

# REvil Ransomware-as-a-Service: An analysis of a ransomware affiliate operation

By Intel 471

Published: 2026-04-01 · Archived: 2026-04-05 18:08:54 UTC

**By the Intel 471 Malware Intelligence team.**

## Summary

REvil aka Sodinokibi, Sodin is a ransomware family operated as a ransomware-as-a-service (RaaS). Deployments of REvil first were observed in April 2019, where attackers leveraged a vulnerability in Oracle WebLogic servers tracked as CVE-2019-2725.

REvil is highly configurable and allows operators to customize the way it behaves on the infected host. Some of its features include:

- Exploits a kernel privilege escalation vulnerability to gain SYSTEM privileges using CVE-2018-8453.
- Whitelists files, folders and extensions from encryption.
- Kills specific processes and services prior to encryption.
- Encrypts files on local and network storage.
- Customizes the name and body of the ransom note, and the contents of the background image.
- Exfiltrates encrypted information on the infected host to remote controllers.
- REvil uses Hypertext Transfer Protocol Secure (HTTPS) for communication with its controllers.

## Capabilities Overview

- Ransomware.

## Behaviour Overview

- Uses a persistence mechanism.
- Encrypts additional resources.
- Supports privilege escalation.

## Adversary intelligence

### Developers

REvil ransomware first was advertised on a Russian language cybercrime forum in June 2019. The main actor associated with advertising and promoting REvil ransomware is called **Unknown** aka **UNKN**. The RaaS is operated as an affiliate service, where affiliates spread the malware by acquiring victims and the REvil operators maintain the malware and payment infrastructure. Affiliates receive 60% to 70% of the ransom payment.

Due to source code and behavior similarities between REvil and GandCrab, it was suggested there might be a connection tying the developers of the two ransomware families together. In addition to the similarities in the code, supplemental evidence tying GandCrab and REvil together is that GandCrab officially "retired" just before REvil appeared in the wild. REvil is maintained actively and is under constant development, just as GandCrab was. The most recent REvil ransomware at the time of this report was version 2.1.

The actor **Unknown** acknowledged the public reports linking REvil to GandCrab with the following statement:

*"We used to be affiliates of the (GandCrab) affiliate program. We bought the source code and started our own business. We developed custom features for our purposes"*

Despite the plausible alibi given by actor **Unknown**, the evidence suggests that REvil is a continuation of the GandCrab RaaS operation with new software, but operated by the same individuals.

The actors behind REvil include their master public key in all REvil binaries. Therefore, they are able to decrypt the files independently of the affiliates running the campaigns.

### Operators

REvil is a RaaS, in the sense that a single group operates and manages the development of the ransomware while access is sold to affiliates. A field in the configuration file named **pid** is used to identify the affiliate that the sample belongs to. We have confirmed that a field in the malware configuration named **sub** is used to identify affiliate campaigns, and not the

affiliates themselves, as frequently is reported. In addition, the attacker's public key can identify the same operator when used across multiple samples.

Two prominent users on the aforementioned Russian language cybercrime forum vouched for **Unknown's** ransomware service:

- **kerberos** - A moderator on the aforementioned forum and long-standing cybercriminal.
- **lalartu** - A cybercrime actor known to participate in GandCrab and REvil ransomware affiliate programs.

The actor **lalartu's** personal information possibly was released for malicious purposes or "doxxed" by an information security researcher known as UnderTheBreach (see: <https://medium.com/@underthebreach/tracking-down-revils-lalartu-by-utilizing-multiple-osint-methods-2bf3a6c65a80>). We have not been able to verify the conclusions asserted in this post.

## Infrastructure

REvil ransomware configurations contain more than 1,000 controllers. Interestingly, the live domains we verified all were WordPress websites, so it is probable they might be compromised by the operators or purchased from other cybercriminals.

The configuration also contains domain names that were not registered at the time of this report, for example:

```
$ whois andersongilmour[.]co[.]uk No match for "andersongilmour[.]co[.]uk."
```

This domain name was not registered at the time of this report.

