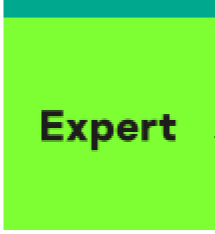
Sodin ransomware exploits Windows vulnerability and processor architecture

securelist.com/sodin-ransomware/91473/

Authors

- 

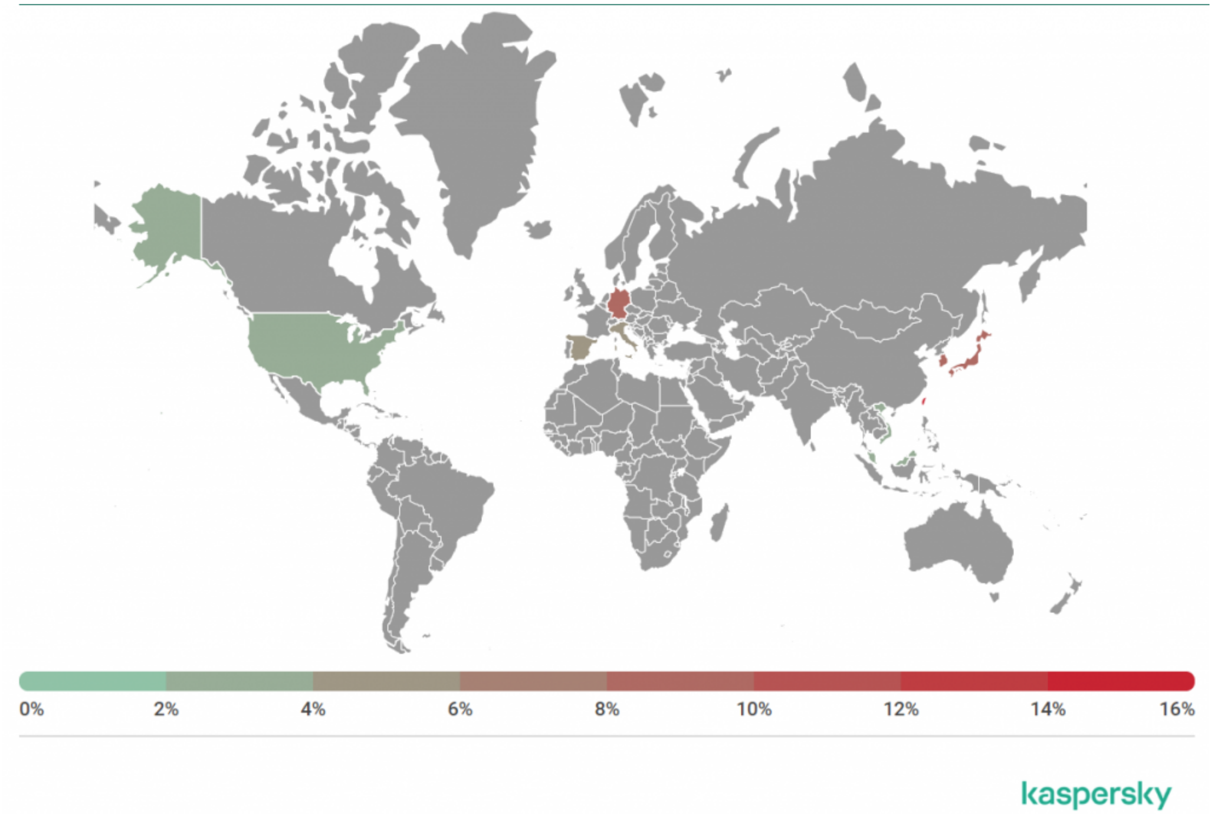
 Orkhan Mamedov
- 

 Artur Pakulov
- 

 Fedor Sinitsyn

When Sodin (also known as Sodinokibi and REvil) appeared in the first half of 2019, it immediately caught our attention for distributing itself through an Oracle Weblogic vulnerability and carrying out attacks on MSP providers. In a detailed analysis, we discovered that it also exploits the CVE-2018-8453 vulnerability to elevate privileges in Windows (rare among ransomware), and uses legitimate processor functions to circumvent security solutions.

According to our statistics, most victims were located in the Asia-Pacific region: Taiwan, Hong Kong, and South Korea.

