WastedLocker: A New Ransomware Variant Developed By The Evil Corp Group

research.nccgroup.com/2020/06/23/wastedlocker-a-new-ransomware-variant-developed-by-the-evil-corp-group

June 23, 2020

Authors: *Nikolaos Pantazopoulos*, *Stefano Antenucci* (@Antelox), *Michael Sandee* and *in close collaboration with NCC's RIFT.*

> *About the Research and Intelligence Fusion Team (RIFT):*
> RIFT leverages our strategic analysis, data science, and threat hunting capabilities to create actionable threat intelligence, ranging from IOCs and detection capabilities to strategic reports on tomorrow's threat landscape. Cyber security is an arms race where both attackers and defenders continually update and improve their tools and ways of working. To ensure that our managed services remain effective against the latest threats, NCC Group operates a Global Fusion Center with Fox-IT at its core. This multidisciplinary team converts our leading cyber threat intelligence into powerful detection strategies.

1. Introduction

WastedLocker is a new ransomware locker we've detected being used since *May 2020*. We believe it has been in development for a number of months prior to this and was started in conjunction with a number of other changes we have seen originate from the *Evil Corp* group in 2020. Evil Corp were previously associated to the *Dridex* malware and *BitPaymer* ransomware, the latter came to prominence in the first half of 2017. Recently Evil Corp has changed a number of TTPs related to their operations further described in this article. We believe those changes were ultimately caused by the unsealing of indictments against *Igor Olegovich Turashev* and *Maksim Viktorovich Yakubets*, and the financial sanctions against Evil Corp in December 2019. These legal events set in motion a chain of events to disconnect the association of the current Evil Corp group and these two specific indicted individuals and the historic actions of Evil Corp.

2. Attribution and Actor Background

We have tracked the activities of the Evil Corp group for many years, and even though the group has changed its composition since 2011, we have been able to keep track of the group's activities under this name.

2.1 Actor Tracking

Business associations are fairly fluid in organised cybercrime groups, Partnerships and affiliations are formed and dissolved much more frequently than in nation state sponsored groups, for example. Nation state backed groups often remain operational in similar form over longer periods of time. For this reason, *cyber threat intelligence reporting can be misleading*, given the difficulty of maintaining assessments of the capabilities of cybercriminal groups which are accurate and current.

As an example, the Anunak group (also known as FIN7 and Carbanak) has changed composition quite frequently. As a result, the public reporting on FIN7 and Carbanak and their various associations in various open and closed source threat feeds can distort the current reality. The *Anunak or FIN7 group has worked closely with Evil Corp, and also with the group publicly referred to as TA505.* Hence, TA505 activity is sometimes still reported as Evil Corp activity, even though these groups have not worked together since the second half of 2017.

It can also be difficult to accurately attribute responsibility for a piece of malware or a wave of infection because commodity malware is typically sold to interested parties for mass distribution, or supplied to associates who have experience in monetising access to a specific type of business, such as financial institutions. Similarly, it is easy for confusion to arise around the many financially oriented organised crime groups which are tracked publicly. *Access to victim organisations is traded as a commodity between criminal actors* and so business links often exist which are not necessarily related to the day to day operations of a group.

2.2 Evil Corp

Nevertheless, despite these difficulties, we feel that we can assert the following with high confidence, due to our in depth tracking of this group as it posed a significant threat to our clients. Evil Corp has been operating the *Dridex malware* since July 2014 and provided access to several groups and individual threat actors. However, towards the end of 2017 Evil Corp became smaller and used Dridex infections almost exclusively for targeted ransomware campaigns by *deploying BitPaymer*. The majority of victims were in *North America* (mainly USA) with a smaller number in *Western Europe* and instances outside of these regions being just scattered, individual cases. During 2018, Evil Corp had a short lived *partnership with TheTrick* group; specifically, leasing out access to BitPaymer for a while, prior to their use of Ryuk.

In 2019 a fork of BitPaymer usually referred to as *DoppelPaymer* appeared, although this was ransomware as a service and thus was not the same business model. We have observed some cooperation between the two groups, but as yet can draw no definitive conclusions as to the current relationship between these two threat actor groups.

After the unsealing of indictments by the US Department of Justice and actions against Evil Corp as group by the US Treasury Department, we detected a short period of inactivity from Evil Corp until January 2020. However, since January 2020 activity has resumed as usual, with victims appearing in the same regions as before. It is possible, however, that this was primarily a strategic move to suggest to the public that Evil Corp was still active as, from around the middle of March 2020, we failed to observe much activity from them in terms of BitPaymer deployments. Of course, this period coincided with the lockdowns due to the COVID19 pandemic.

