

Static and Dynamic Analysis

MST - 2/2

This will be you after today



Why reverse engineer?

- Develop competing products?
- Achieve interoperability
- Analyze malware
- Crack digital rights management
- Locate vulnerabilities

Static vs Dynamic Analysis

- **Static analysis** is the examination of program code without running it
 - Reading assembly, analyzing code patterns
 - Can be difficult for a human or computationally expensive for a computer
- **Dynamic analysis** is the examination of program state as it is running
 - Also known as debugging
 - Involves pausing a program at time $t=x$ and examining the system state
 - Code paths that aren't run won't be easily examined

Static analysis

Static analysis tools

- **xxd**
 - Hex editor
- **strings**
 - View ASCII data in a file
- **objdump**
 - Basic disassembler
- **IDA**
 - Industry standard disassembler

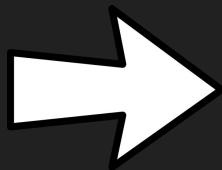
xxd

- Hex editors provide us with the raw bytes in a given file
- No interpretation of file data

```
$ cat hello.txt
Hello MST! Arbitrary data here
$ xxd hello.txt
00000000: 4865 6c6c 6f20 4d53  Hello MS
00000008: 5421 2041 7262 6974  T! Arbit
00000010: 7261 7279 2064 6174  rary dat
00000018: 6120 6865 7265 0a    a here.
```

xxd

```
.data
message:
.string "Hello World!"
.text
.globl main
main:
mov ecx, offset message
push ecx
call puts
pop ecx
mov eax, 0
ret
```

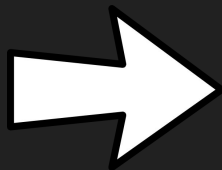


```
$ xxd hello
00000000: 7f45 4c46 0101 0100  .ELF....
00000008: 0000 0000 0000 0000  ....
00000010: 0200 0300 0100 0000  ....
00000018: 2083 0408 3400 0000  ...4...
. . . . .
000041d: b920 a004 0851 e8c8  . ...Q..
0000425: feff ff59 b800 0000  ...Y....
000042d: 00c3 9055 5731 ff56  ...UW1.V
. . . . .
```

Which part is the program code?

xxd

```
.data
message:
.string "Hello World!"
.text
.globl main
main:
mov ecx, offset message
push ecx
call puts
pop ecx
mov eax, 0
ret
```



```
$ xxd hello
00000000: 7f45 4c46 0101 0100  .ELF....
00000008: 0000 0000 0000 0000  .....
00000010: 0200 0300 0100 0000  .....
00000018: 2083 0408 3400 0000  ...4...
. . . . .
000041d: b920 a004 0851 e8c8  . ...Q..
0000425: feff ff59 b800 0000  ...Y....
000042d: 00c3 9055 5731 ff56  ...UW1.V
. . . . .
```

Which part is the program code?

strings

- Prints out any ASCII strings from the file that are 4+ characters in length
- Useful for getting an idea of what data your file contains

```
$ strings hello
/lib/ld-linux.so.2
libc.so.6
_IO_stdin_used
puts
__libc_start_main
__gmon_start__
GLIBC_2.0
PTRh
[^_]
;*2$"8
Hello World!
GCC: (Ubuntu 4.8.4-2ubuntu1~14.04.3)
4.8.4
. . . . .
```

objdump

- Displays tons of useful information about an executable file
- Can display code for each function in the file

```
$ objdump -d hello
```

```
0804841d <main>:
```

804841d:	b9 20 a0 04 08	mov	\$0x804a020,%ecx
8048422:	51	push	%ecx
8048423:	e8 c8 fe ff ff	call	80482f0 <puts@plt>
8048428:	59	pop	%ecx
8048429:	b8 00 00 00 00	mov	\$0x0,%eax
804842e:	c3	ret	

IDA (Interactive Disassembler)

- Gold standard disassembler that everyone uses
- Graphical display, runs on Windows/OSX/Linux (but mainly Windows)
- Provides all the features in tools we've seen, and more
- Extensive code analysis features
- Very expensive

IDA (Interactive Disassembler)

