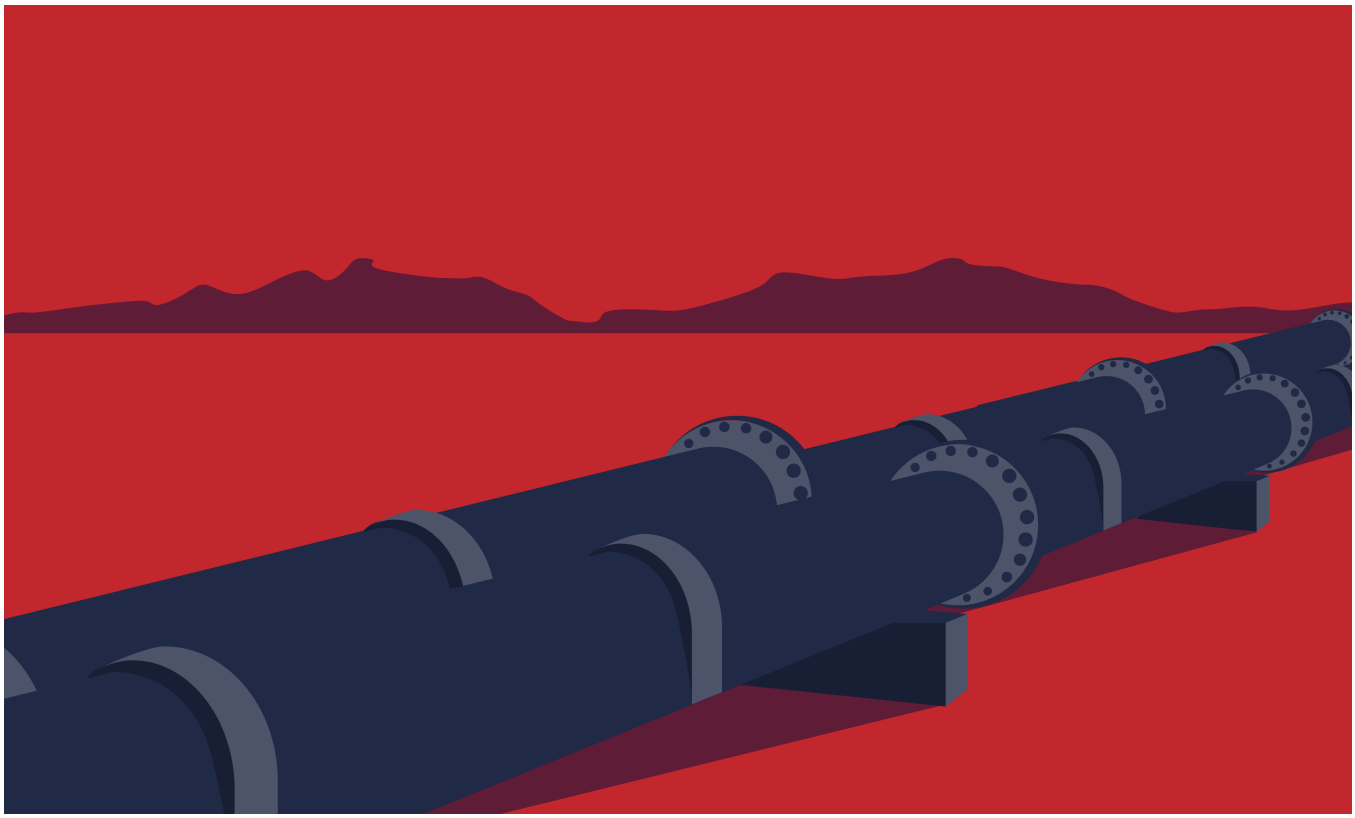
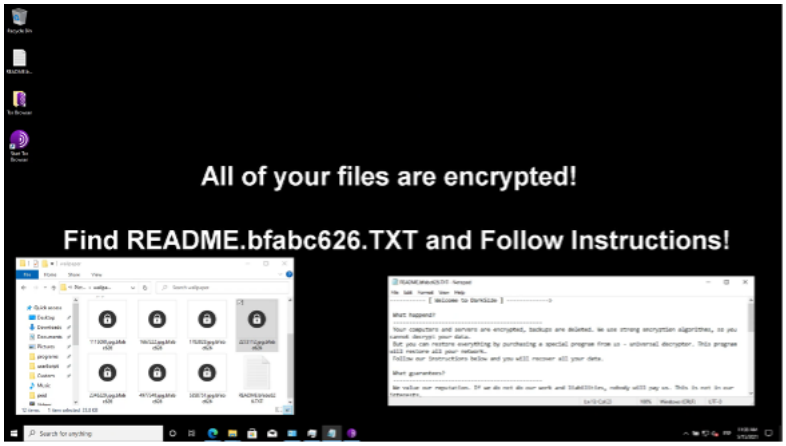
Inside DarkSide, the ransomware that attacked Colonial Pipeline

metabaseq.com/recursos/inside-darkside-the-ransomware-that-attacked-colonial-pipeline





Executive Summary

On May 7th, 2021, Colonial Pipeline reported that its digital infrastructure had been compromised due to a cyberattack, and as a precautionary measure, it would suspend its services until the severity of the situation was determined.

Colonial is the largest pipeline operator in the U.S. and transports more than 3 million barrels of gasoline, diesel, and jet fuel between the U.S. Gulf Coast and the New York Harbor area.

This cyberattack utilized ransomware which encrypted critical information systems and requested payment to recover the information.

Researchers have allegedly attributed this attack to the DarkSide group, whose modus operandi known as Ransomware-as-a-Service (RaaS), involves not only the encryption of information with Salsa20, using an RSA-1024 public key, but also the exfiltration of information used for payment negotiation.

DarkSide has affected numerous organizations in various sectors, including industry, legal, insurance, healthcare, and energy. However, according to a January 27th, 2021 publication on DarkSide's website about its rules of use, affiliated individuals cannot target the funeral service sector, hospitals, nursing homes, or companies distributing the COVID-19 vaccine.

