04_stripped

The next binary combines all of the parts from before, but this time, it is stripped. Here is the code, a simple obfuscator for strings:

```cpp
#include <iostream>

const unsigned int MAX_BUFFER_SIZE = 256;

char g_data[] = {

0x55,0x49,0x48,0x52,0x7e,0x48,0x52,0x7e,0x52,0x44,0x42,0x53,0x44,0x55,0x00
};

class ObfuscatorBase {
    protected:
        char internal[MAX_BUFFER_SIZE];
        size_t len;

    public:
        ObfuscatorBase() {
            int i = 0;
            for(i = 0; i < MAX_BUFFER_SIZE; i++)
            {
                this->internal[i] = 0;
            }
        }

        void load(unsigned char *input, size_t len) {
            int i = 0;
            for(i = 0; i < len; i++)
            {
                this->internal[i] = input[i];
            }
            this->len = len;
        }

        void store(unsigned char *output) {
            int i = 0;
            for(i = 0; i < this->len; i++)
            {
                output[i] = this->internal[i];
            }
        }
        virtual void deobfuscate() = 0;

};

class Obfuscator: public ObfuscatorBase {
    private:
        unsigned char key;

    public:

        Obfuscator(unsigned char key) {
            int i = 0;
            for(i = 0; i < MAX_BUFFER_SIZE; i++)
            {
                this->internal[i] = 0;
            }
            this->key = key;
        }

        void deobfuscate() {
            int i = 0;
            for(i = 0; i < this->len; i++)
            {
                this->internal[i] ^= this->key;
            }
        }
};

int
main()
{
    Obfuscator obf = Obfuscator(0x21);
    obf.load((unsigned char*)&g_data[0], 15);
    obf.deobfuscate();
    obf.store((unsigned char*)&g_data[0]);

    printf("> %s\n", g_data);
}
```

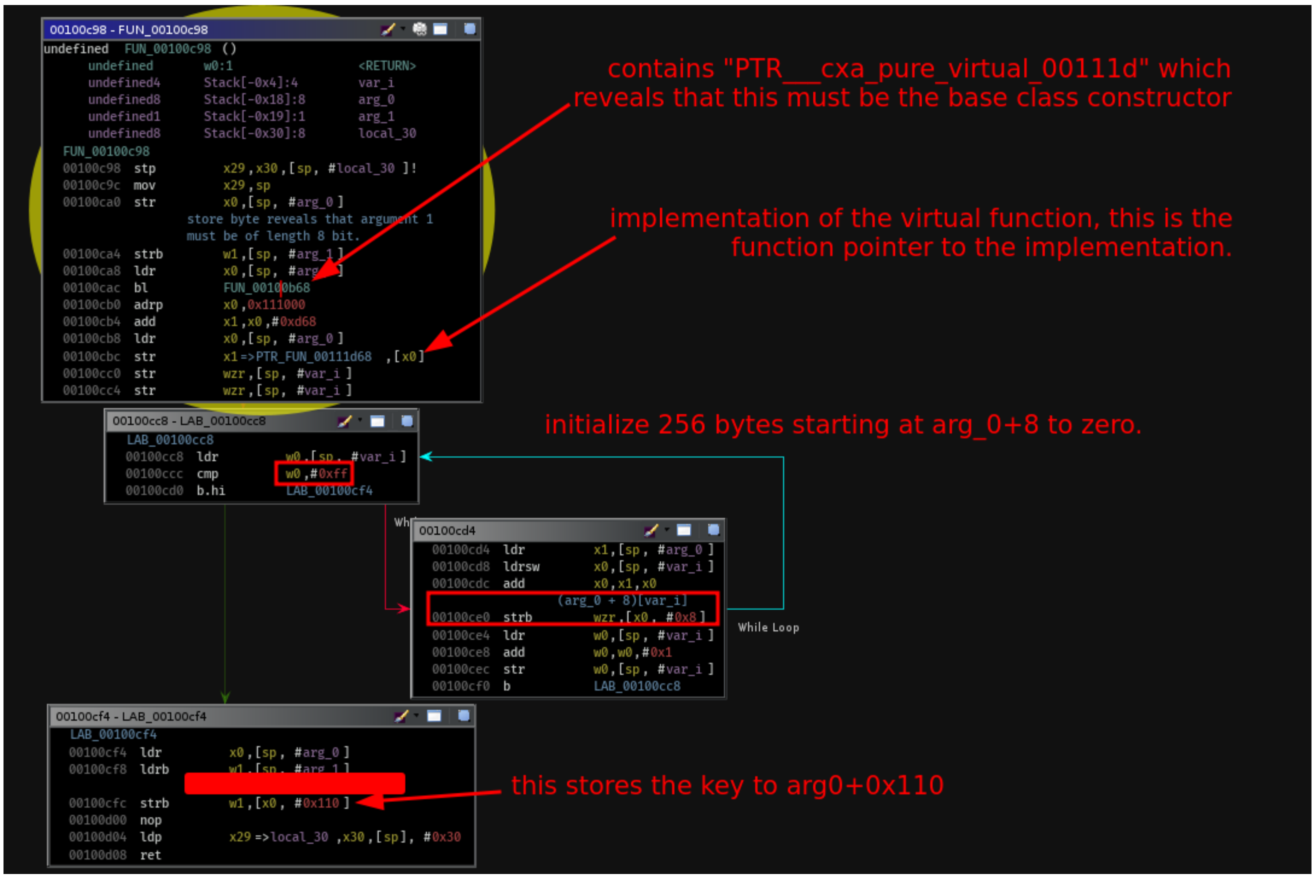
Importing this into ghidra results in no function names being present anymore, including `main`. After some checking the strings, one finds this pretty quickly:

```
undefined FUN_00100a84()
      undefined     w0:1              <RETURN>
      undefined8    Stack[-0x130]:8   local_130
  FUN_00100a84
```

```
00100a84 stp        x29,x30,[sp, #local_130]!
00100a88 mov        x29,sp
00100a8c add        x0,sp,#0x18
00100a90 mov        w1,#0x21
00100a94 bl         FUN_00100c98
00100a98 add        x3,sp,#0x18
00100a9c mov        x2,#0xf
00100aa0 adrp       x0,0x112000
00100aa4 add        x1=>s_UIHR~HR~RDBSDU_00112050,x0,#0x50
00100aa8 mov        x0,x3
00100aac bl         FUN_00100bc0
00100ab0 add        x0,sp,#0x18
00100ab4 bl         FUN_00100d0c
00100ab8 add        x2,sp,#0x18
00100abc adrp       x0,0x112000
00100ac0 add        x1=>s_UIHR~HR~RDBSDU_00112050,x0,#0x50
00100ac4 mov        x0,x2
00100ac8 bl         FUN_00100c34
00100acc adrp       x0,0x112000
00100ad0 add        x1=>s_UIHR~HR~RDBSDU_00112050,x0,#0x50
00100ad4 adrp       x0,0x100000
00100ad8 add        x0=>s_>_%s_00100e28,x0,#0xe28
00100adc bl         <EXTERNAL>::printf
00100ae0 mov        w0,#0x0
00100ae4 ldp        x29=>local_130,x30,[sp], #0x130
00100ae8 ret
```

