05_virtual2

In this binary, which is not stripped after compiling, I added some more complex classes. There is a class `Cipher` which implements a nonsense crypto algorithm. A class `XorCipher` is the child class of `Cipher`. XorCipher overrides methods of Cipher. So this time we have some virtual methods, but not pure virtual.

```
#include <iostream>
#include <cstdint>
#include <string.h>

class Cipher {

protected:
    uint8_t key[32];
    uint8_t iv[16];
public:
    virtual int init(uint8_t *key, uint8_t *iv);
    virtual int encrypt(uint8_t *data, uint32_t datalen);
    virtual int decrypt(uint8_t *data, uint32_t datalen);
    virtual int deinit(void);
};

int
Cipher::init(uint8_t *key, uint8_t *iv)
{
    int i = 0;
    for(i = 0; i < 32; i++)
    {
        this->key[i] = key[i] + iv[i % 16];
    }

    for(i = 0; i < 16; i++)
    {
        this->iv[i] = iv[i];
    }
    return 0;
}

int
Cipher::encrypt(uint8_t *data, uint32_t datalen)
{
    int i = 0;
    std::cout << "Cipher::encrypt" << std::endl;
    for(i = 0; i < datalen; i++)
```

```
    {
        data[i] ^= this->key[i % 32];
        data[i] += 32;
    }
    return 0;
}

int
Cipher::decrypt(uint8_t *data, uint32_t datalen)
{
    int i = 0;
    std::cout << "Cipher::decrypt" << std::endl;
    for(i = 0; i < datalen; i++)
    {
        data[i] -= 32;
        data[i] ^= this->key[i % 32];
    }
    return 0;
}

int
Cipher::deinit(void)
{
    int i = 0;
    for(i = 0; i < 32; i++)
    {
        this->key[i] = 0x00;
    }
    for(i = 0; i < 16; i++)
    {
        this->iv[i] = 0x00;
    }
    return 0;
}

class XorCipher : public Cipher {

    int init(uint8_t *key, uint8_t *iv) override;
    int encrypt(uint8_t *data, uint32_t datalen) override;
    int decrypt(uint8_t *data, uint32_t datalen) override;
    int deinit(void) override;

};

int
XorCipher::init(uint8_t *key, uint8_t *iv)
{
    int i = 0;
    for(i = 0; i < 32; i++)
```

```
    {
        this->key[i] = key[i] ^ iv[i % 16];
    }

    for(i = 0; i < 16; i++)
    {
        this->iv[i] = iv[i];
    }
    return 0;
}

int
XorCipher::encrypt(uint8_t *data, uint32_t datalen)
{
    std::cout << "XorCipher::encrypt" << std::endl;
    int i = 0;
    for(i = 0; i < datalen; i++)
    {
        data[i] ^= this->key[i % 32];
    }
    return 0;
}

int
XorCipher::decrypt(uint8_t *data, uint32_t datalen)
{
    std::cout << "XorCipher::decrypt" << std::endl;
    return this->encrypt(data, datalen);
}

int
XorCipher::deinit(void)
{
    int i = 0;
    for(i = 0; i < 32; i++)
    {
        this->key[i] = 0x00;
    }
    for(i = 0; i < 16; i++)
    {
        this->iv[i] = 0x00;
    }
    return 0;
}

int
main(void)
{
    char message[] = "this is a secret message for you";
```

```
    uint8_t key[32] = {
0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x
0e,0x0f,
0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x
1e,0x1f
    };
    uint8_t iv[16] = {
0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4a,0x4b,0x4c,0x4d,0x
4e,0x4f
    };

    Cipher *cipher = new XorCipher;

    cipher->init(key, iv);

    cipher->encrypt((uint8_t*)&message[0], strlen(message));
    std::cout << "encrypted: " << message << std::endl;

    cipher->decrypt((uint8_t*)&message[0], strlen(message));
    std::cout << "decrypted: " << message << std::endl;

    cipher->deinit();

    return 0;
}
```

