ANALYSIS REPORT

10365227.r1.v1 NUMBER

Malware Analysis Report

2022-09-20 DATE

Notification

Summary

Description

This Malware Analysis Report (MAR) is the result of analytic efforts by the Cybersecurity and Infrastructure Security Agency (CISA) to provide detailed analysis of files associated with CovalentStealer malware, which is designed to identify and exfiltrate files to a remote server. CISA obtained CovalentStealer malware samples during an on-site incident response engagement at a Defense Industrial Base (DIB) Sector organization compromised by advanced persistent threat (APT) actors.

CISA analyzed 19 files associated with CovalentStealer malware. The files are designed to identify file shares on a system, categorize the files, and upload the files to a remote server. The files include two configurations that specifically target the victim's documents using predetermined files paths and user credentials. The two remaining files were identified as open source utilities the threat actor utilized on the victim's system. One file is a publicly available utility used to compress and archive other files. The second file is an open source utility used to extract the Master File Table (MFT) from a volume and can be used for file enumeration.
CISA is distributing this MAR to enable network defense and reduce exposure to APT sponsored malicious cyber activity.

For more information on the confirmed compromise, see Joint CSA: Impacket and Exfiltration Tool Used to Steal Sensitive Information from Defense Industrial Base Organization.

Submitted Files (19)

09605981a072c604e6ef9ad2dd7d2a78b48b07ee3339589bfcf0a466a9190904 (msexch.log)

0b01f392fa030be1ddd549fb79cf280d2a2c745578a56fedd4cb5e9438ae72cb (ntstatus.bat)

0b7d15968d44710b3e7f153c04b5038d03900a6685643bc8efe688c4d5a5deab (ntstatus_temp.log)

157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656 (ntstatus.exe)

25afc6741abfa27f5b50844331772466182ebe3f74bc84f911314d1a68c62cb2 (mqsvn.ini)

30191b3badf3cdbc65d0ffeb68e0f26cef10a41037351b0f562ab52fce7432cc (msexch.exe)

3585c3136686d7d48e53c21be61bb2908d131cf81b826acf578b67bb9d8e9350 (mqsvn.exe)

517faa4a0666ec68842f256f08d987935b6ce9ef64e33f027e084e8f45b9366d (onedrv.dat)

52765525103f5b3b07d0882cc8ee4bb8e279ad5d451e1ed07cae3b98565cce29 (msexch.ini)

5ba0d0bfda372c1f6aa382a70f4ab8427ec998b680510e208fdf878cfda9afe3 (ntstatus.log)

603e75db59285734cfb5a469e984c4e359e660ccb7836ff9c209aec36931bc2b (mqsvn.log)

6a0cd866c849e62f9ccc26575d8794c2e0b14722387742b965d4358e1e0e8b3c (msexch_temp.log)

84164e1e8074c2565d3cd178babd93694ce54811641a77ffdc8d1084dd468afb (onedrv.exe)

91a8b31c126a021f5c156742016acdcca7d83eac4b583bae5d4fd0a85a96813b (onedrv.ini)

b03ac5eaf2131060ee381e5e46ebc705d8d617a90cc61fa4918174545b4fbaa6 (ntstatus.bin)

bfa7adeda4597b70bf74a9f2032df2f87e07f2dbb46e85cb7c091b83161d6b0a (vmware.exe)

da267c72f58ec487761de99d0f3bcfd87771a36afc06716053960633a74139df (ntstatus.ini)
e03a2c8a6e81cf62ba7401c598ea1d4635b08bbf9c2fec080b536dde29e6392f (msexch.bin)
fae38156e9ce12368c846836b87861f4f12e14698cb65f14545205fa56d8c496 (vmware.ps1)

Additional Files (2)

1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da (result.exe)
d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8 (Uploader.exe)

Findings

84164e1e8074c2565d3cd178babd93694ce54811641a77ffdc8d1084dd468afb

Tags

information-stealer uploader

Details

Name	onedrv.exe
Size	791040 bytes
Type	PE32+ executable (GUI) x86-64 Mono/.Net assembly, for MS Windows
MD5	806998079c80f53afae3b0366bac1479
SHA1	9f7378da13ca1da75e12e536c8e2dc4cd2236489
SHA256	84164e1e8074c2565d3cd178babd93694ce54811641a77ffdc8d1084dd468afb
SHA512	3d592a606426386fa5f1224c7d3f82f31f5a4d23f9c67422d774e080725bc5698e7786407863dd50d7172e814871bdfabbbe6dce9545733d995ddd892249ba22
ssdeep	12288:kylzsYTd+LXxWtmtOdnPR3xTexehCkijOcXF8qSH8gdkMdCNGCWJOWCmP8pSMmVN:ky4sO+9ymtsnPRBnlivXPSHxkMNHCNp
Entropy	7.996795

Antivirus

Avira	HEUR/AGEN.1221987

YARA Rules

-
  ```
  rule CISA_10365227_03 : ClientUploader
  {
    meta:
      Author = "CISA Code & Media Analysis"
      Incident = "10365227"
      Date = "2021-12-23"
      Last_Modified = "20211224_1200"
      Actor = "n/a"
      Category = "n/a"
      Family = "n/a"
      Description = "Detects ClientUploader_onedrv"
      MD5_1 = "806998079c80f53afae3b0366bac1479"
      SHA256_1 = "84164e1e8074c2565d3cd178babd93694ce54811641a77ffdc8d1084dd468afb"
    strings:
      $s1 = "Decoder2"
      $s2 = "ClientUploader"
      $s3 = "AppDomain"
      $s4 = { 5F 49 73 52 65 70 47 ?? 44 65 63 6F 64 65 72 73 }
      $s5 = "LzmaDecoder"
      $s6 = "$ee1b3f3b-b13c-432e-a461-e52d273896a7"
    condition:
      uint16(0) == 0x5a4d and all of them
  }
  ```

ssdeep Matches

No matches found.

PE Metadata

Compile Date	2021-09-10 17:59:57-04:00
Internal Name	ClientUploader.exe
Original Filename	ClientUploader.exe

Product Version	1.0.0.0

PE Sections

MD5	Name	Raw Size	Entropy
6b81a95076cc3d6f6dff7d32afa3b7e2	header	512	2.297287
2d3081eb51c7c393e0a670c8bfcf7c24	.text	788992	7.998126
5569bca67ba8c174f30990c07b585dbe	.rsrc	1536	3.966404

Packers/Compilers/Cryptors

Microsoft Visual C++ v6.0

Relationships

84164e1e80...	Used	91a8b31c126a021f5c156742016acdcca7d83eac4b583bae5d4fd0a85a96813b
84164e1e80...	Created	517faa4a0666ec68842f256f08d987935b6ce9ef64e33f027e084e8f45b9366d

Description

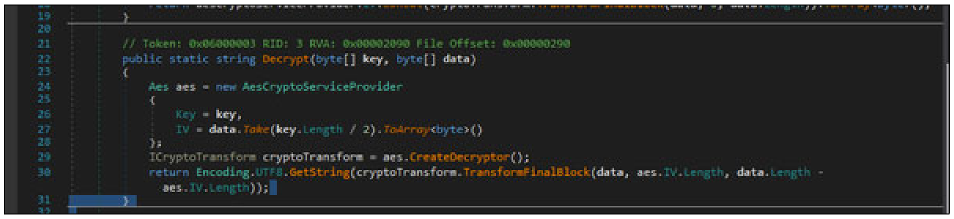
This file has been identified as CovalentStealer malware. The actor utilized code from several open source projects, including ClientUploader. The retained the internal name "ClientUploader.exe". The program is a file management system that is capable of uploading files to the Internet.

When the program is executed, it will spawn an instance of itself in memory called 'koi'. This instance accesses several embedded resources that it uses to locate and manipulate files on the system. The following is a list of the primary embedded resources:

—Begin Embedded Resources—
BaseNetwork – This resource is used to create sessions and establish connections to the server.
FileContainer – This resource is used to access file shares via Server Message Block (SMB). It is also used to enumerate files and directories and sort them by Message Digest 5 (MD5) hash. It maintains Internet Protocol (IP) addresses, logins, domain names, passwords, and paths for shares on the network.
IFileWorker – This resource is a file management program that is capable of moving and categorizing files. It contains compression libraries for Gzip and Brotli, as well as a file blacklist.
Encryption – This resource handles file encryption, decryption and secure communications. It decrypts the configuration file, onedrv.ini (91a8b31c126a021f5c156742016acdcca7d83eac4b583bae5d4fd0a85a96813b) using the hard-coded Advanced Encryption Standard (AES) key 'M(xcHq88q[s=pc7^+u_Gb_}JC%QQwP:h' and an Initialization Vector (IV) using the first half of the AES key (See Figure 1).
OneDriveClient – This resource targets a user's OneDrive account and creates an upload session to send the files to a remote server. It is able to access files in the victim's OneDrive by unique ID (See Figure 2). Files are uploaded to a Microsoft Azure client identified in the configuration file onedrv.ini by client ID.
—End Embedded Resources—

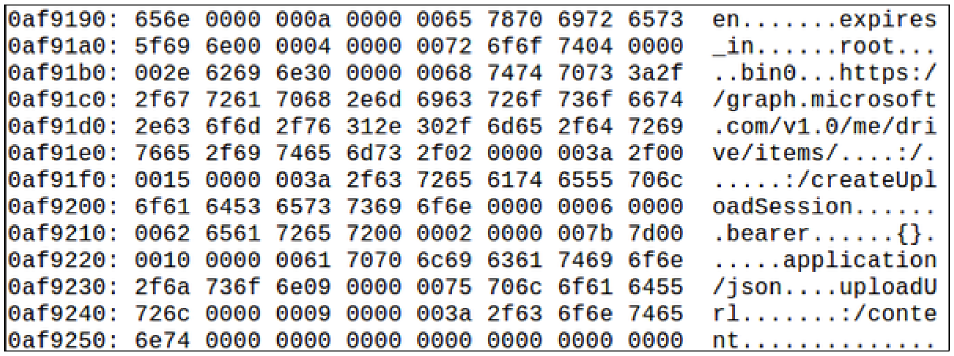
The program runs a debugging routine and will output debugging data to a file with the same name as the malware and with the .dat extension, e.g. onedrv.dat (517faa4a0666ec68842f256f08d987935b6ce9ef64e33f027e084e8f45b9366d).

