THREAT ANALYSIS REPORT: Inside the LockBit Arsenal - The StealBit Exfiltration Tool

By Cybereason Global SOC Team

Archived: 2026-04-05 17:16:07 UTC

The Cybereason Global Security Operations Center (GSOC) issues Cybereason Threat Analysis reports to inform on impacting threats. The Threat Analysis reports investigate these threats and provide practical recommendations for protecting against them.

In this Threat Analysis report, the GSOC investigates the StealBit malware, a data exfiltration tool that the LockBit threat group develops and maintains. The LockBit group provides StealBit to affiliates as part of the group's ransomware affiliate program. Ransomware operators use StealBit to exfiltrate data from compromised systems for double extortion purposes.

This report provides an in-depth insight into the functionalities and architecture of StealBit as well as the evolution of relevant configuration and implementation aspects of StealBit across different samples. The detailed insight into how StealBit works and evolves is important for the timely detection of ransomware attack operations that involve StealBit at the point when malicious actors exfiltrate data before deploying ransomware.

StealBit Malware Key Points

Feature updates and widened target base: A comparative analysis between relatively older and newer StealBit samples shows that StealBit has been undergoing improvement with new features, especially evasion and hiding features. In addition, although older samples do not execute on systems located in the former Soviet countries Russia, Ukraine, Belarus, Tajikistan, Armenia, Azerbaijan, Georgia, Kazakhstan, Kyrgyzstan, Turkmenistan, Uzbekistan, and Moldova, newer StealBit samples do not implement this restriction and execute on any system.

Developed for maximum data exfiltration efficiency: StealBit implements the Microsoft input/output (I/O) completion port threading model to maximize the overall efficiency of data exfiltration activities. For example, StealBit parallelizes the exfiltration of the content of multiple files to shorten the overall exfiltration timespan. This is important to ransomware operators, since fast data exfiltration reduces the chances of being discovered in the process.

Developed for maximum usage convenience and scalability: StealBit implements interprocess communication (IPC) between multiple StealBit processes that run on a single compromised system to designate many files for exfiltration in a scalable manner. In addition, StealBit supports dragging and dropping of files or folders for exfiltration to StealBit windows in scenarios where the StealBit operators have access to the graphical user interface of compromised systems. This feature enables StealBit operators to designate many files for exfiltration in a convenient and scalable manner.

Somewhat incomplete implementation: The implementation of some StealBit features that we analyzed is not complete. This includes features that the LockBit threat group advertises as advantageous to alternative exfiltration tools on the underground market, such as compression of exfiltrated data and a hidden mode of operation. For example, a recent StealBit sample that we analyzed does not compress exfiltrated data and does not properly hide the windows that StealBit creates, making the malware visible in the graphical user interface of the compromised system.

StealBit Malware Detected and prevented: The Cybereason XDR Platform effectively detects and prevents StealBit when the malware exfiltrates data, and also detects and prevents the execution of the related LockBit ransomware, which LockBit affiliates may execute after they use StealBit to exfiltrate data for double extortion.

Cybereason Managed Detection and Response (MDR): The Cybereason GSOC has zero tolerance towards attacks that involve ransomware and data exfiltration tools, such as StealBit, and categorizes such attacks as critical, high-severity incidents. The Cybereason GSOC MDR Team issues a comprehensive report to customers when such an incident occurs. The report provides an in-depth overview of the incident, which helps to scope the extent of compromise and the impact on the customer's environment. In addition, the report provides attribution information when possible as well as recommendations for mitigating and isolating the threat.

StealBit Malware Introduction

The traditional ransomware extortion tactic, where malicious actors demand payment for decrypting data that the actors have encrypted using ransomware, does not always work as intended. Victims may not pay ransom for several reasons, such as lack of financial resources, concerns that ransomware operators may not decrypt the data, or the availability of backups of the encrypted data.

Therefore, many modern ransomware operators use a double extortion tactic: ransomware operators exfiltrate data from compromised systems before encrypting the data, and if the victim refuses to pay ransom for data decryption, the malicious actors threaten to leak the exfiltrated data online or sell the data for profit.

The proliferation of double extortion on the ransomware scene marks a major turning point in the evolution of the ransomware threat, with ransomware actors massively joining in on the trend. For example, in June 2021, TrendMicro reported that it has observed 35 ransomware families that use double extortion — with a growing tendency.

Since the double extortion tactic relies on exfiltrated data, data exfiltration tools are crucial to ransomware operators that use this tactic. Ransomware operators use publicly available tools for data exfiltration, such as Rclone, as well as custom data exfiltration tools that are intended specifically for use in ransomware operations. Some custom data exfiltration tools are Ryuk Stealer, the recently discovered Exmatter, and StealBit.

The StealBit malware is a data (file content) exfiltration tool that the LockBit threat group develops and maintains. StealBit exfiltrates file content to remote attacker-controlled endpoints for double extortion purposes. In addition to StealBit, the LockBit threat group develops and maintains the LockBit ransomware, which has a strong presence on the ransomware threat scene.

As of June 2021, the LockBit group runs a ransomware affiliate program, LockBit 2.0, which provides access to the LockBit ransomware and the StealBit data exfiltration tool to affiliates. As part of affiliate recruitment efforts, the LockBit group advertises the features of the LockBit ransomware and StealBit by comparing the ransomware and StealBit to alternative solutions. The LockBit group claims that StealBit is superior, especially in terms of data exfiltration speed:

Along with the encrypting system, you get access to the fastest stealer all over the world - StealBit automatically downloading all files of the attacked company to our updated blog.

Comparative table of the information download speed of the attacked company

Testing was made on the computer with a speed of Internet of 1 gigabit per second

Downloading method	Speed in megabytes per second	Compression in real time	Hidden mode	drag'n'drop	Time spent for downloading of 10 GB	Time spent for downloading of 100 GB	Time spent for downloading of 10 TB
Stealer - StealBIT	**83,46 MB/s**	**Yes**	**Yes**	**Yes**	**1M 59S**	**19M 58S**	**1D 9H 16M 57S**
Rclone pcloud.com free	4,82 MB/s	No	No	No	**34M 34S**	**5H 45M 46S**	**24D 18M 8S**
Rclone pcloud.com premium	4,38 MB/s	No	No	No	**38M 3S**	**6H 20M 31S**	**26D 10H 11M 45S**
Rclone mail.ru free	3,56 MB/s	No	No	No	**46M 48S**	**7H 48M 9S**	**32D 12H 16M 28S**
Rclone mega.nz free	2,01 MB/s	No	No	No	**1H 22M 55S**	**13H 48M 11S**	**57D 13H 58M 44s**
Rclone mega.nz PRO	1,01 MB/s	No	No	No	**2H 45M**	**1D 03H 30M 9S**	**114D 14H 16M 30S**
Rclone yandex.ru free	0,52 MB/s	No	No	No	**5H 20M 30S**	**2D 05H 25M 7S**	**222D 13H 52M 49S**

The LockBit group advertises StealBit (source: KELA, Twitter)

This report discusses the implementation of StealBit and its internal working principles. In addition, this report provides an overview of the evolution of relevant configuration and implementation aspects of StealBit across different StealBit samples. Previous research documents some aspects of the implementation of StealBit, with a focus on automating the de-obfuscation of relevant StealBit configuration: the IP addresses of the attacker-controlled endpoints to which StealBit exfiltrates file content.

This report provides an in-depth and comprehensive insight into the functionalities, architecture, and evolution of StealBit. The detailed insight into how StealBit works and evolves is important to build proper detection and protection strategies against the malware. This, in turn, is crucial for the timely detection of ransomware operations that involve StealBit at the point when malicious actors exfiltrate data before deploying ransomware.

StealBit Malware Analysis

The Deep Dive Analysis section discusses the implementation of StealBit and its internal working principles. In this section, we focus on a recent StealBit sample with a Secure Hash Algorithm (SHA)-256 hash of *6c9a92955402c76ab380aa6927ad96515982a47c05d54f21d67603814d29e4a5*. The Comparative Analysis section compares different StealBit samples to provide an overview of the evolution of relevant configuration and implementation aspects of StealBit across the samples.