To pay the demands, the organization promotes the use of Monero, a cryptocurrency distinguished by its anonymity and superior security compared to other currencies in the market. DarkSide members use an administrative panel via The Onion Router (TOR) to communicate with victims and manage the delivery of the ransomware to its victims.

Government entities have already identified this specific version of the ransomware, and you can find the Indicators of Compromise (IOC) at the end of the blog.

Colonial has already faced a series of extortions so that the stolen information is not published, and it is presumed that the payment has already been made for an amount close to $5 million USD, reported zdnet. According to researcher Brian Krebs, the group's servers have already been seized, as well as one of the cryptocurrency accounts used to pay its affiliates.

In this blog, we analyze in detail one of the DarkSide samples used during the attack, providing TTPs (Tactics, Techniques, and Procedures) and IOCs (Indicators of Compromise) that serve for monitoring, detection, and eradication strategies of this type of cyberthreats.

Main findings

- DarkSide and SODINOKIBI (REvil),although not identical, share the same *modus operandi*.
- DarkSide does not attack computers with the language of some of the countries formerly known as the Republics of the Soviet Union.
- DarkSide's infrastructure is hosted with a Russian provider.
- Encryption is very fast due toa one-thread-per-processor implementation.
- It encrypts files on hard disks, removable drives, and network drives.

Initial infection

Malicious actors use techniques such as spear-phishing with malicious links to try to infect their victims. In turn, they obtain legitimate credentials through SQL Injection attacks by accessing credentials stored in databases or even in different places on the dark web where compromised credentials of organizations are listed. According to Mandiant, in the case of Colonial, the attackers used valid VPN credentials of an employee who did not have multi-factor authentication enabled, and remote access was achieved without restrictions.

DarkSide's ransomware behavior

Once it has been possible to access a computer of the attacked organization, some versions occupy a Dropper, where their job is to decrypt DarkSide in memory (in some cases with a simple XOR) and execute it.

DarkSide:

Type: Win32 Binary - Delphi

MD5: 222792d2e75782516d653d5cccfcf33b

Name(s): net.bin, Darkside.exe

Compilation Timestamp: Dec 15, 2020 22:26:41

First time viewed: Dec 29, 2020 18:22:15

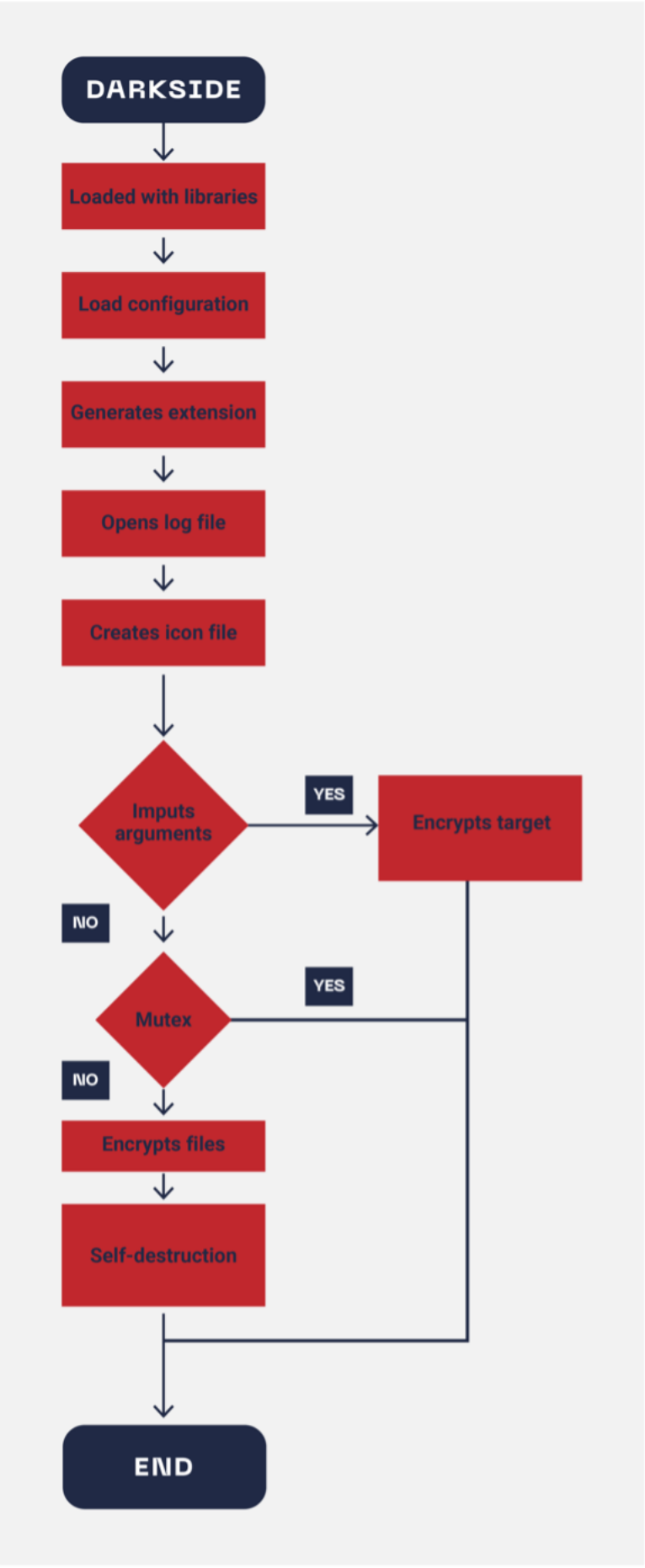
First time analyzed: Dec 28, 06:37:28

Last time analyzed: May 14, 2021 00:06:50

Country of first entry in VT: India

General malware scheme

The following diagram shows a general outline of the ransomware execution flow:


DARKSIDE
Loaded with libraries
Load configuration
Generates extension
Opens log file
Creates icon file
Imputs arguments
YES
Encrypts target
NO
Mutex
YES
NO
Encrypts files
Self-destruction
END

Configuration loaded into memory

The entire malware configuration is loaded (and compressed) in the .data section in memory with a length of 2745 bytes.

