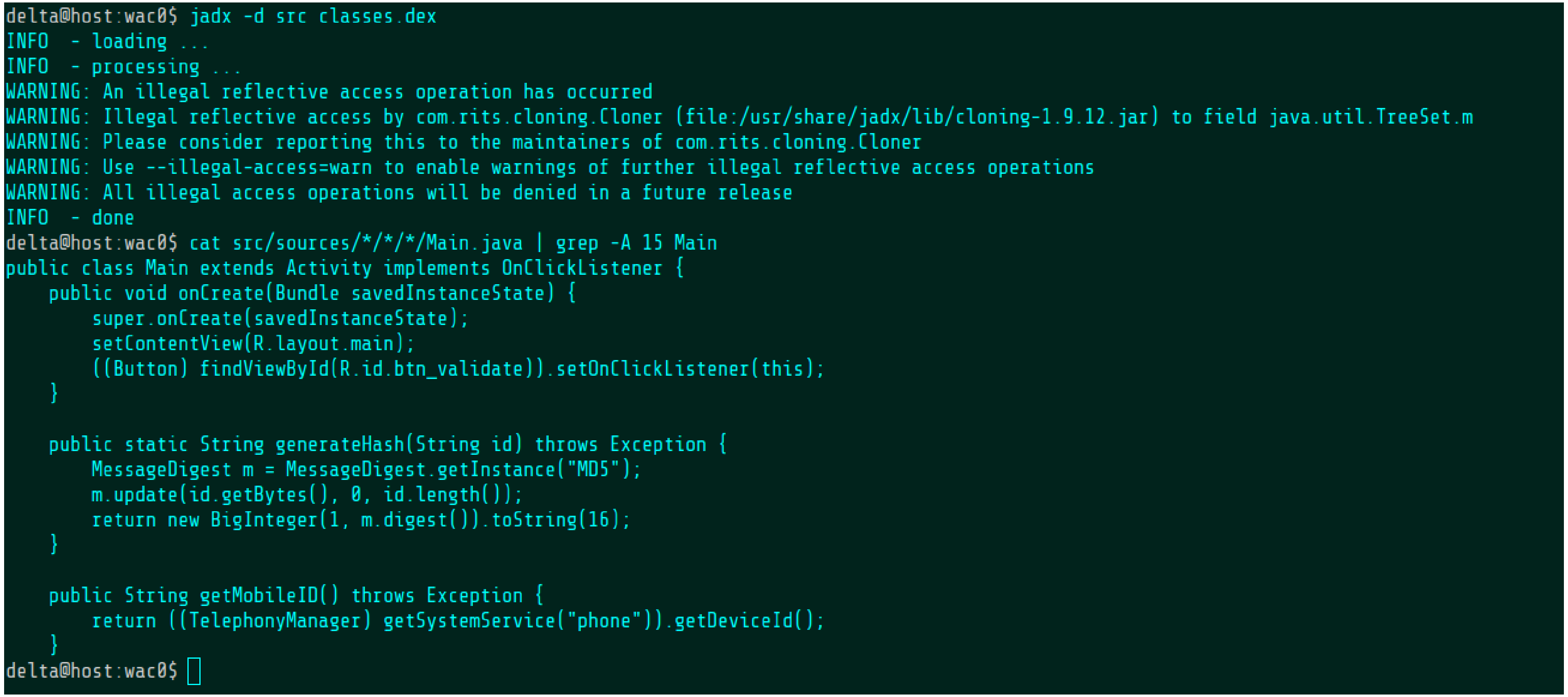
```
delta@host:final$ apktool b dump -o cracked.apk
I: Using Apktool 2.4.1-dirty
I: Checking whether sources has changed...
I: Checking whether resources has changed...
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
delta@host:final$ jarsigner -keystore ~/.android/debug.keystore -storepass android cracked.apk debug
Command line args: [-keystore, /home/delta/.android/debug.keystore, -storepass, android, cracked.apk, debug]
jar signed.

Warning:
The signer's certificate is self-signed.
delta@host:final$ zipalign -f 4 cracked.apk final.apk
delta@host:final$ adb install final.apk
Success
delta@host:final$
```

Works just as well as the previous technique. There are often many paths to a successful crack.

Decompiling Android Apps

Smali is fairly readable but wouldn’t plain Java be even nicer? Well, with jadx you can have both!