It is likely most of the domains present in the configuration are decoys, with only a few real REvil controllers scattered inside the list.

## Detection

This section describes the methods used to detect a REvil sample.

### Static Detection

Unpacked REvil samples can be detected statically by looking up patterns in the code and in cryptographic functions used by the ransomware.

### Dynamic Detection

Dynamic detection of REvil samples is possible by running Yara signatures on the memory resident image. In addition, the ransomware creates the following interesting artifacts:

- A **.txt** file with a ransom note on each directory encrypted by the ransomware.
- A **.bmp** image in the temporary directory set as the desktop background after the encryption.
- A registry key **SOFTWARE** which can be present under either **HKEY\_LOCAL\_MACHINE** or **HKEY\_CURRENT\_USER**.
- An alphanumeric file extension between 5 and 10 characters in length appended to the original extension of an encrypted file.

## Yara Signatures

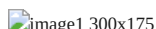
We decided not to release Yara signatures to avoid burning them and affecting the work of other researchers. However, we will release them to network defenders and infosec professionals upon request. Email us at [revil-yara-reqREMOVE@intel471\[.\]com](mailto:revil-yara-reqREMOVE@intel471[.]com) (remove the "REMOVE" text from the email address) from your corporate email address (for validation purposes only) and we will send through our Yara signatures.

## Exploits

REvil ransomware exploits a kernel privilege escalation vulnerability in win32k.sys tracked as CVE-2018-8453 to gain SYSTEM privileges on the infected host. If the configuration instructs a sample to execute this exploit, it will allocate executable memory, decrypt the exploit code in the newly allocated region and invoke it.

The screenshot below shows the window name **sysshadow** and the application programming interface (API)

**DestroyWindow** inside the embedded exploit, proving it is the CVE-2018-8453 vulnerability.

image1 300x175

## Open Source Intelligence (OSINT)

For more information on REvil, we suggest the following public resources:

- REvil/Sodinokibi Ransomware (see: <https://www.secureworks.com/research/revil-sodinokibi-ransomware>).

- McAfee ATR Analyzes Sodinokibi aka REvil Ransomware-as-a-Service – What The Code Tells Us (see: <https://www.mcafee.com/blogs/other-blogs/mcafee-labs/mcafee-atr-analyzes-sodinokibi-aka-revil-ransomware-as-a-service-what-the-code-tells-us/> ).
- Tracking REvil (see: <https://www.kpn.com/security-blogs/Tracking-REvil.htm> ).
- Defeating Sodinokibi/REvil String-Obfuscation in Ghidra (see: <https://blog.nullteilerfrei.de/2020/02/02/defeating-sodinokibi-revil-string-obfuscation-in-ghidra/> ).

### Static Analysis

REvil ransomware incorporates techniques to make the task of static analysis more difficult for an analyst. Most of the strings used during execution are decrypted at runtime only when needed. In addition, the imports are resolved dynamically from 32-bit hashes and placed into global variables early in execution.

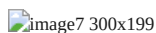
### Hard-coded Strings

Very few strings are hard coded inside REvil ransomware samples and the most interesting are:

- **L“ServicesActive”**: This string is passed to the **OpenSCManagerW** API to retrieve active services.
- **“expand 32-byte kexpand 16-byte”**: The constants used by the Salsa20 symmetric encryption algorithm.

### String Encryption

REvil ransomware decrypts most of the strings it uses during execution at runtime.

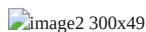


As the screenshot above reveals, these strings are Rivest Cipher 4 (RC4) encrypted and are decrypted using a function we renamed to **rc4\_decrypt\_string**. Below is its prototype and the meaning of each parameter:

```
void rc4_decrypt_string(BYTE *rc4_array, int rc4_key_offset, int rc4_key_length, int buffer_length, BYTE *out_buffer)
```

- **rc4\_array**: A pointer to a contiguous array in the .data section containing the RC4 keys and the encrypted strings. Each RC4 key is followed directly by the string it decrypts.
- **rc4\_key\_offset**: The offset to the RC4 key within the array.
- **rc4\_key\_length**: The length of the RC4 key.
- **buffer\_length**: The length of the RC4 encrypted buffer.
- **out\_buffer**: Pre-allocated memory or stack space where the decrypted string is copied.