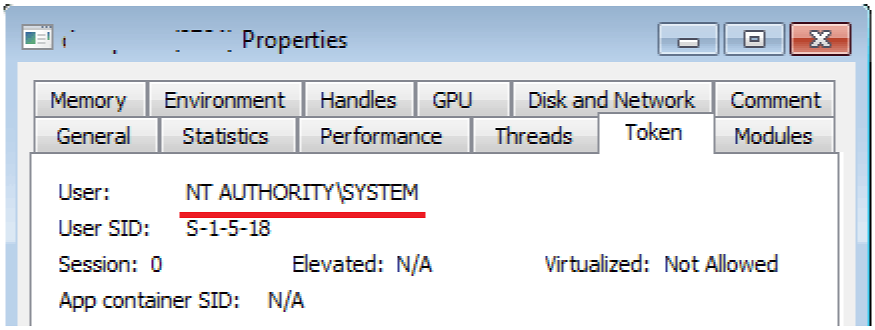


Geographic spread of Sodin ransomware, April – June 2019

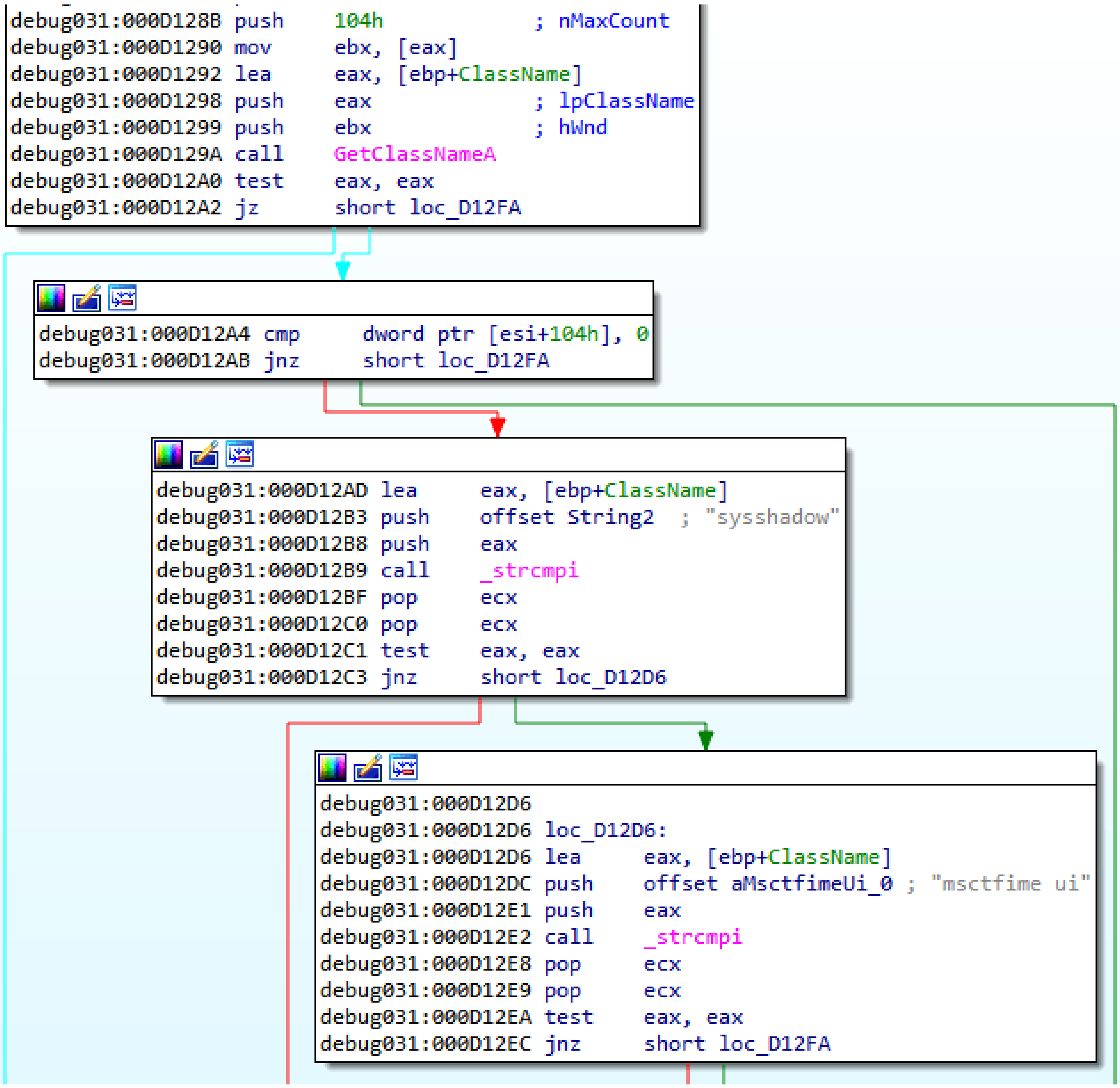
Technical description

Vulnerability exploitation

To escalate privileges, Trojan-Ransom.Win32.Sodin uses a vulnerability in win32k.sys; attempts to exploit it were first detected by our proactive technologies (Automatic Exploit Prevention, AEP) in August last year. The vulnerability was assigned the number CVE-2018-8453. After the exploit is executed, the Trojan acquires the highest level of privileges.