StealBit Malware Deep Dive Analysis

StealBit first checks whether the StealBit process runs in the context of a debugger by evaluating the value of the *NtGlobalFlag* field of the Process Environment Block (PEB). If the value of *NtGlobalFlag* is 0x70, StealBit executes an empty infinite loop:

```
if ( (NtCurrentPeb()->NtGlobalFlag & 0x70) != 0 )
{
  while ( 1 )
    ;
}
```

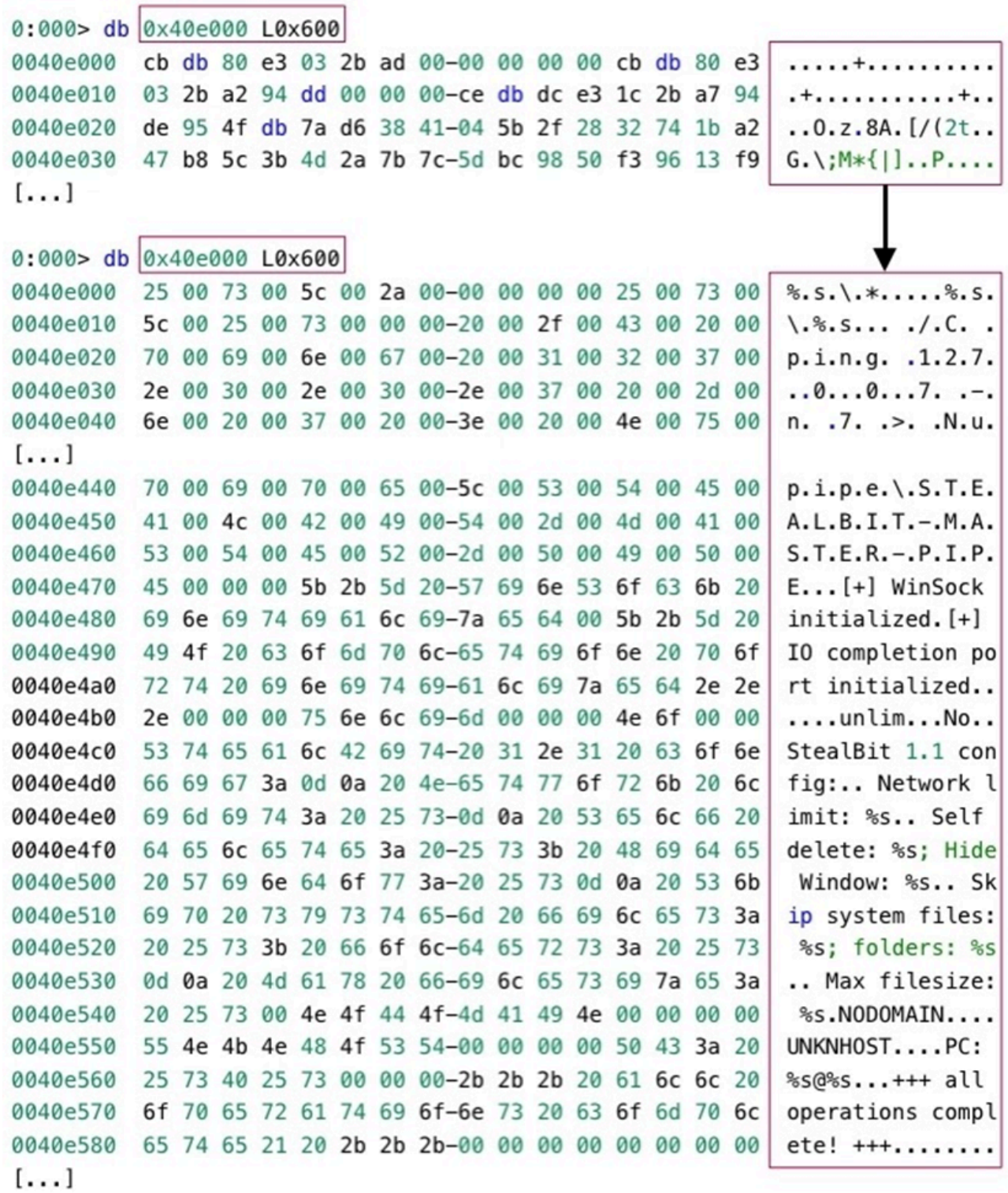
StealBit detects the presence of a debugger

StealBit then de-obfuscates the filenames of the dynamic-link libraries (DLLs) a*dvapi32*, *gdi32*, *gdiplus*, *shell32*, *ntdll*, *ole32*, *user32*, *shlwapi*, *kernel32*, and *ws2_32* and loads the libraries by executing the *LoadLibraryExA* function. StealBit stores the XOR obfuscated filenames of these DLLs in the malware's executable file:

```
0:000> bp SB_6c9a+0x6767 "da poi(@esp); g;"
breakpoint 1 redefined
0:000> g
0019fe10  "advapi32.dll"
0019fe10  "gdi32.dll"
0019fe10  "gdiplus.dll"
0019fe10  "shell32.dll"
0019fe10  "ntdll.dll"
0019fe10  "ole32.dll"
0019fe10  "user32.dll"
0019fe10  "shlwapi.dll"
0019fe10  "kernel32.dll"
0019fe10  "ws2_32.dll"
```

StealBit loads DLLs

StealBit then decrypts RC4-encrypted strings that the malware stores in the malware's executable file. StealBit uses these strings for different purposes throughout the malware's operation. For example, one string specifies a Windows command that StealBit executes, another string specifies the path to a named pipe file that StealBit creates, and StealBit displays some of the strings to the malware operator. We discuss these aspects of the StealBit operation in greater detail later in this section:



StealBit decrypts RC4-encrypted strings

StealBit then configures the process to not display certain Windows error messages by invoking the *NtSetInformationProcess* function and parses the command line parameters that the StealBit operator may have specified. The table below lists the command line parameters that StealBit supports. We discuss the exact impact of these command line parameters on the execution of StealBit in greater detail later in this section:

Command line parameter	**Description**	**Required / optional**	**Default value**
<path to file or folder>	This parameter specifies the filesystem path to the file or the folder whose content StealBit is to exfiltrate. Setting this parameter configures StealBit to read and exfiltrate the content of the file, or the content of the files placed in the folder.	Required	none
-hide/-h yes/y | no/n	This parameter controls the visibility of the graphical user interface of StealBit—that is, this parameter hides (*yes/y*) or displays (*no/n*) windows that StealBit creates.	Optional	*no/n*: StealBit displays windows
-delete/-d yes/y | no/n	This parameter configures StealBit to self-delete (*yes/y*)—that is, to delete the executable file that implements StealBit from the filesystem of the compromised system when StealBit is finished executing—or not to self-delete (*no/n*).	Optional	*no/n*: StealBit does not self-delete
-net/-n <transfer rate> *-once/-o <transfer rate>*	This parameter configures StealBit to exfiltrate file content at the specified rate, where rate is an amount of exfiltrated file content in KBs, MBs, or GBs, over 15 seconds.	Optional	*unlim*: there is no file content exfiltration rate
-skipfiles yes/y | no/n	This parameter configures StealBit to not exfiltrate the content of files with specific filename extensions (*no/n*).	Optional	*yes/y*: StealBit does not consider the filename extensions of files as a criterion for file content exfiltration

-skipfolders yes/y | no/n	This parameter configures StealBit to not exfiltrate the content of files that are placed in specific folders (*no/n*).	Optional	*yes/y*: StealBit does not consider folders as a criterion for file content exfiltration
-file/-f <file size>	This parameter configures StealBit to exfiltrate the content of only those files of a size equal to, or less than the specified file size in KBs, MBs, or GBs.	Optional	*unlim*: there is no maximum file size for file content exfiltration
Examples			
stealbit.exe C:\Users\user\Desktop\file.db -hide y -skipfiles n *stealbit.exe C:\Users\user\Desktop\ -net 5MB -delete y -h y -skipfolders n -file 2GB*			

The command line parameters that StealBit supports

After parsing command line parameters, StealBit creates or opens the named pipe file *\??\pipe\STEALBIT-MASTER-PIPE*. The path to the named pipe file is one of the strings that StealBit has previously decrypted using the RC4 algorithm.

