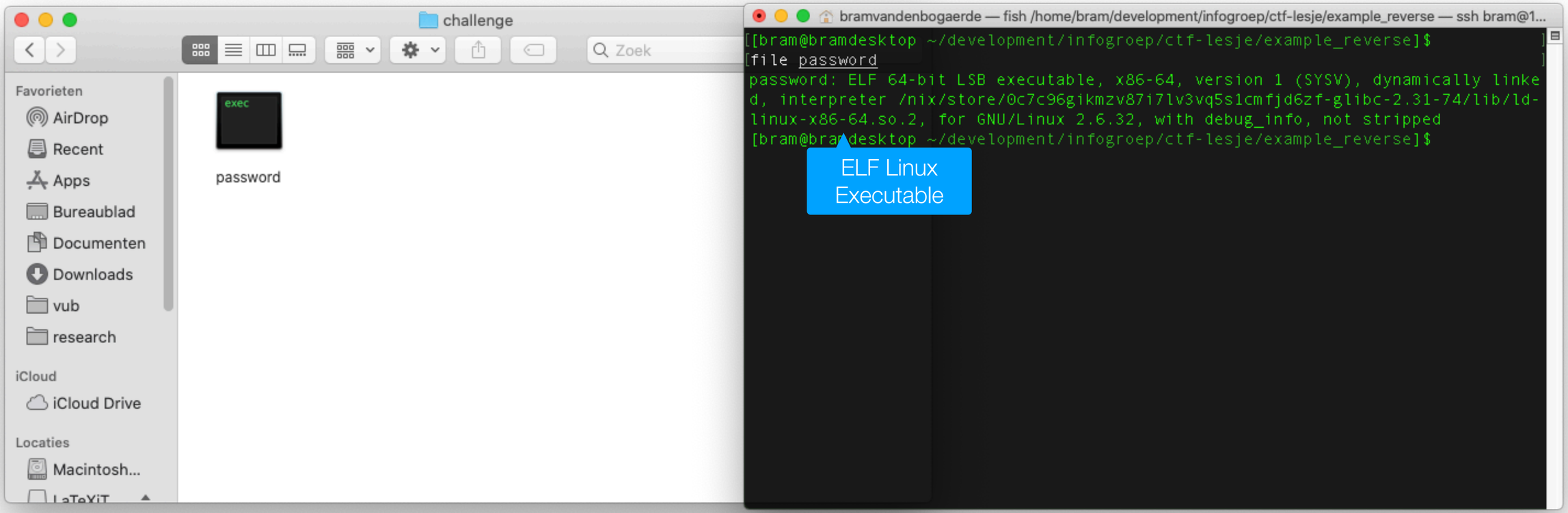
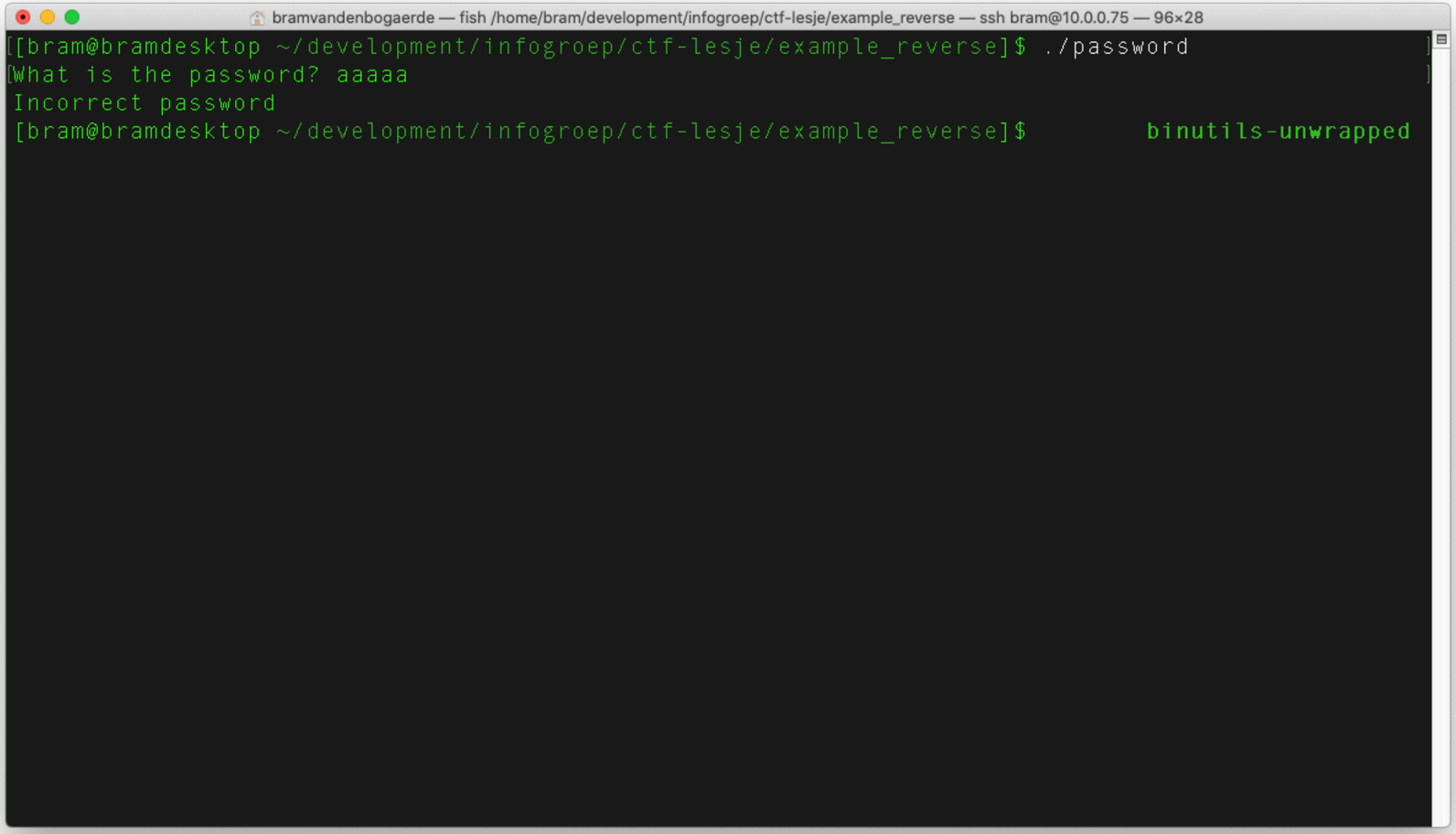
Reverse Engineering

The Challenge



We are given a single “password” binary, it appears to be a Linux binary

The Challenge



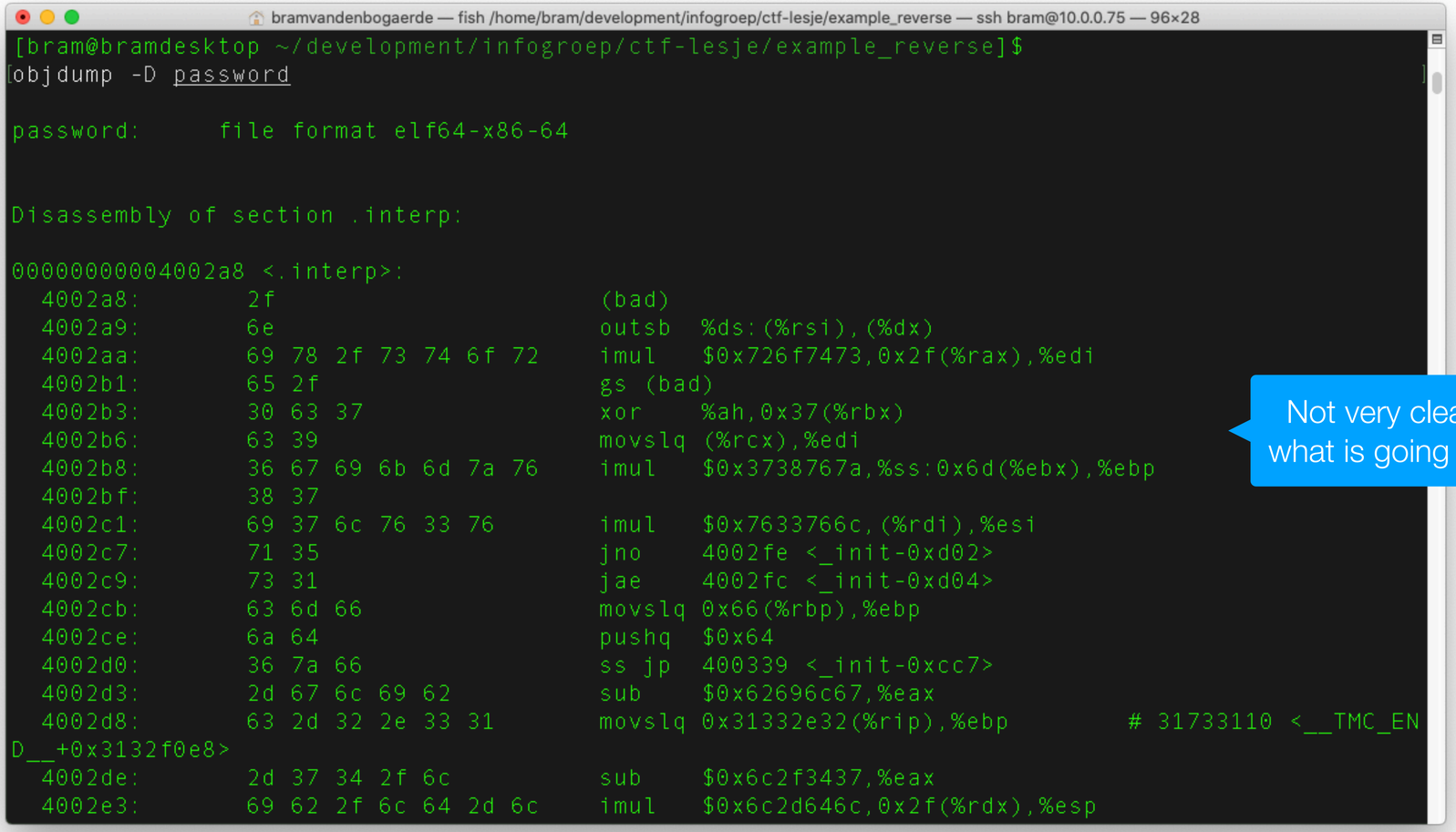
Reverse Engineering

Software reverse engineering (SRE) is the practice of analyzing a **software** system, either in whole or in part, to extract design and implementation information.

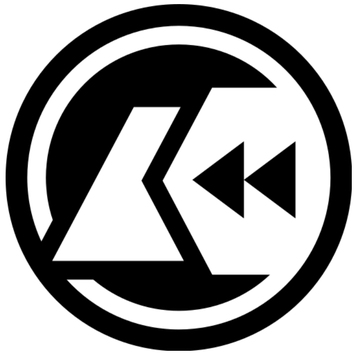
Source: https://link.springer.com/chapter/10.1007/978-3-642-04117-4_31