When the decompression algorithm is applied, 159040 bytes are obtained with a table containing the global configuration of the ransomware (See Figure 4):

- Whitelist files
- Whitelist extensions
- Fast encryption files (these files are encrypted with a faster encryption algorithm due to their larger size)

  Name of processes to stop:
  - vmcompute.exe
  - vmms.exe
  - vmwp.exe
  - svchost.exe
  - TeamViewer.exe
  - explorer.exe

- Name of services to stop: Many of them ensure that the document is not being occupied and can then be encrypted: **sql, oracle, ocssd, dbsnmp, synctime, agntsvc, isqlplussvc, xfssvccon, mydesktopservice, ocautoupds, encsvc, firefox, tbirdconfig, mydesktopqos, ocomm, dbeng50, sqbcoreservice, excel, infopath, msaccess, mspub, onenote, outlook, powerpnt, steam, thebat, thunderbird, visio, winword, wordpad, notepad**
- Name of the server where the stolen data will be sent to
- The message written on the screen background
- Ransom message leaving infected directories (with an embedded key

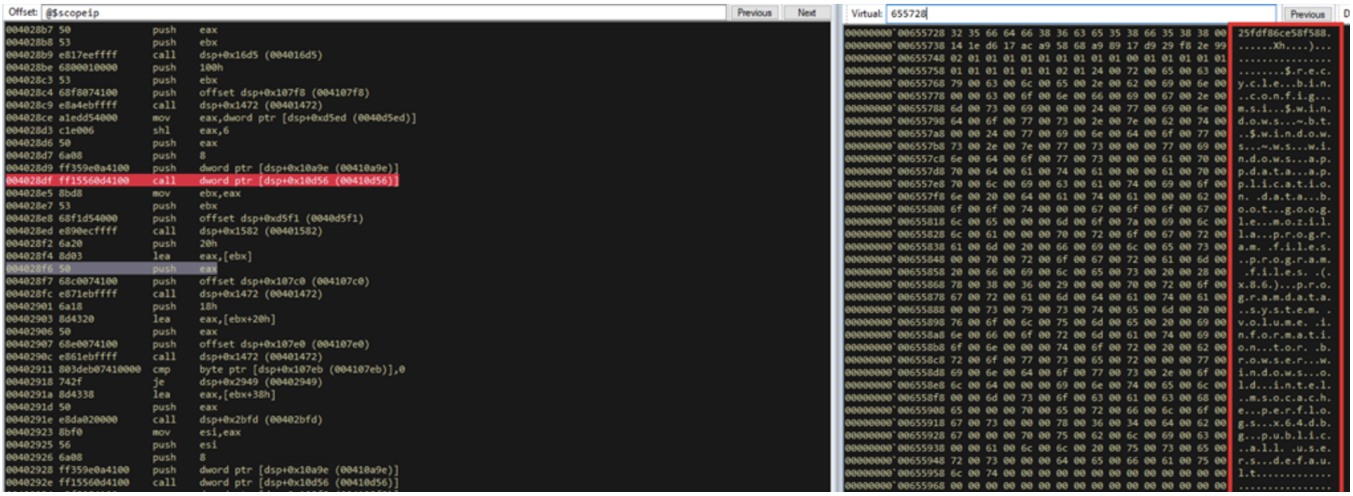


Figure 1: Darkside configurations loaded into memory

List of whitelisted countries and languages

The ransomware makes a call to **GetSystemDefaultUILanguage**, which returns a Language Code Identifier (LCID) of the system and also executes **GetUserDefaultLangID** to get the user's language.

The following codes correspond to the countries that fall within the whitelist. In case the system contains one of these LCIDs, the ransomware will not affect any files.

Russian (Russia)	**ru-RU**	**0x419**
Ukrainian (Ukraine)	uk-UA	0x422
Belarusian (Belarus)	be-BY	0x423
Tajik (Cyrillic, Tajikistan)	tg-Cyrl-TJ	0x428
Armenian (Armenia)	hy-AM	0x42B
Azerbaijani (Latin, Azerbaijan)	az-Latn-AZ	0x42C
Georgian (Georgia)	ka-GE	0x437
Kazakh (Kazakhstan)	kk-KZ	0x43F

Kyrgyz (Kyrgyzstan)	ky-KG	0x440
Turkmen (Turkmenistan)	tk-TM	0x442
Uzbek (Latin, Uzbekistan)	uz-Latn-UZ	0x443
Tatar (Russia)	tt-RU	0x444
Romanian (Moldova)	ro-MD	0x818
Russian (Moldova)	ru-MD	0x819
Azerbaijani (Cyrillic, Azerbaijan)	az-Cyrl-AZ	0x82C
Uzbek (Cyrillic, Uzbekistan)	uz-Cyrl-UZ	0x843
Arabic (Syria)	ar-SY	0x2801

Information collected prior to data encryption

Before starting file encryption, certain information about the infected host is collected:

- lang: Language configured in the system
- username: Current user name
- hostname: Name of the machine
- domain: Determines if it is attached to an AD
- os_type: Operating system type
- os_version: Malware version, in our case 1.8.6.1
- os_arch: Architecture: x86 o x64
- disks: Ratio of free space versus total disk space
- id: Machine guide, universal device identifier

The previous information is printed in a template, with a JSON format as shown below:

"os":{"lang":"%s",."username":"%s","hostname":"%s","domain":"%s","os_type":"windows","os_ve.sion":"%s","os_arch":"%s","disks":"%s.
","id":"%s"}

The generated sequence is printed in the following format:

%.8x=%s&%.8x=%s

Giving the following result:

88bed015=/LkZFINJp3VPMHYvngh5vC5eEnPRdbWNJteVfehRLo+C+fI+92EFe5qDvzMSxSE6vLXevA3RamgvvsHXxbuTZW/ED+t\iA3ggKHt9Y8F

This information is sent to the attackers' central server via HTTP POST request, located at securebestapp20[.]com, a domain created on September 16th, 2020, which is hosted on the infrastructure of the Russian provider Eurobyte (eurobyte[.]ru), which according to SpamHaus, was one of the most used providers for running botnets in 2020 (See Figure 2).

