Title

A Tale of Two Industroyers: It was the Season of Darkness

ISBN

979-8-3503-3131-8

Authors

Salazar, Luis

Castro, Sebastián R

Lozano, Juan

et al.

Publication Date

2024-05-23

DOI

10.1109/sp54263.2024.00162

A Tale of Two Industroyers: It was the Season of Darkness

Luis Salazar*, Sebastián R. Castro*, Juan Lozano*, Keerthi Koneru*, Emmanuele Zambon†
Bing Huang‡, Ross Baldick‡, Marina Krotofil§, Alonso Rojas¶ and Alvaro A. Cardenas*
*University of California, Santa Cruz †Eindhoven University of Technology
‡The University of Texas at Austin §Information Systems Security Partners ¶Axon Group

***Abstract*—In this paper, we study two pieces of malware that attempted to create blackouts in Ukraine. In particular, we design and develop a new sandbox that emulates different networks, devices, and other characteristics so that we can execute malware targeting substation equipment and understand in detail the specific sequence of actions the attackers could perform on substation equipment. We also study the effects that future similar malware can have. Our findings include new malware behavior not previously documented (such as the detailed algorithm for the MMS protocol payload) and an illustration of how attacking different targets will produce different effects.**

1. Introduction

In less than a decade, Ukraine has suffered from several cyber attacks attempting to cause electrical outages. On December 23, 2015, in a dark and cold winter, Ukraine suffered a blackout caused by cyber attacks [1]. This was the first confirmed case worldwide of cyberattacks intentionally targeting the power grid. In this first incident, attackers gained remote access to the industrial networks of power companies, and a remote adversary operated the human-machine interface of operators, opening circuit breakers manually.

A year later, on December 17, 2016, a fifth of Ukraine's capital, Kyiv, experienced another blackout [2], [3]. This time, the target was a transmission utility, and unlike the previous year when remote human attackers opened the circuit breakers, the attack in 2016 was launched automatically by the first known industrial malware targeting the power grid: Industroyer [4] (also known as CrashOverride [5]).

Finally, on April 8, 2022, in the first months of the Russian invasion of Ukraine, operators discovered another malware tailored to attack circuit breakers automatically. This new malware was called Industroyer 2, and it represented yet another attempt to target Ukraine's power grid amid many other Russian cyber-attacks coordinated alongside their kinetic military action [6].

The two Industroyer malware payloads represent a watershed moment for the security of critical infrastructures for several reasons: (1) they represent the first and only known malware samples specifically deployed by malicious actors to target the power grid. (2) Industroyer is the only known malware that caused a power blackout successfully. (3) Industroyer 2 exemplifies how modern wars combine cyber and kinetic attacks to maximize a show of force. (4) While the attackers in 2015 needed a real-time network connection to the power company to open circuit breakers, Industroyer and Industroyer 2 can, in principle, be deployed in air-gapped systems (or systems with strong network segmentation where remote connections are not allowed) because the malware could automatically target power grid equipment without the need for human feedback. (5) Industrial malware may have other advantages over remote (manual) attackers, like the ability to repeatedly send commands to devices faster than an operator could respond to them.

In the remainder of the paper, we will refer to "Industroyer" as "*Industroyer 1*" to avoid ambiguity.

While Industroyer 1 and Industroyer 2 represent the only known malware samples specifically designed to cause power blackouts, they have received little attention from the academic community. There have been several industrial reports about them [4], [5], [7], [8], [9], [10], [11], [12], but unfortunately, these reports tend to focus on the Windows exploits and other indicators of compromise and do not focus on the power grid.

We believe academic researchers working on the security of the power grid want to learn the details of how Industroyer 1 and 2 launched the attacks and the impact that similar malware may have in the future so that we can understand the attacker's decision-making behind the payload design and also anticipate future attacks. As we will explain later in Table 6, these industrial reports do not give us enough details about the design behind the industrial payloads. For example, these reports do not answer (1) whether the payload against the power grid expects a central control room or a substation, (2) how many substations or devices can be attacked, (3) what the specific sequence of industrial commands these malware samples could send to different devices in the power grid, and (4) what is the impact that these attacks may cause in the bulk power system (e.g., can these attacks lead to cascading outages, and if not, what would be the alternate impacts).