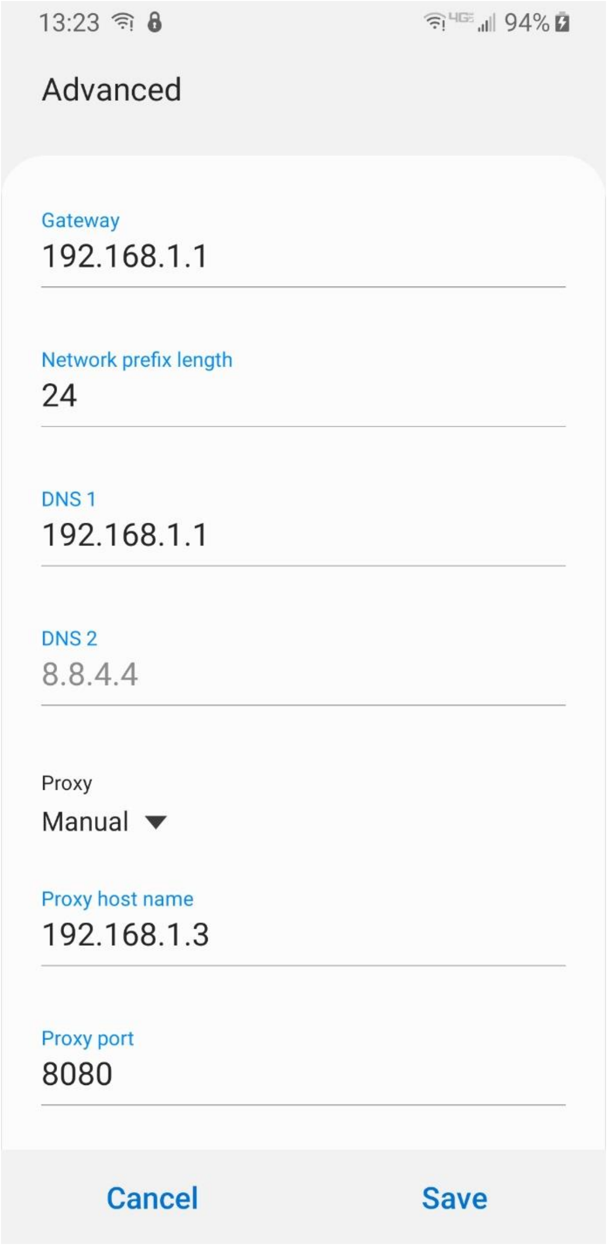
Note: jadx’s output can’t actually be recompiled, so use it as a reference.



```
delta@host:wac0$ jadx -d src classes.dex
INFO  - loading ...
INFO  - processing ...
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.rits.cloning.Cloner (file:/usr/share/jadx/lib/cloning-1.9.12.jar) to field java.util.TreeSet.m
WARNING: Please consider reporting this to the maintainers of com.rits.cloning.Cloner
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
INFO  - done
delta@host:wac0$ cat src/sources/*/*/*/Main.java | grep -A 15 Main
public class Main extends Activity implements OnClickListener {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ((Button) findViewById(R.id.btn_validate)).setOnClickListener(this);
    }

    public static String generateHash(String id) throws Exception {
        MessageDigest m = MessageDigest.getInstance("MD5");
        m.update(id.getBytes(), 0, id.length());
        return new BigInteger(1, m.digest()).toString(16);
    }

    public String getMobileID() throws Exception {
        return ((TelephonyManager) getSystemService("phone")).getDeviceId();
    }
delta@host:wac0$ 
```

Capturing Android Traffic

Step one is to set up a proxy on the WiFi network of your choice. The proxy config will connect to your laptop running mitmproxy.



