NioGuard Security Lab

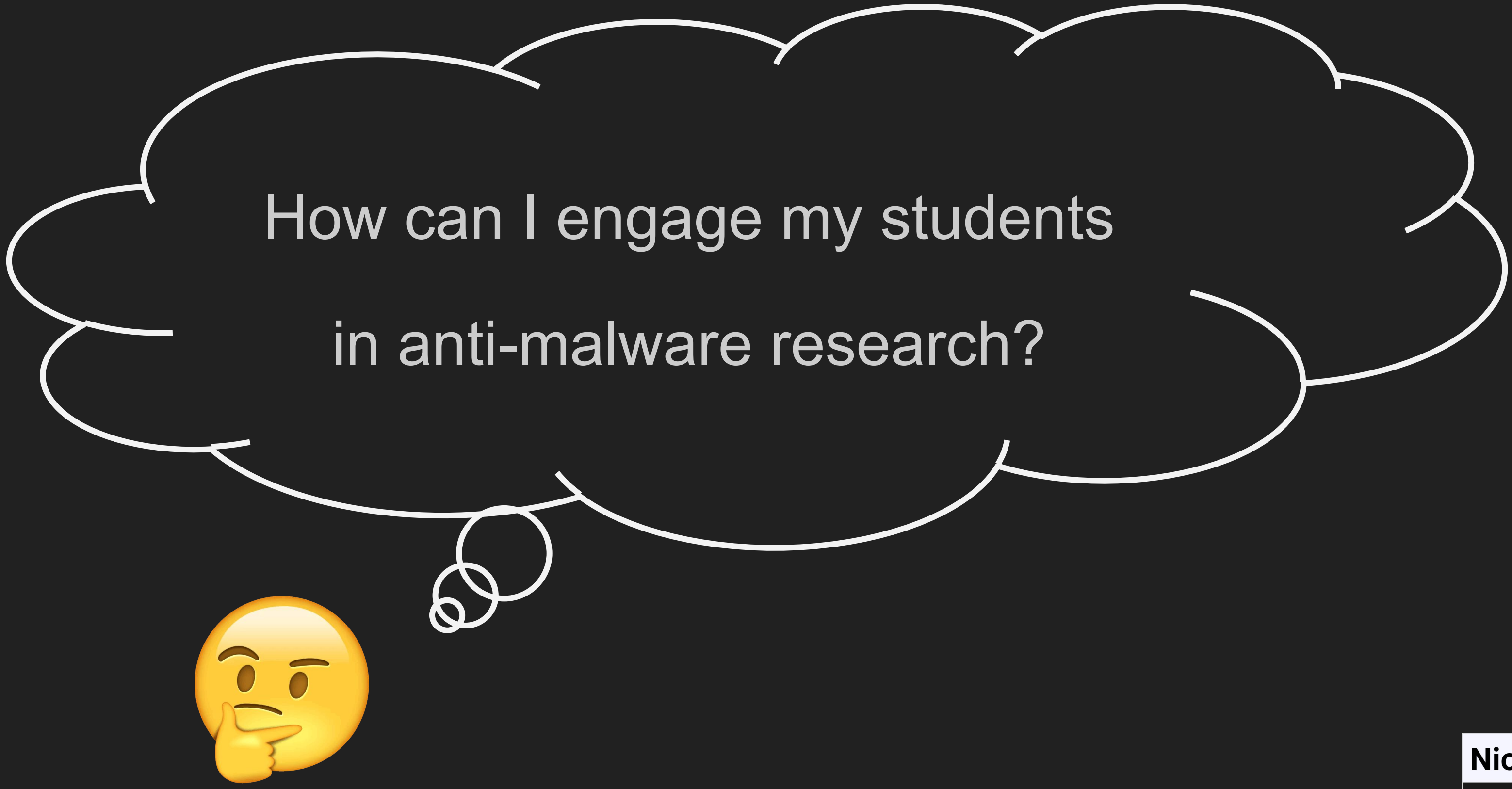
vb 2018
MONTREAL
3 – 5 October 2018

ARTIFICIAL INTELLIGENCE
TO ASSIST WITH
RANSOMWARE CRYPTANALYSIS

Alexander Adamov

CEO/Founder at NioGuard Security Lab,
Lecturer at NURE and BTH Universities

Before



How can I engage my students

in anti-malware research?

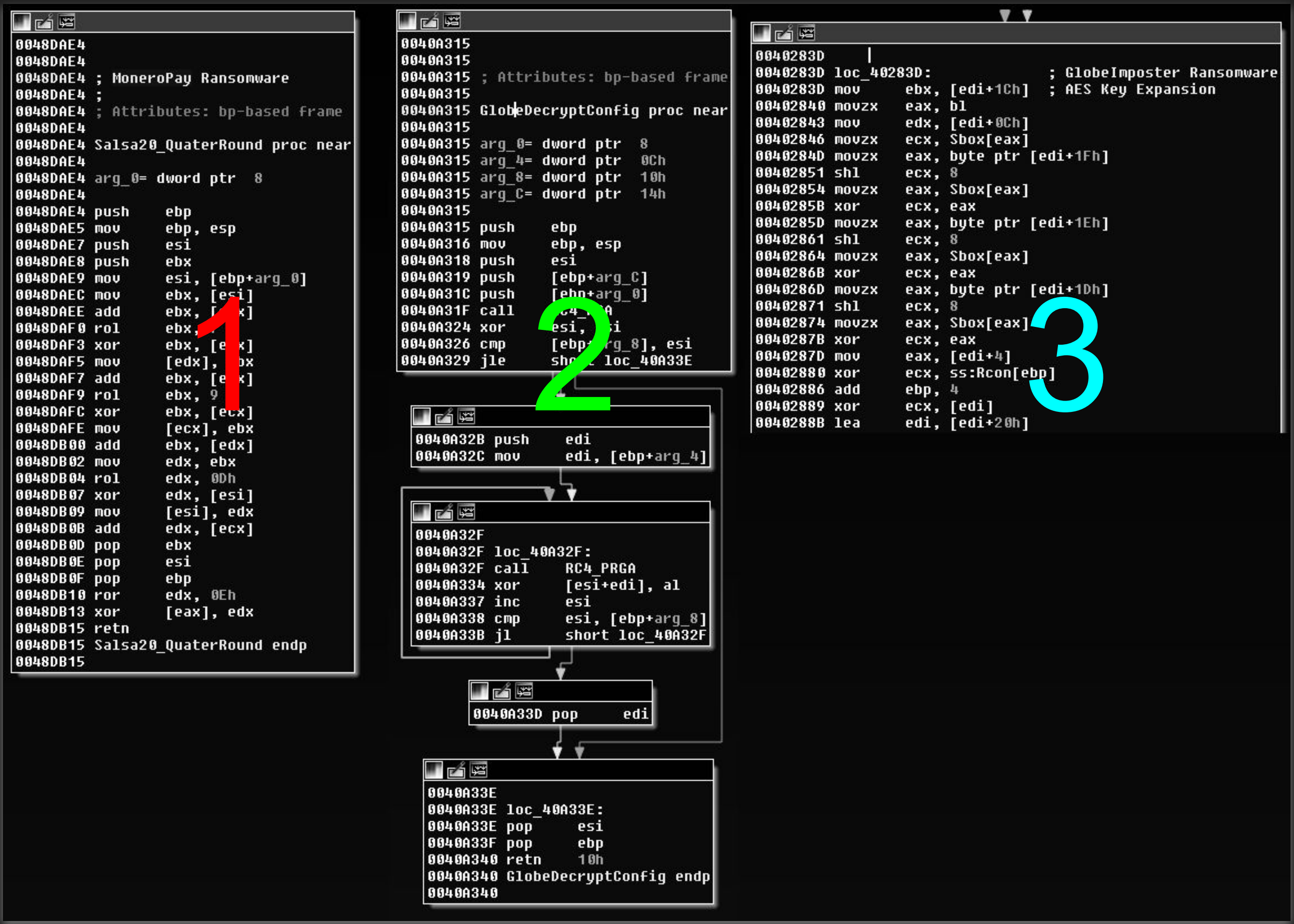
Problem: Ransomware Analysis

Ransomware attack investigation questions:

- Which cipher was used in an attack?
- How does a ransomware generate encryption key(s) and where stores them for future decryption?
- Is it possible to obtain or generate a decryption key or create a decryption tool?

