CloudSorcerer – A new APT targeting Russian government entities

By GReAT

Published: 2024-07-08 · Archived: 2026-04-05 13:05:44 UTC

In May 2024, we discovered a new advanced persistent threat (APT) targeting Russian government entities that we dubbed CloudSorcerer. It's a sophisticated cyberespionage tool used for stealth monitoring, data collection, and exfiltration via Microsoft Graph, Yandex Cloud, and Dropbox cloud infrastructure. The malware leverages cloud resources as its command and control (C2) servers, accessing them through APIs using authentication tokens. Additionally, CloudSorcerer uses GitHub as its initial C2 server.

CloudSorcerer's modus operandi is reminiscent of the CloudWizard APT that we reported on in 2023. However, the malware code is completely different. We presume that CloudSorcerer is a new actor that has adopted a similar method of interacting with public cloud services.

Our findings in a nutshell:

- CloudSorcerer APT uses public cloud services as its main C2s
- The malware interacts with the C2 using special commands and decodes them using a hardcoded charcode table.
- The actor uses Microsoft COM object interfaces to perform malicious operations.
- CloudSorcerer acts as separate modules (communication module, data collection module) depending on which process it's running, but executes from a single executable.

Technical details

Initial start up

MD5	f701fc79578a12513c369d4e36c57224
SHA1	f1a93d185d7cd060e63d16c50e51f4921dd43723
SHA256	e4b2d8890f0e7259ee29c7ac98a3e9a5ae71327aaac658f84072770cf8ef02de
Link time	N/A
Compiler	N/A
File type	Windows x64 executable
File size	172kb
File name	N/A

The malware is executed manually by the attacker on an already infected machine. It is initially a single Portable Executable (PE) binary written in C. Its functionality varies depending on the process in which it is executed. Upon execution, the malware calls the GetModuleFileNameA function to determine the name of the process it is running in. It then compares this process name with a set of hardcoded strings: browser, mspaint.exe, and msiexec.exe. Depending on the detected process name, the malware activates different functions:

- If the process name is mspaint.exe, CloudSorcerer functions as a backdoor module, and performs activities such as data collection and code execution.
- If the process name is msiexec.exe, the CloudSorcerer malware initiates its C2 communication module.
- Lastly, if the process name contains the string “browser” or does not match any of the specified names, the malware attempts to inject shellcode into either the msiexec.exe, mspaint.exe, or explorer.exe processes before terminating the initial process.

The shellcode used by CloudSorcerer for initial process migration shows fairly standard functionality:

- Parse Process Environment Block (PEB) to identify offsets to required Windows core DLLs;
- Identify required Windows APIs by hashes using ROR14 algorithm;
- Map CloudSorcerer code into the memory of one of the targeted processes and run it in a separate thread.

All data exchange between modules is organized through Windows pipes, a mechanism for inter-process communication (IPC) that allows data to be transferred between processes.

CloudSorcerer backdoor module

The backdoor module begins by collecting various system information about the victim machine, running in a separate thread. The malware collects:

- Computer name;
- User name;
- Windows subversion information;
- System uptime.

All the collected data is stored in a specially created structure. Once the information gathering is complete, the data is written to the named pipe \\.\PIPE\[1428] connected to the C2 module process. It is important to note that all data exchange is organized using well-defined structures with different purposes, such as backdoor command structures and information gathering structures.

Next, the malware attempts to read data from the pipe \\.\PIPE\[1428]. If successful, it parses the incoming data into the COMMAND structure and reads a single byte from it, which represents a COMMAND_ID.

```
{
  case 1:
    LastError = SystemDiskInfo(main_mod_struc, struct_Command);
    goto LABEL_52;
  case 2:
    LastError = GetFilesAndFolders(main_mod_struc, struct_Command);
    goto LABEL_52;
  case 3:
    LastError = ShellExecute__0((LPCRITICAL_SECTION)main_mod_struc, struct_Command);
    goto LABEL_52;
  case 4:
    LastError = SHFileOperationW___(main_mod_struc, struct_Command);
    goto LABEL_52;
  case 5:
    LastError = ReadsFiles_(main_mod_struc, struct_Command);
    goto LABEL_52;
  case 6:
    LastError = CreatesAndWritesFile(main_mod_struc, struct_Command);
    goto LABEL_52;
  case 7:
    LastError = RunsMainOperations(main_mod_struc, struct_Command);
    goto LABEL_52;
  case 8:
    LastError = InjectsRecievedCode_(main_mod_struc, struct_Command);
    goto LABEL_52;
  case 0xA:
    LastError = UnmapsSectionAndW_(main_mod_struc, struct_Command);
```

