Kaspersky ICS CERT

kaspersky

Common TTPs of attacks against industrial organizations. Implants for remote access

Kirill Kruglov

Vyacheslav Kopeytsev

Artem Snegirev

20.07.2023 Version 1.1

In 2022 we investigated a series of attacks against industrial organizations in Eastern Europe. In the campaigns, the attackers aimed to establish a permanent channel for data exfiltration, including data stored on air-gapped systems.

Based on similarities found between these campaigns and previously researched campaigns (e.g., ExCone, DexCone), including the use of FourteenHi variants, specific TTPs and the scope of the attack, we have medium to high confidence that a threat actor called APT31, also known as Judgment Panda and Zirconium, is behind the activities described in this report.

To exfiltrate data and deliver next-stage malware, the threat actor (or actors) abuse(s) a cloud-based data storage, e.g., Dropbox or Yandex Disk, as well as a service used for temporary file sharing. They also use C2 deployed on regular virtual private servers (VPS). In addition, the threat actor(s) deploy(s) a stack of implants that collect data from air-gapped networks via infected removable drives.

For most implants, the threat actor(s) use(s) similar implementations of DLL hijacking (often associated with Shadowpad malware) and memory injection techniques, along with using RC4 encryption to hide the payload and to evade detection. In addition, libssl.dll or libcurl.dll was statically linked to implants to implement encrypted C2 communications.

In total we have identified over 15 implants and their variants planted by the threat actor(s) in various combinations.

The entire stack of implants used in attacks can be divided into three categories based on their roles:

- First-stage implants for persistent remote access and initial data gathering
- Second-stage implants for gathering data and files, including from air-gapped systems
- Third-stage implants and tools used to upload data to C2

In this article (which is the first part of the report) we analyze common TTPs of first-stage implants used by threat actors to establish a persistent remote access channel into the infrastructure of industrial organizations.

The full report is available on the Kaspersky Threat Intelligence portal.
For more information please contact ics-cert@kaspersky.com.

Variants of FourteenHi

FourteenHi is a malware family discovered in 2021 in a campaign that was dubbed ExCone (1, 2), active since mid-March 2021 and targeting government entities. In 2022 we discovered new variants used in attacks on the infrastructure of industrial organizations.

Various samples of FourteenHi (both x64 and x86) are significantly different from each other in terms of their code structure, implementations of their loaders, and C2 types. But their core distinctive features, such as the C2 communication protocol and the list of commands, are pretty much the same. The most significant difference exists between x86 and x64 variants of FourteenHi.

Samples for x64 have persistence capabilities and a 2-step C2 communication protocol. They accept a relatively long list of commands, including:

- upload arbitrary files,
- download arbitrary files,
- run arbitrary commands,
- set communication delay,
- start reverse shell,
- terminate own process and remove persistence.

To protect communication with C2, they use the API of the statically linked OpenSSL library. In addition, they use RC4 to encrypt / decrypt the data they send / receive from C2.

FourteenHi x64 code for parsing a C2 response

```
if ( command == 0x253AB )
{
  if ( v7 == 4 )
  {
    handle = kernel32_CreateRemoteThread_902(0i64, 0i64, C2_command_ReadWrite_file, *buffer);
    kernelbase_CloseHandle_425(handle);
  }
}
else if ( command == 0xB8C2D )
{
  exception = check_alloc_exception(24i64);
  qword_1BAAAC251D8 = exception;
  *exception = 0i64;
  exception[1] = -1i64;
  exception[2] = -1i64;
  CreateRemoteThreat_C2_command_CMD_exec(exception, cmd_command, buffer, SHIDWORD(command));
}
```

FourteenHi x64 code for parsing commands in a C2 response

```
if ( subCommand == 0xDE372 )
{
  WriteFile(data, &filePath, dataLen);
}
else if ( subCommand == 0xCB76F )
{
  ReadFile(data, &filePath, dataLen);
}
```

The samples for x86 have no persistence capabilities, are not linked with OpenSSL, but still use RC4 encryption. They use a 1-step communication protocol, but the list of commands is almost the same, except for the removal of persistence mechanisms.

FourteenHi x86 simple switch case for C2 response command matching

```
                align 4
CNC_CommandCode dd offset Command_841_ExecCmd_Send_CnC
                                        ; DATA XREF: sub_401E30+17E↑r
                dd offset Command_842_WriteFile ; jump table for switch statement
                dd offset Command_843_ReadFile_Send_CNC
                dd offset Command_844_Sleep
                dd offset Command_845_Exit
                align 10h
```

The absence of persistence capabilities (which usually require privilege escalation) in variants for x86 and the overall lightness of compiled code make them good candidates for an initial infection stage, which may be used to collect initial information on a host or the local network, download next-stage malware and data stealers, and provide a remote shell for the threat actor. Nevertheless, the threat actor may easily add persistence to the implant by creating a task in Windows Task Scheduler, as we have observed in the wild.