Problem - 2

Custom or hardcoded ciphers in ransomware



```
0048DAE4
0048DAE4
0048DAE4 ; MoneroPay Ransomware
0048DAE4 ;
0048DAE4 ; Attributes: bp-based frame
0048DAE4
0048DAE4 Salsa20_QuaterRound proc near
0048DAE4
0048DAE4 arg_0= dword ptr  8
0048DAE4
0048DAE4 push    ebp
0048DAE5 mov     ebp, esp
0048DAE7 push    esi
0048DAE8 push    ebx
0048DAE9 mov     esi, [ebp+arg_0]
0048DAEC mov     ebx, [esi]
0048DAEE add     ebx, [eax]
0048DAF0 rol     ebx, 7
0048DAF3 xor     ebx, [edx]
0048DAF5 mov     [edx], ebx
0048DAF7 add     ebx, [eax]
0048DAF9 rol     ebx, 9
0048DAFC xor     ebx, [ecx]
0048DAFE mov     [ecx], ebx
0048DB00 add     ebx, [edx]
0048DB02 mov     edx, ebx
0048DB04 rol     edx, 0Dh
0048DB07 xor     edx, [esi]
0048DB09 mov     [esi], edx
0048DB0B add     edx, [ecx]
0048DB0D pop     ebx
0048DB0E pop     esi
0048DB0F pop     ebp
0048DB10 ror     edx, 0Eh
0048DB13 xor     [eax], edx
0048DB15 retn
0048DB15 Salsa20_QuaterRound endp
0048DB15
```

```
0040A315
0040A315
0040A315 ; Attributes: bp-based frame
0040A315
0040A315 GlobeDecryptConfig proc near
0040A315
0040A315 arg_0= dword ptr  8
0040A315 arg_4= dword ptr  0Ch
0040A315 arg_8= dword ptr  10h
0040A315 arg_C= dword ptr  14h
0040A315
0040A315 push    ebp
0040A316 mov     ebp, esp
0040A318 push    esi
0040A319 push    [ebp+arg_C]
0040A31C push    [ebp+arg_0]
0040A31F call    RC4_KSA
0040A324 xor     esi, esi
0040A326 cmp     [ebp+arg_8], esi
0040A329 jle     short loc_40A33E
```

```
0040A32B push    edi
0040A32C mov     edi, [ebp+arg_4]
```

```
0040A32F
0040A32F loc_40A32F:
0040A32F call    RC4_PRGA
0040A334 xor     [esi+edi], al
0040A337 inc     esi
0040A338 cmp     esi, [ebp+arg_8]
0040A33B jl      short loc_40A32F
```

```
0040A33D pop     edi
```

```
0040A33E
0040A33E loc_40A33E:
0040A33E pop     esi
0040A33F pop     ebp
0040A340 retn    10h
0040A340 GlobeDecryptConfig endp
0040A340
```

```
0040283D
0040283D loc_40283D:                  ; GlobeImposter Ransomware
0040283D mov     ebx, [edi+1Ch]       ; AES Key Expansion
00402840 movzx   eax, bl
00402843 mov     edx, [edi+0Ch]
00402846 movzx   ecx, Sbox[eax]
0040284D movzx   eax, byte ptr [edi+1Fh]
00402851 shl     ecx, 8
00402854 movzx   eax, Sbox[eax]
0040285B xor     ecx, eax
0040285D movzx   eax, byte ptr [edi+1Eh]
00402861 shl     ecx, 8
00402864 movzx   eax, Sbox[eax]
0040286B xor     ecx, eax
0040286D movzx   eax, byte ptr [edi+1Dh]
00402871 shl     ecx, 8
00402874 movzx   eax, Sbox[eax]
0040287B xor     ecx, eax
0040287D mov     eax, [edi+4]
00402880 xor     ecx, ss:Rcon[ebp]
00402886 add     ebp, 4
00402889 xor     ecx, [edi]
0040288B lea     edi, [edi+20h]
```

The young researcher

Kateryna Vitiuk - a master student at NURE, Ukraine

- Studies Cyber Security at NURE
- Interested in anti-ransomware research
- Is developing a distributed ledger-based system for her graduation work.

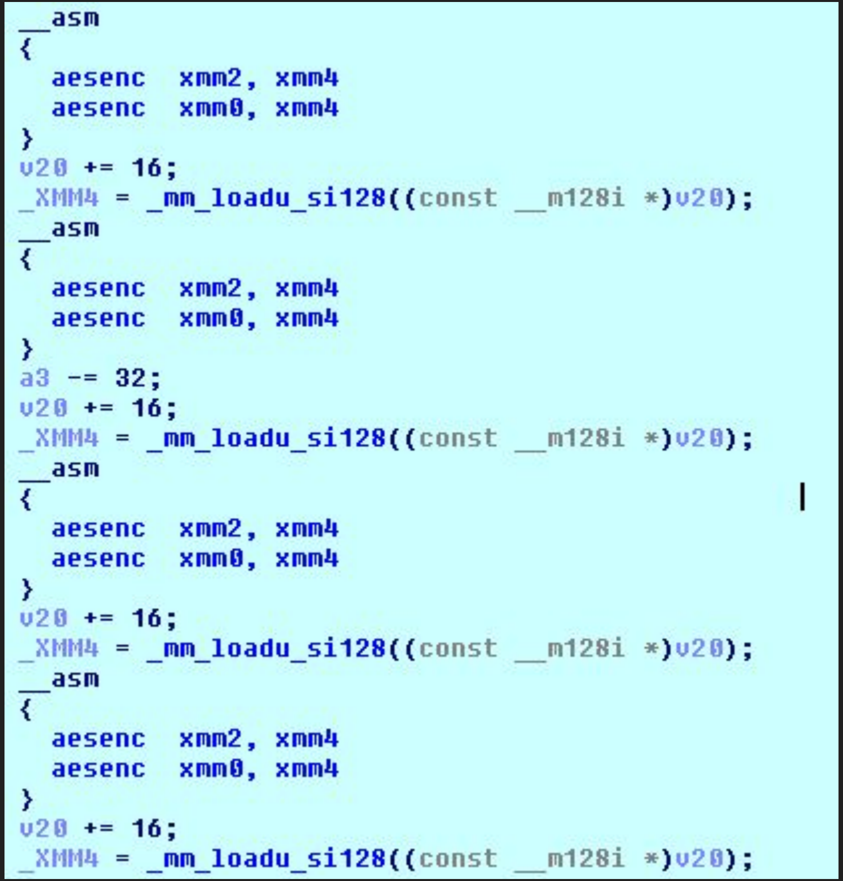
Scope

Ransomware with hardcoded ciphers

- AES-NI, XData
- Locky
- **TeslaCrypt**
- **GlobeImposter**
- **MoneroPay**
- GandCrab
- ...

Out of scope

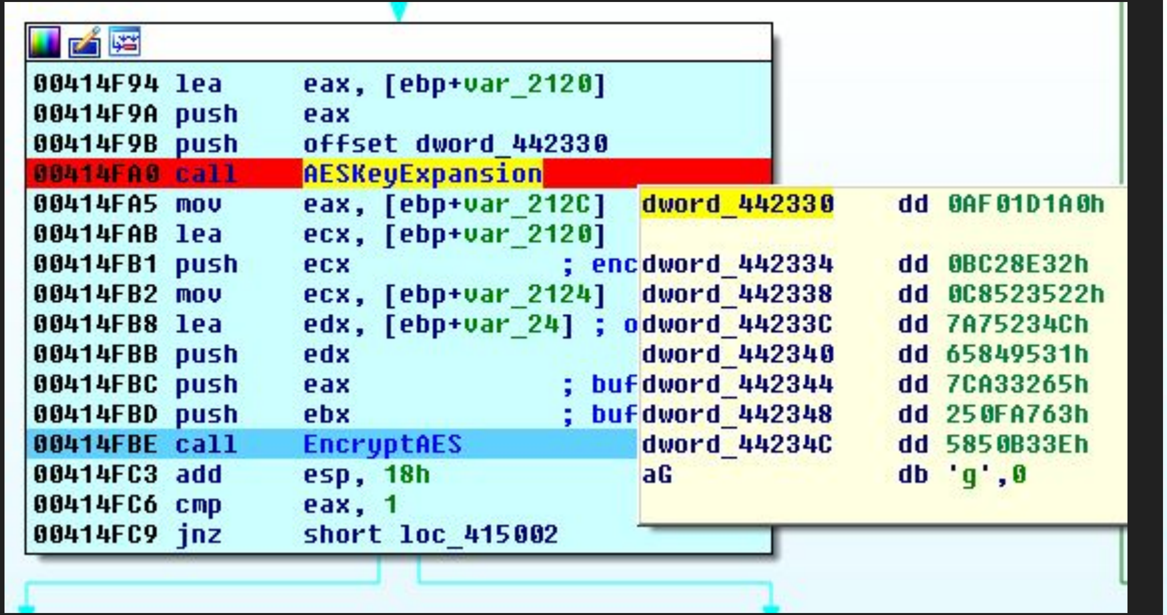
- AES-NI
- XData
- Locky



```
__asm
{
  aesenc  xmm2, xmm4
  aesenc  xmm0, xmm4
}
v20 += 16;
_XMM4 = _mm_loadu_si128((const __m128i *)v20);
__asm
{
  aesenc  xmm2, xmm4
  aesenc  xmm0, xmm4
}
a3 -= 32;
v20 += 16;
_XMM4 = _mm_loadu_si128((const __m128i *)v20);
__asm
{
  aesenc  xmm2, xmm4
  aesenc  xmm0, xmm4
}
v20 += 16;
_XMM4 = _mm_loadu_si128((const __m128i *)v20);
__asm
{
  aesenc  xmm2, xmm4
  aesenc  xmm0, xmm4
}
v20 += 16;
_XMM4 = _mm_loadu_si128((const __m128i *)v20);
```