To answer these questions, in this paper, we design and develop a new industrial sandbox to execute malware targeting the power grid. Anti-virus companies use various tools and sandboxes to execute regular IT malware dynamically. However, as far as we are aware, there is no sandbox to understand the execution of power grid malware because the sandbox needs to represent a high-fidelity simulation of industrial systems and include the unique industrial protocols the malware expects. In this paper, we develop such a sandbox and use it to understand in detail the operation and potential impact of Industroyer 1 and Industroyer 2, revealing previously unreported functionalities of these attacks. We plan to continue developing this sandbox and extend it

to provide other use cases.

As our contributions:

- We provide the first in-depth analysis of Industroyer 1 and 2. Our paper fills the gap between previous industrial reports and the knowledge that power grid operators and industrial control experts need to understand the specific attack commands targeting power grid equipment. Our analysis finds previously unreported behavior of Industroyer 1 and 2.
- We design and develop *NEFICS*[1], a new industrial control sandbox to test malware interacting with power grid equipment. In particular, we provide environments that emulate a control room for a power grid operator with remote connections to substations or a substation network with local connections to electrical equipment. Our software artifacts are openly available.
- We discuss and analyze the evolution of attacks, the attack surface of the grid, and the impact that these attacks may have on the bulk power system.

2. Background

The power grid is a distributed control system operated by a federation of companies, each controlling a specific part of the infrastructure. For simplicity, we call any company monitoring and controlling a section of the power grid a **Transmission and Distribution System Provider (TDSP)**.

A typical TDSP has a central **Control Room (CR)**. The control room is where the physical process is monitored and controlled with the help of human operators. A CR hosts the Supervisory Control and Data Acquisition (SCADA) applications, which collect information from remote devices and remotely actuating equipment. A CR usually has several specialized computers focusing on different aspects of the operation of the power grid, such as a Historian that keeps track of the operational states of the power grid, Human-Machine Interfaces (HMIs) that operators can use to visualize and manage the physical process, and OPC servers used for compatibility. Most of the sensors and actuators monitored and controlled by the CR, can be found in various substations. So, CRs have several remote connections to a set of distributed substations.

Substations are distributed facilities that house equipment to monitor and control the power grid. These include transformers, voltage regulators, telemetry equipment, and protection devices. Substations can be indoor or outdoor facilities and can be staffed or unattended. Some substations have their own Control Room (CR) to monitor and control the local equipment.

A SCADA server in the central CR interacts with **Remote Terminal Units (RTUs)**. RTUs are embedded computers deployed in remote substations to collect data from various devices within the substation and report this to the CR. While RTUs have been the traditional endpoint for communications between a CR and a substation, some modern substation endpoints are called *substation gateways*.

Within a substation, an internal local network connects various devices, including **Intelligent Electronic Devices (IEDs)**. An IED is an embedded computer in a substation that directly monitors and controls power system equipment. IEDs host logical nodes (computer programs) such as **Control Switches (CSWI)**. A CSWI, in turn, controls a physical device called a **Circuit Breaker (CB)**. CBs are independently mounted poles with a chamber that has an isolating material able to stop current or arc effect currents. CBs can be used to disconnect parts of the electric grid.

1. https://github.com/Cyphysecurity/NEFICS

2.1. Industrial Communication Protocols

CR to substation: The communication between a CR and a remote substation is realized by the SCADA system through various industrial protocols. Early SCADA systems used point-to-point serial communication protocols like Modbus and IEC 101, which emerged as the communication standards for connecting a CR with substations. IEC 60870-5-101 (**IEC 101**) enables telecontrol messages between the CR and substations. This protocol uses a low bandwidth bit-serial communication (serial communication) to transmit data objects and services over geographically wide areas. With the advent of TCP/IP networks, new Internet-compatible protocols such as IEC 60870-5-104 (**IEC 104**) and DNP3 became popular for connecting substations to a CR. While these protocols support Internet standards (TCP/IP), they are generally used over private networks leased from telecommunication providers, so, in principle, the links between CRs and substations are not accessible through the public Internet.

