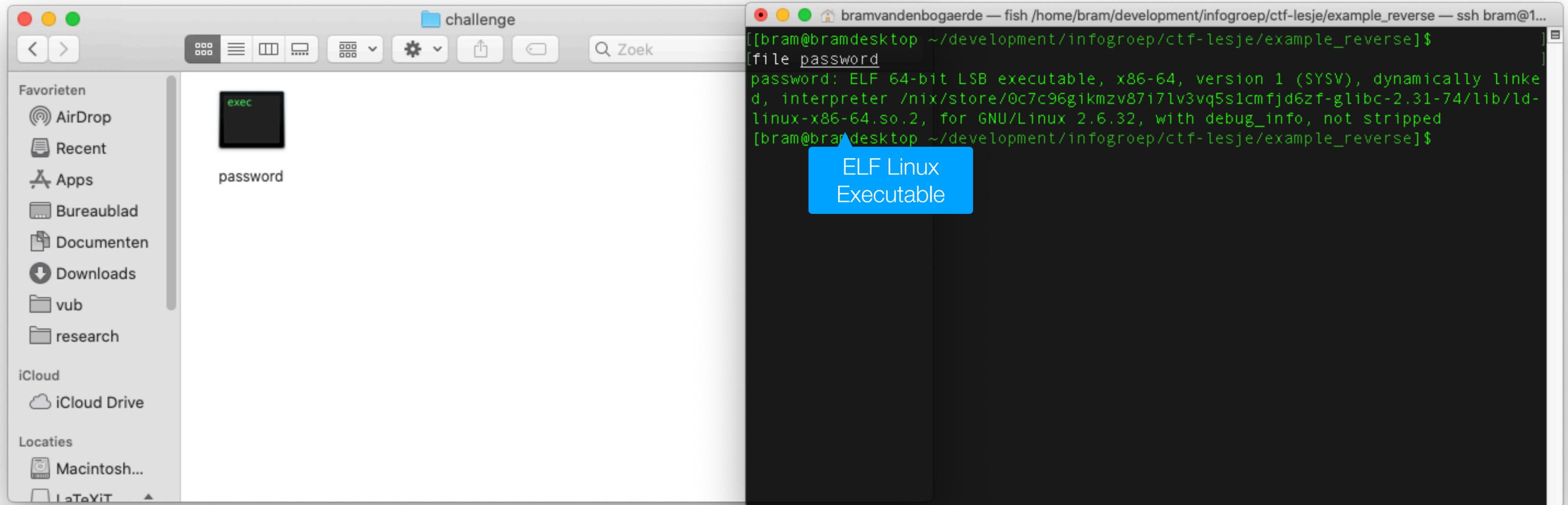


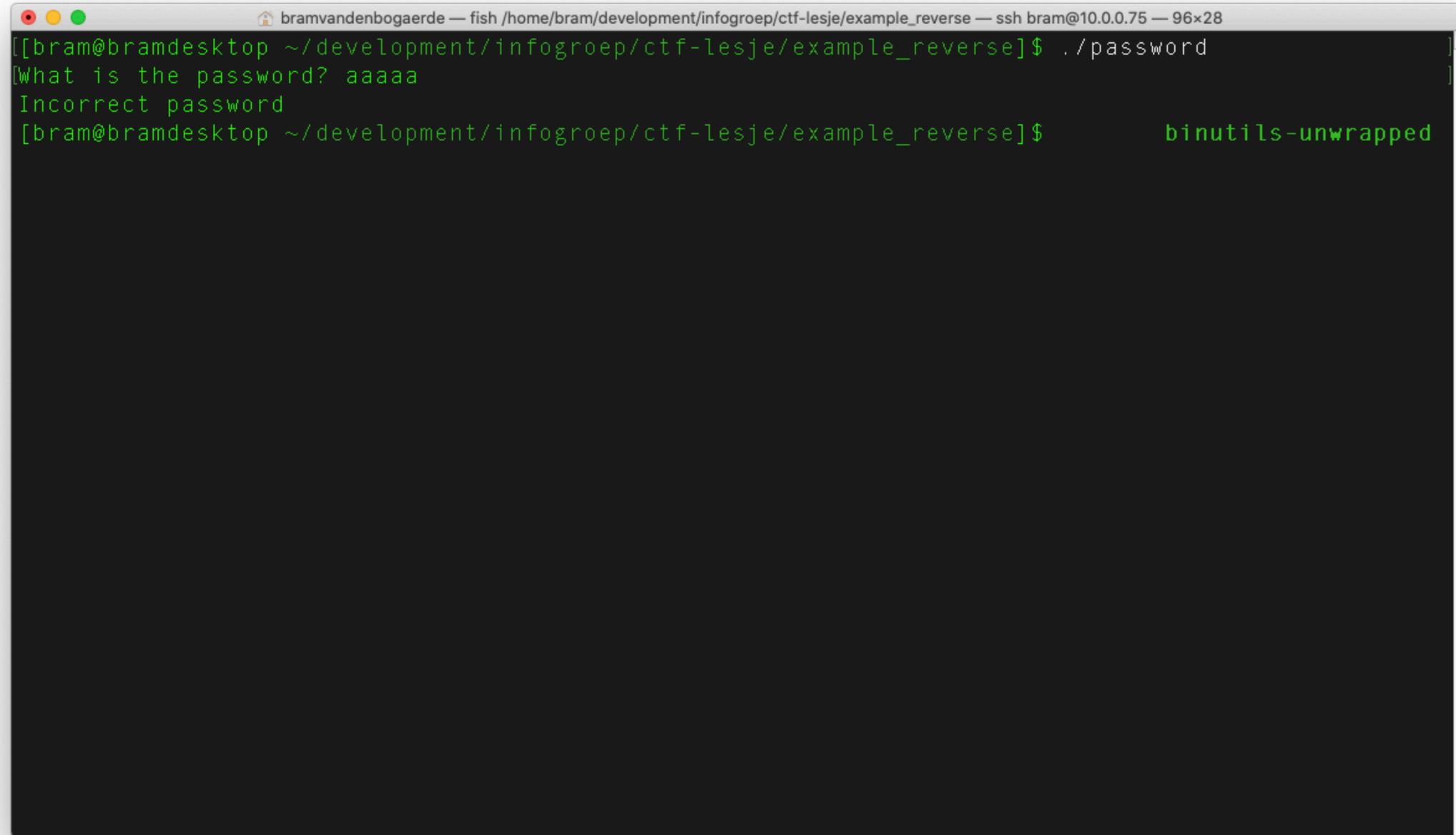
# Reverse Engineering

# The Challenge



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

# The Challenge



```
bramvandenbogaerde — fish /home/bram/development/infogroep/ctf-lesje/example_reverse — ssh bram@10.0.0.75 — 96x28
[[bram@bramdesktop ~/development/infogroep/ctf-lesje/example_reverse]$ ./password ]
[What is the password? aaaaa ]
Incorrect password
[bram@bramdesktop ~/development/infogroep/ctf-lesje/example_reverse]$ binutils-unwrapped
```

# 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](https://link.springer.com/chapter/10.1007/978-3-642-04117-4_31)

# Simple Reversing using objdump

```
objdump -d password
```

```
bramvandenbogaerde — fish /home/bram/development/infogroep/ctf-lesje/example_reverse — ssh bram@10.0.0.75 — 96x28
[bram@bramdesktop ~/development/infogroep/ctf-lesje/example_reverse]$
[objdump -D password]

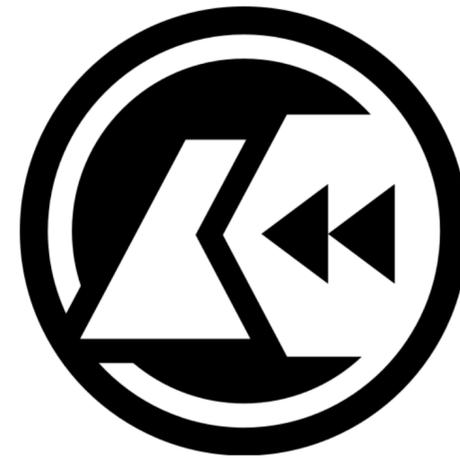
password:      file format elf64-x86-64

Disassembly of section .interp:

00000000004002a8 <.interp>:
4002a8:      2f                (bad)
4002a9:      6e                outsb  %ds:(%rsi), (%dx)
4002aa:      69 78 2f 73 74 6f 72  imul  $0x726f7473,0x2f(%rax),%edi
4002b1:      65 2f            gs (bad)
4002b3:      30 63 37         xor   %ah,0x37(%rbx)
4002b6:      63 39            movslq (%rcx),%edi
4002b8:      36 67 69 6b 6d 7a 76  imul  $0x3738767a,%ss:0x6d(%ebx),%ebp
4002bf:      38 37
4002c1:      69 37 6c 76 33 76  imul  $0x7633766c,(%rdi),%esi
4002c7:      71 35            jno   4002fe <_init-0xd02>
4002c9:      73 31            jae   4002fc <_init-0xd04>
4002cb:      63 6d 66         movslq 0x66(%rbp),%ebp
4002ce:      6a 64            pushq $0x64
4002d0:      36 7a 66         ss jp  400339 <_init-0xcc7>
4002d3:      2d 67 6c 69 62  sub   $0x62696c67,%eax
4002d8:      63 2d 32 2e 33 31  movslq 0x31332e32(%rip),%ebp          # 31733110 <__TMC_EN
D__+0x3132f0e8>
4002de:      2d 37 34 2f 6c  sub   $0x6c2f3437,%eax
4002e3:      69 62 2f 6c 64 2d 6c  imul  $0x6c2d646c,0x2f(%rdx),%esp
```