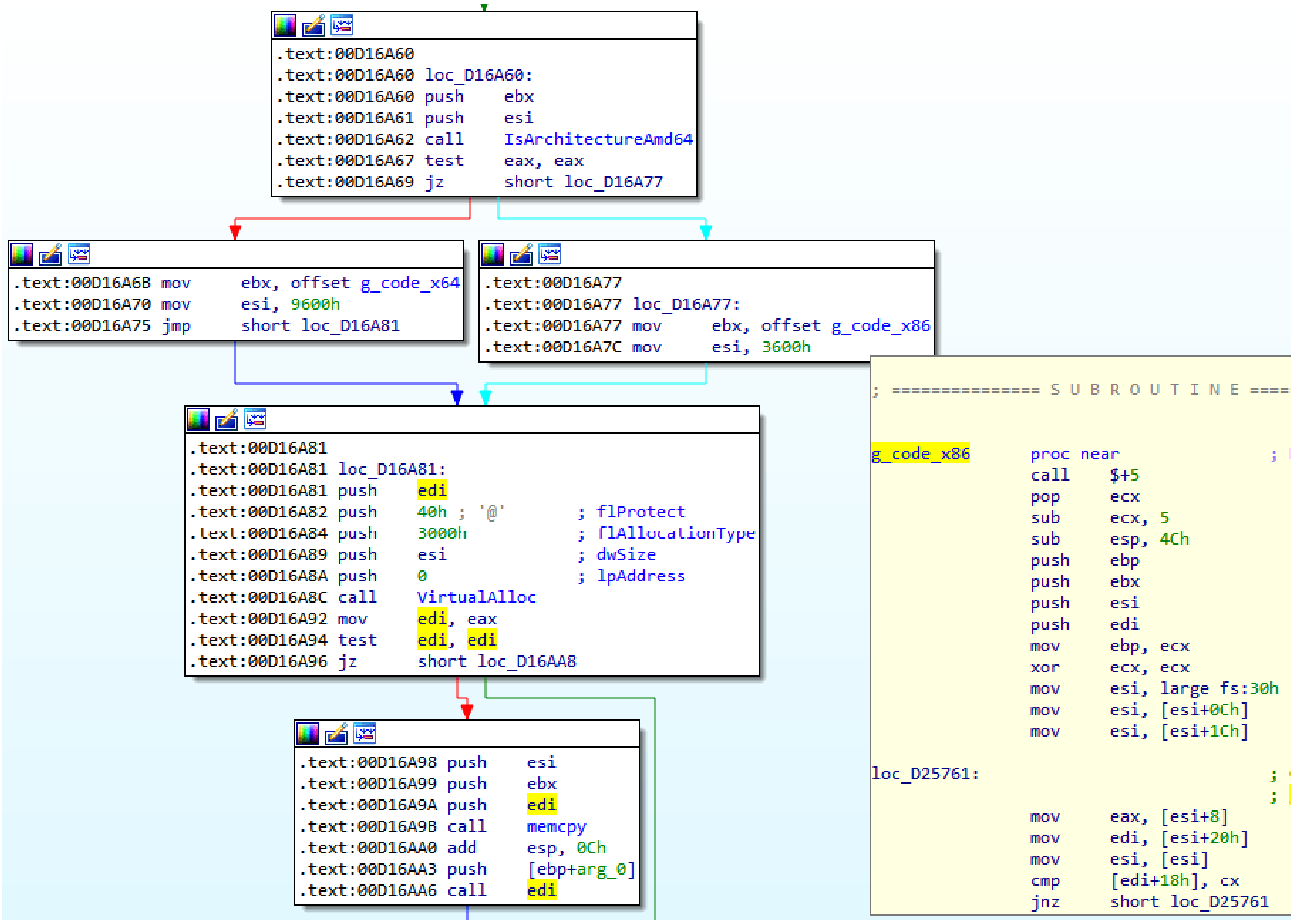


Information about the process token after exploit execution



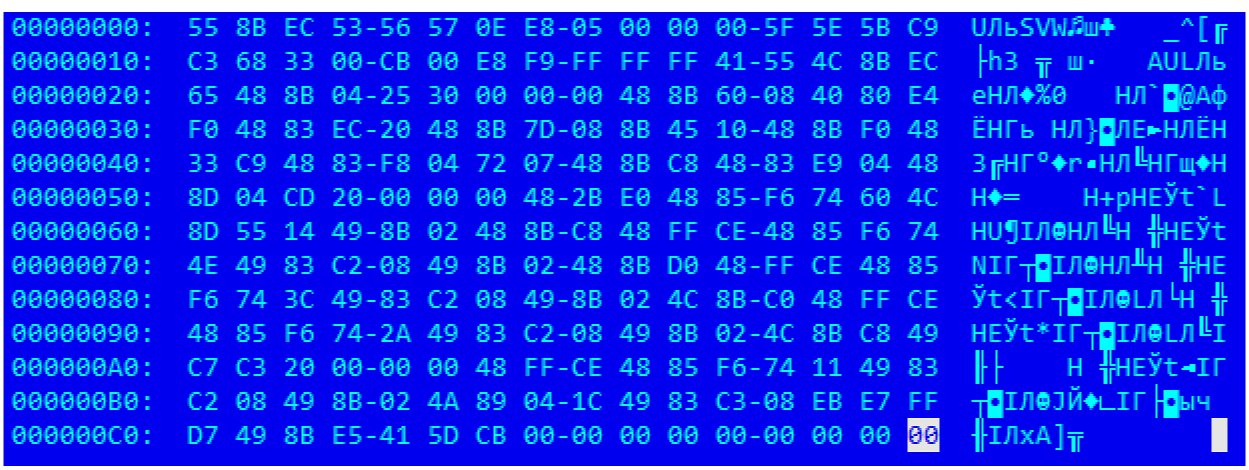
Exploit snippet for checking the window class

Depending on the processor architecture, one of two shellcode options contained in the Trojan body is run:



Procedure for selecting the appropriate shellcode option

Since the binary being analyzed is a 32-bit executable file, we are interested in how it manages to execute 64-bit code in its address space. The screenshot shows a shellcode snippet for executing 64-bit processor instructions:



Shellcode consisting of 32-bit and 64-bit instructions

In a 64-bit OS, the segment selector for 32-bit user mode code is 0x23, while the 64-bit segment selector is 0x33. This is confirmed by looking at the Global Descriptor Table (GDT) in the kernel debugger:

```
3: kd> dg 0 0x40
                                                    P Si Gr Pr Lo
Sel        Base              Limit          Type    l ze an es ng Flags
---- ----------------- ----------------- ---------- - -- -- -- -- --------
0000 00000000`00000000 00000000`00000000 <Reserved> 0 Nb By Np Nl 00000000
0008 00000000`00000000 00000000`00000000 <Reserved> 0 Nb By Np Nl 00000000
0010 00000000`00000000 00000000`00000000 Code RE Ac 0 Nb By P  Lo 0000029b
0018 00000000`00000000 00000000`00000000 Data RW Ac 0 Bg By P  Nl 00000493
0020 00000000`00000000 00000000`ffffffff Code RE Ac 3 Bg Pg P  Nl 00000cfb
0028 00000000`00000000 00000000`ffffffff Data RW Ac 3 Bg Pg P  Nl 00000cf3
0030 00000000`00000000 00000000`00000000 Code RE Ac 3 Nb By P  Lo 000002fb
0038 00000000`00000000 00000000`00000000 <Reserved> 0 Nb By Np Nl 00000000
0040 00000000`25412000 00000000`00000067 TSS32 Busy 0 Nb By P  Nl 0000008b
```