Within a Substation: Ethernet-based communications *within* a substation are defined by the **IEC 61850** standard. IEC 61850 includes three protocols: **MMS** (used by workstations to get data from IEDs), GOOSE (this is how IEDs communicate between themselves), and SV (used by IEDs to get data from the physical process). IEC 61850 also defines a universal language for substation configuration, the Substation Configuration Language (SCL), which includes a logical and abstract representation of the substation components and facilitates the automation of substations.

Finally, **OPC (OLE for Process Control)** is a standard to facilitate intercommunication among ICS devices. By leveraging Microsoft's standards *Object Linking and Embedding (OLE)*, *Component Object Model (COM)*, and *Distributed Component Object Model (DCOM)*, OPC models the interface to translate OPC requests (e.g., read/write) to requests understandable by the target device [13]. What is known as *OPC Classic* is the initial standard that used to work only on Windows systems and is composed by the *OPC Data Access (DA)*, *OPC Alarms & Events (AE)*, and *OPC Historical Data Access (OPC HDA)*. More recently, the OPC Foundation created the *OPC Unified Architecture (UA)* [14]; a standard that works on multiple environments (not only on Windows) and includes multiple capabilities, such as Session Encryption, Message Signing, and Sequenced Packets.

An attacker must understand these protocols to design an effective payload capable of communicating with the equipment in predictable ways. Defenders can also understand the potential impact of the malware by understanding these protocols and the architecture of industrial networks. For example, Industroyer 1 was a Swiss army knife for attacking power grid systems. It had payloads for IEC 101, IEC 104,

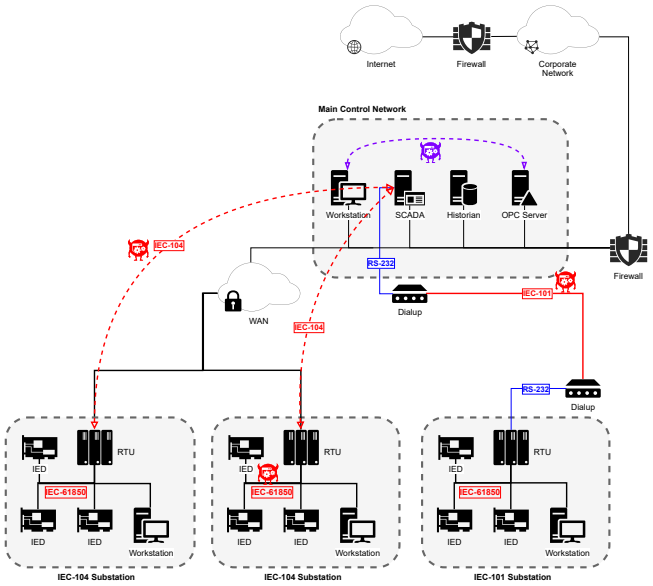


Figure 1: Industroyer capabilities (denoted by the devils) in a power grid. Industroyer can interact using the industrial protocols in red and purple.

IEC 61850, and OPC, while Industroyer 2 only targeted IEC 104. This meant that Industroyer 1 could be deployed in a CR and use IEC 101 or IEC 104 to talk to substations, or if it was deployed in a substation, it could use IEC 61850 to talk to IEDs.

Figure 1 shows how Industroyer 1 could have interacted with several devices in the power grid; e.g., it could be launched from the SCADA computer in the main control room and use serial connections (such as IEC 101) or TCP/IP connections (such as IEC 104) to reach substations. Alternatively, it can be deployed in a workstation within a substation and use IEC 61850 to discover and target IEDs. As far as we are aware, this is the most detailed network architecture representing the capabilities of Industroyer 1 and 2 and the specific locations where they could interact with power grid equipment. This diagram also helps us design a sandbox that can convince Industroyer 1 and 2 to believe they are inside a power grid.