The image below shows the layout of two adjacent elements of the array:



The decrypted strings are not terminated by NULL characters, therefore, the code must terminate these strings explicitly.

### Malware Behavior

The behavior of the following samples was analyzed for this report:

| Sample           | SHA256   |
|------------------|--|
| REvil packed     | 6953d86d09cb8ed34856b56f71421471718ea923cd12c1e72224356756db2ef1 |
| REvil not packed | 372c8276ab7cad70ccf296722462d7b8727e8563c0bfe4344184e1bc3afc27fc |
| REvil not packed | ec0c653d5e10fec936dae340bf97c88f153cc0cdf7079632a38a19c876f3c4fe |

### Process Execution

During its initialization phase, REvil starts by dynamically resolving the imports it needs to function correctly. This is accomplished in a loop that reads hard-coded 32-bit values from an array in the .data section, then each value is decoded and resolved to the correct API by the responsible function **rvt\_resolve\_api**. Then the 32-bit value is overwritten with the pointer to the API.

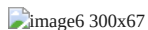


As shown in the disassembly above, additional APIs are resolved by their names with the help of the **GetProcAddress** API.

With everything in place, REvil creates a global mutual exclusion object (mutex) with a hard-coded name i.e., **Global\1DE3C565-E22C-8190-7A66-494816E6C5F5**. This is used to ensure only a single instance of the ransomware sample is running.

The REvil ransomware always attempts to run with elevated privileges and does so using two techniques. One technique relies on exploiting the CVE-2018-8453 vulnerability to gain SYSTEM privileges on the host. However, to determine whether it should execute the exploit or not, REvil checks its configuration. Another technique always is executed if the process is not elevated. It relies on calling **ShellExecuteW** to prompt the user to run the sample as an administrator. This is accomplished in an infinite loop until the user agrees to run the elevated process. Starting with version 2.1, the privilege escalation exploit code was removed.

REvil's configuration is in JavaScript Object Notation (JSON) and is initially RC4 encrypted. It is stored in the following fashion at the beginning of a portable executable (PE) section named **.yhwfq9**:



It is important to mention the cyclic redundancy check (CRC32) value is calculated and validated for the encrypted configuration.

After decrypting the configuration in dynamically allocated memory, REvil parses it using the open-source json-parser C library (see: <https://github.com/udp/json-parser>). For an example of a REvil configuration, see the Appendix section below.

The first JSON field checked is **exp**, which can be **true** or **false** and indicates whether the CVE-2018-8453 vulnerability should be exploited. If the exploit isn't executed or failed, REvil resorts to the second technique previously described in this section.

When REvil executes with higher privileges, it starts its main initialization phase where it reads the necessary JSON fields into the .data section and initializes registry values inside a new subkey named **SOFTWARE**. When doing registry operations in general, REvil first tries to use the **HKEY\_LOCAL\_MACHINE** hive and it switches to use **HKEY\_CURRENT\_USER** only if that fails. The configuration fields, registry keys and their values are described in the below Configuration section.

The other important task in this phase is filling the ransom note template, which is present in base64 inside the configuration field **nbody**.

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

[+] Whats Happen? [+]

Your files are encrypted, and currently unavailable. You can check it: all files on your computer 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).

[+] 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: `hxxp://aplebzu47wgazapdqks6vrcv6zcnjppkxbr6wketf56nf6aq2nmyoyd[.]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: `hxxp://decryptor[.]cc/{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}

Extension name:

{EXT}

!!! DANGER !!!

DONT try to change files by yourself, DONT use any third party software for restoring your data or antivirus solutions - its may entail damage 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.

!!! !!!!

