Random Acts of RE

Robert Graham

About me

Just a hacker (BlackICE, masscan, sidejacking)

I do only reverse engineering as a means to a goal.

I have reverse engineered – I don’t reverse engineer.

- Each effort very different than the last

I don’t do much

1. Get the thing (e.g. firmware update)
2. Extract the thing (e.g. find old version of Mac Stuffit to extract Cisco firmware)
3. Run tool to ‘analyze’ the thing
    a. Massage the thing so the tool knows what to do
    b. Massage some more
4. Document clues
    a. Rename parms/variables
    b. Document data structures
5. Produce disassembled code
    a. Some like working with disassembly, I like working with decompiled

Example: your tools are often confused

ASCII vs. x86

These constants are obviously lower-case letters in the range 0x61-0x7A.

There's only a 1-in-10,000 chance a 4-byte number will match 4 ASCII letters.

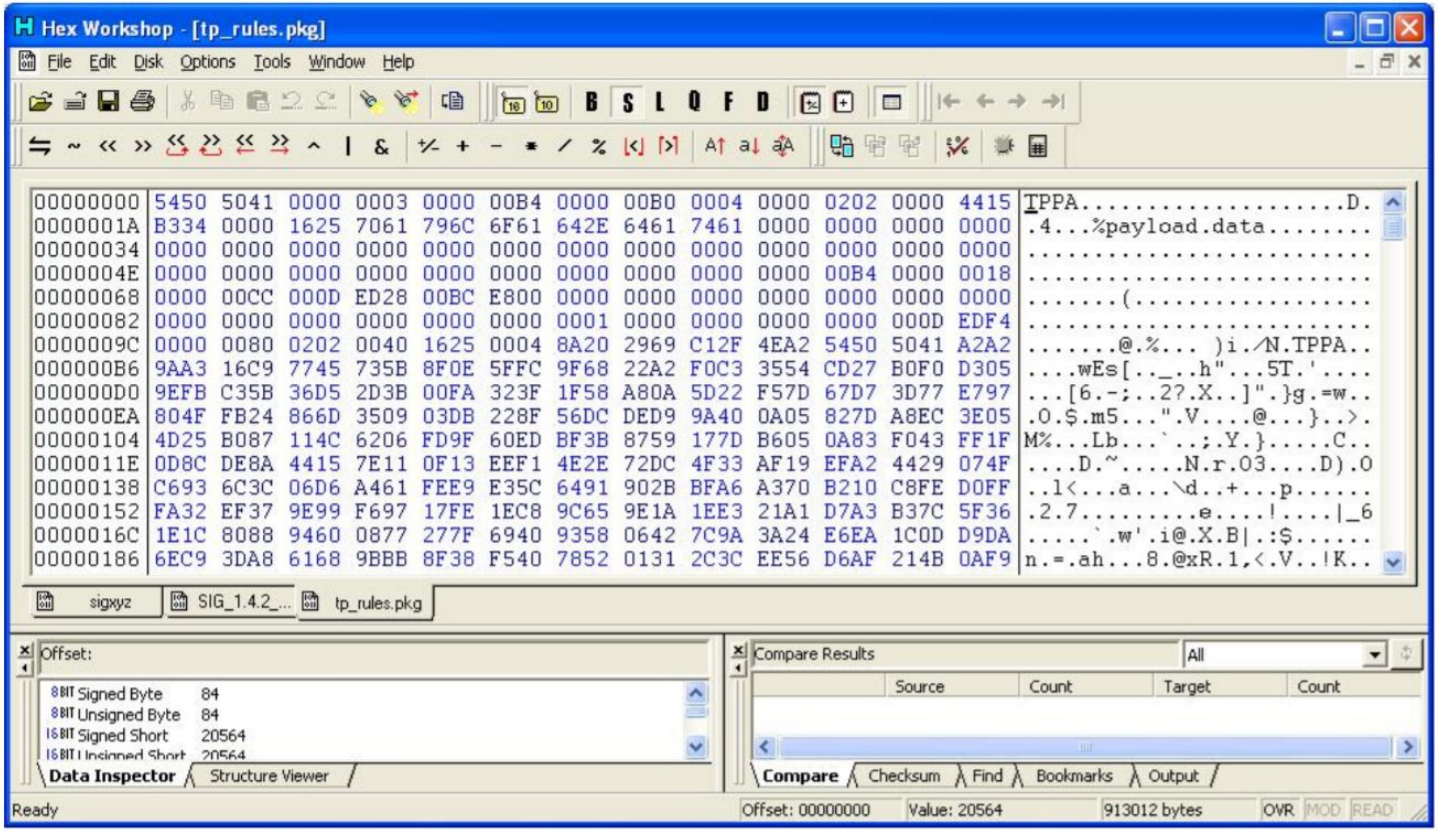
…and can we talk about ESP as a general purpose register??

```
0190eeb8        db      "ek/cloud/SpeechRecognizer;", 0
0190eed3        and     dword [ebx*2+0x6f], ecx
0190eed7        insd    dword [edi], dx
0190eed8        das
0190eed9        imul    esp, dword [esi+0x6c], 0x6b657479
```

Competitive Analysis, #1 - IPS signatures

```
<triggers>
  <trigger version="5500+" confidence="50">
    <payloads>
      <payload>
        <strmatch offset="0" depth="-1">|a0 01 00 00 00 00 00 00[redacted]0|
      </payload>
    </payloads>
  </trigger>
</triggers>

<tests>
  <test version="6024+" confidence="97">
    <protocol type="tcp">
     <ip>
      <saddr><eq><param refid="ip.saddr"/></eq></saddr>
      <daddr><eq><param refid="ip.daddr"/></eq></daddr>
       <tcp>
        <sport><eq><param refid="tcp.sport"/></eq></sport>
        <dport><eq><param refid="tcp.dport"/></eq></dport>
        <established/>
        <flags>A +</flags>
       </tcp>
     </ip>
    </protocol>
    <payloads>
      <payload>
        <strmatch offset="0" depth="-1">|a0 01 00 00 00 00 00 0[redacted]
      </payload>
      <payload>
        <strmatch offset="0" depth="-1">|01 10 08 00 [redacted]|</strmatch>
      </payload>
```

Competitive, #2



Side note

DATA FILES NEED REVERSING TOO!!!

Competitive, #3

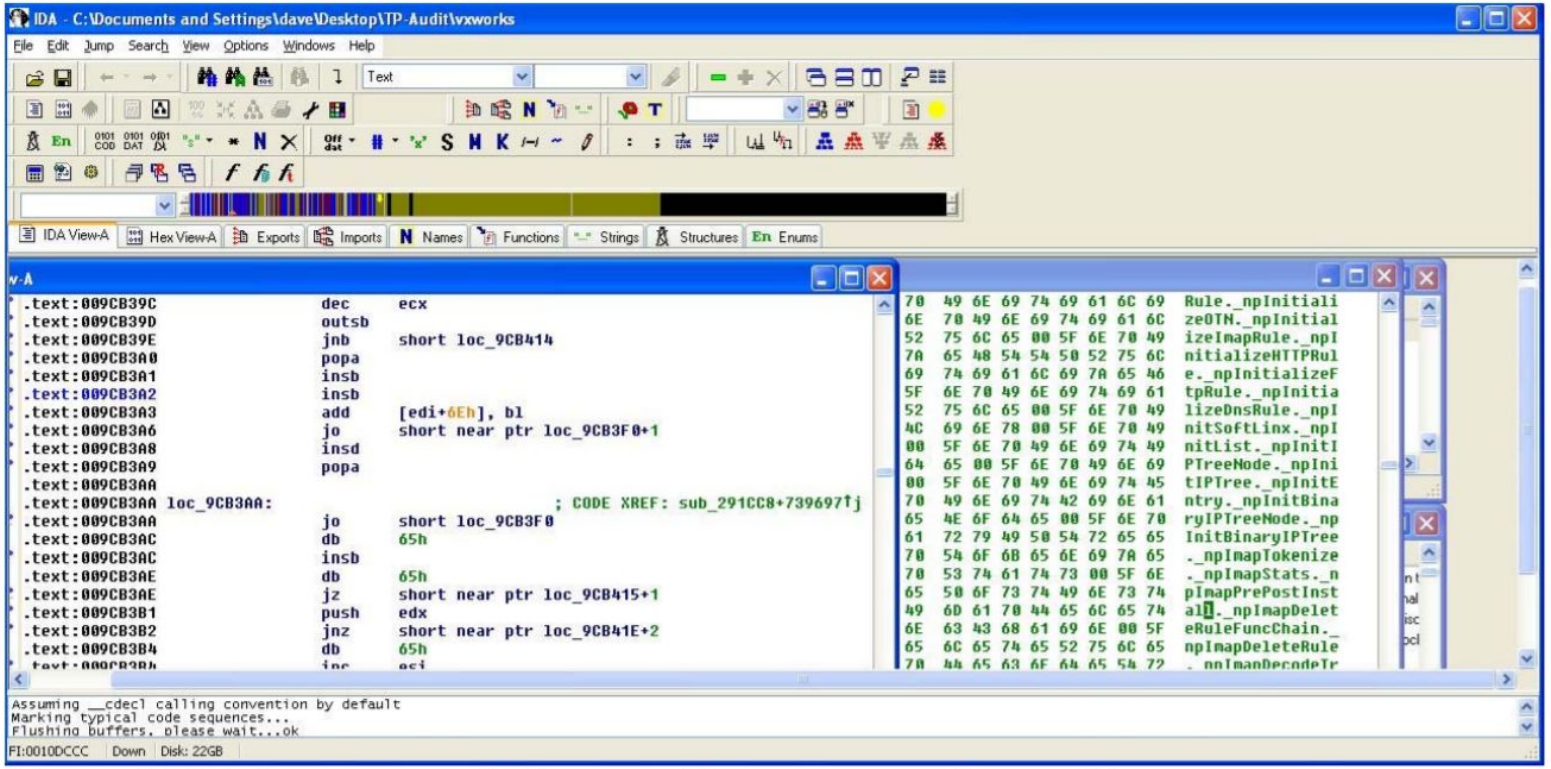
HexWorkshop is a tool for helping you deal with raw unknown binaries. You can