Simple Reversing using `objdump`

`objdump -d password`



Reverse Engineering Tools



Cutter

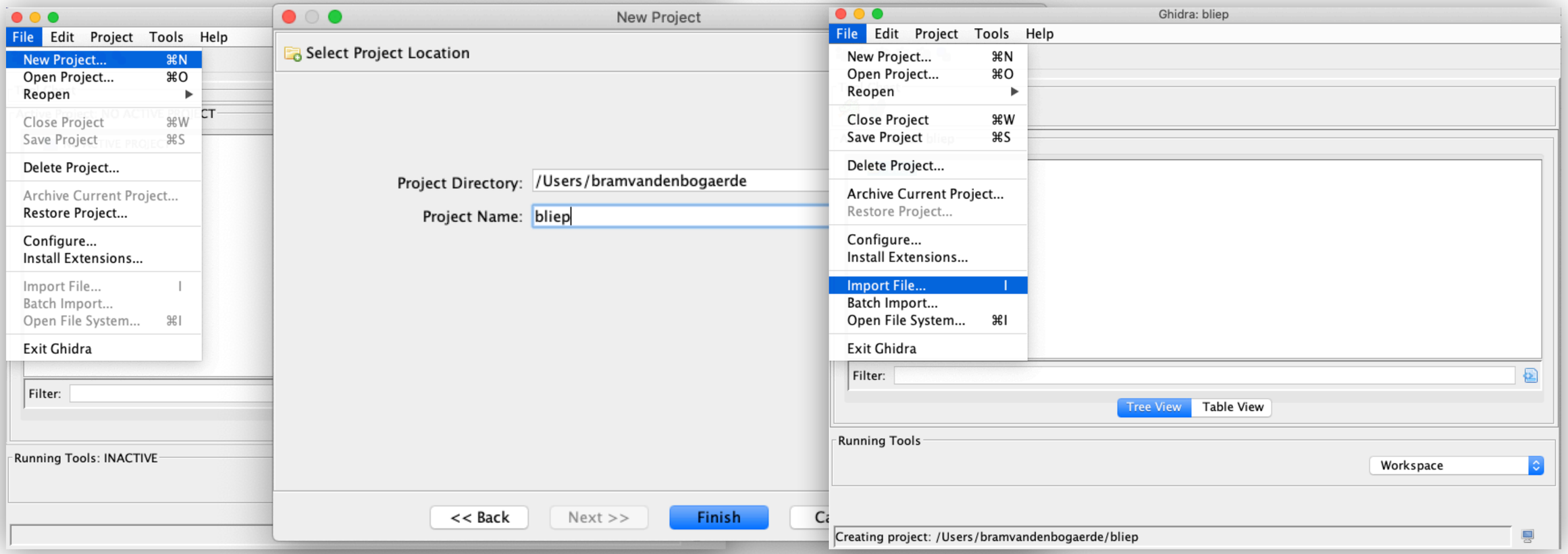
IDA Pro

And more…

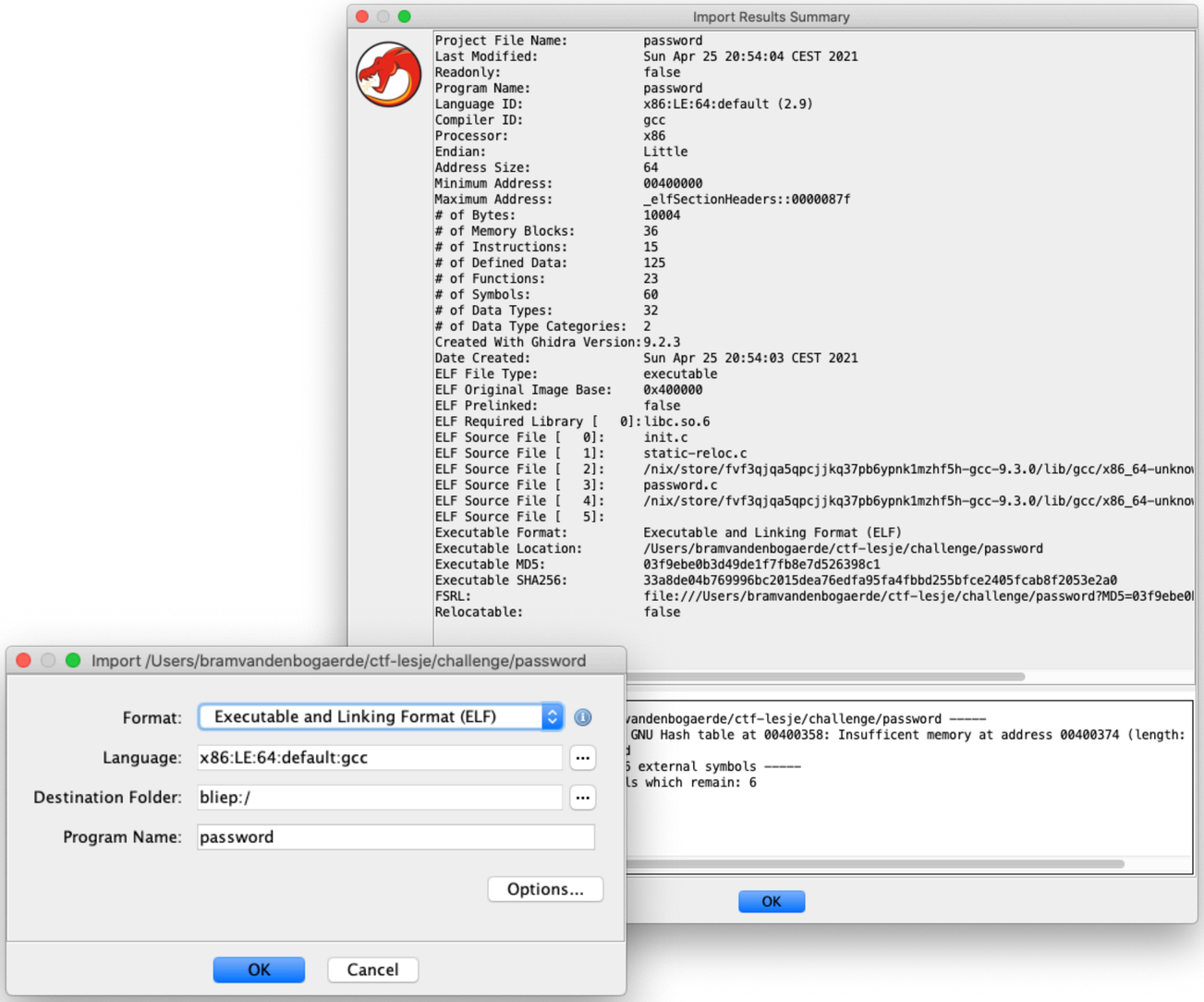
Ghidra Download

http://ghidra-sre.org

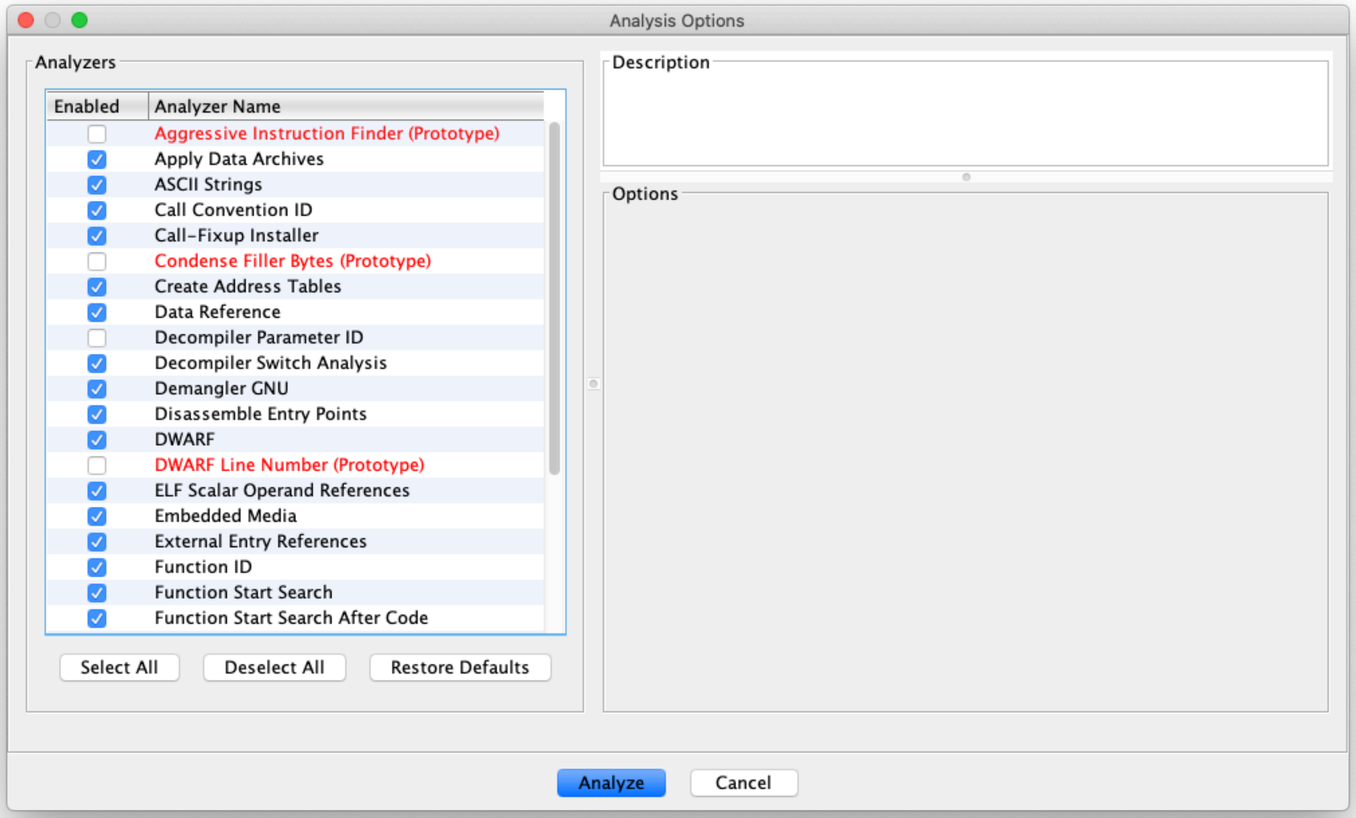
Opening Executable (1)



Opening an executable (2)