Screenshots


```
        }

        // Token: 0x06000003 RID: 3 RVA: 0x00002090 File Offset: 0x00000290
        public static string Decrypt(byte[] key, byte[] data)
        {
            Aes aes = new AesCryptoServiceProvider
            {
                Key = key,
                IV = data.Take(key.Length / 2).ToArray<byte>()
            };
            ICryptoTransform cryptoTransform = aes.CreateDecryptor();
            return Encoding.UTF8.GetString(cryptoTransform.TransformFinalBlock(data, aes.IV.Length, data.Length -
              aes.IV.Length));
        }
```

Figure 1 - This is the AES encryption routine. The routine uses the hard-coded string 'M(xcHq88q[s=pc7^+u_Gb_}JC%QQwP:h' as the AES key and the first half of the key as the IV.



```
0af9190: 656e 0000 000a 0000 0065 7870 6972 6573  en.......expires
0af91a0: 5f69 6e00 0004 0000 0072 6f6f 7404 0000  _in......root...
0af91b0: 002e 6269 6e30 0000 0068 7474 7073 3a2f  ..bin0...https:/
0af91c0: 2f67 7261 7068 2e6d 6963 726f 736f 6674  /graph.microsoft
0af91d0: 2e63 6f6d 2f76 312e 302f 6d65 2f64 7269  .com/v1.0/me/dri
0af91e0: 7665 2f69 7465 6d73 2f02 0000 003a 2f00  ve/items/....:/.
0af91f0: 0015 0000 003a 2f63 7265 6174 6555 706c  .....:/createUpl
0af9200: 6f61 6453 6573 7369 6f6e 0000 0006 0000  oadSession......
0af9210: 0062 6561 7265 7200 0002 0000 007b 7d00  .bearer......{}.
0af9220: 0010 0000 0061 7070 6c69 6361 7469 6f6e  .....application
0af9230: 2f6a 736f 6e09 0000 0075 706c 6f61 6455  /json....uploadU
0af9240: 726c 0000 0009 0000 003a 2f63 6f6e 7465  rl.......:/conte
0af9250: 6e74 0000 0000 0000 0000 0000 0000 0000  nt..............
```

Figure 2 - This is the configuration for the upload session. This module is able to access items in the user's OneDrive by unique ID.

517faa4a0666ec68842f256f08d987935b6ce9ef64e33f027e084e8f45b9366d

Details

Name	onedrv.dat
Size	267224 bytes
Type	ASCII text, with CRLF line terminators
MD5	dc0414dec9a84d6342c5d5fc77bbdbed
SHA1	1dad19123564d7d02c3259ab4b06c90181dc4b37
SHA256	517faa4a0666ec68842f256f08d987935b6ce9ef64e33f027e084e8f45b9366d
SHA512	1d262f06881516ca2274d8fb18bcb4bcf9c0b3229370b0609f3803f356a676b1149e22da6a33957862d8470a8531d9719af07bd75379df2ca29e373604fb32cb
ssdeep	3072:ERNwmyBvqZKFkVfhJnEFbDcazPQLTnVy8JR6Ylb3uQ0PQNIfFrCGdDIBXZuZpZfB:bWrjgA
Entropy	5.360335

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

517faa4a06...	Created_By	84164e1e8074c2565d3cd178babd93694ce54811641a77ffdc8d1084dd468afb

Description

This file contains output from the debugging routine in onedrv.exe (84164e1e8074c2565d3cd178babd93694ce54811641a77ffdc8d1084dd468afb).

91a8b31c126a021f5c156742016acdcca7d83eac4b583bae5d4fd0a85a96813b

Tags

information-stealer

Details

Name	onedrv.ini
Size	1088 bytes
Type	data

MD5	a0ab6d3e643d4dd51ee6ae9079b175a4
SHA1	f179fcc4c41ca5cb443551f88a1074d5176d33f4
SHA256	91a8b31c126a021f5c156742016acdcca7d83eac4b583bae5d4fd0a85a96813b
SHA512	237baa401e0c52ca816cebafa5abf088e9a757f4da452e97210a1fe8eda8c0adc67aa19cacd662dcc98f5bd355d679fb096ff4e97cd54e16c199c66946d65a5e
ssdeep	24:olkc5V0yhsd/AFvaPo3b6EJ2ITY9UI62JPld5oKLeWb6l+vTI:olkq0yK/Ata5EJ2I5nOTvTI
Entropy	7.824751

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

91a8b31c12...	Used_By	84164e1e8074c2565d3cd178babd93694ce54811641a77ffdc8d1084dd468afb

Description

This artifact is the encrypted configuration file for the OneDriveClient module contained in the file ondrv.exe (84164e1e8074c2565d3cd178babd93694ce54811641a77ffdc8d1084dd468afb). The data is decrypted using the hard-coded key 'M(xcHq88q[s=pc7^+u_Gb_}JC%QQwP:h'.

The file contains paths to two archives targeted by the attacker. The file includes the IP address of the server, stolen credential information, and a key to encrypt the uploaded data. NOTE: The decrypted configuration contains confidential client information and therefore is not included in this report.

In addition, the data contains a refresh token for an OAuth client for Microsoft Azure with the Client ID of '7a3b4b84-ed28-4f18-b30d-218788c74a5f'. Speed and compression information as well as times that the OneDrive share can be accessed are also included in the configuration.

157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656

Tags

information-stealer   obfuscated   trojan   uploader

Details

Name	ntstatus.exe
Size	6656 bytes
Type	PE32+ executable (GUI) x86-64 Mono/.Net assembly, for MS Windows
MD5	c435d133b45783cce91a5d4e4fbe3f52
SHA1	9ddfa0669358bc19a166a41fd93cec5a3c88205d
SHA256	157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656
SHA512	e4d43dc23ff78f55bc857608fa33691eb7fb3e132332660b46460e7e7512104bc22484489d3d0fbd136270de9f7060641505ad2854cefd50b31ca6bb31b2ae18
ssdeep	96:nPbVkB7jiZStZC+01RPmaUrfzvDwiFMCnd+taflUTsqzNt:nPbqFiwW+g5maMzDwQMCQwmT
Entropy	4.921630

Antivirus

Adaware	Gen:Variant.Tedy.82790
Bitdefender	Gen:Variant.Tedy.82790
ESET	a variant of MSIL/Agent.VOV trojan

McAfee	Generic trojan.ri
NETGATE	Malware.Generic
Symantec	Process timed out

YARA Rules

```
• rule CISA_10365227_01 : APPSTORAGE
  {
    meta:
       Author = "CISA Code & Media Analysis"
       Incident = "10365227"
       Date = "2021-12-23"
       Last_Modified = "20211224_1200"
       Actor = "n/a"
       Category = "n/a"
       Family = "APPSTORAGE"
       Description = "Detects AppStorage_ntstatus_msexch samples"
       MD5_1 = "c435d133b45783cce91a5d4e4fbe3f52"
       SHA256_1 = "157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656"
       MD5_2 = "baa634fdd2b34956524b5519ee97b8a8"
       SHA256_2 = "30191b3badf3cdbc65d0ffeb68e0f26cef10a41037351b0f562ab52fce7432cc"
    strings:
       $s1 = "026B924DD52F8BE4A3FEE8575DC"
       $s2 = "GetHDDId"
       $s3 = "AppStorage"
       $s4 = "AppDomain"
       $s5 = "$1e3e5580-d264-4c30-89c9-8933c948582c"
       $s6 = "hrjio2mfsdlf235d" wide
    condition:
       uint16(0) == 0x5a4d and all of them
  }
```

ssdeep Matches

No matches found.

PE Metadata

Compile Date	2101-07-23 04:43:10-04:00
Internal Name	AppStorage.exe
Original Filename	AppStorage.exe
Product Version	1.0.0.0

PE Sections

MD5	Name	Raw Size	Entropy
3994632889cebeff28c360da22c696f3	header	512	2.255013
bec2cac9d419ae07e526a03c4a94cb64	.text	4608	5.307382
0551c676439e5d812cb2bab3f2060c1b	.rsrc	1536	3.934855

Packers/Compilers/Cryptors

Microsoft Visual C++ v6.0

Relationships

157a0ffd18...	Related_To	b03ac5eaf2131060ee381e5e46ebc705d8d61 7a90cc61fa4918174545b4fbaa6
157a0ffd18...	Dropped	1352dbb093a337eb8db9d0135adbe0542bb7 e7163616e4f8962919becab171da

157a0ffd18...	Related_To	0b01f392fa030be1ddd549fb79cf280d2a2c74 5578a56fedd4cb5e9438ae72cb

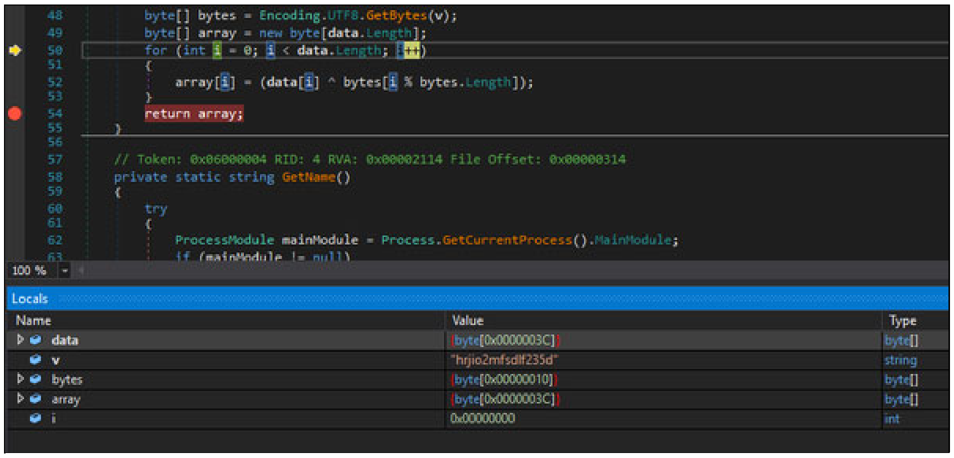
Description

This artifact is an obfuscated .NET executable that is used to decode a variant of the CovalentStealer malware. When executed, the program will check the present name of the program and then look in the current directory for a file with the same name and a .bin extension, e.g. ntstatus.bin (b03ac5eaf2131060ee381e5e46ebc705d8d617a90cc61fa4918174545b4fbaa6).

The program seeks to generate a key called 'HDDId' to decode ntstatus.bin. The embedded string 'hrjio2mfsdlf235d' is used to decode instructions within the program to generate the key (See Figure 3). The first command identifies the machineName of the system. The second command reads the Windows Management Instrumentation (WMI) namespace root/cimv2 to locate the volumeserialnumber of the current drive. Both variables are then modified using an exclusive OR (XOR) routine and the same string above is used to generate the key (See Figure 4). The first part of the key is generated from the volumeserialnumber, and during analysis resolved to '76D55BD2'. The machineName resolved to 'F3124EDD' creating the key '76D55BD2F3124EDD' (See Figure 5). Note: The key is an example.