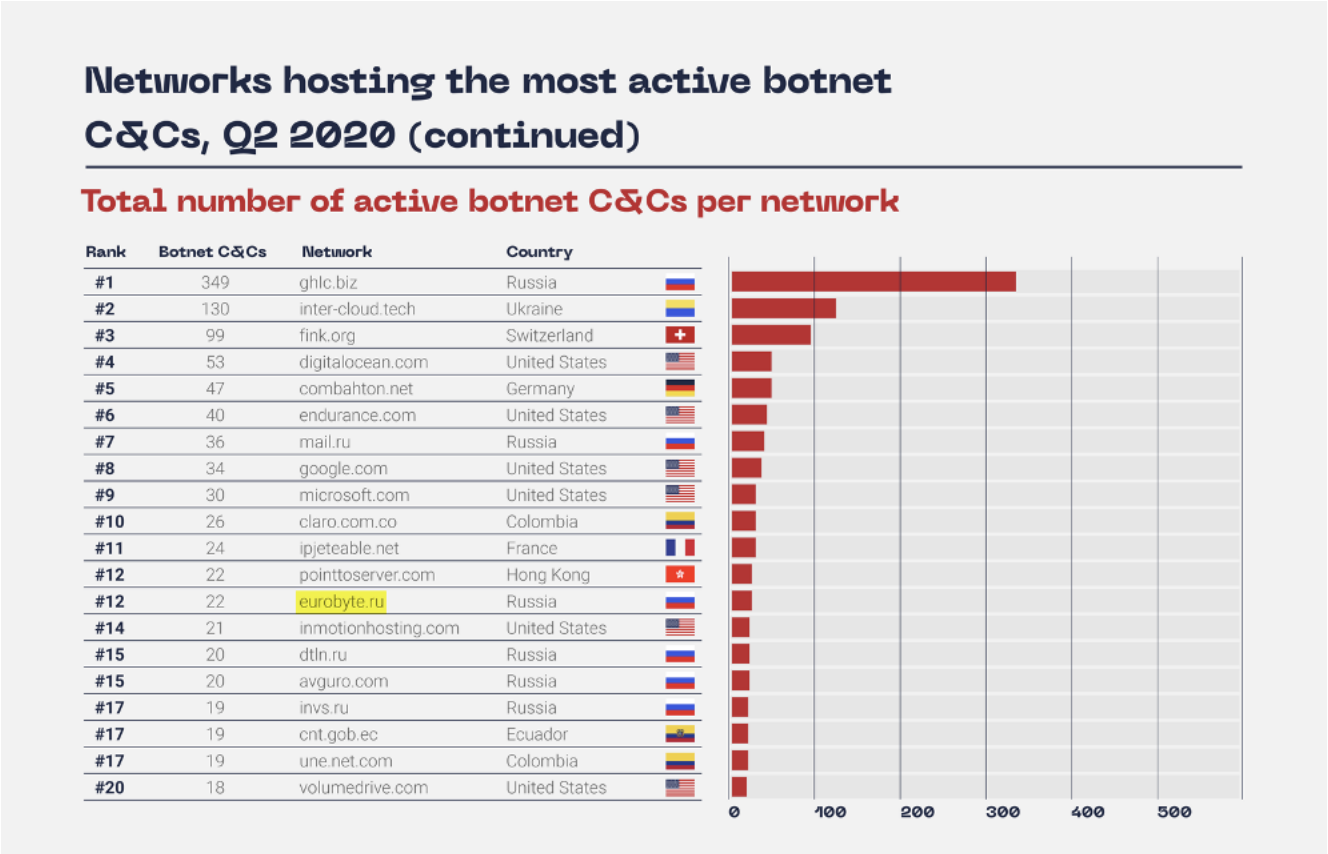


Figure 2: Hosted Botnets in 2020 (Source)

Interestingly, the User-Agent used for communication with the C2 is malformed. At first glance, everything looks normal, but when we look closely at the rv:79.0 field and the Firefox/80 field, we notice a discrepancy. According to the standard, these two fields should be the same, so it may be a factor used by the server to only accept connections from the ransomware controlled by them.

Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:79.0) Gecko/20100101 Firefox/80

In Figure 3, you can see the moment when the domain is accessed using the Windows Internet Connect API.

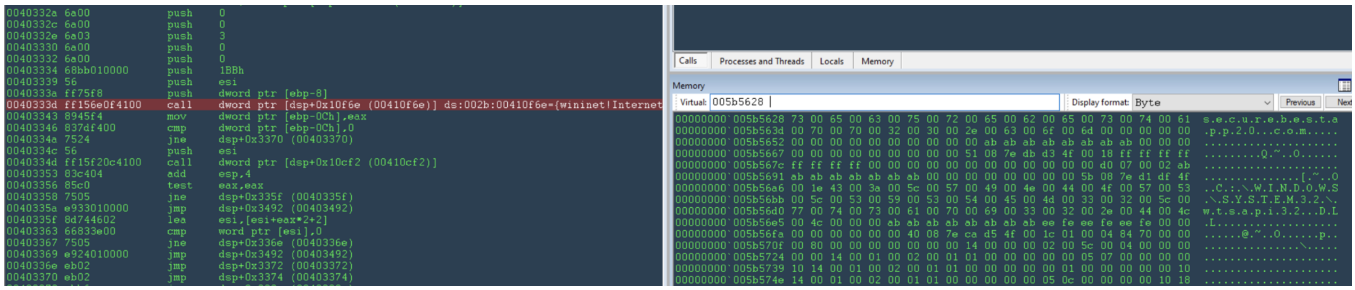


Figure 3: Darkside connecting to the C2

File encryption

In case the malware decides to continue with the encryption (if it is not discarded by the whitelist), it uses a recursive main function that scans each directory of the identified logical drives (see Figure 4):