Gateway
192.168.1.1

Network prefix length
24

DNS 1
192.168.1.1

DNS 2
8.8.4.4

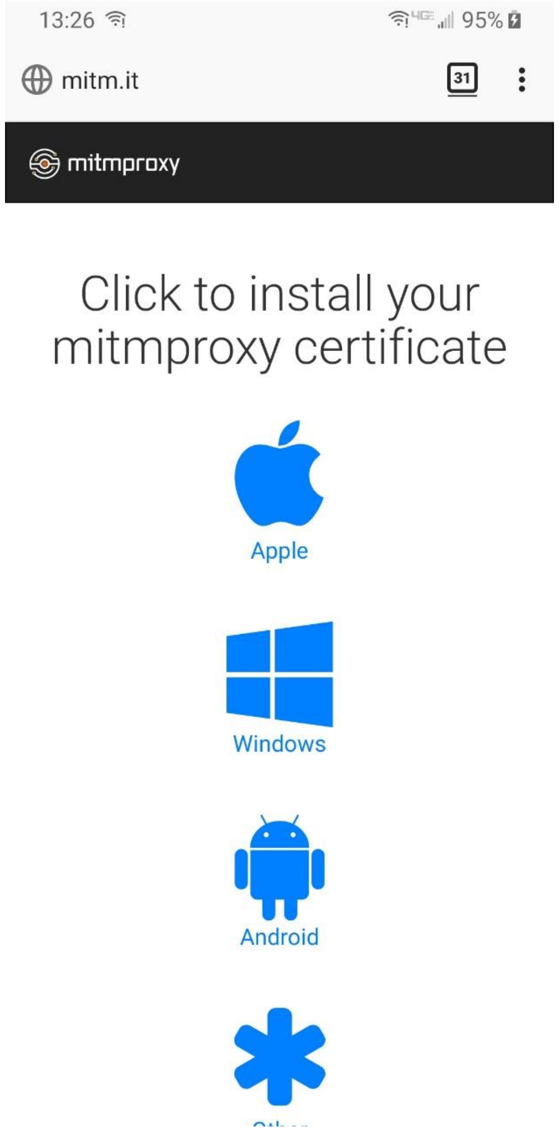
Proxy
Manual

Proxy host name
192.168.1.3

Proxy port
8080

Capturing Android Traffic

Next, fire up mitmproxy and browse (on Android) to mitm.it. Install the certificate.