Main backdoor functionality

Depending on the COMMAND_ID, the malware executes one of the following actions:

- 0x1 – Collect information about hard drives in the system, including logical drive names, capacity, and free space.
- 0x2 – Collect information about files and folders, such as name, size, and type.
- 0x3 – Execute shell commands using the ShellExecuteExW API.
- 0x4 – Copy, move, rename, or delete files.
- 0x5 – Read data from any file.
- 0x6 – Create and write data to any file.
- 0x8 – Receive a shellcode from the pipe and inject it into any process by allocating memory and creating a new thread in a remote process.
- 0x9 – Receive a PE file, create a section and map it into the remote process.
- 0x7 – Run additional advanced functionality.

When the malware receives a 0x7 COMMAND_ID, it runs one of the additional tasks described below:

Command ID	**Operation**	**Description**
0x2307	Create process	Creates any process using COM interfaces, used for running downloaded binaries.

0x2407	Create process as dedicated user	Creates any process under dedicated username.
0x2507	Create process with pipe	Creates any process with support of inter-process communication to exchange data with the created process.
0x3007	Clear DNS cache	Clears the DNS cache.
0x2207	Delete task	Deletes any Windows task using COM object interfaces.
0x1E07	Open service	Opens a Windows service and reads its status.
0x1F07	Create new task	Creates a new Windows task and sets up a trigger for execution using COM objects.
0x2007	Get tasks	Gets the list of all the Windows tasks using COM object interface.
0x2107	Stop task	Stops any task using COM object interface.
0x1D07	Get services	Gets the list of all Windows services.
0x1907	Delete value from reg	Deletes any value from any Windows registry key selected by the actor.
0x1A07	Create service	Creates a new Windows service.
0x1B07	Change service	Modifies any Windows service configuration.
0x1807	Delete reg key	Deletes any Windows registry key.
0x1407	Get TCP/UDP update table	Gets information from Windows TCP/UDP update table.
0x1507	Collect processes	Collects all running processes.
0x1607	Set reg key value	Modifies any Windows registry key.
0x1707	Enumerate reg key	Enumerates Windows registry keys.
0x1307	Enumerate shares	Enumerates Windows net shares.
0x1007	Set net user info	Sets information about a user account on a Windows network using NetUserSetInfo. It allows administrators to modify user account properties on a local or remote machine.
0x1107	Get net members	Gets a member of the local network group.
0x1207	Add member	Adds a user to the local network group.
0xE07	Get net user info	Collects information about a network user.

0xB07	Enumerate net users	Enumerates network users.
0xC07	Add net user	Adds a new network user.
0xD07	Delete user	Deletes a network user.
0x907	Cancel connection	Cancels an existing network connection. This function allows for the disconnection of network resources, such as shared directories.
0x507	File operations	Copies, moves, or deletes any file.
0x607	Get net info	Collects information about the network and interfaces.
0x707	Enumerate connections	Enumerates all network connections.
0x807	Map network	Maps remote network drive.
0x407	Read file	Reads any file as text strings.
0x107	Enumerate RDP	Enumerates all RDP sessions.
0x207	Run WMI	Runs any WMI query using COM object interfaces.
0x307	Get files	Creates list of files and folders.

All the collected information or results of performed tasks are added to a specially created structure and sent to the C2 module process via a named pipe.