Not very clear  
what is going on

# Reverse Engineering Tools



Cutter



IDA Pro



**And more...**

# Ghidra Download



**GHIDRA**

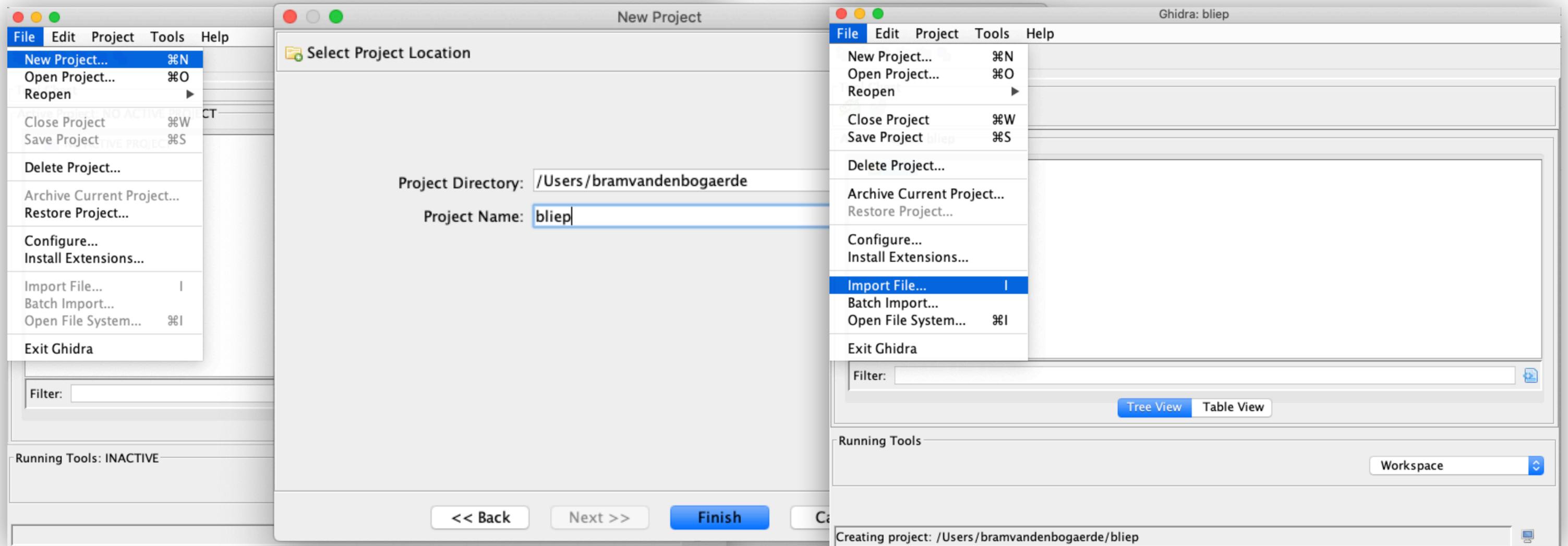
A software reverse engineering (SRE) suite of tools developed by NSA's Research Directorate in support of the Cybersecurity mission

[Download Ghidra v9.2.2](#)

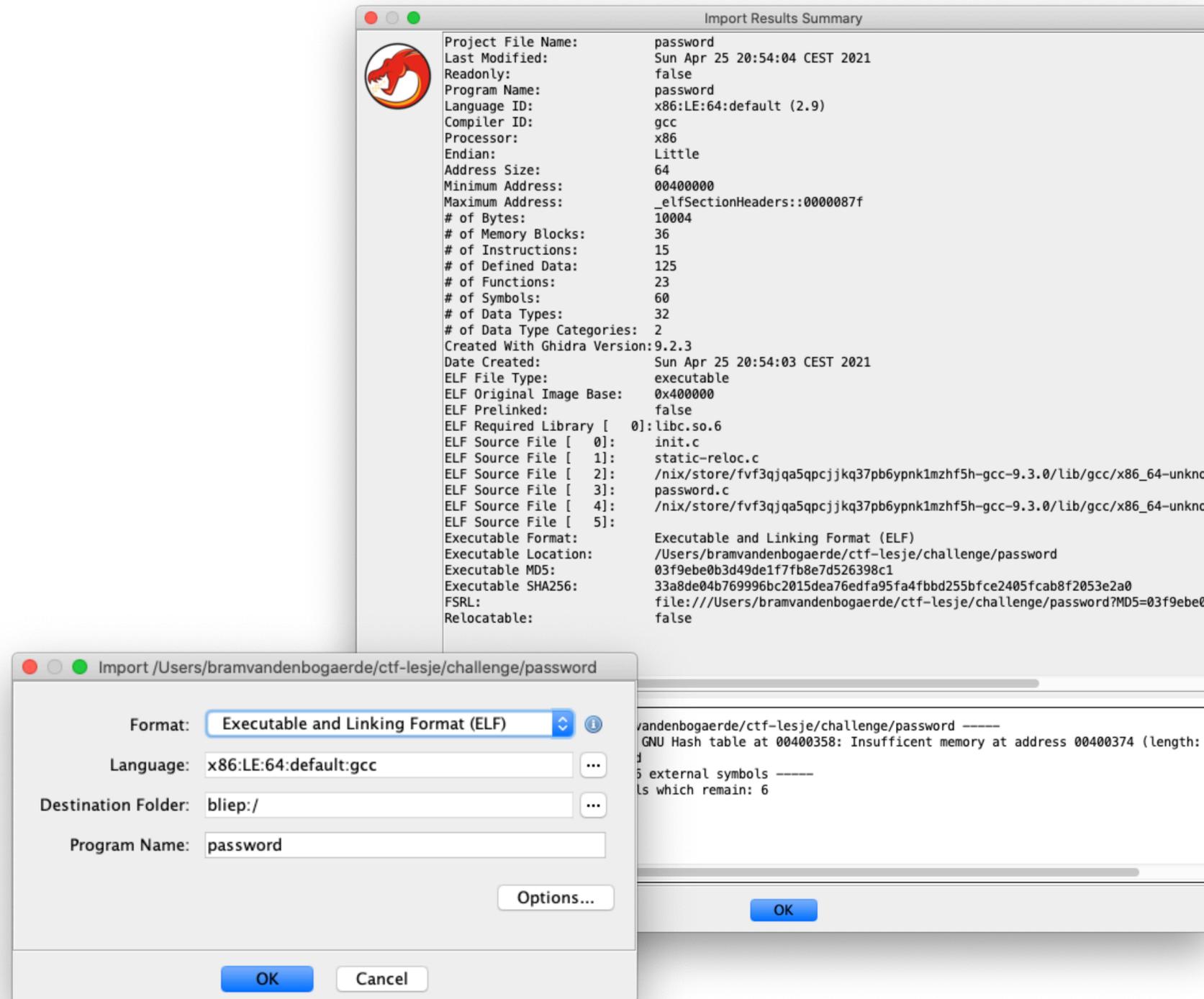
[SHA-256](#) [Latest](#) [Releases](#) [Source](#)

<http://ghidra-sre.org>

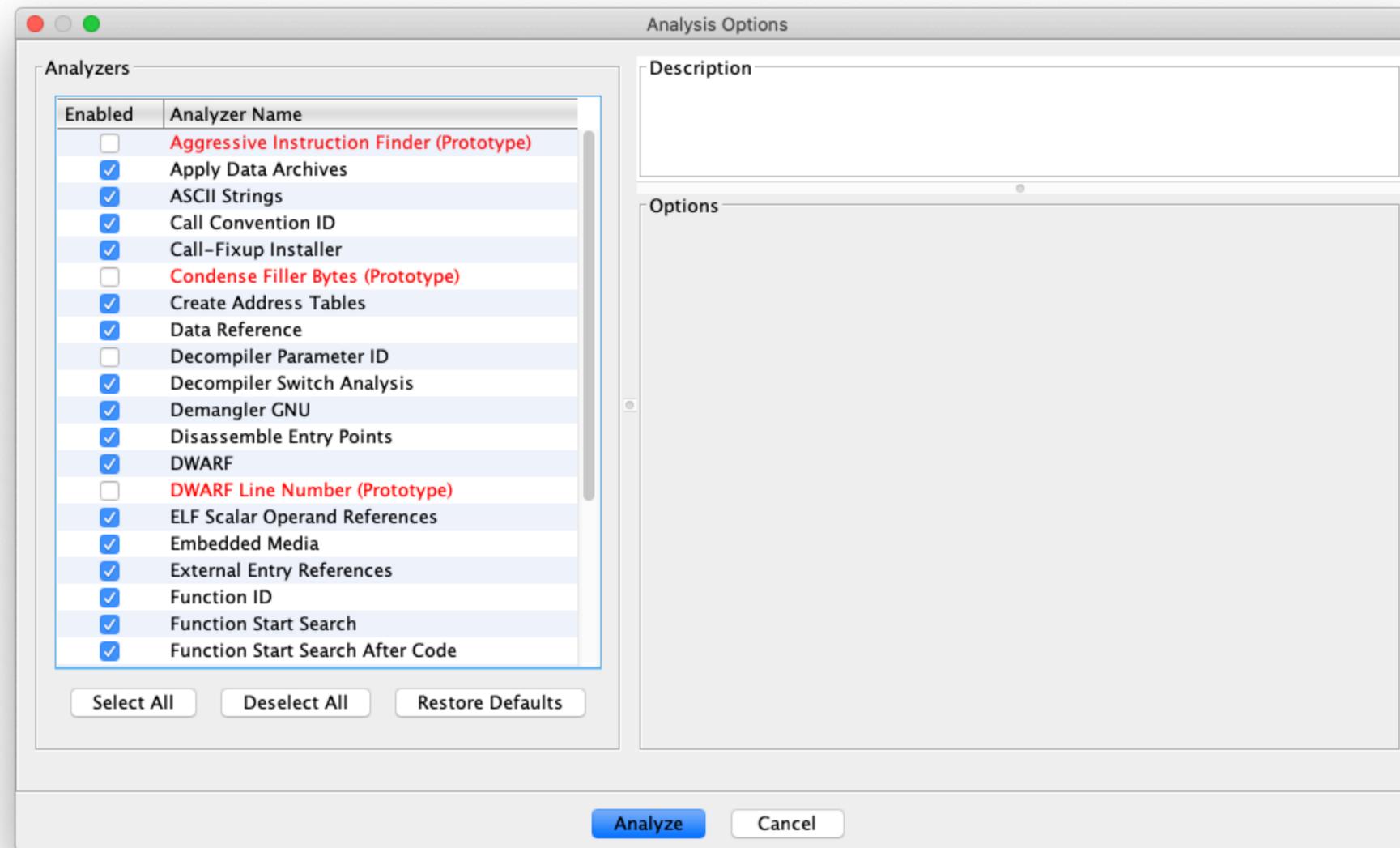
# Opening Executable (1)