The development of new malware takes time and it is probable that they had already started the development of new techniques and malware. Early indications that this work was underway included the use of a variant of Gozi we refer to as Gozi ISFB 2 variant. It is thought that this variant is intended as a replacement for Dridex botnet 501 as one of the persistent components on a target network. Similarly, a customized version of the CobaltStrike loader has been observed, possibly intended as a replacement for the Empire PowerShell framework previously used.

The group has access to *highly skilled exploit and software developers* capable of bypassing network defences on all different levels. The group seems to put a lot of effort into bypassing endpoint protection products; this observation is based on the fact that when a certain version of their malware is detected on victim networks the group is back with an undetected version and able to continue after just a short time. This shows the importance of victims fully understanding each incident that happens. That is, detection or blocking of a single element from the more advanced criminal actors does not mean they have been defeated.

The lengths Evil Corp goes through in order to *bypass endpoint protection tools* is demonstrated by the fact that they abused a victim's email so they could pose as a legitimate potential client to a vendor and request a trial license for a popular endpoint protection product that is not commonly available.

It appears the group regularly finds *innovative but practical approaches to bypass detection* in victim networks based on their practical experience gained throughout the years. They also *demonstrate patience and persistence*. In one case, they successfully compromised a target over 6 months after their initial failure to obtain privileged access. They also display attention to detail by, for example, ensuring that they *obtain the passwords to disable security tools* on a network prior to deploying the ransomware.

2.3 WastedLocker

The new WastedLocker ransomware appeared in May 2020 (a technical description is included below). The ransomware name is derived from the filename it creates which includes an abbreviation of the victim's name and the string '*wasted*'. The abbreviation of the victim's name was also seen in BitPaymer, although a larger portion of the organisation name was used in BitPaymer and individual letters were sometimes replaced by similar looking numbers.

Technically, WastedLocker does not have much in common with BitPaymer, apart from the fact that it appears that victim specific elements are added using a specific builder rather than at compile time, which is similar to BitPaymer. Some similarities were also noted in the ransom note generated by the two pieces of malware. The first WastedLocker example we found contained the victim name as in BitPaymer ransom notes and also included both a protonmail.com and tutanota.com email address. Later versions also contained other Protonmail and Tutanota email domains, as well as Eclipso and Airmail email addresses. Interestingly the user parts of the email addresses listed in the ransom messages are numeric (usually 5 digit numbers) which is similar to the 6 to 12 digit numbers seen used by BitPaymer in 2018.

Evil Corp are selective in terms of the infrastructure they target when deploying their ransomware. Typically, they hit *file servers, database services, virtual machines and cloud environments*. Of course, these choices will also be heavily influenced by what we may term their 'business model' – which also means they should be able to disable or disrupt backup applications and related infrastructure. This increases the time for recovery for the victim, or in some cases due to unavailability of offline or offsite backups, prevents the ability to recover at all.

It is interesting that the group has *not appeared to have engaged in extensive information stealing* or threatened to publish information about victims in the way that the DoppelPaymer and many other targeted ransomware operations have. We assess that the probable reason for not leaking victim information is the unwanted attention this would draw from law enforcement and the public.

3. Distribution

While many things have changed in the TTPs of Evil Corp recently, one very notable element has not changed, the distribution via the *SocGholish* fake update framework. This framework is still in use although it is now used to directly distribute a custom *CobaltStrike* loader, described in 4.1, rather than Dridex as in the past years. One of the more notable features of this framework is the evaluation of wether a compromised victim system is part of a larger network, as a sole enduser system is of no use to the attackers. The SocGholish JavaScript bot has access to information from the system itself as it runs under the privileges of the browser user. The bot collects a large set of information and sends that to the SocGholish server side which, in turn, returns a payload to the victim system. Other methods of distribution also appear to still be in use, but we have not been able to independently verify this at the time of writing.

4. Technical Analysis

4.1 CobaltStrike payloads

The CobaltStrike payloads are embedded inside two types of PowerShell scripts. The first type (which targets Windows 64-bit only) decodes a base64 payload twice and then decrypts it using the AES algorithm in CBC mode. The AES key is derived by computing the SHA256 hash of the hard-coded string '*saN9s9pNlD5nJ2EyEd4rPym68griTOMT*' and the initialisation vector (IV), is derived from the first 16 bytes of the twice base64-decoded payload. The script converts the decrypted payload (a base64-encoded string) to bytes and allocates memory before executing it.

The second type is relatively simpler and includes two embedded base64-encoded payloads, an injector and a loader for the CobaltStrike payload. It appears that both the injector and the loader are part of the '*Donut*' project [3].