If the current StealBit instance is the first one that the malware's operator has executed on the compromised system, StealBit creates the named pipe file *STEALBIT-MASTER-PIPE* by invoking the *NtCreateNamedPipeFile* function and assumes the role of a named pipe server.

We refer to this StealBit instance as a StealBit named pipe server. If not, StealBit opens the named pipe file *STEALBIT-MASTER-PIPE* by invoking the *NtCreateFile* function and assumes the role of a named pipe client. We refer to this StealBit instance as a StealBit named pipe client.

In summary, StealBit implements named pipe-based IPC between multiple StealBit processes that run on a single compromised system. We show later in this section that this enables StealBit operators to designate many files for exfiltration in a scalable manner by executing StealBit named pipe clients with the *<path to file or folder>* command line parameter set to the paths to the files. This makes the overall process for exfiltrating the content of multiple files convenient and efficient for StealBit operators:

```
Breakpoint 1 hit
[..]
SB_6c9a+0x5138:
00405138 ff153cc94000    call    dword ptr [SB_6c9a+0xc93c (0040c93c)]
ds:002b:0040c93c={ntdll!NtCreateNamedPipeFile (77902030)}
[...]
0:000> !obja 0019fef0
Obja +00000000 at 0019fef0:
    Name is \??\pipe\STEALBIT-MASTER-PIPE
    OBJ_CASE_INSENSITIVE
```

```
Breakpoint 1 hit
[...]
SB_6c9a+0x5178:
00405178 ff1530c94000    call    dword ptr [SB_6c9a+0xc930 (0040c930)]
ds:002b:0040c930={ntdll!NtCreateFile (77901a90)}
[...]
0:000> !obja 0x19fef0
Obja +00000000 at 0019fef0:
    Name is \??\pipe\STEALBIT-MASTER-PIPE
    OBJ_CASE_INSENSITIVE
```

StealBit creates or opens the named pipe file STEALBIT-MASTER-PIPE

At this point in the execution flow of StealBit, the execution of a StealBit instance that assumes the role of a named pipe server diverges from the execution of a StealBit instance that assumes the role of a named pipe client. The StealBit Named Pipe Server section discusses the former and the StealBit Named Pipe Client section discusses the latter.

StealBit Named Pipe Server

After creating the named pipe file *STEALBIT-MASTER-PIPE*, the StealBit named pipe server creates and starts two threads: one that creates two windows, and one that shows a message about exfiltration progress.

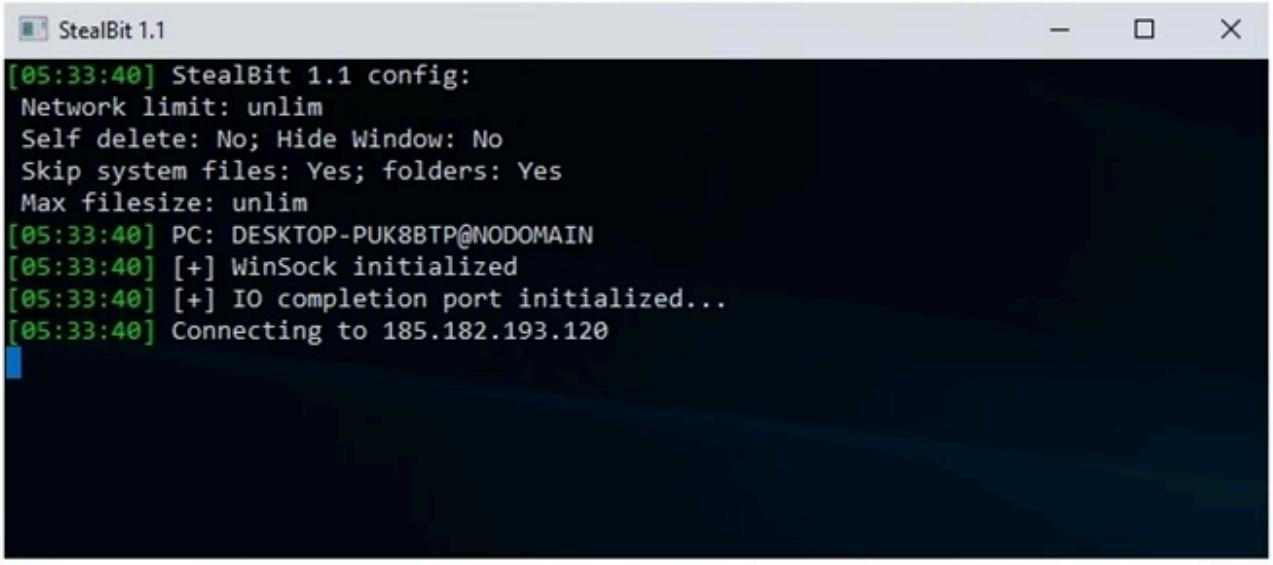
The first thread creates two windows by invoking the *CreateWindowExW* function. The first window is a top-level, parent window, with a title of *StealBit 1.1*. The second window is a child window of the top-level window and is therefore confined to the area of the parent window. The child window can display formatted text, and this window displays the output of StealBit to the malware operator.

We emphasize that setting the *-hide/-h* command line parameter to *yes/y* hides only the child window, while the parent window is still visible. This indicates that the implementation of the window hiding feature of StealBit—that is, of the *-hide/-h* command line parameter—is not complete, because it does not make StealBit invisible in

the Windows graphical user interface by hiding all windows that StealBit creates. This contradicts the claim of the LockBit group that StealBit hides its presence on compromised systems:

Downloading method	Speed in megabytes per second	Compression in real time	Hidden mode
Stealer - StealBIT	83,46 MB/s	Yes	Yes
Rclone pcloud.com free	4,82 MB/s	No	No
Rclone pcloud.com premium	4,38 MB/s	No	No

LockBit claims that StealBit hides its presence on compromised systems (source: KELA, Twitter)



StealBit displays windows when the malware operator sets the command line parameter -hide/-h to no/n (upper image) and yes/y (lower image)

The parent StealBit window supports dragging and dropping files or folders and the *F2* and *Shift+F2* hotkeys. Pressing the *F2* key closes the parent and child window without terminating execution, which effectively makes StealBit invisible in the Windows graphical user interface.

Pressing the key combination *Shift+F2* has no effect. Dragging and dropping a file or folder into the parent StealBit window is equivalent to specifying the *<path to file or folder>* command line parameter. The drag and drop activity causes StealBit to read and exfiltrate the content of the dropped file, or the content of the files placed in the dropped folder, in a way that we discuss later in this section.