What's interesting now is the main function, where the methods are called.

```
                    ************************************************
                    *                    FUNCTION                  *
                    ************************************************
                    undefined main()

     001011f4 fd 7b       stp      x29,x30,[sp, #local_80]!
              b8 a9
     001011f8 fd 03       mov      x29,sp
              00 91
     001011fc f3 0b       str      x19,[sp, #local_70]
              00 f9
     00101200 00 00       adrp     x0,0x101000
              00 90
     00101204 01 e0       add      x1,x0,#0x578
```

```
             15 91
     00101208 e0 43       add      x0,sp,#0x50
             01 91
     0010120c 22 0c       ldp      x2,x3,[x1]⇒s_this_is_a_secret_messa
= "this is a secret messa
             40 a9
= "a secret message for y
     00101210 02 0c       stp      x2,x3,[x0]⇒local_30
             00 a9
     00101214 22 0c       ldp      x2,x3,[x1, #offset s__message_for_yo
= " message for you"
             41 a9
= " for you"
     00101218 02 0c       stp      x2,x3,[x0, #local_20]
             01 a9
     0010121c 21 80       ldrb     w1,[x1, #0x20]⇒s__00101578+32
= ""
             40 39
     00101220 01 80       strb     w1,[x0, #local_10]
             00 39
     00101224 00 00       adrp     x0,0x101000
             00 90
     00101228 00 80       add      x0,x0,#0x5a0
             16 91
     0010122c e2 c3       add      x2,sp,#0x30
             00 91
     00101230 e3 03       mov      x3⇒DAT_001015a0,x0
= 0706050403020100h
             00 aa
     00101234 60 04       ldp      x0,x1,[x3]⇒DAT_001015a0
= 0706050403020100h
             40 a9
= 0F0E0D0C0B0A0908h
     00101238 40 04       stp      x0,x1,[x2]⇒local_50
             00 a9
     0010123c 60 04       ldp      x0,x1,[x3, #offset DAT_001015b0]
= 1716151413121110h
             41 a9
= 1F1E1D1C1B1A1918h
     00101240 40 04       stp      x0,x1,[x2, #local_40]
             01 a9
     00101244 00 00       adrp     x0,0x101000
             00 90
```

```
     00101248 00 00       add      x0,x0,#0x5c0
              17 91
     0010124c 00 04       ldp      x0,x1,[x0]⇒DAT_001015c0
= 4746454443424140h
              40 a9
= 4F4E4D4C4B4A4948h
     00101250 e0 07       stp      x0,x1,[sp, #local_60]
              02 a9
     00101254 00 07       mov      x0,#0x38
              80 d2
                                use new to allocate a XorCipher object
on the heap and get a pointer to it
     00101258 2a fe       bl       <EXTERNAL>::operator.new
void * operator.new(ulong
              ff 97
     0010125c f3 03       mov      x19,x0
              00 aa
     00101260 e0 03       mov      x0,x19
              13 aa
     00101264 73 00       bl       XorCipher::XorCipher
undefined XorCipher(XorCi
              00 94
     00101268 f3 3f       str      x19,[sp, #var_object]
              00 f9
     0010126c e0 3f       ldr      x0,[sp, #var_object]
              40 f9
     00101270 00 00       ldr      x0,[x0]
              40 f9
     00101274 03 00       ldr      x3,[x0]
              40 f9
     00101278 e1 83       add      x1,sp,#0x20
              00 91
     0010127c e0 c3       add      x0,sp,#0x30
              00 91
     00101280 e2 03       mov      x2,x1
              01 aa
     00101284 e1 03       mov      x1,x0
              00 aa
     00101288 e0 3f       ldr      x0,[sp, #var_object]
              40 f9
                        this is "init"
     0010128c 60 00       blr      x3
              3f d6
```

```
     00101290 e0 3f       ldr      x0,[sp, #var_object]
              40 f9
     00101294 00 00       ldr      x0,[x0]
              40 f9
     00101298 00 20       add      x0,x0,#0x8
              00 91
                       resolve the function in the vtable
     0010129c 13 00       ldr      x19,[x0]
              40 f9
     001012a0 e0 43       add      x0,sp,#0x50
              01 91