pkg_hdr.hsl - Notepad

File Edit Format View Help

```
#pragma byteorder(big_endian)

struct pkg_hdr
{
        DWORD Magic;
        DWORD Version;
        DWORD Size;
        DWORD unknw1;
        DWORD unknw2;
        DWORD unknw3;
        DWORD unknw4;
        DWORD unknw4_m;
        int name[16];
        DWORD key_offset;
        DWORD key_len;
        DWORD payload_data_start;
        DWORD unknw6;
        DWORD unknw7;
        DWORD unknw8;
        DWORD unknw9;
        DWORD unknw10;
        DWORD unknw11;
        DWORD unknw12;
        DWORD unknw13;
        DWORD unknw14;
        DWORD unknw15;
        DWORD unknw16;
        DWORD unknw17;
        DWORD unknw18;
        DWORD unknw19;
        DWORD unknw20;
        DWORD unknw21;
        DWORD unknw22;
        DWORD Magic_tail;
};
```

Competitive #4: analyzing the binary



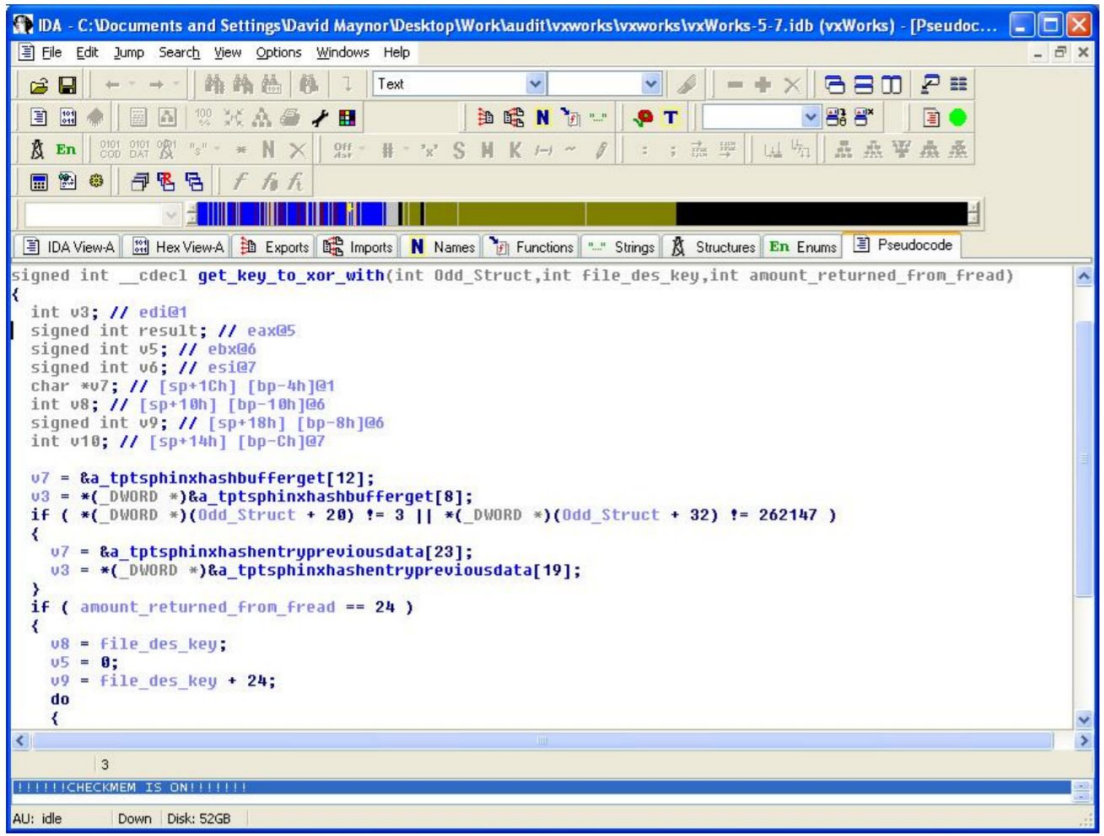
Competitive #4 fixups

vxworks2 - Notepad

File Edit Format View Help

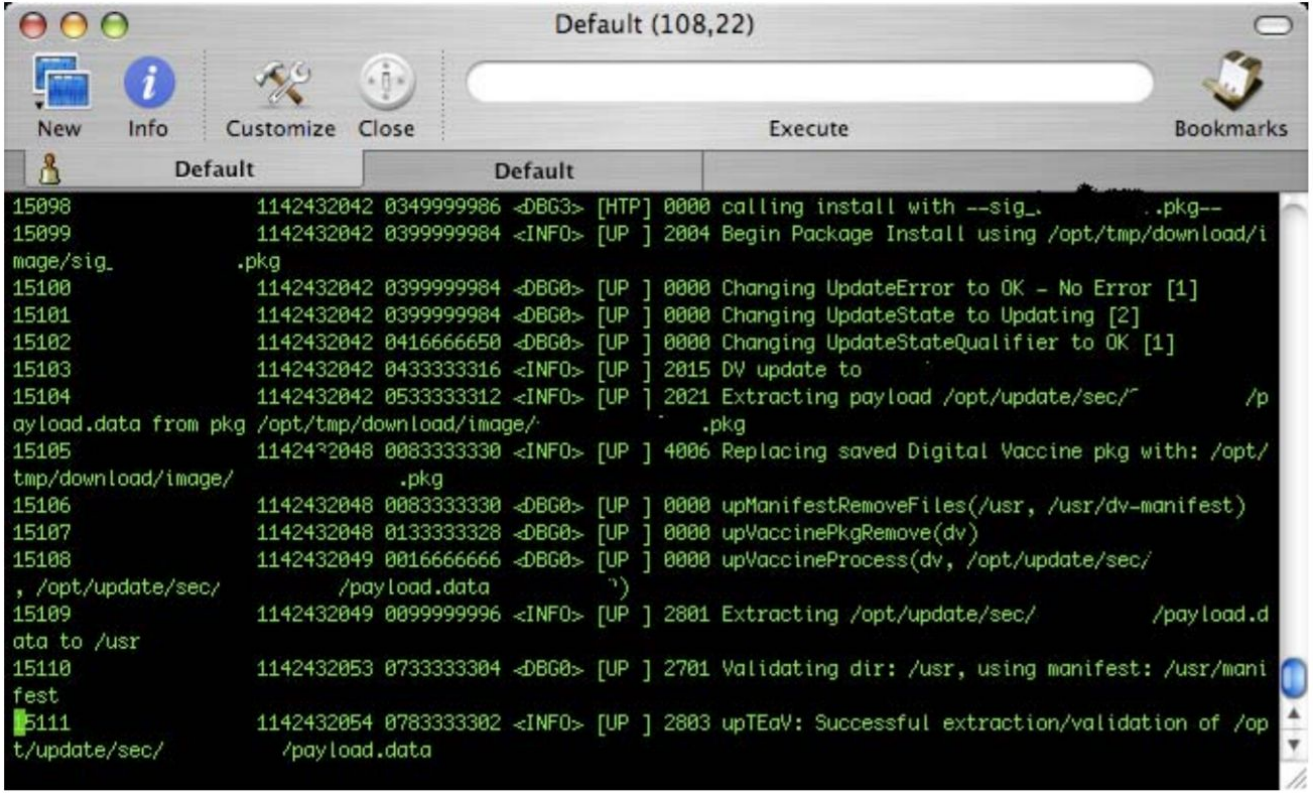
```
        start=0xb394b0;
        end=0xb9cd04;
//      cnt=ScreenEA();
        loop=0;
        Message("CNT=%x\n", cnt);
        cnt=start;
        while(cnt < end){
//      while(loop!=100){
                Message("Currently on: %x\n", cnt);
                make_vxoffset(cnt);
                make_vxoffset(cnt+4);
                name_offset=Dword(cnt);
                func_off=Dword(cnt+4);
                func_off=func_off-0x108000;
                Message("Nameoffset1=%x\n", name_offset);
                name_offset=name_offset-0x108000;
                Message("Name_offset2=%x\n", name_offset);
                name=GetString(name_offset, -1, 0);
                Message("Makename at=%x with=%s\n",func_off, name);
                MakeName(func_off, name);
                cnt=cnt+0x10;
        }
        Message("Done\n");

}


static main ()
{
        DelHotkey("z");
        AddHotkey("z","cleanup_strings");
        DelHotkey(",");
        AddHotkey(",","sym_table");

        return;
}
```