# Opening an executable (2)



# Analyzing an executable



You can customise this but the defaults are mostly fine

# A tour through Ghidra

The screenshot shows the Ghidra IDE interface with several panels and annotations:

- Program Trees:** Shows the file structure of the password program, including sections like .bss, .data, .got, .dynamic, .fini\_array, .init\_array, .eh\_frame, .eh\_frame\_hdr, and .rodata.
- Symbol Tree:** Lists functions such as `__do_global_dtors_aux`, `__gmon_start__`, `__libc_csu_fini`, `__libc_csu_init`, `__libc_start_main`, `_fini`, `_init`, `_start`, and `check_password`.
- Data Type Manager:** Shows data types for the password program, including `password` and `generic_clib_64`.
- Listing: password:** Displays assembly code for the ELF header, including fields like `e_ident_magi...`, `e_ident_class`, `e_ident_data`, `e_ident_vers...`, `e_ident_osabi`, `e_ident_abiv...`, `e_ident_pad`, `e_type`, `e_machine`, `e_version`, `e_entry`, `e_phoff`, `e_shoff`, `e_flags`, `e_ehsize`, `e_phentsize`, `e_phnum`, `e_shentsize`, `e_shnum`, and `e_shstrndx`.
- Decompiler:** Shows the decompiled C code, currently displaying `1 | No Function`.
- Console - Scripting:** A panel at the bottom for running scripts.

Annotations with blue callouts point to specific features:

- "Look here for functions" points to the Symbol Tree.
- "Decompiled C code (if available)" points to the Decompiler panel.
- "Assembly code" points to the Listing: password panel.
- "Useful to find structs or other types" points to the Data Type Manager.

# Finding the main function

The screenshot displays the CodeBrowser interface for a binary named 'password'. The main window shows assembly code for the 'main' function, which is highlighted in blue. The assembly code includes instructions like JMP, ADD, MOV, CMP, JBE, PUSH, MOV, ADD, and LEA. The decompiled C code on the right shows the function signature 'main(void)' and its implementation, including variable declarations, environment variable retrieval, and password checking logic. The left sidebar shows the Program Tree and Symbol Tree, with 'main' selected in the Symbol Tree. The bottom status bar shows the current instruction '004011be PUSH RBP'.

```
Listing: password
004011ab eb 0c JMP LAB_004011b9
LAB_004011ad
004011ad 83 45 f8 01 ADD dword ptr [RBP + local_10],0x1
LAB_004011b1
004011b1 8b 45 f8 MOV EAX,dword ptr [RBP + local_10]
004011b4 83 f8 08 CMP EAX,0x8
004011b7 76 b1 JBE LAB_0040116a
LAB_004011b9
004011b9 8b 45 fc MOV EAX,dword ptr [RBP + local_c]
004011bc 5d POP RBP
004011bd c3 RET

*****
* FUNCTION
*****
undefined main()
AL:1 <RETURN>
undefined8 Stack[-0x10]:8 local_10
undefined8 Stack[-0x18]:8 local_18
undefined1 Stack[-0x88]:1 local_88
main
XREF[1]: 004011a2(j)
XREF[1]: 00401168(j)
XREF[1]: 004011ab(j)
XREF[4]: 004011d2(W),
004011d6(R),
004011e4(W),
00401259(R)
XREF[2]: 00401226(W),
0040122a(R)
XREF[2]: 00401215(*),
00401249(*)
XREF[5]: Entry Point(*),
_start:0040109d(*),
_start:0040109d(*), 0040209
00402140(*)
= 46h F

004011be 55 PUSH RBP
004011bf 48 89 e5 MOV RBP,RSP
004011c2 48 83 c4 80 ADD RSP,-0x80
004011c6 48 8d 3d LEA RDI,[DAT_00402021]
54 0e 00 00
```

```
Decompile: main - (password)
1
2 undefined8 main(void)
3
4 {
5     int iVar1;
6     undefined8 uVar2;
7     char local_88 [112];
8     char *local_18;
9     char *local_10;
10
11     local_10 = getenv("FLAG");
12     if (local_10 == (char *)0x0) {
13         local_10 = no_flag;
14     }
15     printf("What is the password? ");
16     fflush(stdin);
17     local_18 = fgets(local_88,99,stdin);
18     if (local_18 == (char *)0x0) {
19         printf("Could not read from input");
20         uVar2 = 1;
21     }
22     else {
23         iVar1 = check_password(local_88);
24         if (iVar1 == 0) {
25             puts("Incorrect password");
26         }
27         else {
28             puts(local_10);
29         }
30         uVar2 = 0;
31     }
32     return uVar2;
33 }
34
```

# Making Decompile Readable

The screenshot shows the CodeBrowser interface for a file named 'password'. The main window displays the decompiled C code for the 'main' function. A dialog box titled 'Rename Local Variable' is open, showing the variable 'local\_10' being renamed to 'flag'. A blue callout box points to the dialog with the text 'Use L to rename a variable'. The interface includes a menu bar (File, Edit, Analysis, Graph, Navigation, Search, Select, Tools, Window, Help), a toolbar, and several panels: Program Trees, Symbol Tree, Data Type Manager, and Console - Scripting. The Symbol Tree shows the 'main' function selected. The Data Type Manager shows 'password' and 'generic\_clib\_64' as data types. The Console - Scripting panel is empty. The status bar at the bottom shows the current location: '004011cd main CALL 0x00401030'.

```
1 undefined8 main(void)
2
3
4 {
5     int iVar1;
6     undefined8 uVar2;
7     char local_88 [112];
8     char *local_18;
9     char *local_10;
10
11     local_10 = getenv("FLAG");
12     if (local_10 == (char *)0x0) {
13         local_10 = no_flag;
14     }
15     printf("What is the password? ");
16     fflush(stdin);
17     local_18 = fgets(local_88,99,stdin);
18     if (local_18 == (char *)0x0) {
19         printf("#Could not read from input");
20     }
21 }
22
23
24
25
26
27
28     else {
29         puts(local_10);
30     }
31     uVar2 = 0;
32 }
33 return uVar2;
34 }
```

Rename Local Variable dialog box:  
Rename local\_10:   
OK Cancel

Use L to rename a variable

# Cleaned-up version

```
C:\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

```
C:\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

```
C:\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

```
C:\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

```
Decompile: check_password - (password)
1
2 undefined4 check_password(long param_1)
3
4 {
5     uint local_10;
6
7     local_10 = 0;
8     while( true ) {
9         if (8 < local_10) {
10            return 1;
11        }
12        if (*(byte*)(param_1 + (int)local_10) != (key[(int)local_10] ^ password[(int)local_10])) break;
13        local_10 = local_10 + 1;
14    }
15    return 0;
16 }
17
```

# check-password

```
Decompile: check_password - (password)
1
2 undefined4 check_password(long param_1)
3
4 {
5     uint local_10;
6
7     local_10 = 0;
8     while( true ) {
9         if ( 8 < local_10 ) {
10            return 1;
11        }
12        if (*(byte*)(param_1 + (int)local_10) != (key[(int)local_10] ^ password[(int)local_10])) break;
13        local_10 = local_10 + 1;
14    }
15    return 0;
16 }
17
```

Types do not match the ones from main

Undefined Double Word  
Length: 4

# check-password

```
Decompile: check_password - (password)
1
2 undefined4 check_password(long param_1)
3
4 {
5     uint local_10;
6
7     local_10 = 0;
8     while( true ) {
9         if ( 8 < local_10 ) {
10            return 1;
11        }
12        if (*(byte)param_1 + (int)local_10 != (key[(int)local_10] ^ password[(int)local_10])) break;
13        local_10 + 1;
14    }
15    return 0;
16 }
```

Types do not match the ones from main

Undefined Double Word Length: 4

If local\_10 (counter) is bigger than 10 and we didn't break than the password matches

# check-password clean-up

```
Decompile: check_password - (password)
1
2 undefined4 check_password(long param_1)
3
4 {
5   uint local_10;
6
7   local_10 = param_1;
8   while (local_10)
9   {
10    if (local_10 < 0)
11    {
12     if (local_10 < -10)
13     {
14      local_10 = -10;
15     }
16    }
17    return local_10;
18 }
```

- Edit Function Signature
- Retype Return ⌘L
- Commit Params/Return P
- Commit Local Names
- Secondary Highlight ▶
- Copy ⌘C
- Comments ▶
- Find... ⌘F
- References ▶
- Properties

Edit Function at 00401152

```
int check_password (char* input)
```

Function Name:

Calling Convention:

Function Attributes:

- Varargs
- In Line
- No Return
- Use Custom Storage

Function Variables

Index	Datatype	Name	Storage
	undefined4	<RETURN>	EAX:4
1	long	param_1	RDI:8

Call Fixup:

<TAB> or <RETURN> to commit edits, <ESC> to abort

OK Cancel

```
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

```
key
XREF [2]: check_password:00401192(*),
check_password:00401199(R)
00402008 10 07 90  undefine...
          ff 01 30
          45 55 60
00402008 10      undefined110h      [0]      XREF [2]: c
00402009 07      undefined107h      [1]
0040200a 90      undefined190h      [2]
0040200b ff       undefined1FFh      [3]
0040200c 01      undefined101h      [4]
0040200d 30      undefined130h      [5]
0040200e 45      undefined145h      [6]
0040200f 55      undefined155h      [7]
00402010 60      undefined160h      [8]
00402011 00      ??      00h
00402012 00      ??      00h
00402013 00      ??      00h
00402014 00      ??      00h
00402015 00      ??      00h
00402016 00      ??      00h
00402017 00      ??      00h
```

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

# Changing Datatypes

The screenshot shows a debugger window titled 'Listing: password'. The assembly code is as follows:

```
004012ec c3      RET
.....
//
// .rodata
// SHT_PROGBITS [0x402000 - 0x402069]
// ram:00402000-ram:00402069
//
_IIO_stdin_used

00402000 01 00 02 00    int      20001h
00402004 00
00402005 00
00402006 00
00402007 00

00402008 10 07 90    key
          ff 01 30
          45 55 60
00402008 10

00402009 07
0040200a 90
0040200b ff
0040200c 01
0040200d 30
0040200e 45
0040200f 55
00402010 60

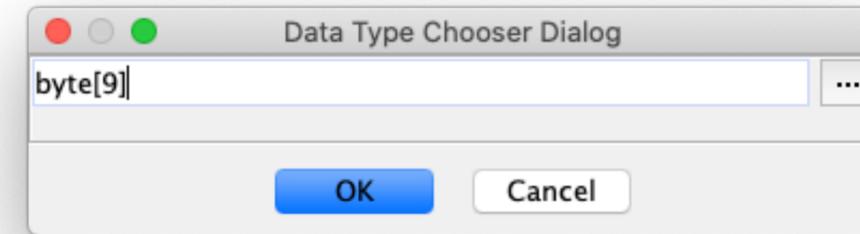
00402011 00
00402012 00
00402013 00
00402014 00
00402015 00
00402016 00
00402017 00

00402018 59 69 f6
          90 66 42
          2a 30 10
```

A context menu is open over the data at address 00402008. The 'Data' option is selected, which has opened a sub-menu titled 'Choose Data Type...'. The sub-menu contains the following options:

- Create Array... [
- TerminatedCString
- TerminatedUnicode
- byte
- char
- double
- dword
- float
- int
- long
- longdouble
- pointer P
- qword
- string
- uint
- ulong
- word

Other options in the main context menu include Bookmark..., Clear Code Bytes, Clear With Options..., Clear Flow and Repair..., Copy, Copy Special..., Paste, Comments, Data (selected), Instruction Info..., Patch Instruction, Processor Manual..., Processor Options..., Function, Edit Label..., Remove Label, Show Label History..., Clear Register Values..., Set Register Values..., Collapse All Data, Expand All Data, and Toggle Expand/Collapse Data.



The data view shows a variable named 'key' at address 00402008. The data is displayed in a more readable format:

```
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]
```

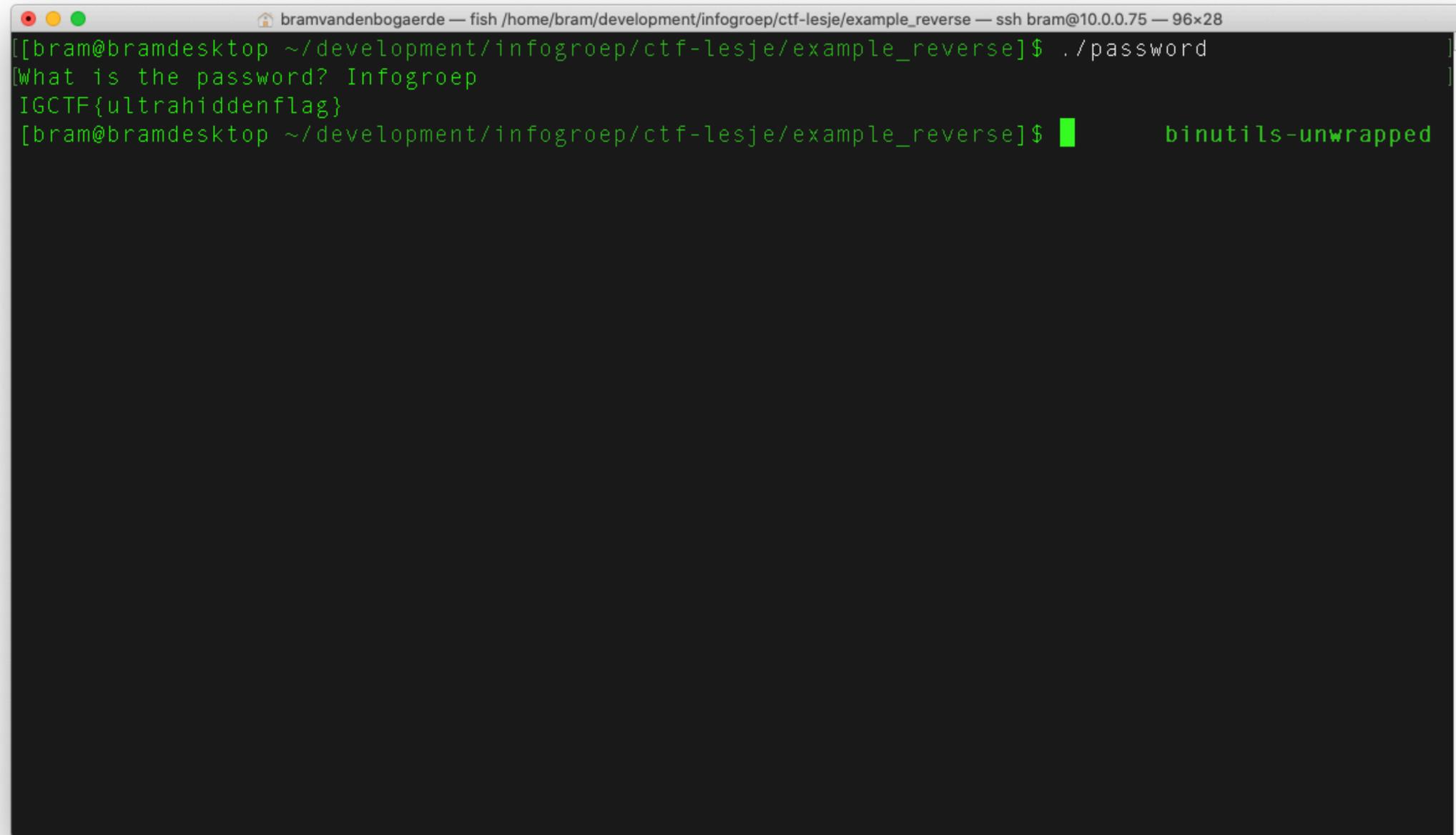
```
password = [ chr(a^b) for (a,b) in zip(password_bytes, key_bytes)]
```

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

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

Password is: Infogroep

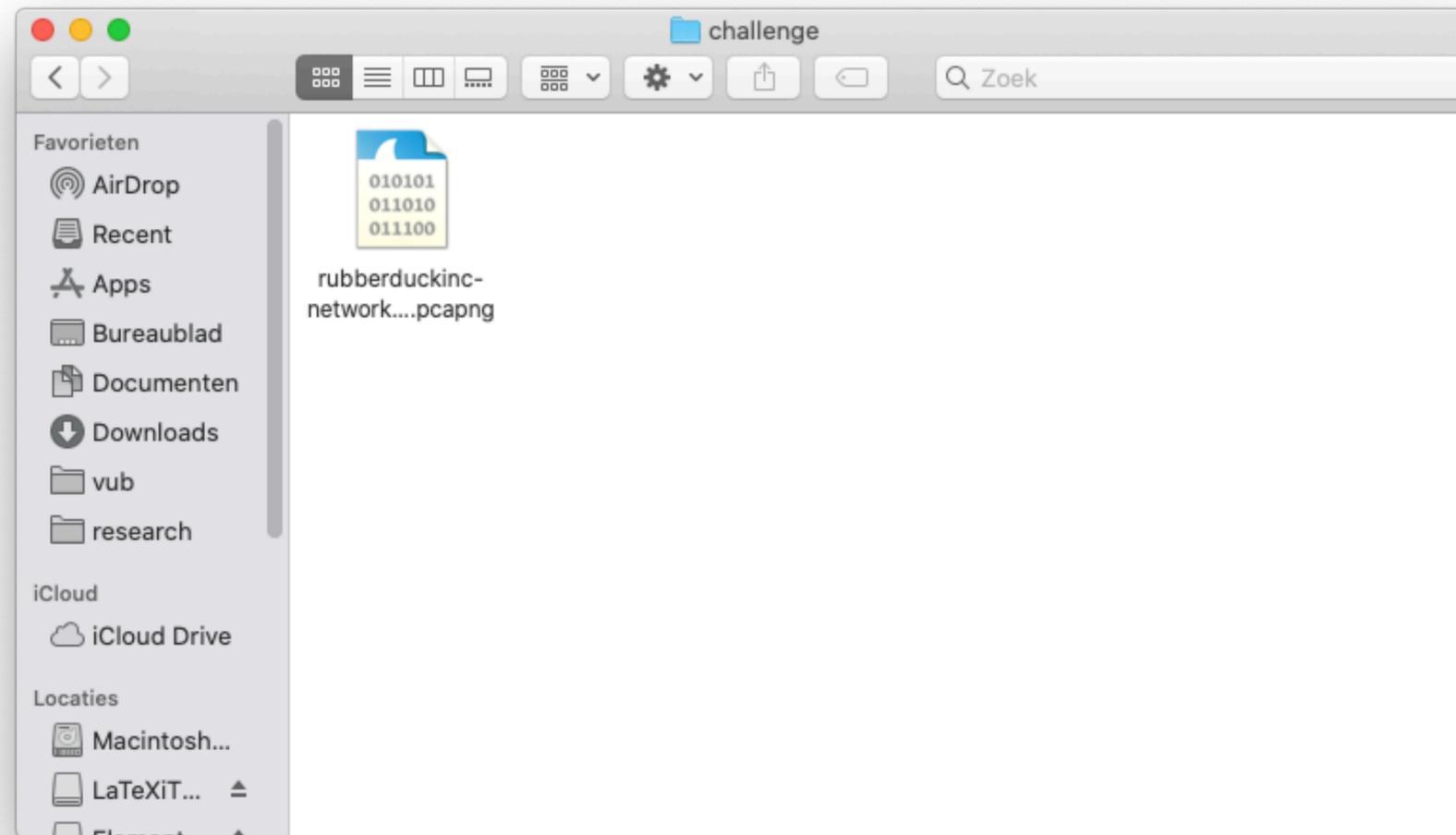
# Testing it out