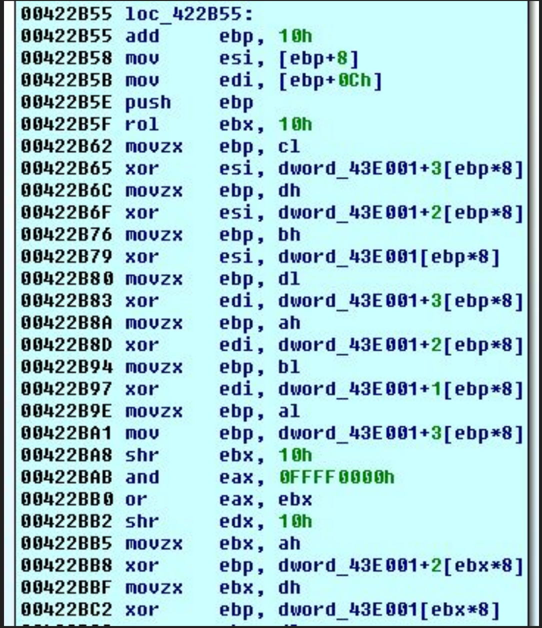
TeslaCrypt 2.1 - File encryption

Session AES-256-CBC key is generated and stored in the memory



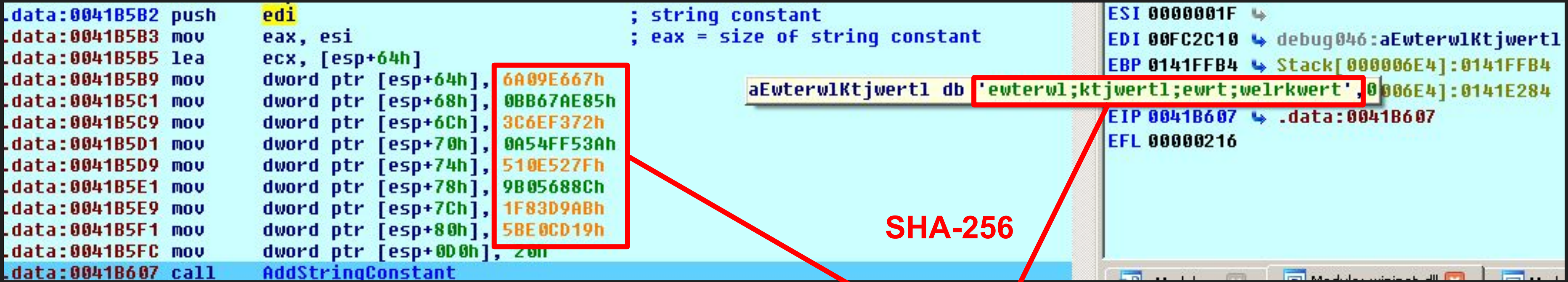
```
00414F94 lea     eax, [ebp+var_2120]
00414F9A push    eax
00414F9B push    offset dword_442330
00414FA0 call    AESKeyExpansion
00414FA5 mov     eax, [ebp+var_212C]
00414FAB lea     ecx, [ebp+var_2120]
00414FB1 push    ecx              ; enc
00414FB2 mov     ecx, [ebp+var_2124]
00414FB8 lea     edx, [ebp+var_24] ; o
00414FBB push    edx
00414FBC push    eax              ; buf
00414FBD push    ebx              ; buf
00414FBE call    EncryptAES
00414FC3 add     esp, 18h
00414FC6 cmp     eax, 1
00414FC9 jnz     short loc_415002
```

```
dword_442330    dd 0AF01D1A0h

dword_442334    dd 0BC28E32h
dword_442338    dd 0C8523522h
dword_44233C    dd 7A75234Ch
dword_442340    dd 65849531h
dword_442344    dd 7CA33265h
dword_442348    dd 250FA763h
dword_44234C    dd 5850B33Eh
aG              db 'g',0
```



```
00422B55 loc_422B55:
00422B55 add     ebp, 10h
00422B58 mov     esi, [ebp+8]
00422B5B mov     edi, [ebp+0Ch]
00422B5E push    ebp
00422B5F rol     ebx, 10h
00422B62 movzx   ebp, cl
00422B65 xor     esi, dword_43E001+3[ebp*8]
00422B6C movzx   ebp, dh
00422B6F xor     esi, dword_43E001+2[ebp*8]
00422B76 movzx   ebp, bh
00422B79 xor     esi, dword_43E001[ebp*8]
00422B80 movzx   ebp, dl
00422B83 xor     edi, dword_43E001+3[ebp*8]
00422B8A movzx   ebp, ah
00422B8D xor     edi, dword_43E001+2[ebp*8]
00422B94 movzx   ebp, bl
00422B97 xor     edi, dword_43E001+1[ebp*8]
00422B9E movzx   ebp, al
00422BA1 mov     ebp, dword_43E001+3[ebp*8]
00422BA8 shr     ebx, 10h
00422BAB and     eax, 0FFFF0000h
00422BB0 or      eax, ebx
00422BB2 shr     edx, 10h
00422BB5 movzx   ebx, ah
00422BB8 xor     ebp, dword_43E001+2[ebx*8]
00422BBF movzx   ebx, dh
00422BC2 xor     ebp, dword_43E001[ebx*8]
```

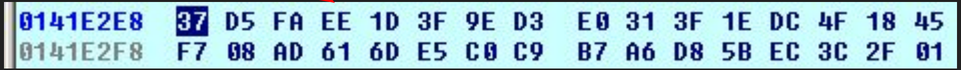
TeslaCrypt 2.1 - C&C traffic encryption



```
.data:0041B5B2 push    edi                          ; string constant
.data:0041B5B3 mov     eax, esi                     ; eax = size of string constant
.data:0041B5B5 lea     ecx, [esp+64h]
.data:0041B5B9 mov     dword ptr [esp+64h], 6A09E667h
.data:0041B5C1 mov     dword ptr [esp+68h], 0BB67AE85h
.data:0041B5C9 mov     dword ptr [esp+6Ch], 3C6EF372h
.data:0041B5D1 mov     dword ptr [esp+70h], 0A54FF53Ah
.data:0041B5D9 mov     dword ptr [esp+74h], 510E527Fh
.data:0041B5E1 mov     dword ptr [esp+78h], 9B05688Ch
.data:0041B5E9 mov     dword ptr [esp+7Ch], 1F83D9ABh
.data:0041B5F1 mov     dword ptr [esp+80h], 5BE0CD19h
.data:0041B5FC mov     dword ptr [esp+0D0h], 20h
.data:0041B607 call    AddStringConstant
```

```
ESI 0000001F ↳
EDI 00FC2C10 ↳ debug046:aEwterwlKtjwertl
EBP 0141FFB4 ↳ Stack[000006E4]:0141FFB4
aEwterwlKtjwertl db 'ewterwl;ktjwertl;ewrt;welrkwert',0 006E4]:0141E284
EIP 0041B607 ↳ .data:0041B607
EFL 00000216
```

Hardcoded AES-256-CBC key:

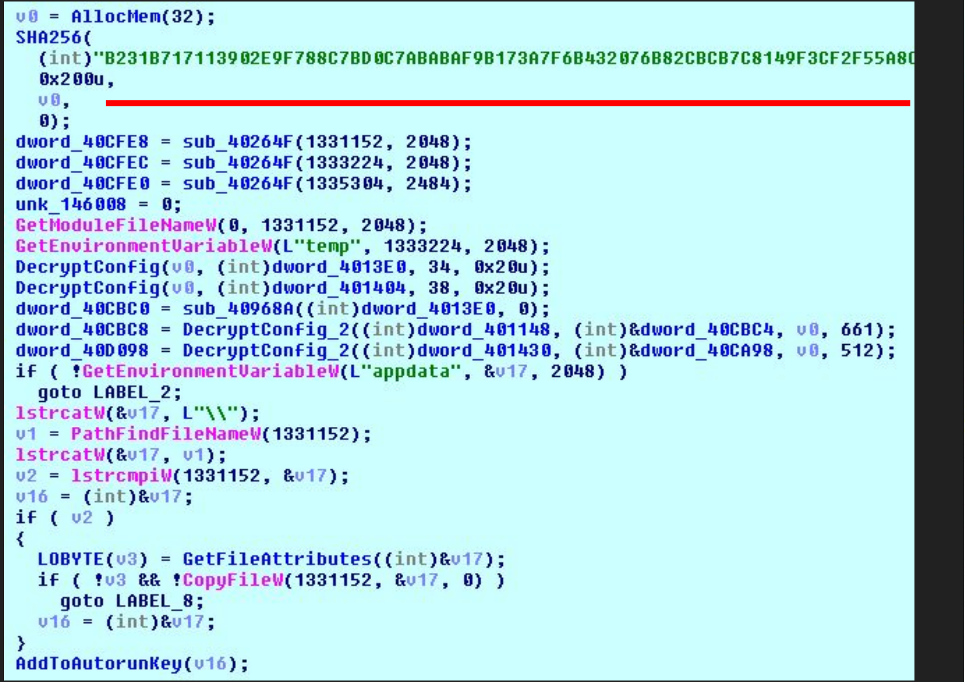


```
0141E2E8  37 D5 FA EE 1D 3F 9E D3  E0 31 3F 1E DC 4F 18 45
0141E2F8  F7 08 AD 61 6D E5 C0 C9  B7 A6 D8 5B EC 3C 2F 01
```