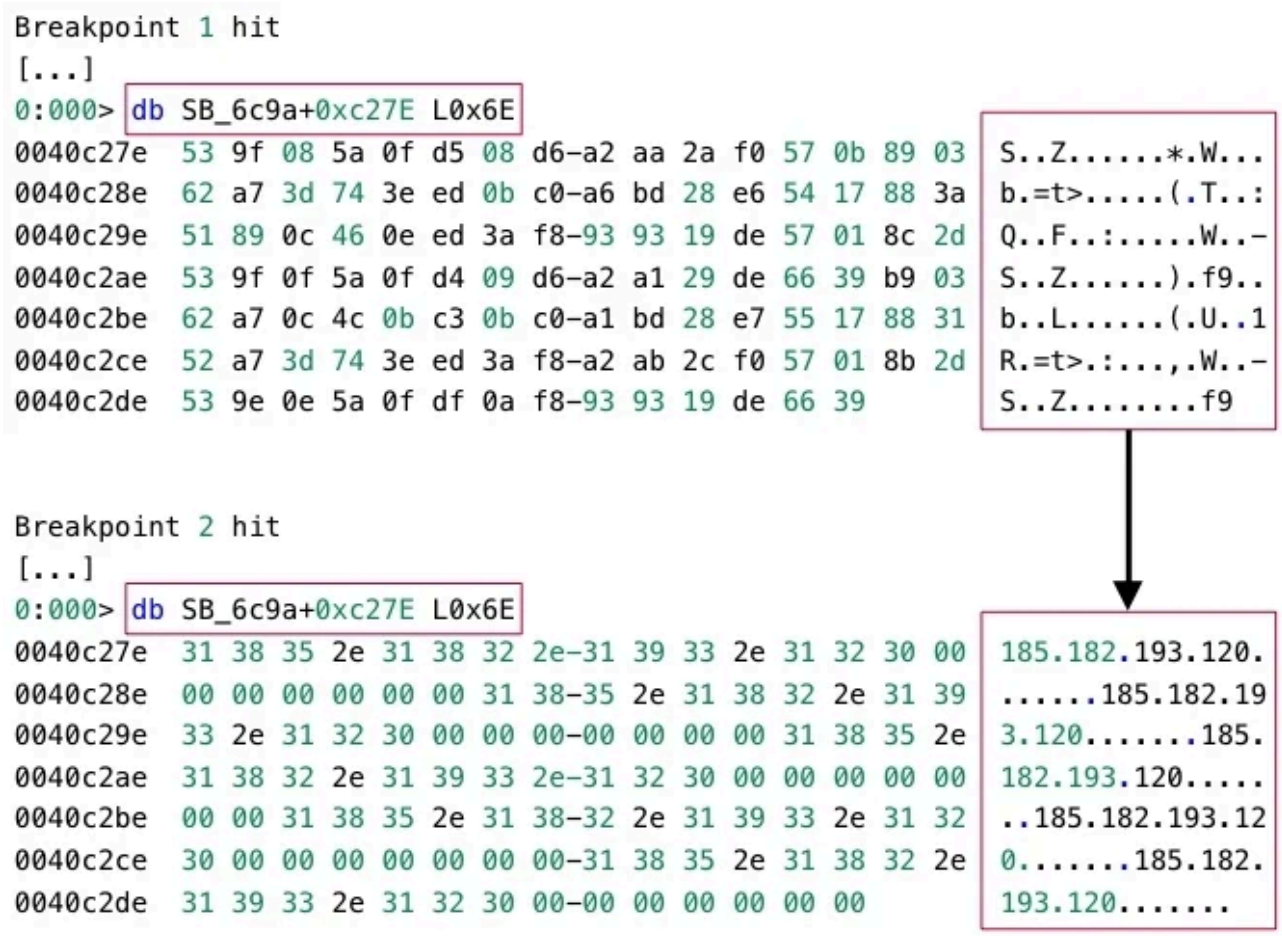
The drag and drop feature enables malicious actors to conveniently provide many file or folder paths to StealBit for file content exfiltration in scenarios where the StealBit operators have access to the graphical user interface of compromised systems, such as through an Remote Desktop Protocol (RDP) session. This makes the overall process for exfiltrating the content of many files practically convenient and scalable for StealBit operators.

The second thread is active during the overall operation of StealBit and displays a message in the StealBit window that informs the operator about the progress of file content exfiltration when exfiltration takes place. In the form of a format string, the message is: *Stats: %I64d files (size %S), read speed %S/sec (compression ratio %I64d%%), upload %S/sec*. This format string is one of the strings that StealBit has previously decrypted using the RC4 algorithm.

After creating and starting the two threads, StealBit displays the values of the configuration settings that StealBit operators can configure by setting the values of the StealBit command line parameters. In addition, StealBit displays the computer name of the compromised system and the name of the domain to which the system belongs (if any; see the figure above).

StealBit then initializes the Windows Socket networking library, which StealBit uses for communication with the attacker-controlled endpoints to which StealBit may exfiltrate file content. StealBit de-obfuscates five IP addresses of these endpoints, which the malware stores in XOR obfuscated form in the StealBit executable file. StealBit also stores a string that uniquely identifies the set of the endpoint IP addresses across StealBit samples, such as *DI0AN*. We refer to this string as the *StealBit configuration ID*:

```
for ( i = 0; i < 0x7C; ++i )
    *((_BYTE *)&word_40C270 + i) ^= byte_40C260[i & 0xF];
```



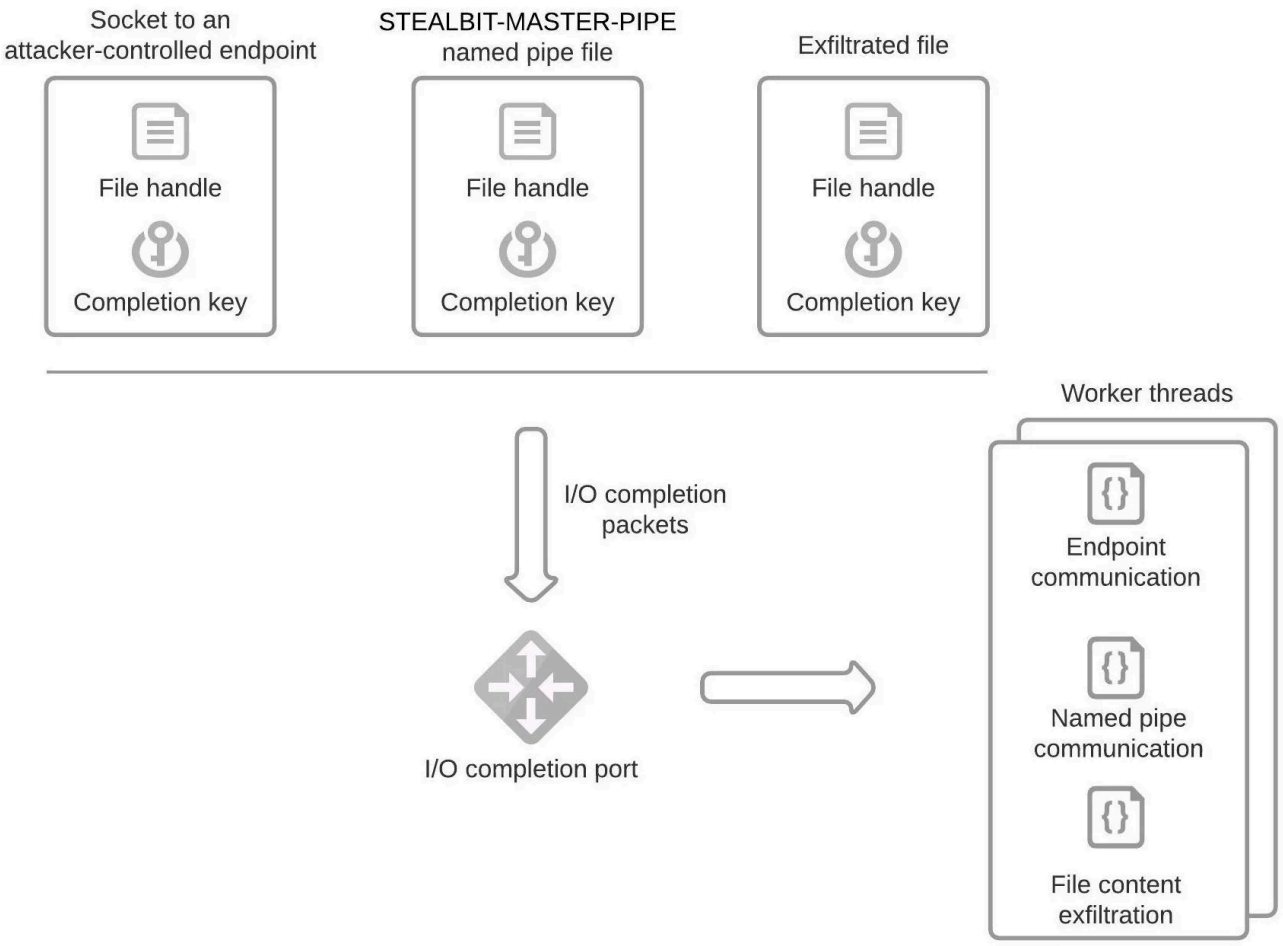
StealBit de-obfuscates IP addresses of attacker-controlled endpoints to which StealBit may exfiltrate file content

StealBit Malware Threading: I/O Completion Port

After initializing the Windows Socket library, StealBit establishes its core functionality: the Microsoft I/O completion port threading model for processing multiple asynchronous I/O requests in parallel. StealBit implements the I/O completion port threading model to maximize the overall efficiency of file content exfiltration activities on compromised systems. For example, as we show later in this section, StealBit parallelizes the exfiltration of the content of multiple files to shorten the overall exfiltration timespan. This is important to ransomware operators, since fast data exfiltration reduces the chances of being discovered in the process.