13:26 95%

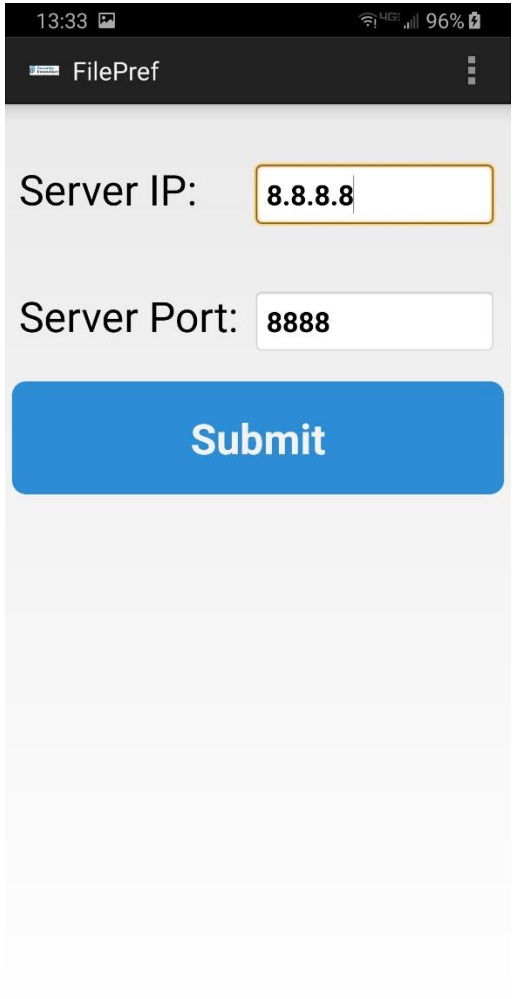
mitm.it

mitmproxy

Click to install your mitmproxy certificate

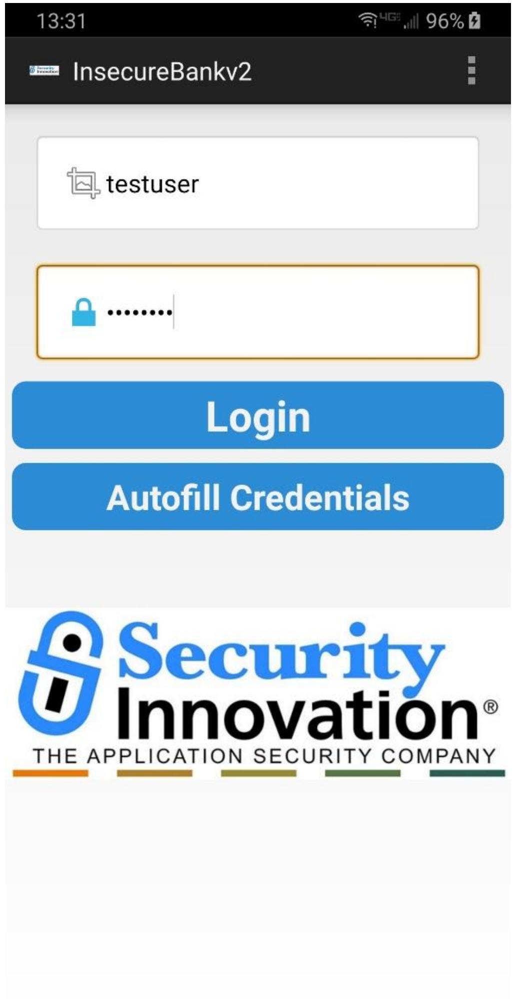
Capturing Traffic on Android

I’m using InsecureBankv2 for the demo. I’ve set it to try and send the login request to Google’s famous 8.8.8.8 DNS server.