```
bramvandenbogaerde — fish /home/bram/development/infogroep/ctf-lesje/example_reverse — ssh bram@10.0.0.75 — 96x28
[[bram@bramdesktop ~/development/infogroep/ctf-lesje/example_reverse]$ ./password
]
[What is the password? Infogroep
]
IGCTF{ultrahiddenflag}
[bram@bramdesktop ~/development/infogroep/ctf-lesje/example_reverse]$ binutils-unwrapped
```

# Network Forensics

# The Challenge



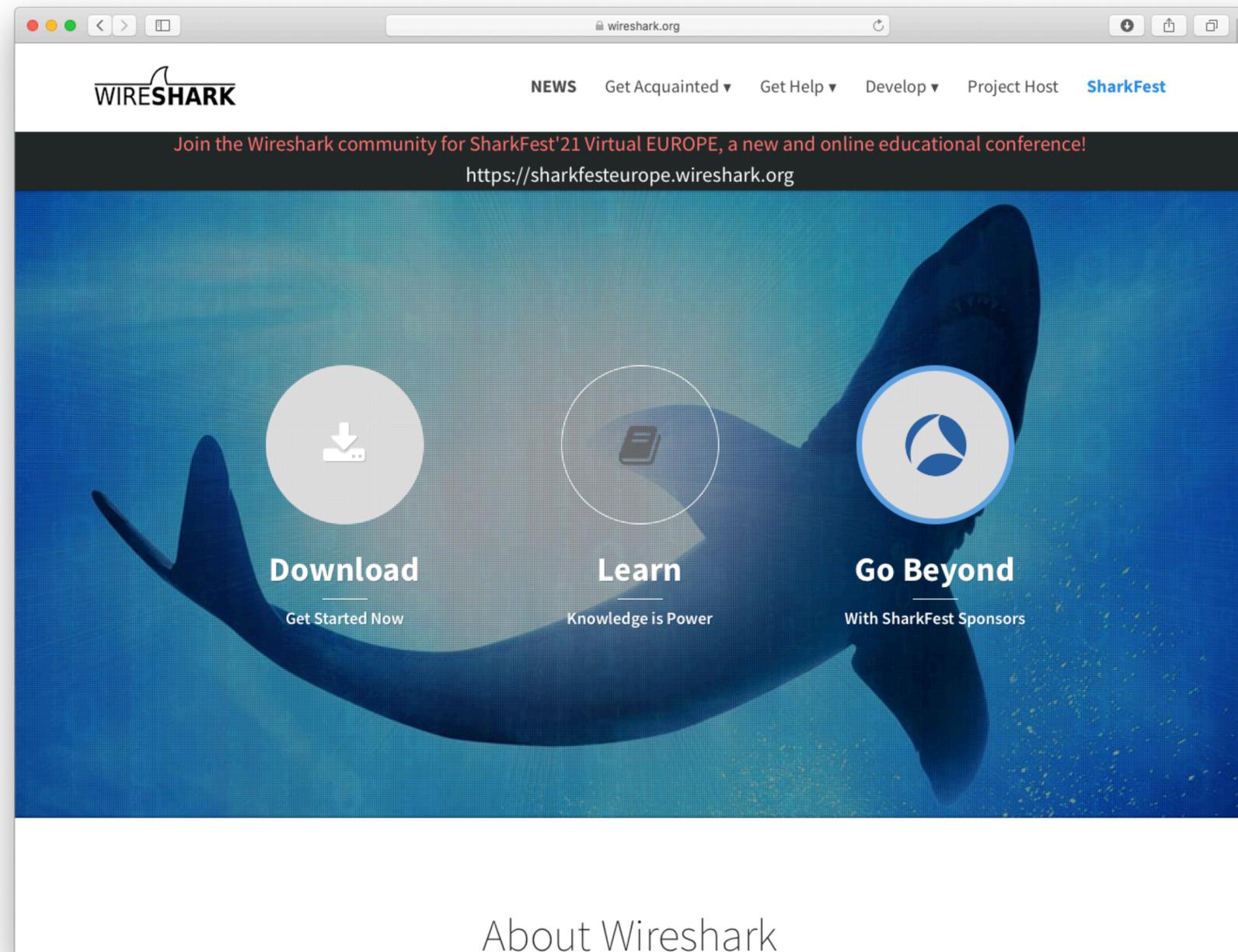
We only got a PCAP file, find information in it to get the flag

# 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

The screenshot displays the Wireshark interface with a network capture file named 'rubberduckinc-networklog.pcapng'. The interface is divided into several sections:

- Filtering, "search bar":** Located at the top, it shows the display filter field with the text 'Apply a display filter ... <%%/>'. A blue callout points to this area.
- Packet list:** A table listing captured packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. A blue callout points to this table.
- Packet analysis:** The middle section shows the detailed view of the selected packet (Frame 1). It includes information about the Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and SSH Protocol layers. A blue callout points to this section.
- Raw packet data:** The bottom section displays the raw packet data in hexadecimal and ASCII. A blue callout points to this section.

At the bottom of the window, the status bar indicates 'Packets: 598 · Displayed: 598 (100.0%)' and 'Profile: Default'.

Packet list

Filtering, "search bar"

Packet analysis

Raw packet data

# Making Sense of the Capture

The image shows the Wireshark network protocol analyzer interface. The main window displays a list of captured packets with columns for No., Time, Source, and Destination. A menu is open over the 'Statistics' tab, with 'Protocol Hierarchy' selected. A separate window titled 'Wireshark - Protocol Hierarchy Statistics - rubberduckinc-networklog.pcapng' is overlaid, showing a tree view of protocols and a corresponding table of statistics.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	598	100.0	114784	6115	0	0	0
Ethernet	100.0	598	7.3	8372	446	0	0	0
Internet Protocol Version 4	95.3	570	9.9	11400	607	0	0	0
User Datagram Protocol	21.1	126	0.9	1008	53	0	0	0
Simple Service Discovery Protocol	0.2	1	0.1	126	6	1	126	6
Data	20.9	125	1.9	2238	119	125	2238	119
Transmission Control Protocol	25.1	150	62.8	72040	3837	90	3748	199
SSH Protocol	8.7	52	51.9	59624	3176	52	59624	3176
Hypertext Transfer Protocol	1.3	8	6.6	7552	402	1	156	8
MIME Multipart Media Encapsulation	0.3	2	1.0	1148	61	2	1672	89
Line-based text data	0.5	3	0.1	76	4	3	76	4
JavaScript Object Notation	0.2	1	4.0	4578	243	1	4578	243
HTML Form URL Encoded	0.2	1	0.0	30	1	1	310	16
Internet Control Message Protocol	49.2	294	16.4	18816	1002	294	18816	1002
Address Resolution Protocol	4.7	28	0.7	784	41	28	784	41

Packet list details (Packet 2):  
No. 2 | Time 0.000047371 | Source 172.18.0.4 | Destination 172.18.0.2  
Encrypted packet (1...)  
2 [ACK] Seq=1 Ack=...

Annotations:  
- "http traffic can be inspected, as it is not encrypted" (points to HTTP protocols in the hierarchy)  
- "Lots of ssh traffic, but encrypted" (points to SSH Protocol in the hierarchy)

# Filtering

Filter for http traffic



rubberduckinc-networklog.pcapng

http

No.	Time	Source	Destination	Protocol	Length	Info
56	172.18.0.2	172.18.0.5	172.18.0.5	HTTP	222	GET /system/VICTIMID.php HTTP/1.1
44	172.18.0.5	172.18.0.2	172.18.0.2	HTTP/...	4842	HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
72	172.18.0.2	172.18.0.5	172.18.0.5	HTTP	96	POST /system/receive_coordinates.php?victim_id=299f13906703539efb32197ebc8c1e5d HTTP/1.1 (application/x-www-form-urlencoded)