The loading scheme is more or less the same for all of the variants and consists of three main components used by the threat actor to deploy an implant on a victim's machine:

1. Legitimate application that is vulnerable to DLL hijacking.
2. Malicious DLL that is loaded via DLL hijacking and is used to read and decrypt the FourteenHi payload from a binary data file and inject it into some system process such as svchost.exe or msiexec.exe.
3. A binary data file containing the FourteenHi binary code encrypted with RC4.

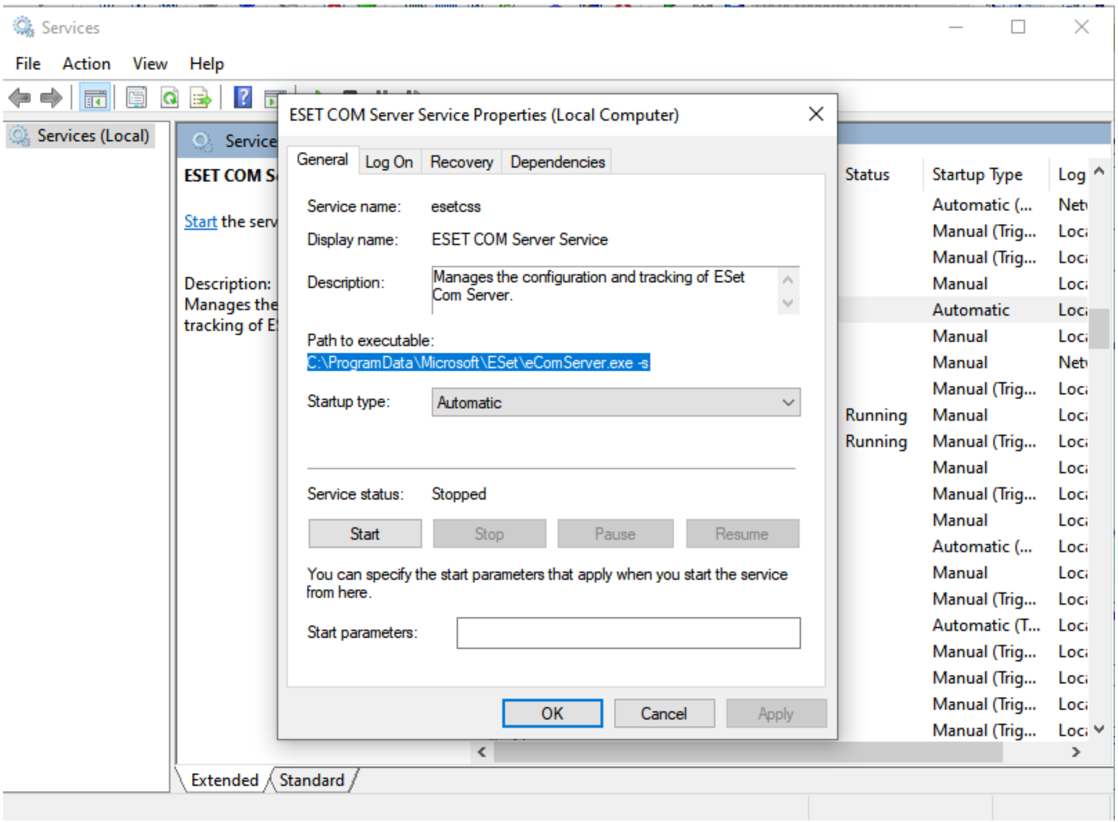
All known variants of FourteenHi have config data embedded in their code and encrypted with RC4. The configuration defines the campaign ID, C2 address and port. The configuration of FourteenHi x64 also defines the name and description of the Windows service it creates for persistence when executed without parameters.

MeatBall backdoor

The MeatBall backdoor is a new implant that we discovered in the process of researching attacks. It has vast remote access capabilities, including making lists of running processes, connected devices and disks, performing file operations, capturing screenshots, using remote shell, and self-updating. The implant exists in variants for x86 and x64.

The implant uses a loading scheme based on the DLL hijacking technique, but unlike many other implants, the payload is stored in the malicious DLL loader itself, not in a separate file.