Part of the GDT in OS Windows 10 x64

The selector 0x23 points to the fourth segment descriptor (0x23 >> 3), and the selector 0x33 to the sixth (the null descriptor is not used). The **Nl** flag indicates that the segment uses 32-bit addressing, while the **Lo** flag specifies 64-bit. It is important that the base addresses of these segments are equal. At the time of shellcode execution, the selector 0x23 is located in the segment register **cs**, since the code is executed in a 32-bit address space. With this in mind, let's take a look at the listing of the very start of the shellcode:

```
00000000: 55                         push       ebp
00000001: 8BEC                       mov        ebp,esp
00000003: 53                         push       ebx
00000004: 56                         push       esi
00000005: 57                         push       edi
00000006: 0E                         push       cs
00000007: E805000000                 call       000000011 --↓1
0000000C: 5F                         pop        edi
0000000D: 5E                         pop        esi
0000000E: 5B                         pop        ebx
0000000F: C9                         leave
00000010: C3                         retn ; -^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-
```

Saving the full address 0x23:0xC

After executing the command for RVA addresses 6 and 7, the long return address is stored at the top of the stack in the format **selector:offset**, and takes the form 0x23:0x0C. In the stack at offset 0x11, a DWORD is placed whose low-order word contains the selector 0x33 and whose high-order word encodes the instruction **retf**, the opcode of which is equal to 0xCB.

```
00000011: 683300CB00                     push        000CB0033 ;' ╥ 3'
00000016: E8F9FFFFFF                     call        000000014 --↑1
0000001B: 41                             inc         ecx
0000001C: 55                             push        ebp
```

Saving the full address 0x33:0x1B to 64-bit code

```
00000014: CB                             retf ; -^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-
```

Switching to 64-bit mode

The next instruction **call** (at the address RVA 0x16) performs a near intrasegment jump to this **retf** instruction (RVA 0x14), having sent the short return address (offset 0x1b) to the stack. As such, at the time of execution of the **retf** instruction, the top of the stack contains the address in the format **selector:offset**, where the selector equals 0x33 and the offset is 0x1b. After executing the **retf** command, the processor proceeds to execute the code at this address, but now in 64-bit mode.

```
0000001B: 4155                           push        r13
0000001D: 4C8BEC                         mov         r13,rsp
00000020: 65488B042530000000             mov         rax,gs:[000000030]
00000029: 488B6008                       mov         rsp,[rax][8]
0000002D: 4080E4F0                       and         spl,0F0 ;'Ë'
00000031: 4883EC20                       sub         rsp,020 ;' '
00000035: 488B7D08                       mov         rdi,[rbp][8]
00000039: 8B4510                         mov         eax,[rbp][010]
0000003C: 488BF0                         mov         rsi,rax
0000003F: 4833C9                         xor         rcx,rcx
00000042: 4883F804                       cmp         rax,4
```

64-bit shellcode

The return to 32-bit mode is performed at the very end of the shellcode.

```
000000C1: 498BE5                         mov         rsp,r13
000000C4: 415D                           pop         r13
000000C6: CB                             retf ; -^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-
```

Returning to 32-bit mode

The **retf** command makes a far intrasegment jump to the address 0x23:0x0C (it was placed in the instruction stack at the very start of the shellcode, at the RVA address 6-7). This technique of executing 64-bit code in a 32-bit process address space is called Heaven's Gate, and was first described around ten years ago.

Trojan configuration

Stored in encrypted form in the body of each Sodin sample is a configuration block containing the settings and data required for the Trojan to work.


```
{
    "pk": "1g3/QEQPOQ7S3fBLZ0wvu/B9NfpLLvf8mByoN3or9E0=",
    "pid": "5",
    "sub": "367",
    "dbg": false,
    "fast": true,
    "wipe": true,
    "wht": {
        "fld": ["windows", "program files (x86)", "$recycle.bin", "programdata", "boot", "perflogs", "appdata", "mozilla", "pro
        "fls": ["ntuser.dat", "boot.ini", "autorun.inf", "ntuser.ini", "thumbs.db", "ntldr", "bootsect.bak", "ntuser.dat.log",
        "ext": ["icl", "nomedia", "msc", "ldf", "diagcab", "drv", "msp", "key", "wpx", "idx", "386", "lock", "rom", "icns", "ms
    },
    "wfld": ["backup"],
    "prc": ["wordpad.exe", "outlook.exe", "tbirdconfig.exe", "agntsvc.exe", "thebat.exe", "mydesktopservice.exe", "sqbcoreservi
    "dmn": "[redacted]
    "net": true,
    "nbody": "LQAtAC0APQA9AD0AIABXAGUAbABjAG8AbQBlAC4AIABBAGcAYQBpAG4ALgAgAD0APQA9AC0ALQAtAA0ACgANAAoAWwArAF0AIABXAGgAYQB0AHMAI
    "nname": "{EXT}-readme.txt",
    "exp": true,
    "img": "QQBsAGwAIABvAGYAIAB5AG8AdQByACAAZgBpAGwAZQBzACAAYQByAGUAIABlAG4AYwByAHkAcAB0AGUAZAAhAA0ACgANAAoARgBpAG4AZAAgAHsARQB
}
```