3. Sandbox Design

Malware analysis is more effective if it can be deployed in a sandbox that represents a realistic environment where it is intended to be deployed. A securely isolated virtual machine is sufficient to understand general malware affecting system resources. However, we must emulate or simulate new equipment and industrial protocols to interact with the malware. A cost-effective solution is to use a simulated environment replicating the conditions of such critical infrastructures, allowing us to observe both the malware's digital behavior and the attack's physical repercussions.

We evaluated some existing power grid testbeds to determine whether they would be a suitable starting point for our purposes (Table 1). We looked into whether the system provides a suitable device and network components simulation, is fully open-source, easily extensible, and supports all the protocols that Industroyer uses.

While searching for different testbeds, we noticed that the ones with an accurate power grid model lacked the capabilities to support the required network protocols (in particular, none supported IEC-101 and IEC-104), and those with accurate network simulations lacked a reliable physical model. Most of them have some external component such as a system-in-the-loop (SITL) or hardware-in-the-loop (HITL), making it difficult to replicate, or use proprietary modules that prevent it from being easily extensible, as its original goal constraints the design. HITL testbeds aim to test and refine a system by adding hardware components that reflect the changes in the physical system. In contrast, SITL testbeds add an existing system to simulated real-time devices to test and refine these simulations.

TABLE 1: Power grid / Smart Grid testbeds.

	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]
Physical simulation	●	●	●	○	○	●	○	○	●
SITL / HITL	●	○	●	●	●	●	●	●	●
Network simulation	●	●	●	●	●	●	○	●	●
Proprietary modules	●	●	●	●	●	●	●	●	●
Industroyer protocols	0/4	0/4	0/4	2/4	2/4	2/4	1/4	1/4	1/4

○: Not supported ●: Supported

Overall, these environments rely on proprietary software or hardware, limiting their extensibility and openness. In addition, none of them support all the devices or the industrial protocols we need in our study; therefore, we now propose a new open-source, extensible solution. We identified the following criteria when designing our sandbox:

Isolation: It must be isolated from any other network.
Secure-coupling: We must be able to integrate a machine infected with malware.
Network tracking: A network traffic capture is essential to understand how the malware interacts with its targets.
Flexibility: It must support various industrial control protocols. In this specific instance, power grid protocols for remote substation control and substation automation.
Physical simulation: It must simulate changes in a physical process.
Customizable: The simulation scenarios must be configurable.
Extensible: It must support future scenarios with different processes.

Figure 2 illustrates the proposed design. **Isolation**, **Secure-coupling**, and **Network traffic** are three characteristics that a virtual network linked to a virtual machine's network interface offer. Two virtual machines linked by a host-only network in the hypervisor prevent potentially hazardous network traffic from leaving the secure environment (Figure 2 ②). Moreover, having all the malicious traffic routed to the simulated network via a virtual machine interface allows us to capture any network traffic for analysis.

To implement the virtual network itself, kernel namespaces, often used in container solutions, provide a viable way of simulating multiple hosts in a single Linux machine. Mininet uses the same concept to create virtual network

hosts. Since it is Python-based, we can develop a way to integrate a physical simulation into the network emulation provided by Mininet. Moreover, Mininet uses Open vSwitch to interconnect the simulated hosts, providing a software-defined network solution that allows us to connect the simulation to a virtual machine's host-only network interface, integrating the infected host to the simulated network (Figure 2 ①). This solution offers a secure and isolated network to capture and analyze the network traffic.

We can achieve **Flexibility**, **Extensibility**, and **Customizable** characteristics by implementing the sandbox in separate modules. Any industrial control protocol supported by the sandbox would be a separate module, dynamically loaded and enabled as required. By making the sandbox modular by design, we can incrementally add new physical processes and industrial control protocols as needed. Moreover, implementing a handler component to parse a configuration file with the description of the desired scenario allows us to build complex topologies dynamically without explicitly developing the scenario but rather the modules comprising it.