When the vulnerable host application is executed without parameters, the implant calls IsNTAdmin and, if it has sufficient privileges, creates a service named “esetcss”. Otherwise it simply adds itself to the registry key “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\esetcss” to be automatically executed at OS startup.



Service created by the MeatBall implant

In both cases the implants are configured to be executed with the parameter “-S”, which tells the implant to read the payload from its own module (.dll) file, decrypt the payload using a one-byte XOR key, start “svchost.exe”, and inject the decrypted payload into it. Then it starts the main C2 communication loop by calling ResumeThread for “svchost.exe”.

The implant is statically linked with libssl.dll, which is used for SSL encryption of C2 communication.

Command codes	Description
0x2, 0x11	Update C2 address
0x3	List running processes
0x5	List connected devices
0x6	List connected disks
0x7, 0x8	Collect datetime attributes for files in the folder specified
0x9	Terminate process
0xB	Write file
0xC	Create file
0xD, 0xF	Upload size and content of a file
0x10	Delete file
0x13	Run file
0x14	Close C2 connection
0x15, 0x1C, 0x1D, 0x1E	Terminate own process
0x16, 0x17,0x18, 0xA, 0x1F	Create remote shell
0x19	Delete files in a folder recursively
0x1A, 0x1B	Capture screenshot

Implant using Yandex Cloud as C2

Another interesting implant we found was one that uses the Yandex Cloud data storage as a C2 (https://cloud-api.yandex[.]net) similarly to the malware described in an earlier report. The implant uses a DLL hijacking based loading scheme, in which the malicious DLL decrypts the implant's body stored in a separate file and injects it into a legitimate process's memory.

The implant uses statically linked libcurl.dll for SSL-encrypted communication. First it creates a mutex named "*Njg8*" to prevent more than one instance of itself from being executed at any time, then it collects the following data on the host:

- Computer name
- User name
- IP address
- MAC address
- OS version
- Path to %System%

To upload the data collected to C2, the implant sends a request using an embedded API token to create a directory with a name that is unique to the victim host. Then it creates a file with the prefix "1770_" and the extension ".dat", saving all information collected in that file.

The main loop of the implant periodically checks a cloud folder named "content" for the latest uploaded files with prefixes "1780_", "1781_" and "1784_":

- Files with prefixes "1780_" and "1781_" contain code in the PE format, e.g., a legitimate application and a malicious DLL for next-stage DLL hijacking.
- Files with the prefix "1784_" contain commands to be executed using cmd.exe. The output is then stored in a log file, which is immediately uploaded back to C2 and removed from the victim host.

All uploaded and downloaded data is encrypted with the RC4 algorithm.

Strings found in a sample which uses Yandex Disk

```
aCrateDir        db 'crate dir',0Ah,0     ; DATA XREF: sub_452050+5A2↑o
                 align 10h
aUploadHostInfo  db 'upload host info',0Ah,0 ; DATA XREF: sub_452050+75D↑o
                 align 4
aBeginExeccomma  db 'begin execCommand',0Ah,0 ; DATA XREF: sub_452050+A82↑o
                 align 4
aSleeptimeD      db 'sleeptime:%d',0Ah,0  ; DATA XREF: sub_452050+BB5↑o
                 align 4
asc_627A48       db '/',0                 ; DATA XREF: sub_452C30+78↑o
                                          ; .text:loc_45C9CF↑o ...
                 align 4
aContent_0       db '/content/',0         ; DATA XREF: sub_452C30+11D↑o
                 align 4
a1780            db '1780',0              ; DATA XREF: sub_452C30+1D2↑o
                                          ; sub_452C30+342↑o
                 align 10h
a1781            db '1781',0              ; DATA XREF: sub_452C30+20E↑o
                 align 4
a1784            db '1784',0              ; DATA XREF: sub_452C30+3EA↑o
```

Log containing the result of command execution using cmd

```
C:\Windows\system32>sc query WinCoreSvc

SERVICE_NAME: WinCoreSvc
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 0   (0x0)
        SERVICE_EXIT_CODE  : 0   (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

C:\Windows\system32>
```

Conclusion

The tendency to abuse cloud services (e.g., Dropbox, Yandex, Google, etc.) is not new, but it continues to expand, because it is hard to restrict / mitigate in cases when an organization's business processes depend on using such services.

Threat actors keep making it more difficult to detect and analyze threats by hiding payloads in encrypted form in separate binary data files and by hiding malicious code in the memory of legitimate applications via DLL hijacking and a chain of memory injections.

