## 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\rightarrowkey[i] = key[i] + iv[i % 16];
    for(i = 0; i < 16; i++)
        this\rightarrowiv[i] = iv[i];
    return 0;
int
Cipher::encrypt(uint8_t *data, uint32_t datalen)
    int i = 0;
    std::cout << "Cipher::encrypt" << std::endl;</pre>
    for(i = 0; i < datalen; i++)</pre>
```

```
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;</pre>
    for(i = 0; i < datalen; i++)</pre>
        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\rightarrowkey[i] = 0 \times 00;
    for(i = 0; i < 16; i++)
        this \rightarrow iv[i] = 0 \times 00;
    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\rightarrowkey[i] = key[i] ^{\prime} iv[i % 16];
    for(i = 0; i < 16; i++)
         this \rightarrow iv[i] = iv[i];
    return 0;
int
XorCipher::encrypt(uint8_t *data, uint32_t datalen)
    std::cout << "XorCipher::encrypt" << std::endl;</pre>
    int i = 0;
    for(i = 0; i < datalen; i \leftrightarrow b)
         data[i] '= this→key[i % 32];
    return 0;
int
XorCipher::decrypt(uint8_t *data, uint32_t datalen)
    std::cout << "XorCipher::decrypt" << std::endl;</pre>
    return this→encrypt(data, datalen);
int
XorCipher::deinit(void)
    int i = 0;
    for(i = 0; i < 32; i++)
         this \rightarrow key[i] = 0×00;
    for(i = 0; i < 16; i++)
         this \rightarrow iv[i] = 0×00;
    return 0;
int
main(void)
    char message[] = "this is a secret message for you";
```

```
uint8 t key[32] = {
0 \times 00, 0 \times 01, 0 \times 02, 0 \times 03, 0 \times 04, 0 \times 05, 0 \times 06, 0 \times 07, 0 \times 08, 0 \times 09, 0 \times 08, 0 \times 06, 0 \times 04, 0 \times 09, 0 \times 08, 0 \times 09, 0 \times 08, 0 \times 09, 0 \times 
0e,0×0f,
0 \times 10, 0 \times 11, 0 \times 12, 0 \times 13, 0 \times 14, 0 \times 15, 0 \times 16, 0 \times 17, 0 \times 18, 0 \times 19, 0 \times 1a, 0 \times 1b, 0 \times 1c, 0 \times 1d, 0 \times 10, 0 \times 
1e,0×1f
                                               };
                                               uint8_t iv[16] = {
0×40,0×41,0×42,0×43,0×44,0×45,0×46,0×47,0×48,0×49,0×4a,0×4b,0×4c,0×4d,0x
4e,0×4f
                                             };
                                               Cipher *cipher = new XorCipher;
                                               cipher→init(key, iv);
                                                cipher→encrypt((uint8_t*)&message[0], strlen(message));
                                                std::cout << "encrypted: " << message << std::endl;</pre>
                                               cipher→decrypt((uint8_t*)&message[0], strlen(message));
                                                std::cout << "decrypted: " << message << std::endl;</pre>
                                               cipher→deinit();
                                                return 0;
```

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

```
******************
                              FUNCTION
              *****
             undefined main()
                 stp
001011f4 fd 7b
                        x29,x30,[sp, #local_80]!
      b8 a9
001011f8 fd 03
                 mov
                        x29,sp
      00 91
                        x19,[sp, #local_70]
001011fc f3 0b
                 str
      00 f9
00101200 00 00
                 adrp
                        x0,0×101000
      00 90
00101204 01 e0
                 add
                        x1, x0, \#0 \times 578
```

```
15 91
     00101208 e0 43
                          add
                                   x0,sp,#0\times50
             01 91
     0010120c 22 0c
                                 x2,x3,[x1]⇒s_this_is_a_secret_messa
                      ldp
= "this is a secret messa
            40 a9
= "a secret message for y
     00101210 02 0c
                                x2,x3,[x0] \Rightarrow local_30
                         stp
             00 a9
                                  x2,x3,[x1, #offset s__message_for_yo
     00101214 22 0c
                         ldp
= " message for you"
            41 a9
= " for you"
                         stp x2,x3,[x0, #local_20]
     00101218 02 0c
            01 a9
                    ldrb w1,[x1, #0×20]⇒s__00101578+32
     0010121c 21 80
            40 39
                                  w1,[x0, #local_10]
     00101220 01 80
                         strb
            00 39
     00101224 00 00
                          adrp
                               x0,0×101000
             00 90
     00101228 00 80
                                  x0,x0,#0×5a0
                         add
            16 91
     0010122c e2 c3
                         add
                                  x2,sp,#0×30
             00 91
     00101230 e3 03
                         mov
                                  x3 \Rightarrow DAT_001015a0, x0
= 0706050403020100h
             00 aa
     00101234 60 04
                         ldp
                                  x0,x1,[x3] \Rightarrow DAT_001015a0
= 0706050403020100h
            40 a9
= 0F0E0D0C0B0A0908h
     00101238 40 04
                         stp x0,x1,[x2] \Rightarrow local 50
             00 a9
     0010123c 60 04
                         ldp x0,x1,[x3, #offset DAT_001015b0]
= 1716151413121110h
            41 a9
= 1F1E1D1C1B1A1918h
     00101240 40 04
                                x0,x1,[x2, #local 40]
                         stp
             01 a9
     00101244 00 00
                                  x0,0×101000
                          adrp
             00 90
```