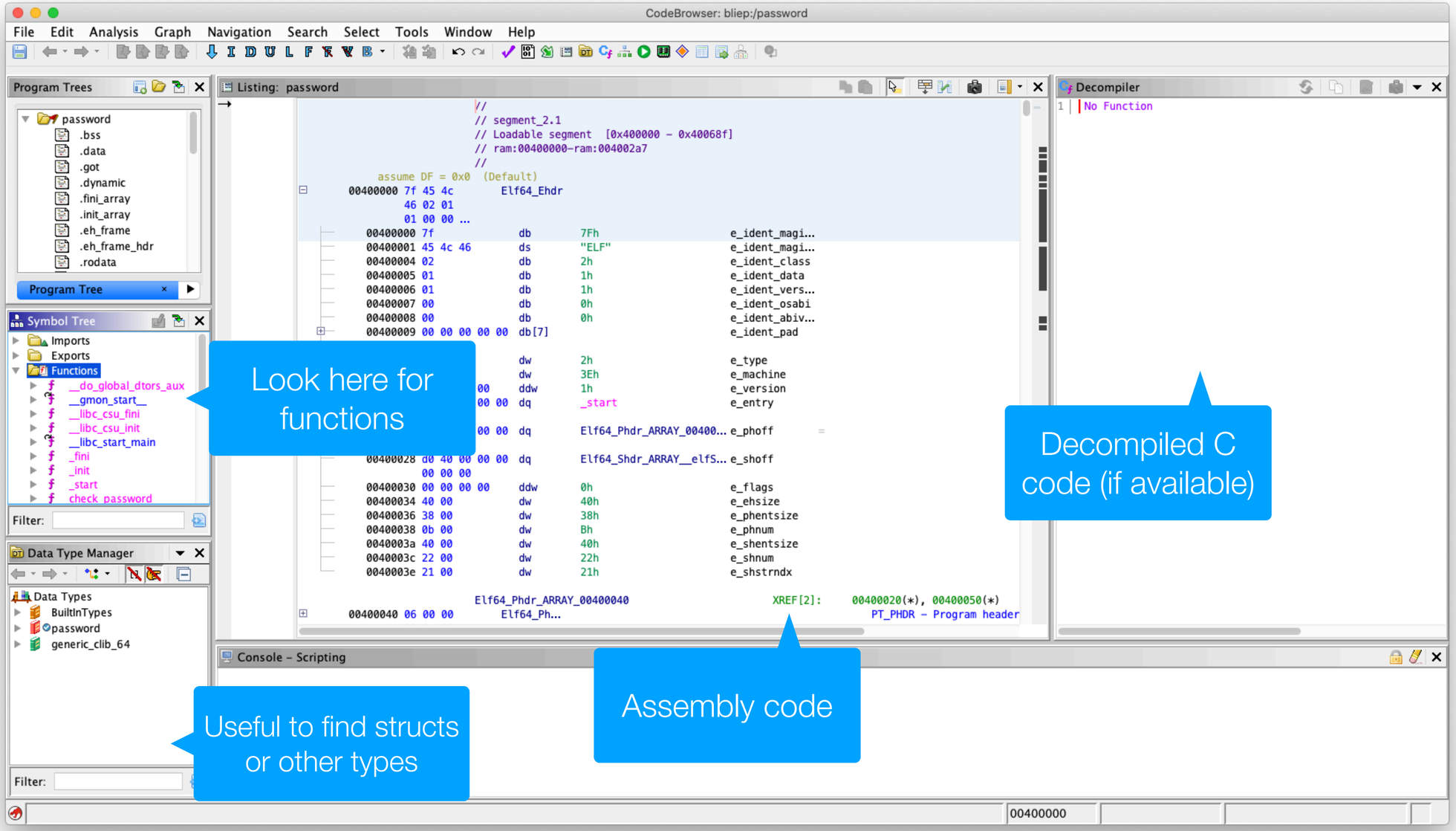


Analyzing an executable

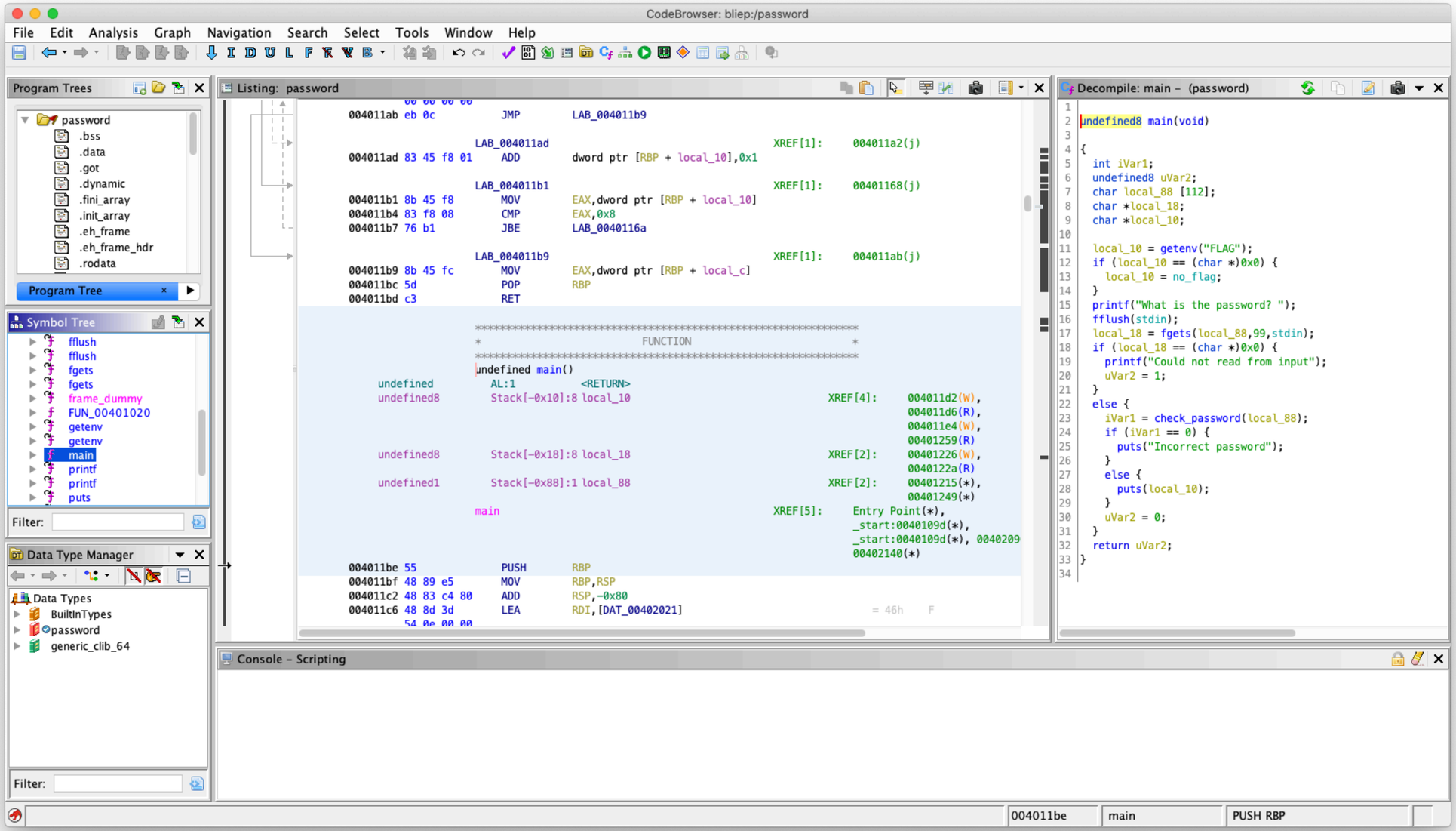


You can customise this but the defaults are mostly fine

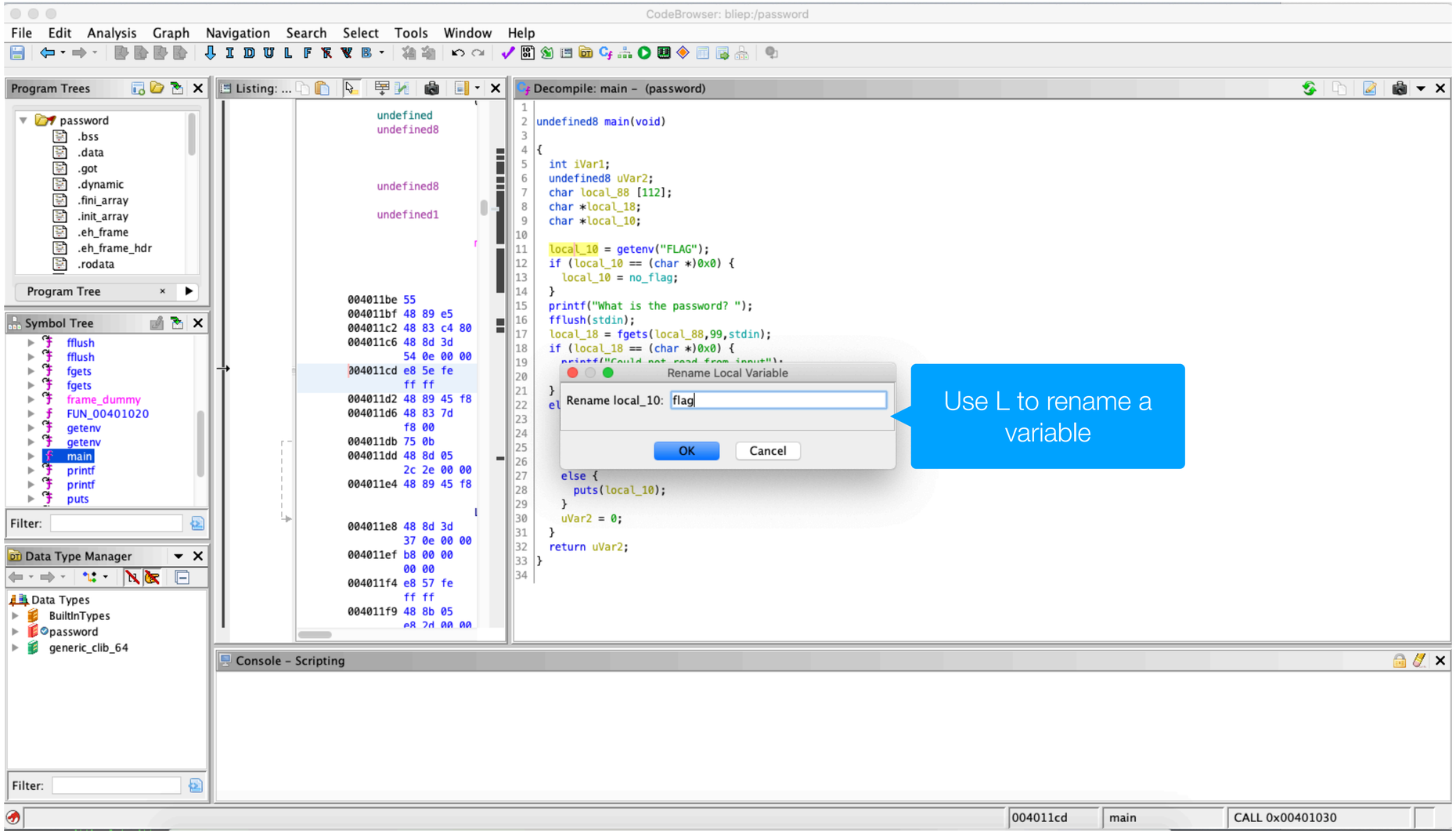
A tour through Ghidra



Finding the main function



Making Decompilation Readable



Cleaned-up version

Decompile: main - (password)

```
1
2  undefined8 main(void)
3
4  {
5    int result;
6    undefined8 retval;
7    char input [112];
8    char *inputCode;
9    char *flag;
10
11   flag = getenv("FLAG");
12   if (flag == (char *)0x0) {
13     flag = no_flag;
14   }
15   printf("What is the password? ");
16   fflush(stdin);
17   inputCode = fgets(input,99,stdin);
18   if (inputCode == (char *)0x0) {
19     printf("Could not read from input");
20     retval = 1;
21   }
22   else {
23     result = check_password(input);
24     if (result == 0) {
25       puts("Incorrect password");
26     }
27     else {
28       puts(flag);
29     }
30     retval = 0;
31   }
32   return retval;
33 }
34
```

Cleaned-up version

Decompile: main - (password)

```
1
2  undefined8 main(void)
3
4  {
5    int result;
6    undefined8 retval;
7    char input [112];
8    char *inputCode;
9    char *flag;
10
11   flag = getenv("FLAG");
12   if (flag == (char *)0x0) {
13     flag = no_flag;
14   }
15   printf("What is the password? ");
16   fflush(stdin);
17   inputCode = fgets(input,99,stdin);
18   if (inputCode == (char *)0x0) {
19     printf("Could not read from input");
20     retval = 1;
21   }
22   else {
23     result = check_password(input);
24     if (result == 0) {
25       puts("Incorrect password");
26     }
27     else {
28       puts(flag);
29     }
30     retval = 0;
31   }
32   return retval;
33 }
34
```

Usually runs on a remote computer, flag is in env variable :(

Cleaned-up version

Decompile: main - (password)

```
1
2  undefined8 main(void)
3
4  {
5    int result;
6    undefined8 retval;
7    char input [112];
8    char *inputCode;
9    char *flag;
10
11   flag = getenv("FLAG");
12   if (flag == (char *)0x0) {
13     flag = no_flag;
14   }
15   printf("What is the password? ");
16   fflush(stdin);
17   inputCode = fgets(input,99,stdin);
18   if (inputCode == (char *)0x0) {
19     printf("Could not read from input");
20     retval = 1;
21   }
22   else {
23     result = check_password(input);
24     if (result == 0) {
25       puts("Incorrect password");
26     }
27     else {
28       puts(flag);
29     }
30     retval = 0;
31   }
32   return retval;
33 }
34
```

Usually runs on a remote computer, flag is in env variable :(

Buffer overflow not possible, buffer is large enough

Cleaned-up version

Decompile: main - (password)

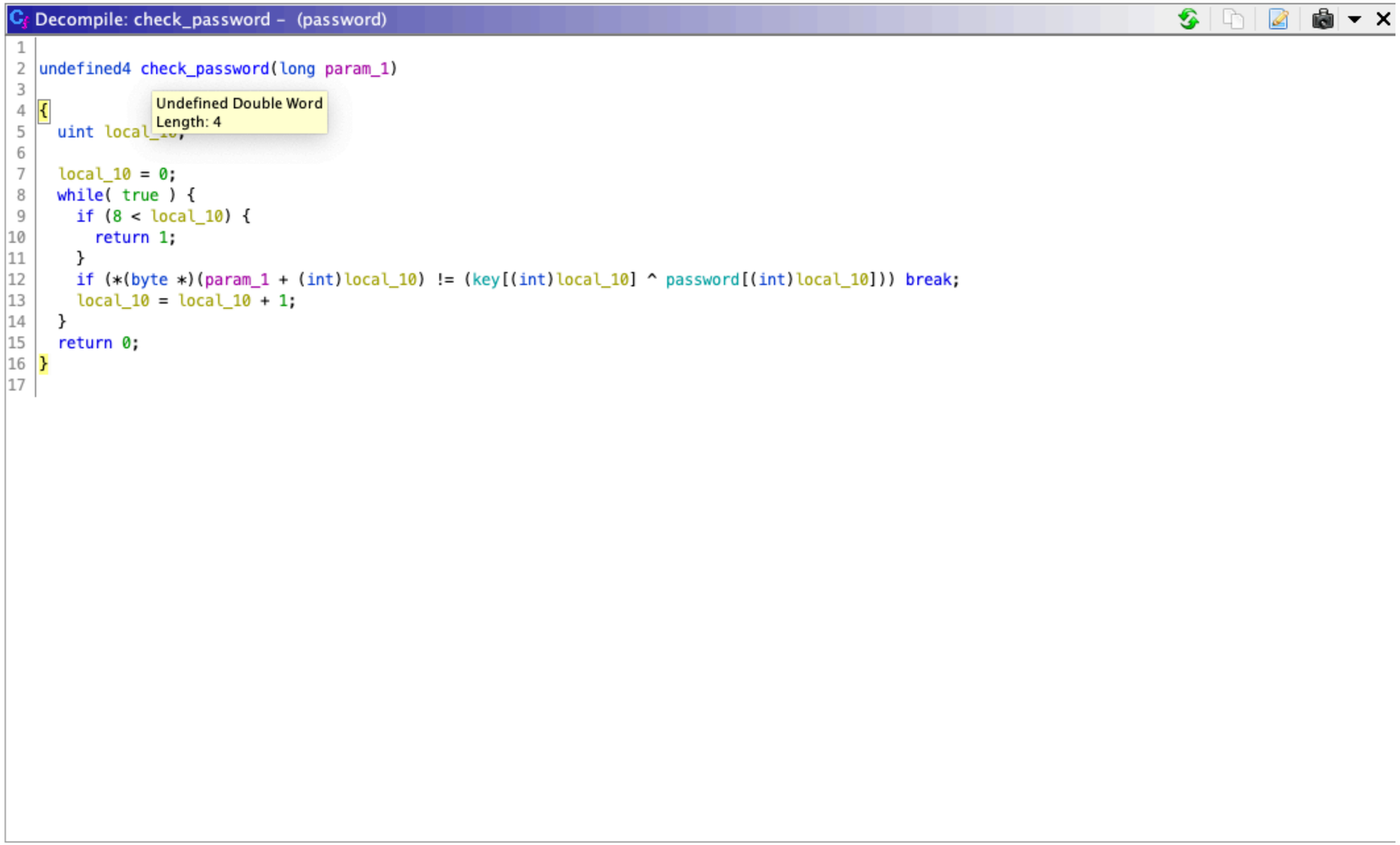
```
1
2  undefined8 main(void)
3
4  {
5    int result;
6    undefined8 retval;
7    char input [112];
8    char *inputCode;
9    char *flag;
10
11   flag = getenv("FLAG");
12   if (flag == (char *)0x0) {
13     flag = no_flag;
14   }
15   printf("What is the password? ");
16   fflush(stdin);
17   inputCode = fgets(input,99,stdin);
18   if (inputCode == (char *)0x0) {
19     printf("Could not read from input");
20     retval = 1;
21   }
22   else {
23     result = check_password(input);
24     if (result == 0) {
25       puts("Incorrect pas
26     }
27     else {
28       puts(flag);
29     }
30     retval = 0;
31   }
32   return retval;
33 }
34
```