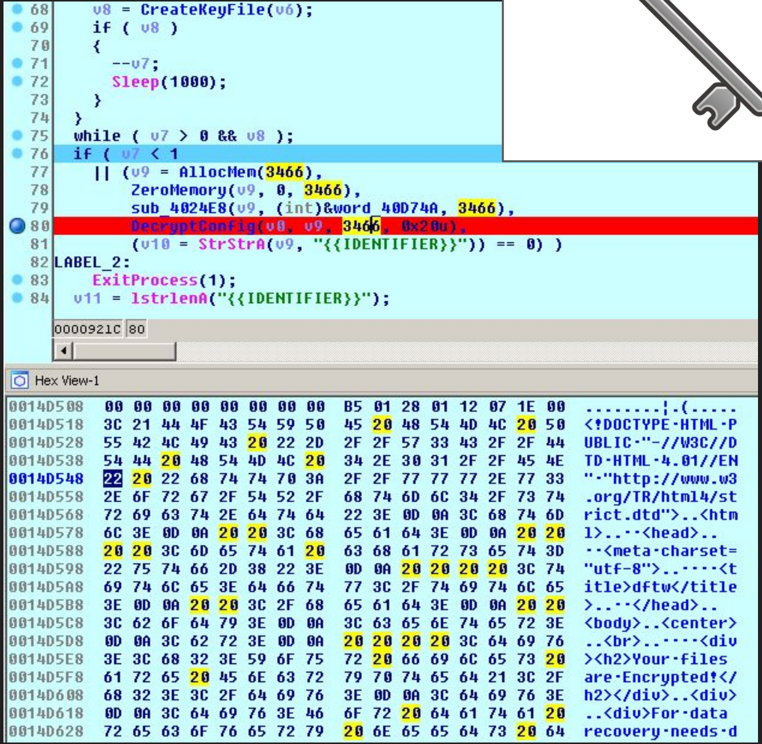
IV: DEADBEEF0000BEEFDEAD0000BEEFDEAD

GlobeImposter - Config extraction

RC4-256



```
v0 = AllocMem(32);
SHA256(
  (int)"B231B717113902E9F788C7BD0C7ABABAF9B173A7F6B432076B82CBCB7C8149F3CF2F55A8C
  0x200u,
  v0,
  0);
dword_40CFE8 = sub_40264F(1331152, 2048);
dword_40CFEC = sub_40264F(1333224, 2048);
dword_40CFE0 = sub_40264F(1335304, 2484);
unk_146008 = 0;
GetModuleFileNameW(0, 1331152, 2048);
GetEnvironmentVariableW(L"temp", 1333224, 2048);
DecryptConfig(v0, (int)dword_4013E0, 34, 0x20u);
DecryptConfig(v0, (int)dword_401404, 38, 0x20u);
dword_40CBC0 = sub_40968A((int)dword_4013E0, 0);
dword_40CBC8 = DecryptConfig_2((int)dword_401148, (int)&dword_40CBC4, v0, 661);
dword_40D098 = DecryptConfig_2((int)dword_401430, (int)&dword_40CA98, v0, 512);
if ( !GetEnvironmentVariableW(L"appdata", &v17, 2048) )
  goto LABEL_2;
lstrcatW(&v17, L"\\");
v1 = PathFindFileNameW(1331152);
lstrcatW(&v17, v1);
v2 = lstrcmpiW(1331152, &v17);
v16 = (int)&v17;
if ( v2 )
{
  LOBYTE(v3) = GetFileAttributes((int)&v17);
  if ( !v3 && !CopyFileW(1331152, &v17, 0) )
    goto LABEL_8;
  v16 = (int)&v17;
}
AddToAutorunKey(v16);
```



```
    v8 = CreateKeyFile(v6);
    if ( v8 )
    {
      --v7;
      Sleep(1000);
    }
  }
  while ( v7 > 0 && v8 );
  if ( v7 < 1
    || (v9 = AllocMem(3466),
        ZeroMemory(v9, 0, 3466),
        sub_4024E8(v9, (int)&word_40D74A, 3466),
        DecryptConfig(v0, v9, 3466, 0x20u),
        (v10 = StrStrA(v9, "{{IDENTIFIER}}")) == 0) )
LABEL_2:
    ExitProcess(1);
  v11 = lstrlenA("{{IDENTIFIER}}");
```

0000921C 80

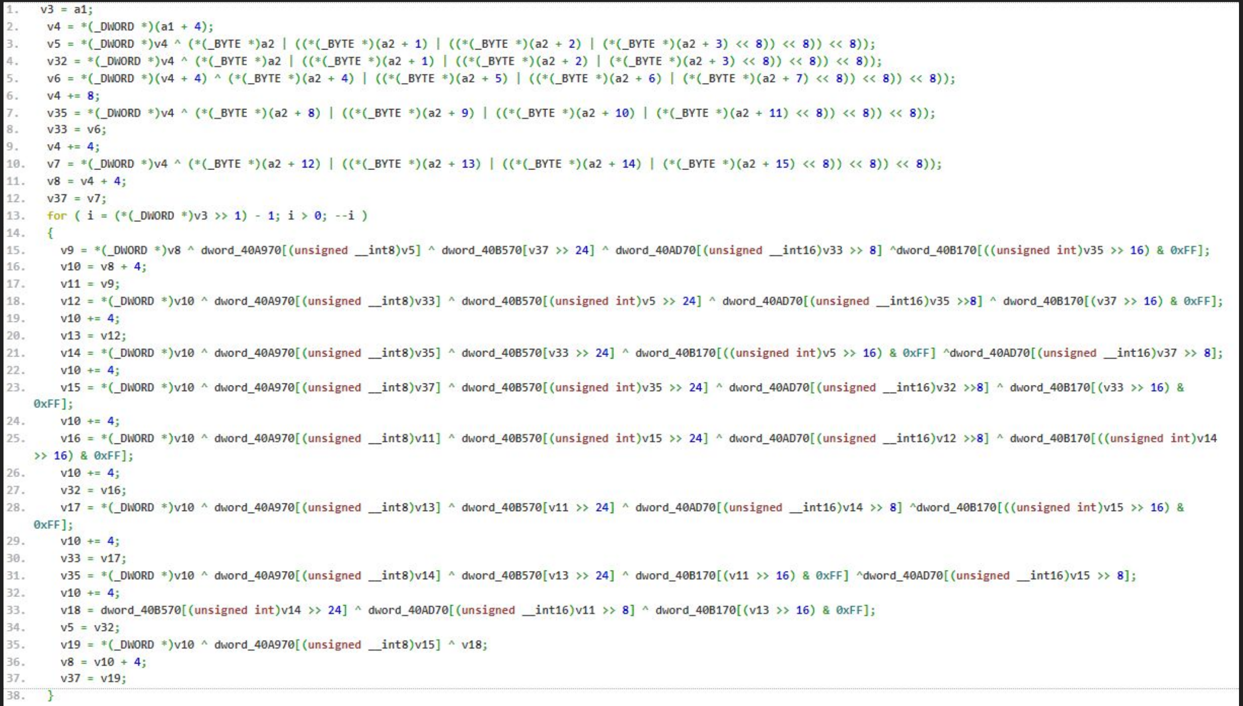
Hex View-1

```
0014D508  00 00 00 00 00 00 00 00  B5 01 28 01 12 07 1E 00  ........¦.(.....
0014D518  3C 21 44 4F 43 54 59 50  45 20 48 54 4D 4C 20 50  <!DOCTYPE·HTML·P
0014D528  55 42 4C 49 43 20 22 2D  2F 2F 57 33 43 2F 2F 44  UBLIC·"-//W3C//D
0014D538  54 44 20 48 54 4D 4C 20  34 2E 30 31 2F 2F 45 4E  TD·HTML·4.01//EN
0014D548  22 20 22 68 74 74 70 3A  2F 2F 77 77 77 2E 77 33  "·"http://www.w3
0014D558  2E 6F 72 67 2F 54 52 2F  68 74 6D 6C 34 2F 73 74  .org/TR/html4/st
0014D568  72 69 63 74 2E 64 74 64  22 3E 0D 0A 3C 68 74 6D  rict.dtd">..<htm
0014D578  6C 3E 0D 0A 20 20 3C 68  65 61 64 3E 0D 0A 20 20  l>..··<head>..
0014D588  20 20 3C 6D 65 74 61 20  63 68 61 72 73 65 74 3D  ··<meta·charset=
0014D598  22 75 74 66 2D 38 22 3E  0D 0A 20 20 20 20 3C 74  "utf-8">..····<t
0014D5A8  69 74 6C 65 3E 64 66 74  77 3C 2F 74 69 74 6C 65  itle>dftw</title
0014D5B8  3E 0D 0A 20 20 3C 2F 68  65 61 64 3E 0D 0A 20 20  >..··</head>..
0014D5C8  3C 62 6F 64 79 3E 0D 0A  3C 63 65 6E 74 65 72 3E  <body>..<center>
0014D5D8  0D 0A 3C 62 72 3E 0D 0A  20 20 20 20 3C 64 69 76  ..<br>..····<div
0014D5E8  3E 3C 68 32 3E 59 6F 75  72 20 66 69 6C 65 73 20  ><h2>Your·files
0014D5F8  61 72 65 20 45 6E 63 72  79 70 74 65 64 21 3C 2F  are·Encrypted!</
0014D608  68 32 3E 3C 2F 64 69 76  3E 0D 0A 3C 64 69 76 3E  h2></div>..<div>
0014D618  0D 0A 3C 64 69 76 3E 46  6F 72 20 64 61 74 61 20  ..<div>For·data
0014D628  72 65 63 6F 76 65 72 79  20 6E 65 65 64 73 20 64  recovery·needs·d
```

