

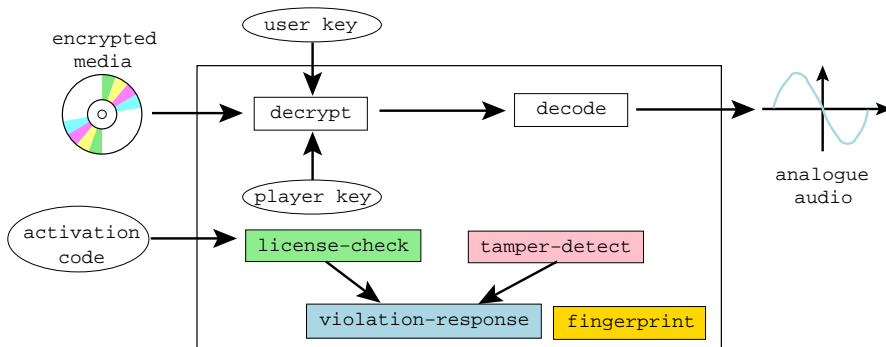
Outline

- 1 The Adversary
- 2 A Cracking Example!

Who's our adversary?

- What does a typical program look like?
- What **valuables** does the program contain?
- What is the adversary's **motivation** for attacking your program?
- What **information** does he start out with as he attacks your program?
- What is his overall **strategy** for reaching his goals?
- What **tools** does he have to his disposal?
- What specific **techniques** does he use to attack the program?

Example Program



Example Program

```
1 typedef unsigned int uint;
2 typedef uint* waddr_t;
3 uint player_key = 0xbabeca75;
4 uint the_key;
5 uint* key = &the_key;
6 FILE* audio;
7 int activation_code = 42;
8
9 void FIRST_FUN(){
10 uint hash (waddr_t addr, waddr_t last) {
11     uint h = *addr;
12     for (; addr<=last; addr++) h^=*addr;
13     return h;
14 }
15 void die(char* msg) {
16     fprintf(stderr, "%s!\n", msg);
17     key = NULL;
```

Example Program

```
19 uint play(uint user_key ,
20           uint encrypted_media [],
21           int media_len) {
22     int code;
23     printf("Please enter activation code: ");
24     scanf("%i",&code);
25     if (code!=activation_code) die("wrong code");
26
27     *key = user_key ^ player_key;
```

Example Program

```
27 int i;
28 for(i=0;i<media_len;i++) {
29     uint decrypted = *key ^ encrypted_media[i];
30     asm volatile (
31         "jmp L1          \n\t"
32         ".align 4       \n\t"
33         ".long 0xb0b5b0b5\n\t"
34         "L1:           \n\t"
35     );
36     if (time(0) > 1221011472) die("expired");
37     float decoded = (float)decrypted;
38     fprintf(audio,"%f\n",decoded); fflush(audio);
39 }
40 }
```

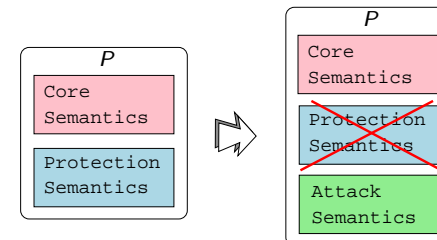
Example Program

```
41 void LAST_FUN(){}
42 uint player_main (uint argc , char *argv []) {
43     uint user_key = ...
44     uint encrypted_media[100] = ...
45     uint media_len = ...
46     uint hashVal = hash((waddr_t)FIRST_FUN,
47                        (waddr_t)LAST_FUN);
48     if (hashVal != HASH) die("tampered");
49     play(user_key , encrypted_media , media_len);
50 }
```

What's the Adversary's Motivation?

The adversary's wants to

- remove the **protection semantics**.
- add his own **attack semantics** (ability to save game-state, print,...)
- ensure that the core semantics remains unchanged.



What does he want to do to our Player program?

- get decrypted digital media
- extract the `player_key`
- use the program after the expiration date
 - remove use-before check
 - remove activation code
- distribute the program to other users
 - remove fingerprint `0xb0b5b0b5`
- reverse engineer the algorithms in the player

What are the methods of attack?