Usually runs on a remote computer, flag is in env variable :(

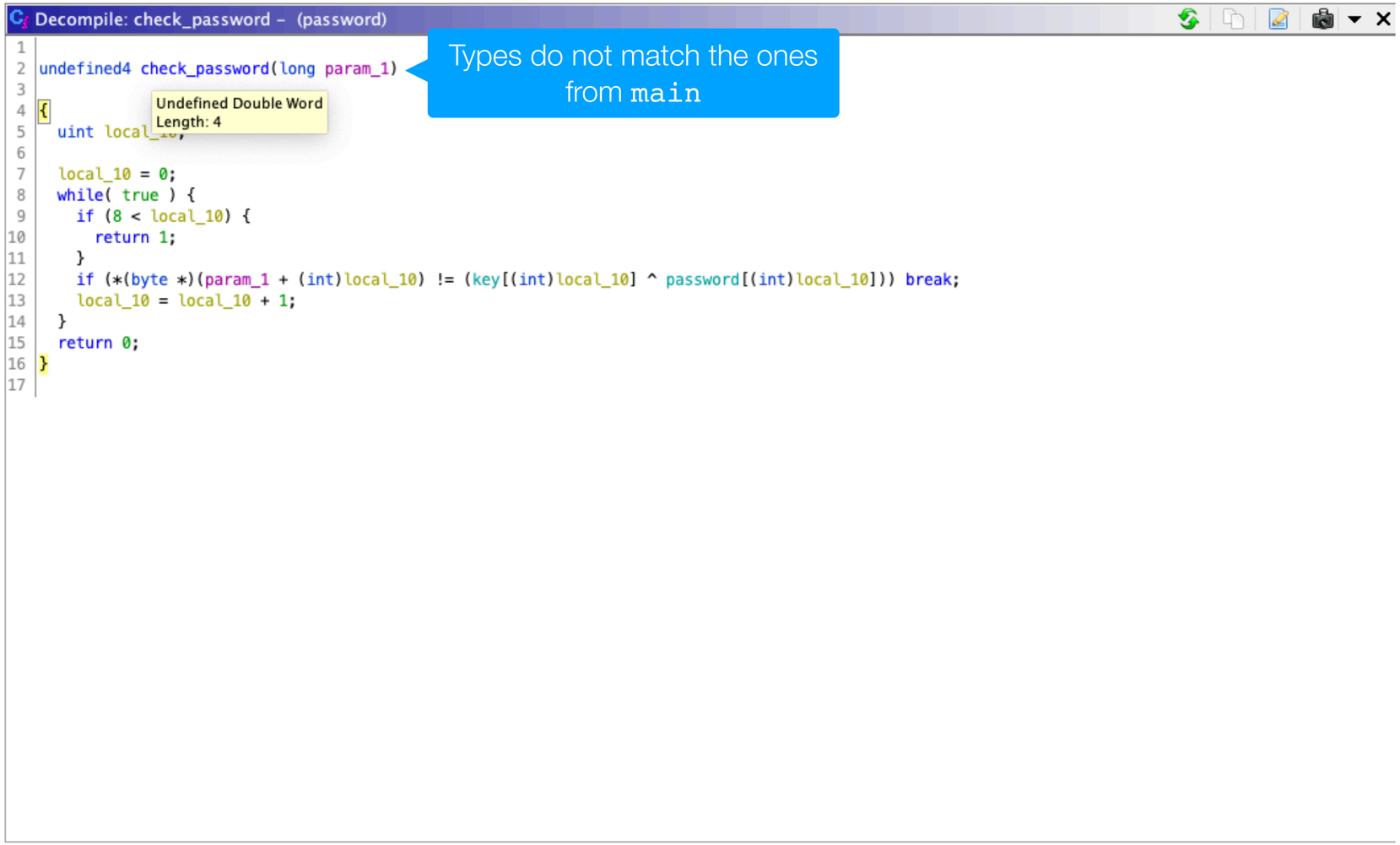
Buffer overflow not possible, buffer is large enough