An interesting behaviour can be spotted in the CobaltStrike payloads that are delivered from the second type of PowerShell scripts. In these, the loader has been modified with the purpose of detecting CrowdStrike software (*Figure 1*). If the `C:\\Program Files\\CrowdStrike` directory exists, then the '*FreeConsole*' Windows API is called after loading the CobaltStrike payload. Otherwise, the 'FreeConsole' function is called before loading the CobaltStrike beacon. It is assumed that this is an attempt to bypass CrowdStrike's endpoint solution, although it still unclear if this is the case.

```
CS_found_flag = 0;
if ( GetFileAttributesA("C:\\Program Files\\CrowdStrike") == -1 )
  FreeConsole();
else
  CS_found_flag = 1;
malware_load_Cobalt();
while ( 1 )
{
  Sleep(0x270Eu);
  if ( CS_found_flag )
    FreeConsole();
  CS_found_flag = 0;
}
```

Figure 1: Decompilation showing CrowdStrike specific detection logic

4.2 The Crypter

WastedLocker is protected with a custom crypter, referred to as ***CryptOne*** by Fox-IT InTELL. On examination, the code turned out to be very basic and used also by other malware families such as: *Netwalker*, *Gozi ISFB v3*, *ZLoader* and *Smokeloader*.

The crypter mainly contains junk code to increase entropy of the sample and hide the actual code. We have found 2 crypter variants with some code differences, but mostly with the same logic applied.

The first action performed by the crypter code is to check some specific registry key. In the variants analysed the registry key is either: **interface\{b196b287-bab4-101a-b69c-00aa00341d07}** or **interface\{aa5b6a80-b834-11d0-932f-**

00a0c90dcaa9}. These keys relate to the ***UCOMIEnumConnections*** Interface and the ***IActiveScriptParseProcedure32*** interface respectively. If the key is not detected, the crypter will enter an infinite loop or exit, thus it is used as an anti-analysis technique.

In the next step the crypter allocates a memory buffer calling the *VirtualAlloc* API. A while loop is used to join a series of data blobs into the allocated buffer, and the contents of this buffer are then decrypted with an XOR based algorithm. Once decrypted, the crypter jumps into the data blob which turns out to be a shellcode responsible for decrypting the actual payload. The shellcode copies the encrypted payload into another buffer allocated by calling the *VirtualAlloc* API, and then decrypts this with an XOR based algorithm in a similar way to that described above. To execute the payload, the shellcode replaces the crypter's code in memory with the code of the payload just decrypted, and jumps to its entry point.

As noted above, we have observed this crypter being used by other malware families as well. Related information and IOCs can be found in the Appendix.

4.3 WastedLocker Ransomware

WastedLocker aims to encrypt the files of the infected host. However before the encryption procedure runs, WastedLocker performs a few other tasks to ensure the ransomware will run properly.

First, Wastedlocker decrypts the strings which are stored in the .bss section and then calculates a *DWORD* value that is used later for locating decrypted strings that are related to the encryption process. This is described in more detail in the *String encryption* section. In addition, the ransomware creates a log file ***lck.log*** and then sets an exception handler that creates a crash dump file in the *Windows temporary folder* with the filename being the ransomware's binary filename.

If the ransomware is not executed with administrator rights or if the infected host runs Windows Vista or later, it will attempt to elevate its privileges. In short, WastedLocker uses a well-documented *UAC bypass method* [1] [2]. It chooses a random file (EXE/DLL) from the Windows system32 folder and copies it to the %APPDATA% location under a different hidden filename. Next, it creates an alternate data stream (*ADS*) into the file named bin and copies the ransomware into it. WastedLocker then copies winsat.exe and winmm.dll into a newly created folder located in the Windows temporary folder. Once loaded, the hijacked DLL (*winmm.dll*) is patched to execute the aforementioned ADS.

The ransomware supports the following command line parameters (*Table 1*):

Parameter	Purpose
-r	i. *Delete shadow copies* ii. *Copy the ransomware binary* file to %windir%\system32 and *take ownership* of it (takeown.exe /F filepath) and *reset the ACL permissions* iii. *Create and run a service*. The service is deleted once the encryption process is completed.
-s	Execute service's entry
-p directory_path	Encrypt files in a specified directory and then proceed with the rest of the files in the drive
-f directory_path	Encrypt files in a specified directory

Table 1 – WastedLocker command line parameters

It is also worth noting that in case of any failure from the first two parameters (*-r* and *–s*), the ransomware proceeds with the encryption but applies the following registry modifications in the registry key Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap:

Name	Modification
ProxyBypass	Deletes this key
IntranetName	Deletes this key
UNCAsIntranet	Sets this key to 0
AutoDetect	Sets this key to 1

Table 2 – Registry keys

The above modifications apply to both 32-bit and 64-bit systems and is possibly done to ensure that the ransomware can access remote drives. However, a bug is included in the architecture identification code. The ransomware authors use a well-known method to identify the operating system architecture. The ransomware reads the memory address **0x7FFE0300** (*KUSER_SHARED_DATA*) and checks if the pointer is zero. If it is then the 32-bit process of the ransomware is running in a Windows 64-bit host (*Figure 2*). The issue is that this does not work on *Windows 10* systems.

```
v9 = (&unk_100B2F8 + dword_100A644);      // Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap
malware_change_reg_zonemap(0, (&unk_100B2F8 + dword_100A644));
if ( !MEMORY[0x7FFE0300] )
  malware_change_reg_zonemap(0x100, v9);  // same but for 64bit
```

Figure 2: Decompilation showing method used to identify operating system architecture

Additionally, WastedLocker chooses a random name from a generated name list in order to generate filename or service names. The ransomware creates this list by reading the registry keys stored in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control and then separates their names whenever a capital letter is found. For example, the registry key AppReadiness will be separated to two words, App and Readiness.