- Removable drives: Floppy, Thumb drive, flash card
- Hard drives in the computer
- Mounted network drives

```
void fun_encrypt_local_drives(void)

{
  uint drives_string_len;
  int drive_type;
  undefined *valid_drives_buff_cpy;
  undefined valid_drives_buffer [256];
  undefined4 local_24;
  undefined4 local_20;
  undefined current_drive [24];

  drives_string_len = (*_GetLogicalDriveStringsW)(0x80,valid_drives_buffer);
  if (drives_string_len != 0) {
    valid_drives_buff_cpy = valid_drives_buffer;
    drives_string_len = drives_string_len >> 2;
    do {
      drive_type = (*_GetDriveTypeW)(valid_drives_buff_cpy);
      if (((drive_type == 3) || (drive_type == 2)) || (drive_type == 4)) {
        local_24 = 0x5c005c;
        local_20 = 0x5c003f;
        (*_wcscpy)(current_drive,valid_drives_buff_cpy);
        f_encrypt_files(&local_24);
      }
      valid_drives_buff_cpy = valid_drives_buff_cpy + 8;
      drives_string_len = drives_string_len - 1;
    } while (drives_string_len != 0);
  }
  return;
}
```

Figure 4: Identifying the logical units to be encrypted

The general process is detailed in the flow diagram in Figure 5.

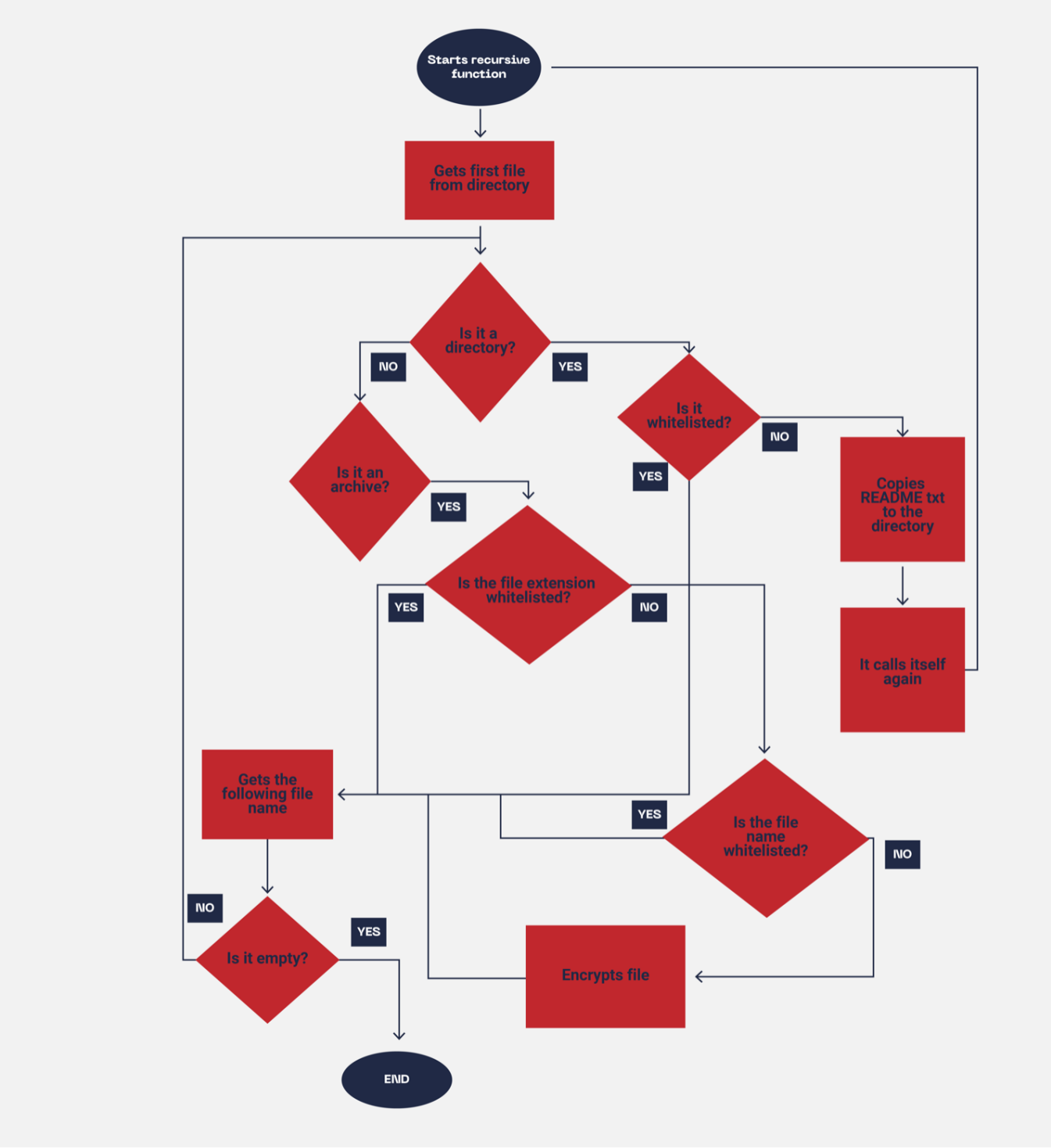


Figure 5: File encryption

The process starts by evaluating whether the path has a directory. If it does, the process copies the “readme” file with the attackers’ message to that destination.

If not, the path is viewed as a file and the process evaluates if the filename is present in the whitelist embedded in the malware, as shown below:

- autorun.inf
- boot.ini
- boot
- font.bin
- bootsect.bak
- desktop.ini
- icon

- cache.db
- ntldr
- ntuser.dat
- ntuser.dat.log
- ntuser.ini

If the file contains one of the following extensions, it is not encrypted:

```
2.adv
3.ani
4.bat
5.bin
6.cab
7.cmd
8.com
9.cpl
10.cur
11.deskthemepack
12.diagcab
13.diagcfg
14.diagpkg
15.dll
16.drv
17.exe
18.hlp
19.icl
20.icns
21.ico
22.ics
23.idx
24.ldf
25.lnk
26.mod
27.mpa
28.msc
29.msp
30.ms
31.styles
32.msu
33.nls
34.no
35.media
36.ocx
37.prf
38.ps1
39.rom
40.rtp
41.scr
42.shs
43.spl
44.sys
45.theme
46.theme
47.pack
48.wpx
49.lock
50.key
51.hta
52.msi
53.pdb
54.sql
55.sqlite
```