28	172.18.0.5	172.18.0.2	172.18.0.2	HTTP	308	HTTP/1.1 200 OK (text/html)
1...	172.18.0.2	172.18.0.5	172.18.0.5	HTTP	755	POST /system/receive_network.php?victim_id=299f13906703539efb32197ebc8c1e5d HTTP/1.1 (text/plain)
7...	172.18.0.5	172.18.0.2	172.18.0.2	HTTP	289	HTTP/1.1 200 OK (text/html)
4...	172.18.0.2	172.18.0.5	172.18.0.5	HTTP	525	POST /system/process_portscan.php?victim=299f13906703539efb32197ebc8c1e5d HTTP/1.1 (text/plain)
528	131.8069334...	172.18.0.5	172.18.0.2	HTTP	239	HTTP/1.1 200 OK (text/html)

- ▶ Frame 69: 222 bytes on wire (1776 bits), 222 bytes captured (1776 bits) on interface br-86cc041ff151, id 0
- ▶ Ethernet II, Src: 02:42:ac:12:00:02 (02:42:ac:12:00:02), Dst: 02:42:ac:12:00:05 (02:42:ac:12:00:05)
- ▶ Internet Protocol Version 4, Src: 172.18.0.2, Dst: 172.18.0.5
- ▶ Transmission Control Protocol, Src Port: 33092, Dst Port: 80, Seq: 1, Ack: 1, Len: 156
- ▶ Hypertext Transfer Protocol

Seems to be communication with the attacker

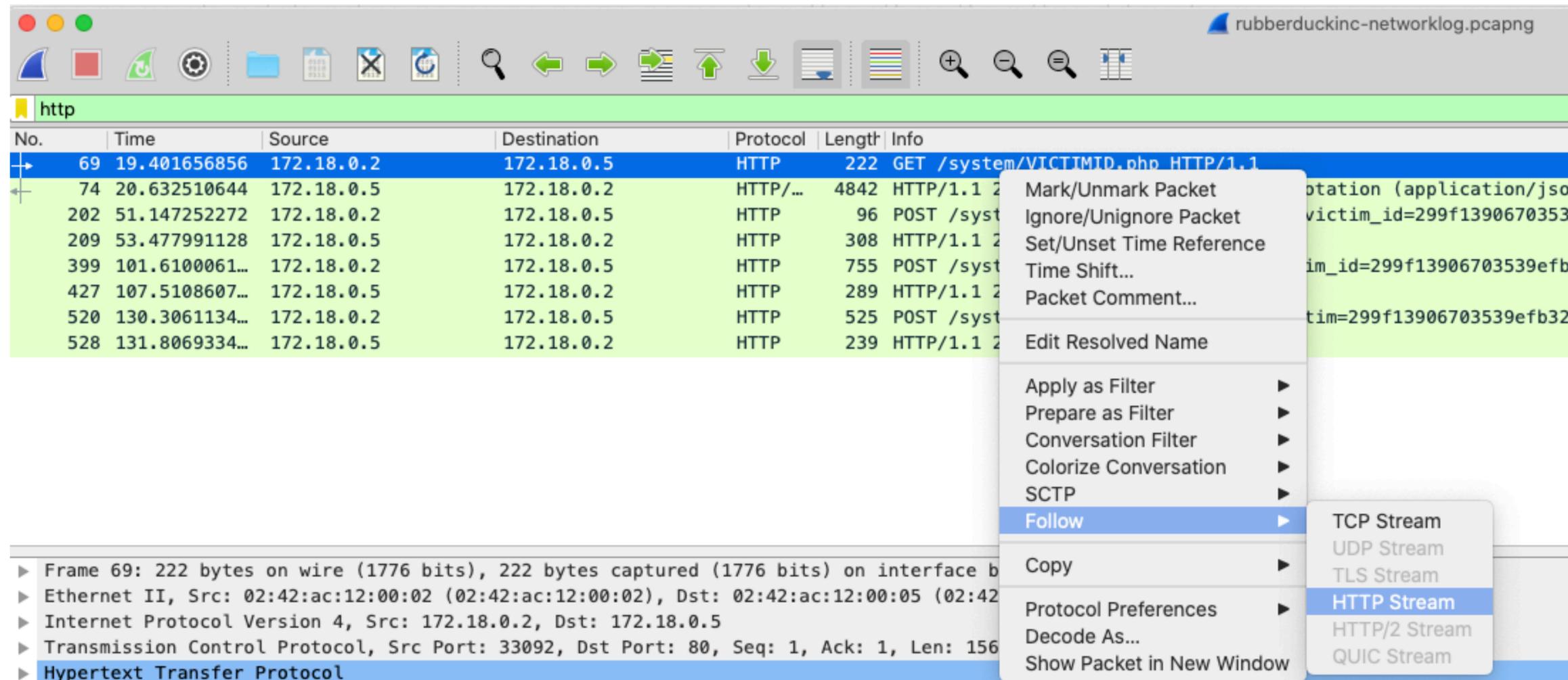
```
0000 02 42 ac 12 00 05 02 42 ac 12 00 02 08 00 45 00  ·B·····B·····E·
0010 00 d0 f2 56 40 00 40 06 ef a5 ac 12 00 02 ac 12  ···V@·@·······
0020 00 05 81 44 00 50 44 9c 60 e6 79 56 4c 71 80 18  ···D·PD·`·yVLq·
0030 01 f6 58 ee 00 00 01 01 08 0a 2b 70 73 ec 94 6b  ···X·····+ps·k
0040 5a 51 47 45 54 20 2f 73 79 73 74 65 6d 2f 56 49  ZQGET /s ystem/VI
0050 43 54 49 4d 49 44 2e 70 68 70 20 48 54 54 50 2f  CTIMID.p hp HTTP/
0060 31 2e 31 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a  1.1·Use r-Agent:
0070 20 43 6f 6f 72 64 69 6e 61 74 65 73 4d 61 73 74  Coordin atesMast
0080 65 72 2f 35 2e 31 2e 30 0d 0a 41 63 63 65 70 74  er/5.1.0 ··Accept
0090 3a 20 2a 2f 2a 0d 0a 41 63 63 65 70 74 2d 45 6e  : */*·A ccept-En
00a0 63 6f 64 69 6e 67 3a 20 69 64 65 6e 74 69 74 79  coding: identity
00b0 0d 0a 48 6f 73 74 3a 20 31 37 32 2e 31 38 2e 30  ··Host: 172.18.0
00c0 2e 35 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20  .5·Conn ection:
00d0 4b 65 65 70 2d 41 6c 69 76 65 0d 0a 0d 0a  Keep-Alive····
```

Hypertext Transfer Protocol: Protocol

Packets: 598 · Displayed: 8 (1.3%)

Profile: Defau

# Viewing an entire HTTP connection

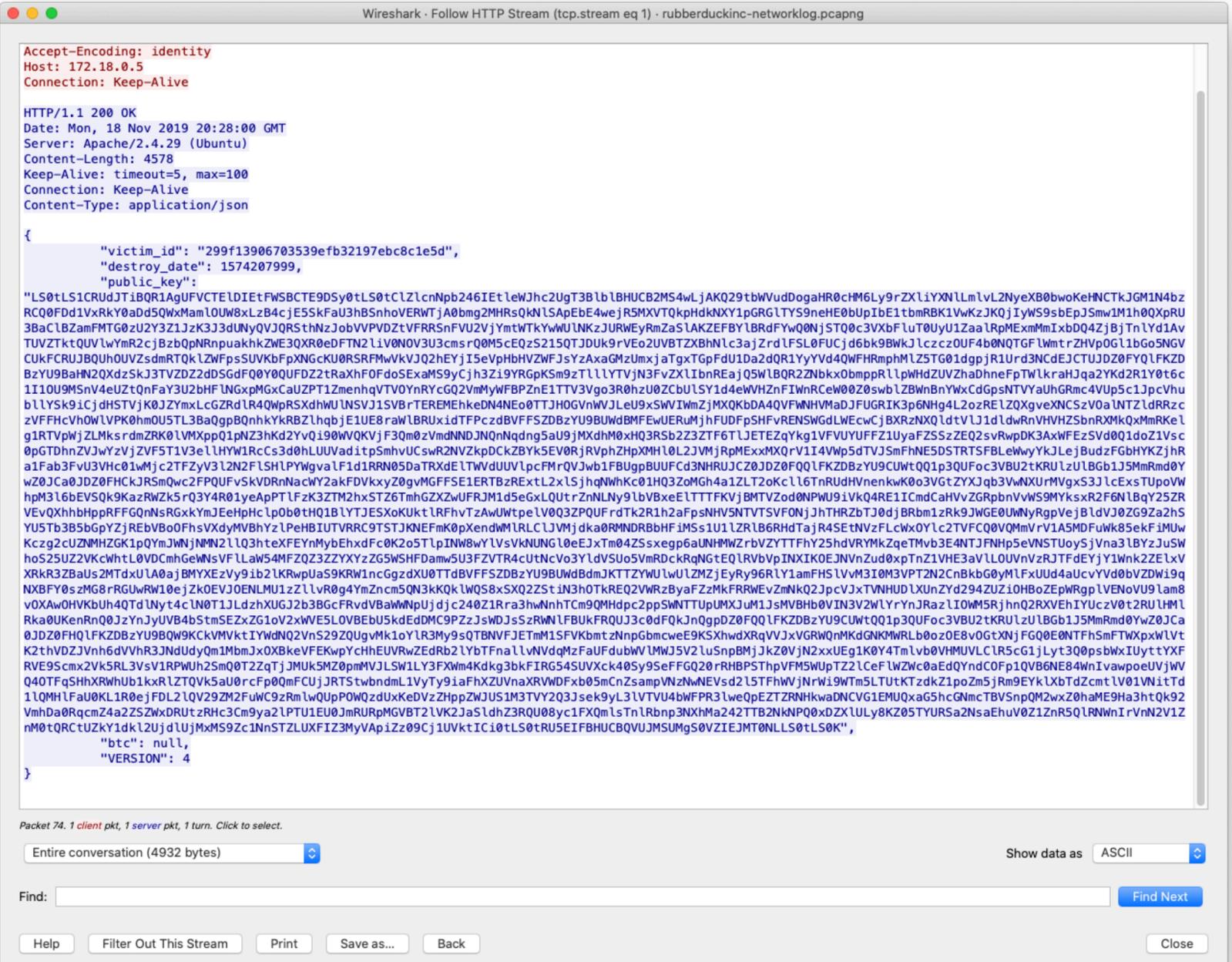


The screenshot shows the Wireshark interface with a packet capture named 'rubberduckinc-networklog.pcapng'. The packet list pane shows several HTTP packets. Packet 69 is selected, and the 'Follow' context menu is open, with 'HTTP Stream' selected. The packet details pane shows the structure of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
69	19.401656856	172.18.0.2	172.18.0.5	HTTP	222	GET /system/VICTIMID.php HTTP/1.1
74	20.632510644	172.18.0.5	172.18.0.2	HTTP/...	4842	HTTP/1.1 200 application/json
202	51.147252272	172.18.0.2	172.18.0.5	HTTP	96	POST /system/victim_id=299f13906703539...
209	53.477991128	172.18.0.5	172.18.0.2	HTTP	308	HTTP/1.1 200
399	101.6100061...	172.18.0.2	172.18.0.5	HTTP	755	POST /system/victim_id=299f13906703539efb3...
427	107.5108607...	172.18.0.5	172.18.0.2	HTTP	289	HTTP/1.1 200
520	130.3061134...	172.18.0.2	172.18.0.5	HTTP	525	POST /system/victim_id=299f13906703539efb321...
528	131.8069334...	172.18.0.5	172.18.0.2	HTTP	239	HTTP/1.1 200

▶ Frame 69: 222 bytes on wire (1776 bits), 222 bytes captured (1776 bits) on interface b  
▶ Ethernet II, Src: 02:42:ac:12:00:02 (02:42:ac:12:00:02), Dst: 02:42:ac:12:00:05 (02:42  
▶ Internet Protocol Version 4, Src: 172.18.0.2, Dst: 172.18.0.5  
▶ Transmission Control Protocol, Src Port: 33092, Dst Port: 80, Seq: 1, Ack: 1, Len: 156  
▶ Hypertext Transfer Protocol

# Viewing an entire HTTP connection



Wireshark - Follow HTTP Stream (tcp.stream eq 1) - rubberduckinc-networklog.pcapng