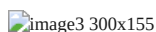
REvil fills this template with a file extension (EXT), a user identifier (UID) and a key (KEY):

- The extension is generated randomly from alphanumeric characters and is between 5 and 10 characters in length.
- The user identifier is a hardware ID generated from the main volume's serial number and the system's CPU information. This ID is calculated as follows: Retrieve the main volume 32-bit serial number. Generate a CRC32 checksum of the volume serial with a seed value equal to 1337. Retrieve central processing unit (CPU) information i.e., "Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz." Generate a CRC32 checksum of the CPU information string with the volume serial CRC32 hash as a seed value. Concatenate the CPU information CRC32 hash with the volume serial number into a 16-byte hexadecimal string i.e., 8100F233BC097E90.
- The key submitted by the victim to the operator through the website is a JSON string:

```
{ "ver": "%d", # REvil's version: hard coded to 0x200 (v2.00). "pid": "%s", # The "pid" field in the configuration. "sub": "%s", # The "sub" field in the configuration. "pk": "%s", # The public key "pk" field in the configuration. "uid": "%s", # The hardware ID. "sk": "%s", # The victim's generated secret key encrypted with the "pk" public key. This key is necessary for decryption. "unm": "%s", # The username of the Windows account. "net": "%s", # The computer name. "grp": "%s", # The domain name. "lng": "%s", # Language i.e., "fr-FR". "bro": "%s", # "true" or "false". Is immune to infection, whitelisted language and keyboard layout. "os": "%s", # Windows product name. "bit": %d, # CPU architecture: 86 or 64. "dsk": "%s", # A base64 encoded structure with information on the mounted volumes. "ext": "%s" # The encrypted files extension i.e., ".g19b9wy" }
```

This is encrypted with a hard-coded public key in the binary code before it is stored in the registry and the template. It also is communicated to the controllers later in execution. See the Encryption section below for information on how REvil encrypts data.

Next, REvil checks the configuration field **dbg** to see if it's running in debug mode. If that is not the case, geolocation checks based on the system's language and the keyboard layout are conducted so the ransomware does not attempt to encrypt files on whitelisted systems. The following are whitelisted system language IDs for the analyzed sample:



#### The whitelisted keyboard layouts include:

- Romanian
- Russian
- Ukrainian
- Belarusian
- Estonian
- Latvian
- Lithuanian
- Tajik
- Persian
- Armenian
- Azerbaijani
- Georgian
- Kazakh
- Kyrgyz
- Turkmen
- Uzbek
- Tatar

For the check to succeed and REvil to exit, both a whitelisted system language and a whitelisted keyboard layout must be present. Otherwise, the ransomware continues its execution normally.

At this stage, REvil performs the following actions before starting the encryption. First, it will try to stop and delete services if the names match one of the regular expressions in the **svc** JSON configuration list, for example:

```
<strong>"svc":</strong>[ "memtas", "crm", "quickbooks", "svc$", "veeam", "oracle", "mepocs", "exchange", "pos", "vss", "sql", "backup", "qb", "sophos", "sage" ]
```

Then it will terminate all processes with names that match the elements of the **prc** JSON array, for instance:

```
<strong>"prc":</strong>[ "w3wp", "thunderbird", "mydesktopqos", "powerpnt", "outlook", "srv", "infopath", "msaccess", "ocautoupds" ]
```

Finally REvil will delete the volume shadow copies. The way this is accomplished depends on the Windows version:

- Windows versions 5.1 and earlier:

```
cmd.exe /c vssadmin.exe Delete Shadows /All /Quiet & bcdedit /set {default} recoveryenabled No & bcdedit /set {default} bootstatuspolicy ignoreallfailures
```

- Windows versions 5.2 and later, under PowerShell:

```
Get-WmiObject Win32_Shadowcopy | ForEach-Object {$_.Delete();}
```

At this stage, REvil enters the file encryption phase. It is possible to run REvil with command-line arguments that will impact how encryption is carried out. The following table describes these arguments:

| Argument | Description   |
|----------|---|
| -full    | Encrypts all the data in the target files.  |
| -fast    | Only encrypts the first MB of each file. Mutually exclusive with the -full argument.  |
| -path    | Recursively encrypts the files inside a single directory specified after the argument. The desktop background image remains unchanged when this argument is supplied. |
| -nolan   | Does not encrypt files on shared network storage.   |
| -nolocal | Does not encrypt files on local storage.  |

Two values that the **et** configuration field can take are equivalent to the command-line arguments **-full** and **-fast**.