GlobeImposter - File encryption

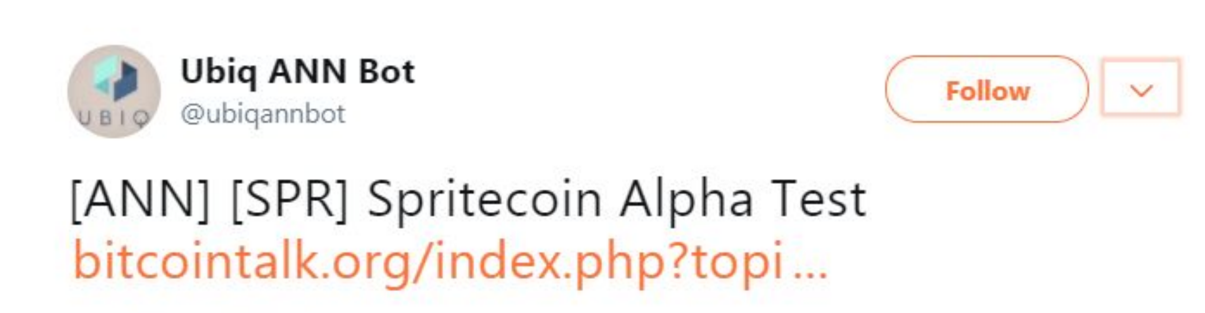
Generated AES-256 file keys using SHA-256

IV = SHA256 (File size & 8000000Fh4)



```
v3 = a1;
 v4 = *(_DWORD *)(a1 + 4);
 v5 = *(_DWORD *)v4 ^ (*(_BYTE *)a2 | ((*(_BYTE *)(a2 + 1) | ((*(_BYTE *)(a2 + 2) | (*(_BYTE *)(a2 + 3) << 8)) << 8)) << 8));
 v32 = *(_DWORD *)v4 ^ (*(_BYTE *)a2 | ((*(_BYTE *)(a2 + 1) | ((*(_BYTE *)(a2 + 2) | (*(_BYTE *)(a2 + 3) << 8)) << 8)) << 8));
 v6 = *(_DWORD *)(v4 + 4) ^ (*(_BYTE *)(a2 + 4) | ((*(_BYTE *)(a2 + 5) | ((*(_BYTE *)(a2 + 6) | (*(_BYTE *)(a2 + 7) << 8)) << 8)) << 8));
 v4 += 8;
 v35 = *(_DWORD *)v4 ^ (*(_BYTE *)(a2 + 8) | ((*(_BYTE *)(a2 + 9) | ((*(_BYTE *)(a2 + 10) | (*(_BYTE *)(a2 + 11) << 8)) << 8)) << 8));
 v33 = v6;
 v4 += 4;
 v7 = *(_DWORD *)v4 ^ (*(_BYTE *)(a2 + 12) | ((*(_BYTE *)(a2 + 13) | ((*(_BYTE *)(a2 + 14) | (*(_BYTE *)(a2 + 15) << 8)) << 8)) << 8));
 v8 = v4 + 4;
 v37 = v7;
 for ( i = (*(_DWORD *)v3 >> 1) - 1; i > 0; --i )
 {
   v9 = *(_DWORD *)v8 ^ dword_40A970[(unsigned __int8)v5] ^ dword_40B570[v37 >> 24] ^ dword_40AD70[(unsigned __int16)v33 >> 8] ^dword_40B170[((unsigned int)v35 >> 16) & 0xFF];
   v10 = v8 + 4;
   v11 = v9;
   v12 = *(_DWORD *)v10 ^ dword_40A970[(unsigned __int8)v33] ^ dword_40B570[(unsigned int)v5 >> 24] ^ dword_40AD70[(unsigned __int16)v35 >>8] ^ dword_40B170[(v37 >> 16) & 0xFF];
   v10 += 4;
   v13 = v12;
   v14 = *(_DWORD *)v10 ^ dword_40A970[(unsigned __int8)v35] ^ dword_40B570[v33 >> 24] ^ dword_40B170[((unsigned int)v5 >> 16) & 0xFF] ^dword_40AD70[(unsigned __int16)v37 >> 8];
   v10 += 4;
   v15 = *(_DWORD *)v10 ^ dword_40A970[(unsigned __int8)v37] ^ dword_40B570[(unsigned int)v35 >> 24] ^ dword_40AD70[(unsigned __int16)v32 >>8] ^ dword_40B170[(v33 >> 16) & 0xFF];
   v10 += 4;
   v16 = *(_DWORD *)v10 ^ dword_40A970[(unsigned __int8)v11] ^ dword_40B570[(unsigned int)v15 >> 24] ^ dword_40AD70[(unsigned __int16)v12 >>8] ^ dword_40B170[((unsigned int)v14 >> 16) & 0xFF];
   v10 += 4;
   v32 = v16;
   v17 = *(_DWORD *)v10 ^ dword_40A970[(unsigned __int8)v13] ^ dword_40B570[v11 >> 24] ^ dword_40AD70[(unsigned __int16)v14 >> 8] ^dword_40B170[((unsigned int)v15 >> 16) & 0xFF];
   v10 += 4;
   v33 = v17;
   v35 = *(_DWORD *)v10 ^ dword_40A970[(unsigned __int8)v14] ^ dword_40B570[v13 >> 24] ^ dword_40B170[(v11 >> 16) & 0xFF] ^dword_40AD70[(unsigned __int16)v15 >> 8];
   v10 += 4;
   v18 = dword_40B570[(unsigned int)v14 >> 24] ^ dword_40AD70[(unsigned __int16)v11 >> 8] ^ dword_40B170[(v13 >> 16) & 0xFF];
   v5 = v32;
   v19 = *(_DWORD *)v10 ^ dword_40A970[(unsigned __int8)v15] ^ v18;
   v8 = v10 + 4;
   v37 = v19;
 }
```

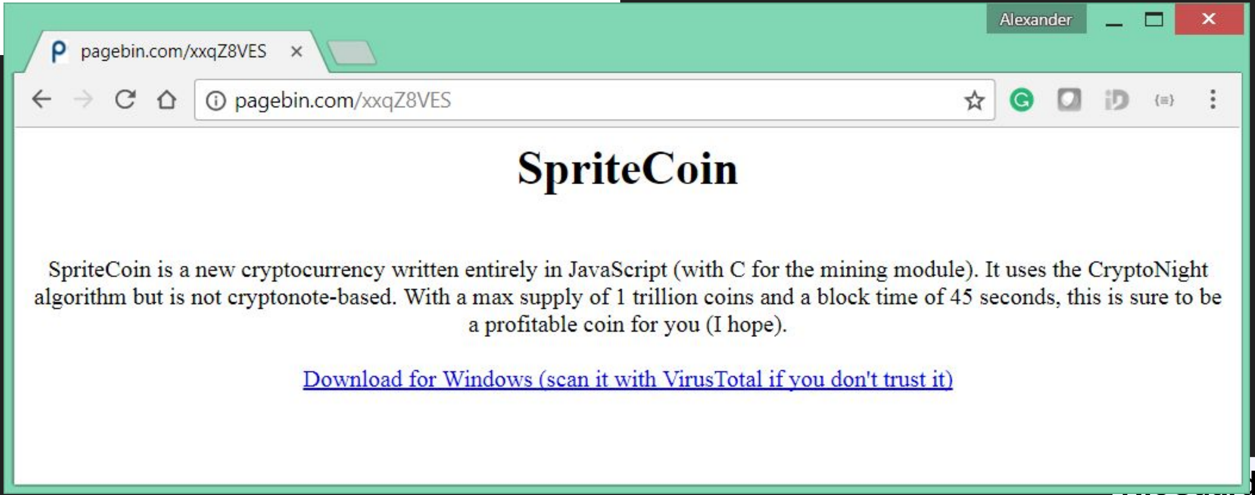
MoneroPay (SpriteCoin)



Ubiq ANN Bot
@ubiqannbot

[ANN] [SPR] Spritecoin Alpha Test bitcointalk.org/index.php?topi…

9:35 PM - 6 Jan 2018



pagebin.com/xxqZ8VES

SpriteCoin

SpriteCoin is a new cryptocurrency written entirely in JavaScript (with C for the mining module). It uses the CryptoNight algorithm but is not cryptonote-based. With a max supply of 1 trillion coins and a block time of 45 seconds, this is sure to be a profitable coin for you (I hope).