```
00101248 00 00
                       add
                               x0,x0,#0×5c0
           17 91
    0010124c 00 04
                      ldp x0,x1,[x0] \Rightarrow DAT 001015c0
= 4746454443424140h
           40 a9
= 4F4E4D4C4B4A4948h
    00101250 e0 07
                      stp x0,x1,[sp, #local_60]
           02 a9
    00101254 00 07
                   mov x0,#0×38
           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
                   mov
    00101260 e0 03
                              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,#0×20
           00 91
    0010127c e0 c3
                      add
                              x0,sp,#0×30
           00 91
    00101280 e2 03
                               x2,x1
                      mov
           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
                               х3
           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,#0×8
            00 91
                    resolve the function in the vtable
    0010129c 13 00
                    ldr x19,[x0]
            40 f9
    001012a0 e0 43
                       add
                              x0,sp,#0×50
            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, \#0 \times 50
            01 91
                    this is the string length
    001012b0 e2 03
                    mov w2,w1
            01 2a
                    this is the message
    001012b4 e1 03
                               x1,x0
                    mov
            00 aa
    001012b8 e0 3f
                       ldr x0,[sp, #var_object]
            40 f9
                    holds the address of "encrypt"
                       blr
    001012bc 60 02
                               x19
            3f d6
    001012c0 00 00
                       adrp x0,0×101000
            00 90
    001012c4 01 60
                    add x1 \Rightarrow s_{encrypted} : 00101558, x0, \#0 \times 558
= "encrypted: "
           15 91
    001012c8 80 00
                       adrp x0,0×111000
            00 90
    001012cc 00 d8 ldr x0 \Rightarrow std::cout,[x0, \#0 \times fb0] \Rightarrow \rightarrow std::c
           47 f9
= 00113048
    001012d0 08 fe bl <EXTERNAL>::std::operator<<
basic_ostream * operator<</pre>
            ff 97
```

```
001012d4 e2 03
                         mov
                                   x2,x0
            00 aa
     001012d8 e0 43
                         add
                                   x0, sp, #0 \times 50
            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<</pre>
            ff 97
    001012e8 e2 03 mov
                                  x2,x0
            00 aa
     001012ec 80 00
                                  x0,0×111000
                      adrp
            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,</pre>
undefined operator << (basi
            ff 97
                      this part fetches the "decrypt" function
                      by add x0, x0, \#0 \times 10 which is:
                      x0 = object \rightarrow init
                      x0 + 0 \times 10 = object \rightarrow 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, \#0 \times 10
            00 91
     00101308 13 00
                         ldr
                                  x19,[x0]
            40 f9
     0010130c e0 43
                         add
                                  x0,sp,#0\times50
            01 91
    00101310 ec fd
                     bl <EXTERNAL>::strlen
size_t strlen(char * __s)
            ff 97
    00101314 e1 03
                                  w1,w0
                        mov
            00 2a
```