The screenshot displays the IDA Pro disassembler interface for the file `C:\Documents and Settings\Administrator\My Documents\Downloads\StartClean\StartClean.idb (StartClean.exe)`. The main window shows assembly code with a control flow graph (CFG) overlaid. The CFG consists of several basic blocks connected by control flow edges. The assembly code includes instructions such as `cmp eax, 1`, `jz short loc_402794`, `mov esi, [esp+4+hDlg]`, `push esi`, `call sub_401150`, `add esp, 4`, `test eax, eax`, `jz short loc_4027C1`, `mov duword_40274C, 1`, `push 1`, `push esi`, `call ds:EndDialog`, `mov eax, 1`, `pop esi`, `retn 10h`, `loc_4027C1: push 0`, `push 0`, `push offset aIncorrectCode`, `push esi`, `call ds:MessageBoxA`, `pop esi`, `mov eax, 1`, `pop esi`, `retn 10h`, `loc_4027DA: mov esi, [esp+4+hDlg]`, `push 1`, `push esi`, `call ds:EndDialog`, `mov eax, 1`, `pop esi`, `retn 10h`, `sub_402770 endp`, `loc_40278E: mov eax, eax`, `push esi`, `retn 10h`.

The left sidebar shows the **Functions window** with a list of functions including `_parse_cmdline`, `_setSBCS`, `_strncat`, `_strncpy`, `_x_ismbbtype`, `_x_cptlookup`, `start`, `sub_401000`, `sub_401150`, `sub_401280`, and `sub_4013C0`. The **Graph overview** window shows a high-level view of the CFG. The **Strings window** at the bottom lists various strings found in the program, including `(SP)(a)b`, `700WPa`, `b*****`, `pproco(b)a\b`, `Code`, `Name`, `Software\Start Clean\Configuration`, and `Program`.

The status bar at the bottom indicates the address `AD: 1d1e` and the disk location `Disk: 11GB`.

IDA - Z:\Security and CTF\NSA CODEBREAKER\server.idb (server)

File Edit Jump Search View Debugger Options Windows Help

Library function Data Regular function Unexplored Instruction External symbol

Functions window

Function name

- driver_ioctl
- listen
- strchr
- fputc
- memset_chk
- qsort
- socket
- fprintf_chk
- sigaction
- strcmp
- tcgetattr
- abort
- gethostbyname
- shutdown
- strtol
- fputs
- connect
- recv
- close
- __ctype_tolower_loc
- __assert_fail
- __ctype_b_loc
- send
- calloc
- main
- start
- sub_80499A0
- sub_80499B0
- sub_8049A20
- sub_8049A40
- sub_8049A70
- sub_8049AC0
- server_enc_and_send
- process_client_connection
- sub_804A5E0

Line 86 of 3238

Graph overview

Output window

Caching 'Strings window'... ok

Python

AU: idle Down Disk: 2091GB

100.00% (4283,1084) (792,80) 000015A6 080495A6: main+26 (Synchronized with Hex View-1)

loc_80495FC:

```
mov eax, ds:optarg
mov [ebp+name], eax
jmp short loc_80495AD
```

loc_80495AD:

```
push edi
push offset aUsingKeyS ; "using key %s\n"
push 1 ; DWORD
push ds:stdout ; DWORD
call __fprintf_chk
call sub_804A7D0
pop eax
pop ecx
push dword_81A91B0 ; int
call offset aBeginRsaPrivat ; "-----BEGIN RSA PRIVATE KEY-----\nMIIEpA"...
push 0 ; char *
push 0 ; int
mov ebx, eax
push 0 ; int
push eax ; int
call PEM_read_bio_PrivateKey
add esp, 20h
test eax, eax
mov [ebp+var_64], eax
jz loc_8049840
```

loc_804961B:

```
test edi, edi
jz loc_8049970
```

loc_8049970:

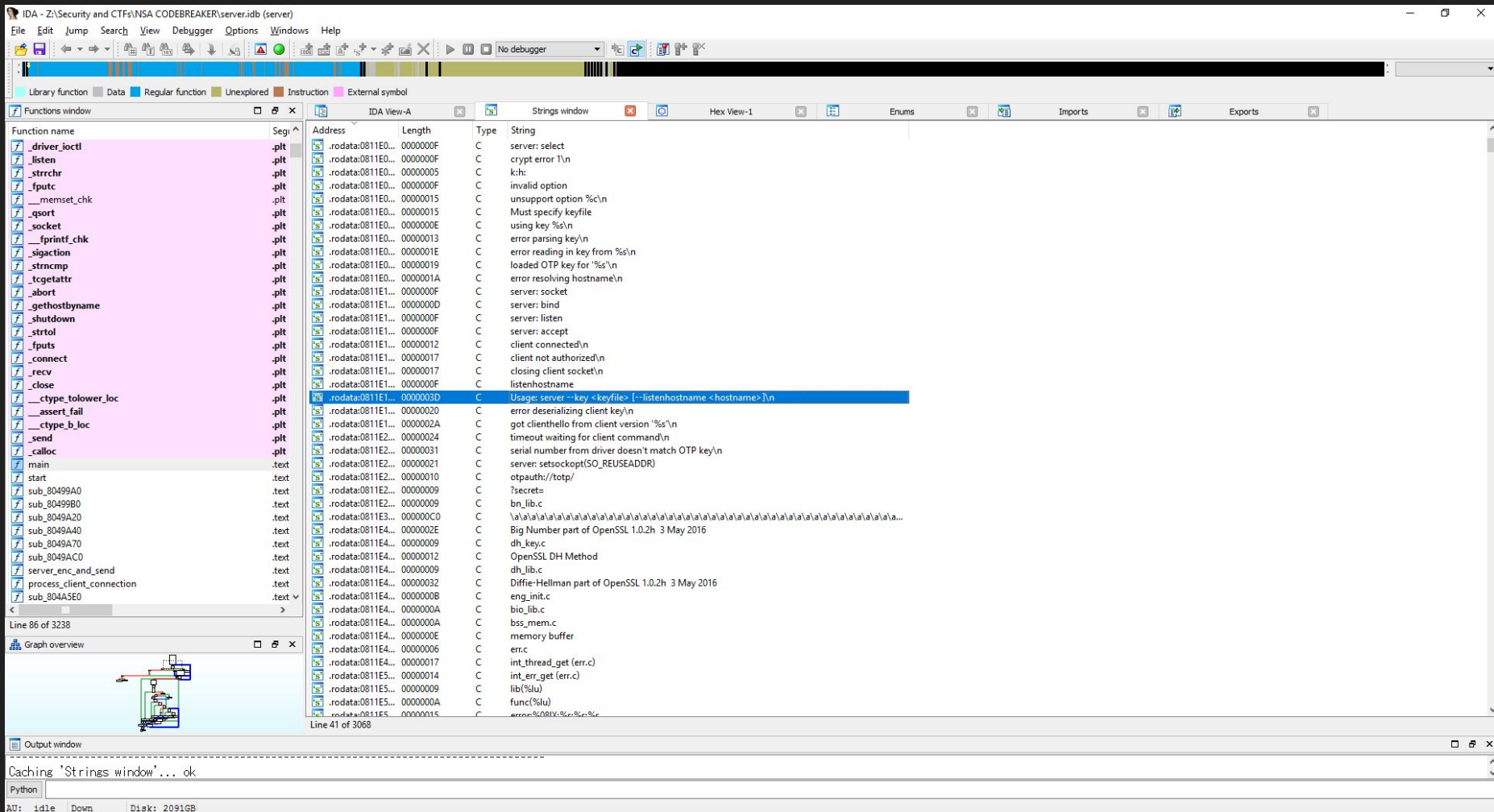
```
sub esp, 0Ch
push offset aMustSpecifyKey ; "Must specify keyfi
call sub_8049A70
with err: sc-analysis failed
```

loc_8049613:

```
mov edi, ds:optarg
jmp short loc_80495AD
```

loc_8049613:

```
sub esp, 0Ch
push ebx ; ptr
call BIO_vfree
pop ecx
pop ebx
push [ebp+var_64] ; int
push edi ; filename
call sub_8049AC0
add esp, 10h
```



IDA - Z:\Security and CTF\NSA CODEBREAKER\server.idb (server)

File Edit Jump Search View Debugger Options Windows Help

Library function Data Regular function Unexplored Instruction External symbol

Functions window

Function name

- driver_ioctl
- listen
- strchr
- fputc
- __memset_chk
- qsort
- socket
- __fprintf_chk
- sigaction
- strncmp
- tcgetattr
- abort
- gethostbyname
- shutdown
- strtol
- fputs
- connect
- recv
- close
- __ctype_tolower_loc
- __assert_fail
- __ctype_b_loc
- send
- calloc
- main
- start
- sub_80499A0
- sub_80499B0
- sub_8049A20
- sub_8049A40
- sub_8049A70
- sub_8049AC0
- server_enc_and_send
- process_client_connection
- sub_804A5E0