Additionally, the presence of one of these command-line arguments overrides the value of this field. Below are the values this field can take with a description of each value:

| Encryption type values | Description   |
|------------------------|---|
| 0                      | Equivalent to the command-line argument -full   |
| 1                      | Equivalent to the command-line argument -fast   |
| 2                      | Encrypts 1 MB then skips the number of MBs supplied in the spsize field repeatedly, until the end of the file is reached. |

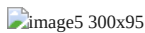
Before encrypting a file or directory, REvil determines whether its name matches any entry in the whitelist configuration entries. The whitelisted folders, files and extensions are lists stored in the **whl** JSON configuration field as **fld**, **fls** and **ext**.

REvil ransomware also avoids encrypting a file more than once by determining if it previously was encrypted. This is achieved by examining whether the metadata it stores at the end of encrypted files exists.

To encrypt the files, REvil relies on multithreading and input/output (I/O) completion ports, allowing it to perform asynchronous I/O and process multiple files simultaneously.

Each file is encrypted with a different key derived from a single public key linked to the victim. See the Encryption section below for details on how REvil implements file encryption.

After the encryption is complete, REvil generates a bitmap image and sets it as the desktop background.



Afterward, REvil ransomware reads the Boolean configuration field **net** to determine whether it should communicate with its controllers. See the Communications section below for more details.

Finally, REvil ransomware marks its binary code for deletion during the next reboot and terminates execution.

### Configuration

The table below describes each field of the REvil JSON configuration:

| Argument | Description                                  |
|----------|--|
| pk       | The attacker's public key encoded in base64. |

|        |  |
|--------|--|
| pid    | The affiliate ID. In v2.0 and earlier, this was an integer. From 2.1 it became a Bcrypt hash.  |
| sub    | An integer value. The campaign identifier.   |
| dbg    | Boolean value that determines if REvil should run in debug mode.   |
| et     | An integer value specifying the encryption type.- 0: Fast encryption. - 1: Full encryption. - 2: Encrypt 1 MB then skip the number of MBs specified by the spsize field. |
| spsize | Specifies the number of MBs to skip when the encryption type is 2.   |
| wipe   | A Boolean value that indicates whether REvil should wipe folders specified in the wfld element. We have not seen this option implemented in the analyzed samples.        |
| wfld   | The folders to be wiped by REvil.  |
| wht    | Contains three lists of elements that REvil will not encrypt: - fld: Whitelisted folders. - fls: Whitelisted files. - ext: Whitelisted extensions.                       |
| prc    | A list of regular expressions that REvil matches against processes to terminate them.  |
| net    | If this Boolean value is set to true, REvil communicates with its controllers.   |
| dmn    | A string of controllers separated by a semicolon.  |
| svc    | A list of regular expressions to match against running services to stop and delete them.   |
| nbody  | The template of the ransom note encoded in base64.   |
| nname  | The file name of the ransom note to be dropped inside encrypted directories.   |
| exp    | Indicates if REvil should exploit the CVE-2018-8453 vulnerability to escalate privileges.  |
| img    | The text to be written in the background image encoded in base64.  |
| arn    | When set to true, REvil persists in the system.  |

The registry values that REvil creates are described below:

| Value name | Description  |
|------------|--|
| 1TfXk      | Victim's secret key encrypted with the attacker's public key present in the configuration. |
| 2YEdLY     | Victim's secret key encrypted with a master public key hard coded in the binary code.      |
| aah        | The attacker's public key.   |
| fdle       | The victim's public key.   |
| AaZW1s3    | The extension appended to files after encryption.  |
| QaUXNv2P   | The victim's key encrypted with a second hard-coded public key in the binary code.         |

### Encryption

REvil ransomware implements encryption schemes involving an elliptic curve called Curve25519, Salsa20, SHA-3, CRC32 and Advanced Encryption Standard (AES). We identified the specific open-source implementations utilized by REvil:

- Curve25519: <https://github.com/vstakhov/opt-cryptobox/tree/master/curve25519> .
- Salsa20: <https://cr.yp.to/snuffle/salsa20/merged/salsa20.c> .