We should check the `check_password` function

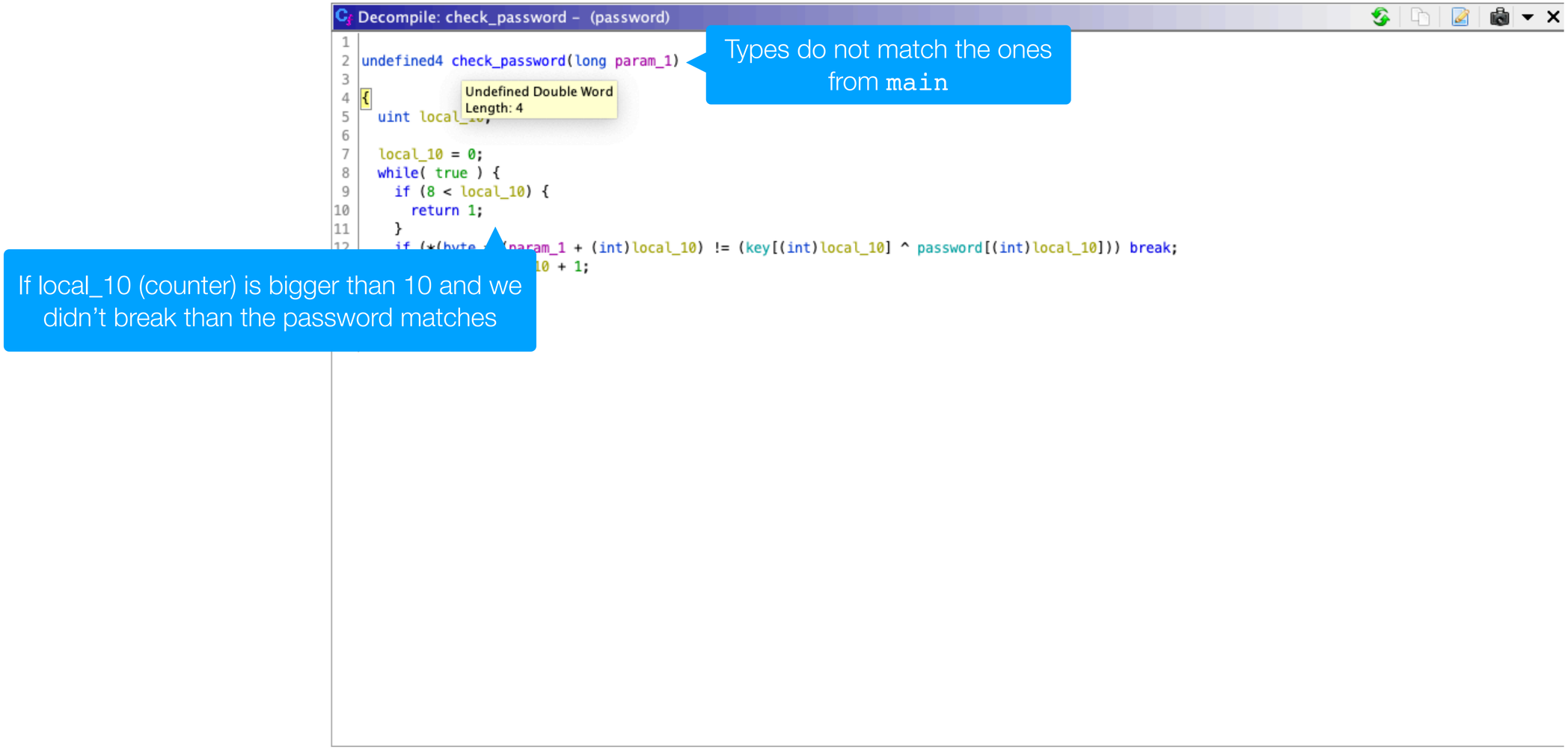
check-password



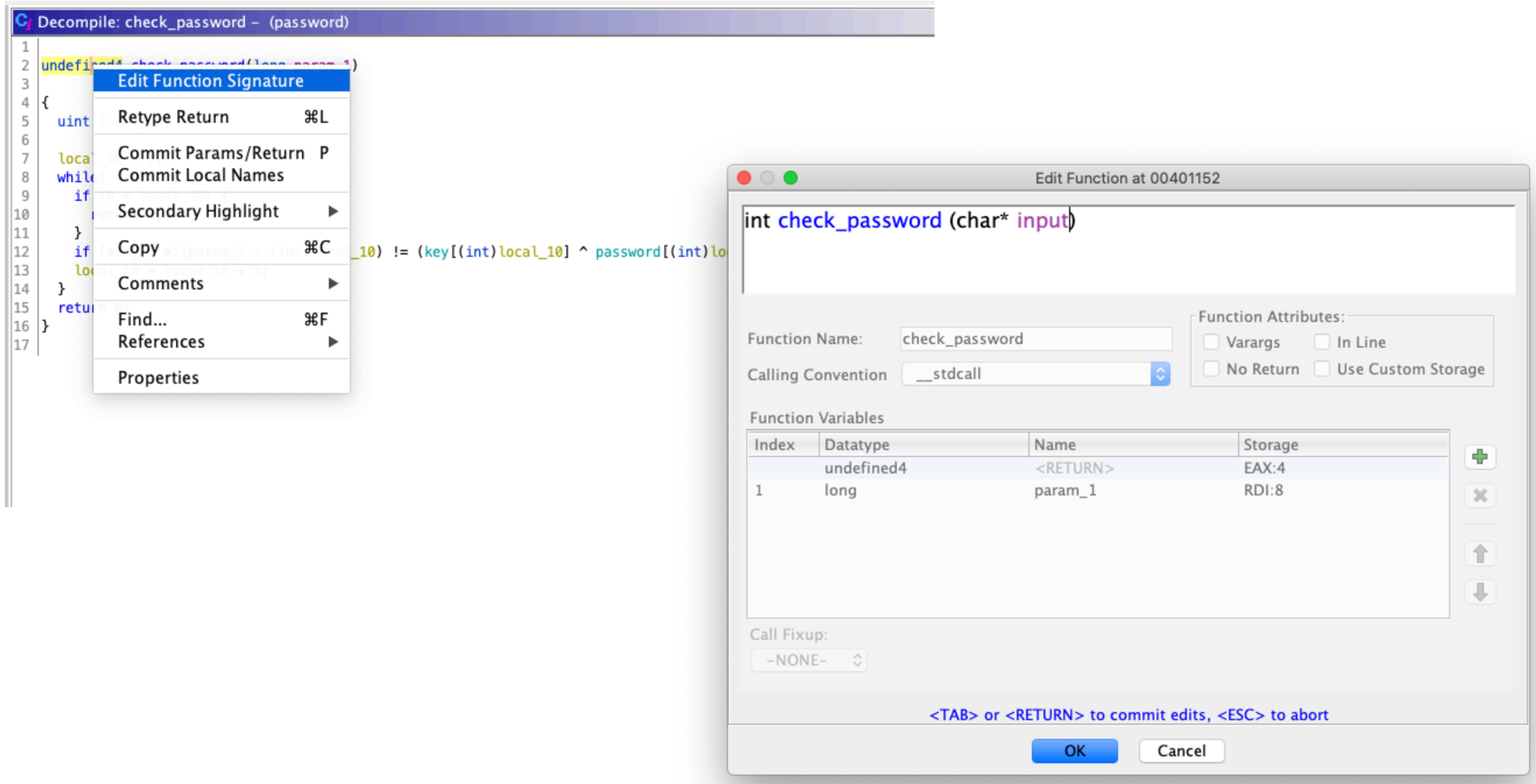
check-password



check-password



check-password clean-up



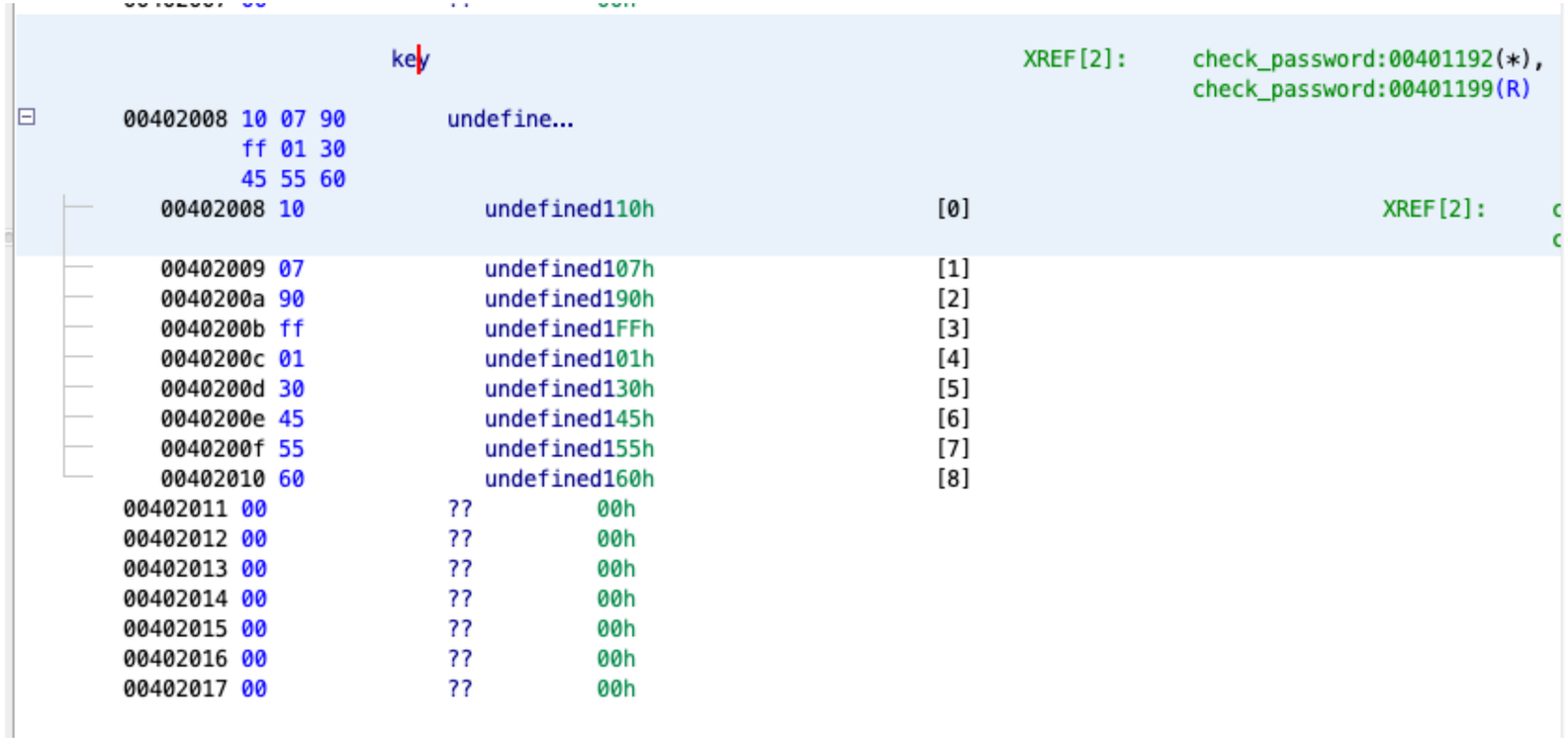
Decompile: check_password - (password)

```
1
2  int check_password(char *input)
3
4  {
5    uint counter;
6
7    counter = 0;
8    while( true ) {
9      if (8 < counter) {
10       return 1;
11     }
12     if (input[(int)counter] != (key[(int)counter] ^ password[(int)counter])) break;
13     counter = counter + 1;
14   }
15   return 0;
16 }
17
```

We check the input character by character

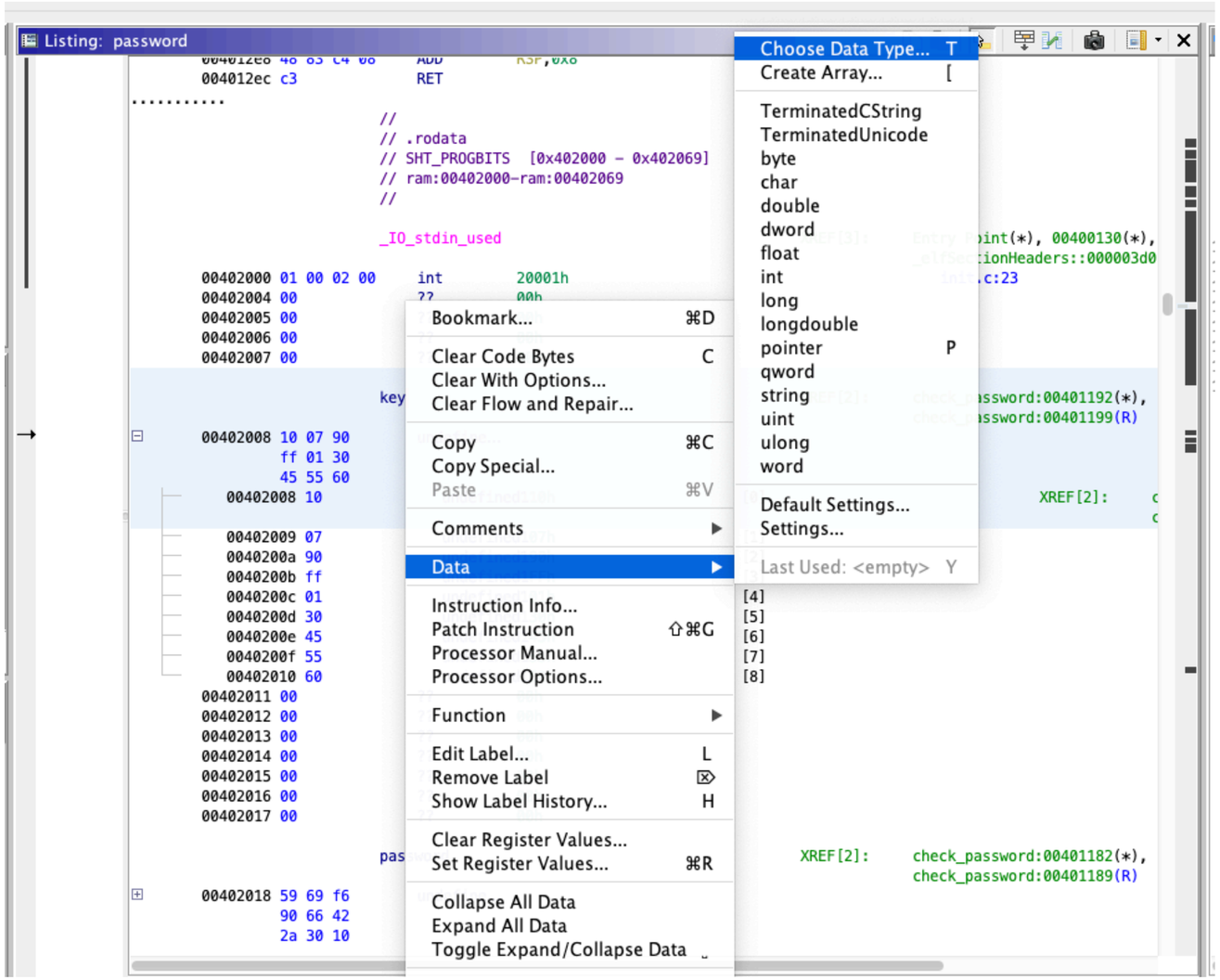
What is **`key`** and **`password`**?

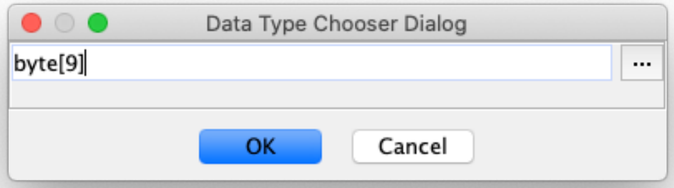
The key



Just double click on the variable to view it in the assembly view

Changing Datatypes






```
                    key
00402008 10 07 90       db[8]
         ff 01 30
         45 55
   00402008 [0]           10h,  7h, 90h, FFh
   0040200c [4]            1h, 30h, 45h, 55h
```

Much more readable

Finding the original password

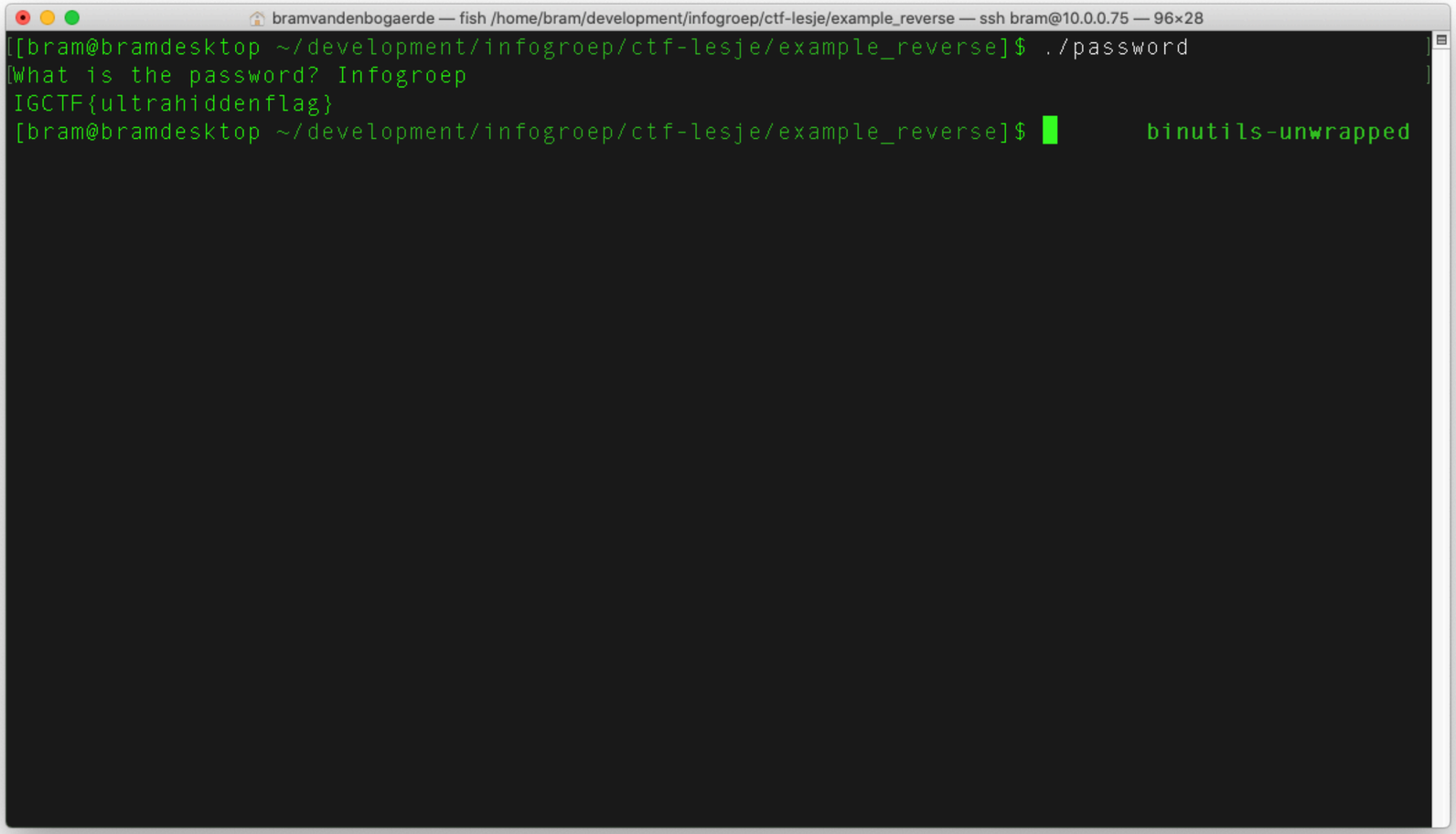
Password bytes obtained in the same way as the key

```
password_bytes = [0x59, 0x69, 0xF6, 0x90, 0x66, 0x42, 0x2A, 0x30, 0x10]

key_bytes = [0x10, 0x07, 0x90, 0xFF, 0x01, 0x30, 0x45, 0x55, 0x60]

password = [ chr(a^b) for (a,b) in zip(password_bytes, key_bytes)]

print("".join(password))
```