To generate the correct key the machineName and volumeserialnumber must match the victim's system, otherwise it fails to decode ntstatus.bin and the program will terminate. This method is used to thwart independent analysis of the file, ntstatus.bin.

Screenshots

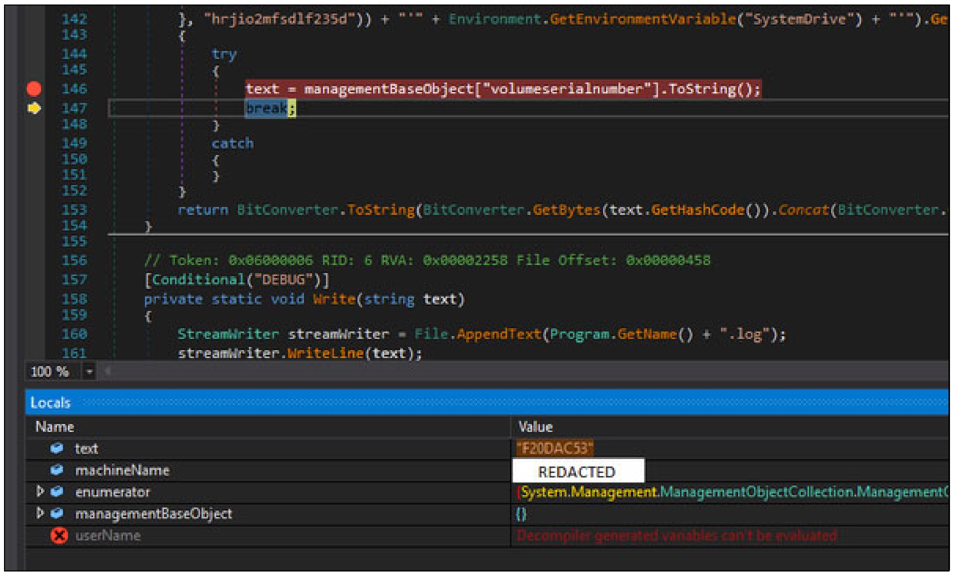


```
        byte[] bytes = Encoding.UTF8.GetBytes(v);
        byte[] array = new byte[data.Length];
        for (int i = 0; i < data.Length; i++)
        {
            array[i] = (data[i] ^ bytes[i % bytes.Length]);
        }
        return array;
    }

    // Token: 0x06000004 RID: 4 RVA: 0x00002114 File Offset: 0x00000314
    private static string GetName()
    {
        try
        {
            ProcessModule mainModule = Process.GetCurrentProcess().MainModule;
            if (mainModule != null)
```

Name	Value	Type
data	{byte[0x0000003C]}	byte[]
v	"hrjio2mfsdlf235d"	string
bytes	{byte[0x00000010]}	byte[]
array	{byte[0x0000003C]}	byte[]
i	0x00000000	int

Figure 3 - Screenshot of the XOR routine using the string 'hrjio2mfsdlf235d'.

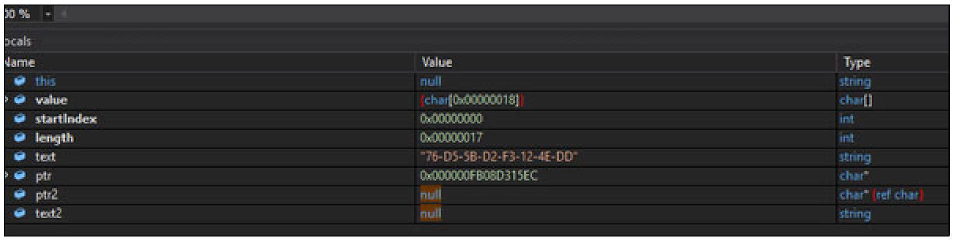


```
            }, "hrjio2mfsdlf235d")) + "'" + Environment.GetEnvironmentVariable("SystemDrive") + "'").Ge
            {
                try
                {
                    text = managementBaseObject["volumeserialnumber"].ToString();
                    break;
                }
                catch
                {
                }
            }
            return BitConverter.ToString(BitConverter.GetBytes(text.GetHashCode()).Concat(BitConverter.
        }

        // Token: 0x06000006 RID: 6 RVA: 0x00002258 File Offset: 0x00000458
        [Conditional("DEBUG")]
        private static void Write(string text)
        {
            StreamWriter streamWriter = File.AppendText(Program.GetName() + ".log");
            streamWriter.WriteLine(text);
```

Name	Value
text	"F20DAC53"
machineName	REDACTED
enumerator	{System.Management.ManagementObjectCollection.Management(
managementBaseObject	{}
userName	Decompiler generated variables can't be evaluated

Figure 4 - The program collects the machineName and volumeserialnumber to generate the HDDId key.



```
00 %
ocals
Name            Value                          Type
  this          null                           string
> value         {char[0x00000018]}             char[]
  startIndex    0x00000000                     int
  length        0x00000017                     int
  text          "76-D5-5B-D2-F3-12-4E-DD"      string
> ptr           0x000000FB08D315EC             char*
  ptr2          null                           char* {ref char}
  text2         null                           string
```

Figure 5 - This is the generated HDDId key used to decode ntstatus.bin

b03ac5eaf2131060ee381e5e46ebc705d8d617a90cc61fa4918174545b4fbaa6

Tags

information-stealer obfuscated uploader

Details

Name	ntstatus.bin
Size	1834496 bytes
Type	data
MD5	d5a7b90177cdf81c2e1de40dc834d764
SHA1	d5dee0a05101cf9ed3c3ca76cf01f518c3ef922c
SHA256	b03ac5eaf2131060ee381e5e46ebc705d8d617a90cc61fa4918174545b4fbaa6
SHA512	cfccd6701a69047c7de246601d2cd41cdc87d314bdcf070778938dad22e3bf5911d3beca0d75379dabdda1ad3c229c3bec329b840f5e4828c8bab41c1cdff159
ssdeep	24576:vsGNL+Kei7j3iTeG0fYHTlyAUoFwZJuaEh68w8To7FgunNZG10guctbAgYMEc+1B:DNb7dEh68E7204hEVF
Entropy	6.681125

Antivirus

Symantec	Unavailable (production)

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

b03ac5eaf2...	Related_To	157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656
b03ac5eaf2...	Contains	1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da

Description

This is an obfuscated version of CovalentStealer malware. The file is decoded by ntstatus.exe (157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656) using the key '76D55BD2F3124EDD'. The decoded file is called result.exe (1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da) and is detailed in this report.

1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da

Tags

information-stealer uploader

Details

Name	result.exe
Size	1834496 bytes
Type	PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows
MD5	27a0ba098b8403570c7b1e0863c2d6c5
SHA1	22cb98b9548ffd1010b2799a791ef42b8943f3c9
SHA256	1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da
SHA512	7eb71e11947a762d8a9a396de21d6b704f8021acc0ddfc7a959897569d429f3347c9bd1c3206703375d09a81defd3d1f9bba0ea137157d8546b862ded030c4c2
ssdeep	49152:F2f6rfgMSneK065JlYaDmxZF5ax00MSMoOKiYyBg9FzfJNFL5QPWES2s1B+dBrSY:F2f6rfgMSneK065JlYaDmxZF5ax00MSt
Entropy	5.579937

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

97	d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8

PE Metadata

Compile Date	2021-10-19 20:19:25-04:00
Import Hash	f34d5f2d4577ed6d9ceec516c1f5a744
Internal Name	ClientUploader.exe
Original Filename	ClientUploader.exe
Product Version	1.0.0.0

PE Sections

MD5	Name	Raw Size	Entropy
8a2ac318e59571d7c72221d67498bd5f	header	512	2.722440
be70af56c305ef153e32ecc2430d4d8a	.text	1831936	5.581972
5488f249cf62feed84546911d54f96f2	.rsrc	1536	3.971470
f80d2b416a07808182a35c49f6967d8f	.reloc	512	0.101910

Relationships

1352dbb093...	Created	5ba0d0bfda372c1f6aa382a70f4ab8427ec998b680510e208fdf878cfda9afe3
1352dbb093...	Created	0b7d15968d44710b3e7f153c04b5038d03900a6685643bc8efe688c4d5a5deab
1352dbb093...	Used	da267c72f58ec487761de99d0f3bcfd87771a36afc06716053960633a74139df
1352dbb093...	Dropped_By	157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656
1352dbb093...	Created	0b01f392fa030be1ddd549fb79cf280d2a2c745578a56fedd4cb5e9438ae72cb
1352dbb093...	Contained_Within	b03ac5eaf2131060ee381e5e46ebc705d8d617a90cc61fa4918174545b4fbaa6

Description

This artifact has been identified as CovalentStealer malware. When the program is executed it will decrypt and read the configuration file ntstatus.ini (da267c72f58ec487761de99d0f3bcfd87771a36afc06716053960633a74139df) in the current directory. It uses the hard-coded AES-256-CBC key ‘M(xcHq88q[s=pc7^+u_Gb_}JC%QQwP:h’ to decrypt the file. The configuration file will include a path to the directory containing the targeted files, compression parameters, and connection parameters for connecting to a system on the Internet to upload data.

The malware has several primary modules. The module IFileWorker contains the following functions:

—Begin IFileWorker Functions—
Brotli. – This function contains the Brotli compression library to compress and decompress files.
ContainersFilesWorker. – This function keeps track of uploaded files. It compares the files to a hash list for the file and path before uploading and also compares them to a whitelist and a blacklist by file extension. It also logs the status of each file in the upload process.
Extension. – This function checks the file extension to determine if the file needs to be compressed.
File Archive. – This function verifies the size of the file and disposition before compressing the file.
FileBlock. – This function converts the file data into a byte stream.
FileContainers. – This function segregates files by file type based on the extension.
GZip. – This function contains the Gzip compression library to compress and decompress files.
Logger. – This function logs debug status messages and telemetry data from other functions and outputs them to a file using the base name and the .dat extension, e.g. ntstatus.dat (See Figure 6).
WhiteAndBlackList. – This function maintains a list of files by name and a list of files by extension that match the whitelist or blacklist from the configuration file.
—End IFileWorker Functions—

Note: The actor utilized this code from the open source project IFileWorker.