```
Accept-Encoding: identity
Host: 172.18.0.5
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Mon, 18 Nov 2019 20:28:00 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Length: 4578
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json

{
  "victim_id": "299f13906703539efb32197ebc8c1e5d",
  "destroy_date": 1574207999,
  "public_key":
"LS0tLS1CRUdJTTIBQR1AgUFVCTEldIEtFWSBCTE9DSy0tLS0tLzL1cnNpb246IEtLeWJhc2UgT3BlbHhUCB2MS4wLjAKQ29tbWVudDogaHR0cHM6Ly9rZXliYXNlLmVlL2NyeXB0bwoKeHNCTkJKM1N4bz
RCQ0FDd1VxRkY0aDd5QWxMam1OUW8xLzB4cjlE5SkFaU3hBSnhoVERWTJA0bmg2MHRsQkNLSApEbE4wejlR5MXVTQkpHdXNXY1pGRGLTY59neHE0bUpIbE1tbnRBK1VwKzJKQjIyWS9sbEpJSmw1M1h0QXpRU
3BaCLBZamFMTG0zU2Y3Z1JzK3J3dUNyQVJQRStHnzJobVVPVZztVFRSFnFVU2VjYmTmYkYwUWLNKzJURWEyRmZaSLAKZEFBYlBRdFYwQ0NjSTQ0c3VXbFluT0UyU1ZaaLRpMEXmMmIxbDQ4ZjBjTnLyd1Av
TUVZTktQUVlwYmR2cjBzbQpNRnpuakhkZWE3QXR0eDFTN2LiV0NOV3U3cmsrQ0M5cEQzS215Q2JD0k9rVEo2UVBTZXBNlC3ajZrdlFSL0FUCj d6bk9BwKJlczcz0UF4b0NQTGF lWmtrZHVp0G11bGo5NGV
CukFCRUJBQh0UUVzsdmRTQkLzWFpsSUVKbFpXNGcKU0RSRfMwKvJQ2hEYjI5eVpHbHVZWFJySjAxaGMzUmxjaTgxTGPdU1Da2dQR1YyYVd4QWFHRmPhMLZ5TG01dgpjR1Urd3NCdEJCTUJDZ0FYQlFKZD
BzYU9BaHN2QXdzSkJ3TVZDZ2d5GdFQ0Y0UFDZ2tRaXhF0FdoSExaM59yCjh3i2i9YRgPksm9zTlLLYTVjN3FvZXLIbnREajQ5WLBQR2ZNbkx0bmpRllpWHDZUVZhaDhneFpTWlkrAHJqa2YKd2R1Y0t6c
1I10U9MSnV4eUZtQnFaY3U2bHF LNgxpMGxCaUZPT1ZmenhqVTVOYnRycGQ2VmMyWFBPZnE1TTV3Vgo3R0hzU0ZCbUlsY1d4eWVHZnFiWnRCeW0Z0swbLZBWNbnYWxCdGpsNTVyaUHGRCmC4VUp5c1JpcVhu
bllYsk9iCjdHSTVjK0JZYmxLcGZRdLR4QWpRSXdhWUlnSVJ1SVBRTEREMEhkeDN4NE0TTJHOGVnWVJLEu9xSWVlWmZjMXQkBA4QVFNHVMaDjFUGRIK3p6NHg4L2ozRELZQXgveXNCSzV0a lNTZldRRzc
zVFFhCvH0wLVPK0hm0U5TL3BaQgpBQnhkYkRBZlhqbjE1UE8rawLBRUxidTFPczdBFVFSZDBzYU9BUwdbMFUEWUerUmjHfUDFpSHFVRENSWgdLWecwCjBXRzNXQldtVlJldldwRnVHVHZSbnRXMKQxMmRkeL
g1RTVpWjZLMksrdmZRK0lVMXppQ1pNZ3hKd2YvQ190WVQKvjF3Qm0zVmdNNDJNqNqndng5aU9jMXdhM0xHQ3RSb2Z3ZTF6TLJETEzQYkg1VFVUYUZZ1UyaFZSSzZEQ2svRwpDK3AxWFEzSvd0Q1doZ1Vs c
0pGTDhnZVjwYzVjZVF5T1V3eLlHYW1RcCs3d0hLUUVaditpSmhvUCsRw2NVZkPcDKZBYk5EV0RjRVphZHPXMH0L2JVMjRPMEXmMXQrV1I4VWp5dTVJSmFhNE5DSTRTSFBLewWyYkLjLejBudzFGbHYKZjHR
a1Fab3FvU3VHc01wMj c2TFZyV3L2N2FLSHLPYwgalF1d1RRN05DaTRXDElTWvDUUVlPcFMrQVjwb1FBUpBUUFcd3NHRUJCZ0JDZ0FQlFKZDBzYU9CUWtQQ1p3QUFoc3VBU2tKRULzULBGB1J5MmRmd0Y
wZ0JCa0JDZ0FHCKJRSmQwc2FPQUFvSvKVDNRnNacwY2akFDVxkyZ0gvMGFFSE1ERTBzRExtLzLx5jhqNWhKc01HQ3ZomGh4a1ZL2ZokClL6TnRUdHVnNenkwK03VgtZYXjqb3VwNXUrmVgX53JlcExsTUpoVW
hpM3l6bEV5Qk9KazRWZk5rQ3Y4R01yeApPTLFzK3ZTM2hxSTZ6TmhGXZwUFRJm1d5eGxLQutRznNLNj9lVbVXeELTTTKVjBMTVZod0NPWu9iVkQ4RE1ICmdCaHvVZGRpbnVwS9MYksXR2F6NlBqY25ZR
EvEQXhhbHppRFFGQnNsRGxkYmJEEHpHclp0b0tHQ1BLYTJESXoKUKtLRfHvTzAwUwtpelV0Q3ZPQUFrdTk2R1h2aFpsNHV5NTVSVFONjJhTHRZbTJ0djBRbm1zRk9JWGE0UWNYRgpVejbldVJ0ZG9Za2hS
YU5Tb3B5bGpYZjRebVBo0FhsVXdyMVBhYzlpEhBIUTVRRC9TSTJKNEFMk0pXendWmLRCLJVMj dka0RMNDRbBHFIM5s1U1LZRlB6RHDTajR4SEtNVzFLcWx0YlC2TVFCQ0VQmMvVr1A5MDfUwK85eKfIMUw
Kczg2cuZNMHZGK1pQYmJWmJNMN2LlQ3hteXFEYnMybEhxdFc0K2o5TlpInW8wYlVsVKNUNGl0eEjXtm04ZSxegg6aUNHMWzrbvZYTTFhY25hdVRYMkZqeTMvb3E4NTJFNHP5eVNSTUoysjVna3lBYzJuSw
hoS25U2ZVKcwhTL0VDcmhGwNsVFLaW54MFZQZ3ZYXyzZG5WShFDamw5U3FZVTR4cUtNcVo3YldVSUo5VmRdckRqNGtEQlRvVpINXIKOEJNVnZud0xpTnZ1VHE3aVlL0UvNvzRJTfdeYjY1Wnk2ZE lXv
XRkR3ZBaUs2MTdxUa0ajBMYXezV9ib2lKRwpUa59KRW1ncGgzdXU0TtdBVFfSZDBzYU9BUwdbMjKTTZYUwLwLZMzjEyRy96RlY1amFHSlvvM3I0M3VPT2N2CnBkbG0yMlFXUud4aUcvYVd0bVZDWi9q
NXBFY0szMG8rGUwRW10ejZk0EVJ0ENLMU1zZlLvR0g4YmZncm5QN3kKQkLWQ58xSXQ2ZStiN3h0TKREQ2VWRzByaFZmKFRREvZmNkQ2JpCvJxTVNHUdLXUnZYd294ZUZ10HBoZepwRgpLVeNOvU9lam8
v0XAwOHVkbU4hQtdlNyt4c lN0T1JLdzhXUGJ2b3BGcFRvdVBawWnpUjdj c240Z1Rra3hwnhTcm9QMhdpc2ppSWNTTUpUMXJum1JsvMBhb0VIN3V2WlYrYnJRazlIOWM5RjhnQ2RXVEhIYUczV0t2RUlHMl
Rka0UKenRnQ0JzYnJyUVB4bStmSEZxZG1oV2xWVE5L0VBEBU5kdEdDM9PZzJsdWJzSsZRNWlFBUKFRUJ3c0dfQkJnQgpdZ0FQlFKZDBzYU9CUWtQQ1p3QUFoc3VBU2tKRULzULBGB1J5MmRmd0YwZ0JCa
0JDZ0FHQlFKZDBzYU9BQW9KCKVMVktIYwNq2VnS29ZQUGvMk1oYlR3My9sQTBNVFJETm1SFVKbmtzNnpGbmCweE9K5XhwdXrQVjVGRWQnMKdGNKMRLb0z0E8v0GtXnjFGQ0E0NTFhSmFTWpxwLVt
K2thVDZJVnh6dVVR3JNdUdyQm1MbmJx0XBkeVFEKwYcHhEUVRwZEdRb2lYbTfna lLVVdqMzFaUfduBwLmWJ5V2LUnSpBMjJkZ0VjN2xxUEg1K0Y4Tmlvb0VHMUVLCR5cG1jLyt3Q0psbWxIUytYXF
RVE9S cmx2VksRL3vsV1RPWUh2SmQ0T2ZqTjJMUk5M20pmMVJLSW1LY3FXWm4Kdkg3bkFIRG54SUVXck40S9SeFFGQ20rRHPBPSThpVFM5WUpTZ2LceFlwZw0aEdQYndC0Fp1QVB6NE84WnIvawpoeUvjwV
Q40TFqSHhXRWhUb1kxRLZTQVksaU0rcFp0QmFCUjJRTStwbnmL1VyTy9iaFhXZUVnaXRvWDFxb05mCnZsampVNzNwNEVsd2L5TFhWvjnrWi9WTm5LUTktZdkZ1poZm5jRm9EYkLXbTdzcmTlV01VnitD
1LQMhlfAu0KL1R0ejFDL2lQV29ZM2FuWC9zRmLwQUpp0WQzdUxKeDvzZHpPzWJUSIM3TVY2Q3Jsek9yL3lVTVU4WfPR3lweQpEzTRNHkwaDNCVGIEMUQxaG5hcGNmCTBVSnPM2wxZ0haME9Ha3htQk92
VmhDa0RqcmZ4a2Z5ZwXDRUtZRHc3Cm9ya2LPTU1EU0JmRURpMGVB2LVK2Ja5ldhZ3RQU08yc1FXQm lSrnLrbnp3NXhMa242T2B2NKP00xDZXLULy8KZ05TYURS2aNsAehuV0Z1ZnR5Q1RNWnIrvN2V1Z
nM0tQRctUZkY1dkl2UjdUjMxM59Zc1NnSTZLUXFIz3MyVApiZz09Cj1UVktICl0tLS0tRUSElFBHUCBQVUJMSUmG50VZIEJMT0NLLS0tLS0K",
  "btc": null,
  "VERSION": 4
}
```

Packet 74. 1 client pkt, 1 server pkt, 1 turn. Click to select.

Entire conversation (4932 bytes) Show data as ASCII

Find: Find Next

Help Filter Out This Stream Print Save as... Back Close

# Victim Coordinates

Wireshark · Follow HTTP Stream (tcp.stream eq 2) · rubberduckinc-networklog.pcapng