```
00101318 e0 43
                           add
                                    x0, sp, #0 \times 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 \Rightarrow s_{decrypted:_00101568,x0,\#0\times568}
= "decrypted: "
             15 91
     00101334 80 00
                           adrp x0,0×111000
             00 90
     00101338 00 d8
                          ldr
                                   x0 \Rightarrow std :: cout, [x0, #0 \times fb0] \Rightarrow \rightarrow std :: c
             47 f9
= 00113048
     0010133c ed fd
                      bl <EXTERNAL>::std::operator<<
basic_ostream * operator<</pre>
             ff 97
     00101340 e2 03
                                   x2,x0
                      mov
             00 aa
     00101344 e0 43
                          add
                                    x0, sp, #0 \times 50
             01 91
     00101348 e1 03
                       mov
                                    x1,x0
             00 aa
     0010134c e0 03
                                    x0,x2
                          mov
             02 aa
     00101350 e8 fd
                          bl
                                    <EXTERNAL>::std::operator<<
basic_ostream * operator<</pre>
             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</pre>
              ff 97
     00101368 e0 3f
                            ldr
                                     x0,[sp, #var_object]
              40 f9
     0010136c 00 00
                            ldr
                                     x0,[x0]
              40 f9
                       object+0×18 = "deinit"
     00101370 00 60
                            add
                                     x0, x0, \#0 \times 18
              00 91
     00101374 01 00
                            ldr
                                      x1,[x0]
              40 f9
     00101378 e0 3f
                                     x0,[sp, #var_object]
                            ldr
              40 f9
     0010137c 20 00
                            blr
                                      x1
              3f d6
     00101380 00 00
                                     w0,#0×0
                            mov
              80 52
     00101384 f3 0b
                                      x19,[sp, #local_70]
                            ldr
              40 f9
     00101388 fd 7b
                            ldp
                                      x29 \Rightarrow local_80, x30, [sp], #0 \times 80
              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.

```
00101248 add
                    x0,x0,#0x5c0
0010124c ldp
                    x0, x1, [x0] => DAT_001015c0
00101250 stp
                    x0,x1,[sp, #local_60]
00101254 mov
                    x0,#0x38
00101258 bl
                    <EXTERNAL>::operator.new
0010125c mov
                    x19,x0
00101260 mov
                    x0,x19
00101264 bl
                    XorCipher::XorCipher
                                                  of XorCiphe
                    x19,[sp, #var_object]
00101268 str
0010126c ldr
                    x0,[sp, #var_object]
00101270 ldr
                    x0,[x0]
00101274 ldr
                    xB,[x0]
                                               function poin
00101278 add
                    x1,sp,#0x20
0010127c add
                    x0,sp,#0x30
00101280 mov
                    x2,x1
00101284 mov
                    x1,x0
00101288 ldr
                    x0,[sp, #v __object]
                this is "init
0010128c blr
00101290 ldr
                    x0,[sp, #var_object]
00101294 ldr
                    x0,[x0]
00101298 add
                    x0,x0,#0x8
                resolve the function in the vtable
0010129c ldr
                    x19,[x0]
001012a0 add
                    x0,sp,#0x50
001012a4 bl
                    <EXTERNAL>::strlen
001012a8 mov
                    w1,w0
001012ac add
                    x0,sp,#0x50
                this is the string length
001012b0 mov
                    w2,w1
                this is the message
001012b4 mov
                    x1,x0
001012b8 ldr
                    x0,[sp, #var_object]
                holds the address of "encrypt"
001012bc blr
                   x19
                    x0,0x101000
                                                  call "encrypt"
001012c0 adrp
001012c4 add
                    x1=>s_encrypted:_00101558,x0,#0x558
001012c8 adrp
                    x0,0x111000
                    x0=>std::cout,[x0, #0xfb0]=>->std::cout
001012cc ldr
001012d0 bl
                    <EXTERNAL>::std::operator<<
```

As you can see from the table of function pointers:

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

```
decrypt = table + 0×10
deinit = table + 0×18
```

Let's look at the next part:

```
x0 = object->init
                x0 + 0x10 = object->decrypt
001012fc ldr
                    x0,[sp, #var_object]
00101300 ldr
                    x0,[x0]
00101304 add
                    x0,x0,#0x10
                    x19,[x0]
                                              get address and call
00101308 ldr
0010130c add
                    x0,sp,#0x50
                                                             'decrypt"
00101310 bl
                    <EXTERNAL>::strlen
00101314 mov
                    w1,w0
00101318 add
                    x0,sp,#0x50
0010131c mov
                    w2,w1
00101320 mov
                    x1,x0
00101324 ldr
                    x0,[,, #var_object]
00101328 blr
                    x19
0010132c adrp
                    x0,0x101000
00101330 add
                    x1=>s_decrypted:_00101568 ,x0,#0x568
00101334 adrp
                    x0=>std::cout,[x0, #0xfb0]=>->std::cout
00101338 ldr
0010133c bl
                    <EXTERNAL>::std::operator<<
00101340 mov
                    x2,x0
00101344 add
                    x0,sp,#0x50
00101348 mov
                    x1,x0
0010134c mov
                    x0,x2
00101350 bl
                    <EXTERNAL>::std::operator<<
00101354 mov
                    x2,x0
                    x0,0x111000
00101358 adrp
0010135c ldr
                    x1=><EXTERNAL>::std::endl<char,std::char_traits<c ...</pre>
00101360 mov
                    x0,x2
00101364 bl
                    <EXTERNAL>::std::basic_ostream<char,std::char_tra ...
00101368 ldr
                    x0,[sp, #var_object]
                    va.[va]
0010136c ldr
                object+0x18 = "deinit"
00101370 add
                    x0,x0,#0x18
00101374 ldr
                    x1,[x0]
00101378 ldr
                    x0,[sp, #var_object]
0010137c blr
```

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.