Server IP: 8.8.8.8

Server Port: 8888

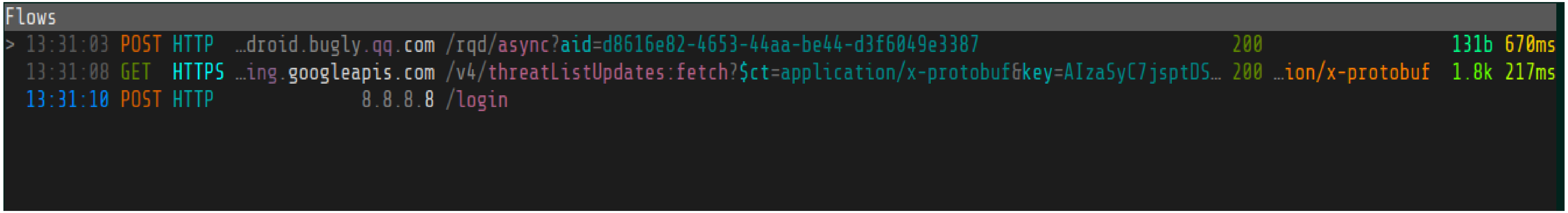


testuser

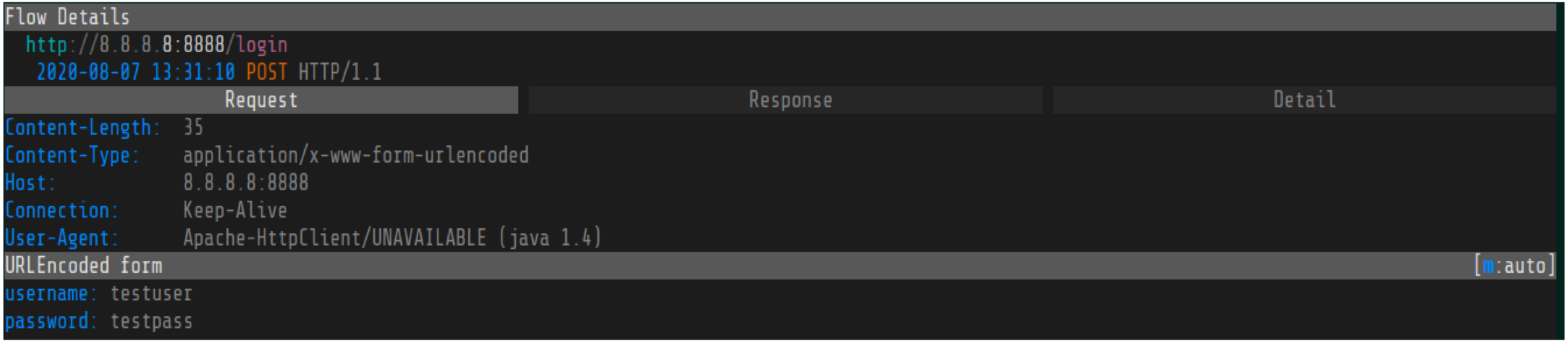
••••••••

Capturing Traffic on Android

Boom! Plaintext creds in full view!



```
Flows
> 13:31:03 POST HTTP  …droid.bugly.qq.com /rqd/async?aid=d8616e82-4653-44aa-be44-d3f6049e3387             200                    131b 670ms
  13:31:08 GET  HTTPS …ing.googleapis.com /v4/threatListUpdates:fetch?$ct=application/x-protobuf&key=AIzaSyC7jsptDS… 200 …ion/x-protobuf  1.8k 217ms
  13:31:10 POST HTTP             8.8.8.8 /login
```



```
Flow Details
  http://8.8.8.8:8888/login
   2020-08-07 13:31:10 POST HTTP/1.1
            Request                      Response                      Detail
Content-Length:   35
Content-Type:     application/x-www-form-urlencoded
Host:             8.8.8.8:8888
Connection:       Keep-Alive
User-Agent:       Apache-HttpClient/UNAVAILABLE (java 1.4)
URLEncoded form                                                                                    [m:auto]
username: testuser
password: testpass
```

Reverse Engineering with Angr

What the heck is it?

- A tool for “concolic analysis” and “symbolic execution”.

What the heck does that mean?

- Breaks the program into a set of logical symbols
- Uses the dark sorceries of discrete mathematics to find the input that satisfies a set of constraints
- Simulates various execution paths without actually running the program

PENGUIN MODERN : 46

JORGE LUIS BORGES

THE GARDEN OF FORKING PATHS

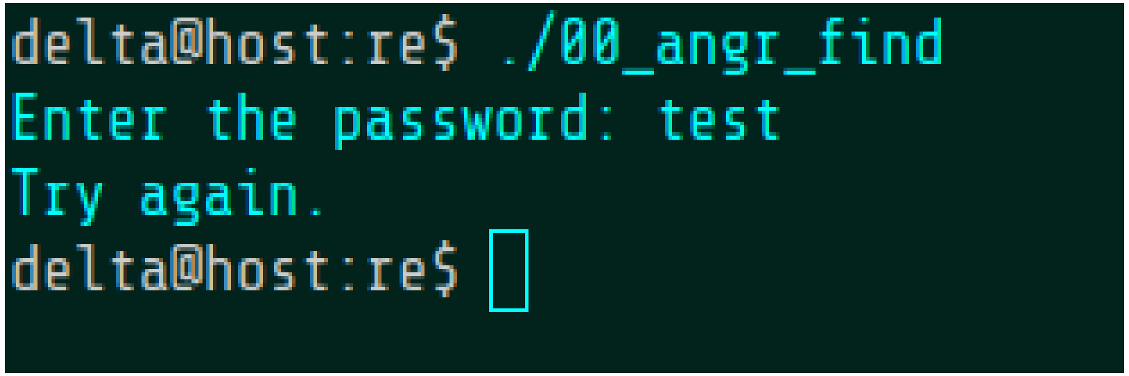
What the heck is it?

- Invented by legendary CTF team Shellphish.
- Can be further automated to hack complex, unknown binaries in seconds with zero user input.
- I’m not joking. This has literally already been done. Shellphish won a DARPA challenge that way.

Infinite realities, Morty! In one of them, this program is already hacked!