You can easily identify this as the main function from the code. Now, let's look through it.

The first call to `FUN_00100c98` must be the constructor. If you look at addresses `00100a8c - 00100a94`, you can see our Obfuscator object in `x0`, while `w1` contains the xor key. Here is the constructor:



```
00100c98 - FUN_00100c98
undefined  FUN_00100c98 ()
     undefined      w0:1              <RETURN>
     undefined4     Stack[-0x4]:4     var_i
     undefined8     Stack[-0x18]:8    arg_0
     undefined1     Stack[-0x19]:1    arg_1
     undefined8     Stack[-0x30]:8    local_30
   FUN_00100c98
   00100c98  stp       x29,x30,[sp, #local_30 ]!
   00100c9c  mov       x29,sp
   00100ca0  str       x0,[sp, #arg_0 ]
                    store byte reveals that argument 1
                    must be of length 8 bit.
   00100ca4  strb      w1,[sp, #arg_1 ]
   00100ca8  ldr       x0,[sp, #arg_0 ]
   00100cac  bl        FUN_00100b68
   00100cb0  adrp      x0,0x111000
   00100cb4  add       x1,x0,#0xd68
   00100cb8  ldr       x0,[sp, #arg_0 ]
   00100cbc  str       x1=>PTR_FUN_00111d68  ,[x0]
   00100cc0  str       wzr,[sp, #var_i ]
   00100cc4  str       wzr,[sp, #var_i ]

00100cc8 - LAB_00100cc8
   LAB_00100cc8
   00100cc8  ldr       w0,[sp, #var_i ]
   00100ccc  cmp       w0,#0xff
   00100cd0  b.hi      LAB_00100cf4

00100cd4
   00100cd4  ldr       x1,[sp, #arg_0 ]
   00100cd8  ldrsw     x0,[sp, #var_i ]
   00100cdc  add       x0,x1,x0
                    (arg_0 + 8)[var_i]
   00100ce0  strb      wzr,[x0, #0x8]
   00100ce4  ldr       w0,[sp, #var_i ]
   00100ce8  add       w0,w0,#0x1
   00100cec  str       w0,[sp, #var_i ]
   00100cf0  b         LAB_00100cc8

00100cf4 - LAB_00100cf4
   LAB_00100cf4
   00100cf4  ldr       x0,[sp, #arg_0 ]
   00100cf8  ldrb      w1,[sp, #arg_1 ]
[redacted]
   00100cfc  strb      w1,[x0, #0x110 ]
   00100d00  nop
   00100d04  ldp       x29=>local_30 ,x30,[sp], #0x30
   00100d08  ret
```