The module OneDriveClient contains the following functions:

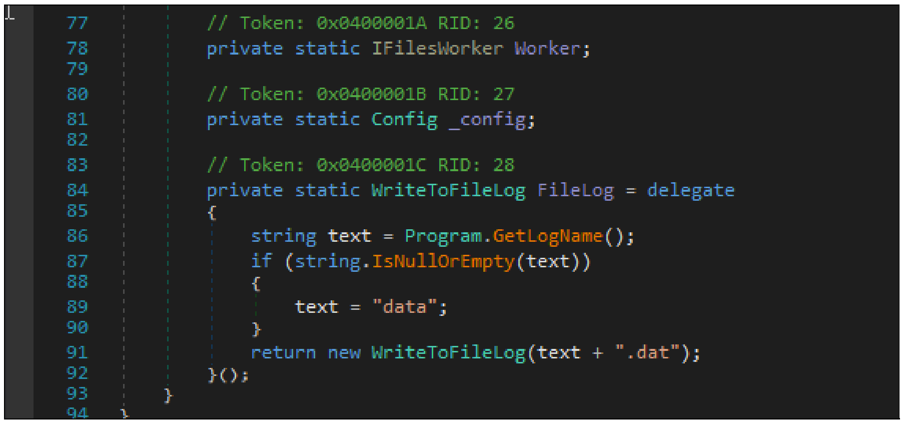
—Begin OneDriveClient Functions—
OneDrive. – This function uploads files to a Uniform Resource Locator (URL). It configures speed, buffer size, time, etc. based on the parameters in the configuration file, ntstatus.ini. Then, it reports the status of each file to the IFileWorker.Logger function. The following are examples of the OneDrive commands:

```
—Begin OneDrive Commands—
OneDriveClient.OneDriveChannel+<Send>
OneDriveClient.OneDrive+<GetAccessToken>
OneDriveClient.OneDrive+<UploadData>
OneDriveClient.OneDrive+<UploadFile>
OneDriveClient.OneDrive+<UploadLargeFile>
OneDriveClient.OneDrive+<GetUploadUrl>
OneDriveClient.OneDrive+<UploadPartWithStopwatch>
OneDriveClient.OneDrive+<UploadPart>
OneDriveClient.OneDrive+<UploadSmallFileWithStopWatch>
OneDriveClient.OneDrive+<UploadSmallFile>
—End OneDriveClient Functions—
```

OneDriveChannel. – This function establishes the connection to the server.
OneDriveChannelSettings. – This function reads the ClientID, Redirect, Refresh Token, and Scopes from the configuration file, ntstatus.ini to negotiate the connection to the client.
UploadedFiles. – This function logs the hash and the file path of the uploaded files and records the information into two files where ntstatus.log contains a list of file hashes and ntstatus_temp.log contains a list of file path hashes (See Figure 7).
—End OneDriveClient Functions—

The program also contains supporting libraries for the SMB protocol versions 2 and 3. The libraries have the capacity to maintain a list of IP addresses, logins, domainNames, passwords, and SMB clients that can be used to attempt to search for and log into SMB file stores. Files can be searched by file path, file status (e.g., open or closed), and file attributes (e.g. shared, read only, etc.).

Screenshots

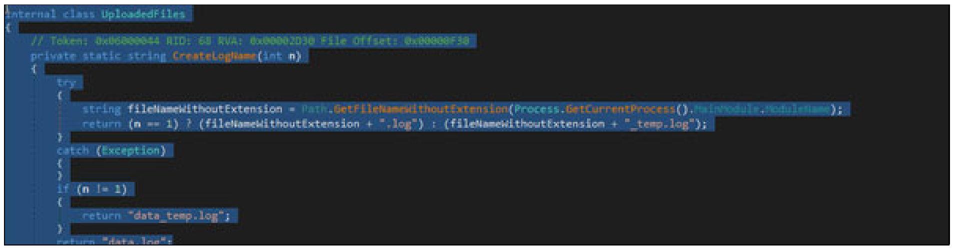


```
        // Token: 0x0400001A RID: 26
        private static IFilesWorker Worker;

        // Token: 0x0400001B RID: 27
        private static Config _config;

        // Token: 0x0400001C RID: 28
        private static WriteToFileLog FileLog = delegate
        {
            string text = Program.GetLogName();
            if (string.IsNullOrEmpty(text))
            {
                text = "data";
            }
            return new WriteToFileLog(text + ".dat");
        }();
    }
}
```

Figure 6 - The IFileWorker.Logger function is used to generate the log file for debug and telemetry data.



```
internal class UploadedFiles
{
    // Token: 0x06000044 RID: 68 RVA: 0x00002D30 File Offset: 0x00000F30
    private static string CreateLogName(int n)
    {
        try
        {
            string fileNameWithoutExtension = Path.GetFileNameWithoutExtension(Process.GetCurrentProcess().MainModule.ModuleName);
            return (n == 1) ? (fileNameWithoutExtension + ".log") : (fileNameWithoutExtension + "_temp.log");
        }
        catch (Exception)
        {
        }
        if (n != 1)
        {
            return "data_temp.log";
        }
        return "data.log";
```

Figure 7 - The OneDriveClient.UploadedFiles function records MD5 hashes of uploaded files into the file ntstatus.log and MD5 hashes of the file paths into the file ntstatus_temp.log.

da267c72f58ec487761de99d0f3bcfd87771a36afc06716053960633a74139df

Tags

information-stealer uploader

Details

Name	ntstatus.ini
Size	3392 bytes
Type	data
MD5	b1a7c2ae593e814cfecdcff709b02615
SHA1	ababa956175b2ddae7ec92162a8464b40b79064a
SHA256	da267c72f58ec487761de99d0f3bcfd87771a36afc06716053960633a74139df
SHA512	f511508878f821f80f10d387a60c7bab14c7384cd4ce0a68c73b0331d13d4b716805e3a53794ef0def0062d08eea489ef6239c53c2fa2d7f1c3478aba7e204b1
ssdeep	96:m74SD0f7Z2wXZ/BFmcktZdsczgmwL1COPP8yeTY4l9N:s4SDA73Zqlt7gmYQEUyMY4jN
Entropy	7.948675

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

da267c72f5...	Used_By	1352dbb093a337eb8db9d0135adbe0542bb7 e7163616e4f8962919becab171da

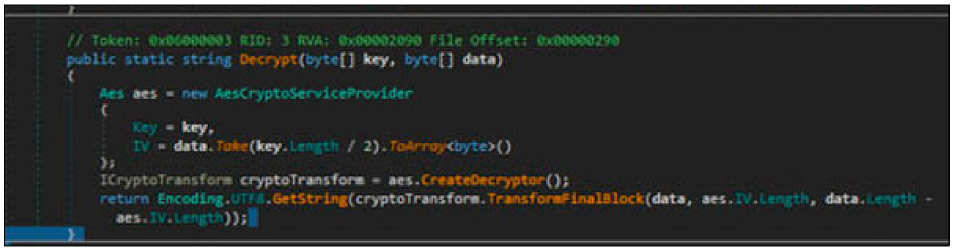
Description

This artifact is the encrypted configuration file for the OneDriveClient module contained in the file result.exe (1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da) detailed in this report. The data is decrypted using the hard-coded AES-256-CBC key 'M(xcHq88q[s=pc7^+u_Gb_}JC%QQwP:h'. The algorithm uses an IV that is derived from the first half of the encryption key (See Figure 8).

The file contains multiple paths to archives targeted by the attacker. The file includes the IP address of the server, stolen credential information, and a key to encrypt the uploaded data. NOTE: The decrypted configuration contains confidential client information and therefore is not included in this report.

In addition, the data contains a refresh token for an OAuth client for Microsoft Azure with the Client ID of '7a3b4b84-ed28-4f18-b30d-218788c74a5f'. Speed and compression information as well as times that the OneDrive share can be accessed are also included in the configuration.

Screenshots



```
// Token: 0x06000003 RID: 3 RVA: 0x00002090 File Offset: 0x00000290
public static string Decrypt(byte[] key, byte[] data)
{
    Aes aes = new AesCryptoServiceProvider
    {
        Key = key,
        IV = data.Take(key.Length / 2).ToArray<byte>()
    };
    ICryptoTransform cryptoTransform = aes.CreateDecryptor();
    return Encoding.UTF8.GetString(cryptoTransform.TransformFinalBlock(data, aes.IV.Length, data.Length -
      aes.IV.Length));
}
```

Figure 8 - This is the AES encryption routine. The routine uses the hard-coded string 'M(xcHq88q[s=pc7^+u_Gb_}JC%QQwP:h' as the AES key and the first half of the key as the IV.

0b01f392fa030be1ddd549fb79cf280d2a2c745578a56fedd4cb5e9438ae72cb

Details

Name	ntstatus.bat
Size	91 bytes
Type	ASCII text, with CRLF line terminators
MD5	d287a50bd0b95d1f153dc071d43e45d3
SHA1	cf1d9da39f4847ee735d46157232585068387763
SHA256	0b01f392fa030be1ddd549fb79cf280d2a2c745578a56fedd4cb5e9438ae72cb
SHA512	1507fd6f41c853f84b7b036280ac6c21556ce5cf10b4008c2902020291255b5bb55e63ebda9921032fd8ebf7f9fd8fff bb7de40e696601bee1486a6155b2a5ed
ssdeep	3:nlKsoFDLAdAIvVNIGfMMAylJooORKQExLAdAn:n25ABvoGfdICFRZENAC
Entropy	4.579538

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

0b01f392fa...	Created_By	1352dbb093a337eb8db9d0135adbe0542bb7 e7163616e4f8962919becab171da

0b01f392fa...	Related_To	157a0ffd18e05bfd90a4ec108e5458cbde0101 5e3407b3964732c9d4ceb71656

Description

This artifact is a batch file (.bat) that terminates the current process of ntstatus.exe (157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656). It then changes to the directory C:\windows \modemlogs\ and invokes a new instance of ntstatus.exe.

5ba0d0bfda372c1f6aa382a70f4ab8427ec998b680510e208fdf878cfda9afe3

Details

Name	ntstatus.log
Size	17520 bytes
Type	data
MD5	5753ddd324c2054718252c834d93aac9
SHA1	a2e852b0d911ced7011a7b954fc379c0d0564fc5
SHA256	5ba0d0bfda372c1f6aa382a70f4ab8427ec998b680510e208fdf878cfda9afe3
SHA512	c326d682fdad505f414bbbbbbcd219d40f8f9948c40ffcfd28a5ac5d9cfec647d5f2712ea23eb79bfafd19edfb49577a75f0f99c616abc444da62820eeee4dc6
ssdeep	384:VEiJb1Xwe87kARzd/CT74lZzRdNKHa7QYopmafni+/5vFdlg:VONdKgVm8Qognie5vFdlg
Entropy	7.989546

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

5ba0d0bfda...	Created_By	1352dbb093a337eb8db9d0135adbe0542bb7 e7163616e4f8962919becab171da

Description

This artifact is a log file created by the OneDriveClient.UploadedFiles function contained in the file result.exe (1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da). The file contains the MD5 hash of each file that has been uploaded to the remote server.