C2 module

The C2 module starts by creating a new Windows pipe named \\.\PIPE\[1428]. Next, it configures the connection to the initial C2 server by providing the necessary arguments to a sequence of Windows API functions responsible for internet connections:

- InternetCrackUrlA;
- InternetSetOptionA;
- InternetOpenA;
- InternetConnectA;
- HttpOpenRequestA;
- HttpSendRequestA

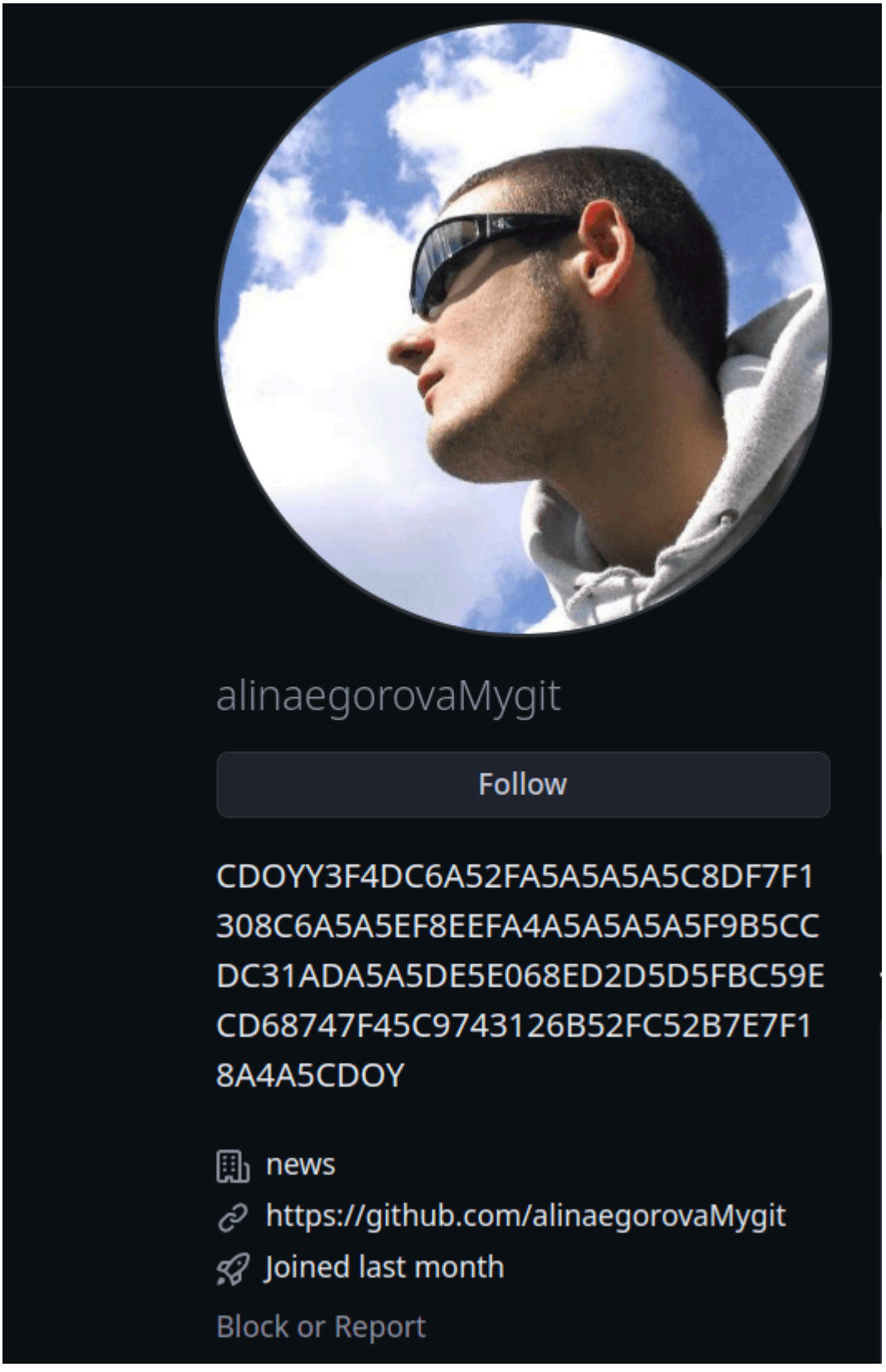
The malware sets the request type (“GET”), configures proxy information, sets up hardcoded headers, and provides the C2 URL.

```
InternetSetOptionA(0i64, 37u, 0i64, 0);   // refresh proxy data
                                          //
Buffer_1 = 0x84000100;
if ( nScheme == INTERNET_SCHEME_HTTPS )
  Buffer_1 = 0x84C10900;
Buffer = Buffer_1;
n_3_ = flag_2;
if ( flag_2 < 3 )
{
  // "Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko"
  while ( !aMozilla50Windo[260 * n_3_] )
  {
    flag_2 = ++n_3_;
    if ( n_3_ >= 3 )
      goto LABEL_14;
  }
  wsprintfA(&szAgent, "%s", &aMozilla50Windo[0x104 * n_3_]);// "Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
  n_3_ = flag_2;
}
LABEL_14:
n_3__1 = n_3_ + 1;
if ( n_3__1 >= 3 )
  n_3__1 = 0;
flag_2 = n_3__1;
hInternet = InternetOpenA(&szAgent, 0, 0i64, 0i64, 0);
hInternet_2 = hInternet;
if ( hInternet )
{
  hConnect = InternetConnectA(hInternet, &szServerName, UrlComponents.nPort, 0i64, 0i64, 3u, 0, 0i64);
  if ( hConnect )
```