     001012a4 07 fe       bl       <EXTERNAL>::strlen
size_t strlen(char * __s)
              ff 97
     001012a8 e1 03       mov      w1,w0
              00 2a
     001012ac e0 43       add      x0,sp,#0x50
              01 91
                       this is the string length
     001012b0 e2 03       mov      w2,w1
              01 2a
                       this is the message
     001012b4 e1 03       mov      x1,x0
              00 aa
     001012b8 e0 3f       ldr      x0,[sp, #var_object]
              40 f9
                       holds the address of "encrypt"
     001012bc 60 02       blr      x19
              3f d6
     001012c0 00 00       adrp     x0,0x101000
              00 90
     001012c4 01 60       add      x1⇒s_encrypted:_00101558,x0,#0x558
= "encrypted: "
              15 91
     001012c8 80 00       adrp     x0,0x111000
              00 90
     001012cc 00 d8       ldr      x0⇒std::cout,[x0, #0xfb0]⇒→std::c
= ??
              47 f9
= 00113048
     001012d0 08 fe       bl       <EXTERNAL>::std::operator<<
basic_ostream * operator<
              ff 97
```

```
     001012d4 e2 03       mov      x2,x0
              00 aa
     001012d8 e0 43       add      x0,sp,#0x50
              01 91
     001012dc e1 03       mov      x1,x0
              00 aa
     001012e0 e0 03       mov      x0,x2
              02 aa
     001012e4 03 fe       bl       <EXTERNAL>::std::operator<<
basic_ostream * operator<
              ff 97
     001012e8 e2 03       mov      x2,x0
              00 aa
     001012ec 80 00       adrp     x0,0x111000
              00 90
     001012f0 01 d0       ldr      x1=>EXTERNAL>::std::endl<char,std::
= ??
              47 f9
= 00113008
     001012f4 e0 03       mov      x0,x2
              02 aa
     001012f8 06 fe       bl       <EXTERNAL>::std::basic_ostream<char,
undefined operator<<(basi
              ff 97
                       this part fetches the "decrypt" function
                       by add x0,x0,#0x10 which is:
                       x0 = object->init
                       x0 + 0x10 = object->decrypt
     001012fc e0 3f       ldr      x0,[sp, #var_object]
              40 f9
     00101300 00 00       ldr      x0,[x0]
              40 f9
     00101304 00 40       add      x0,x0,#0x10
              00 91
     00101308 13 00       ldr      x19,[x0]
              40 f9
     0010130c e0 43       add      x0,sp,#0x50
              01 91
     00101310 ec fd       bl       <EXTERNAL>::strlen
size_t strlen(char * __s)
              ff 97
     00101314 e1 03       mov      w1,w0
              00 2a
```

```
     00101318 e0 43       add      x0,sp,#0×50
              01 91
     0010131c e2 03       mov      w2,w1
              01 2a
     00101320 e1 03       mov      x1,x0
              00 aa
     00101324 e0 3f       ldr      x0,[sp, #var_object]
              40 f9
     00101328 60 02       blr      x19
              3f d6
     0010132c 00 00       adrp     x0,0×101000
              00 90
     00101330 01 a0       add      x1⇒s_decrypted:_00101568,x0,#0×568
= "decrypted: "
              15 91
     00101334 80 00       adrp     x0,0×111000
              00 90
     00101338 00 d8       ldr      x0⇒std::cout,[x0, #0×fb0]⇒→std::c
= ??
              47 f9
= 00113048
     0010133c ed fd       bl       <EXTERNAL>::std::operator<<
basic_ostream * operator<
              ff 97
     00101340 e2 03       mov      x2,x0
              00 aa
     00101344 e0 43       add      x0,sp,#0×50
              01 91
     00101348 e1 03       mov      x1,x0
              00 aa
     0010134c e0 03       mov      x0,x2
              02 aa
     00101350 e8 fd       bl       <EXTERNAL>::std::operator<<
basic_ostream * operator<
              ff 97
     00101354 e2 03       mov      x2,x0
              00 aa
     00101358 80 00       adrp     x0,0×111000
              00 90
     0010135c 01 d0       ldr      x1⇒<EXTERNAL>::std::endl<char,std::
= ??
              47 f9
= 00113008
```