This subsection delves into the details of how these algorithms are used to encrypt data and files.

### Encryption keys

REvil ransomware relies on Curve25519 to generate public and secret key pairs and to create shared keys for encryption. What follows defines the Curve25519 keys referred to throughout this subsection:

| Key-pair name                     | Description  |
|-----------------------------------|--|
| crypt (crypt_public/crypt_secret) | The victim's main key pair used for file encryption. |
| file (file_public / file_secret)  | A random key pair generated for each encrypted file. |

|   |  |
|---|--|
| attacker (attacker_public / attacker_secret)          | The attacker's key pair. attacker_public is present under the pk configuration field.  |
| master (master_public / master_secret)                | The master key pair. master_public is hard coded inside the binary code and is the same across all REvil samples.  |
| user_config (user_config_public / user_config_secret) | The user configuration key pair. user_config_public is hard coded in the binary code and is used to encrypt the user key we find encrypted in the ransom note. |

#### Data encryption

REvil ransomware encrypts all important data it stores in the registry. For instance, the user key present in the ransom note and in the registry is a JSON dictionary encrypted using the method we describe in the paragraphs that follow.

To encrypt a buffer, REvil invokes the following function:

```
BYTE* rvl_encrypt_data(BYTE *hispublic, BYTE *buffer, int buffer_length, BYTE *out_buffer_length)
```

This function requires a 32-byte Curve25519 public key, a buffer to encrypt and its length. The end result is a pointer to the buffer in the return value and its length in the output parameter **out\_buffer\_length**.

Internally, this function performs the following actions:

- Allocate buffer\_length bytes plus an extra 56 bytes to hold the final data.
- Zero out the first 4 bytes of the allocated buffer.
- Copy the buffer to encrypt after the zeroed double word.
- Generate a new Curve25519 key pair we call new\_public and new\_secret.
- Calculate a Curve25519 shared key between the hispublic key supplied in the arguments and the new\_secret. The recipient of the message can use the secret key and new\_public to obtain the same shared key.
- Hash the shared key using the SHA-3 algorithm.
- Clear the shared key from memory.
- Clear new\_secret from memory.
- Generate a random 128-bit AES initialization vector.
- Encrypt the buffer using AES with the SHA-3 hash as a key. The buffer to encrypt includes the prepended 32-bit NULL value.
- Clear the SHA-3 hash from memory.
- Calculate the CRC32 of the encrypted buffer.
- Append the following elements to the encrypted buffer in this order:
  - The generated new\_public key (32 bytes).
  - The initialization vector (16 bytes).
  - CRC32 of the encrypted buffer (4 bytes).

The end result looks like:



As an example, after decoding the user key present in the ransom note using base64, we see it respects this same format:



In order for the attackers to decrypt this data and retrieve the JSON dictionary and as a result the encrypted victim's crypt\_secret key, they must do the following:

- Verify the CRC32 hash of the encrypted data.
- Calculate a Curve25519 shared key using the new\_public key in the data and the attacker's secret user\_config\_secret.
- Hash the shared key using SHA-3.
- Retrieve the AES initialization vector and decrypt the buffer using AES.
- Remove the NULL double word at the start of the buffer.

It is important to mention the victim's *crypt\_secret* key is encrypted the same way using the attacker's *attacker\_public* key (registry value "1TfXk") and the *master\_public* key (registry value "2YEdLY").

#### File encryption

REvil ransomware encryption works by encrypting the files in 1 MB increments unless the file size is smaller, the remaining bytes to encrypt are less than 1 MB or when the encryption type specified is not full encryption. Contents of the file are read, encrypted and written back to the file overwriting the original content. After the encryption is complete, metadata is written to the end of file and the file is renamed to include the generated file extension (registry value **AaZW1s3**).