Setting up internet connection

The malware then connects to the initial C2 server, which is a GitHub page located at https://github[.]com/alinaegorovaMygit. The malware reads the entire web page into a memory buffer using the InternetReadFile call.

The GitHub repository contains forks of three public projects that have not been modified or updated. Their purpose is merely to make the GitHub page appear legitimate and active. However, the author section of the GitHub page displays an interesting string:



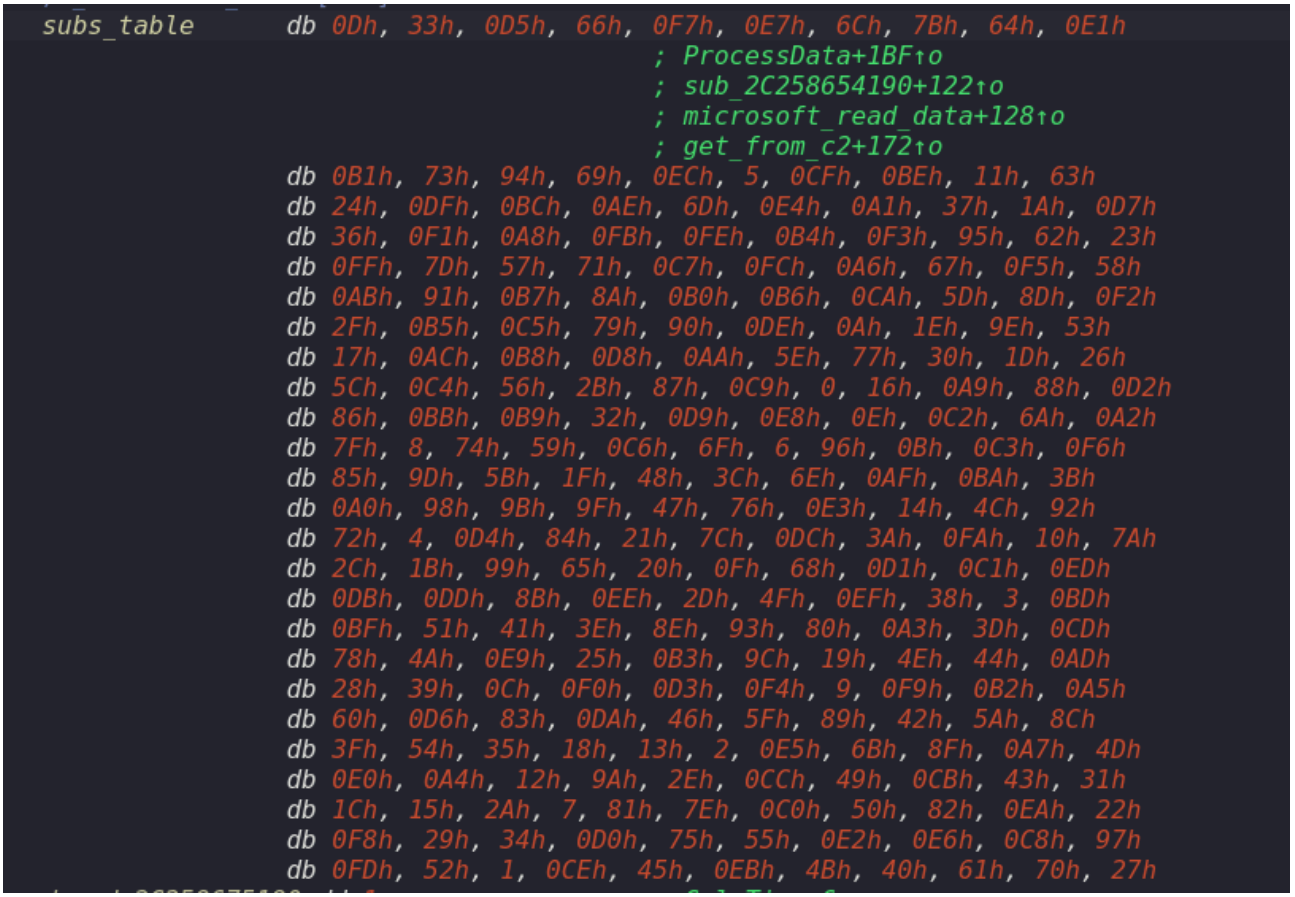
Hex string in the author section

We found data that looks like a hex string that starts and ends with the same byte pattern – “CDOY”. After the malware downloads the entire GitHub HTML page, it begins parsing it, searching specifically for the character sequence “CDOY”. When it finds it, it copies all the characters up to the second delimiter “CDOY” and then

stores them in a memory buffer. Next, the malware parses these characters, converting them from string values to hex values. It then decodes the string using a hardcoded charcode substitution table – each byte from the parsed string acts as an index in the charcode table, pointing to a substitutable byte, thus forming a new hex byte array.

```
      {
        low_nibble = next_char - 0x37;
      }
    }
    else
    {
      low_nibble = next_char - 0x30;
    }
    real_processed_data_[++current_index] = low_nibble | (16 * high_nibble);
  }
  while ( current_index < half_distance_1 );
}
get_main_struct_offset();
offset_counter = 0i64;
if ( (int)half_distance_1 > 0 )
{
  do
  {
    index_1 = (unsigned __int8)real_processed_data_[++offset_counter];
    real_processed_data_[offset_counter] = subs_table[index_1];
  }
  while ( offset_counter < half_distance_1 );
}
v30 = (void *)*((_QWORD *)parsed_response + 1);
```