0b7d15968d44710b3e7f153c04b5038d03900a6685643bc8efe688c4d5a5deab

Details

Name	ntstatus_temp.log
Size	17520 bytes
Type	data
MD5	adfac9c5ef66c21b85fde6503c025b58
SHA1	d7950ad0cc1798f2184be502fcb12bc0a6f27864
SHA256	0b7d15968d44710b3e7f153c04b5038d03900a6685643bc8efe688c4d5a5deab
SHA512	f14a0b26627b15f628a702deca3ec1696c518cdd05f70426d5a4631a8ec6ced60ab96bfdadcbb362c27932de9a95f4794656379a5512eac3774f84e569fe2671
ssdeep	384:gyf7wfPR70mHa7Kdghm5dnB9Yr+DLPim849pbm0NNzt0B1rzLw2nd:wBvKKdghAB9YreLPF84r1N5t0B1XT
Entropy	7.990357

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

0b7d15968d...	Created_By	1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da

Description

This artifact is a log file created by the OneDriveClient.UploadedFiles function contained in the file result.exe (1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da). The file contains the MD5 hash of the file path for each file that has been uploaded to the remote server.

3585c3136686d7d48e53c21be61bb2908d131cf81b826acf578b67bb9d8e9350

Tags

downloader information-stealer trojan uploader

Details

Name	mqsvn.exe
Size	114688 bytes
Type	PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows
MD5	63cf36ac25788e13b41b1eb6bfc0c6b6
SHA1	22ab6af92ddd984bd054c21799742a5e498e8453
SHA256	3585c3136686d7d48e53c21be61bb2908d131cf81b826acf578b67bb9d8e9350
SHA512	52ecffb0004f5aee6f3a0c7e0edcbe1079845e20a712ac26854921dea9b46ece0d5f89698e833804ebdc9c3f525a8cc8c7a6d781b0caf3164b81cea17edae5c8
ssdeep	3072:KNcJNunM5p0TKWODtcT1hR3o92JoeEcfcEcKHWjUNSGdyRCOKFWc70OrZKqaJjLt:Kyf0M5p0TKWwcBhR3o92JoRcJhHMUNSz
Entropy	5.801283

Antivirus

IKARUS	Trojan.MSIL.Crypt

YARA Rules

- rule CISA_10365227_02 : ClientUploader
```
{
  meta:
    Author = "CISA Code & Media Analysis"
    Incident = "10365227"
    Date = "2021-12-23"
    Last_Modified = "20211224_1200"
    Actor = "n/a"
    Category = "n/a"
    Family = "n/a"
    Description = "Detects ClientUploader_mqsvn"
    MD5_1 = "63cf36ac25788e13b41b1eb6bfc0c6b6"
    SHA256_1 = "3585c3136686d7d48e53c21be61bb2908d131cf81b826acf578b67bb9d8e9350"
  strings:
    $s1 = "UploadSmallFileWithStopWatch"
```

```
        $s2 = "UploadPartWithStopwatch"
        $s3 = "AppVClient"
        $s4 = "ClientUploader"
        $s5 = { 46 69 6C 65 43 6F 6E 74 61 69 6E 65 72 2E 46 69 6C 65 41 72 63 68 69 76 65 }
        $s6 = { 4F 6E 65 44 72 69 76 65 43 6C 69 65 6E 74 2E 4F 6E 65 44 72 69 76 65 }
    condition:
        uint16(0) == 0x5a4d and all of them
}
```

ssdeep Matches

No matches found.

PE Metadata

Compile Date	2021-06-30 15:10:41-04:00
Company Name	Microsoft Corporation
File Description	AppVClient.exe
Internal Name	None
Legal Copyright	© Microsoft Corporation. All rights reserved.
Original Filename	None
Product Name	AppVClient.exe
Product Version	10.0.19041.84

PE Sections

MD5	Name	Raw Size	Entropy
bdd5c1c64355001493f1f48cc64646a3	header	512	2.279615
204dc02c928d7206969d5e40f4ed4de4	.text	112640	5.814718
c574847bfb2e8be8830c3d846238d2d6	.rsrc	1536	4.261328

Packers/Compilers/Cryptors

Microsoft Visual C++ v6.0

Relationships

3585c31366...	Used	25afc6741abfa27f5b50844331772466182ebe3f74bc84f911314d1a68c62cb2
3585c31366...	Created	603e75db59285734cfb5a469e984c4e359e660ccb7836ff9c209aec36931bc2b

Description

This artifact is a variant of CovalentStealer malware. The program is a file management system that is capable of uploading files to the Internet.

This variant of CovalentStealer malware contains two main modules, FileContainer and OneDriveClient, with the following functions:

```
—Begin Functions—
ClientUploader.Program<Main>
FileContainer.FileArchive<Add>
FileContainer.FileStorage<GetData>
OneDriveClient.OneDriveChannel<Send>
OneDriveClient.OneDrive<GetAccessToken>
OneDriveClient.OneDrive<UploadData>
OneDriveClient.OneDrive<UploadFile>
OneDriveClient.OneDrive<UploadLargeFile>
OneDriveClient.OneDrive<GetUploadUrl>
OneDriveClient.OneDrive<UploadPartWithStopwatch>
OneDriveClient.OneDrive<UploadPart>
OneDriveClient.OneDrive<UploadSmallFileWithStopWatch>
OneDriveClient.OneDrive<UploadSmallFile>
—End Functions—
```

The FileContainer module is used to enumerate and categorize files on the system. This module is capable of generating an MD5 hash of each file and compressing files using the Gzip or Brotli algorithms. The OneDriveClient module is used to upload files to a Microsoft Azure server on the Internet.

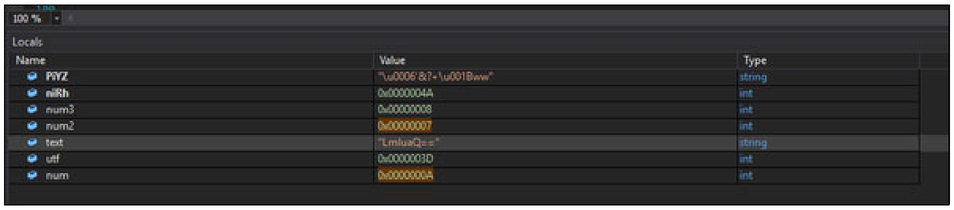
The program will look for a configuration file with the same name as the application and the .ini extension, e.g. mqsvn.ini (25afc6741abfa27f5b50844331772466182ebe3f74bc84f911314d1a68c62cb2). Alternatively, if this file is not found it will look for the file 'config.ini' (See Figure 9).

The configuration file is decoded using the AES-256-CBC key M(xcHq88q[s=pc7^+u_Gb_}JC%QQwP:h that is derived from the de-serialized string TSh4Y0hxODhxW3M9cGM3Xit1X0diX31KQyVRUXdQOmg= embedded in the file. The first 16 bytes of the key are then used as an IV (See Figure 8 above).

Other strings were de-serialized to provide additional parameters for the malware program. For example, the string LmJtcDsuanBnOy5qcGVnOy50aWZmOy50AWV7LnBuZw== decoded to a block list of files that the program is supposed to skip containing the extensions '.bmp;.jpg;.jpeg;.tiff;.tif;.png' and the string LmRvY3g7Lnhsc3g7LnBwdHg= decoded to a list of file extensions that the program is supposed to compress before encrypting and exfiltrating. The extensions included '.docx;.xlsx;.pptx' (See Figure 10).

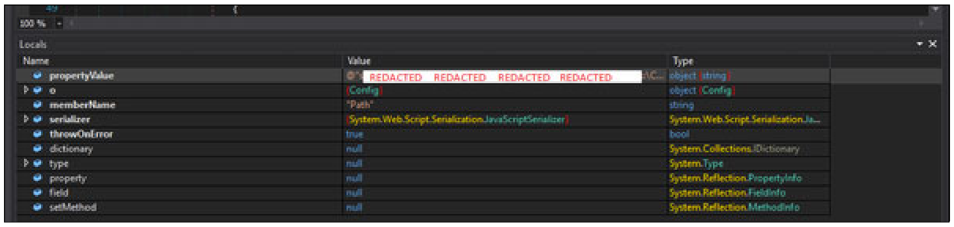
The configuration file contains a refresh token for an OAuth client for Microsoft Azure as well as a ClientID. In addition, it contains a path to the files targeted for uploading, upload times, an encryption key to encrypt the files before uploading, and compression parameters.

Screenshots



```
Locals
Name      Value                    Type
PiYZ      "\u0006'&?+\u001Bww"     string
niRh      0x0000004A               int
num3      0x00000008               int
num2      0x00000007               int
text      "LmluaQ=="               string
utf       0x0000003D               int
num       0x0000000A               int
```

Figure 9 - The ClientUploader program attempts to load a configuration file with an .ini extension from the current directory. The base64 encoded string 'Lmlua@==' represents the .ini extension.



```
Locals
Name            Value                                                   Type
propertyValue   @"REDACTED REDACTED REDACTED REDACTED\C...              object {string}
o               {Config}                                                object {Config}
memberName      "Path"                                                  string
serializer      {System.Web.Script.Serialization.JavaScriptSerializer}  System.Web.Script.Serialization.Ja...
throwOnError    true                                                    bool
dictionary      null                                                    System.Collections.IDictionary
type            null                                                    System.Type
property        null                                                    System.Reflection.PropertyInfo
field           null                                                    System.Reflection.FieldInfo
setMethod       null                                                    System.Reflection.MethodInfo
```

FIgure 10 - The ClientUploader program uses the JavaScriptSerializer routine to decode the parameters required to harvest and upload the documents.

25afc6741abfa27f5b50844331772466182ebe3f74bc84f911314d1a68c62cb2

Details

Name	mqsvn.ini
Size	800 bytes
Type	data
MD5	14b8e37952e1f532be9db40f654e6ac7
SHA1	01d6b5df5761904b7c8c6c4e34490675d4fa0f36
SHA256	25afc6741abfa27f5b50844331772466182ebe3f74bc84f911314d1a68c62cb2
SHA512	c427510f53e54eeea55e2b747bb58f46488f983c47699772d774a94038bc16b12d332741db958c63324258130b9d0376ae2687d5e7a622d9a853717680833f56
ssdeep	24:Y4yqp1BHGwUtSiW0nwPQV1iIN1RBZchbLWuL6e7ZeY:tyqLBm9tSawPPIn7Kqm7t
Entropy	7.761942

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

25afc6741a...	Used_By	3585c3136686d7d48e53c21be61bb2908d13 1cf81b826acf578b67bb9d8e9350

Description

This artifact is the encrypted configuration file for the OneDriveClient module contained in the file mqsvn.exe (3585c3136686d7d48e53c21be61bb2908d131cf81b826acf578b67bb9d8e9350). The data is decrypted using the de-serialized key M(xcHq88q[s=pc7^+u_Gb_}JC%QQwP:h found in mqsvn.exe, detailed in this report.