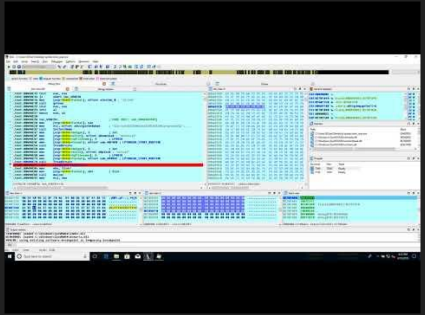
Download for Windows (scan it with VirusTotal if you don't trust it)

MoneroPay

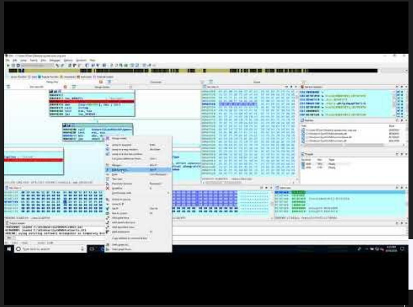
- A victim’s computer name (%COMPUTERNAME%)
- A user name (%USERNAME%)
- A user profile strings (%USERPROFILE%)
- C&C address: jmqapf3nflatei35.onion

→ Salsa20 session key

Encrypting:



Decrypting:



Signature-based detection

Ransomware	Symmetric cipher	Data source	Signature detection (Yara, KANAL PEiD)
GlobeImposter	AES-256-CBC; RC4, 16-byte key	PE file	List of primes, Big numbers, CryptGenKey import
		Memory dump	List of primes, Big numbers, CryptGenKey import, Rijndael_AES_CHAR, Rijndael_AES_LONG
TeslaCrypt	AES-256-CBC	PE file	N/A
		Memory dump	CryptGenKey import, Big numbers
MoneroPay	Salsa20, 32-byte key	PE file	N/A
		Memory dump	N/A

The proposed method

1. Obtaining patterns of the ciphers in ASM
2. Code normalization
3. Matching the crypto pattern in ransomware using the Bitap algorithm
   - diff_match_patch.match_main(code, pattern, expected location)
   - diff_match_patch.Match_Threshold = 0.5 (default)
   - diff_match_patch.Match_Distance = 1000 characters (default)
4. Obtaining diffs vectors using the Myer's algorithm for the matched patterns
5. Calculating the Levenshtein distance for diffs vectors
6. Comparing the found Levenshtein distances with the matching threshold
7. If the code is matched, add it to the library of the crypto patterns

Crypto patterns generation problem

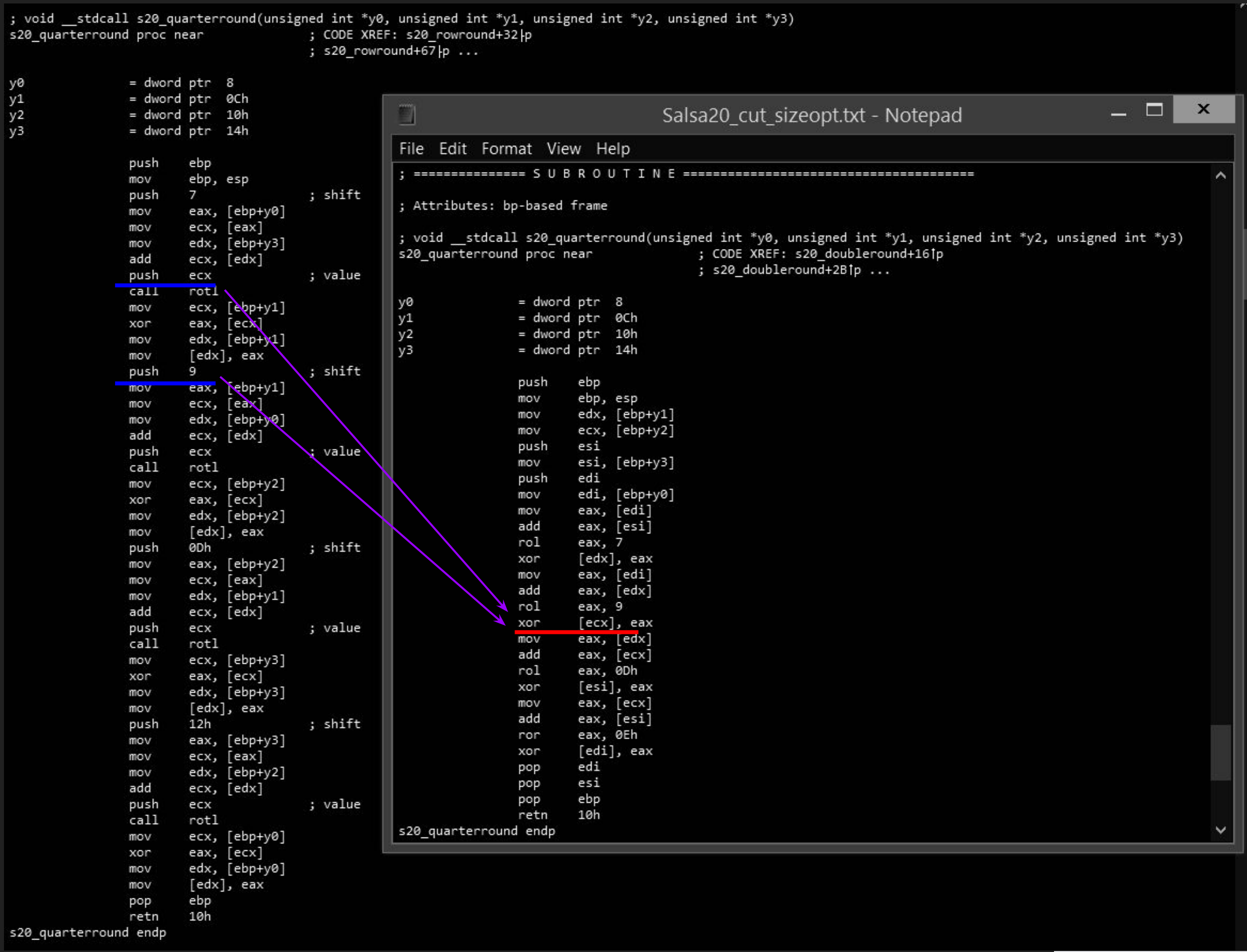
Different compiler options:

- Optimization
    - \O1 - Minimize size
    - \O2 - Maximize speed
    - \Ox - Full optimization
- Security check (/GS-)
- Calling convention
    - _stdcall (/Gz)
    - _cdecl (/Gd)
    - _fastcall (/Gr)
    - _vectorcall (/Gv)
- Platform (x86/x64)

Size does matter

Salsa20 QR

No opt vs. Minimize size (O1)