Finally, the **Physical emulation** must provide realistic data over the implemented protocols. We must interconnect the physical simulation with the network simulation (Figure 2 ③). For this purpose, we identified two primary components for our sandbox: the communication with the network emulation and the physical process.

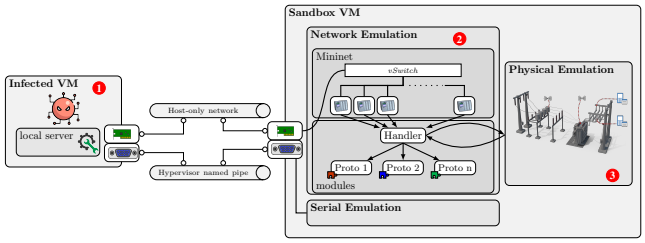


Figure 2: Sandbox architecture.

Since one of our requirements is to support various industrial control protocols, albeit not necessarily at the same time, each simulated host must be able to dynamically enable or disable the support for an industrial control protocol. Moreover, the network topology and simulated hosts must be configurable. Therefore, there needs to be a component that parses the configuration requirements and uses each supported protocol as needed. Since this component will essentially handle the information flow of the simulated industrial device, we could tie this component to the physical simulation, requesting and updating values from the simulated physical process and translating these values into the industrial control protocol.

With this information handler in mind, we can implement the different supported industrial control protocols into modules that the handler can dynamically import to enable a specific protocol and communicate with the infected machine or any standard industrial control client.

This handler presented the primary implementation challenge. Once the sandbox instantiates a device, the virtual device must follow with an instantiation of this handler. Therefore, each virtual device implementation must have a unique handler that bridges the device with the different network protocols and the physical simulation. Following our flexibility and extensibility requirements, these handlers must be imported and instantiated at runtime. To overcome this complexity, we implemented a generic launcher that checks the device and handler modules against the provided configuration at runtime and creates the required instances. Using this launcher component, we do not restrict the implementation to a particular scenario and allow the sandbox to be extended to include additional physical processes and devices.

So far, we have implemented modules for handling IEC-60870-101, IEC-60870-104, and Modbus TCP using Scapy, IEC-61850 using the iec61850bean library, a Matrikon OPC server for OPC-DA, and CPPPO for Ethernet/IP CIP.

IEC-60870-101 (Serial) and IEC-60870-104 (TCP/IP) and both remote control protocols are used by operators to control multiple substations from a central control room. Therefore, this scenario simulates the remote communication between an infected host in a control center and multiple remote substations linked by a WAN or a legacy serial dial-up connection. We use the IEC-61850 module to simulate devices within the local LAN of a substation. Thus, the infected machine in this scenario would be a local operator workstation inside the substation within the local LAN.

For the physical process simulation, we can either have one virtual host do all the simulations and coordinate with the other hosts or have each host simulate part of the physical process and synchronize the simulation state with the relevant hosts. In either case, we need the simulated hosts to communicate with each other and synchronize the state of any physical sensors and actuators handled by the simulated devices. This physical communication must be: (1) **Fast**, meaning that the physical process simulation must be as realistic as possible and run in real-time. (2) **Up-to-date**, meaning that each device holds only the most recent data. (3) **Concise**, meaning that we only send the necessary values, and (4) **Addressable** as we want the values to reach the desired device.

A UDP protocol is sufficient for the **fast** and **up-to-date** requirements, as there would be no issue with a missing packet if a device receives a more recent packet, and the lack of flow control offers a speed advantage over a TCP protocol. As for **conciseness** and **addressability**, a fixed-size structure of bytes carrying the necessary information fulfills this requirement. We designed a simple small payload that allows us to transmit concise sensor and actuator values to and from the simulated devices, with addressable device IDs, as most industrial control protocols do.