4.4 Strings Encryption

The strings pertaining to the ransomware are encrypted and stored in the .bss section of the binary file. This includes the ransom note along with other important information necessary for the ransomware's tasks. The strings are decrypted using a key that combined the size and raw address of the .bss section, as well as the ransomware's compilation timestamp.

The code's authors use an interesting method to locate the encrypted strings related to the encryption process. To locate one of them, the ransomware calculates a checksum that is looked up in the encrypted strings table. The checksum is derived from both a constant value that is unique to each string and a fixed value, which are bitwise XORed. The encrypted strings table consists of a struct like shown below for each string.

```
struct ransomware_string
{
WORD total_size; // string_length + checksum + ransom_string
WORD string_length;
DWORD Checksum;
BYTE[string_length] ransom_string;
};
```

4.5 Encryption Process

The encryption process is quite straightforward. The ransomware targets the following drive types:

- Removable
- Fixed
- Shared
- Remote

Instead of including a list of extension targets, WastedLocker includes a list of directories and extensions to exclude from the encryption process. Files with a size less than *10 bytes* are also ignored and in case of a large file, the ransomware encrypts them in blocks of *64MB*.

Once a drive is found, the ransomware starts searching for and encrypting files. Each file is encrypted using the *AES* algorithm with a newly generated AES key and IV (*256-bit in CBC mode*) for each file. The AES key and IV are encrypted with an embedded public *RSA key* (*4096 bits*). The RSA encrypted output of key material is converted to base64 and then stored into the ransom note.

For each encrypted file, the ransomware creates an additional file that includes the ransomware note. The encrypted file's extension is set according to the targeted organisations name along with the prefix **wasted** (hence the name we have gave to this ransomware). For example, ***test.txt.orgnamewasted*** (encrypted data) and ***test.txt.orgnamewasted_info*** (ransomware note). The ransomware note and the list of excluded directories and extensions is available in the Appendix. Finally, once the encryption of each file has been completed, the ransomware updates the log file with the following information:

- Number of targeted files
- Number of files which were encrypted
- Number of files which were not encrypted due to access rights issues

4.6 WastedLocker Decrypter

During our analysis, we managed to identify a decrypter for WastedLocker. The decrypter requires administrator privileges and similarl to the encryption process, it reports the number of files which were successfully decrypted (*Figure 3*).

Figure 3: Command line output of the decrypter of WastedLocker

References

1. hxxps://medium.com/tenable-techblog/uac-bypass-by-mocking-trusted-directories-24a96675f6e
2. hxxps://github.com/hfirefox/UACME
3. hxxps://github.com/TheWover/donut/

Appendix

Ransom note

```
*ORGANIZATION_NAME*

YOUR NETWORK IS ENCRYPTED NOW

USE *EMAIL1* | *EMAIL2* TO GET THE PRICE FOR YOUR DATA

DO NOT GIVE THIS EMAIL TO 3RD PARTIES

DO NOT RENAME OR MOVE THE FILE

THE FILE IS ENCRYPTED WITH THE FOLLOWING KEY:
[begin_key]*[end_key]
KEEP IT
```

*Excluded extensions (in addition to **orgnamewasted** and **orgnamewasted_info**)*

```
*\ntldr
*.386
*.adv
*.ani
*.bak
*.bat
*.bin
*.cab
*.cmd
*.com
*.cpl
*.cur
*.dat
*.diagcab
*.diagcfg
*.dll
*.drv
*.exe
*.hlp
*.hta
*.icl
*.icns
*.ics
*.idx
*.ini
*.key
*.lnk
*.mod
*.msc
*.msi
*.msp
*.msstyles
*.msu
*.nls
*.nomedia
*.ocx
*.ps1
*.rom
*.rtp
*.scr
*.sdi
*.shs
*.sys
*.theme
*.themepack
*.wim
*.wpx
*\bootmgr
*\grldr
```