Angr Management

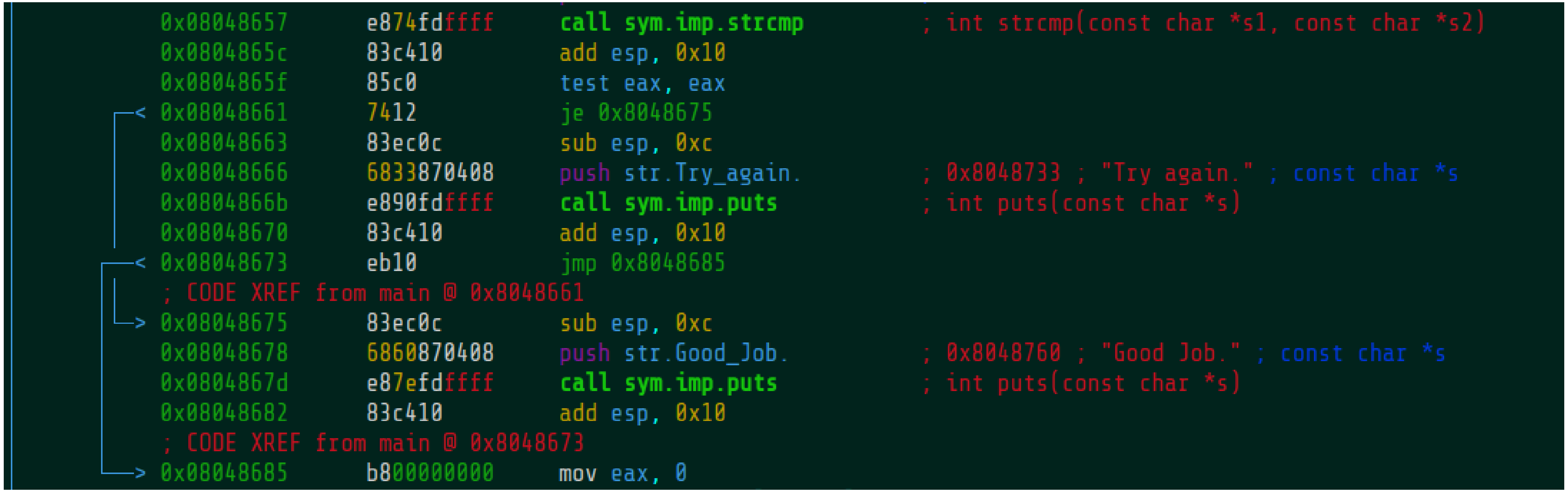
This is program that will be attacked.



```
delta@host:re$ ./00_angr_find
Enter the password: test
Try again.
delta@host:re$ 
```

Angr Management

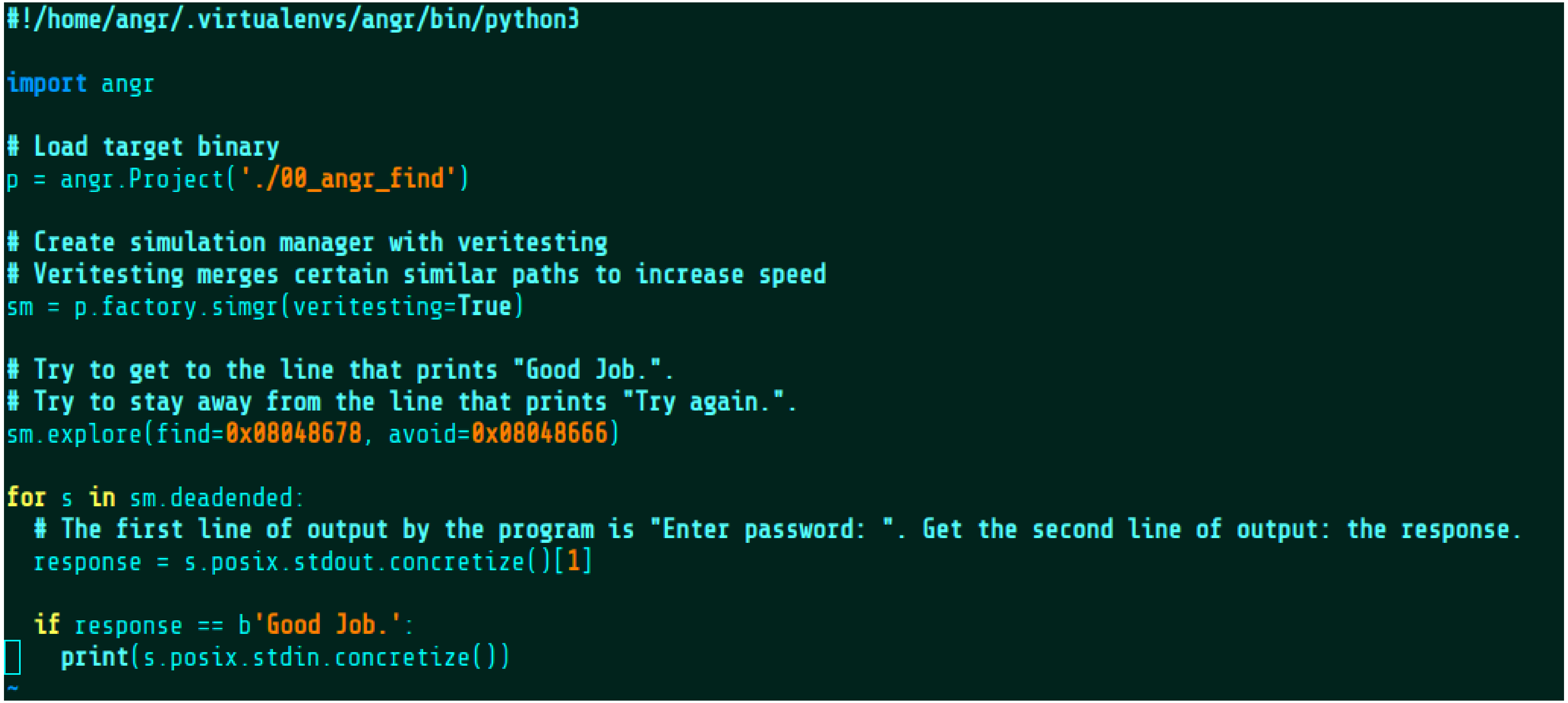
The first step is to find the memory addresses you want to hit and those you want to miss. Note the “Try again.” and “Good Job.” messages with associated control flow arrows.



```
        0x08048657      e874fdffff     call sym.imp.strcmp         ; int strcmp(const char *s1, const char *s2)
        0x0804865c      83c410         add esp, 0x10
        0x0804865f      85c0           test eax, eax
     ┌─< 0x08048661      7412           je 0x8048675
     │   0x08048663      83ec0c         sub esp, 0xc
     │   0x08048666      6833870408     push str.Try_again.         ; 0x8048733 ; "Try again." ; const char *s
     │   0x0804866b      e890fdffff     call sym.imp.puts           ; int puts(const char *s)
     │   0x08048670      83c410         add esp, 0x10
    ┌──< 0x08048673      eb10           jmp 0x8048685
    ││   ; CODE XREF from main @ 0x8048661
    │└─> 0x08048675      83ec0c         sub esp, 0xc
    │    0x08048678      6860870408     push str.Good_Job.          ; 0x8048760 ; "Good Job." ; const char *s
    │    0x0804867d      e87efdffff     call sym.imp.puts           ; int puts(const char *s)
    │    0x08048682      83c410         add esp, 0x10
    │    ; CODE XREF from main @ 0x8048673
    └──> 0x08048685      b800000000     mov eax, 0
```