The I/O completion port threading model works by creating an I/O completion port and associating one or more file handles with that port. When an asynchronous I/O operation on one of these file handles completes, the Windows operating system queues to the port an I/O completion packet:

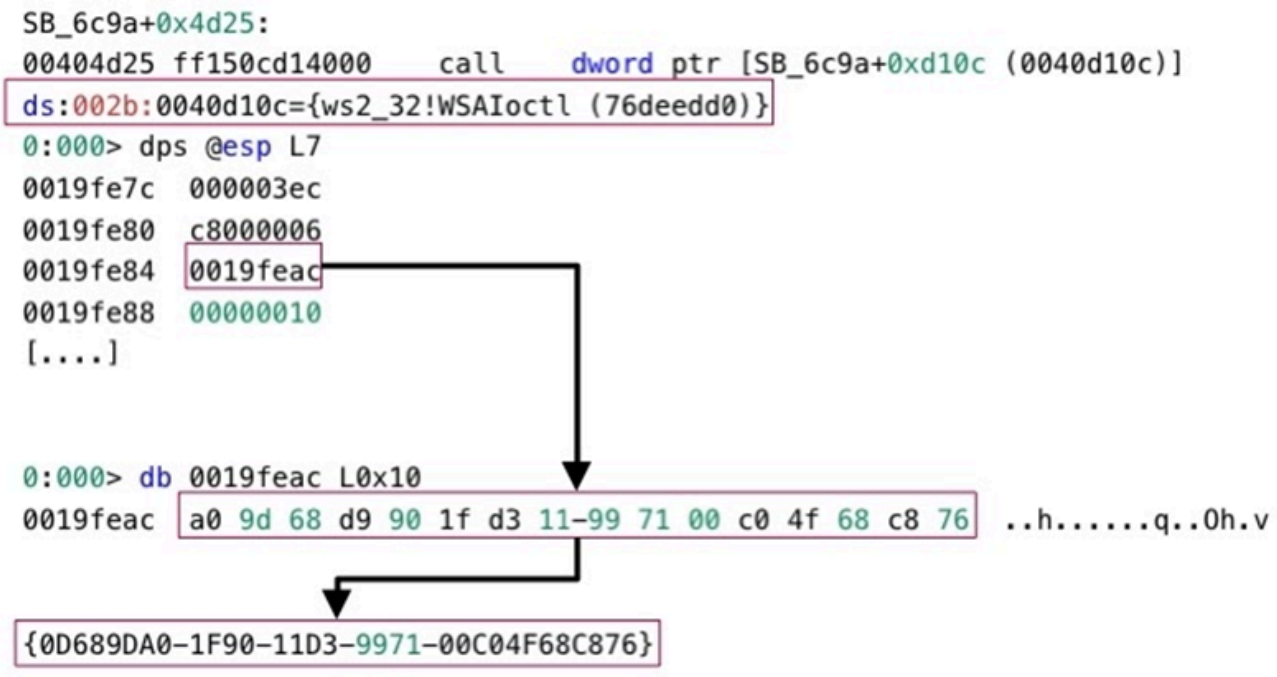
I/O completion packets carry information about the I/O operation. The application can then process I/O completion packets by removing them from the queue in a first-in-first-out (FIFO) order. In addition to a file handle, an application may associate a handle-specific I/O completion key with an I/O completion port. I/O completion keys can carry arbitrary data, which is typically data related to the handle. The figure below depicts the I/O completion port threading model that StealBit implements:



StealBit implements the I/O completion port threading model

StealBit creates an I/O completion port by invoking the *ZwCreateIoCompletion* function. StealBit also creates threads for processing I/O completion packets that Windows queues to the port, which we refer to as StealBit *worker threads*. StealBit creates as many worker threads as processors are available on the compromised system. StealBit then associates three file handles (and I/O completion keys) with the I/O completion port by invoking the *ZwSetInformationFile* function:

- - A handle to the socket to an attacker-controlled endpoint to which StealBit exfiltrates file content: This assigns available worker threads to handle the communication with the attacker-controlled endpoint. StealBit attempts to connect to each of the five IP addresses that the malware has de-obfuscated. If StealBit cannot establish a connection to any of these IP addresses, the malware indefinitely attempts to establish a connection. If the connection to one of these IP addresses succeeds, StealBit opens a socket to the attacker-controlled endpoint and associates the socket handle and an I/O completion key with the I/O completion port. In addition, to make static analysis difficult, StealBit obtains an address to the *TransmitPackets* function at runtime by invoking the *WSAIoctl* function. The *TransmitPackets* function is crucial to StealBit, since the malware uses this function to exfiltrate file content. *WSAIoctl* returns an address to *TransmitPackets* if an application provides the globally unique identifier (GUID) of the *TransmitPacket* function, *{0D689DA0-1F90-11D3-9971-00C04F68C876}*, as a parameter to *WSAIoctl*:



StealBit obtains an address to the TransmitPackets function at runtime

- A handle to the named pipe *STEALBIT-MASTER-PIPE*: This assigns available worker threads to handle the communication with StealBit named pipe clients. The section StealBit Named Pipe Client discusses the activities that the worker threads conduct when StealBit named pipe clients send data to the StealBit named pipe server.
- A handle to a file whose content StealBit is to exfiltrate: This assigns available worker threads to handle file content exfiltration upon successful file read operations on the file. This parallelizes file content exfiltration and shortens the overall timespan of file content exfiltration activities. In addition to exfiltrating read file content, it is the StealBit named pipe server, and not the StealBit named pipe client, that reads file content for exfiltration purposes.

StealBit Malware File Content Exfiltration

The StealBit named pipe server reads and exfiltrates the content of the file or the folder, whose file system path is either provided by a StealBit named pipe client or specified as the value of the *<path to file or folder>* command line parameter by the StealBit operator. Section StealBit Named Pipe Client discusses the communication between the StealBit named pipe server and client in more detail.

If the StealBit operator has specified a file path as the value of the *<path to file or folder>* command line parameter, the StealBit named pipe server first evaluates whether the path leads to a file or a folder. If the path leads to a file, StealBit reads the content of the file only if the file meets one or more of these requirements:

- The length of the name of the file is less than, or equal to, four characters.
- The filename extension of the file is not present in a list of filename extensions, which StealBit stores in hashed format in the malware's executable file. StealBit enforces this criterion only if the StealBit operator has set the *-skipfiles* command line parameter to *no/n.*

In addition, the size of the file has to be less than or equal to 0.53 GB. The command line parameter *-file/-f* does not have an impact on the execution of the StealBit sample that we analyzed. This indicates that the implementation of the -file/-f command line parameter is not complete.

If the path leads to a folder, StealBit iterates the folder recursively to enumerate files placed in the folder and sub-folders. If the StealBit operator has set the *-skipfolders* command line parameter to *no/n*, StealBit enumerates files only from those folders that are not present in a list of folders, which StealBit stores in hashed format in the malware's executable file. After enumerating the files in a folder, StealBit reads the content of each file, except the content of system files (*FILE_ATTRIBUTE_SYSTEM*), if the above conditions are fulfilled.

Before reading content from a file, StealBit opens the file and then associates the handle to the file and an I/O completion key with the I/O completion port that StealBit has created. StealBit invokes the *ZwReadFile* function to read the content of the file in equal-sized blocks. StealBit calculates the block size as a function of the total file size—the bigger the file, the bigger the block size.

Each successful file content read operation issues an I/O completion packet to the I/O completion port. The available worker threads process this packet and exfiltrate the file content to an attacker-controlled endpoint using the *TransmitPackets* function, whose address StealBit has previously obtained.

To evade exfiltration detection mechanisms that monitor the amount of sent data to remote endpoints over time, StealBit operators can configure StealBit to exfiltrate file content at a given rate (amount of exfiltrated file content over 15 seconds) by configuring the *-net/-n* or *-once/-o* command line parameters. These parameters control the file content exfiltration rate by controlling the rate at which StealBit reads file content.