Excluded directories

```
*\$recycle.bin*
*\appdata*
*\bin*
*\boot*
*\caches*
*\dev*
*\etc*
*\initdr*
*\lib*
*\programdata*
*\run*
*\sbin*
*\sys*
*\system volume information*
*\users\all users*
*\var*
*\vmlinuz*
*\webcache*
*\windowsapps*
c:\program files (x86)*
c:\program files*
c:\programdata*
c:\recovery*
c:\users\ %USERNAME%\appdata\local\temp*
c:\users\ %USERNAME%\appdata\roaming*
c:\windows*
```

IoCs

IoCs related to targeted ransomware attacks are a generally misunderstood concept in the case of targeted ransomware. Each ransomware victim has a custom build configured or compiled for them and so the knowing the specific hashes used against historic victims does not provide any protection at all. Even if behavioural patterns of the ransomware or network related indicators of the ransomware stage are given (should they exist), it is arguable whether detection of the attack at that stage would allow prevention of the actual attack. We do include known ransomware hashes here; however, please note that these are for RESEARCH PURPOSES ONLY. Blocking files based on these file attributes in any endpoint protection product will not provide any value.

At Fox-IT we focus mainly on detection of the initial stages of such attacks (such as the initial stage of infection) by detecting the various methods of infection delivery as well as the lateral movement stage which typically involves scanning, exploitation and/or credential dumping. Providing these IoCs to the wider public would, however, be counterproductive as the threat actors would simply change these methods or work around the indicators. However, we have included some of them to provide historical as well as current protection or detection against this particular threat, and provide a better understanding of this threat actor. It is also hoped this information will help other organisations to conduct further research into this particular threat.

CobaltStrike

This particular set of domains is used as C&C by the group for CobaltStrike lateral movement activity, using a custom loader, Note that in 2020 the group has completely switched to using CobaltStrike and is no longer using the Empire PowerShell framework as it is no longer being updated by the original creators.

CobaltStrike *C&C* Domains

```
adsmarketart.com
advancedanalysis.be
advertstv.com
amazingdonutco.com
cofeedback.com
consultane.com
dns.proactiveads.be
mwebsoft.com
rostraffic.com
traffichi.com
typiconsult.com
websitelistbuilder.com
```

CobaltStrike *Beacon* config

```
SETTING_PROTOCOL: short: 8 (DNS: 0, SSL: 1)
SETTING_PORT: short: 443
SETTING_SLEEPTIME: int: 45000
SETTING_MAXGET: int: 1403644
SETTING_JITTER: short: 37
SETTING_MAXDNS: short: 255
SETTING_PUBKEY: ''
SETTING_PUBKEY SHA256:
14f2890a18656e4e766aded0a2267ad1c08a9db11e0e5df34054f6d8de749fe7
ptr SETTING_DOMAINS: websitelistbuilder.com,/jquery-3.3.1.min.js
ptr SETTING_USERAGENT: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like
Gecko
ptr SETTING_SUBMITURI: /jquery-3.3.2.min.js
SETTINGS_C2_RECOVER:
    print: True
    append: 1522
    prepend: 84
    prepend: 3931
    base64url: True
    mask: True
SETTING_C2_REQUEST (transform steps):
   _HEADER: Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
   _HEADER: Referer: http://code.jquery.com/
   _HEADER: Accept-Encoding: gzip, deflate
   BUILD: metadata
   BASE64URL: True
   PREPEND: __cfduid=
   HEADER: Cookie
SETTING_C2_POSTTREQ (transform steps):
   _HEADER: Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
   _HEADER: Referer: http://code.jquery.com/
   _HEADER: Accept-Encoding: gzip, deflate
   BUILD: metadata
   MASK: True
   BASE64URL: True
   PARAMETER: __cfduid
   BUILD: output
   MASK: True
   BASE64URL: True
   PRINT: True
ptr DEPRECATED_SETTING_SPAWNTO:
ptr SETTING_SPAWNTO_X86: %windir%\syswow64\rundll32.exe
ptr SETTING_SPAWNTO_X64: %windir%\sysnative\rundll32.exe
ptr SETTING_PIPENAME:
SETTING_CRYPTO_SCHEME: short: 0 (CRYPTO_LICENSED_PRODUCT)
SETTING_DNS_IDLE: int: 1249756273
SETTING_DNS_SLEEP: int: 0
ptr SETTING_C2_VERB_GET: GET
ptr SETTING_C2_VERB_POST: POST
SETTING_C2_CHUNK_POST: int: 0
SETTING_WATERMARK: int: 305419896 (0x12345678)
SETTING_CLEANUP: short: 1
SETTING_CFG_CAUTION: short: 0
ptr SETTING_HOST_HEADER:
SETTING_HTTP_NO_COOKIES: short: 1
SETTING_PROXY_BEHAVIOR: short: 2
```

```
SETTING_EXIT_FUNK: short: 0
SETTING_KILLDATE: int: 0
SETTING_GARGLE_NOOK: int: 154122
ptr SETTING_GARGLE_SECTIONS:
'`\x02\x00Q\xfd\x02\x00\x00\x00\x03\x00\xc0\xa0\x03\x00\x00\xb0\x03\x000\xce\x03'
SETTING_PROCINJ_PERMS_I: short: 4
SETTING_PROCINJ_PERMS: short: 32
SETTING_PROCINJ_MINALLOC: int: 17500
ptr SETTING_PROCINJ_TRANSFORM_X86: '\x02\x90\x90'
ptr SETTING_PROCINJ_TRANSFORM_X64: '\x02\x90\x90'
ptr SETTING_PROCINJ_STUB: *p?'??????]
ptr SETTING_PROCINJ_EXECUTE: BntdllRtlUserThreadStart
SETTING_PROCINJ_ALLOCATOR: short: 1
Deduced metadata:
 WANTDNS: False
 SSL: True
 MAX ENUM: 55
 Version: CobaltStrike v4.0 (Dec 5, 2019)
```