```
POST /system/receive_coordinates.php?victim_id=299f13906703539efb32197ebc8c1e5d HTTP/1.1
User-Agent: CoordinatesMaster/5.1.0
Accept: */*
Accept-Encoding: identity
Host: 172.18.0.5
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 30

lat=50.8214167&lon=004.3955833HTTP/1.1 200 OK
Date: Mon, 18 Nov 2019 20:28:32 GMT
Server: Apache/2.4.29 (Ubuntu)
X-Received: yes
Content-Length: 21
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

Coordinates received!
```

1 client pkt, 1 server pkt, 1 turn.

Entire conversation (552 bytes) Show data as ASCII

Find:  Find Next

Help Filter Out This Stream Print Save as... Back Close

# Port Scan Results

Wireshark · Follow HTTP Stream (tcp.stream eq 4) · rubberduckinc-networklog.pcapng

```
POST /system/process_portscan.php?victim=299f13906703539efb32197ebc8c1e5d HTTP/1.1
Host: 172.18.0.5
User-Agent: CoordinatesMaster/5.1.0
Accept: */*
Content-Length: 459
Content-Type: multipart/form-data; boundary=-----05afca8d8006358e

-----05afca8d8006358e
Content-Disposition: form-data; name="payload"; filename="export.txt"
Content-Type: text/plain

Generated by Angry IP Scanner 3.6.1
https://angryip.org

Scanned 192.168.0.123 - 192.168.0.123
Nov 18, 2019, 5:02:19 PM
```

IP	Ping	Hostname	Ports
192.168.0.123	0..ms	[n/a]	80

```
-----05afca8d8006358e--
HTTP/1.1 200 OK
Date: Mon, 18 Nov 2019 20:29:51 GMT
Server: Apache/2.4.29 (Ubuntu)
X-Received-Bytes: 268
Content-Length: 3
Content-Type: text/html; charset=UTF-8

OK
```

Packet 520. 1 client pkt, 1 server pkt, 1 turn. Click to select.

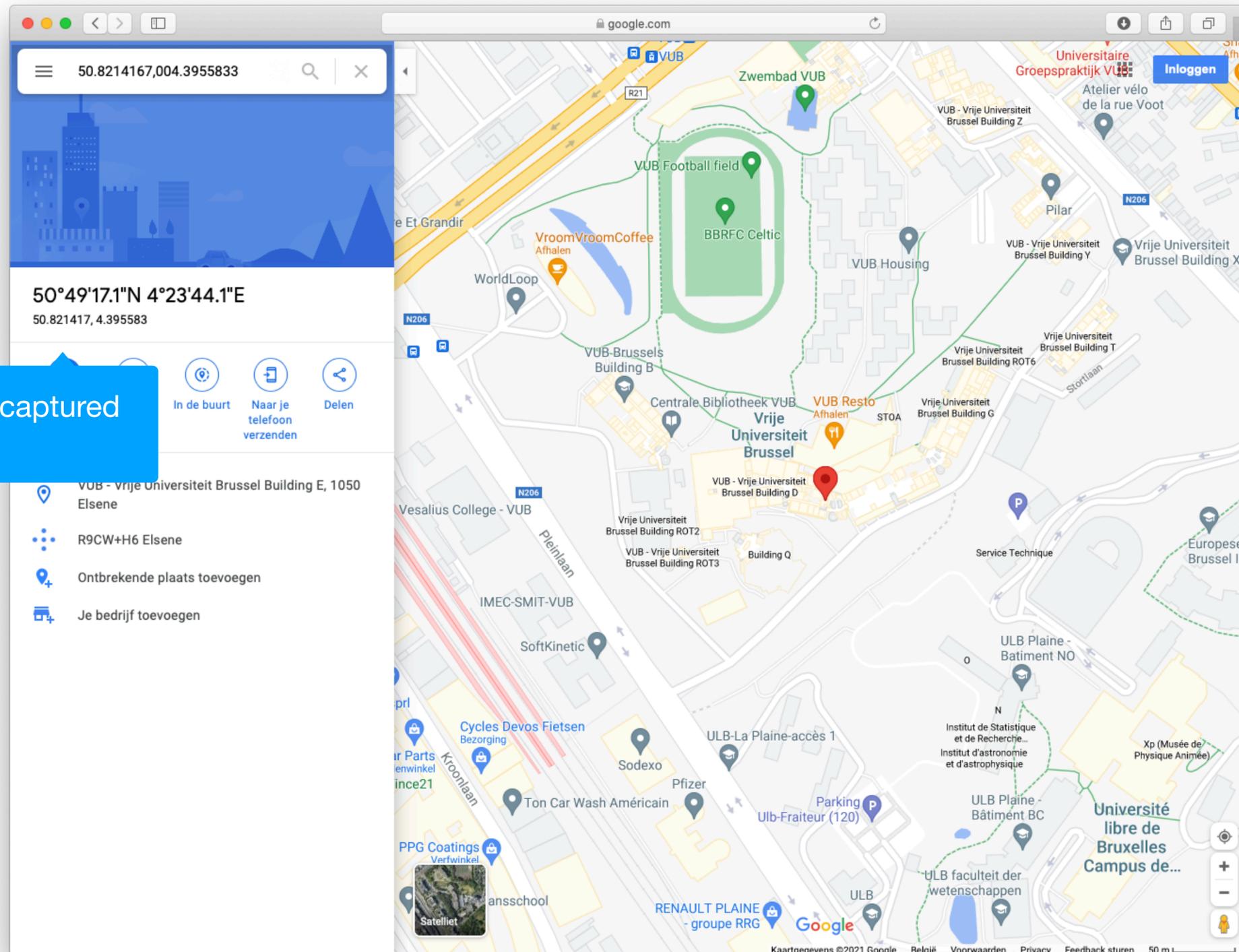
Entire conversation (893 bytes) Show data as ASCII

Find:  Find Next

Help Filter Out This Stream Print Save as... Back Close

Victims ip address is  
192.168.0.123

# Where is the victim?



Coordinates from captured request

# Final Part



Open WiFi network named  
“RubberDuckInvestmentInc” at these coordinates



Surf to <http://192.168.0.123> to get the flag