The file contains a path to an archive targeted by the attacker. The file includes the AES-256-CBC key 1khvo39Q2evpi**&R$*^Rjhko8tve2b7 that is used to encrypt the harvested documents before they are uploaded to the Internet.

In addition, the data contains a refresh token for an OAuth client for Microsoft Azure with the Client ID of '7a3b4b84-ed28-4f18-b30d-218788c74a5f'. Speed and compression information as well as times that the OneDrive share can be accessed are also included in the configuration. NOTE: The decrypted configuration contains confidential client information and is therefore not included in this report.

603e75db59285734cfb5a469e984c4e359e660ccb7836ff9c209aec36931bc2b

Details

Name	mqsvn.log
Size	39504 bytes
Type	data
MD5	444ccf674588f47ab5638fb08db98b01
SHA1	4fcf2c22d2ea70430580b487a7834c165deee5d0
SHA256	603e75db59285734cfb5a469e984c4e359e660ccb7836ff9c209aec36931bc2b
SHA512	843cdead51e290ee5466f51f316c5199259b7e55b752efbdcfa83a5c64a0477a4ddcd3ab63785e9e25c01095670073 884943fa0419797c0b74d30a9ae240d0cf
ssdeep	768:eYarzB8pLwTFL/FX8ANpGMVYO5kELiD4Z8xKzvkA6A3zZesChaFRR:eYaXB8pKF18ANkMX6ELh8xivpzZDC4FH
Entropy	7.995061

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

603e75db59...	Created_By	3585c3136686d7d48e53c21be61bb2908d13 1cf81b826acf578b67bb9d8e9350

Description

This artifact contains encrypted MD5 hashes of files that have been uploaded to the Internet by the file mqsvn.exe (3585c3136686d7d48e53c21be61bb2908d131cf81b826acf578b67bb9d8e9350).

30191b3badf3cdbc65d0ffeb68e0f26cef10a41037351b0f562ab52fce7432cc

Tags

information-stealer obfuscated uploader

Details

Name	msexch.exe
Size	6656 bytes
Type	PE32+ executable (GUI) x86-64 Mono/.Net assembly, for MS Windows
MD5	baa634fdd2b34956524b5519ee97b8a8
SHA1	cdc7e3b6905f69d8330c4b0f71494a7db7ac61e7
SHA256	30191b3badf3cdbc65d0ffeb68e0f26cef10a41037351b0f562ab52fce7432cc
SHA512	cdcd245fc1dc5072918950b1950527f0b6284453f527623cb600afc775f2cde507278273c75b4af972ac976c06fa73d414350b92c24c7a1dec44aa05527ca532
ssdeep	96:LDuLc7D604Vp9Rzj1HhaUA3zvDwi0MX7gtKflUTsqzNt:LDuw6rVd3aP7Dw9MEQmT
Entropy	4.869180

Antivirus

Adaware	Gen:Variant.Tedy.82790
Bitdefender	Gen:Variant.Tedy.82790

YARA Rules

```
• rule CISA_10365227_01 : APPSTORAGE
  {
    meta:
      Author = "CISA Code & Media Analysis"
      Incident = "10365227"
      Date = "2021-12-23"
      Last_Modified = "20211224_1200"
      Actor = "n/a"
      Category = "n/a"
      Family = "APPSTORAGE"
      Description = "Detects AppStorage_ntstatus_msexch samples"
      MD5_1 = "c435d133b45783cce91a5d4e4fbe3f52"
      SHA256_1 = "157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656"
      MD5_2 = "baa634fdd2b34956524b5519ee97b8a8"
      SHA256_2 = "30191b3badf3cdbc65d0ffeb68e0f26cef10a41037351b0f562ab52fce7432cc"
    strings:
      $s1 = "026B924DD52F8BE4A3FEE8575DC"
      $s2 = "GetHDDId"
      $s3 = "AppStorage"
      $s4 = "AppDomain"
      $s5 = "$1e3e5580-d264-4c30-89c9-8933c948582c"
      $s6 = "hrjio2mfsdlf235d" wide
    condition:
      uint16(0) == 0x5a4d and all of them
  }
```

ssdeep Matches

No matches found.

PE Metadata

Compile Date	2083-06-18 19:48:42-04:00
Internal Name	AppStorage.exe
Original Filename	AppStorage.exe
Product Version	1.0.0.0

PE Sections

MD5	Name	Raw Size	Entropy
9b75c9220e4242a6403f02bb9da3d198	header	512	2.261868
a69c4d0928332121839c97d955246112	.text	4608	5.236469
0551c676439e5d812cb2bab3f2060c1b	.rsrc	1536	3.934855

Packers/Compilers/Cryptors

Microsoft Visual C++ v6.0

Relationships

30191b3bad...	Related_To	e03a2c8a6e81cf62ba7401c598ea1d4635b08bbf9c2fec080b536dde29e6392f
30191b3bad...	Dropped	d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8

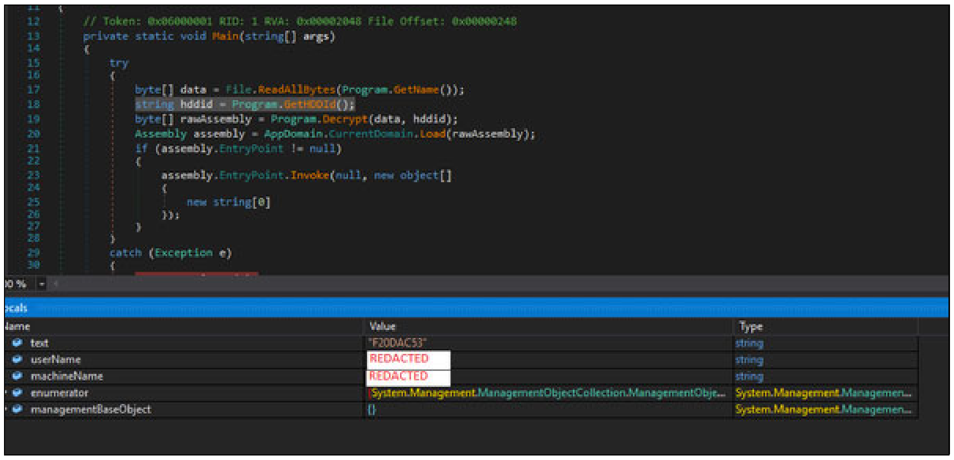
Description

This artifact is an obfuscated .NET executable that is used to decode a variant of the CovalentStealer malware. When executed, the program will check the present name of the program and then look in the current directory for a file with the same name and a .bin extension, e.g. msexch.bin (e03a2c8a6e81cf62ba7401c598ea1d4635b08bbf9c2fec080b536dde29e6392f).

The program seeks to generate a key called 'HDDId' to decode msexch.bin. The embedded string 'hrjio2mfsdlf235d' is used to decode instructions within the program to generate the key (See Figure 3 above). This function is similar to the function described in ntstatus.exe detailed elsewhere in this report, however it will take one additional variable to generate the key. The first command identifies the current userName on the system while the second command identifies the machineName. The third command reads the WMI namespace root/cimv2 to locate the volumeserialnumber of the current drive. All of the variables are then modified using an XOR routine and the same string above is used to generate the key (See Figure 11). The first part of the key is generated from the volume serial number which, during analysis resolved to '76D55BD2'. The second part of the key is resolved from the userName, which during analysis resolved to '34BD153B'. The last part of the key is resolved from the machineName, which resolved to 'F3124EDD' creating the key '76D55BD234BD153BF3124EDD' (See Figure 12). Note: The key is an example.

To generate the correct key, the userName, machineName, and volumeserialnumber must match the victim's system, otherwise it fails to decode msexch.bin and the program will terminate. This method is used to thwart independent analysis of the file, msexch.bin.

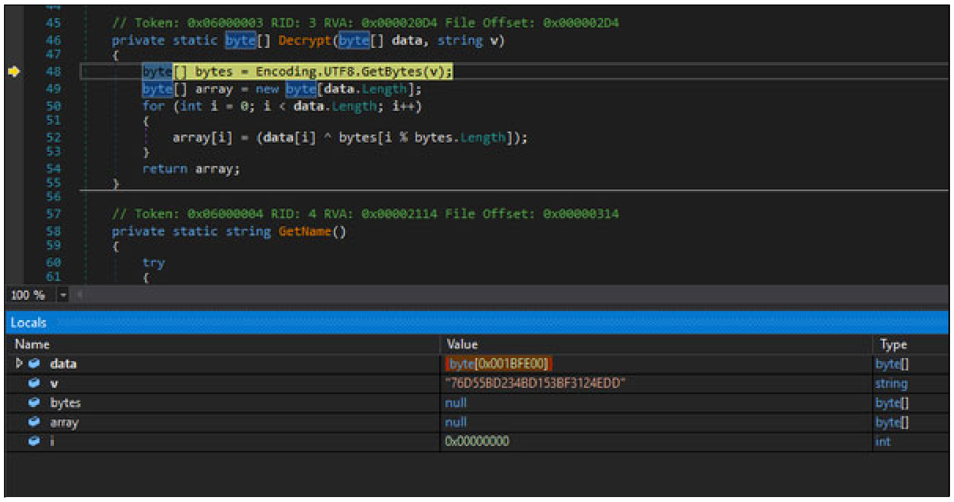
Screenshots



```
    // Token: 0x06000001 RID: 1 RVA: 0x00002048 File Offset: 0x00000248
    private static void Main(string[] args)
    {
        try
        {
            byte[] data = File.ReadAllBytes(Program.GetName());
            string hddid = Program.GetHDDId();
            byte[] rawAssembly = Program.Decrypt(data, hddid);
            Assembly assembly = AppDomain.CurrentDomain.Load(rawAssembly);
            if (assembly.EntryPoint != null)
            {
                assembly.EntryPoint.Invoke(null, new object[]
                {
                    new string[0]
                });
            }
        }
        catch (Exception e)
        {
```

```
Locals
Name                      Value                                                        Type
  text                    "F20DAC53"                                                   string
  userName                REDACTED                                                     string
  machineName             REDACTED                                                     string
  enumerator              {System.Management.ManagementObjectCollection.ManagementObje...  System.Management.Managemen...
  managementBaseObject    {}                                                           System.Management.Managemen...
```