After making the initial validations, the malware proceeds to encrypt the files swiftly by occupying one thread per available processor following the following steps:

1. An input/output (e/o) completion port is created via the CreateIoCompletionPort API
2. Files to be encrypted are added to a queue via PostQueuedCompletionPort API
3. The encrypted files are obtained from the queue via GetQueuedCompletionPort

The file to be encrypted is passed to the function at address 0xD6209C, which starts this process. Figure 6 shows the use of the APIs to enqueue and remove encrypted files.

```
loc_D65BD5:
push    0FFFFFFFFh
lea     eax, [ebp+__lp_overlapped] ; Load Effective Ad
push    eax
lea     eax, [ebp+__completion_key] ; Load Effective A
push    eax
lea     eax, [ebp+__bytes_transferred] ; Load Effectiv
push    eax
push    Handle_CompletionPort_dword_D71024
call    kernel32_GetQueuedCompletionStatus_off_D70DAE
mov     ebx, [ebp+__lp_overlapped]
test    eax, eax        ; Logical Compare
jnz     short loc_D65C43 ; Jump if Not Zero (ZF=0)
```

```
loc_D666EB:
lea     eax, [ebx]      ; Load Effective Address
push    eax             ; _DWORD
push    0               ; _DWORD
push    0               ; _DWORD
push    Handle_CompletionPort_dword_D71028 ; _DWORD
call    kernel32_PostQueuedCompletionStatus_off_D70DB2
test    eax, eax        ; Logical Compare
jnz     short loc_D6671C ; Jump if Not Zero (ZF=0)
```

Figure 6: Functions related to encryption

The use of the stream cipher Salsa20 for file encryption is confirmed. In Figure 7, we can see the quarter-round function described in Wikipedia, which is used to customize the matrix, and on the right side (in the function sub_D6209C), the implementation made by the malware.

```
b ^= (a + d) <<< 7;
c ^= (b + a) <<< 9;
d ^= (c + b) <<< 13;
a ^= (d + c) <<< 18;
```

```
v22 = __ROL4__(LODWORD(v21) + v16->m64_i32[0], 7) ^ v16[2].m64_i32[0];
v23 = __ROL4__(v16->m64_i32[0] + v22, 9) ^ v16[4].m64_i32[0];
v24 = __ROL4__(v22 + v23, 13) ^ LODWORD(v21);
v16->m64_i32[0] ^= __ROL4__(v23 + v24, 18);
```

Figure 7: Confirming use of Salsa algorithm20

With this matrix, a 64-byte block is generated and used to encrypt the files, to which this block (encrypted with RSA) is also added at the end. In Figure 8, you can see the encrypted block in memory and how it ends up being added to the encrypted files on disk.

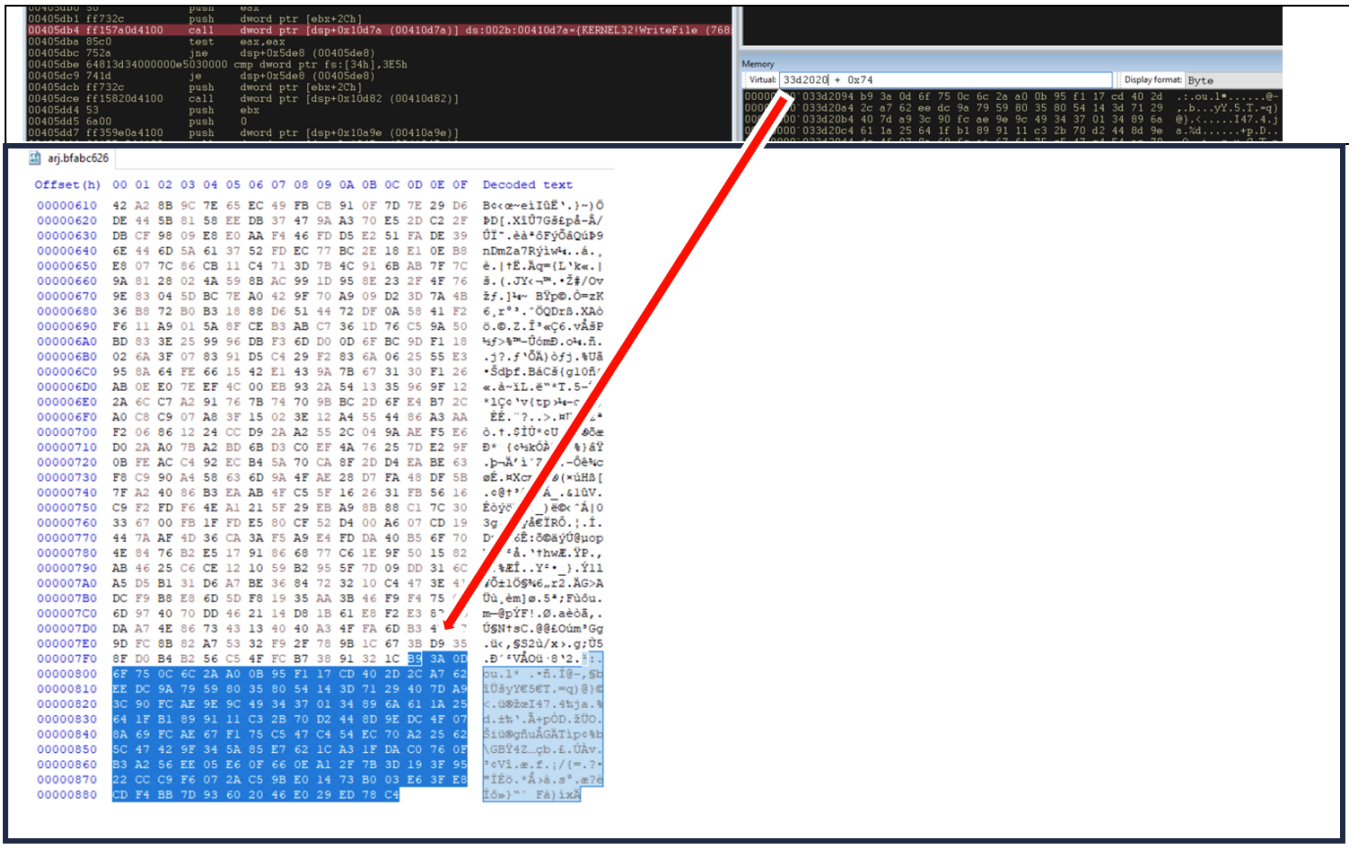


Figure 8: Salsa20 block added at the end of the encrypted file

Once the files have been encrypted, the following information is sent to the server:

"{"id":"%s","uid":"%s","enc-num":"%u","enc-size":"%s","skip-num":"%u","elapsed-time":"%u%u"}

- id: Previously generated victim ID
- uid: again the victim's ID
- enc-num: number of infected files
- enc-size: total number of encrypted bytes
- skip-num: number of files that were not encrypted (on the whitelist)
- elapsed-time: length of time of infection

Shadow copy removal

Shadow copies are a mechanism by which Windows generates a backup of the information, if they are deleted, there is no way to restore the information once encrypted.

In the case of DarkSide, the command is encoded in hexadecimal and uses the call to CreateProcessW to invoke it as follows:

powershell -ep bypass -c "(0..61) | % {$s+=[char][byte]

('0x'+'4765742D576D694F626A6563742057696E33325F536861646F77636F7079207C20466F72456163682D4F626A656374207B245F2E44656

Which via VMI, it gets the shadow copies and deletes them:

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

Generation of the wallpaper

Once the victim's files have been encrypted, the malware will change the wallpaper of their computer to inform them. The image seen in that wallpaper is generated by the malware and is stored in the C:\ProgramData directory <id>BMP with <id> being the id generated for that particular victim, as shown in Figure 9.

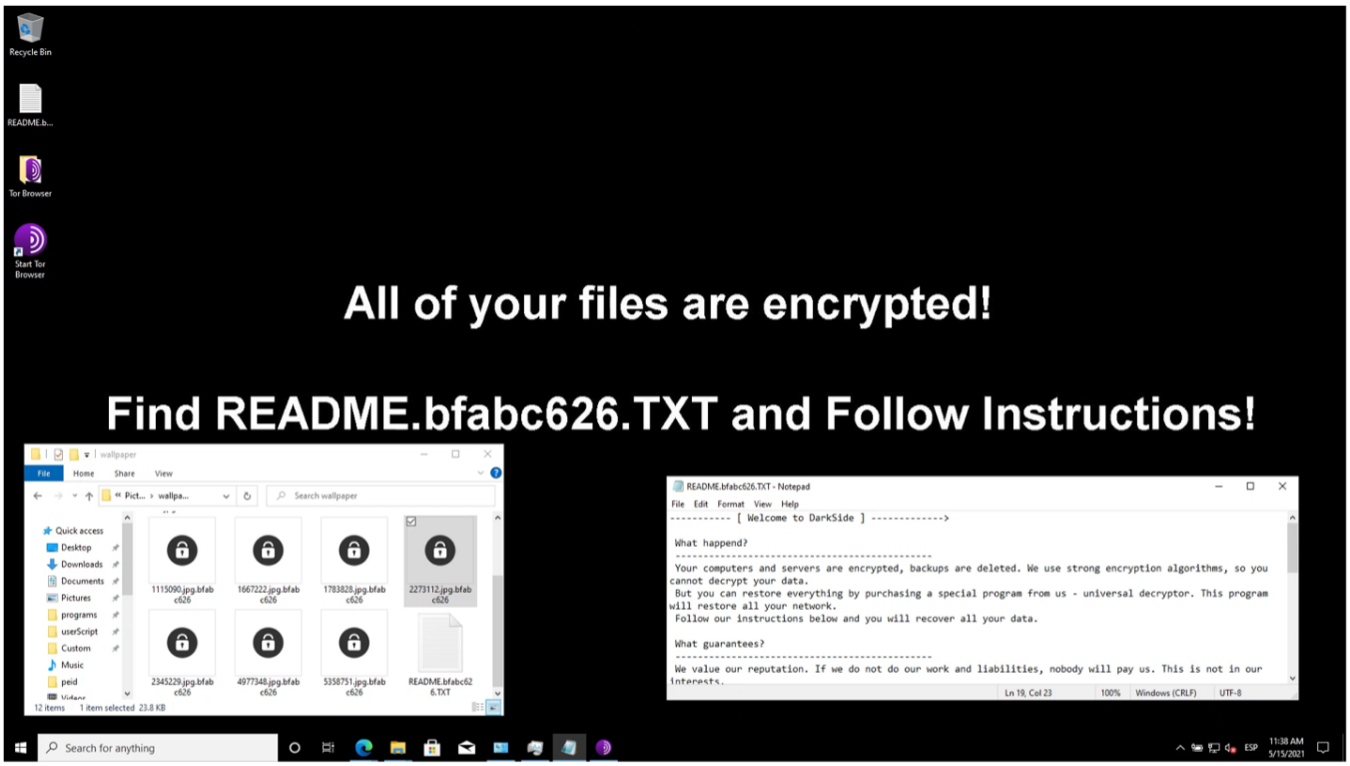


Figure 9: Message for the victim

Cleanup after encryption

Once the file encryption tasks are finished, the malware deletes itself, usually to make forensic analysis more difficult. For this task, it calls ShellExecuteW function with the following command:

cmd.exe /C DEL /F /Q C:\%USERNAME%\<ruta_del_malware>\darkside.exe >> NULL

DarkSide and REvil similarities

// Configuration

In both cases, the binaries occupy a configuration that is encrypted and embedded within the data section of the binary. This configuration in both cases affects some parameters of the execution, in addition to containing the address of the connection server and the attacker's public key. However, the format differs; while REvil uses a single string with JSON format, DarkSide has a division between the configurations and the public key. In addition, for DarkSide, the configuration after being decrypted must be unpacked. Figure 10 shows these differences:

Figure 10: Comparison of configurations between Darkside and REvil

// Generation of the Mutex

In regards to the generation of the Mutex of the process, in the case of REvil, it is a fixed value that corresponds to “Global\206в87e0-0e60-df25-dd8f-8e4e7e7d1e3bf0”. In the case of Darkside, the generation of this Mutex is dynamic. First, a template is generated, which is “Global\XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX” and then a crc32 is generated from the contents of the executable, and the X values are replaced with the generated value. This process implies that each build of the executable will have its own unique Mutex, allowing the malware to know if it is on a host that hasn’t been infected before. In the case of the sample we have, the result is "Global1609d3ee11333b13ecaf8b7d62361a1de1”.