- 1 the **black box** phase
 - feed the program inputs,
 - record its outputs,
 - draw conclusions about its behavior.
- 2 the **dynamic analysis** phase
 - execute the program
 - record which parts get executed for different inputs.
- 3 the **static analysis** phase
 - examining the executable code directly
 - use disassembler, decompiler, ...

What are the methods of attack?

- 4 the **editing** phase
 - use understanding of the internals of the program
 - modify the executable
 - disable license checks
- 5 the **automation** phase.
 - encapsulates his knowledge of the attack in an automated *script*
 - use in future attacks.

Outline

- 1 The Adversary
- 2 A Cracking Example!

Let's crack!

- Let's get a feel for the types of techniques attackers typically use.
- Our example cracking target will be the DRM player.
- Our chief cracking tool will be the gdb debugger.

Step 1: Learn about the executable file

```
> file player
player: ELF 64-bit LSB executable, dynamically linked

> objdump -T player
DYNAMIC SYMBOL TABLE:
0xa4  scanf
0x90  fprintf
0x12  time

> objdump -x player | egrep 'rodata|text|Name'
Name          Size      VMA          LMA          File offset
.text         0x4f8     0x4006a0     0x4006a0     0x6a0
.rodata       0x84      0x400ba8     0x400ba8     0xba8

> objdump -f player | grep start
start address 0x4006a0
```

Step 2: Breaking on library functions

- Treat the program as a black box
- Feed it inputs to see how it behaves.

```
> player 0xca7ca115 1 2 3 4
Please enter activation code: 42
expired!
Segmentation fault
```

- Find the assembly code equivalent of
`if (time(0) > some value)...`
- Replace it with
`if (time(0) <= some value)...`

Example Program

```
27  int i;
28  for (i=0; i<media_len; i++) {
29      uint decrypted = *key ^ encrypted_media[i];
30      asm volatile (
31          "jmp L1          \n\t"
32          ".align 4       \n\t"
33          ".long 0xb0b5b0b5 \n\t"
34          "L1:           \n\t"
35      );
36      if (time(0) > 1221011472) die("expired");
37      float decoded = (float)decrypted;
38      fprintf(audio, "%f\n", decoded); fflush(audio);
39  }
40 }
```

Step 2: Breaking on library functions

At 0x4008bc is the offending conditional branch:

```
> gdb -write -silent --args player 0xca7ca115 \
      1000 2000 3000 4000
(gdb) break time
Breakpoint 1 at 0x400680
(gdb) run
Please enter activation code: 42
Breakpoint 1, 0x400680 in time()
(gdb) where 2
#0 0x400680 in time
#1 0x4008b6 in ??
(gdb) up
#1 0x4008b6 in ??
(gdb) disassemble $pc-5 $pc+7
0x4008b1    callq  0x400680
0x4008b6    cmp    $0x48c72810,%rax
0x4008bc    jle    0x4008c8
```

A Cracking Example!

17/44

Step 2: Breaking on library functions

Patch the executable:

- replace the `jle` with a `jg` (x86 opcode 0x7f)

```
(gdb) set {unsigned char}0x4008bc = 0x7f
(gdb) disassemble 0x4008bc 0x4008be
0x4008bc    jg     0x4008c8
```

A Cracking Example!

18/44

Step 3: Static pattern-matching

- search the executable for character strings.

```
> player 0xca7ca115 1000 2000 3000 4000
tampered!
Please enter activation code: 99
wrong code!
Segmentation fault
```

A Cracking Example!

19/44

Example Program

```
19 uint play(uint user_key ,
20           uint encrypted_media [] ,
21           int media_len) {
22     int code;
23     printf("Please enter activation code: ");
24     scanf("%i",&code);
25     if (code!=activation_code) die("wrong code");
26
27     *key = user_key ^ player_key;
```

A Cracking Example!