To obtain the password we XOR the key with the encrypted password

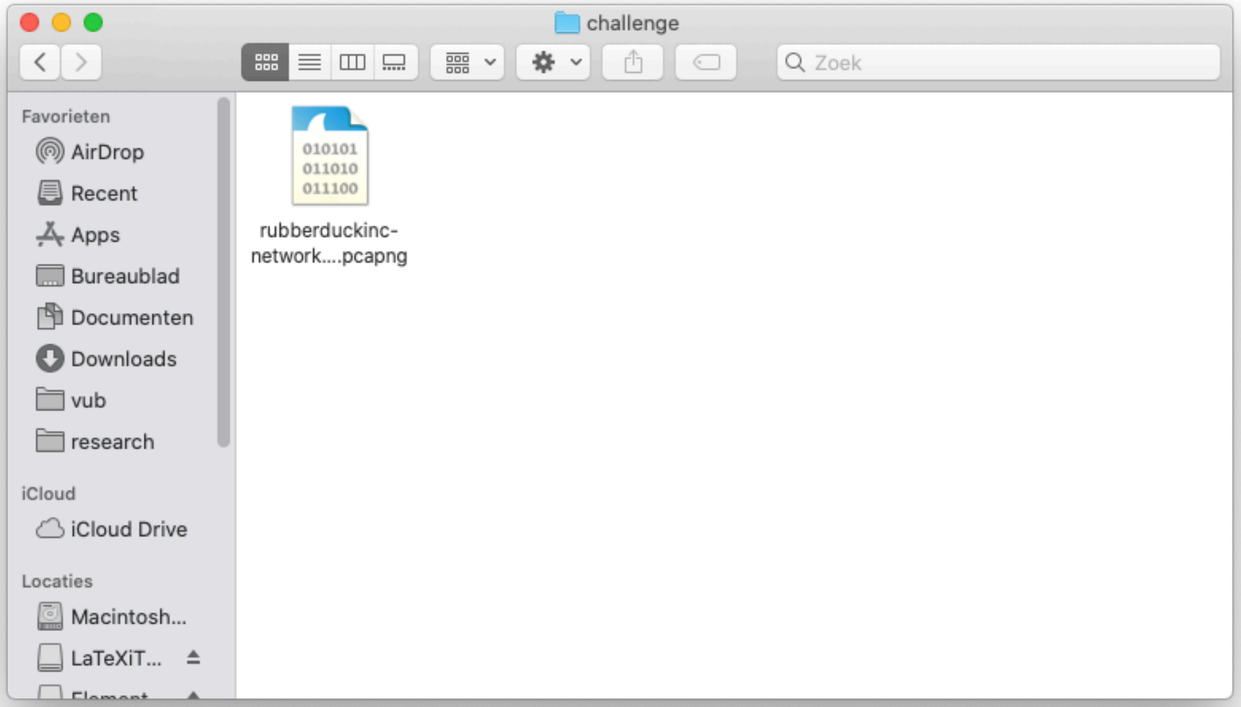
Password is: Infogroep

Testing it out



Network Forensics

The Challenge



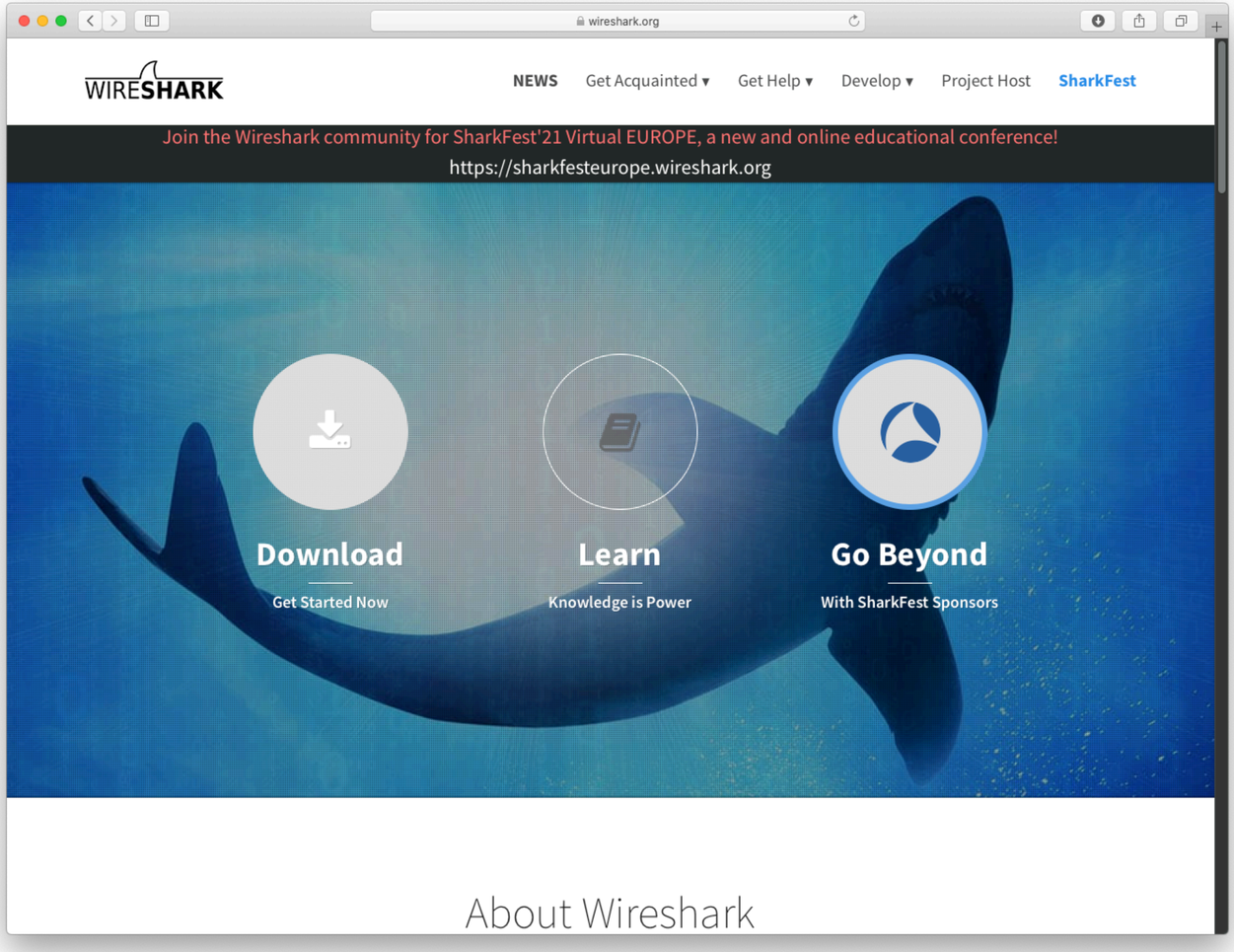
We only got a PCAP file, find information in it to get the flag

What is PCAP file?

“Network analyzers like Wireshark create .pcap files to **collect** and **record packet data from a network**. Packet collection tools like **Wireshark** allow you to collect network traffic and translate it into a format that’s **human-readable**. There are many reasons why PCAP is used to monitor networks. Some of the most common include monitoring bandwidth usage, identifying rogue DHCP servers, detecting malware, DNS resolution, and incident response.”

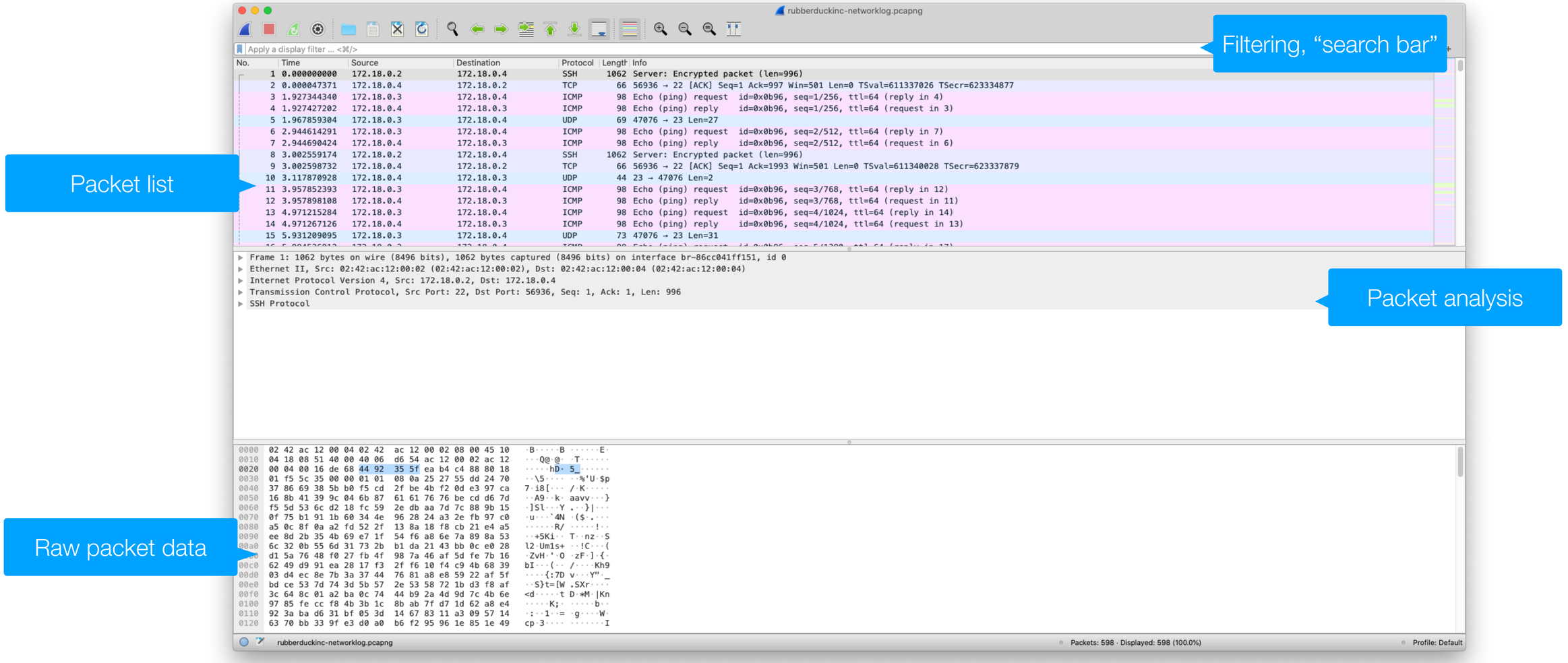
Source: https://www.comparitech.com/net-admin/pcap-guide/