As we mentioned earlier, the file read activity issues I/O completion packets to the StealBit I/O completion port and instructs available worker threads to exfiltrate the read content. StealBit controls the file content reading rate by delaying invocations of the *ZwReadFile* function for continuously adjusted time periods, such that the total amount of read file content over 15 seconds does not exceed the exfiltration rate that the StealBit operator has specified.

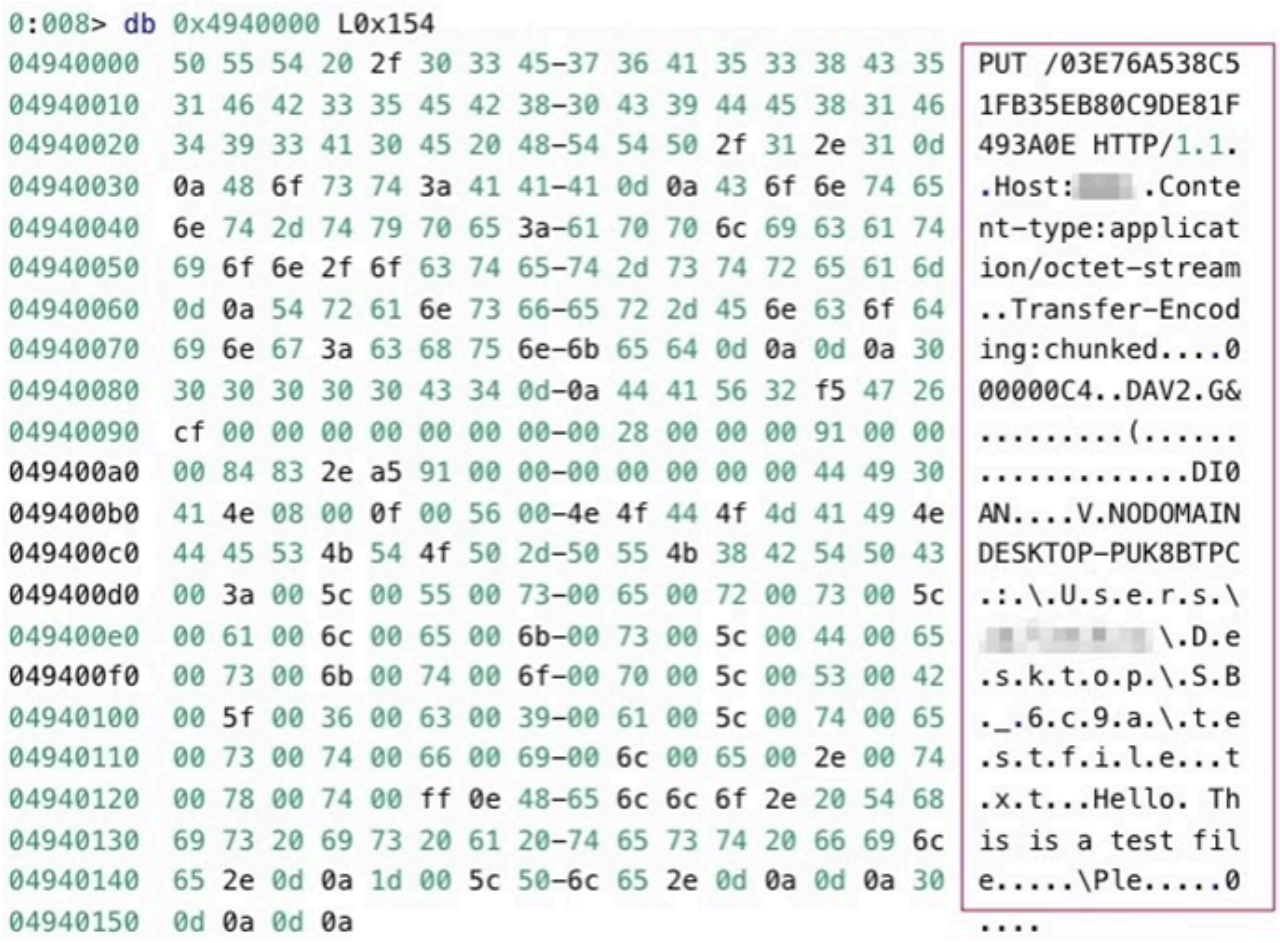
Every time StealBit reads file content using the *ZwReadFile* function, available StealBit worker threads exfiltrate the read file content by issuing the Hypertext Transfer Protocol 1.1 (HTTP 1.1) *PUT* request to an attacker-controlled endpoint. StealBit stores exfiltrated file content on the attacker-controlled endpoint as a resource that has a random name, which StealBit generates for each file whose content the malware exfiltrates (for example, *03E76A538…* in the figure below). The data that StealBit sends to the attacker-controlled endpoint includes:

- A Distributed Authoring and Versioning 2 (DAV2*)* header (*DAV2...* in the figure below)
- The StealBit *configuration ID* (for example, *DI0AN* in the figure below)
- The computer name of the compromised system and the name of the domain (if any) to which the system belongs (for example, *NODOMAIN* and *DESKTOP-PUK8BTP* in the figure below)
- The absolute path to the file whose content StealBit exfiltrates (for example, *C:\Users\<user>\Desktop\SB_6c9a\testfile.txt* in the figure below)
- The file content that StealBit exfiltrates (for example, *Hello. This is a test file.* in the figure below).

The file content is not compressed. This contradicts the claim of the LockBit threat group that StealBit compresses exfiltrated file content:

Downloading method	Speed in megabytes per second	Compression in real time
Stealer - StealBIT	83,46 MB/s	Yes
Rclone pcloud.com free	4,82 MB/s	No
Rclone pcloud.com premium	4,38 MB/s	No

LockBit claims that StealBit compresses exfiltrated file content (source: KELA, Twitter)



StealBit exfiltrates uncompressed file content

The StealBit sample that we analyzed does not execute indefinitely in order to keep the StealBit worker threads that handle I/O completion packets active in a typical server fashion. To the contrary, after creating worker threads and establishing the I/O completion port threading model, StealBit processes the *<path to file or folder>* command line parameter and exfiltrates file content if the StealBit operator has specified a valid parameter value.

StealBit then waits until the worker threads have processed all I/O completion packets, and then closes the named pipe file *STEALBIT-MASTER-PIPE*. Next, depending on the value of the *-delete/-d* command line parameter, StealBit empties the content of its executable file and deletes the file. StealBit conducts these activities by invoking the *ShellExecuteExW* function to execute these commands, where *<file size>* is the size of the StealBit executable file in bytes and *<file path>* is the path to the StealBit executable file:

- *ping 127.0.0.7 -n 7 > Nul*
- *fsutil file setZeroData offset=0 length=<file size> <file path>*
- *del /f /q <file path>*

Finally, StealBit terminates its execution:

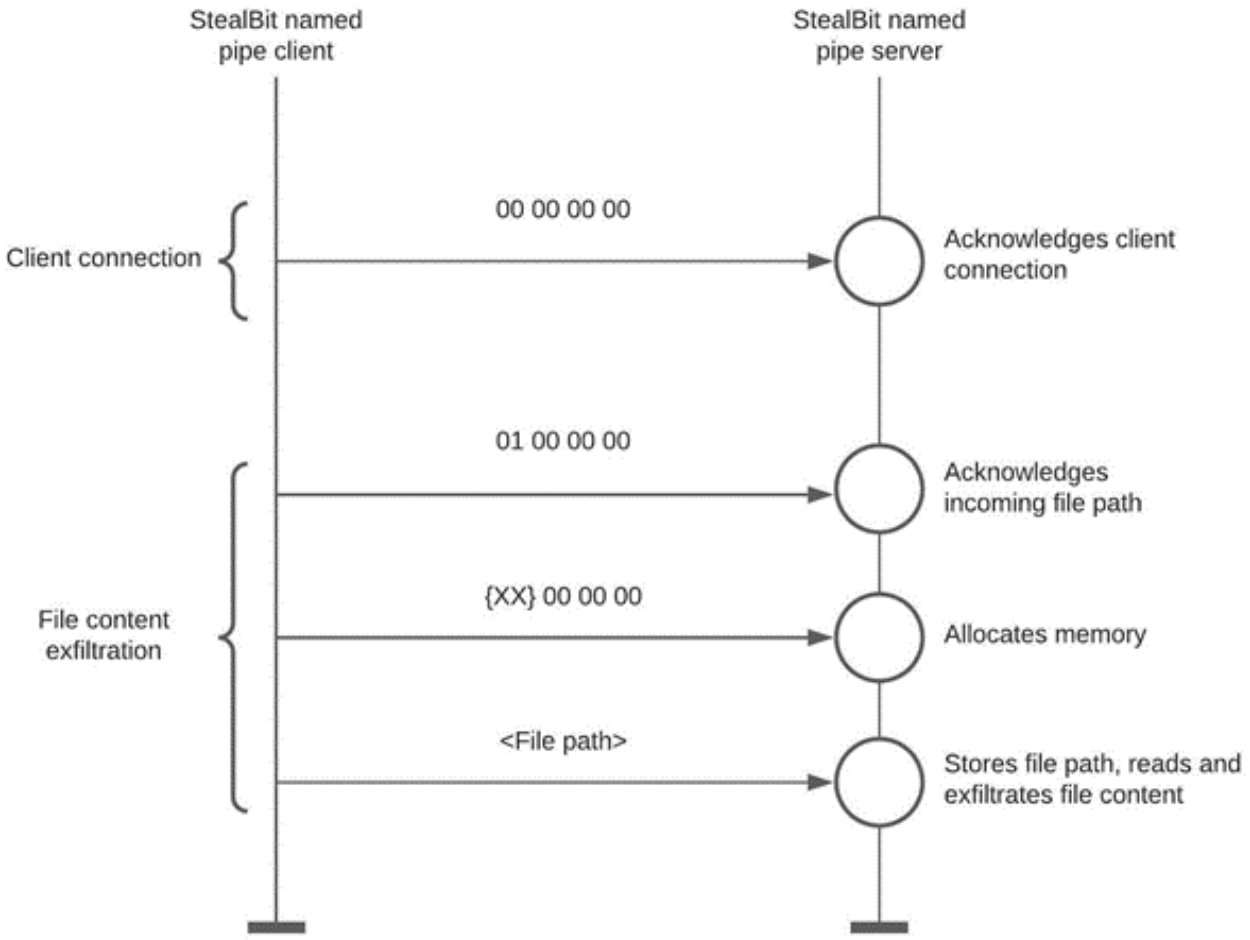
```
0:000> du 0x30000
00030000  " /C ping 127.0.0.7 -n 7 > Nul & "
00030040  "fsutil file setZeroData offset=0"
00030080  " length=1048576 "C:\Users\[redacted]\"
000300c0  "Desktop\SB_6c9a\SB_6c9a" & Del /"
00030100  "f /q "C:\Users\[redacted]\Desktop\SB_"
00030140  "6c9a\SB_6c9a""
[...]
SB_6c9a+0x3759:
00403759 ff15a4cc4000    call    dword ptr [SB_6c9a+0xcca4 (0040cca4)]
ds:002b:0040cca4={shell32!ShellExecuteExW (75016240)}
```

StealBit deletes its executable file

StealBit Malware Named Pipe Client

After opening the named pipe file *STEALBIT-MASTER-PIPE*, the StealBit named pipe client delegates file content reading and exfiltration to the StealBit named pipe server. To do this, the StealBit named pipe client communicates with the StealBit named pipe server by following a communication protocol.

The figure below depicts this protocol. When a StealBit named pipe client sends data to a StealBit named pipe server, this action issues an I/O completion packet to the I/O completion port that the StealBit named pipe server creates (see section StealBit Named Pipe Server). The worker threads of the StealBit named pipe server then process this packet. The StealBit named pipe server uses the worker threads that handle the communication with StealBit named pipe clients to conduct the server's activities that are depicted in the figure below:



The StealBit named pipe client communicates with the StealBit named pipe server

After opening *STEALBIT-MASTER-PIPE* and therefore connecting to the StealBit named pipe server, the StealBit named pipe client sends the four bytes *00 00 00 00* to the server to announce the client's presence. The StealBit named pipe server keeps track of the state of the connection. When the StealBit named pipe client announces itself, the server acknowledges the client's presence by updating the state of the connection to indicate successful client connection.

The StealBit named pipe client then processes the value of the *<path to file or folder>* command line parameter in the same manner as the StealBit named pipe server (see section StealBit Named Pipe Server). However, in contrast to the StealBit named pipe server, the StealBit named pipe client does not read and exfiltrate file content, but delegates this task to the server as follows:

- - The client sends the four bytes *01 00 00 00* to the server to indicate that the client is about to send a file path to the server. This file path is the path to the file whose content the server is to read and exfiltrate. The StealBit named pipe server acknowledges the communication by updating the state of the connection to indicate the incoming file path.
  - The client sends four bytes to the server such that the bytes specify the length of the file path in a null-terminated Unicode string format. For example, the client sends the bytes *3E 00 00 00* to the server when the client is about to send the file path *C:\Users\user\Desktop\file.txt* to the server (*0x3E* in hexadecimal format is *62* in decimal format). The StealBit named pipe server updates the

state of the connection and allocates a virtual memory region of a size that is the same as the file path length that the client has sent.
- The client sends the file path to the server. The StealBit named pipe server updates the state of the connection, stores the file path in the previously allocated memory region, and then reads and exfiltrates the content of the file at the file path (see section StealBit Named Pipe Server).
- Delegating file content reading and exfiltration to the StealBit named pipe server enables malicious actors to designate many files for exfiltration in a scalable manner by executing StealBit named pipe clients with the *<path to file or folder>* command line parameter set to the paths to the files.

The StealBit named pipe client then closes the connection to the server and, depending on the value of the *-delete/-d* command line parameter, deletes its executable file in the same manner as the StealBit named pipe server. The StealBit named pipe client then terminates its execution. The command line parameters *-hide/-h*, *-net/-n*, and *-once/-o* do not have an impact on the execution of the StealBit named pipe client.

StealBit Malware Comparative Analysis

The table below lists selected StealBit samples that represent StealBit samples that the security community has observed at the time of writing of this report, in terms of the configuration and implementation aspects of StealBit that are in the scope of this report. For referencing purposes, each sample has a codename with the prefix *SB_* and a suffix that is the first four hexadecimal numbers of the sample's SHA-256 hash:

SB_3407	
SHA-256 Hash	3407f26b3d69f1dfce76782fee1256274cf92f744c65aa1ff2d3eaaaf61b0b1d
First submission to VirusTotal	2021-08-06
SB_107d	
SHA-256 Hash	107d9fce05ff8296d0417a5a830d180cd46aa120ced8360df3ebfd15cb550636
First submission to VirusTotal	2021-09-09
SB_6c9a	

SHA-256 Hash	6c9a92955402c76ab380aa6927ad96515982a47c05d54f21d67603814d29e4a5
First submission to VirusTotal	2021-11-08
SB_6b9a	
SHA-256 Hash	6b9aa479a5f9c6bfee52046c1afa579977dfcde868fdad3f18fdcd1779535068
First submission to VirusTotal	2021-11-26

Representative StealBit samples

The tables below compare the selected StealBit samples (column ‘Sample’) considering the following configuration and implementation aspects:

- IP addresses and geolocations of attacker-controlled endpoints to which StealBit exfiltrates data (column ‘IP addresses’ and ‘Location’).
- The debugger detection method that StealBit implements as an anti-analysis measure (column ‘Debugger detection’).
- Command line parameters and the respective malware features (column ‘Command line parameters’).
- A named pipe IPC infrastructure that makes exfiltrating the content of multiple files practically convenient and efficient for StealBit operators (column ‘IPC’).
- The I/O completion port threading model to maximize the overall efficiency of data exfiltration activities (column ‘I/O completion’).
- Conditions for execution and file content exfiltration (column ‘Execution conditions’):

