

Adam Van Prooyen

[About](#)[Blog](#)[Projects](#)

Lost Decryption Writeup

SECCON Quals 2016

`“ I created my own cipher and encrypted the
very important file. However, I lost the
decryption program because of file system
error, so now I cannot read the file. Please
help me ”`

This challenge included a cipher binary, libencrypt.so, key.bin, and flag.enc. I thought this challenge was really fun because it mixed reverse engineering and basic block cryptography.

The first thing I did was to try and run it to see what it did. Unfortunately, the binary is missing the libdecrypt.so, so I was unable to get it running properly. Static analysis it is!

cipher binary

Throwing the cipher binary into Binary Ninja, I found that this program itself was pretty straightforward aside from mostly using registers instead of the stack for variable storage. The usage function was particularly helpful:

```
usage:
00000c10 lea     rdi, [rel usage_str] {"cipher (encrypt|decrypt) key inp..."}
00000c17 sub     rsp, _8
00000c1b call    puts
00000c20 mov     edi, 0xffffffff
00000c25 call    exit
{ Does not return }
```

“ ./cipher (encrypt|decrypt) key input
output] ”

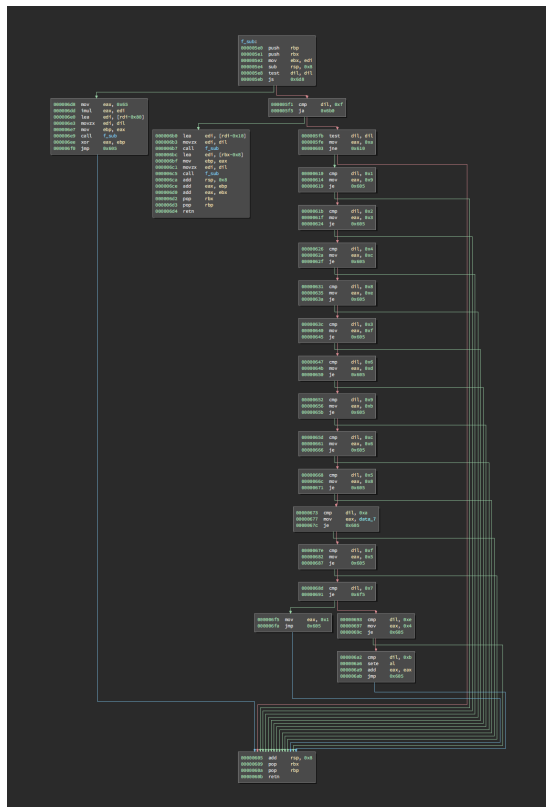
With this knowledge, I was able to figure out what files were being opened. After loading in the input file and the key, the binary enters a loop where the input is encrypted or decrypted 16 bytes at a time and incrementally written to the output file:

```
00000a73 mov     rsi, qword [rsp {key_buf_addr}]
00000a77 mov     rdi, rbx // rbx = input
00000a7a call    r15 // enc/dec(input, key_buf)
00000a7d mov     rcx, rbp // rbp = output fp
00000a80 mov     edx, 0x10
00000a85 mov     esi, 0x1
00000a8a mov     rdi, rbx
00000a8d call    fwrite // fwrite(input, 1, 16, out_fp)
00000a92 jmp     0x8f1
```

libencrypt.so

When I felt like I had an good understanding of how cipher functioned, I decided it was time to figure out how the encryption itself worked. Opening libencrypt.so, I found that the encrypt function was very compact.