Figure 11 - The program collects the userName, machineName, and Volume Serial Number to generate the HDDId key.



```
    // Token: 0x06000003 RID: 3 RVA: 0x000020D4 File Offset: 0x000002D4
    private static byte[] Decrypt(byte[] data, string v)
    {
        byte[] bytes = Encoding.UTF8.GetBytes(v);
        byte[] array = new byte[data.Length];
        for (int i = 0; i < data.Length; i++)
        {
            array[i] = (data[i] ^ bytes[i % bytes.Length]);
        }
        return array;
    }

    // Token: 0x06000004 RID: 4 RVA: 0x00002114 File Offset: 0x00000314
    private static string GetName()
    {
        try
        {
```

```
100 %
Locals
Name        Value                           Type
data        byte[0x001BFE00]                byte[]
v           "76D55BD234BD153BF3124EDD"      string
bytes       null                            byte[]
array       null                            byte[]
i           0x00000000                      int
```

Figure 12 - Screenshot of the generated HDDId key used to decode msexch.bin.

e03a2c8a6e81cf62ba7401c598ea1d4635b08bbf9c2fec080b536dde29e6392f

Tags

information-stealer obfuscated uploader

Details

Name	msexch.bin
Size	1834496 bytes
Type	data
MD5	bd95f0df1272e5b2854b304c71930168
SHA1	2d28c56daf370370d1c4d95fd25e4f0a04ceda07
SHA256	e03a2c8a6e81cf62ba7401c598ea1d4635b08bbf9c2fec080b536dde29e6392f
SHA512	b01a5b459f0b3b619b742f717e7b536cf713dded36b542d5546a59333c6008aaab0c844a9979b4450dc1a1ced5af41beebfda41191920a678026c63fdf7934dd
ssdeep	24576:KNCSFczkVbstNn2l4Evj6ZalDLdjFu1u1Ww1YfduAiG52Qqlsvz66ZG+b38tTnt4:hz7ePzJuss4caq
Entropy	6.682404

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

e03a2c8a6e...	Related_To	30191b3badf3cdbc65d0ffeb68e0f26cef10a41037351b0f562ab52fce7432cc

Description

This is an obfuscated version of CovalentStealer malware. The file is decoded by msexch.exe using the key '76D55BD234BD153BF3124EDD'. The decoded file is called Uploader.exe (d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8) and is detailed in this report.

d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8

Tags

information-stealer uploader

Details

Name	Uploader.exe
Size	1834496 bytes
Type	PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows
MD5	f54ae2b0d51bb4cdc2a142733f122311
SHA1	184adab2435e4b0f9b02521fed5e56390b5e775f
SHA256	d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8
SHA512	97ed8086dde00af3cbf51c02073aec28957a6bf354799f489ee7c457e82e0b21d7d2fb6ba46589675ed22d51aa0d973ab7d4132a2aeeb0adf15da618d4fb83cd
ssdeep	49152:Z2f6rfgMSneK065JIYaDmxZF5ax00MSMoOKiYyBg9FzfJNFL5QPWES2s1B+dBrSC:Z2f6rfgMSneK065JIYaDmxZF5ax00MSt
Entropy	5.580993

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

97	1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da

PE Metadata

Compile Date	2021-09-24 14:56:17-04:00
Import Hash	f34d5f2d4577ed6d9ceec516c1f5a744
Internal Name	ClientUploader.exe
Original Filename	ClientUploader.exe
Product Version	1.0.0.0

PE Sections

MD5	Name	Raw Size	Entropy
a1eef53765269a304aaa217af7ede436	header	512	2.725476
489bbfac9377f3ef9a60f9d64d9ccda8	.text	1831936	5.583032
5488f249cf62feed84546911d54f96f2	.rsrc	1536	3.971470
fbf8fada938118d358a40e73eb0c8bb9	.reloc	512	0.101910

Relationships

d221ca9c51...	Used	52765525103f5b3b07d0882cc8ee4bb8e279ad5d451e1ed07cae3b98565cce29
d221ca9c51...	Created	09605981a072c604e6ef9ad2dd7d2a78b48b07ee3339589bfcf0a466a9190904
d221ca9c51...	Created	6a0cd866c849e62f9ccc26575d8794c2e0b14722387742b965d4358e1e0e8b3c
d221ca9c51...	Dropped_By	30191b3badf3cdbc65d0ffeb68e0f26cef10a41037351b0f562ab52fce7432cc

Description

This artifact is a variant of the CovalentStealer program. When the program is executed it will decrypt and read the configuration file msexch.ini (52765525103f5b3b07d0882cc8ee4bb8e279ad5d451e1ed07cae3b98565cce29) in the current directory. It uses the hard-coded AES-256-CBC key 'M(xcHq88q[s=pc7^+u_Gb_}JC%QQwP:h' to decrypt the file. The configuration file will include a path to the directory containing the targeted files, compression parameters, and connection parameters for connecting to a system on the Internet to upload data.

ClientUploader has several primary modules. The module IFileWorker contains the following functions:

—Begin IFileWorker Functions—
Brotli. – This function contains the Brotli compression library to compress and decompress files.
ContainersFilesWorker. – This function keeps track of uploaded files. It compares the files to a hash list for the file and path before uploading and also compares them to a whitelist and a blacklist by file extension. It also logs the status of each file in the upload process.
Extension. – This function checks the file extension to determine if the file needs to be compressed.
File Archive. – This function verifies the size of the file and disposition before compressing the file.
FileBlock. – This function converts the file data into a byte stream.
FileContainers. – This function segregates files by file type based on the extension.
GZip. – This function contains the Gzip compression library to compress and decompress files.
Logger. – This function logs debug status messages and telemetry data from other functions and outputs them to a file using the base name and the .dat extension, e.g. msexch.dat (See Figure 4 above).
WhiteAndBlackList. – This function maintains a list of files by name and a list of files by extension that match the whitelist or blacklist from the configuration file.
—End IFileWorker Functions—

The module OneDriveClient contains the following functions:

—Begin OneDriveClient Functions—
OneDrive. – This function uploads files to a URL. It configures speed, buffer size, time, etc. based on the parameters in the configuration file, msexch.ini. Then, it reports the status of each file to the IFileWorker.Logger function. The following are examples of the OneDrive commands:

```
—Begin OneDrive Commands—
OneDriveClient.OneDriveChannel+<Send>
OneDriveClient.OneDrive+<GetAccessToken>
OneDriveClient.OneDrive+<UploadData>
OneDriveClient.OneDrive+<UploadFile>
OneDriveClient.OneDrive+<UploadLargeFile>
OneDriveClient.OneDrive+<GetUploadUrl>
OneDriveClient.OneDrive+<UploadPartWithStopwatch>
OneDriveClient.OneDrive+<UploadPart>
OneDriveClient.OneDrive+<UploadSmallFileWithStopWatch>
OneDriveClient.OneDrive+<UploadSmallFile>
—End OneDriveClient Functions—
```

OneDriveChannel. – This function establishes the connection to server.
OneDriveChannelSettings. – This function reads the ClientID, Redirect, Refresh Token, and Scopes from the configuration file, msexch.ini to negotiate the connection to the client.
UploadedFiles. – This function logs the hash and the file path of the uploaded files and records the information into two files where msexch.log contains a list of file hashes and msexch_temp.log contains a list of file path hashes (See Figure 7 above).
—End OneDriveClient Functions—

The program also contains supporting libraries for the SMB protocol versions 2 and 3. The libraries have the capacity to maintain a list of IP addresses, logins, domainNames, passwords, and SMB clients that can be used to attempt to search for and log into SMB file stores. Files can be searched by file path, file status (e.g., open or closed), and file attributes (e.g. shared, read only, etc.).

52765525103f5b3b07d0882cc8ee4bb8e279ad5d451e1ed07cae3b98565cce29

Tags

information-stealer uploader

Details

Name	msexch.ini
Size	4816 bytes
Type	data
MD5	d3951137283e84d42f85bb91f0ccfcdd
SHA1	450982b1420a97dcedb15fb058e00e108d240bb7

SHA256	52765525103f5b3b07d0882cc8ee4bb8e279ad5d451e1ed07cae3b98565cce29
SHA512	082594fced158d5597e1b34ec220fd873365f3ec282add680fc84d4b31010c2485e97611049c2d1432b6a1014784e06d3b11f14a815252a28c0c38c4eb5a31e1
ssdeep	96:XaMTeYZR1Bm3AboPwVUJyWvihHbP11Ho+5EGsW7MlDz1v7Yrtgx3X:XaWZZR1Bx9VP16+5jRQlDR8U
Entropy	7.963703

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

5276552510...	Used_By	d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8

Description

This artifact is the encrypted configuration file for the OneDriveClient module contained in the file Uploader.exe (d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8) detailed in this report. The data is decrypted using the hard-coded AES-256-CBC key ‘M(xcHq88q[s=pc7^+u_Gb_}JC%QQwP:h’. The algorithm uses an IV that is derived from the first half of the encryption key (See Figure 8 above).

The file contains multiple paths to archives targeted by the attacker. The file includes the IP address of the server, stolen credential information, and a key to encrypt the uploaded data. NOTE: The decrypted configuration contains confidential client information and therefore is not included in this report.

In addition, the data contains a refresh token for an OAuth client for Microsoft Azure with the Client ID of ‘7a3b4b84-ed28-4f18-b30d-218788c74a5f’. Speed and compression information as well as times that the OneDrive share can be accessed are also included in the configuration.

09605981a072c604e6ef9ad2dd7d2a78b48b07ee3339589bfcf0a466a9190904

Details

Name	msexch.log
Size	103904 bytes
Type	data
MD5	30ea2a37c7174ed8c3ab88aecee0002b
SHA1	3a6f2826aab7948d8b930f6bf13897160c198807
SHA256	09605981a072c604e6ef9ad2dd7d2a78b48b07ee3339589bfcf0a466a9190904
SHA512	0a78caf6257b8b58578181a9555bf9cee24b1bfced078855145f79757701a53a15968d9bb6acc74fdc9469bd28fa82a53b8d52669fa3952824f51339bd94ad7a
ssdeep	3072:OcopRvQlpMV/EN6PmW9tV/PUdpogFeSQx7:CpVFp8/pFhPUdponR7
Entropy	7.998490

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

09605981a0...	Created_By	d221ca9c519ae04c7724baca8d36c2ce77454 e0f9aa0f119ecfa9246973a92f8

Description

This artifact is a log file created by the OneDriveClient.UploadedFiles function contained in the file Uploader.exe (d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8). The file contains the MD5 hash of each file that has been uploaded to the remote server.