Wireshark

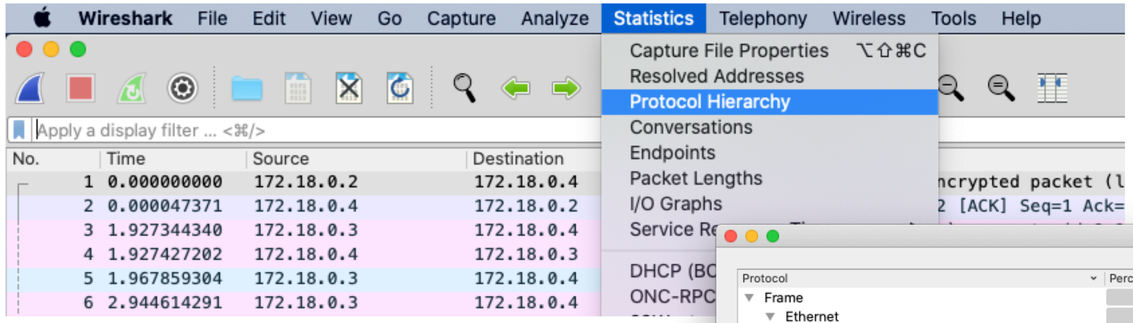


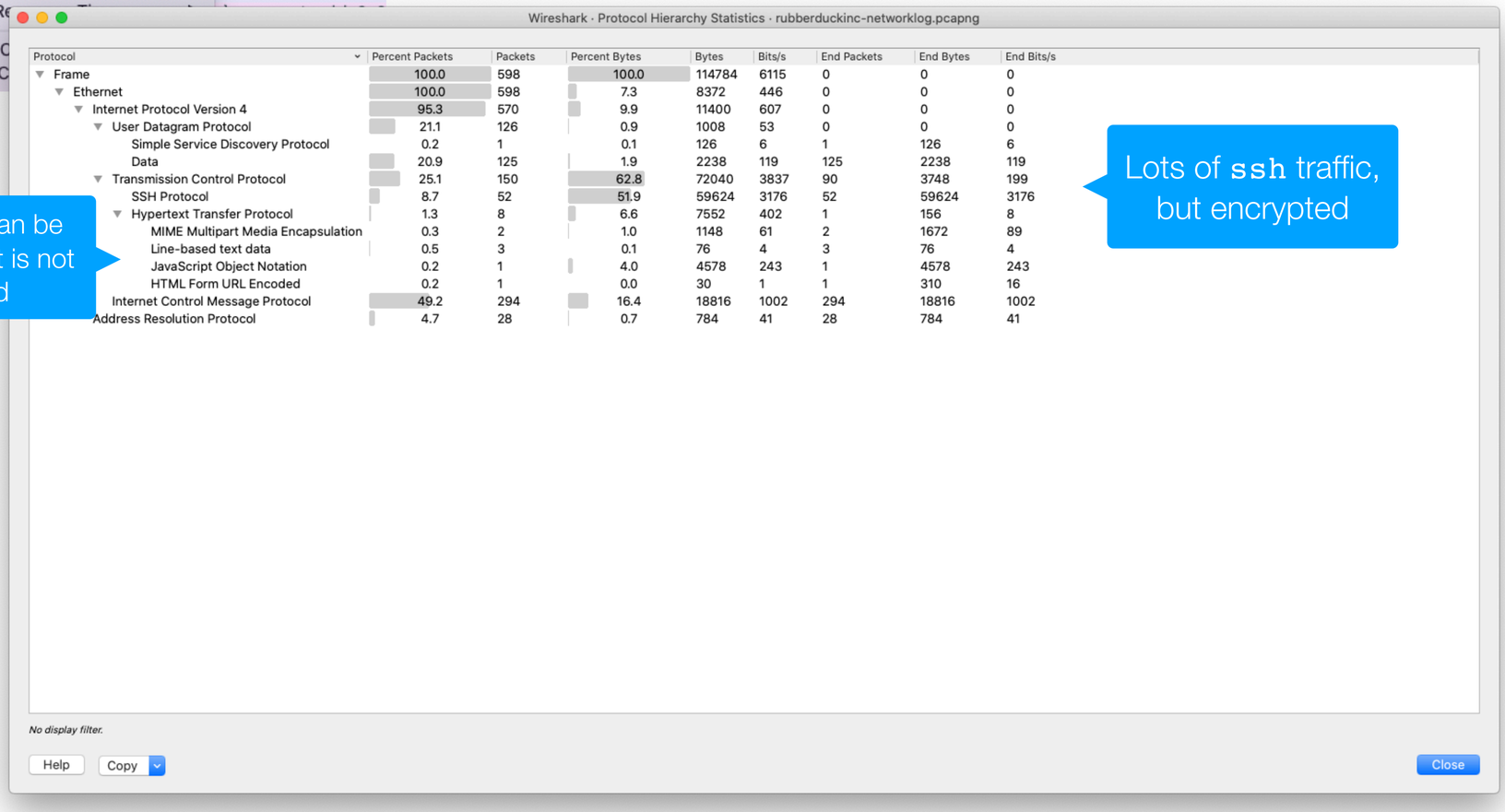
wireshark.org

Viewing capture in Wireshark



Making Sense of the Capture

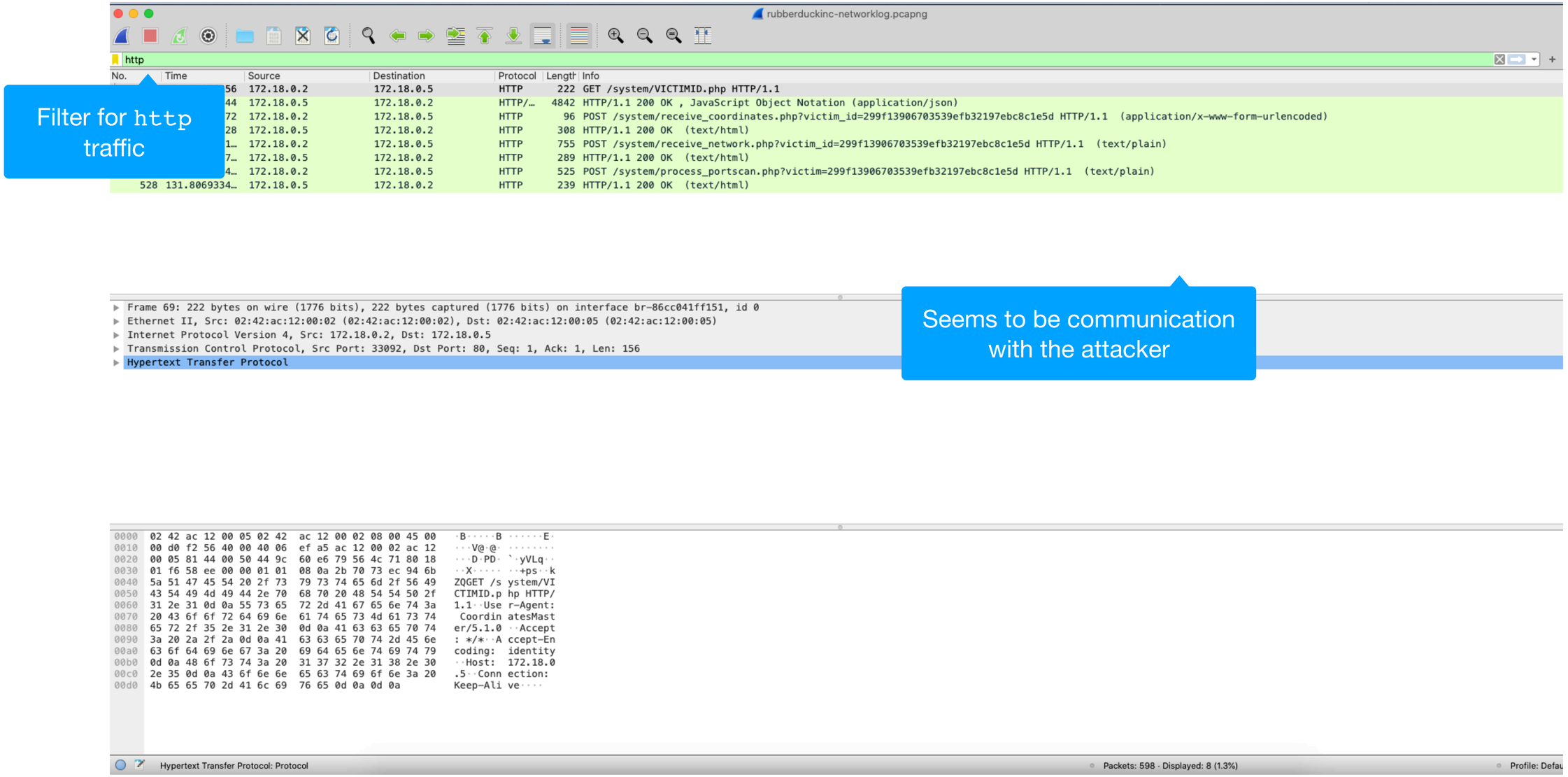




`http` traffic can be inspected, as it is not encrypted

Lots of `ssh` traffic, but encrypted

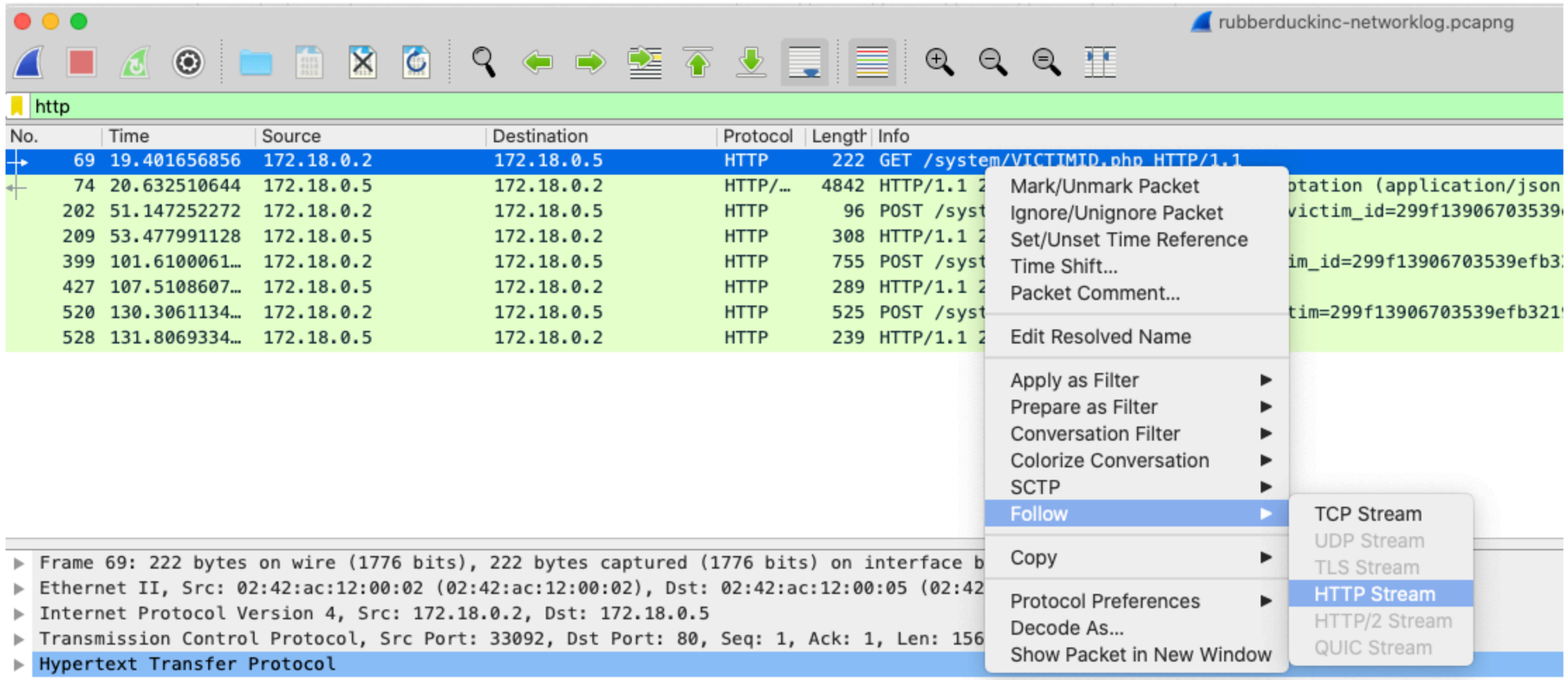
Filtering



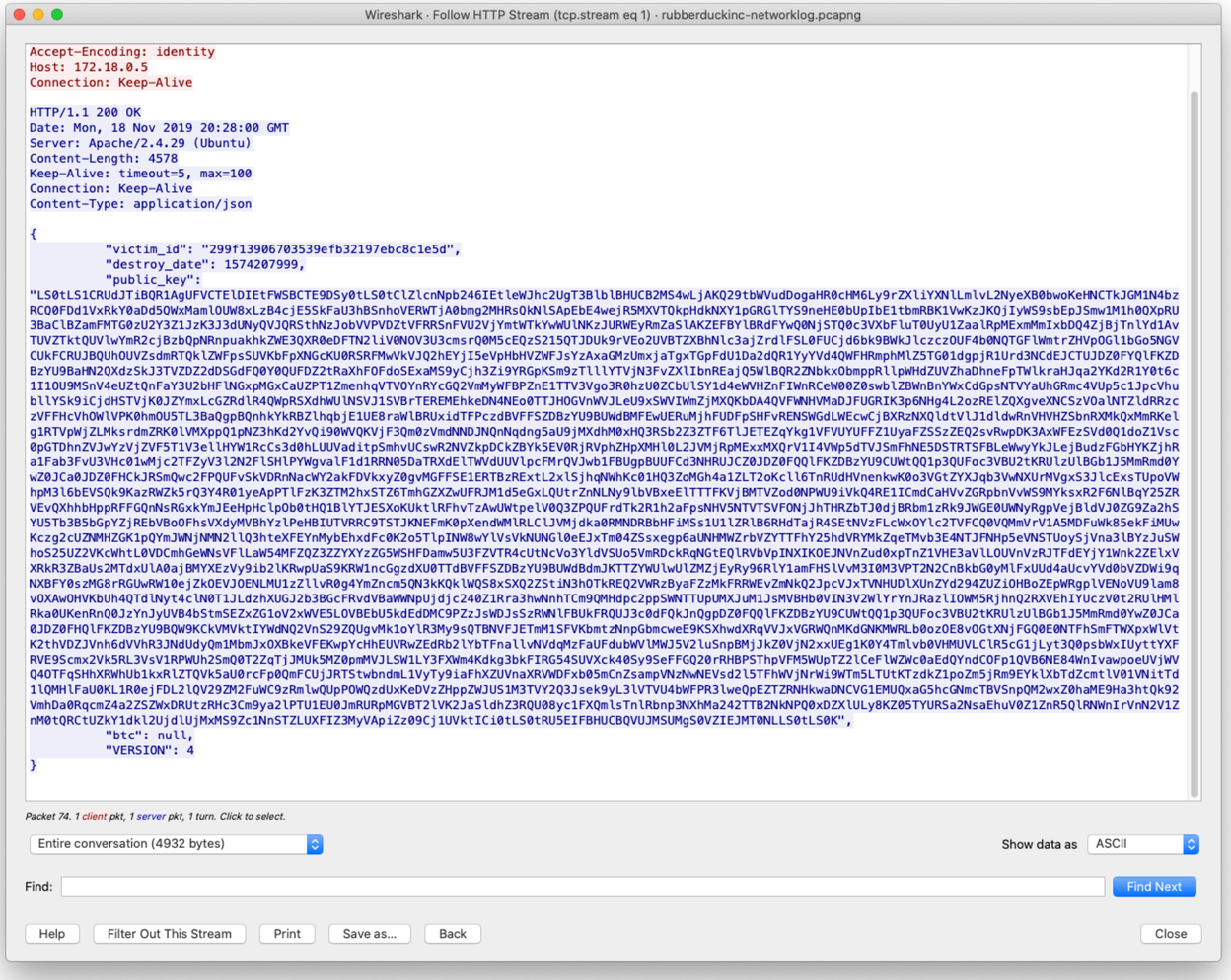
Filter for `http` traffic

Seems to be communication with the attacker