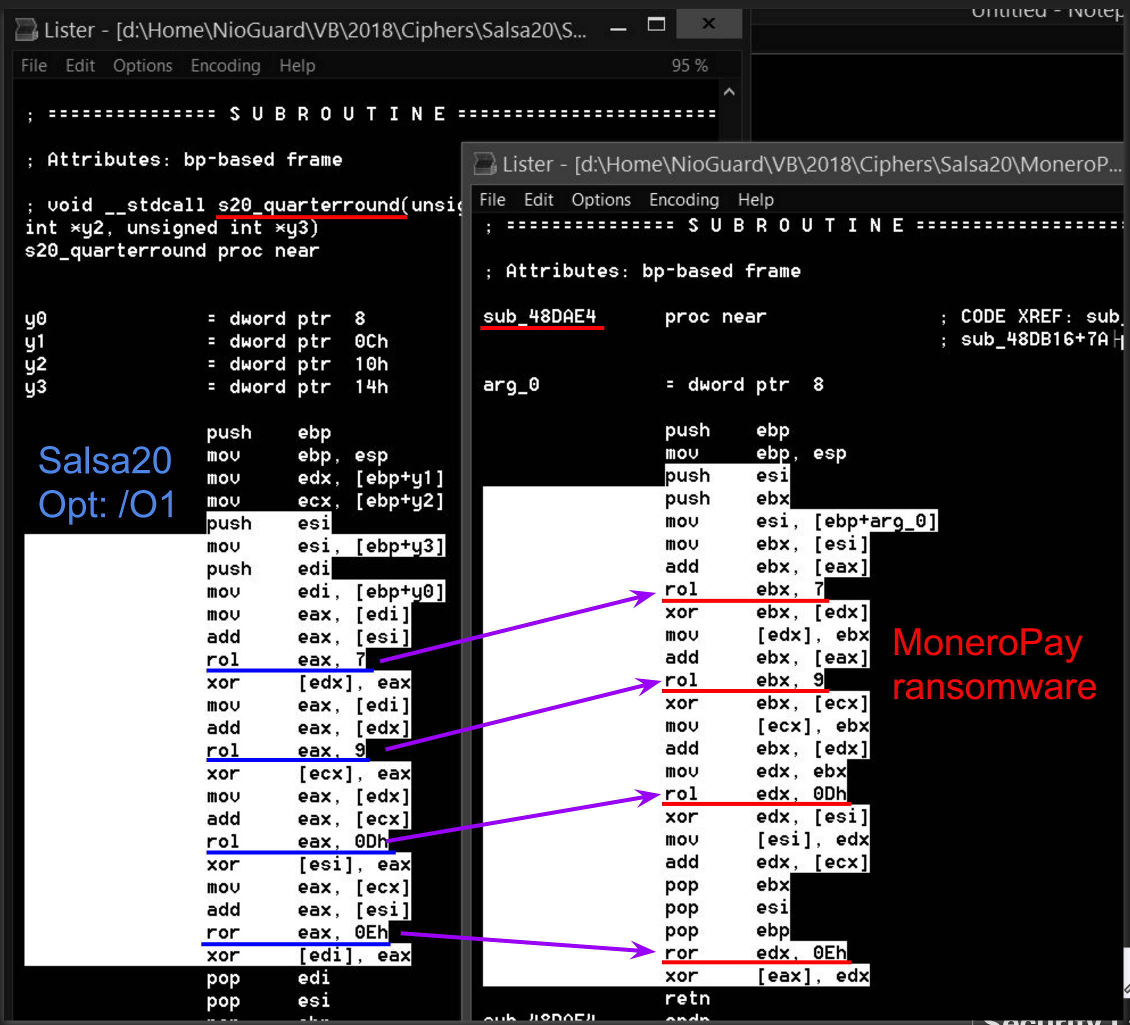
```
; void __stdcall s20_quarterround(unsigned int *y0, unsigned int *y1, unsigned int *y2, unsigned int *y3)
s20_quarterround proc near              ; CODE XREF: s20_rowround+32↓p
                                        ; s20_rowround+67↓p ...

y0              = dword ptr  8
y1              = dword ptr  0Ch
y2              = dword ptr  10h
y3              = dword ptr  14h

                push    ebp
                mov     ebp, esp
                push    7               ; shift
                mov     eax, [ebp+y0]
                mov     ecx, [eax]
                mov     edx, [ebp+y3]
                add     ecx, [edx]
                push    ecx             ; value
                call    rotl
                mov     ecx, [ebp+y1]
                xor     eax, [ecx]
                mov     edx, [ebp+y1]
                mov     [edx], eax
                push    9               ; shift
                mov     eax, [ebp+y1]
                mov     ecx, [eax]
                mov     edx, [ebp+y0]
                add     ecx, [edx]
                push    ecx             ; value
                call    rotl
                mov     ecx, [ebp+y2]
                xor     eax, [ecx]
                mov     edx, [ebp+y2]
                mov     [edx], eax
                push    0Dh             ; shift
                mov     eax, [ebp+y2]
                mov     ecx, [eax]
                mov     edx, [ebp+y1]
                add     ecx, [edx]
                push    ecx             ; value
                call    rotl
                mov     ecx, [ebp+y3]
                xor     eax, [ecx]
                mov     edx, [ebp+y3]
                mov     [edx], eax
                push    12h             ; shift
                mov     eax, [ebp+y3]
                mov     ecx, [eax]
                mov     edx, [ebp+y2]
                add     ecx, [edx]
                push    ecx             ; value
                call    rotl
                mov     ecx, [ebp+y0]
                xor     eax, [ecx]
                mov     edx, [ebp+y0]
                mov     [edx], eax
                pop     ebp
                retn    10h
s20_quarterround endp
```

```
; =============== S U B R O U T I N E =======================================

; Attributes: bp-based frame

; void __stdcall s20_quarterround(unsigned int *y0, unsigned int *y1, unsigned int *y2, unsigned int *y3)
s20_quarterround proc near              ; CODE XREF: s20_doubleround+16↑p
                                        ; s20_doubleround+2B↑p ...

y0              = dword ptr  8
y1              = dword ptr  0Ch
y2              = dword ptr  10h
y3              = dword ptr  14h

                push    ebp
                mov     ebp, esp
                mov     edx, [ebp+y1]
                mov     ecx, [ebp+y2]
                push    esi
                mov     esi, [ebp+y3]
                push    edi
                mov     edi, [ebp+y0]
                mov     eax, [edi]
                add     eax, [esi]
                rol     eax, 7
                xor     [edx], eax
                mov     eax, [edi]
                add     eax, [edx]
                rol     eax, 9
                xor     [ecx], eax
                mov     eax, [edx]
                add     eax, [ecx]
                rol     eax, 0Dh
                xor     [esi], eax
                mov     eax, [ecx]
                add     eax, [esi]
                ror     eax, 0Eh
                xor     [edi], eax
                pop     edi
                pop     esi
                pop     ebp
                retn    10h
s20_quarterround endp
```

Crypto patterns

Salsa20 QuarterRound crypto block in MoneroPay ransomware

‘rol eax, 7’ != ‘rol ebx, 7’


Lister - [d:\Home\NioGuard\VB\2018\Ciphers\Salsa20\S...
File Edit Options Encoding Help
95 %
Untitled - Notep
Lister - [d:\Home\NioGuard\VB\2018\Ciphers\Salsa20\MoneroP...
File Edit Options Encoding Help

```
; =============== S U B R O U T I N E =======================

; Attributes: bp-based frame

; void __stdcall s20_quarterround(unsig
int *y2, unsigned int *y3)
s20_quarterround proc near

y0              = dword ptr  8
y1              = dword ptr  0Ch
y2              = dword ptr  10h
y3              = dword ptr  14h

                push    ebp
                mov     ebp, esp
                mov     edx, [ebp+y1]
                mov     ecx, [ebp+y2]
                push    esi
                mov     esi, [ebp+y3]
                push    edi
                mov     edi, [ebp+y0]
                mov     eax, [edi]
                add     eax, [esi]
                rol     eax, 7
                xor     [edx], eax
                mov     eax, [edi]
                add     eax, [edx]
                rol     eax, 9
                xor     [ecx], eax
                mov     eax, [edx]
                add     eax, [ecx]
                rol     eax, 0Dh
                xor     [esi], eax
                mov     eax, [ecx]
                add     eax, [esi]
                ror     eax, 0Eh
                xor     [edi], eax
                pop     edi
                pop     esi
```

Salsa20 Opt: /O1

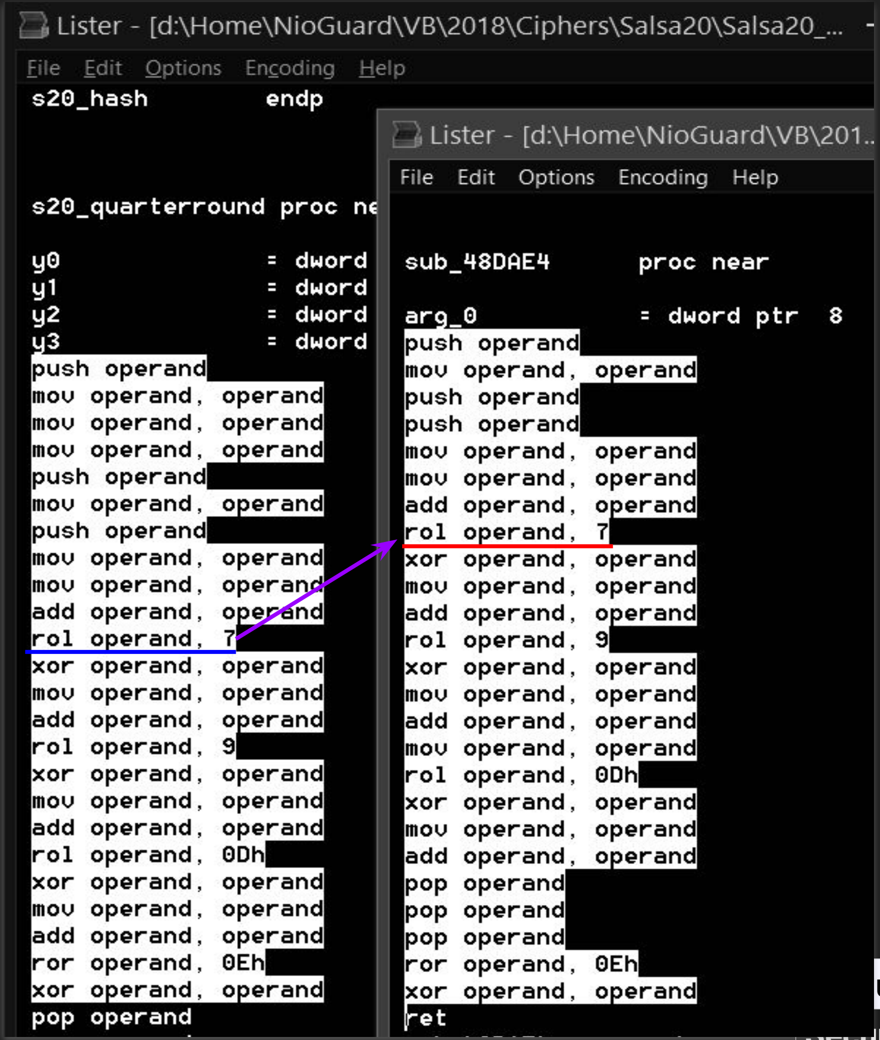
```
; =============== S U B R O U T I N E ==================

; Attributes: bp-based frame

sub_48DAE4      proc near               ; CODE XREF: sub_
                                        ; sub_48DB16+7A↑

arg_0           = dword ptr  8

                push    ebp
                mov     ebp, esp
                push    esi
                push    ebx
                mov     esi, [ebp+arg_0]
                mov     ebx, [esi]
                add     ebx, [eax]
                rol     ebx, 7
                xor     ebx, [edx]
                mov     [edx], ebx
                add     ebx, [eax]
                rol     ebx, 9
                xor     ebx, [ecx]
                mov     [ecx], ebx
                add     ebx, [edx]
                mov     edx, ebx
                rol     edx, 0Dh
                xor     edx, [esi]
                mov     [esi], edx
                add     edx, [ecx]
                pop     ebx
                pop     esi
                pop     ebp
                ror     edx, 0Eh
                xor     [eax], edx
                retn
```