Competitive #5 zero in on the encryptions



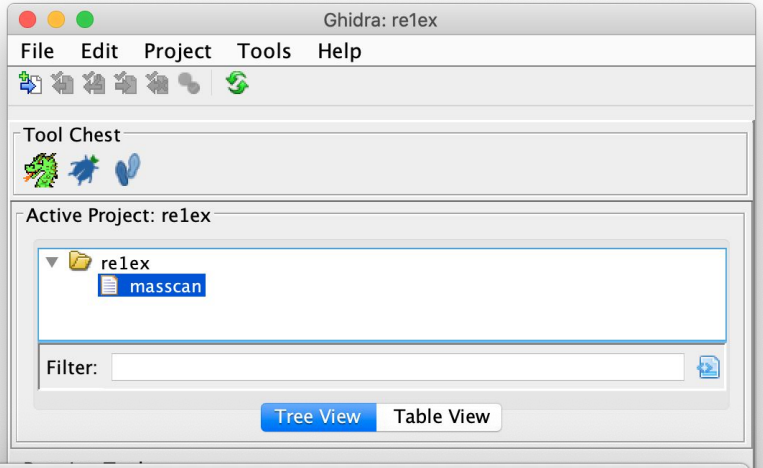
Competitive #6 - navigating the binary

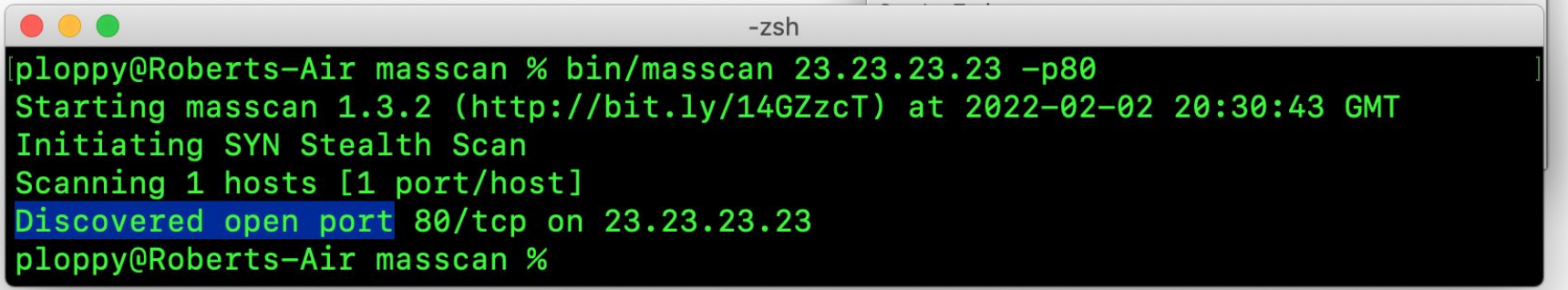
- Log files
- Error messages
- File open/read
- Crypto routines



Example#1 going backwards from strings

In this example, I'm going to work with a stripped version of 'masscan'. Masscan is fun because it has no dependencies

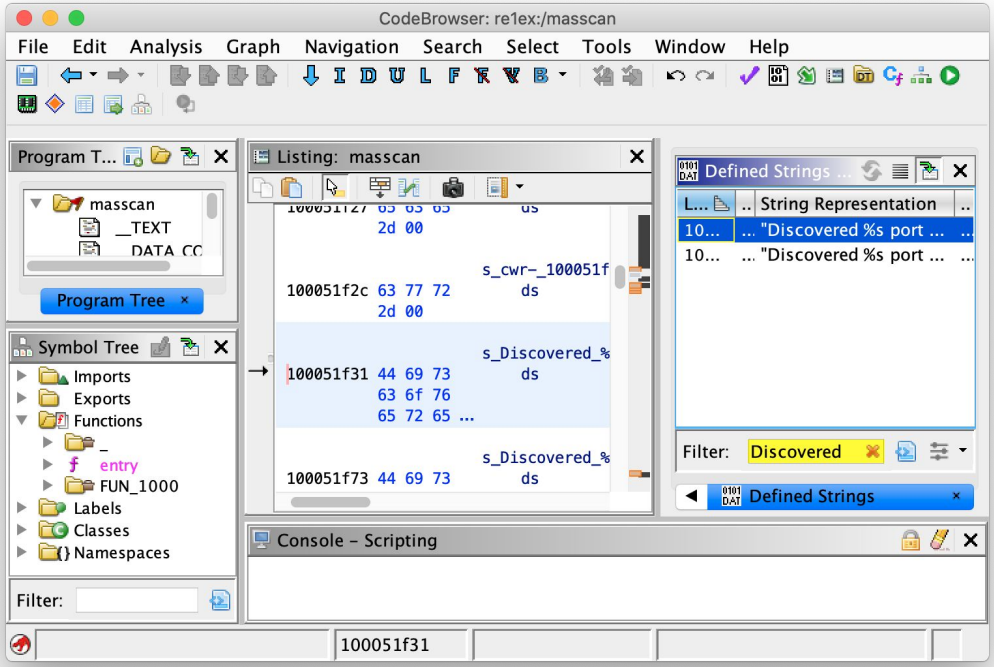




Example #2

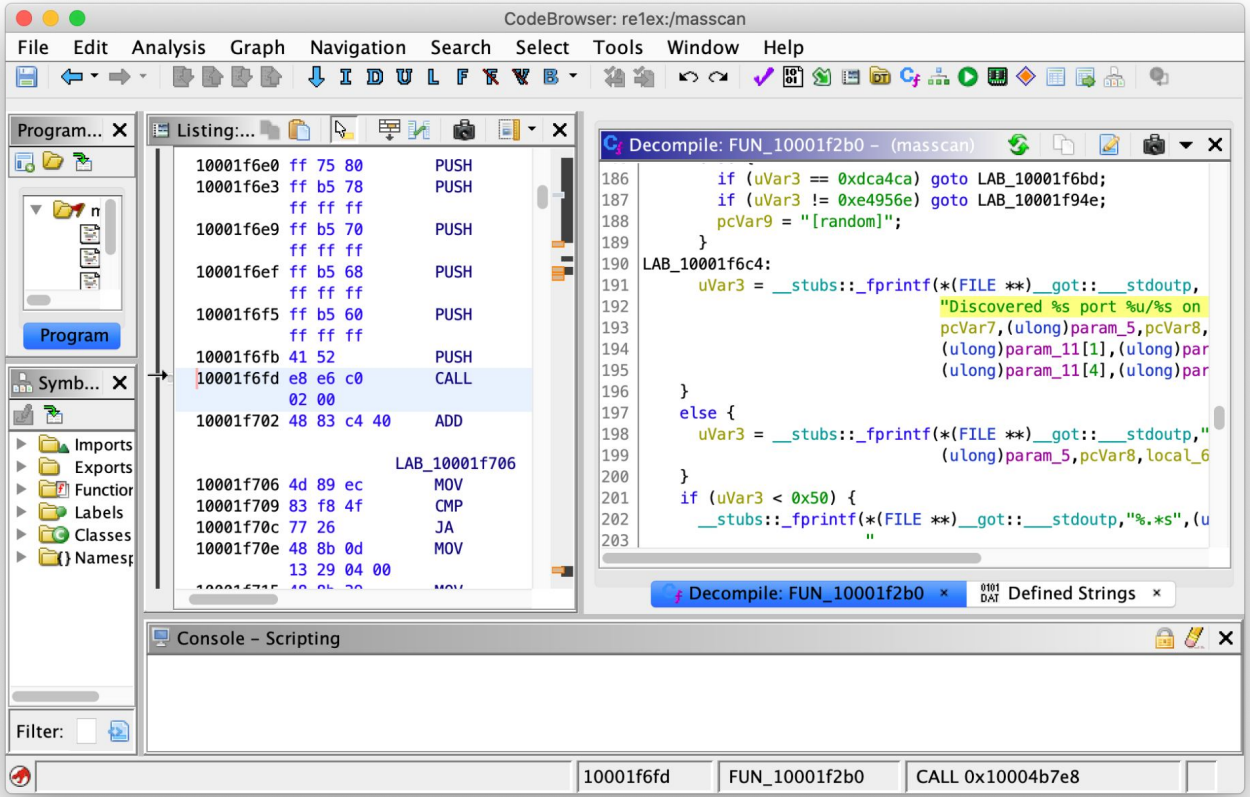
Stripped, so function table useless

But what do I have? Strings. So start with strings.