Angr Management

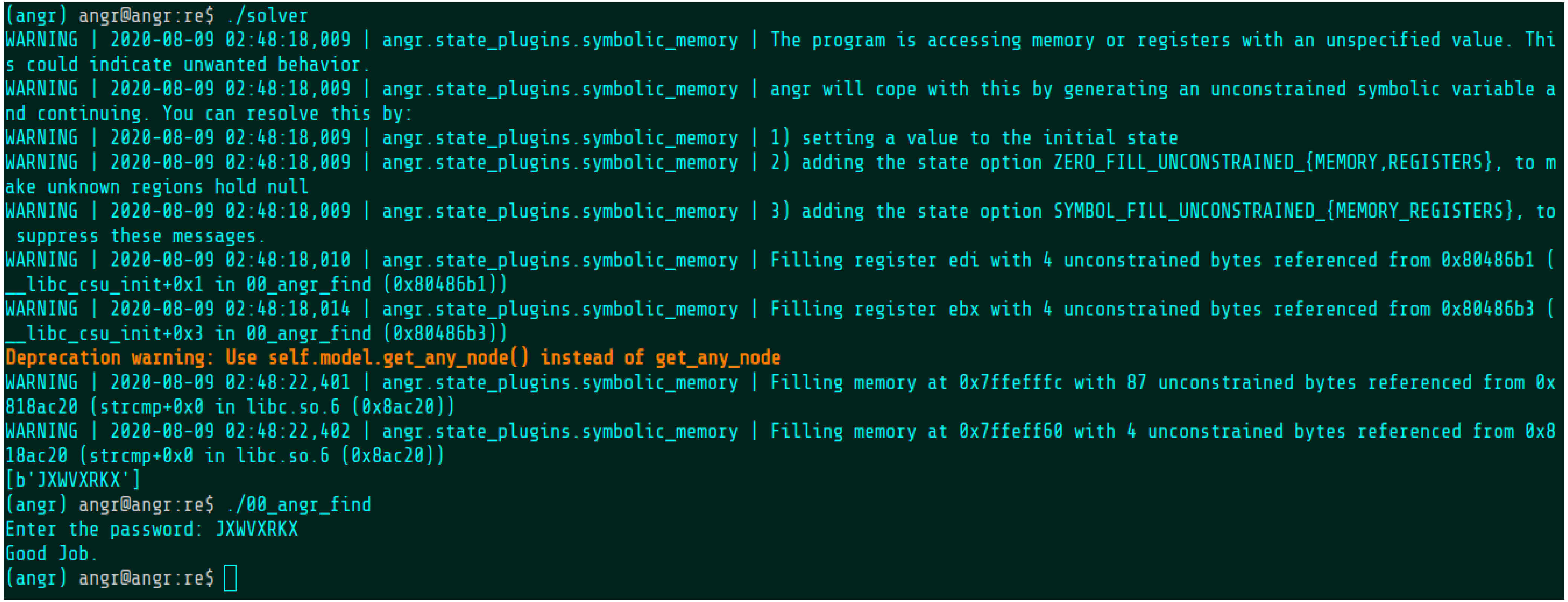
Now you just need a script like this. Nothing too crazy, right?



```
#!/home/angr/.virtualenvs/angr/bin/python3

import angr

# Load target binary
p = angr.Project('./00_angr_find')

# Create simulation manager with veritesting
# Veritesting merges certain similar paths to increase speed
sm = p.factory.simgr(veritesting=True)

# Try to get to the line that prints "Good Job.".
# Try to stay away from the line that prints "Try again.".
sm.explore(find=0x08048678, avoid=0x08048666)

for s in sm.deadended:
  # The first line of output by the program is "Enter password: ". Get the second line of output: the response.
  response = s.posix.stdout.concretize()[1]

  if response == b'Good Job.':
    print(s.posix.stdin.concretize())
~
```