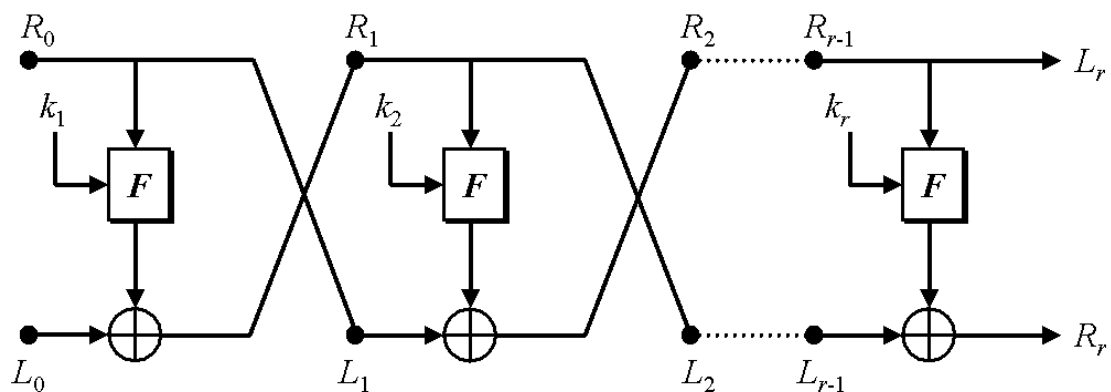
On a high level, encrypt separates the 16 byte input and key into two 8 byte chunks L, R and k_L, k_R respectively. Then, L is xored against f(R, k_R) and k_R is set to f(k_R, 0x9104f95de694dc50). Finally, L and R along with k_L and k_R are swapped. This is repeated 14 times.



Feistel networks

While I usually have nothing better to do with my time than invert nonsensical functions, there is an easier way to deal with decryption. It turns out that this encryption algorithm is almost a textbook feistel network implementation, the same kind of construction used by DES.

Feistel networks have the nice property that for any round function, in this case f , one can easily invert (i.e. decrypt) the whole network without having to invert the round function.




The encryption and decryption algorithms are identical except that the order of the keys (called the key schedule) is reversed. And since feistel networks are symmetric cryptographic algorithms, the key used for encryption is also used for decryption.

Decryption

Now that I had identified the encryption scheme, I needed to implement the decryption algorithm. For this, I was going to need to be able to call `f` and there is no way I was going to reimplement it by hand.

Since Python `ctypes` was giving me a hard time, I just decided to just mmap `libencrypt.so` into a C program and call `f` directly. This ended up working better than I expected and I am probably going to use this technique more in the future.

```
fd = open("libencrypt.so", O_RDWR);
fstat(fd, &st);
libencrypt = mmap(NULL, st.st_size, PROT_EXEC, MAP_PRIVATE, fd, 0);
f = (uint64_t (*)(uint64_t, uint64_t)) &libencrypt[0x700];
```



Because decryption requires the key schedule to be applied in reverse order, I needed to start the feistel network with the last key. Since there is no way to generate the last sub-key on the spot or generate a previous key, I calculated all sub-keys at once and stored them in reverse order:

```
void gen_keys(uint64_t k_L, uint64_t k_R) {
    int i;

    for (i = 0; i < ROUNDS; i++) {
        keys[ROUNDS - i - 1] = k_R;
        k_R = f(k_R, c);
        swap(&k_L, &k_R);
    }
}
```

Now that I had the decryption key schedule and the ciphertext, I just needed to run through the feistel network on a 16 byte piece of the ciphertext:

```

void decrypt(char *ct) {
    uint64_t k_n, L, R;
    int i;

    L = *(uint64_t *) ct;
    R = *((uint64_t *) ct + 1);
    for (i = 0; i < ROUNDS; i++) {
        k_n = keys[i];
        L ^= f(R, k_n);
        swap(&L, &R);
    }
    *(uint64_t *) ct = L;
    *((uint64_t *) ct + 1) = R;
}

```

Running the decrypt function on the whole ciphertext yields the flag:

```

for (i = 0; i < 3; i++) {
    decrypt(&ct[i * 16]);
}
printf("%s\n", ct);

```

“ Flag :

SECCON{Decryption_Of_Feistel_is_EASY!} ”

Downloads

lost.c

lost_decryption.zip

3 Comments

van.prooyen

 Login ▾

 Recommend

 Share

Sort by Best ▾



Join the discussion...