6a0cd866c849e62f9ccc26575d8794c2e0b14722387742b965d4358e1e0e8b3c

Details

Name	msexch_temp.log
Size	103904 bytes
Type	data
MD5	20b7eb0af9b9e7403a298f7966d5a1d4
SHA1	b2018e61e8b435b6a172b35774377ebc16fd0168
SHA256	6a0cd866c849e62f9ccc26575d8794c2e0b14722387742b965d4358e1e0e8b3c
SHA512	3695120b452c103f54c4eb738648621f162850ec32aca734ecdd552755ecced1500aaf789ec1bf45afc5df4fcfd6144ca4d1fff415a25656dd5493f81b221bfe
ssdeep	3072:2H05Z4/LivljqjSXZa8HaDhpfUcJkm0YK/:29ivlmjSX9qnUcdi
Entropy	7.998385

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

6a0cd866c8...	Created_By	d221ca9c519ae04c7724baca8d36c2ce77454 e0f9aa0f119ecfa9246973a92f8

Description

This artifact is a log file created by the OneDriveClient.UploadedFiles function contained in the file Uploader.exe (d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8). The file contains the MD5 hash of the path for each file that has been uploaded to the remote server.

fae38156e9ce12368c846836b87861f4f12e14698cb65f14545205fa56d8c496

Tags

information-stealer

Details

Name	vmware.ps1
Size	10436 bytes
Type	ASCII text
MD5	4825b1e32ff062f4671d5420661695af
SHA1	0cbf85f88e2fb0bc721357acdd543d5a1957886f
SHA256	fae38156e9ce12368c846836b87861f4f12e14698cb65f14545205fa56d8c496
SHA512	a58298346cdf35e432d755942ef2690c6e3182a4fab03df163142e42cdcb0d7bc3810c647078a779d15ee0676b0eacfa59c38512671dc86264b42f2c8d69edb8

ssdeep	192:k9XNMA6GyvE0XJvP0EN3ab3Akz9JUWCUVCRB7/dUV /TpraVm5efUo9wQUyfa3gpA:k9XNMA6pXJvPCUjUmUvaME8obUaYgpj8
Entropy	4.979828

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Description

This artifact is a script called Export-MFT.ps1 written in PowerShell used to collect the MFT from a system volume. The benign open source script is available on GitHub.

bfa7adeda4597b70bf74a9f2032df2f87e07f2dbb46e85cb7c091b83161d6b0a

Details

Name	vmware.exe
Size	497104 bytes
Type	PE32 executable (console) Intel 80386, for MS Windows
MD5	0acb06da48d86e1ef15c27a4f5a3bddd
SHA1	12dd7a86001ff2b6b661cd7de60ca6aadc9b78ae
SHA256	bfa7adeda4597b70bf74a9f2032df2f87e07f2dbb46e85cb7c091b83161d6b0a
SHA512	98fbcd4e190e0bc17dc712bbbe808c7d24610c334925381544fb16a8f75931db1c5f6597cafbe6a12a9050e482e553 51bedb76b40573f8a7489e3c7755bdecd2
ssdeep	12288:1NsUjyDukqiudnJkx3piQLmGLvdnTJ0CRUyF1l3Kl:1mkyDuZiCccQLmGpTrCm1l3g
Entropy	6.459391

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

PE Metadata

Compile Date	2014-12-02 05:07:13-05:00
Import Hash	1324fa350b5f878451cc28b429b96e9b
Company Name	Alexander Roshal
File Description	Command line RAR
Internal Name	Command line RAR
Legal Copyright	Copyright © Alexander Roshal 1993-2014
Original Filename	None
Product Name	WinRAR
Product Version	5.20.0

PE Sections

MD5	Name	Raw Size	Entropy
98efedab8c1234a79df40e93dc82e136	header	1024	2.635435

0b760a9dbbf12c5d32ca265879aabdb2	.text	410112	6.587893
3874d7a1d17b892215dc07687ac3b75c	.rdata	27136	4.857459
e28ebcc7f9a5e3d463ee9d9de071e085	.data	8192	3.720474
5ad98aabb9c5996ee180a98ff9543866	.rsrc	31232	3.540367
ec534cec214c136ef4552b79103e2eaa	.reloc	14336	5.427399

Packers/Compilers/Cryptors

Microsoft Visual C++ ?.?

Description

This artifact is a benign publicly available version of the Roshal archiver (RAR), version 5.20.0. RAR.exe is used to compress and archive other files.

Relationship Summary

84164e1e80...	Used	91a8b31c126a021f5c156742016acdcca7d83eac4b583bae5d4fd0a85a96813b
84164e1e80...	Created	517faa4a0666ec68842f256f08d987935b6ce9ef64e33f027e084e8f45b9366d
517faa4a06...	Created_By	84164e1e8074c2565d3cd178babd93694ce54811641a77ffdc8d1084dd468afb
91a8b31c12...	Used_By	84164e1e8074c2565d3cd178babd93694ce54811641a77ffdc8d1084dd468afb
157a0ffd18...	Related_To	b03ac5eaf2131060ee381e5e46ebc705d8d617a90cc61fa4918174545b4fbaa6
157a0ffd18...	Dropped	1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da
157a0ffd18...	Related_To	0b01f392fa030be1ddd549fb79cf280d2a2c745578a56fedd4cb5e9438ae72cb
b03ac5eaf2...	Related_To	157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656
b03ac5eaf2...	Contains	1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da
1352dbb093...	Created	5ba0d0bfda372c1f6aa382a70f4ab8427ec998b680510e208fdf878cfda9afe3
1352dbb093...	Created	0b7d15968d44710b3e7f153c04b5038d03900a6685643bc8efe688c4d5a5deab
1352dbb093...	Used	da267c72f58ec487761de99d0f3bcfd87771a36afc06716053960633a74139df
1352dbb093...	Dropped_By	157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656
1352dbb093...	Created	0b01f392fa030be1ddd549fb79cf280d2a2c745578a56fedd4cb5e9438ae72cb
1352dbb093...	Contained_Within	b03ac5eaf2131060ee381e5e46ebc705d8d617a90cc61fa4918174545b4fbaa6
da267c72f5...	Used_By	1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da
0b01f392fa...	Created_By	1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da
0b01f392fa...	Related_To	157a0ffd18e05bfd90a4ec108e5458cbde01015e3407b3964732c9d4ceb71656
5ba0d0bfda...	Created_By	1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da
0b7d15968d...	Created_By	1352dbb093a337eb8db9d0135adbe0542bb7e7163616e4f8962919becab171da

3585c31366...	Used	25afc6741abfa27f5b50844331772466182ebe3f74bc84f911314d1a68c62cb2
3585c31366...	Created	603e75db59285734cfb5a469e984c4e359e660ccb7836ff9c209aec36931bc2b
25afc6741a...	Used_By	3585c3136686d7d48e53c21be61bb2908d131cf81b826acf578b67bb9d8e9350
603e75db59...	Created_By	3585c3136686d7d48e53c21be61bb2908d131cf81b826acf578b67bb9d8e9350
30191b3bad...	Related_To	e03a2c8a6e81cf62ba7401c598ea1d4635b08bbf9c2fec080b536dde29e6392f
30191b3bad...	Dropped	d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8
e03a2c8a6e...	Related_To	30191b3badf3cdbc65d0ffeb68e0f26cef10a41037351b0f562ab52fce7432cc
d221ca9c51...	Used	52765525103f5b3b07d0882cc8ee4bb8e279ad5d451e1ed07cae3b98565cce29
d221ca9c51...	Created	09605981a072c604e6ef9ad2dd7d2a78b48b07ee3339589bfcf0a466a9190904
d221ca9c51...	Created	6a0cd866c849e62f9ccc26575d8794c2e0b14722387742b965d4358e1e0e8b3c
d221ca9c51...	Dropped_By	30191b3badf3cdbc65d0ffeb68e0f26cef10a41037351b0f562ab52fce7432cc
5276552510...	Used_By	d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8
09605981a0...	Created_By	d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8
6a0cd866c8...	Created_By	d221ca9c519ae04c7724baca8d36c2ce77454e0f9aa0f119ecfa9246973a92f8

Recommendations

CISA recommends that users and administrators consider using the following best practices to strengthen the security posture of their organization's systems. Any configuration changes should be reviewed by system owners and administrators prior to implementation to avoid unwanted impacts.

- Maintain up-to-date antivirus signatures and engines.
- Keep operating system patches up-to-date.
- Disable File and Printer sharing services. If these services are required, use strong passwords or Active Directory authentication.
- Restrict users' ability (permissions) to install and run unwanted software applications. Do not add users to the local administrators group unless required.
- Enforce a strong password policy and implement regular password changes.
- Exercise caution when opening e-mail attachments even if the attachment is expected and the sender appears to be known.
- Enable a personal firewall on agency workstations, configured to deny unsolicited connection requests.
- Disable unnecessary services on agency workstations and servers.
- Scan for and remove suspicious e-mail attachments; ensure the scanned attachment is its "true file type" (i.e., the extension matches the file header).
- Monitor users' web browsing habits; restrict access to sites with unfavorable content.
- Exercise caution when using removable media (e.g., USB thumb drives, external drives, CDs, etc.).
- Scan all software downloaded from the Internet prior to executing.
- Maintain situational awareness of the latest threats and implement appropriate Access Control Lists (ACLs).

Additional information on malware incident prevention and handling can be found in National Institute of Standards and Technology (NIST) Special Publication 800-83, **"Guide to Malware Incident Prevention & Handling for Desktops and Laptops".**

Contact Information

- 1-888-282-0870
- CISA Service Desk (UNCLASS)
- CISA SIPR (SIPRNET)
- CISA IC (JWICS)

CISA continuously strives to improve its products and services. You can help by answering a very short series of questions about this product at the following URL: https://us-cert.cisa.gov/forms/feedback/

Document FAQ

What is a MIFR? A Malware Initial Findings Report (MIFR) is intended to provide organizations with malware analysis in a timely manner. In most instances this report will provide initial indicators for computer and network defense. To request additional analysis, please contact CISA and provide information regarding the level of desired analysis.

What is a MAR? A Malware Analysis Report (MAR) is intended to provide organizations with more detailed malware analysis acquired via manual reverse engineering. To request additional analysis, please contact CISA and provide information regarding the level of desired analysis.

Can I edit this document? This document is not to be edited in any way by recipients. All comments or questions related to this document should be directed to the CISA at 1-888-282-0870 or CISA Service Desk.

Can I submit malware to CISA? Malware samples can be submitted via three methods:

- Web: https://malware.us-cert.gov
- E-Mail: submit@malware.us-cert.gov
- FTP: ftp.malware.us-cert.gov (anonymous)

CISA encourages you to report any suspicious activity, including cybersecurity incidents, possible malicious code, software vulnerabilities, and phishing-related scams. Reporting forms can be found on CISA's homepage at www.cisa.gov.