Decoding algorithm

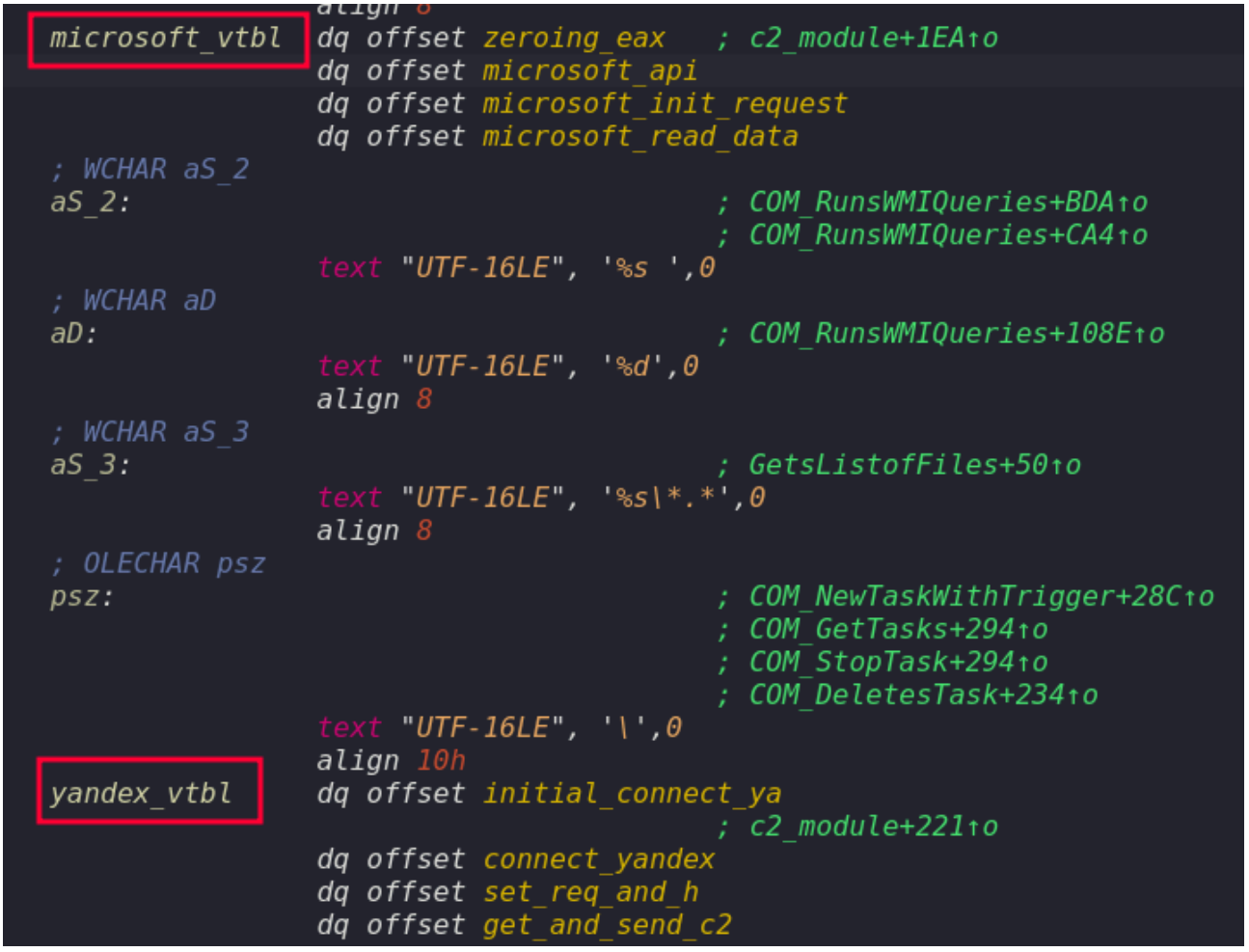


Charcode table

Alternatively, instead of connecting to GitHub, CloudSorcerer also tries to get the same data from hxxps://my.mail[.]ru/, which is a Russian cloud-based photo hosting server. The name of the photo album contains the same hex string.

The first decoded byte of the hex string is a magic number that tells the malware which cloud service to use. For example, if the byte is “1”, the malware uses Microsoft Graph cloud; if it is “0”, the malware uses Yandex cloud. The subsequent bytes form a string of a bearer token that is used for authentication with the cloud’s API.

Depending on the magic number, the malware creates a structure and sets an offset to a virtual function table that contains a subset of functions to interact with the selected cloud service.



Different virtual tables for Yandex and Microsoft

Next, the malware connects to the cloud API by:

- Setting up the initial connection using InternetOpenA and InternetConnectA;
- Setting up all the required headers and the authorization token received from the GitHub page;
- Configuring the API paths in the request;
- Sending the request using HttpSendRequestExA and checking for response errors;
- Reading data from the cloud using InternetReadFile.

The malware then creates two separate threads – one responsible for receiving data from the Windows pipe and another responsible for sending data to it. These threads facilitate asynchronous data exchange between the C2 and backdoor modules.

Finally, the C2 module interacts with the cloud services by reading data, receiving encoded commands, decoding them using the character code table, and sending them via the named pipe to the backdoor module. Conversely, it receives the command execution results or exfiltrated data from the backdoor module and writes them to the cloud.