Decrypted Trojan configuration block

The Sodin configuration has the following fields:

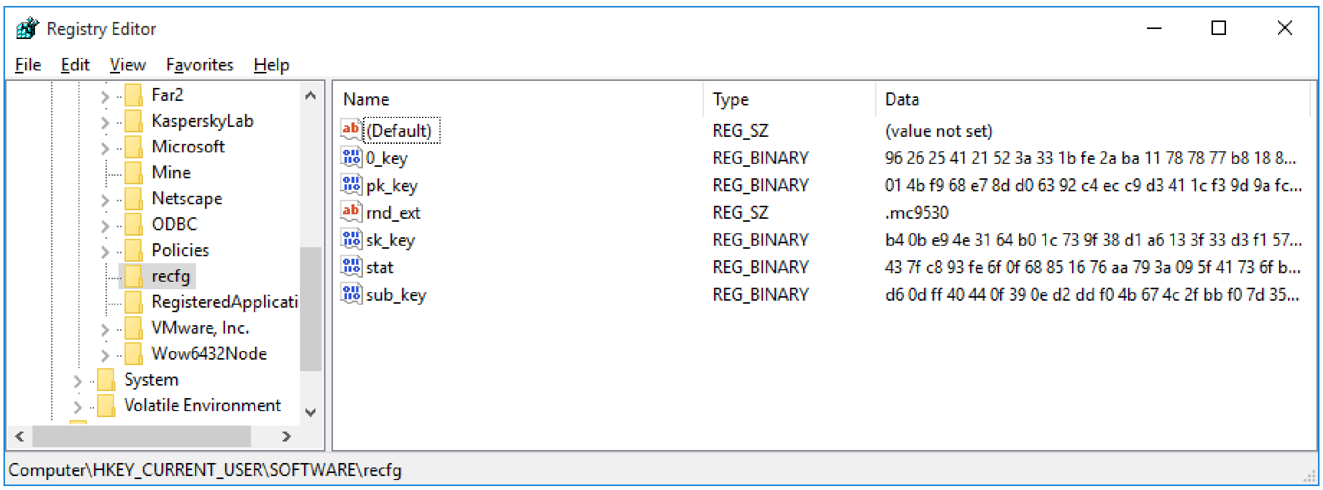
Field	Purpose
pk	distributor public key
pid	probably distributor id
sub	probably campaign id
dbg	debug build
fast	fast encryption mode (maximum 0x100000 bytes)
wipe	deletion of certain files and overwriting of their content with random bytes
wfld	names of directories in which the Trojan deletes files
wht	names of directories and files, and list of extensions not to be encrypted
prc	names of processes to be terminated
dmn	server addresses for sending statistics
net	sending infection statistics
nbody	ransom note template
nname	ransom note file name template
exp	use of exploit for privilege escalation

img	text for desktop wallpaper

Cryptographic scheme

Sodin uses a hybrid scheme to encrypt victim files. The file contents are encrypted with the Salsa20 symmetric stream algorithm, and the keys for it with an elliptic curve asymmetric algorithm. Let’s take a closer look at the scheme.

Since some data is stored in the registry, this article uses the names given by the ransomware itself. For entities not in the registry, we use invented names.



Data saved by the Trojan in the registry

Key generation

The Sodin configuration block contains the **pk** field, which is saved in the registry under the name **sub_key** – this is the 32-byte public key of the Trojan distributor. The key is a point on the Curve25519 elliptic curve.

When launched, the Trojan generates a new pair of elliptic curve session keys; the public key of this pair is saved in the registry under the name **pk_key**, while the private key is encrypted using the ECIES algorithm with the **sub_key** key and stored in the registry under the name **sk_key**. The ECIES implementation in this case includes the Curve25519 elliptic curve, the SHA3-256 cryptographic hash, and the AES-256 block cipher in CFB mode. Other ECIES implementations have been encountered in Trojans before, for example, in SynAck targeted ransomware.