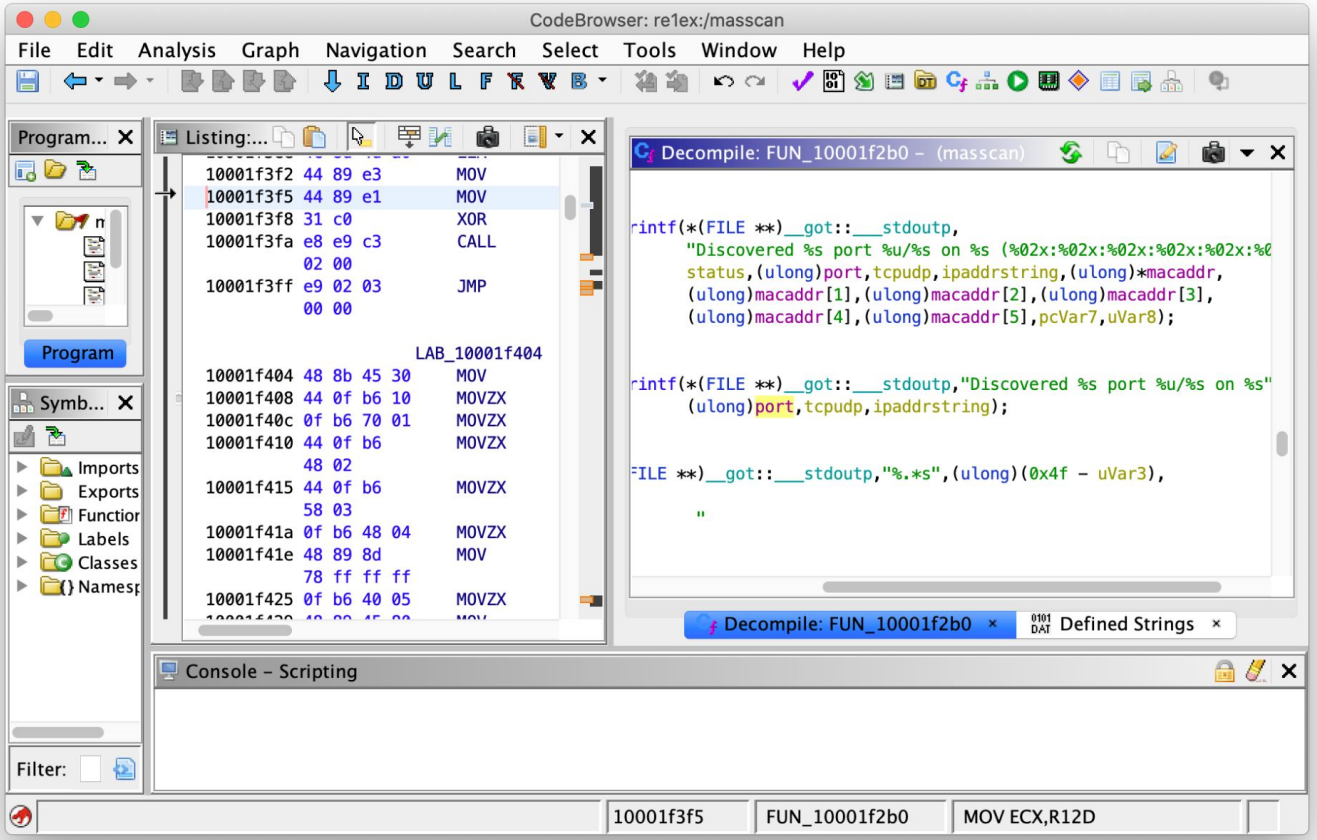
Example #3

Right click, go to “References”, switch to Decompiler view, scroll down to function



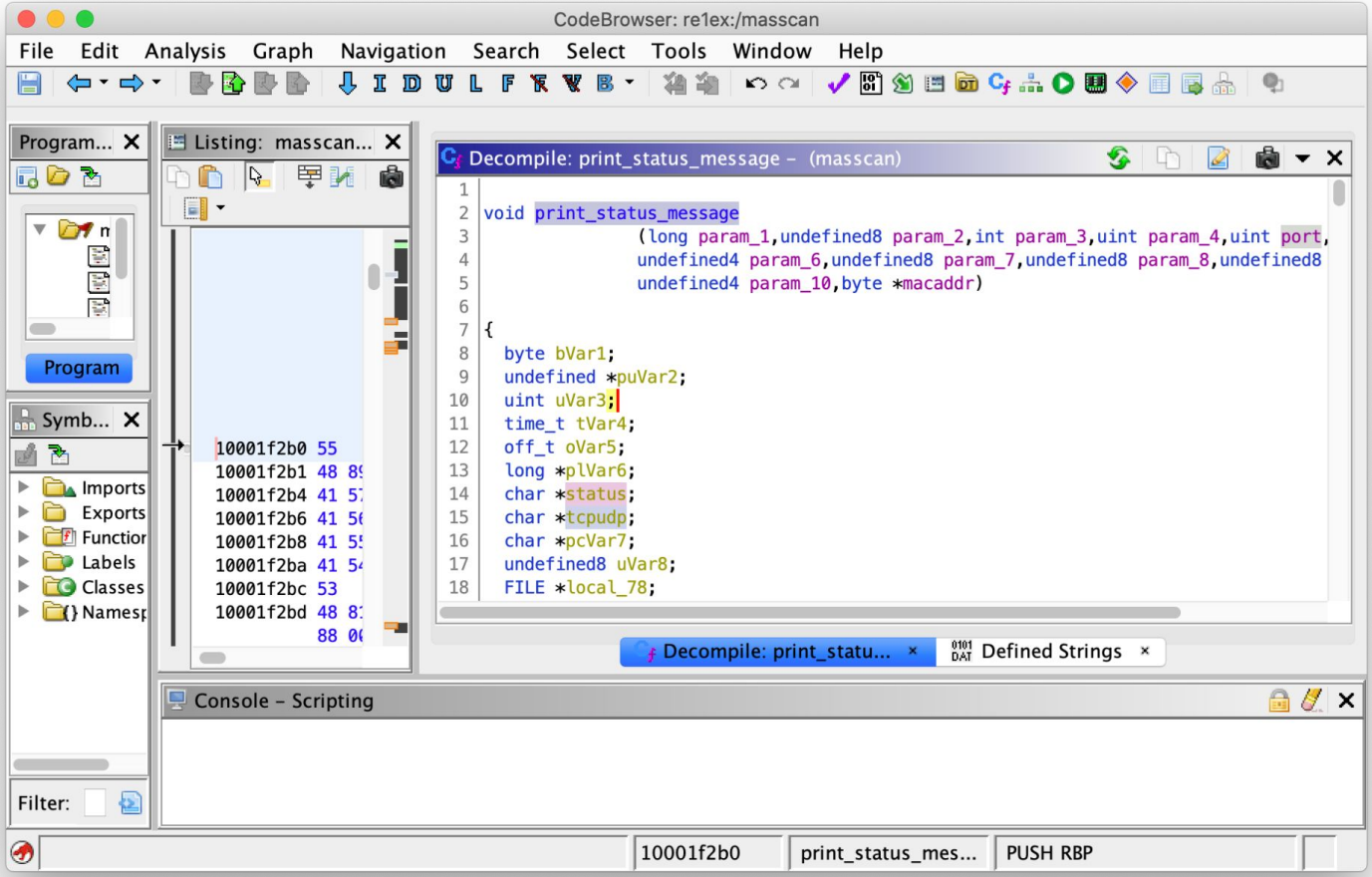
Example #4

Rename the variables in this function



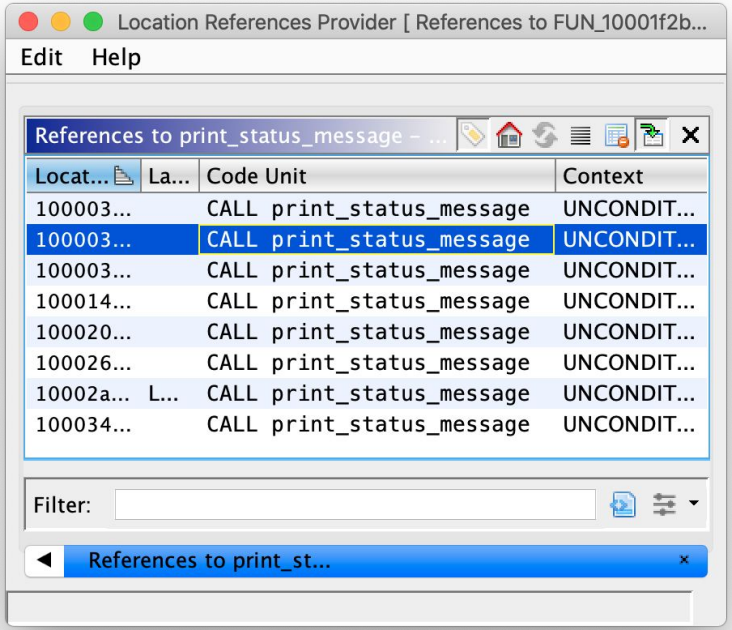
Example

Rename and highlight



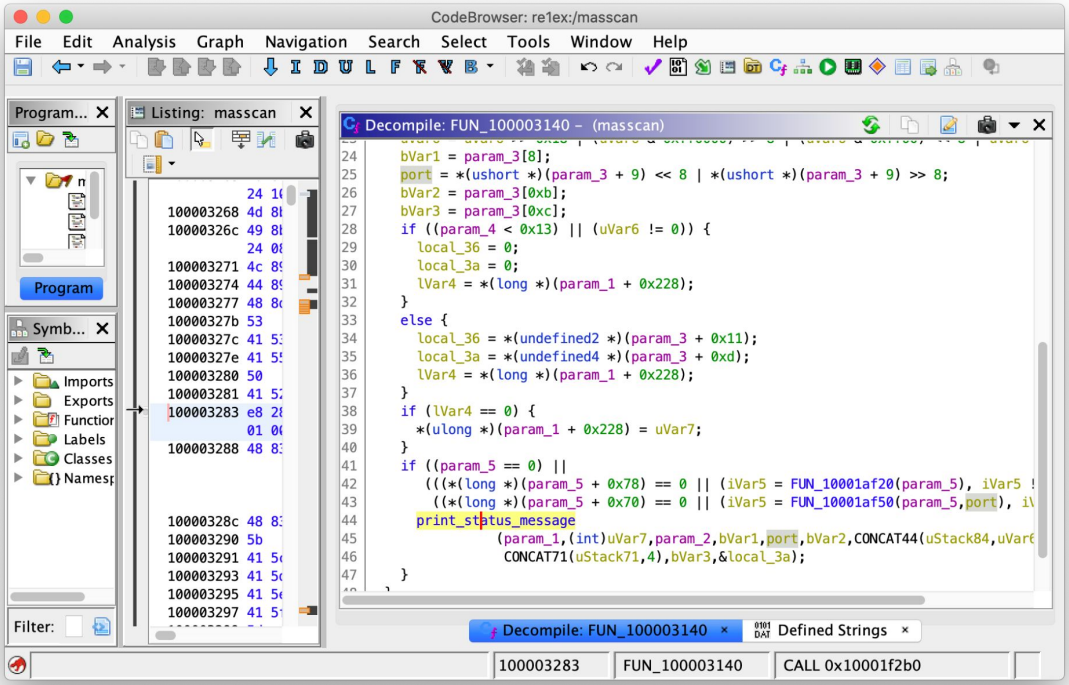
Example

Now recursively find those callers of our “print_status_messag e()” function. Now we have many to choose from.