20/44

Step 3: Static pattern-matching

- the code that checks the activation code looks something like this:

```
addr1:  .ascii "wrong code"
...
      cmp    read_value,activation_code
      je     somewhere
addr2:  move   addr1, reg0
      call  printf
```

Step 3: Static pattern-matching

- 1 search the data segment to find address `addr1` where `"wrong code"` is allocated.
- 2 search through the text segment for an instruction that contains that address as a literal:

```
(gdb) find 0x400ba8,+0x84,"wrong code"
0x400be2
(gdb) find 0x4006a0,+0x4f8,0x400be2
0x400862
(gdb) disassemble 0x40085d 0x400867
0x40085d    cmp    %eax,%edx
0x40085f    je     0x40086b
0x400861    mov   $0x400be2,%edi
0x400866    callq 0x4007e0
```

Step 3: Static pattern-matching

- Replace the jump-on-equal with a jump-always

```
(gdb) set {unsigned char}0x40085f = 0xeb
(gdb) disassemble 0x40085f 0x400860
0x40085f    jmp    0x40086b
```

Step 4: Watching memory

- the program still crashes with a segmentation violation
- the edits cause the tamper detection mechanism to kick in!

```
> player 0xca7ca115 1000 2000 3000 4000
tampered!
Please enter activation code: 55
Segmentation fault
```

Example Program

```
1 typedef unsigned int uint;
2 typedef uint* waddr_t;
3 uint player_key = 0xbabeca75;
4 uint the_key;
5 uint* key = &the_key;
6 FILE* audio;
7 int activation_code = 42;
8
9 void FIRST_FUN(){
10 uint hash (waddr_t addr, waddr_t last) {
11     uint h = *addr;
12     for (; addr<=last; addr++) h^=*addr;
13     return h;
14 }
15 void die(char* msg) {
16     fprintf(stderr, "%s!\n", msg);
17     key = NULL;
18 }
```

A Cracking Example!

25/44

Example Program

```
27 int i;
28 for(i=0;i<media_len;i++) {
29     uint decrypted = *key ^ encrypted_media[i];
30     asm volatile (
31         "jmp L1          \n\t"
32         ".align 4       \n\t"
33         ".long 0xb0b5b0b5\n\t"
34         "L1:            \n\t"
35     );
36     if (time(0) > 1221011472) die("expired");
37     float decoded = (float)decrypted;
38     fprintf(audio, "%f\n", decoded); fflush(audio);
39 }
40 }
```

A Cracking Example!

26/44

Step 4: Watching memory

- 1 let the program run until it crashes
- 2 rerun the program while watching the address
- 3 find the location which sets it to an illegal value

```
(gdb) run
Program received signal SIGSEGV
0x40087b in ?? ()
(gdb) disassemble 0x40086b 0x40087d
0x40086b    mov     0x2009ce(%rip),%rax    # 0x601240
0x400872    mov     0x2009c0(%rip),%edx    # 0x601238
0x400878    xor     -0x14(%rbp),%edx
0x40087b    mov     %edx,(%rax)
```

A Cracking Example!

27/44

Step 4: Watching memory

- 1 set a watchpoint
- 2 rerun the program from the beginning

```
(gdb) watch *0x601240
(gdb) run
tampered!
Hardware watchpoint 2: *0x601240

Old value = 6296176
New value = 0

0x400811 in ?? ()

(gdb) disassemble 0x400806 0x400812
0x400806    movq   $0x0,0x200a2f(%rip)    # 0x601240
0x400811    leaveq
```

A Cracking Example!

28/44

Step 4: Watching memory

- the instruction at 0x400806 is setting the word at address 0x601240 to 0!
- This corresponds to

```
void die(char* msg) {  
    fprintf(stderr, "%s!\n", msg);  
    key = NULL;  
}
```

Step 4: Watching memory

- overwrite with a sequence of `nop` instructions (x86 opcode 0x90):

```
(gdb) set {unsigned char}0x400806 = 0x90  
    ....  
(gdb) set {unsigned char}0x400810 = 0x90  
  
(gdb) disassemble 0x400806 0x400812  
0x400806    nop  
    ...  
0x400810    nop  
0x400811    leaveq
```

Step 5: Recovering internal data

- 1 ask the debugger to print out decrypted media data!

```
(gdb) hbreak *0x4008a6  
(gdb) commands  
>x/x -0x8+$rbp  
>continue  
>end  
(gdb) cont  
Please enter activation code: 42  
Breakpoint 2, 0x4008a6  
0x7fffffffdc88: 0xbabec99d  
Breakpoint 2, 0x4008a6  
0x7fffffffdc88: 0xbabecda5  
    ...
```

Step 6: Tampering with the environment

- 1 To avoid triggering the timeout, wind back the system clock!
- 2 Change the library search path to force the program to pick up hacked libraries!
- 3 Hack the OS (we'll see this later).