Seg

37 v5 = a2;

38 v30 = *MK_FP(_GS_, 20);

39 while (1)

40 {

41 v6 = getopt_long(v4, v5, "k:h:", &longopts, 0);

42 if (v6 == -1)

43 break;

44 if (v6 != 104)

45 {

46 if (v6 != 107)

47 {

48 if (v6 != 63)

49 {

50 __fprintf_chk(stdout, 1, "unsupport option %c\n");

51 exit(1);

52 }

53 sub_8049A70("invalid option");

54 }

55 v3 = (char *)optarg;

56 }

57 }

58 if (!v3)

59 LABEL_45:

60 sub_8049A70("Must specify keyfile");

61 __fprintf_chk(stdout, 1, "using key %s\n");

62 sub_804A7D0();

63 v7 = (void *)BIO_new_mem_buf(aBeginRsaPrivat, dword_81A91B0);

64 v19 = (void *)PEM_read_bio_PrivateKey((int)v7, 0, 0, 0);

65 if (!v19)

66 {

67 __fprintf_chk(stderr, 1, "error parsing key\n");

68 goto LABEL_32;

69 }

70 BIO_vfree(v7);

71 ptr = sub_8049AC0(v3, (int)v19);

72 if (!ptr)

73 {

74 __fprintf_chk(stderr, 1, "error reading in key from %s\n");

75 goto LABEL_36;

76 }

77 v9 = 0;

78 __printf_chk(1, "loaded OTP key for '%s'\n", ptr[2], v8);

000018DD main:74

Line 86 of 3238

Graph overview

Output window

804A7E0: using guessed type int __cdecl sub_804A7E0(_DWORD, _DWORD);

81AA9C0: using guessed type int optarg;

Python

AU: idle Down Disk: 2091GB

What's important about this

- objdump & IDA will be very useful for this week's homework
- Learning IDA is essential if you're interested in the field
 - [IDA Pro Book](#)
 - [IDA Free version](#)

Dynamic analysis

Dynamic Analysis (debugging)

- Analyze a program during runtime to see system and memory state
- Step through code instructions to follow what happens

ltrace

strace

`gdb`

Homework

Homework

- Crack our program!
- You must find the correct password that the program accepts
- Submit both the correct password and an explanation of how you found it
- You can use any tool we've talked about, or similar tools, to help solve it

Tips

- gdb is useful for stepping through the program and following execution
- IDA can show you an overarching program structure that is easier to follow than straight assembly
- The binary is x64, so parameters are passed in RDI, RSI, RDX, RCX, R8, R9

```
$ ./crackme
```

```
What is the secret password?
```

```
testing123
```

```
Nope, that is not correct. Sorry!
```

```
$ ./crackme
```

```
What is the password?
```

```
(HIDDEN)
```

```
Well done! You found the secret password!
```


Homework grading

- Submit your assembly code file to cm7bv@virginia.edu with the subject “MST Assignment 2 - <YOUR_UVA_ID>”
 - eg: “MST Assignment 2 - cm7bv”
- Also, include a brief (1-paragraph) description of what you did and how it went

Useful Resources

- [IDA Free version](https://www.hex-rays.com/products/ida/support/download_freeware.shtml)
(https://www.hex-rays.com/products/ida/support/download_freeware.shtml)
- [IDA Pro book](http://staff.ustc.edu.cn/~sycheng/ssat/books/The.IDA.Pro.Book.2ed.pdf)
(<http://staff.ustc.edu.cn/~sycheng/ssat/books/The.IDA.Pro.Book.2ed.pdf>)
- [CS 2150 gdb tutorial](https://aaronbloomfield.github.io/pdr/tutorials/02-gdb/index.html)
(<https://aaronbloomfield.github.io/pdr/tutorials/02-gdb/index.html>)
- [IDA Pro tutorial](http://resources.infosecinstitute.com/basics-of-ida-pro-2/) (<http://resources.infosecinstitute.com/basics-of-ida-pro-2/>)