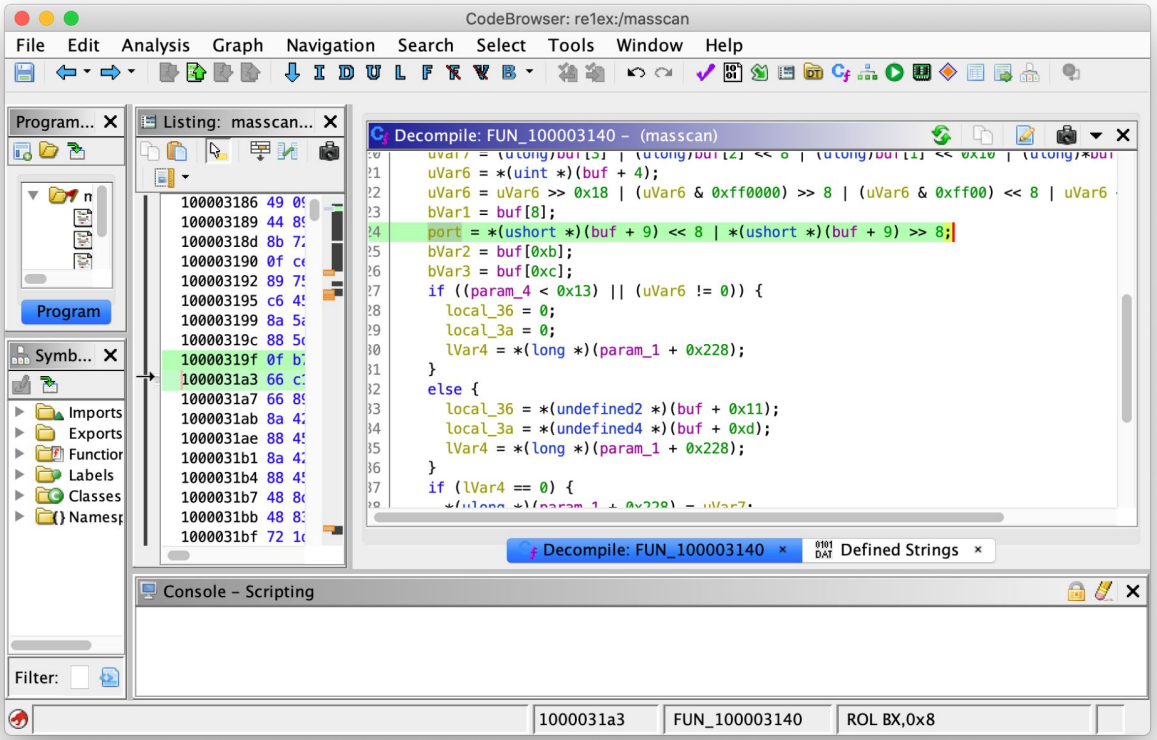
Example

Go up the call tree

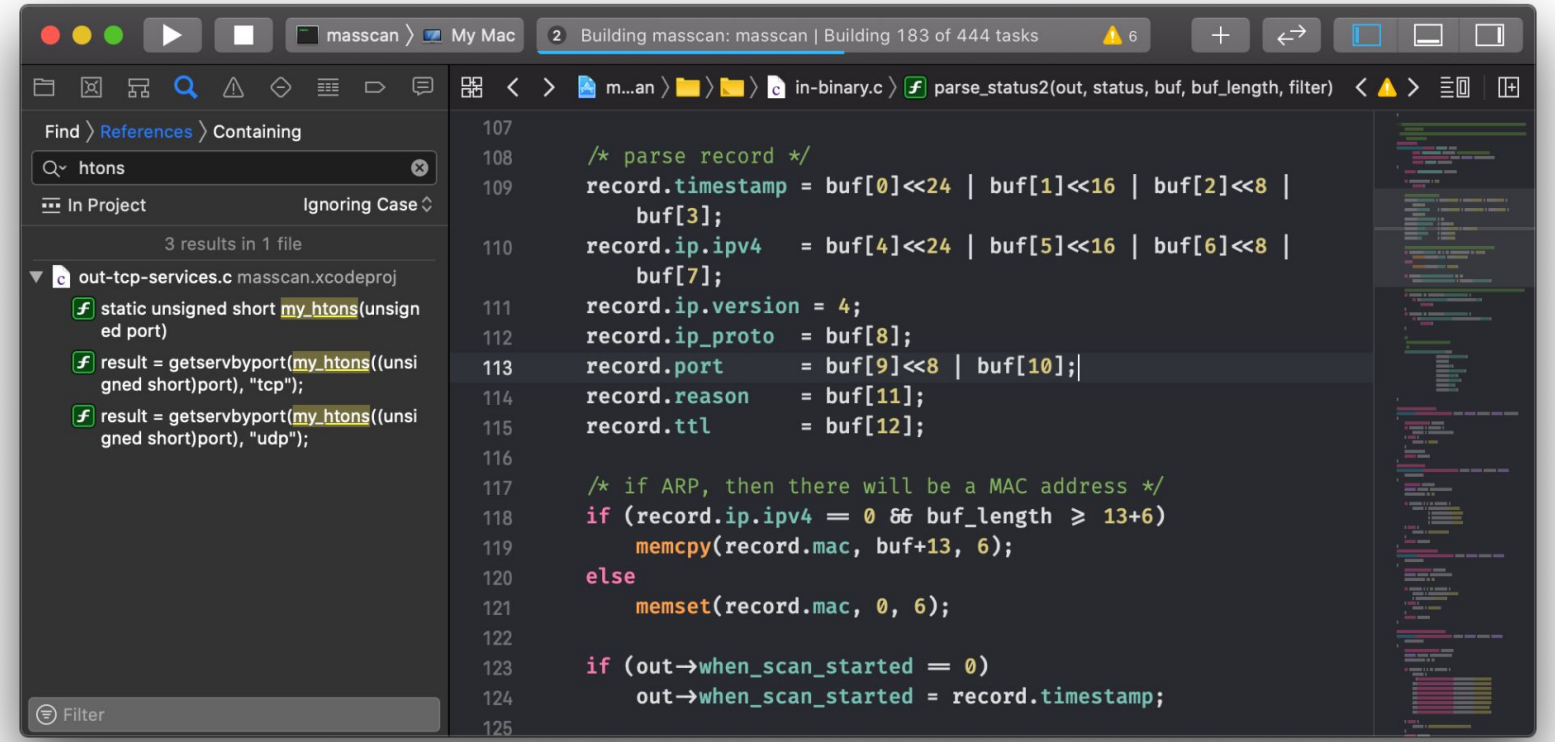


port = htons(*(short*)buf);

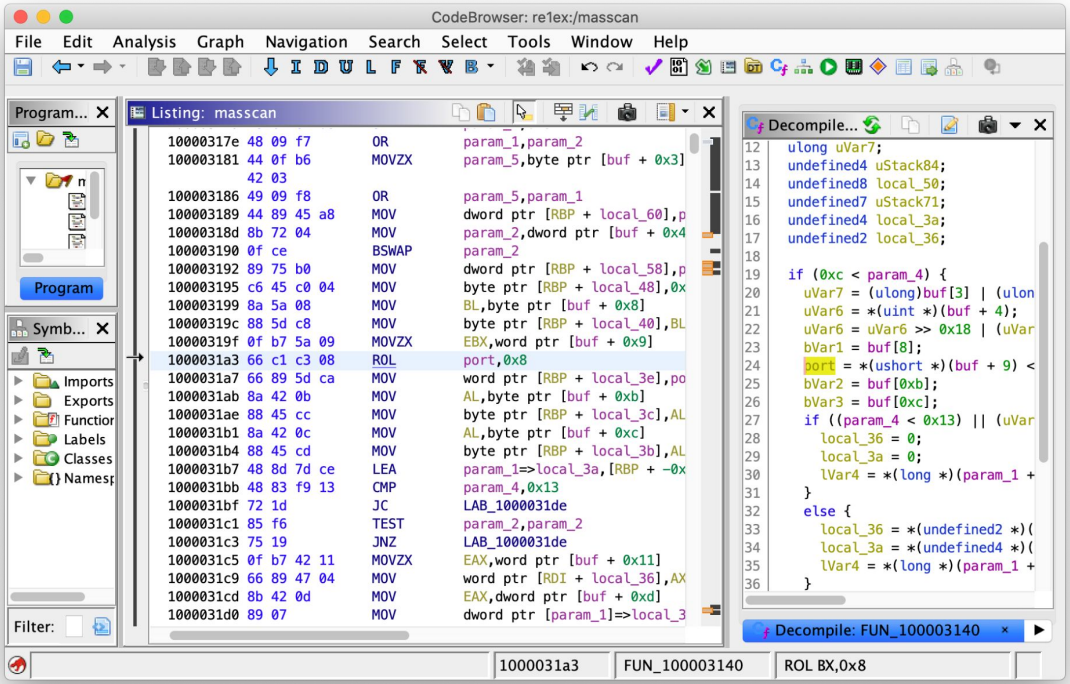
Reverse engineering requires some knowledge of engineering – recognizing patterns from engineering, in this case, htons()