Infrastructure

GitHub page

The GitHub page was created on May 7, 2024, and two repositories were forked into it on the same day. On May 13, 2024, another repository was forked, and no further interactions with GitHub occurred. The forked repositories were left untouched. The name of the C2 repository, “Alina Egorova,” is a common Russian female name; however, the photo on the GitHub page is of a male and was copied from a public photo bank.

Mail.ru photo hosting

This page contains the same encoded string as the GitHub page. There is no information about when the album was created and published. The photo of the owner is the same as the picture from the photo bank.

Cloud infrastructure

Service	Main URL	Initial path
Yandex Cloud	cloud-api.yandex.net	/v1/disk/resources?path= /v1/disk/resources/download?path= /v1/disk/resources/upload?path=
Microsoft Graph	graph.microsoft.com	/v1.0/me/drive/root:/Mg/%s/%s:/content
Dropbox	content.dropboxapi.com	/2/files/download /2/files/upload

Attribution

The use of cloud services is not new, and we reported an example of this in our overview of the CloudWizard APT (a campaign in the Ukrainian conflict with ties to Operation Groundbait and CommonMagic). However, the likelihood of attributing CloudSorcerer to the same actor is low, as the code and overall functionality of the malware are different. We therefore assume at this point that CloudSorcerer is a new actor that has adopted the technique of interacting with public cloud services.

Victims

Government organizations in the Russian Federation.

Conclusions

The CloudSorcerer malware represents a sophisticated toolset targeting Russian government entities. Its use of cloud services such as Microsoft Graph, Yandex Cloud, and Dropbox for C2 infrastructure, along with GitHub for initial C2 communications, demonstrates a well-planned approach to cyberespionage. The malware’s ability to dynamically adapt its behavior based on the process it is running in, coupled with its use of complex inter-process communication through Windows pipes, further highlights its sophistication.

While there are similarities in modus operandi to the previously reported CloudWizard APT, the significant differences in code and functionality suggest that CloudSorcerer is likely a new actor, possibly inspired by previous techniques but developing its own unique tools.

Indicators of Compromise

File Hashes (malicious documents, Trojans, emails, decoys)

Domains and IPs

hxxps://github[.]com/alinaegorovaMygit	CloudSorcerer C2
hxxps://my.mail[.]ru/yandex.ru/alinaegorova2154/photo/1	CloudSorcerer C2

Yara Rules

```
1  rule apt_cloudsorcerer {
2  meta:
3  description = "Detects CloudSorcerer"
4  author = "Kaspersky"
5  copyright = "Kaspersky"
6  distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER
7  OR SHARE ON ANY THREAT INTEL PLATFORM"
8  version = "1.0"
9  last_modified = "2024-06-06"
10 hash = "F701fc79578a12513c369d4e36c57224"
11 strings:
12 $str1 = "Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko"
13 $str2 = "c:\\windows\\system32\\mspaint.exe"
14 $str3 = "C:\\Windows\\system32\\msiexec.exe"
15 $str4 = "\\\\.\\PIPE\\"
16 condition:
17 uint16(0) == 0x5A4D and
```

```
18  all of ($str*)
19  }
20
```

MITRE ATT&CK Mapping

Tactic	Technique	Technique Name
Execution	T1059.009	Command and Scripting Interpreter: Cloud API
	T1559	Inter-Process Communication
	T1053	Scheduled Task/Job
	T1047	Windows Management Instrumentation
Persistence	T1543	Create or Modify System Process
	T1053	Scheduled Task/Job
Defense Evasion	T1140	Deobfuscate/Decode Files or Information
	T1112	Modify Registry
Discovery	T1083	File and Directory Discovery
	T1046	Network Service Discovery
	T1057	Process Discovery
	T1012	Query Registry
	T1082	System Information Discovery
Collection	T1005	Data from Local System
Command and Control	T1102	Web Service
	T1568	Dynamic Resolution
Exfiltration	T1567	Exfiltration Over Web Service
	T1537	Transfer Data to Cloud Account