*Custom **CobaltStrike** loader samples (sha256 hashes):*

```
2f72550c99a297558235caa97d025054f70a276283998d9686c282612ebdbea0
389f2000a22e839ddafb28d9cf522b0b71e303e0ae89e5fc2cd5b53ae9256848
3dfb4e7ca12b7176a0cf12edce288b26a970339e6529a0b2dad7114bba0e16c3
714e0ed61b0ae779af573dce32cbc4d70d23ca6cfe117b63f53ed3627d121feb
810576224c148d673f47409a34bd8c7f743295d536f6d8e95f22ac278852a45f
83710bbb9d8d1cf68b425f52f2fb29d5ebbbd05952b60fb3f09e609dfcf1976c
91e18e5e048b39dfc8d250ae54471249d59c637e7a85981ab0c81cf5a4b8482d
adabf8c1798432b766260ac42ccdd78e0a4712384618a2fc2e3695ff975b0246
b0354649de6183d455a454956c008eb4dec093141af5866cc9ba7b314789844d
bc1c5fecadc752001826b736810713a86cfa64979b3420ab63fe97ba7407f068
c781c56d8c8daedbed9a15fb2ece165b96fdda1a85d3beeba6bb3bc23e917c90
c7cde31daa7f5d0923f9c7591378b4992765eac12efa75c1baaaefa5f6bdb2b6
f093b0006ef5ac52aa1d51fee705aa3b7b10a6af2acb4019b7bc16da4cabb5a1
```

.NET injector (*Donut*) (sha256 hash):

```
6088e7131b1b146a8e573c096386ff36b19bfad74c881ca68eda29bd4cea3339
```

Gozi ISFB v2

This particular set contains C&C domains, bot version, Group ID, RSA key and Serpent encryption keys for 2 Gozi variants used for persistence in victim networks during 2020.

Gozi *C&C* Domains

```
bettyware.xyz
celebratering.xyz
fakeframes.xyz
gadgetops.xyz
hotphonecall.xyz
justbesarnia.xyz
kordelservers.xyz
tritravlife.xyz
veisllc.xyz
wineguroo.xyz
```

Gozi *versions*

```
217119
217123
```

Gozi *Group ID*

```
30000
```

Gozi *RSA key*

```
00020000BEA9877343AD9F6EA8E122A5A540C071E96AB5E0C8D73991BFACB8D7867125966C60153EB1
```

Gozi *serpent network encryption keys:*

```
8EzkwaSgkg565AyQ
eptDZELKvZUseoAH
GbdG3H7PgSVEme2r
RQ5btM2UfoCHAMKN
```

Gozi *samples* (sha256 hashes)

```
5706e1b595a9b7397ff923223a6bc4e4359e7b1292eaed5e4517adc65208b94b
ba71ddcab00697f42ccc7fc67c7a4fccb92f6b06ad02593a972d3beb8c01f723
c20292af49b1f51fac1de7fd4b5408ed053e3ebfcb4f0566a2d4e7fafadde757
cf744b04076cd5ee456c956d95235b68c2ec3e2f221329c45eac96f97974720a
```

WastedLocker samples (sha256 hashes)

```
5cd04805f9753ca08b82e88c27bf5426d1d356bb26b281885573051048911367
887aac61771af200f7e58bf0d02cb96d9befa11deda4e448f0a700ccb186ce9d
8897db876553f942b2eb4005f8475a232bafb82a50ca7761a621842e894a3d80
bcdac1a2b67e2b47f8129814dca3bcf7d55404757eb09f1c3103f57da3153ec8
e3bf41de3a7edf556d43b6196652aa036e48a602bb3f7c98af9dae992222a8eb
ed0632acb266a4ec3f51dd803c8025bccd654e53c64eb613e203c590897079b3
```

The following IoCs are specifically related to the crypter used by Evil Corp, which we refer to as CryptOne. Given that CryptOne is used by more malware families and variations than just those related to Evil Corp it is likely that CryptOne is a third party service.