Step 9: Decompilation

```
L080482A0(A8, Ac, A10) {
    ebx = A8;
    esp = "Please enter activation code: ";
    eax = L080499C0();
    V4 = ebp - 16;
    *esp = 0x80a0831;
    eax = L080499F0();
    eax = *(ebp - 16);
    if(eax != *L080BE2CC) {
        V8 = "wrong code";
        V4 = 0x80a082c;
        *esp = *L080BE704;
        eax = L08049990();
        *L080BE2C8 = 0;
    }
}
```

A Cracking Example!

37/44

Example Program

```
19 uint play(uint user_key ,
20           uint encrypted_media [],
21           int media_len) {
22     int code;
23     printf("Please enter activation code: ");
24     scanf("%i",&code);
25     if (code!=activation_code) die("wrong code");
26
27     *key = user_key ^ player_key;
```

A Cracking Example!

38/44

```
    eax = *L080BE2C8;
    edi = 0;
    ebx = ebx ^ *L080BE2C4;
    *eax = ebx;
    eax = A10;
    if(eax <= 0) {} else {
        while(1) {
            esi = *(Ac + edi * 4);
L08048368: *esp = 0;
            if(L08056DD0() > 1521011472) {
                V8 = "expired";
                V4 = 0x80a082c;
                *esp = *L080BE704;
                L08049990();
                *L080BE2C8 = 0;
            }
        }
    }
```

Example Program

```
1 typedef unsigned int uint;
2 typedef uint* waddr_t;
3 uint player_key = 0xbabeca75;
4 uint the_key;
5 uint* key = &the_key;
6 FILE* audio;
7 int activation_code = 42;
8
9 void FIRST_FUN(){}
10 uint hash (waddr_t addr, waddr_t last) {
11     uint h = *addr;
12     for (;addr<=last;addr++) h^=*addr;
13     return h;
14 }
15 void die(char* msg) {
16     fprintf(stderr,"%s!\n",msg);
17     key = NULL;
```

A Cracking Example!

40/44

```

    ebx = ebx ^ esi;
    (save)0;
    edi = edi + 1;
    (save)ebx;
    esp = esp + 8;
    V8 = *esp;
    V4 = "%f\n"; *esp = *L080C02C8;
    eax = L08049990();
    eax = *L080C02C8;
    *esp = eax;
    eax = L08049A20();
    if (edi == A10) { goto L080483a7; }
    eax = *L080BE2C8; ebx = *eax;
}
ch = 176; ch = 176;
goto L08048368;
}
L080483a7:
}

```

Example Program

```

1 typedef unsigned int uint;
2 typedef uint* waddr_t;
3 uint player_key = 0xbabeca75;
4 uint the_key;
5 uint* key = &the_key;
6 FILE* audio;
7 int activation_code = 42;
8
9 void FIRST_FUN(){}
10 uint hash (waddr_t addr, waddr_t last) {
11     uint h = *addr;
12     for (; addr<=last; addr++) h^=*addr;
13     return h;
14 }
15 void die(char* msg) {
16     fprintf(stderr, "%s!\n", msg);
17     key = NULL;

```

```

L080483AF(A8, Ac) {
    ...
    ecx = 0x8048260;
    edx = 0x8048230;
    eax = *L08048230;
    if (0x8048260 >= 0x8048230) {
        do {
            eax = eax ^ *edx;
            edx = edx + 4;
        } while (ecx >= edx);
    }
    if (eax != 318563869) {
        V8 = "tampered";
        V4 = 0x80a082c;
        *esp = *L080BE704;
        L08049990();
        *L080BE2C8 = 0;
    }
    V8 = A8 - 2;
    V4 = ebp + -412;
    *esp = *(ebp + -416);
    return(L080482A0());
}

```

Discussion

Who is our prototypical cracker? He can

- **pattern-match** on static code and execution patterns,
- relate external program behavior to internal code locations,
- **disassemble** and **decompile** binary machine code,
- **debug** binary code without access to source code,
- **compare** (statically or dynamically) two closely related versions of the same program,
- **modify** the executable and its execution environment.