John Sa • 7 months ago

I got an error with your script (and 5.6.8.22). seem to be it can't explorer path

to get_flag

```
ook(0x21000510, <class 'angr.simos.IFuncResolver'>, strcpy())
WARNING | 2016-11-02 01:03:25,261 | angr.project | Address is already hooked [h
ook(0x21000500, <class 'angr.simos.IFuncResolver'>, strcat())
WARNING | 2016-11-02 01:03:25,262 | angr.project | Address is already hooked [h
ook(0x21000480, <class 'angr.simos.IFuncResolver'>, __memcpy_chk())
WARNING | 2016-11-02 01:03:25,264 | angr.project | Address is already hooked [h
ook(0x210004d0, <class 'angr.simos.IFuncResolver'>, strcat())
WARNING | 2016-11-02 01:03:25,266 | angr.project | Address is already hooked [h
ook(0x210004a0, <class 'angr.simos.IFuncResolver'>, __memset_chk())
creating state
adding constraints to stdin
creating path and explorer
running explorer
found solution
Traceback (most recent call last):
  File "solve.py", line 66, in <module>
    main()
  File "solve.py", line 53, in main
    correct_input = ex._f.state.posix.dumps(0) # ex._f is equiv. to ex.found[0]
  File "/home/.../.virtualenvs/angr/local/lib/python2.7/site-packages/angr/surv
eyors/explorer.py", line 117, in _f
    return self.found[0]
IndexError: list index out of range
```

^ | v • Reply • Share ›



Adam Van Prooyen Mod → John Sa • 7 months ago

Interesting, I'm running angr version 5.6.10.12 on Ubuntu 15.04 installed using virtualenv. I just tested the script and its working for me. Can you tell me more about your environment.

^ | v • Reply • Share ›



John Sa → Adam Van Prooyen • 7 months ago

Im currently working on Ubuntu 16.04, use virtualenvwrapper (again, angr 5.6.8.22)

EDIT: After update my angr version, it work correctly

```
ook(0x210004d0, <class 'angr.simos.IFuncResolver'>, strcspn())
WARNING | 2016-11-02 07:22:18,267 | angr.project | Address is already hooked [h
ook(0x210004a0, <class 'angr.simos.IFuncResolver'>, __memcpy_chk())
WARNING | 2016-11-02 07:22:18,270 | angr.project | Address is already hooked [h
ook(0x210004b0, <class 'angr.simos.IFuncResolver'>, strpbrk())
WARNING | 2016-11-02 07:22:18,271 | angr.project | Address is already hooked [h
ook(0x21000530, <class 'angr.simos.IFuncResolver'>, strcpy())
WARNING | 2016-11-02 07:22:18,273 | angr.project | Address is already hooked [h
ook(0x21000520, <class 'angr.simos.IFuncResolver'>, strncat())
WARNING | 2016-11-02 07:22:18,276 | angr.project | Address is already hooked [h
ook(0x210004a0, <class 'angr.simos.IFuncResolver'>, __memcpy_chk())
WARNING | 2016-11-02 07:22:18,277 | angr.project | Address is already hooked [h
ook(0x210004f0, <class 'angr.simos.IFuncResolver'>, strcat())
WARNING | 2016-11-02 07:22:18,279 | angr.project | Address is already hooked [h
ook(0x210004c0, <class 'angr.simos.IFuncResolver'>, __memset_chk())
WARNING | 2016-11-02 07:22:20,342 | angr.project | Re-hooking symbol read
'\x80@\x88\xa6 @ \x86@v@ /\x96vC\x80p D AC`@\`v 0 \x87\x80 @X@ @@\x80`@h
\x9d@ W@UB \x80( \x8c@ @ \x80\x80 F$th 6 J B k[Q` @@@Xp, I @\x80 @
\x80 \x80 @ @ \x80 \x80 \x80
Total time: 283.454195023
```



Btw, thanks for the reply so much

^ | v • Reply • Share ›

* * * * *

Written by **Adam Van Prooyen** on **11 December 2016**