Sample	**IP addresses**	**Location**
SB_3407	88.80.147[.]102	Bulgaria
	168.100.11[.]72	The Netherlands
	139.60.160[.]200	United States
	193.38.235[.]234	Russia

Sample	IP addresses	Location
	174.138.62[.]35	United States
SB_107d	93.190.139[.]223 168.100.11[.]72 139.60.160[.]200 193.38.235[.]234 174.138.62[.]35	The Netherlands The Netherlands United States Russia United States
SB_6c9a	185.182.193[.]120	The Netherlands
SB_6b9a	185.182.193[.]120	The Netherlands

Comparison of StealBit samples: Attacker-controlled endpoints

Sample	Debugger detection	Command line parameters	IPC	I/O completion	Execution conditions
SB_3407	*NtGlobalFlag*	*<path to file or folder>*	Yes	Yes	Location
SB_107d		*<path to file or folder>*			Location
SB_6c9a		*<path to file or folder>, -hide/-h, -delete/d, -net/-n, -once/-o, -skipfiles, -skipfolders, -file/-f*			None
SB_6b9a		*<path to file or folder>, -hide/-h, -delete/-d, -net/-n, -once/-o, -skipfiles, -skipfolders, -file/-f*			None

Comparison of StealBit samples

The majority of the attacker-controlled endpoints to which the StealBit samples that we analyzed exfiltrate data are located in western countries, with the Netherlands and the United States at the top of the list. All StealBit samples implement named pipe-based IPC and the I/O completion port threading model for maximum exfiltration efficiency, usage convenience, and scalability. In addition, all StealBit samples detect the presence of a debugger attached to the StealBit process by evaluating the value of the *NtGlobalFlag* field of the PEB and execute an empty infinite loop if a debugger is present.

Older Versus Newer Versions of StealBit Malware

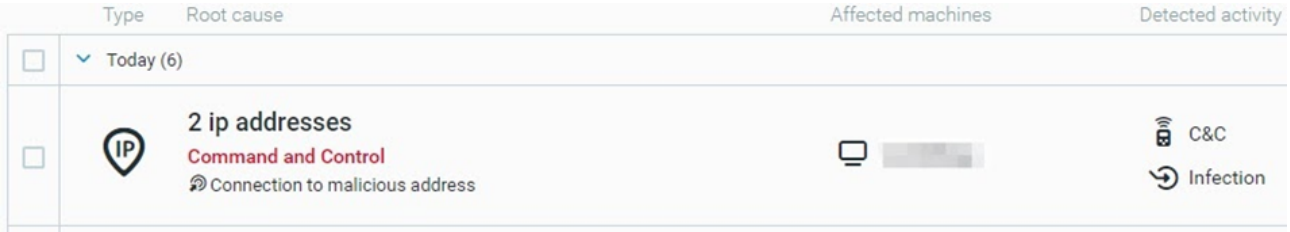
A major difference between the StealBit samples that we analyzed is the command line parameters and the respective malware features that the samples support. Relatively older StealBit samples do not support the command line parameters *-hide/-h*, *-delete/-d*, *-net/-n*, *-once/-o*, *-skipfiles*, *-skipfolders*, and *-file/-f* and the features that these parameters configure, such as self-deletion and data exfiltration rate.

This indicates that StealBit has been undergoing improvement with new features, especially evasion and hiding features. Another major difference is that relatively older samples do not execute on systems located in the former Soviet countries of Russia, Ukraine, Belarus, Tajikistan, Armenia, Azerbaijan, Georgia, Kazakhstan, Kyrgyzstan, Turkmenistan, Uzbekistan, and Moldova. StealBit determines the location of a compromised system based on the system's default language. Relatively newer samples do not implement this restriction and execute on any system.

Detection and Prevention of StealBit Malware

Cybereason XDR Platform

The Cybereason XDR Platform detects and stops StealBit when the malware exfiltrates data, using multi-layer protection that employs threat intelligence, machine learning, and next-gen antivirus (NGAV) capabilities to detect and block malware. The Cybereason platform also detects malicious actors that execute the related LockBit ransomware:



The Cybereason XDR Platform detects StealBit based on threat intelligence

Cybereason GSOC MDR

Cybereason GSOC recommends the following:

- Enable the **Anti-Malware** feature on the Cybereason NGAV, and enable the **Detect** and **Prevent** modes of this feature.

- Regularly monitor outgoing network traffic for data exfiltration activities.
- Threat Hunting with Cybereason: The Cybereason MDR team provides its customers with custom hunting queries for detecting specific threats - to find out more about threat hunting and Managed Detection and Response with the Cybereason Defense Platform, contact a Cybereason Defender here.
    - For Cybereason customers: More details available on the NEST including custom threat hunting queries for detecting this threat.

Cybereason is dedicated to teaming with Defenders to end cyber attacks from endpoints to the enterprise to everywhere. Learn more about *Cybereason XDR powered by Google Chronicle*, check out our Extended Detection and Response (XDR) Toolkit, or schedule a demo today to learn how your organization can benefit from an operation-centric approach to security.

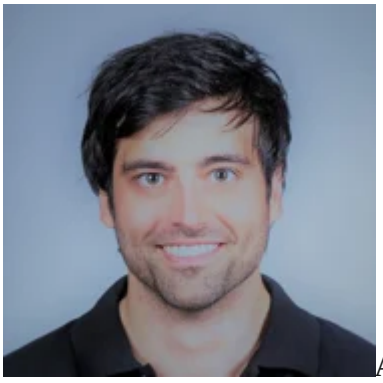
Indicators of Compromise for StealBit Malware

Executables	SHA-256 hash: *3407f26b3d69f1dfce76782fee1256274cf92f744c65aa1ff2d3eaaaf61b0b1d* SHA-256 hash: *107d9fce05ff8296d0417a5a830d180cd46aa120ced8360df3ebfd15cb550636* SHA-256 hash: *6c9a92955402c76ab380aa6927ad96515982a47c05d54f21d67603814d29e4a5* SHA-256 hash: *6b9aa479a5f9c6bfee52046c1afa579977dfcde868fdad3f18fdcd1779535068*
Named pipe files	*STEALBIT-MASTER-PIPE*
IP addresses	*88.80.147[.]102* *168.100.11[.]72* *139.60.160[.]200* *193.38.235[.]234* *174.138.62[.]35* *93.190.139[.]223* *185.182.193[.]120*

MITRE ATT&CK Techniques for StealBit Malware

Execution	Privilege Escalation	Defense Evasion	Discovery	Exfiltration
Native API	Abuse Elevation Control Mechanism: Bypass User Account Control	Indicator Removal on Host: File Deletion	File and Directory Discovery	Data Transfer Size Limits
Inter-Process Communication		Obfuscated Files or Information	System Information Discovery	Exfiltration Over C2 Channel
		Hide Artifacts: Hidden Window	System Location Discovery	

About the Researchers:



Aleksandar Milenkoski, Senior Malware and Threat Analyst, Cybereason Global SOC

Aleksandar Milenkoski is a Senior Malware and Threat Analyst with the Cybereason Global SOC team. He is involved primarily in reverse engineering and threat research activities. Aleksandar has a PhD in system security. For his research activities, he has been awarded by SPEC (Standard Performance Evaluation Corporation), the Bavarian Foundation for Science, and the University of Würzburg, Germany. Prior to Cybereason, his work focused on research in intrusion detection and reverse engineering security mechanisms of the Windows 10 operating system.

 **Kotaro Ogino, Security Analyst, Cybereason Global SOC**

Kotaro Ogino is a Security Analyst with the Cybereason Global SOC team. He is involved in threat hunting, administration of Security Orchestration, Automation, and Response (SOAR) systems, and Extended Detection and Response (XDR). Kotaro has a bachelor of science degree in information and computer science.

About the Author

Cybereason Global SOC Team

The Cybereason Global SOC Team delivers 24/7 Managed Detection and Response services to customers on every continent. Led by cybersecurity experts with experience working for government, the military and multiple

industry verticals, the Cybereason Global SOC Team continuously hunts for the most sophisticated and pervasive threats to support our mission to end cyberattacks on the endpoint, across the enterprise, and everywhere the battle moves.

All Posts by Cybereason Global SOC Team