Curiously, the same private session key is also encrypted with another public key hardcoded into the body of the Trojan, regardless of the configuration. We will call it the **public skeleton key**. The encryption result is stored in the registry under the name **0_key**. It turns out that someone who knows the private key corresponding to the **public skeleton key** is able to decrypt the victim’s files, even without the private key for **sub_key**. It seems like the Trojan developers built a loophole into the algorithm allowing them to decrypt files behind the distributors’ back.

```
sk_key_data = RegQuery(HKEY_LOCAL_MACHINE, &software_recfg, &sk_key, &sk_key_type, &sk_key_size);
if ( !sk_key_data )
  sk_key_data = RegQuery(HKEY_CURRENT_USER, &software_recfg, &sk_key, &sk_key_type, &sk_key_size);
0_key_data = RegQuery(HKEY_LOCAL_MACHINE, &software_recfg, &0_key, &0_key_type, &0_key_size);
if ( !0_key_data )
  0_key_data = RegQuery(HKEY_CURRENT_USER, &software_recfg, &0_key, &0_key_type, &0_key_size);
if ( sub_key_data
  && sub_key_size == 32
  && sub_key_type == REG_BINARY
  && pk_key_data
  && pk_key_size == 32
  && pk_key_type == REG_BINARY
  && sk_key_data
  && sk_key_size == 88
  && sk_key_type == REG_BINARY
  && 0_key_data
  && 0_key_size == 88
  && 0_key_type == REG_BINARY )
{
  memcpy(g_sub_key, sub_key_data, 32);
  memcpy(g_pk_key, pk_key_data, 32);
  memcpy(g_sk_key, sk_key_data, 88);
  memcpy(g_0_key, 0_key_data, 88);
}
else
{
  curve25519_generate_keys(session_priv, g_pk_key);
  pk_key_size = 32;
  sub_key_size = 32;
  sk_key_data = ECIES_encrypt(g_sub_key, session_priv, 32, &sk_key_size);
  0_key_data = ECIES_encrypt(g_skeleton_pub_key, session_priv, 32, &0_key_size);
  zero_mem(session_priv, 32u);
  if ( !sk_key_data || !0_key_data )
    return 0;
  memcpy(g_sk_key, sk_key_data, sk_key_size);
  memcpy(g_0_key, 0_key_data, 0_key_size);
  if ( !RegSet(HKEY_LOCAL_MACHINE, &software_recfg, &sub_key, 3u, g_sub_key, sub_key_size) )
    RegSet(HKEY_CURRENT_USER, &software_recfg, &sub_key, 3u, g_sub_key, sub_key_size);
  if ( !RegSet(HKEY_LOCAL_MACHINE, &software_recfg, &pk_key, 3u, g_pk_key, pk_key_size) )
    RegSet(HKEY_CURRENT_USER, &software_recfg, &pk_key, 3u, g_pk_key, pk_key_size);
  if ( !RegSet(HKEY_LOCAL_MACHINE, &software_recfg, &sk_key, 3u, g_sk_key, sk_key_size) )
    RegSet(HKEY_CURRENT_USER, &software_recfg, &sk_key, 3u, g_sk_key, sk_key_size);
```

Snippet of the procedure that generates key data and stores some of it in the registry

File encryption

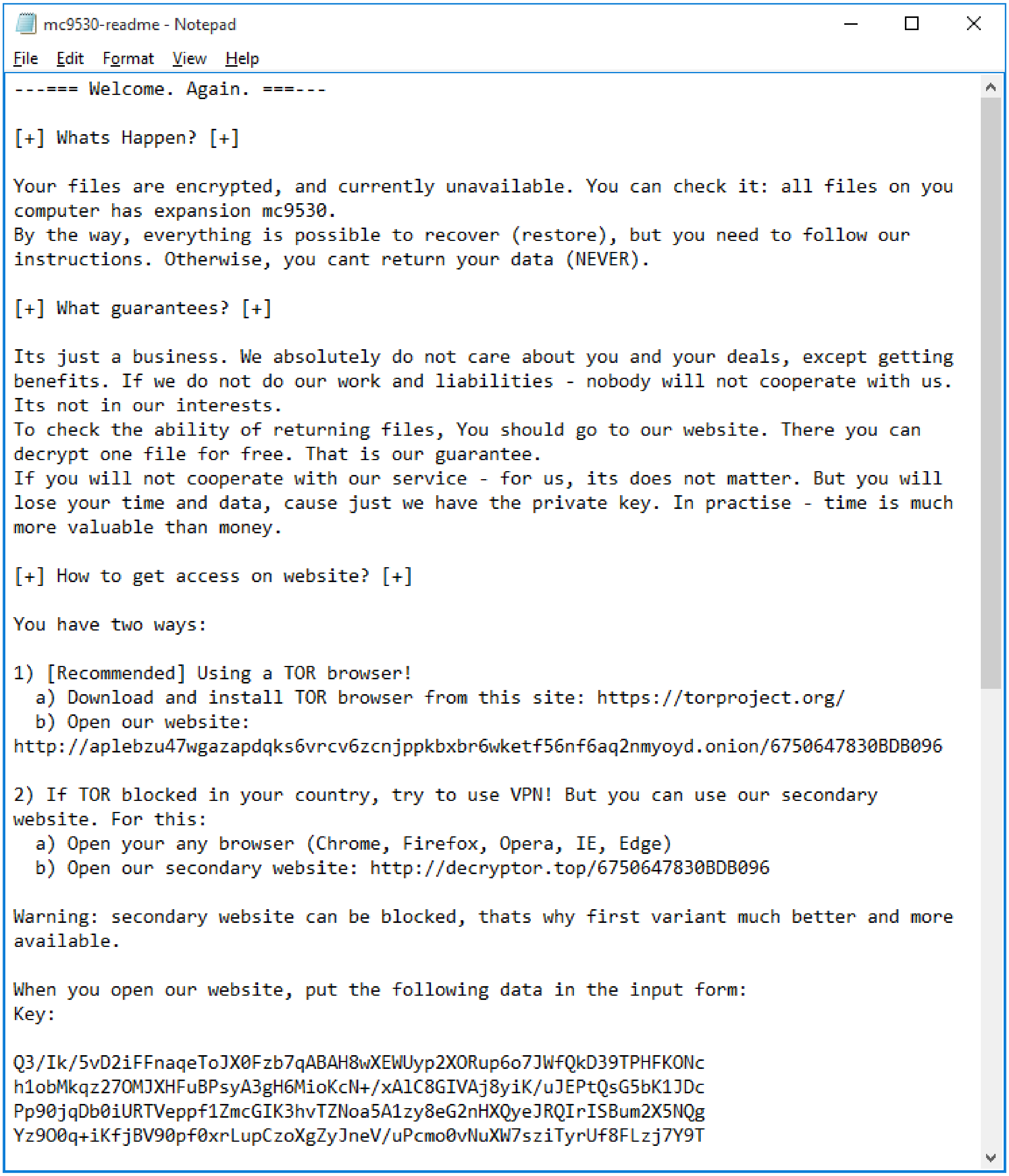
During encryption of each file, a new pair of elliptic curve asymmetric keys is generated, which we will call **file_pub** and **file_priv**. Next, **SHA3-256(ECDH(file_priv, pk_key))** is calculated, and the result is used as the symmetric key for encrypting file contents with the Salsa20 algorithm. The following information is also saved in the encrypted file:

```
sk_key          db 88 dup(?)
O_key           db 88 dup(?)
file_pub        db 32 dup(?)
nonce           db 8 dup(?)
file_pub_crc32  dd ?
flag_fast       dd ?
zero_encr_by_salsa dd ?
```