List of metadata extracted from ***Gozi ISFB v3*** samples

Bot version	3.00.854
RSA key	00040000C3DC07D4E1AC941077214371F45B5FD- DDF389654D0851D66809BC989ABA850C27D3718D195EE1388087F21F- FE759184C185959D1AB5DBC40C3D94C88F46FE8AA1- CA94CB07CF110866559456F9DF6F1EAE9C3002F1A257A2F99E3EB3E- F6C727516BA65CE56C82E23CBBE87E1EE95F34DD7D- C0D07B7C1F57B71BC49DC35DEB2- CAB000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000010001
Group IDs	202004081 202004091 202004141 202004231 202005041 80000
Serpent keys	1qzRaTGYO5dpREYI 8JbpEEfNYPlY0AN4 dLwZ7QwI57AkzZEl UEwFH6L9iBbdJxAf uIIXQ4B05dT8AytD vuARb2EPotEtfAX2 Z6fiC4XCvQmfkgua
C&Cs	hxxps://devicelease.xyz hxxps://guiapocos.xyz hxxps://ludwoodgroup.xyz hxxps://respondcritique.xyz hxxps://triomigratio.xyz hxxps://uplandcaraudio.xyz hxxps://woofwoofacademy.xyz

ZLoader (*MD5: fb95561e8ed7289d015e945ad470e6db*)

RC4 key	das32hfkAN3R2TCS
Botnet name	pref
Nonce	0x7
Static config RC4 key	kyqvkjlpclbcnagbhiwo
Version	1.2.22.0
C&Cs	hxxp://advokat-hodonin.info/gate.php hxxp://penaz.info/gate.php
Binary Distribution	hxxp://paiolets.com/install.exe

Netwalker ransomware (*MD5: 198b2443827f771f216cd8463c25c5d8*)

SmokeLoader (*MD5: 2143d279be8d1bb4110b7ebe8dc3afbc*)

RC4 send	0x69A84992
RC4 recv	0x5D7C6D5B
C&Cs	hxxp://flablenitev.site/index.php hxxp://lendojekam.xyz/index.php hxxp://lgrarcosbann.club/index.php hxxp://lpequdeliren.fun/index.php hxxp://transvil2.xyz/index.php
Binary Distribution	hxxps://szn.services/1.exe hxxps://utenti.info/1.exe hxxps://utenti.live/1.exe

SecTool checker (*MD5: b33753fae7bd1e68e0b1cc712b5fb867*)

We have found a sample crypted by the CryptOne crypter as used by WastedLocker, which is capable of detecting/disabling a list of security software. It is believed that this tool is used during ransomware deployment, but we have no specific evidence that it was used by Evil Corp. However in the past we have seen execution of commands listed in the tool to disable Microsoft Windows Defender.

List of Registry Keys checked	Software\ESET SYSTEM\ControlSet001\Services\MBAMService
List of Mutex checked	00082fbb-a419-43f4-bd80-e3631ebbf4c8 069e4409-bd54-4a1f-8e37-49da2cf6a537 0ca9a8d3-01bf-4f9e-bfc7-7eb51e67e0c4 12a2c0fc-00d2-4614-b4ae-c18eb500a088 138be83c-2a52-4c31-9ee8-bfd4eac53d72 15417794-7485-46f6-9965-d34730ea0f48 168cb052-69eb-45be-be07-d4f323dc67d6 16ed8dab-ee6b-44ea-8cea-31c66d6864b9 172821eb-729d-4307-a56f-63063b2677de 17689d7a-89bf-4e2a-a49c-9e4e5a51a9d7 197a1689-8bb1-4fcd-80e9-32b86e3751f5 1a379834-6135-41e7-9cf7-e79a9f705fbc 1cce886d-1841-4e18-963b-15f2e90a3c44 1e8e5806-2e99-4002-b62c-7a78a6641874 1f1769de-42fa-4883-b37c-f0de488de557 240187f4-b097-4a3c-a6fa-2ca5b1e0b373 25f07256-3b46-4531-aa3e-e1729d9aa7cb 274f61dd-3fed-4bfe-9aa6-8a012339a41f 27a0f05f-41fa-43f1-86b9-7e48bde3d716 2a942be2-9252-4d60-9483-3651a92192a5 2c0c5f0d-6ad7-4c97-b1a8-2c706d03a4f8 39309b80-cef5-4ce1-b215-0719723c4c30 3c159c86-0e90-47d1-ad37-788c00ba2948 3f78ca48-011c-4ffb-abfa-c9f659e4a820 3ffd4715-4991-4bc8-9c51-2e3aeb6e737e 3G1S91V5ZA5fB56W 48353b4f-51f9-4961-bcc1-c8d5163a8978 4d6a57e9-e692-4da2-8ba8-adb25645e4b8 4e1ac580-d3cf-4961-81eb-072dff249c17 4e5e7d5e-a1fe-4de7-ad53-5f4aaecd7402 55731fe5-97ad-47dc-953f-37a8aca1451b 5962654a-a395-4714-96f2-2419ab2172bf 5e76294a-2787-4ae2-9ddc-b792b0c45ec2 60f8896b-a437-4e79-9e29-96522ca88c4c 62e64ec9-d662-4595-bf77-634764dcf810 67f4e0eb-54cc-4779-b3c3-fe277c8478ae 6b264507-ba91-4d85-86c9-1e827315cbe0 722cbc3c-acc8-4296-a8dd-7d06e5ca7d57 7eb5ccec-3fd7-4826-b681-02a6129aa108 81baf7c7-3010-49b9-9f56-d53fca06c04d 85e6784c-7904-41ee-99b4-8b286e19da70 8AZB70HDFK0WOZIZ 8f1a37f6-9cff-447e-a00c-cb19512de134 9b765102-98e7-43e2-a003-f8cbdfab8a64 9f093bf8-480b-414c-a8e8-5d9c6da83576