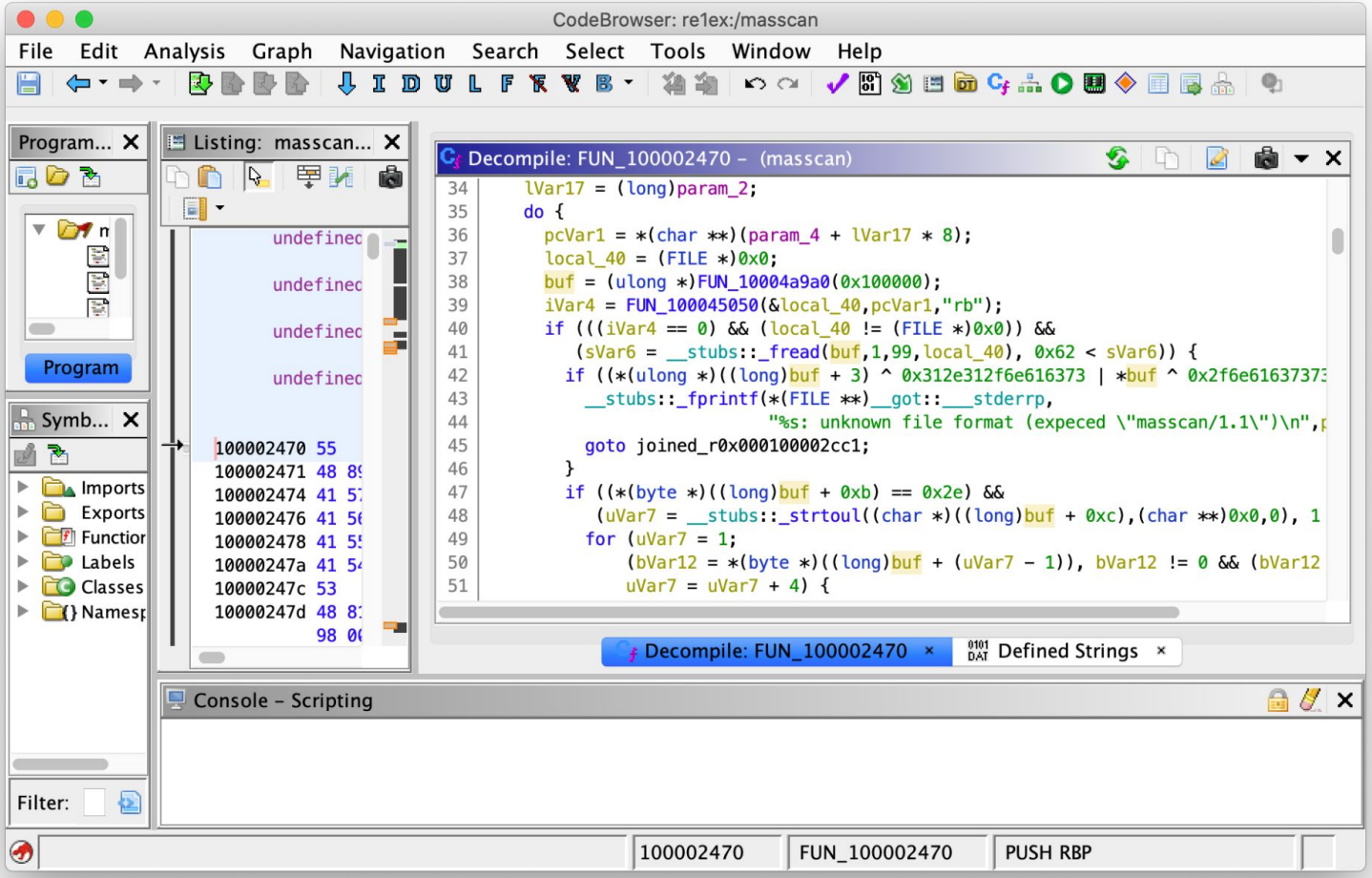
DISTRACTION: don’t use htons()



DISTRACTION (continued)



Example


CodeBrowser: re1ex:/masscan
File Edit Analysis Graph Navigation Search Select Tools Window Help
Program...
Program
Symb...
Imports
Exports
Functior
Labels
Classes
Namesp
Filter:
Listing: masscan...
undefinec
undefinec
undefinec
undefinec
100002470 55
100002471 48 8
100002474 41 5
100002476 41 5
100002478 41 5
10000247a 41 5
10000247c 53
10000247d 48 8
98 0
Decompile: FUN_100002470 - (masscan)
34 lVar17 = (long)param_2;
35 do {
36 pcVar1 = *(char **)(param_4 + lVar17 * 8);
37 local_40 = (FILE *)0x0;
38 buf = (ulong *)FUN_10004a9a0(0x100000);
39 iVar4 = FUN_100045050(&local_40,pcVar1,"rb");
40 if (((iVar4 == 0) && (local_40 != (FILE *)0x0)) &&
41 (sVar6 = __stubs::_fread(buf,1,99,local_40), 0x62 < sVar6)) {
42 if ((*(ulong *)((long)buf + 3) ^ 0x312e312f6e616373 | *buf ^ 0x2f6e6163737
43 __stubs::_fprintf(*(FILE **)__got::___stderrp,
44 "%s: unknown file format (expeced \"masscan/1.1\")\n",
45 goto joined_r0x000100002cc1;
46 }
47 if ((*(byte *)((long)buf + 0xb) == 0x2e) &&
48 (uVar7 = __stubs::_strtoul((char *)((long)buf + 0xc),(char **)0x0,0), 1
49 for (uVar7 = 1;
50 (bVar12 = *(byte *)((long)buf + (uVar7 - 1)), bVar12 != 0 && (bVar12
51 uVar7 = uVar7 + 4) {
Decompile: FUN_100002470
Defined Strings
Console - Scripting
100002470
FUN_100002470
PUSH RBP

Q: Where to start?

A: Anywhere.

You don’t have what you want, so start with what you have.

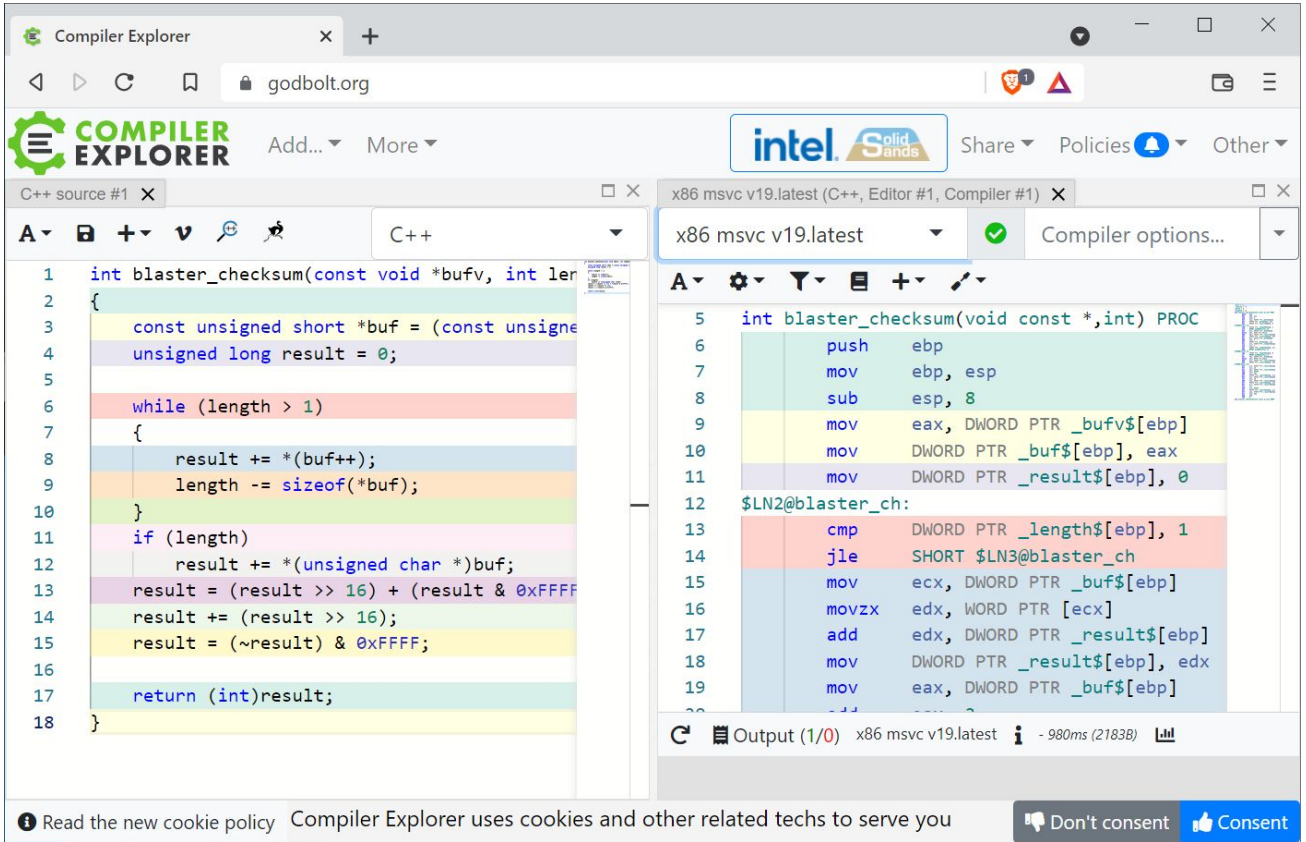
Each of my projects is different. I like to start with ‘strings’ because no matter how much things have been stripped, there always strings to start with.