For each given file, REvil ransomware starts encryption by initializing a structure used throughout the process. Its first part includes the Windows **OVERLAPPED** structure used for asynchronous I/O followed by custom fields REvil uses. The most interesting part of this structure was reverse engineered to the following:

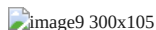
```
<strong>typedef</strong> <strong>struct</strong> _rvl_filecrypt_struct { OVERLAPPED Overlapped;
<em>/*.SNIP..*/</em> <strong>struct</strong> _rvl_metadata { BYTE crypt_secret_w_attacker_pub[88]; <em>// same
as registry value "1TfXk".</em> BYTE crypt_secret_w_master_pub[88]; <em>// same as registry value "2YEdLY".
</em> BYTE file_public[32]; <em>// The file_public key.</em> BYTE salsa20_iv[8]; <em>// salsa20 initialization
vector.</em> DWORD crc32_file_public; <em>// CRC32 hash of file_public.</em> DWORD et; <em>// The encryption
type.</em> DWORD spsize; <em>// spsize field if applicable.</em> DWORD encrypted_null; <em>// A NULL value
encrypted with salsa20.</em> } metadata; <em>/*.SNIP..*/</em> } rvl_filecrypt_struct, *prvl_filecrypt_struct;
```

This metadata substructure is appended to the end of every encrypted file, is used by the decrypter and by REvil to verify a file previously was encrypted.

The Salsa20 symmetric encryption key is the SHA-3 hash of a shared key derived from the victim's *crypt\_public* key and the secret of a generated key pair: *file\_secret*.

To decrypt a file, both the victim's *crypt\_secret* key and the *file\_public* key must be known:

- It is possible to decrypt the *crypt\_secret* knowing either the operators' *attacker\_secret* or *master\_secret* keys and by applying the same logic described in the Data encryption subsection.
- It is possible to retrieve the per file *file\_public* key from the end of an encrypted file after validating its CRC32 hash.
- Having attained both a public and a secret key from previous steps, generate a shared Curve25519 key and hash it using SHA-3.
- Copy the value of the Salsa20 initialization vector from the metadata structure at the end of the file.
- Test decrypting the encrypted NULL double word with the SHA-3 hash as a key.
- Determine the encryption type used to perform correct decryption of the file, then start decrypting accordingly.



By following this scheme, the attackers are sure not to divulge their secret keys when sending decrypters to victims. On their side, attackers only have to decrypt the victim's *crypt\_secret* key and send back a decrypter embedding this key.

### Persistence mechanism

Prior to version 2.1, REvil ransomware persists on the machine if the **arn** configuration field is set to **true**. It writes its path to the registry key **SOFTWARE** for persistence. The value name of the entry for the analyzed sample is **k51299BQXH**.

Before terminating execution, REvil marks its executable file for deletion during the next reboot. As a result, the persistence path will become invalid.

The persistence mechanism was removed from REvil version 2.1.

### Protections

REvil ransomware does not implement any protections.

### Communications

REvil ransomware only initiates communication with its controllers when the configuration field **net** is set to **true**. In that case, REvil extracts the controller list from the **dmn** configuration string and proceeds to build a custom URL in a loop for each controller before initiating communications.

Each controller is preceded by the string **https://**, followed by a random path chosen from hard-coded values and then terminated by a random file name. The regular expression below matches all random URL paths generated by REvil ransomware:

```
\\(wp-
content|static|content|include|uploads|news|data|admin)\\(images|pictures|image|temp|tmp|graphic|assets|pics|game)\\([a-
z]{2}){1,10}\\.(jpg|png|gif)
```

REvil then proceeds to send a POST request containing the victim's key stored in the registry value **QaUXNv2P** to the controllers. Afterward, the controller response is read into a buffer that subsequently is ignored and freed.