Recommendations

- Install security software with support for centralized security policy management on all servers and workstations and keep the antivirus databases and program modules of your security solutions up-to-date.
- Check that all security software components are enabled on all systems and that a policy is in place which requires the administrator password to be entered in the event of attempts to disable protection.
- Consider using Allowlisting and Application Control technologies to prevent unknown applications from being executed.
- Consider using the Golden image configuration mode for Allowlisting and Application Control to prevent any software that is not allowed (including known vulnerable applications) from being executed.
- Consider restricting internet access from the OT network by default, allowing access to specific users for limited periods of time and only when it is required to perform their duties.

Appendix I – Indicators of compromise

Note: The indicators in this section are valid at the time of publication.

The full version of indicators of compromise, including Yara rules, is available in a .ioc file on the Kaspersky Threat Intelligence portal.

Variants of FourteenHi

MD5

```
7332710D10B26A5970C5A1DDF7C83FBA (mpsvc.dll)
2A1CFA6D17627EAAA7A63F73038A93DA (taskhost.doc)
BB02A5D3E8807D7B13BE46AD478F7FBB (cclib.dll)
22E66E0BE712F2843D8DB22060088751 (ToastUI.exe.png)
D75C7BD965C168D693CE8294138136AE (ToastUI.exe.dat)
```

C2 IP/URL

```
sfb.odk-saturn[.]com/dialin/login
87.121.52[.]86
```

Backdoor.Win32.MeatBall

MD5

```
FFF248DB8066AE3D30274996BAEDDAB6 (oleacc.dll)
```

C2 IP/URL

```
freetranslatecenter[.]com
help.freetranslatecenter[.]com
onlinenewscentral[.]com
onlinemapservices[.]com
search.onlinemapservices[.]com
help.onlinemapservices[.]com
apps.onlinemapservices[.]com
edit.onlinemapservices[.]com
booking-onlines[.]com
81.28.13[.]74
92.38.160[.]142
92.38.188[.]135
92.38.190[.]55
```

```
103.221.222[.]133
193.109.78[.]243
193.124.112[.]206
194.87.95[.]125
```

Implant using Yandex Cloud as C2

MD5

```
A05D6D7A6A1E9669FC4C61223DA3953F (dbghelp.dll)
2F5C889A819CFE0804005F7CE5FD956E (vmService.pkg)
```

Appendix II – MITRE ATT&CK Mapping

The table below contains all the TTPs identified in the analysis of the activity described in this report.

Tactic	Technique Number	Technique Name and Description
Execution	**T1204.002**	**User Execution: Malicious File** A system is infected when the user runs the malware believing it to be a legitimate document.
	T1059.003	**Command and Scripting Interpreter: Windows Command Shell** Uses cmd.exe to execute multiple commands.
	T1106	**Native API** Uses the CreateProcessW function to execute commands in the Windows command line interpreter
	T1053.005	**Scheduled Task/Job: Scheduled Task** Malware is executed with a Windows task created by the threat actor.
Persistence	**T1547.001**	**Registry Run Keys / Startup Folder:** Malware achieves persistence by adding itself to the Registry as a startup program.
	T1543.003	**Create or Modify System Process: Windows Service** Installs itself as a service to achieve persistence.
	T1053.005	**Scheduled Task/Job: Scheduled Task** Malware is executed with a Windows task created by the threat actor.
Defense Evasion	**T140**	**Deobfuscate/Decode Files or Information** Uses RC4 key to decrypt the malware configuration, as well as to protect communication.
	T1055.002	**Process Injection: Portable Executable Injection** Malware injects itself into various legitimate processes upon execution (msiexec.exe, svchost.exe).
	T1497.001	**System Checks** Employs various system checks to detect and avoid virtualization and analysis environments.

	T1497.003	**Time Based Evasion** Employs various time-based methods to detect and avoid virtualization and analysis environments.
	T1574.002	**Hijack Execution Flow: DLL Side-Loading** Threat actors abuse a legitimate application binary to load malicious DLL.
Discovery	**T1033**	**System Owner/User Discovery** Threat actors use systeminfo, whoami, and net utilities to get information about the user and the infected system.
	T1057	**Process Discovery** Threat actors use tasklist to enumerate running processes.
Command and Control	**T1071.001**	**Application Layer Protocol: Web Protocols** Malware uses HTTPS and raw TCP for communication with C2.
	T1573.001	**Encrypted Channel: Symmetric Cryptography** Malware uses RC4 and SSL TLS v3 (using libssl.dll) to encrypt communication.
Exfiltration	**T1041**	**Exfiltration Over C2 Channel** Threat actors exfiltrate data using Dropbox, Yandex Disk, Yandex email and temporary file sharing services as a C2 channel

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