Blaster.c – reverse engineering a worm

Reverse engineer the code all the way back to the comments – find bugs and anomalies that can help us find open-source

```
1464        /* Major bug here: the worm writer incorrectly calculates the
1465         * IP checksum over the entire packet. This is incorrect --
1466         * the IP checksum is just for the IP header itself, not for
1467         * the TCP header or data. However, Windows fixes the checksum
1468         * anyway, so the bug doesn't appear in the actual packets
1469         * themselves.
1470         */
1471        ip.checksum = blaster_checksum(buf, sizeof(ip) + sizeof(tcp));
```

Godbolt forward engineering



Obsessive reverse engineering

I spend a lot of time obsessively documenting and reversing exactly what's going on. The difference between -1 and 0xFFFFFFFF is important. All my friends are lazy about it

```
ADD eax,-1

ADD eax,0xFFFFFFFF
```

BlackICE #1 - reversing Microsoft

- One of the first personal firewalls
- No APIs existed to intercept packets
- So we had to reverse engineer the kernel
  - Find any function that processes incoming packets
  - Figure out how it works, how to mess with it safely
  - Put our code in some slack area of memory
  - Insert JMP instruction, to our code, then we jump back
  - On operating system boot, hunt for the function, do our patch
  - On shutdown, remove our patch

BlackICE #2 patent

How to patch live

- Multi-CPU responding to interrupts
- Impossible to make a multibyte change without a crash
- Find one byte JMP change, to JMP back a few bytes
- Catch CPU in a infinite loop
- Put multibyte change after it to jump to our code
- Then break open the loop with one byte change
-

(10) **Patent No.: US 8,006,243 B2**
(45) **Date of Patent: Aug. 23, 2011**

(10) **Patent No.: US 7,181,486 B1**
(45) **Date of Patent: Feb. 20, 2007**

BlackICE #3 - WinXP SP2

They radically changed the network stack we couldn’t find the function

We are on list of “must work software”

So they reverse engineered what we were doing

They stuck the pattern we looked for in the code.

- Their packet-receive function jumped to, and after, jumped right back.
- We found it and patched it
- So we continued to work

Legal #1 reverse engineering

So yea, they caught us reverse engineering their stack.

Their lawyers made noises about it.

Their techies told ‘em to shut up.

FYI: to prove we reverse engineered them, they had to reverse engineer us

So, counter-suit?

5. **LIMITATION ON REVERSE ENGINEERING, DECOMPILATION, AND DISASSEMBLY**. You may not reverse engineer, decompile, or disassemble the Product, except and only to the extent that it is expressly permitted by applicable law notwithstanding this limitation.
6. **TERMINATION**. Without prejudice to any other rights, Microsoft may cancel this EULA if you do not abide by the terms and conditions of this EULA, in which case you must destroy all copies of the Product and all of its component parts.

Legal #2 - Reverse engineer is VERY LEGAL, except…

Fundamentally, reverse engineering is legal and encouraged.

But software licenses always forbid it, so in practice, software engineering is effectively illegal.

There are various ways to get around it, such as clean room specifications and such.

But in practice, most of what we do, people don’t care, as in the BlackICE example above.

Don’t believe a lawyer “you can’t do it”, but find a good one that explains the risks if you do it.

Legal #3 - DMCA and exceptions

DMCA adds a whole new twist on it

- Can’t reverse for circumventing digital rights protections

But so many exceptions

- Every 3 years Library of Congress issues exceptions.
- One exception for cybersecurity research is regularly granted.
- New exceptions for right-to-repair (e.g. John Deere) are coming

Legal #4 - user vs. agreer

Example: Airtight, WiFi intrusion prevention system

So we through WiFi packets at it in order to test how it responded

So they made legal threats

But we weren’t the customer who agreed to the EULA. We were an outsider.

DLL/PRELOAD injection

Interact live while it’s running.

Write code to walk through data structures, log out a socket/file-handle

Deterministically, parse the stack

Instead of deterministically, grep through looking for the things you want

- I.e. send packet “robertgraham” to process, have your code look for that pattern in memory, then again search elsewhere in memory for that 4 byte pointer

ROT3 (Caeser cipher) in Mike Lindell's Absolute Proof

```
public: String^ GetData(String^ val)
{
    String^ rnx = "!a-bn1poe4-kjb=87rger7rge3$#%#$4675!@#$34(^%!@#$225(&%*";
    String^ key = rnx->Remove(17, 5);
    String^ decrypt = "";
    int ValIndex = Convert::ToInt32(rnx->ToCharArray()[25].ToString());
    for (int iChar = 0; iChar < val->Length; iChar++)
    {
        decrypt += (Char)(val[iChar] - ValIndex);
    }
    return decrypt;
}
```

The best reverse engineering tool

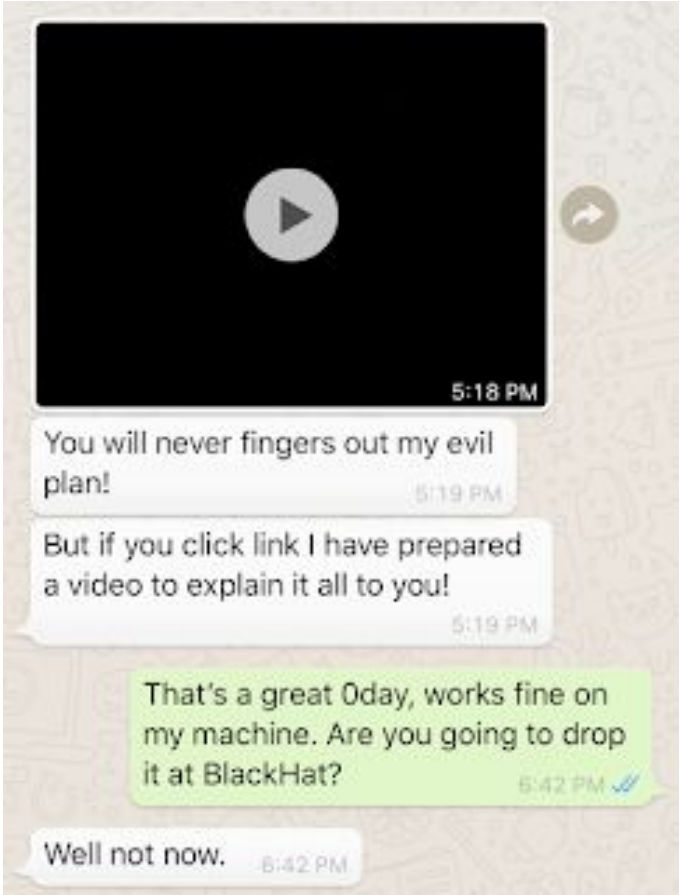
DEBUGGER

WhatsApp

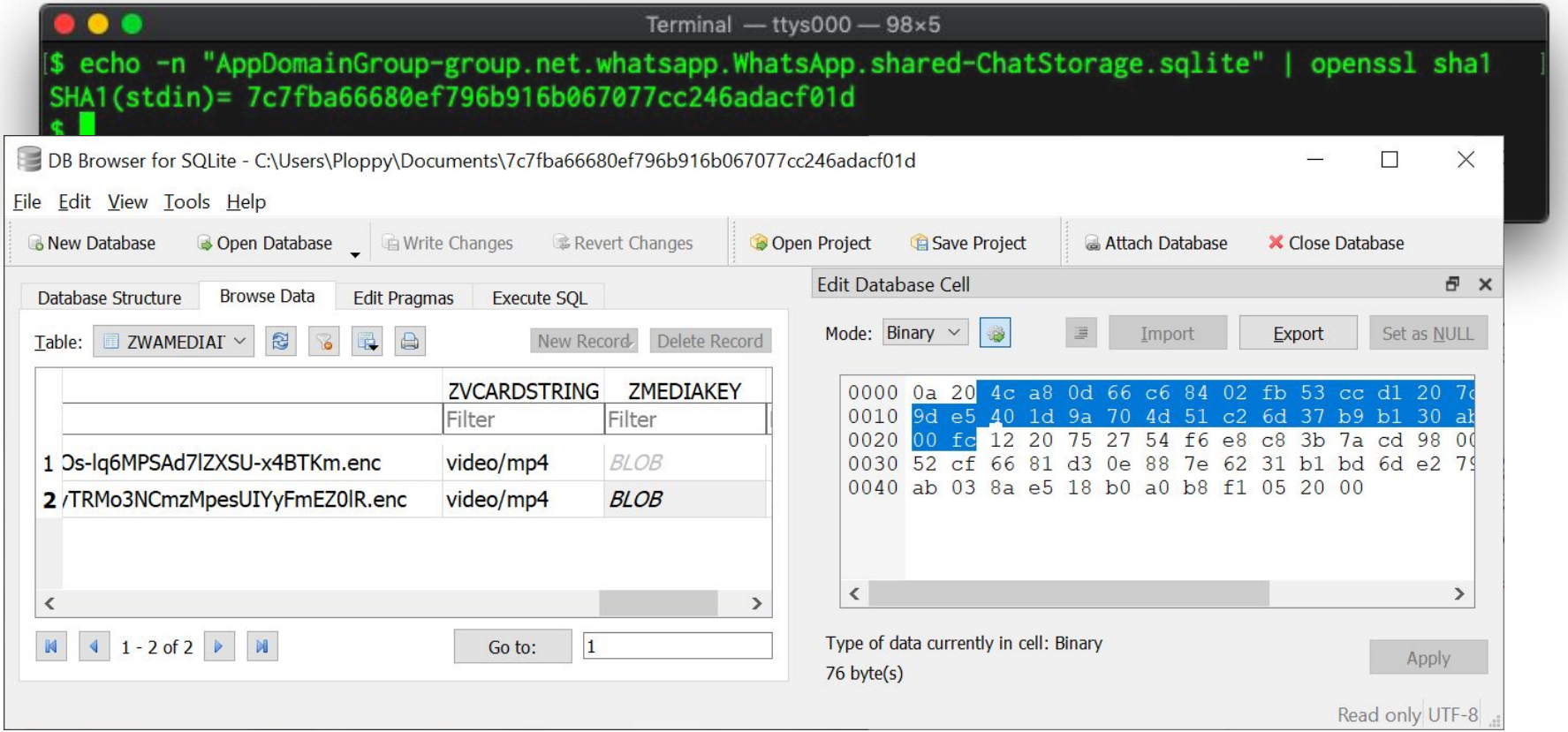
Claim: Jeff Bezos security team claims the Saudi Prince hacked his phone by sending a video, encrypted by WhatsApp, containing an 0day