### Appendix

#### Appendix 1: REvil configuration

Example of a REvil configuration. We redacted most of the controllers for readability:

```
{ "pk": "Y09E7out83RseZgGnLR2DxiFRbXiteYQir0JcZ0jpl0=",
"pid": "$2a$10$7qQ70sylvX5aslSuSa9AWurg843zRR.433XEtfk2rGURjN9e.xNz6", "sub": "3195", "dbg":
<strong>false</strong>, "et": 1, "wipe": <strong>false</strong>, "wht": { "fld": [ "programdata", "perflogs",
"application data", "appdata", "$windows.bt", "system volume information", "windows", "tor browser", "google",
"msocache", "boot", "windows.old", "$windows.ws", "mozilla", "intel" ], "fls": [ "boot.ini", "ntuser.dat",
"iconcache.db", "autorun.inf", "bootsect.bak", "desktop.ini", "thumbs.db", "ntuser.ini", "ntuser.dat.log",
"ntldr", "bootfont.bin" ], "ext": [ "msp", "rom", "rtp", "shs", "mod", "cur", "msc", "nomedia", "deskthemepack",
"diagcab", "diagcfg", "msstyles", "scr", "hta", "idx", "ics", "lock", "diagpkg", "icns", "msi", "themepack",
"key", "bin", "theme", "bat", "cab", "nls", "spl", "icl", "sys", "drv", "lnk", "cmd", "adv", "cpl", "ico",
"com", "exe", "ocx", "dll", "hlp", "mpa", "prf", "wpx", "ani", "msu", "ps1", "386" ] }, "wfld": [ "backup" ],
"prc": [ "w3wp", "thunderbird", "mydesktopqos", "powerpnt", "outlook", "srv", "infopath", "msaccess",
"ocautoupds", "qb", "core", "mispub", "store", "ssms", "dbeng50", "ax32", "sql", "exchange", "onenote", "ocssd",
"sage", "pos", "word", "java", "sophos", "xfssvcon", "visio", "synctime", "oracle", "crm", "excel", "dbs",
"ocomm", "svc$" ],
"dmn": "sweering[.]fr;shiresresidential[.]com;bogdanpeptine[.]ro;ruralarcoiris[.]com;echtveilig[.]nl", "net":
<strong>false</strong>, "svc": [ "memtas", "crm", "quickbooks", "svc$", "veeam", "oracle", "mepocs", "exchange",
"pos", "vss", "sql", "backup", "qb", "sophos", "sage" ],
"nbody": "LQAtAC0APQ9AD0AIABXAGUAbABjAG8AbQ8LAC4AIABBAGcAYQBpAG4ALgAgAD0APQ9AC0ALQAtAA0ACgANAAoAWwArAF0AIABXAGgAYQB0AHMAIABIAGEAcABwAGUA
"nname": "{EXT}-readme.txt", "exp": <strong>false</strong>,
"img": "QQB$AGwAIABvAGYAIAB5AG8AdQBvACAAZgBpAGwAZQBzACAAYQByAGUAIABLAG4AYwByAHkAcAB0AGUAZAAhAA0ACgANAAoARgBpAG4AZAAgAHSARQBvAFQAFQAtAHIAZQ
"arn": <strong>false</strong>
}
```

**Anti-virus (AV) classifications**

| AV                   | Alias                      |
|----------------------|----------------------------|
| ALYac                | Trojan.Ransom.Sodinokibi   |
| Ikarus               | Trojan-Ransom.Sodinokibi   |
| ClamAV               | Win.Ransomware.Sodinokibi  |
| ESET-NOD32           | Win32/Filecoder.Sodinokibi |
| Fortinet             | W32/Sodinokibi.B!tr.ransom |
| Malwarebytes         | Ransom.Sodinokibi          |
| Microsoft            | Ransom:Win32/Sodinokibi    |
| Rising               | Ransom.Sodin               |
| TrendMicro HouseCall | Win32.SODINOKIB.SMTH       |
| ViRobot              | Trojan.Win32.Z.Sodinokibi  |
| Webroot              | W32.Ransom.Sodinokibi      |

Source: <https://intel471.com/blog/revil-ransomware-as-a-service-an-analysis-of-a-ransomware-affiliate-operation/>