contains "PTR___cxa_pure_virtual_00111d" which reveals that this must be the base class constructor
implementation of the virtual function, this is the function pointer to the implementation.
initialize 256 bytes starting at arg_0+8 to zero.
While Loop
this stores the key to arg0+0x110

Back in the main, we can proceed to the next function:

```
undefined FUN_00100a84()
      undefined      w0:1               <RETURN>
      undefined8     Stack[-0×130]:8    local_130
  FUN_00100a84
  00100a84 stp        x29,x30,[sp, #local_130]!
  00100a88 mov        x29,sp
  00100a8c add        x0,sp,#0×18
  00100a90 mov        w1,#0×21
  00100a94 bl         FUN_00100c98

  00100a98 add        x3,sp,#0×18
                  string length
  00100a9c mov        x2,#0×f
  00100aa0 adrp       x0,0×112000
                  string pointer
  00100aa4 add        x1⇒s_UIHR~HR~RDBSDU_00112050,x0,#0×50
                  obfuscator object.
  00100aa8 mov        x0,x3
                  call member function of obfuscator object
  00100aac bl         FUN_00100bc0
-- SNIP --
```

The function gets the arguments (object, string pointer, string length). Again, from the constructor you can piece together what's going on in FUN_00100bc0:

```
     00100bc0 sub         sp,sp,#0x30

     00100bc4 e0 0f       str      x0,[sp, #arg_this]
              00 f9
     00100bc8 e1 0b       str      x1,[sp, #arg_buffer]
              00 f9
     00100bcc e2 07       str      x2,[sp, #arg_buflen]
              00 f9
     00100bd0 ff 2f       str      wzr,[sp, #var_i]
              00 b9
     00100bd4 ff 2f       str      wzr,[sp, #var_i]
              00 b9
                      LAB_00100bd8                          XREF[1]:
 00100c18(j)
     00100bd8 e0 2f       ldrsw    x0,[sp, #var_i]
              80 b9
     00100bdc e1 07       ldr      x1,[sp, #arg_buflen]
              40 f9
     00100be0 3f 00       cmp      x1,x0
              00 eb
     00100be4 c9 01       b.ls     LAB_00100c1c
              00 54
     00100be8 e0 2f       ldrsw    x0,[sp, #var_i]
              80 b9
     00100bec e1 0b       ldr      x1,[sp, #arg_buffer]
              40 f9
                      index into buffer
                      buffer[i]
     00100bf0 20 00       add      x0,x1,x0
              00 8b
     00100bf4 02 00       ldrb     w2,[x0]
              40 39
     00100bf8 e1 0f       ldr      x1,[sp, #arg_this]
              40 f9
     00100bfc e0 2f       ldrsw    x0,[sp, #var_i]
              80 b9
                      index into object, however the offset
                      +0x8 to get to the internal buffer is
                      added below (0x00100c08)
     00100c00 20 00       add      x0,x1,x0
             00 8b
     00100c04 e1 03       mov      w1,w2
             02 2a
                        offset added.
     00100c08 01 20       strb     w1,[x0, #0x8]
             00 39
     00100c0c e0 2f       ldr      w0,[sp, #var_i]
             40 b9
     00100c10 00 04       add      w0,w0,#0x1
             00 11
     00100c14 e0 2f       str      w0,[sp, #var_i]
             00 b9
     00100c18 f0 ff       b        LAB_00100bd8
             ff 17
                        LAB_00100c1c                          XREF[1]:
00100be4(j)
     00100c1c e0 0f       ldr      x0,[sp, #arg_this]
             40 f9
     00100c20 e1 07       ldr      x1,[sp, #arg_buflen]
             40 f9
                          save the string length to this+0x108
     00100c24 01 84       str      x1,[x0, #0x108]
             00 f9
     00100c28 1f 20       nop
             03 d5
     00100c2c ff c3       add      sp,sp,#0x30
             00 91
     00100c30 c0 03       ret
             5f d6
```