MoneroPay ransomware

Security Lab

Normalization

Replace all CPU registers names with ‘operand’ string


Lister - [d:\Home\NioGuard\VB\2018\Ciphers\Salsa20\Salsa20_...
File
Edit
Options
Encoding
Help
Lister - [d:\Home\NioGuard\VB\201..
File
Edit
Options
Encoding
Help

```
s20_hash        endp


s20_quarterround proc ne

y0                = dword
y1                = dword
y2                = dword
y3                = dword
push operand
mov operand, operand
mov operand, operand
mov operand, operand
push operand
mov operand, operand
push operand
mov operand, operand
mov operand, operand
add operand, operand
rol operand, 7
xor operand, operand
mov operand, operand
add operand, operand
rol operand, 9
xor operand, operand
mov operand, operand
add operand, operand
rol operand, 0Dh
xor operand, operand
mov operand, operand
add operand, operand
ror operand, 0Eh
xor operand, operand
pop operand
```

```
sub_48DAE4      proc near

arg_0           = dword ptr  8
push operand
mov operand, operand
push operand
push operand
mov operand, operand
mov operand, operand
add operand, operand
rol operand, 7
xor operand, operand
mov operand, operand
add operand, operand
rol operand, 9
xor operand, operand
mov operand, operand
add operand, operand
mov operand, operand
rol operand, 0Dh
xor operand, operand
mov operand, operand
add operand, operand
pop operand
pop operand
pop operand
ror operand, 0Eh
xor operand, operand
ret
```

Myer’s diff algorithm

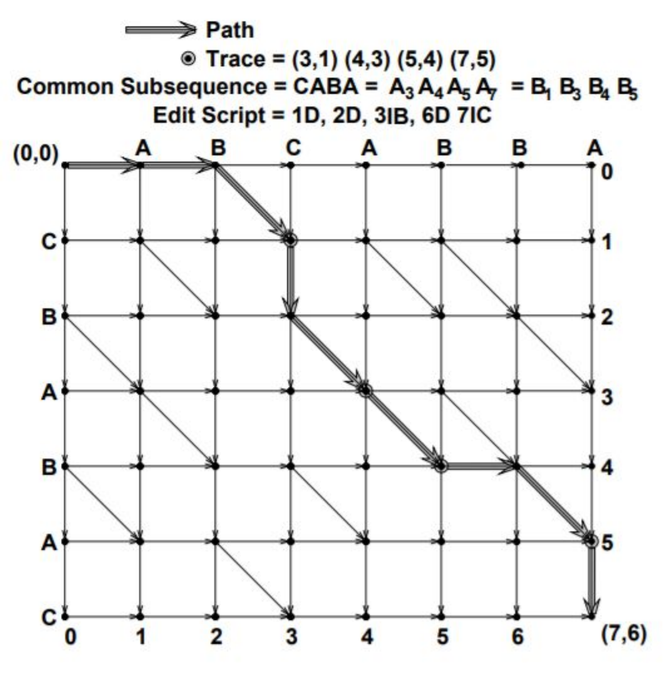
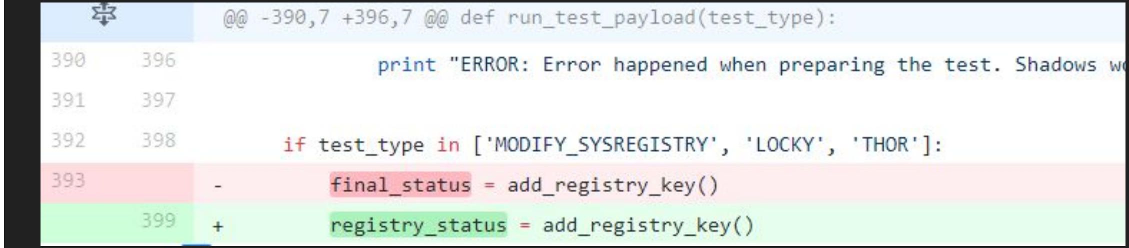

Path
Trace = (3,1) (4,3) (5,4) (7,5)
Common Subsequence = CABA = $A_3 A_4 A_5 A_7$ = $B_1 B_3 B_4 B_5$
Edit Script = 1D, 2D, 3IB, 6D 7IC
(0,0)
A
B
C
A
B
B
A
C
B
A
B
A
C
0
1
2
3
4
5
6
(7,6)

Fig. 1. An edit graph



```
@@ -390,7 +396,7 @@ def run_test_payload(test_type):
                print "ERROR: Error happened when preparing the test. Shadows wo

        if test_type in ['MODIFY_SYSREGISTRY', 'LOCKY', 'THOR']:
-           final_status = add_registry_key()
+           registry_status = add_registry_key()
```

Source: https://neil.fraser.name/writing/diff/myers.pdf

Diffs vectors & Levenshtein distance

```
(0, 'functionprocnearpushoperandmovoperand,operand')
(-1, 'movoperand,operandmovoperand,operandpushoperandmovoperand,')
(1, 'push')
(0,'operandpushoperandmovoperand,operandmovoperand,operandaddoperand,operandroloperand,7xoroperand,operandmovoperand
,operandaddoperand,operandroloperand,9xoroperand,operandmovoperand,operandaddoperand,')
(1, 'operandmovoperand,')
(0, 'operandroloperand,0Dhxoroperand,operandmovoperand,operandaddoperand,operand')
(-1, 'ror')
(1, 'pop')
(0, 'operand')
(-1, ',0Ehxoroperand,')
(1, 'pop')
(0, 'operandpopoperand')
(-1, 'pop')
(1, 'roroperand,0Ehxor')
(0, 'operand')
(-1, 'pop')
(1, ',')
(0, 'operandret')
(-1, 'n10h')
(0, 'functionendp')
```

Levenshtein distance: 118 characters

Results

Recognizing AES (key expansion) in the TeslaCrypt ransomware

Iteration No	1	2	3	4	5
Expected location	0	1500	3000	10000	20000
Matched location	115	1473	2986	10006	19953
Levenshtein distance	95	**60**	93	76	75
Correct match in ransomware	FALSE	**TRUE**	FALSE	FALSE	FALSE

Results

Recognizing AES (key expansion) in the GlobeImposter ransomware

Iteration No	1	2	3	4	5
Expected location	100	1000	4400	10000	20000
Matched location	399	999	4425	9968	19991
Levenshtein distance	61	113	**50**	132	91
Correct match in ransomware	FALSE	FALSE	**TRUE**	FALSE	FALSE

Results

Recognizing RC4 (PRGA) in the GlobeImposter ransomware

Iteration No	1	2	3	4	5
Expected location	0	500	800	1000	1500
Matched location	340	340	828	1063	1553
Levenshtein distance	**20**	**20**	76	75	83
Correct match in ransomware	**TRUE**	**TRUE**	FALSE	FALSE	FALSE

Results

Recognizing Salsa20 (quatterround) in the MoneroPay ransomware

Iteration No	1	2	3	4	5
Expected location	0	100	1000	1500	3000
Matched location	2	100	1000	1500	3094
Levenshtein distance	**118**	146	177	619	389
Correct match in ransomware	**TRUE**	FALSE	FALSE	FALSE	FALSE

Limitations

- Obfuscated code
- Packed code
- Differences in call trees (function hierarchy) require code roll out
    - [Workaround]: only small code patterns can be used
- The method strongly depends on the expected location of the crypto code

Conclusion

- It is possible to find the crypto primitives in ransomware with the given limitations.
- Master students can conduct research on malware and AI
- Using open source libraries prevents reinventing the wheel and boosts the research process

Acknowledgements

- Google
  - The Diff-Match-Patch libraries contributors
  - VirusTotal team
- Vlad Kolbasin, an AI/ML guru, GlobalLogic
- Dr. Anders Carlsson, General Manager of ENGENSEC project, BTH
- Prof. Vladimir Hahanov and Prof. Svetlana Chumachenko, NURE

References

@Alex_Ad

- Research results: https://github.com/AlexanderAda/NioGuardSecurityLab/tree/master/RansomwareAnalysis/DiffMatchPatterns
- The Google's Diff-Match-Patch libraries repository, https://github.com/google/diff-match-patch
- Crypto Yara rules:
  - https://github.com/Yara-Rules/rules/tree/ae82fb6e1e3145a85f52c4856985f7743796aae6/Crypto
  - https://github.com/x64dbg/yarasigs
  - https://github.com/polymorf/findcrypt-yara
- PEiD Tool, http://peid.has.it
- Ransomware samples
  - TeslaCrypt: 9e3827dffc24d1da72cb3d423bddf4cd535fa636062e4ea63421ef327fec56ad
  - GlobeImposter: a0e5bced56025f875721043df981c400fc28e4efc68ffe42ac665633de085ab1
  - MoneroPay: ababb37a65af7c8bde0167df101812ca96275c8bc367ee194c61ef3715228ddc