Analysis: how WhatsApp encrypts videos

https://blog.erratasec.com/2020/01/how-to-decrypt-whatsapp-end-to-end.html



WhatsApp app uses SQLite backend



Phone app to website hacking

Find with strings: https://example.com/foo/bar

Find in code: https://example.com/foo/bar?id=107

Their code:

```
"SELECT * FROM Users WHERE UserId = " + $id
```

My query: https://example.com/foo/bar?id=107;+DROP+TABLE+customers;

Their code:

```
"SELECT * FROM Users WHERE UserId = " + “107; DROP TABLE customers;”

"SELECT * FROM Users WHERE UserId = 107; DROP TABLE customers;”
```

Always debug malware in a virtual machine

A lot of malware does weird things on startup

Among these is unpacking

Among these is obfuscation

You'll need to run the code partway through to unpack, decrypt, and unobfuscate things, to get a memory image to the point where you can then start reading the code.

If you don't do this in a virtual machine, you'll eventually make a mistake and infect yourself and the network you are attached to.

Superfish #1

https://blog.erratasec.com/2015/02/extracting-superfish-certificate.html

Use ‘strings’ to grab all strings from executable, which also gets embedded certificates

```
4844  pc2g&h
4845  IhvcD
4846  Unknown error
4847  equence
4848  operation
4849  SocketAsync
4850  -----BEGIN ENCRYPTED PRIVATE KEY-----
4851  MIICxjBABgkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIDHHhyAEZQoICAggA
4852  MBQGCCqGSIb3DQMHBAiHEg+MCYQ30ASCAoDEvGvFRHvtWOb5Rc0f3lbVKqeUvWSz
4853  xQn+rZELHnwb6baolmbFcsi6XkacVzL/EF7Ll4de/CSQ6pZZCCvfDzov0mPOuGve
4854  SAe7hbAcol7+JWVfzbnVTblPf0i7mwSvK61cKq7YfcKJ2os/uJGpeX9zraywWyFx
4855  f+EdTr348dOez8uHkURyY1cvSHsIdITALkChOonAYT68SVighTeB6xOCwfmsHx+X
4856  3Qbhom2YCIxfJiaAoz2/LndCpDaEfOrVrxXFOKXrIbmeDEyjDQj16AVni9uuaj7l
4857  NiO3zrrqxsfdVINPaAYRKQnS102jXqkH01z72c/MpMMC6dwZswF5V3R7RSXngyBn
```

Superfish #2 PEMcrack

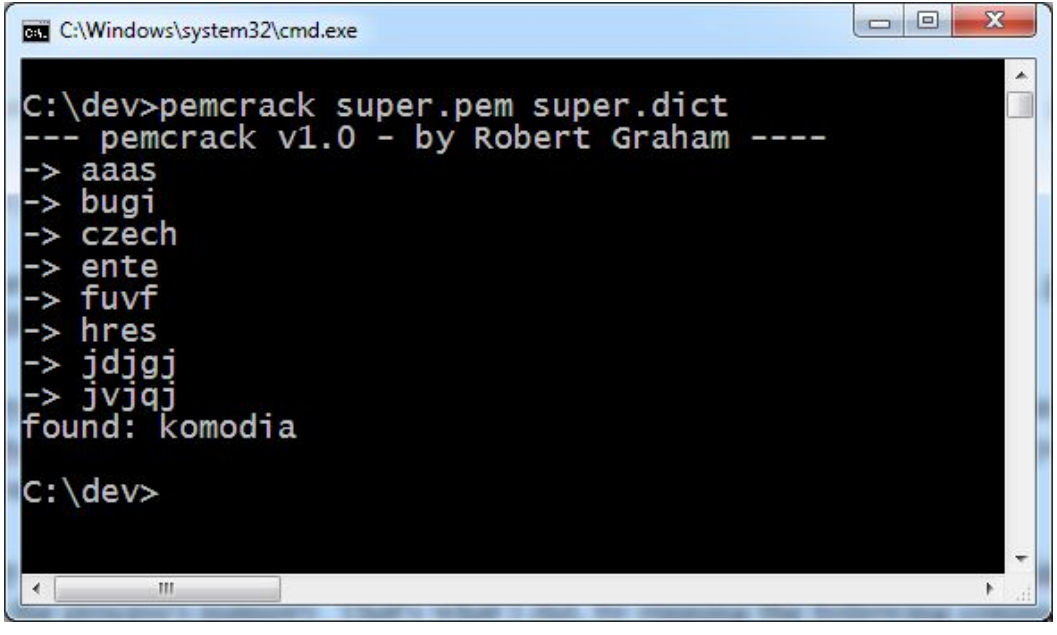
https://github.com/robertdavidgraham/pemcrack

Quick and dirty tool

```
50
51      /* attempt to read the file using the current password */
52      p = PEM_read_PrivateKey(fp,&pkey, NULL, password);
53      if (p) {
54          /* FOUND!!! print and exit the program */
55          printf("found: %s\n", password);
56          exit(0);
57      }
```

Superfish #2

```
strings super.dmp > super.txt

grep "^[a-z]*$" super.txt | sort | uniq > super.dict
```



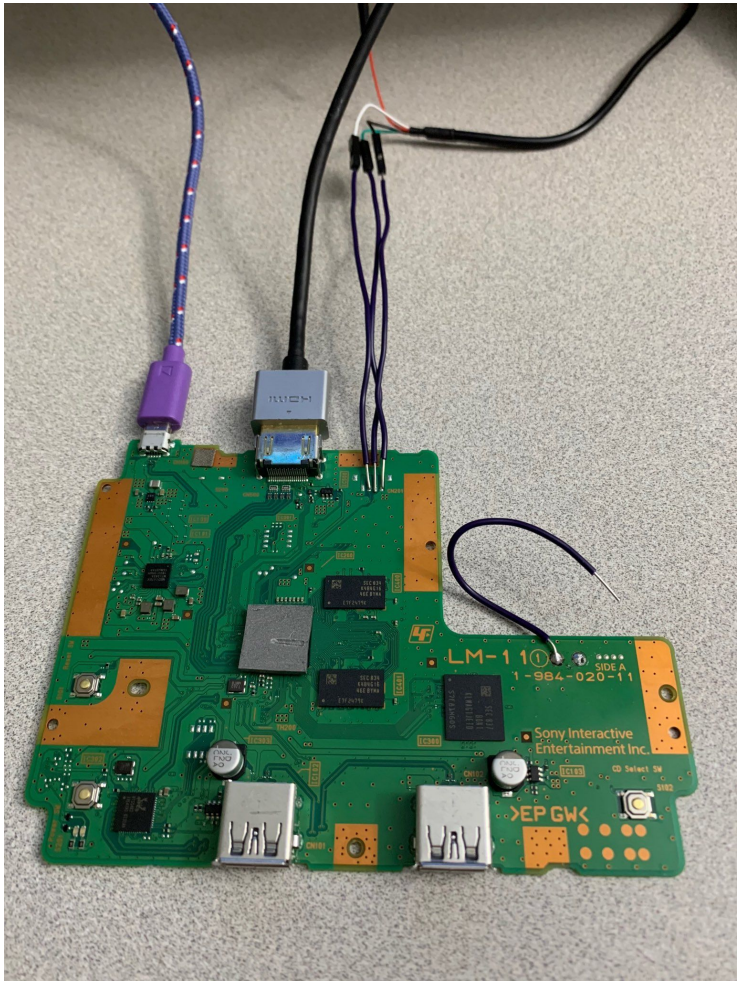
BTW: Don’t be afraid to solder wires

There’s a lot of fun to be had with devices.

This shows soldering on a serial port to a Playstation Classic.

Or, you’ll be soldering onto JTAG.

These are overwhelmingly standard ports.



Conclusion

Every reverse engineering task started with wondering “where do I start?”

Just jump in anywhere.

The key is simply self confidence, know that the impossible will start to become possible as you build up clues.

All hacking is the foolish belief that you’ll succeed at the impossible.