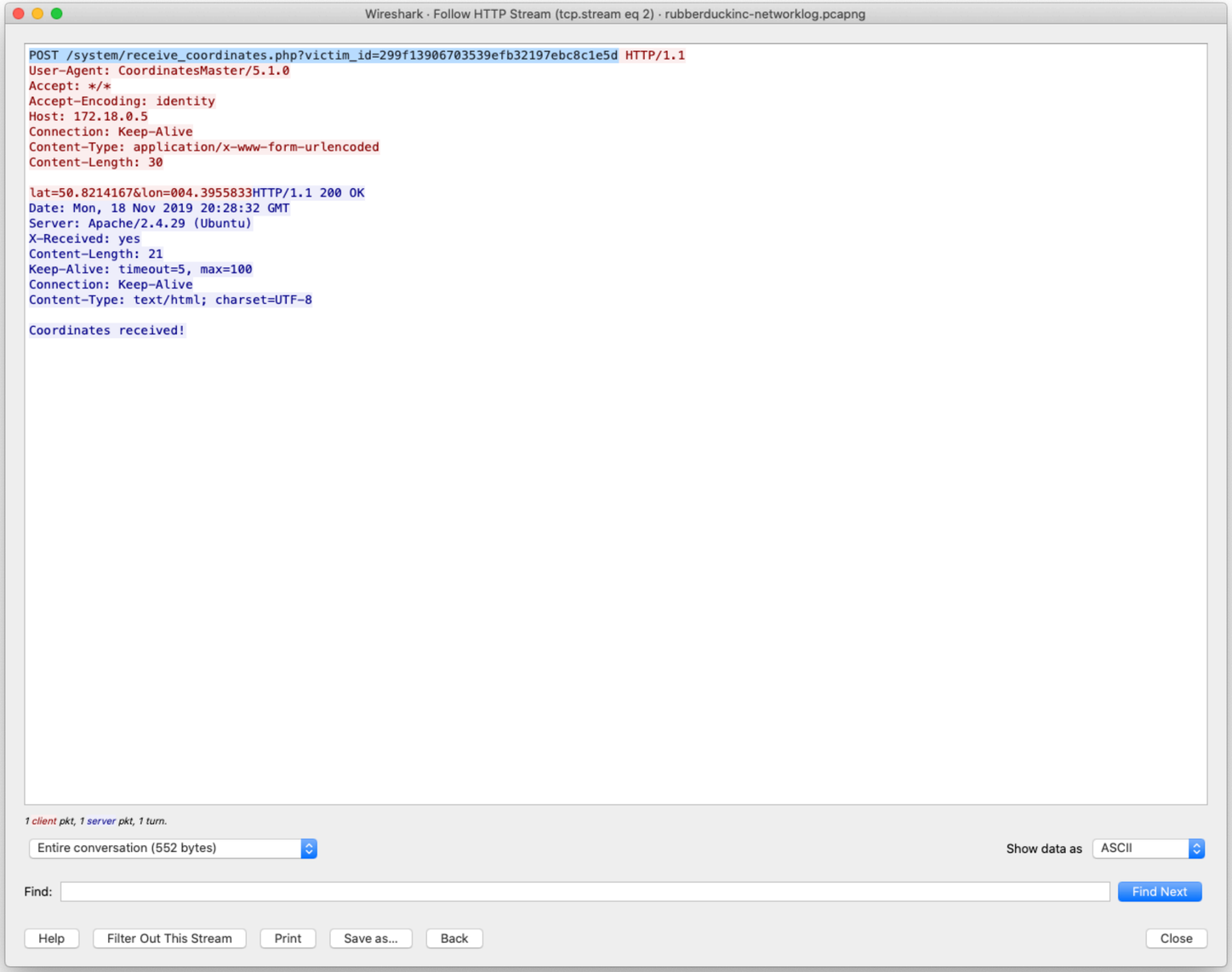
Viewing an entire HTTP connection



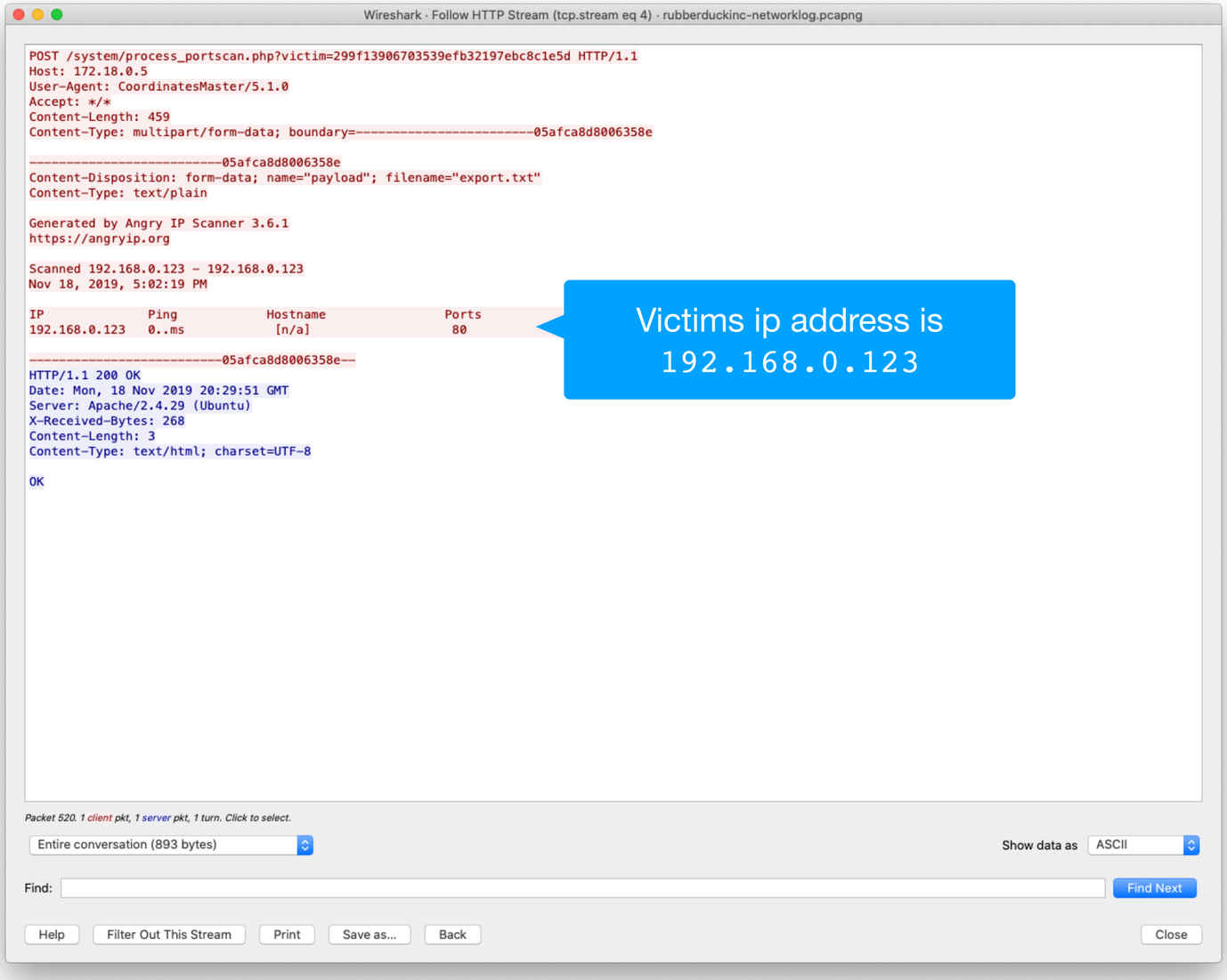
Viewing an entire HTTP connection



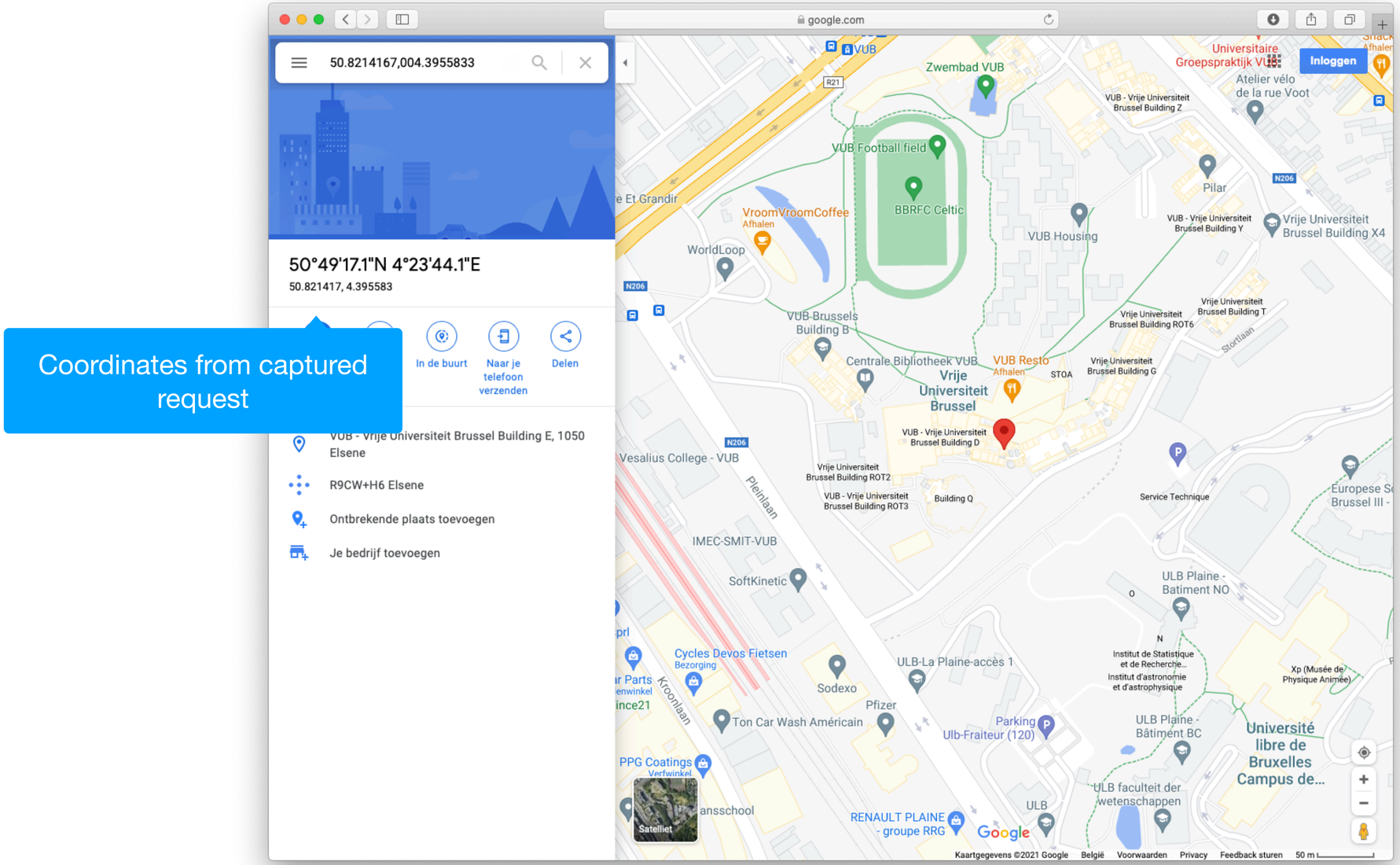
Victim Coordinates



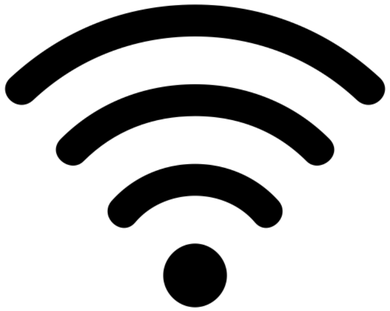
Port Scan Results



Where is the victim?



Final Part



Open WiFi network named
“RubberDuckInvestmentInc” at these coordinates

Surf to http://192.168.0.123 to get the flag