// Packaging

The REvil executable is packaged with a custom algorithm, making it much more challenging to analyze, while all DarkSide samples found so far are unpackaged.

// Generating the file extension

A random file extension is added to every infected file. This extension prevents the malware from applying more than one encryption process to the host files, so it is essential always to generate the same file extension. In the case of REvil, this extension is generated by a crc32 of the contents of the cpuid function call. For DarkSide, a crc32 is made to the contents of the HKEY_LOCAL_MACHINE_MACHINE_SOFTWARE_Microsoft_Cryptography_MachineGuid registry.

Recommendations

Ransomware attacks have become the top method used by attackers worldwide to obtain large amounts of money in a short period of time. ransomware payments have tripled from 2019 to date. In the case of Colonial, due to the critical nature of their business and the need to restart operations immediately, the company did not have many options but to pay the total of $5 million USD. Unfortunately, ransomware attacks are global and are increasing in frequency and scale.

In Latin America and Mexico, the cases continue to rise. In November 2019, the IT systems of PEMEX were compromised by ransomware; according to bleepingcomputer, attackers asked for the sum of $4.9 million USD. In 2021 alone, we’ve seen attacks on multiple Mexican institutions, including banks, with gigabytes of information were leaked on the deep web, due to a possible lack of payment, along with an attack on the National Lottery of Mexico.

Why are ransomware attacks growing exponentially?

The ease of executing ransomware attacks through services known as ransomware as a service (RaaS) reduces the barriers of entry. RaaS allows non-technical people to hire a service that allows them to compromise companies with minimal effort, sharing profits with the creators of the service.

How can we combat this threat that is here to stay for years to come?

First of all, accept that sooner or later, your organization is going to be infected with ransomware unless you proactively make changes. The first step is to strengthen your processes, people, and technology by testing your systems against a ransomware attack. Metabase Q offers a different spin on ransomware as a Service via its APT Simulation service. By replicating multiple ransomware families like WannaCry, Ryuk, REvil, DarkSide, Avaddon, etc., in your network, we are able to test how well your systems would respond. The benefits include:

Strengthen your organization’s Ransomware monitoring, detection and eradication capabilities.

- Processes: Gap detection and strengthening of policies and procedures established to react to an incident
- People: Incident response training of SOC personnel
- Technology: Identifying gaps in your security solutions: SMTP Gateway, Endpoint, Lateral Movement, Event Correlation, Malicious Callbacks, etc. Is your investment yielding the expected results?

This new service reverse engineers emerging threats such as ransomware to reproduce the malicious code. Unlike RaaS, Metabase Q has the control to execute the ransomware without the potential side effects of irreversible damage, such as deleting shadow copies or publishing sensitive information on the Deepweb. By utilizing the TTPs (Techniques, Tactics, and Procedures) and IOCs (Indicators of Compromise) used by malware in the real world, we can train and strengthen your processes, people, and technology.

We train your team to detect and fight real ransomware in your organization without having to pay millions of dollars for the ransom. Contact us at: contact@metabaseq.com

Thank you, Bryan Gonzalez & Jose Zorrilla from Ocelot, for your support during this analysis.

Indicators of compromise (IOCs)

// Registry Changes

HKCR.bfabc626(Default)
HKCR\bfabc626\DefaultIcon(Default), Data: C:\Users\root\AppData\Local\bfabc626.ico"
HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Explorer\GlobalAssocChangedCounter", Data: 74

HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Content\CachePrefix Data: “”
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Cookies\CachePrefix", Data: Cookie:
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\History\CachePrefix Data: Visited:
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass, Data: 1
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName, Data: 1
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet , Data: 1
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect, Data: 0
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass, Data: 1
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName, Data: 1
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet, Data: 1
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect, Data: 0
HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-637130822-2221118250-14948887-1001\Device\HarddiskVolume4\Windows\System32\WindowsPowerShell\v1.0\powershell.exe, Data: 59 D4 90 A5 A5 4C D7 01 00 00 00 00 00 00 00 00
HKCU\Control Panel\Desktop\WallPaper, Data: C:\ProgramData\bfabc626.BMP"
HKCU\Control Panel\Desktop\WallpaperStyle, Data: 10
HKCU\Control Panel\Desktop\Wallpaper, Data: C:\ProgramData\bfabc626.BMP
// Created files

C:\%USERNAME%\AppData\Local\bfabc626.ico

C:\ProgramData\bfabc626.BMP

C:\%USERNAME%\AppData\Local\Temp\3582-490

//URLs seen

The last part is variable so it is not recommended to use it as a detection pattern.

hxxp://securebestapp20[.]com/mhzPjMHjEl

hxxp://securebestapp20[.]com/mhzpjmhjel

hxxps://securebestapp20[.]com/i7zMFQGyg0

hxxp://securebestapp20[.]com/adbeeccba

hxxps://securebestapp20[.]com/TpqTgJUS3v

hxxp://securebestapp20[.]com/ddbcebcd

hxxp://securebestapp20[.]com/bbaededade

hxxps://securebestapp20[.]com/Fg7xJ8MG6

Binaries communicating to securebestapp20[.]com

F9fc1a1a95d5723c140c2a8effc93722

F75ba194742c978239da2892061ba1b4

Cfcfb68901ffe513e9f0d76b17d02f96

9d418ecc0f3bf45029263b0944236884

91e2807955c5004f13006ff795cb803c

6a7fdab1c7f6c5a5482749be5c4bf1a4

3fd9b0117a0e79191859630148dcdc6d

222792d2e75782516d653d5cccfcf33b

E44450150e8683a0addd5c686cd4d202

C2764be55336f83a59aa0f63a0b36732