Angr Management

And behold! The program is defeated by the sheer power of that black magic known as discrete math!



```
(angr) angr@angr:re$ ./solver
WARNING | 2020-08-09 02:48:18,009 | angr.state_plugins.symbolic_memory | The program is accessing memory or registers with an unspecified value. Thi
s could indicate unwanted behavior.
WARNING | 2020-08-09 02:48:18,009 | angr.state_plugins.symbolic_memory | angr will cope with this by generating an unconstrained symbolic variable a
nd continuing. You can resolve this by:
WARNING | 2020-08-09 02:48:18,009 | angr.state_plugins.symbolic_memory | 1) setting a value to the initial state
WARNING | 2020-08-09 02:48:18,009 | angr.state_plugins.symbolic_memory | 2) adding the state option ZERO_FILL_UNCONSTRAINED_{MEMORY,REGISTERS}, to m
ake unknown regions hold null
WARNING | 2020-08-09 02:48:18,009 | angr.state_plugins.symbolic_memory | 3) adding the state option SYMBOL_FILL_UNCONSTRAINED_{MEMORY_REGISTERS}, to
 suppress these messages.
WARNING | 2020-08-09 02:48:18,010 | angr.state_plugins.symbolic_memory | Filling register edi with 4 unconstrained bytes referenced from 0x80486b1 (
__libc_csu_init+0x1 in 00_angr_find (0x80486b1))
WARNING | 2020-08-09 02:48:18,014 | angr.state_plugins.symbolic_memory | Filling register ebx with 4 unconstrained bytes referenced from 0x80486b3 (
__libc_csu_init+0x3 in 00_angr_find (0x80486b3))
Deprecation warning: Use self.model.get_any_node() instead of get_any_node
WARNING | 2020-08-09 02:48:22,401 | angr.state_plugins.symbolic_memory | Filling memory at 0x7ffefffc with 87 unconstrained bytes referenced from 0x
818ac20 (strcmp+0x0 in libc.so.6 (0x8ac20))
WARNING | 2020-08-09 02:48:22,402 | angr.state_plugins.symbolic_memory | Filling memory at 0x7ffeff60 with 4 unconstrained bytes referenced from 0x8
18ac20 (strcmp+0x0 in libc.so.6 (0x8ac20))
[b'JXWVXRKX']
(angr) angr@angr:re$ ./00_angr_find
Enter the password: JXWVXRKX
Good Job.
(angr) angr@angr:re$ 
```

Thank you
for coming to ~~my~~ our
~~TED talk.~~
final presentation!