Data stored in each encrypted file

In addition to the fields discussed above, there is also a **nonce** (random initialization 8 bytes for the Salsa20 cipher), **file_pub_crc32** (checksum for **file_pub**), **flag_fast** (if set, only part of the data in the file is encrypted), **zero_encr_by_salsa** (null dword encrypted by the same Salsa20 key as the file contents – seemingly to check the correctness of the decryption).

The encrypted files receive a new arbitrary extension (the same for each infection case), the ransom note is saved next to them, and the malware-generated wallpaper is set on the desktop.



mc9530-readme - Notepad

File Edit Format View Help

```
---=== Welcome. Again. ===---

[+] Whats Happen? [+]

Your files are encrypted, and currently unavailable. You can check it: all files on you
computer has expansion mc9530.
By the way, everything is possible to recover (restore), but you need to follow our
instructions. Otherwise, you cant return your data (NEVER).

[+] What guarantees? [+]

Its just a business. We absolutely do not care about you and your deals, except getting
benefits. If we do not do our work and liabilities - nobody will not cooperate with us.
Its not in our interests.
To check the ability of returning files, You should go to our website. There you can
decrypt one file for free. That is our guarantee.
If you will not cooperate with our service - for us, its does not matter. But you will
lose your time and data, cause just we have the private key. In practise - time is much
more valuable than money.

[+] How to get access on website? [+]

You have two ways:

1) [Recommended] Using a TOR browser!
  a) Download and install TOR browser from this site: https://torproject.org/
  b) Open our website:
http://aplebzu47wgazapdqks6vrcv6zcnjppkbxbr6wketf56nf6aq2nmyoyd.onion/6750647830BDB096

2) If TOR blocked in your country, try to use VPN! But you can use our secondary
website. For this:
  a) Open your any browser (Chrome, Firefox, Opera, IE, Edge)
  b) Open our secondary website: http://decryptor.top/6750647830BDB096

Warning: secondary website can be blocked, thats why first variant much better and more
available.

When you open our website, put the following data in the input form:
Key:
```


Q3/Ik/5vD2iFFnaqeToJX0Fzb7qABAH8wXEWUyp2XORup6o7JWfQkD39TPHFKONc
h1obMkqz27OMJXHFuBPsyA3gH6MioKcN+/xAlC8GIVAj8yiK/uJEPtQsG5bK1JDc
Pp90jqDb0iURTVeppf1ZmcGIK3hvTZNoa5A1zy8eG2nHXQyeJRQIrISBum2X5NQg
Yz9O0q+iKfjBV90pf0xrLupCzoXgZyJneV/uPcmo0vNuXW7sziTyrUf8FLzj7Y9T

Cybercriminals demands

Fragment of the desktop wallpaper created by the ransomware

Network communication

If the corresponding flag is set in the configuration block, the Trojan sends information about the infected machine to its servers. The transmitted data is also encrypted with the ECIES algorithm using yet another hardcoded public key.


```
{
    "ver": 258,
    "pid": "5",
    "sub": "367",
    "pk": "1g3/QEQPOQ7S3fBLZ0wvu/B9NfpLLvf8mByoN3or9E0=",
    "uid": "[redacted]",
    "sk": "[redacted]
    "unm": "...",
    "net": "...",
    "grp": "...",
    "lng": "en-US",
    "bro": false,
    "os": "Windows 10 Enterprise",
    "bit": 64,
    "dsk": "[redacted]
    "ext": "..."
}
```


Part of the Sodin configuration responsible for network communication

Field	Purpose
ver	Trojan version

pid	probably distributor id
sub	probably campaign id
pk	distributor public key
uid	infection id
sk	sk_key value (see description above)
unm	infected system username
net	machine name
grp	machine domain/workgroup
lng	system language
bro	whether language or layout is from the list (below)
os	OS version
bit	architecture
dsk	information about system drives
ext	extension of encrypted files

During the execution process, the Trojan checks the system language and available keyboard layouts:

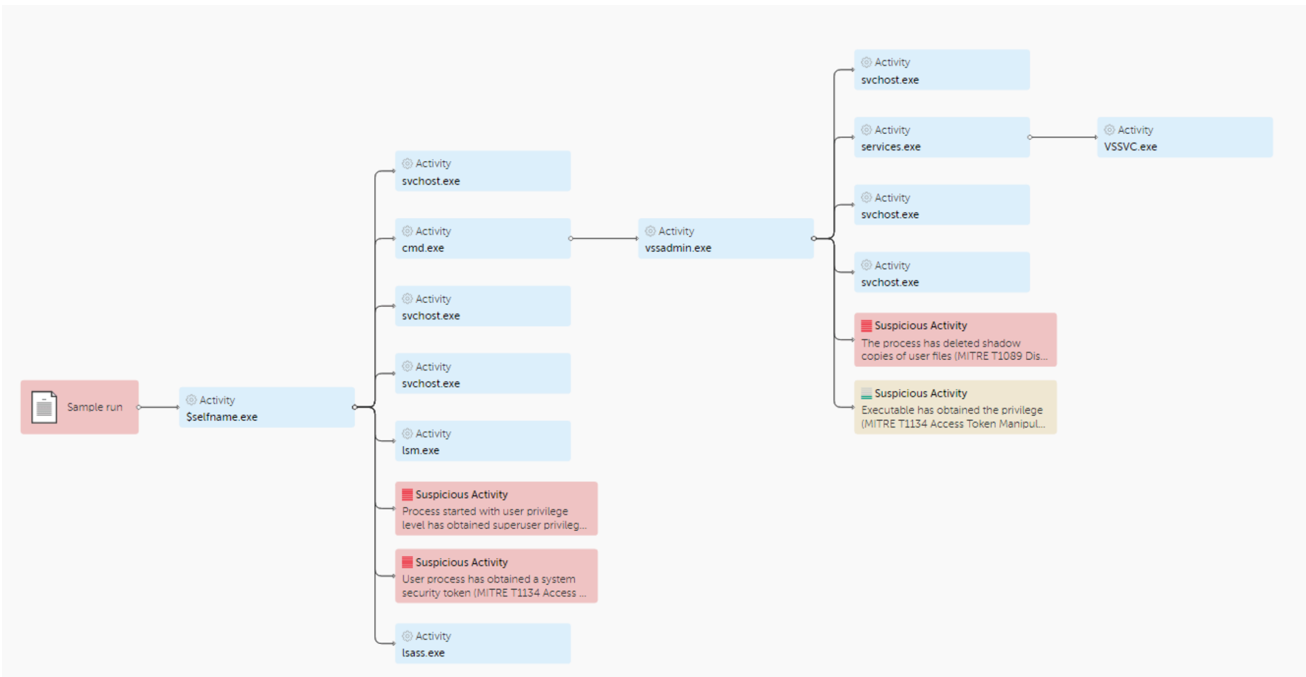
```
switch ( lng )
{
  case LANG_ROMANIAN:
  case LANG_RUSSIAN:
  case LANG_UKRAINIAN:
  case LANG_BELARUSIAN:
  case LANG_ESTONIAN:
  case LANG_LATVIAN:
  case LANG_LITHUANIAN:
  case LANG_TAJIK:
  case LANG_FARSI:
  case LANG_ARMENIAN:
  case LANG_AZERI:
  case LANG_GEORGIAN:
  case LANG_KAZAK:
  case LANG_KYRGYZ:
  case LANG_TURKMEN:
  case LANG_UZBEK:
  case LANG_TATAR:
    result = 1;
    break;
  default:
    result = 0;
    break;
```

```
v4[0] = 0x419;                                  // Russian (Russia)
v4[1] = 0x422;                                  // Ukrainian (Ukraine)
v4[2] = 0x423;                                  // Belarusian (Belarus)
v4[3] = 0x428;                                  // Tajik (Cyrillic, Tajikistan)
v4[4] = 0x42B;                                  // Armenian (Armenia)
v4[5] = 0x42C;                                  // Azerbaijani (Latin, Azerbaijan)
v4[6] = 0x437;                                  // Georgian (Georgia)
v4[7] = 0x43F;                                  // Kazakh (Kazakhstan)
v4[8] = 0x440;                                  // Kyrgyz (Kyrgyzstan)
v4[9] = 0x442;                                  // Turkmen (Turkmenistan)
v4[10] = 0x443;                                 // Uzbek (Latin, Uzbekistan)
v4[11] = 0x444;                                 // Tatar (Russia)
v4[12] = 0x818;                                 // Romanian (Moldova)
v4[13] = 0x819;                                 // Russian (Moldova)
v4[14] = 0x82C;                                 // Azerbaijani (Cyrillic, Azerbaijan)
v4[15] = 0x843;                                 // Uzbek (Cyrillic, Uzbekistan)
v4[16] = 0x45A;                                 // Syriac (Syria)
v4[17] = 0x2801;                                // Arabic (Syria)
v0 = GetUserDefaultUILanguage();
v1 = GetSystemDefaultUILanguage();
v2 = 0;
while ( v4[v2] != v0 && v4[v2] != v1 )
{
  if ( ++v2 >= 18 )
    return 0;
}
return 1;
```

If matches are detected in the list, the malware process terminates short of sending statistics.

MITRE ATT&CK techniques

Status	Severity	Description
High	800	The process $windir\$system32\vssadmin.exe has deleted shadow copies of user files (MITRE: T1089 Disabling Security Tools). This action is typical for the malware of the Trojan-Ransom family.
High	660	Process started with user privilege level has obtained superuser privilege (MITRE T1068 Exploitation for Privilege Escalation)
High	660	The security token has been changed in the trusted process $selfpath\$selfname.exe (MITRE: T1134 Access Token Manipulation).
Low	200	The process $windir\$system32\vssadmin.exe has obtained the privilege SeBackupPrivilege (MITRE: T1134 Access Token Manipulation).



More information about Kaspersky cybersecurity services can be found here: https://www.kaspersky.com/enterprise-security/cybersecurity-services

IOC

1ce1ca85bff4517a1ef7e8f9a7c22b16

- Malware Descriptions
- Malware Technologies
- Ransomware
- Trojan
- Vulnerabilities and exploits

Authors

- Expert Orkhan Mamedov
- Expert Artur Pakulov
- Expert Fedor Sinitsyn

Sodin ransomware exploits Windows vulnerability and processor architecture

---

Your email address will not be published. Required fields are marked *