Project3

0316025 賴文揚

Retrieving RSA Modulus

- “Server Hello, Certificate,…” packet -> Secure Sockets Layer -> … -> Certificate -> get modulus value
  - 00e29831eca1ed909e119abc3fd28afe0f7526fc8e3bf1584e4160598327512e1d421c94c28b6224ab8ed96d9298565f72b3ac7cd7cf0c7f4c3b865407d787803e02bcc577f0fe961dd3e6087d86fc854e650681621a1516b92fc6113ac8755d42066efbded530b684631ca1b3e5484c8ed468af41e7e6f240810c8333ccb36a9daac56f5e6515c246f9b433af6c82d6836a111322f71113ae246f42862d9a7e16558b01625ccd5b3fcebcff7dffacde58010bddd58245cac74498875330fbc05ca9fa312af69355ed9a52ec567dbbfc821d3d85717587bfecf7b5891ceee089bc4e36b2cea518c11d05bda4e0be048eca484ac416573c698cda48be5d7518be8d
- Convert it to decimal.

Factorization

- Using factordb
  - Get p = 165496247436666790176644765450146822175238160321715295438224142061506544367523569093501025581265148522515911545522516334416008202505578645693393969776575527113088596739731879688745300711275501631413633830292363304950267213188283170437974782399257186202072520421528176141023967220402935995118767230551264312971
    q = 172843236782782894775242174425482199225662999034227337066715380828301431226860628018223276607609396125109893659339063744320238109537353962290920204393020892616343596012988637012116405051022753833961405337753215882631910970812354386000809274828899211116773063488392562730725564517340380403647420455477343411533

Decrypt Https

- Using factorize_and_retrieve_key.py to gain private key.
- Wireshark decrypt the traffic, then get IV and key.
  - IV: acd08fa0ec3db8a8eb2b43e13cc5f71e0f510c90539f733a364efb4682cac62f
    Key: 6f78020ff31bad18bcae739bf6118216e24959ed810293e2e796dbc11034ce80

Disassembly

- Using Snowman to decompile the binary and compare with raw assembly ‘objfile’ (Gain by objdump)
- Copy IV

```
v10 = reinterpret_cast<struct s0*>(0x804c0e0);
v11 = reinterpret_cast<struct s0*>(reinterpret_cast<int32_t>(ebp5) - 44);
// copy IV
fun_8048940(v11, 0x804c0e0, 32);
```

- Xor with IV

```
v12 = reinterpret_cast<void*>(0);
// 32 times
while (reinterpret_cast<int32_t>(v12) <= reinterpret_cast<int32_t>(31)) {
    // xor with IV
    eax13 = static_cast<uint32_t>(*reinterpret_cast<unsigned char*>(reinterpret_cast<uint32_t>(v12) + (reinterpret_c
    *reinterpret_cast<signed char*>(reinterpret_cast<uint32_t>(v12) + reinterpret_cast<unsigned char>(v9)) = *reinte
    v12 = reinterpret_cast<void*>(reinterpret_cast<uint32_t>(v12) + 1);
}
```

Diassembly

- Xor with key

```
// 32 times
v14 = reinterpret_cast<struct s4*>(0);
while (reinterpret_cast<int32_t>(v14) <= reinterpret_cast<int32_t>(31)) {
    // xor with key
    eax15 = static_cast<uint32_t>(v14->f134529216) ^ static_cast<uint32_t>(*reinterpret_cast<u
    *reinterpret_cast<signed char*>(reinterpret_cast<uint32_t>(v14) + reinterpret_cast<unsign
    v14 = reinterpret_cast<struct s4*>(reinterpret_cast<uint32_t>(v14) + 1);
}
```

- Next 32byte block

```
v10 = v9;
v11 = reinterpret_cast<struct s0*>(reinterpret_cast<int32_t>(ebp5) - 44);
// store cipher as next IV
fun_8048940(v11, v10, 32);
// next block
v9 = reinterpret_cast<struct s0*>(reinterpret_cast<unsigned char>(v9) + 32);
```

Decrypt and Eliminate Padding Byte

- According to the encryption function, we can figure out the decryption function.

```c
void decrypt(int out_fd, uint8_t* buffer){
  uint8_t tmp_key[32];
  memcpy(tmp_key, buffer, 32);
  for (int i=0;i<32;i++){
    buffer[i] = IV[i]^buffer[i];
  }

  for (int i=0;i<32;i++){
    buffer[i] = key[i]^buffer[i];
  }
  memcpy(IV, tmp_key, 32);
  write(out_fd, buffer, 32);
}

int main(){

  int len;
  uint8_t buffer[32];
  int in_fd = open("top_secret.jpg.enc", O_RDONLY);
  int out_fd = open("top_secret.jpg", O_CREAT | O_WRONLY, 0644);
  while ( (len = read(in_fd, buffer, 32)) != 0){
    decrypt(out_fd, buffer);
  }
  return 0;
}
```

decrypt.c

Decrypt and Eliminate Padding Byte

- Using the program ‘decrypt’ (compiled from ‘decrypt.c’) and get the plaintext of the top_secret.jpg.
- Work with following command to detect padding byte and eliminate it (‘ff d9’ is indicated to EOF of jpg file.).

```
$ ./decrypt.c
$ hexdump –C top_secret.jpg
…
00002850 45 00 14 51 45 00 14 51 45 00 74 ff d9 03 03 03
$ truncate –size=-3 top_secret.jpg
```

Finished

- Open top_secret.jpg