9f7e0dc2-bc5c-497e-aa70-f8072e71550c
ab7d92f2-968a-461e-9da6-e569dedb0a91
ARScenes
ASUSNet20
ATYNKAJP30Z9AQ
b22d1dd8-e3ea-4764-ba9b-0ebf41fddee7
b3e32042-d969-43d1-b20c-bcf8da5ba436
beb41e13-5e33-450f-a9c5-3e5a382d224d
BiosChecksumChecker
bitcoreguard
BlueEye
c3c2a8b3-fc8a-4fe3-8f24-6f2a757a5012
ca1b68fd-56d5-4355-94b2-ed6ab0857890
CBKZiOPASRHKL
CDNetStreamer2.r05
cf3573d5-bf4f-4094-bbea-ced8efde2257
China1839099
China4150039
CryptoMaxima
D1JozWrldD
d86a1229-2cb7-409b-a3de-5366eec3db90
d8ba5865-ac00-4df1-8437-eb144077e031
dad17f2e-5f30-4313-b1c3-5ae8c2149757
dec0f5aa-1fd1-458f-916c-693887610891
e3024a8f-3f2b-4e06-ac36-0997c1090d00
ed3a7d1d-ed6f-4c8f-86d4-44dcde3b32f8
f1e7974a-30e1-423c-9745-bbb7ff7dbf71
f378f238-6503-4544-8e43-cbe4bbf3615e
f967041f-20dd-4d31-a34a-f5e04bdfdf7b
FamilyWeekend
fbac80bd-ba6a-4cd5-92d9-3a31a87f7af6
fda765a3-b5a2-4417-9097-3b18dc6fe6fb
fe711d65-f31a-4c22-a12f-cec65d231941
FixLCD
FMPsDSCVol
FoloDrite
Hk4kKLL0ZAF8a
HTTPBalancer_v2.15
I0N8129AZR1A
ImageCreator_v4.2
InRAMQueue
IntelBIOSReader
IwS01003993
JerkPatrol
JKLSXX1ZA1QRLER
KDOWEtRVAB
LenovoSuite
MaverickMeerkat
MDISequencer
MK5Cheats
MLIXNJ9AEGPSE
MLIXNJAEGPSE

	MovieFinder N800HANOI NattyNarwhal NeoNetPlasma NeonRhythmbox NetRegistry NetworkLighter NHO9AZB7HDK0WAZMM NMOZAQcxzER NNDRIOZ8933 OMXBJSJ3WA1ZIN OneiricOcelot OnlineShopFinder P79zA00FfF3 PCV5ATULCN PJOQT7WD1SAOM PrecisePangolin PSHZ73VLLOAFB QOSUser2.r10 QuantalQuetzal RaringRingtail RaspberryManualViewer RedParrot RouteMatrix SoloWrite sqlcasheddbm SSDOptimizerV13 StreamCoder1.0 Tropic819331 UEFIConfig UtopicUnicorn VHO9AZB7HDK0WAZMM VideoBind VirginPoint VirtualDesktopKeeper VirtualPrinterDriver VividVervet VRK1AlIXBJDA5U3A WinDuplicity WireDefender wwallmutex
Commands executed	C:\Windows\system32\WindowsPowershell\v1.0\powershell.exe Set-MpPreference -DisableBehaviorMonitoring $true ; Set-MpPreference -MAPSReporting 0 ; Set-MpPreference -ExclusionProcess rundll32.exe ; Set-MpPreference -ExclusionExtension dll C:\Windows\System32\netsh.exe advfirewall firewall add rule name="Rundll32" dir=out action=allow protocol=any program="C:\Windows\system32\rundll32.exe"