Now, for the deobfuscation routine, we continue in main:

```
-- SNIP --

  ; call to deobfuscation function
  00100ab0 add        x0,sp,#0x18
  00100ab4 bl         FUN_00100d0c

  00100ab8 add        x2,sp,#0x18
  00100abc adrp       x0,0x112000
  00100ac0 add        x1=>s_UIHR~HR~RDBSDU_00112050,x0,#0x50
  00100ac4 mov        x0,x2
  00100ac8 bl          FUN_00100c34
  00100acc adrp        x0,0x112000
  00100ad0 add         x1⇒s_UIHR~HR~RDBSDU_00112050,x0,#0x50
  00100ad4 adrp        x0,0x100000
  00100ad8 add         x0⇒s_>_%s_00100e28,x0,#0xe28
  00100adc bl          <EXTERNAL>::printf
  00100ae0 mov         w0,#0x0
  00100ae4 ldp         x29⇒local_130,x30,[sp], #0x130
  00100ae8 ret
```

Inside the method `FUN_00100d0c`:

```
                      ************************************************
                      *                    FUNCTION                  *
                      ************************************************
                      undefined obf_deobfuscate()
          undefined     w0:1       <RETURN>
          undefined4    Stack[-0x4 var_i
XREF[7]:  00100d14(W),

00100d18(W),

00100d1c(R),

00100d34(R),

00100d54(R),

00100d64(R),

00100d6c(W)
          undefined8    Stack[-0x1 var_this
XREF[5]:  00100d10(W),

00100d20(R),

00100d30(R),

00100d40(R),

00100d50(R)
                      obf_deobfuscate                       XREF[4]:
FUN_00100a84:00100ab4(c),
00100eb8, 00100fb8(*),

00111d68(*)
     00100d0c ff 83       sub      sp,sp,#0x20
             00 d1
     00100d10 e0 07       str      x0,[sp, #var_this]
             00 f9
     00100d14 ff 1f       str      wzr,[sp, #var_i]
             00 b9
     00100d18 ff 1f       str      wzr,[sp, #var_i]
             00 b9
                      LAB_00100d1c                          XREF[1]:
00100d70(j)
     00100d1c e1 1f       ldrsw    x1,[sp, #var_i]
             80 b9
     00100d20 e0 07       ldr      x0,[sp, #var_this]
             40 f9
                      this is the string length
     00100d24 00 84       ldr      x0,[x0, #0x108]
             40 f9
     00100d28 3f 00       cmp      x1,x0
             00 eb
     00100d2c 42 02       b.cs     LAB_00100d74
             00 54
     00100d30 e1 07       ldr      x1,[sp, #var_this]
             40 f9
     00100d34 e0 1f       ldrsw    x0,[sp, #var_i]
             80 b9
     00100d38 20 00       add      x0,x1,x0
             00 8b
                      fetch the next byte from
                      the internal buffer
     00100d3c 01 20       ldrb     w1,[x0, #0x8]
             40 39
     00100d40 e0 07       ldr      x0,[sp, #var_this]
             40 f9
                      fetch the key from this+0x110
     00100d44 00 40       ldrb     w0,[x0, #0x110]
             44 39
     00100d48 20 00       eor      w0,w1,w0
             00 4a
     00100d4c 02 1c       and      w2,w0,#0xff
```

```
            00 12
     00100d50 e1 07       ldr      x1,[sp, #var_this]
            40 f9
     00100d54 e0 1f       ldrsw    x0,[sp, #var_i]
            80 b9
     00100d58 20 00       add      x0,x1,x0
            00 8b
     00100d5c e1 03       mov      w1,w2
            02 2a
     00100d60 01 20       strb     w1,[x0, #0x8]
            00 39
     00100d64 e0 1f       ldr      w0,[sp, #var_i]
            40 b9
     00100d68 00 04       add      w0,w0,#0x1
            00 11
     00100d6c e0 1f       str      w0,[sp, #var_i]
            00 b9
     00100d70 eb ff       b        LAB_00100d1c
            ff 17
                      LAB_00100d74                          XREF[1]:
00100d2c(j)
     00100d74 1f 20       nop
            03 d5
     00100d78 ff 83       add      sp,sp,#0x20
            00 91
     00100d7c c0 03       ret
            5f d6
```

This allows us to piece together the object:

```
    field_0    this + 0x00
    field_1    this + 0x08     internal
    field_2    this + 0x108    len
    field_3    this + 0x110    xorkey
```

I will leave it at that, as the next function which just stores the deobfuscated bytes to `g_data` (no pun intended) it is not interesting.