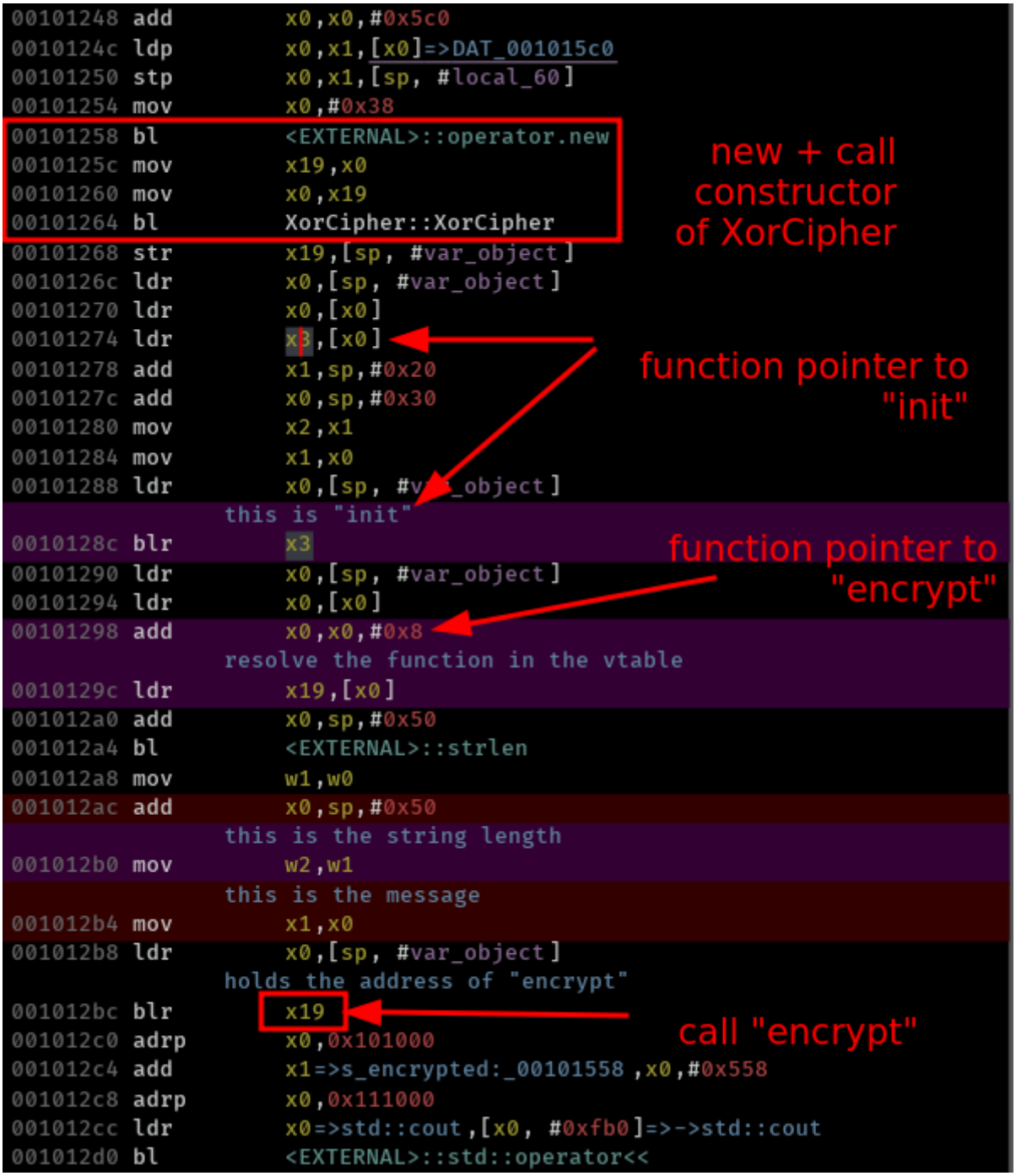
```
     00101360 e0 03       mov      x0,x2
              02 aa
     00101364 eb fd       bl       <EXTERNAL>::std::basic_ostream<char,
undefined operator<<(basi
              ff 97
     00101368 e0 3f       ldr      x0,[sp, #var_object]
              40 f9
     0010136c 00 00       ldr      x0,[x0]
              40 f9
                      object+0x18 = "deinit"
     00101370 00 60       add      x0,x0,#0x18
              00 91
     00101374 01 00       ldr      x1,[x0]
              40 f9
     00101378 e0 3f       ldr      x0,[sp, #var_object]
              40 f9
     0010137c 20 00       blr      x1
              3f d6
     00101380 00 00       mov      w0,#0x0
              80 52
     00101384 f3 0b       ldr      x19,[sp, #local_70]
              40 f9
     00101388 fd 7b       ldp      x29⇒local_80,x30,[sp], #0x80
              c8 a8
     0010138c c0 03       ret
              5f d6
```

Here is the table which holds pointer to the methods of XorEncrypt. This is taken directly from the XorCipher constructor call `XorCipher::XorCipher`.

```
                      PTR_init_00111d28                       XREF[1]:
XorCipher:00101450(*)
     00111d28 88 0f       addr     XorCipher::init
              10 00
              00 00
     00111d30 5c 10       addr     XorCipher::encrypt
              10 00
              00 00
     00111d38 18 11       addr     XorCipher::decrypt
              10 00
              00 00
     00111d40 7c 11       addr     XorCipher::deinit
```

```
10 00
00 00
```

Let's see this as an image, with annotations. I'll break this into parts, below is the constructor, init and encrypt calls. The annotations in the disassembly and added arrows should give a good idea whats going on.

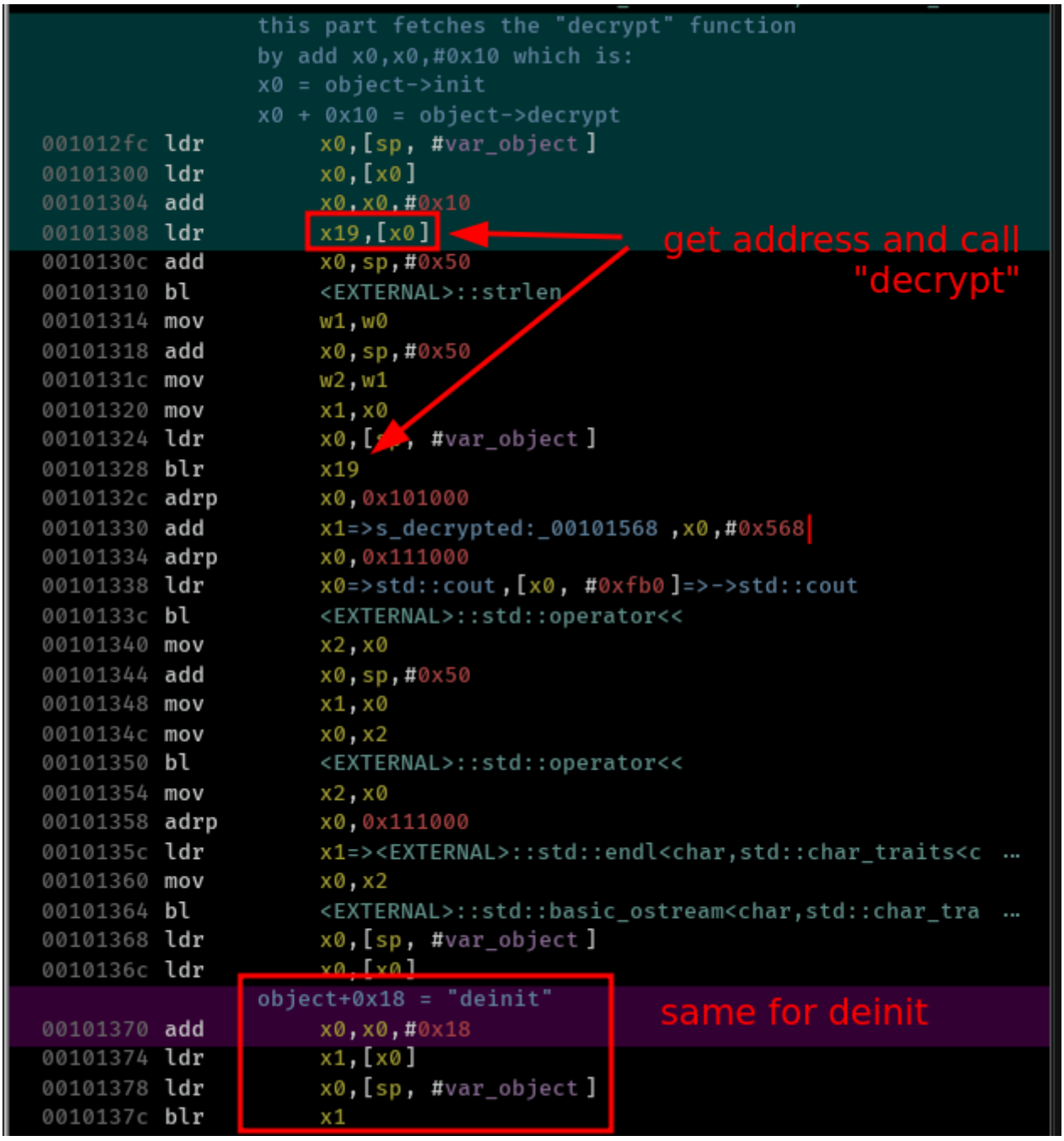


As you can see from the table of function pointers:

```
init    = table + 0×00
encrypt = table + 0×08
```

```
decrypt = table + 0x10
deinit  = table + 0x18
```

Let's look at the next part:



Once you know what the functions in the table do, using the offsets makes it easy to get an idea which methods are called. Of course, this binary is not stripped and contains prints to stdout to guide the analysis, making it even more easy to figure out the workings of the code.