Our proposed solution is a UDP protocol that uses fixed 28-byte packets containing sender and receiver IDs, a message type, two integer values, and two float values. Sender and Receiver IDs make the protocol addressable in the application layer, meaning that the sender can broadcast the packet without knowing the recipient's IP address and effectively transmit a value, which is helpful in scenarios in which we dynamically build a network topology without knowing the IP addresses that Mininet assigns to the hosts. The message type allows us to customize the messages and specify the packet's data, be it a range, set-point, sensor value, or actuator value. Since we want a standardized way

of sending the data, we always send a fixed-size payload, and it is up to the message type to determine the semantics of the four values.

Using this protocol, we construct the physical simulation in separate modules that we instantiate as needed. These modules use the protocol to communicate with the simulation handlers, providing them with the up-to-date state of the sensors and receiving any state changes for the actuators to simulate the modifications in the simulation.

4. Malware Analysis

TABLE 2: Malware samples.

Sample	Industroyer	SHA-256
101.dll	1	a319551ef72492b3cd489de676b2f6e7938a5ef23e572d36dd742b599686caac
104.dll	1	7907dd95c1d36cf3dc842a1bd804f0db511a0f68f4b3d382c23a3c974a383cad
61850.dll	1	4e7d2b269088c1575a31668d86de95fd3dde6caa88051d7ec110f7f150058789
haslo.exe	1	ad23c7930dae02de1ea3c6836091b5fb3c62a89bf2bcfb83b4b39ede15904910
opc.exe	1	156bd34d713d0c8419a5da040b3c2dd48c4c6b00d8a47698e412db16b1ffac0f
svchost.exe	1	7cc9ac6383437dd96161b93b017500a22a2c8d05f58778b9b9fce8ea73304546
40_115.exe	2	d69665f56ddef7ad4e71971f06432e59f1510a7194386e5f0e8926aea7b88e00

Our Industroyer samples are summarized in Table 2. Industroyer 1 consists of a set of DLLs and executable files. In contrast, Industroyer 2 is a single executable file. Both malware versions are post-exploitation tools (final payload), which are deployed after a successful intrusion.

The attackers built Industroyer 1 as a malicious framework. A launcher component (svchost.exe) executes the desired framework payload, masquerading it as a Windows service for persistence. The adversaries can launch a different service for each supported industrial control protocol (i.e., 101.dll, 104.dll, 61850.dll, and opc.exe) using the corresponding configuration file, initiating the malicious payload. Once they complete their primary objective, they can execute the wiper component (haslo.exe), which wipes critical industrial information from the infected host and prevents a prompt solution against the attack.

Unlike its predecessor, Industroyer 2 is a single standalone executable file with a hard-coded configuration that executes all the necessary actions for the attack and terminates its execution once it accomplishes the primary objective. During the Industroyer 2 intrusions, adversaries also deployed several wiper components (CaddyWiper for Windows systems, ORCSHRED, SOLOSHRED, and AWFULSHRED for Linux and Solaris systems), following the same general modus operandi adopted during the Industroyer 1 attack.

4.1. Methodology

To understand the malware capabilities, we studied each payload sample through three sequential stages: static analysis, dynamic analysis, and semantic analysis.

Static analysis: Initial sample exploration to gather sufficient information about each malicious image without executing them. Some of the tests performed involve strings extraction, gathering of import/export functions, and detection of anti-debugging mechanisms (e.g., obfuscation/encryption). In this stage, we determined various execution constraints, such as the file format of the expected configuration files for specific payloads, a control flow graph analysis to determine the expected values by the malware, and entry points for further debugging.

Dynamic analysis: Behavioral study of the sample. The objective is to understand how the malware behaves when executed. Based on the execution constraints obtained from the static analysis, we deployed an isolated environment that simulated the scenario expected by the malicious software. The output of this stage provides information about the interaction between the malware and its environment (e.g., packet captures, log files, etc.) and helps us to craft the adequate packets for the *NEFICS* modules in scenarios where the malware waits for certain values and packet sequences.

Semantic analysis: Semantic study of the sample. The objective is to understand the real impact of the malware execution against the physical system. We simulate physical systems and electricity network services in our testing environment using the modules we designed for *NEFICS* to determine how the malware can affect the targeted infrastructure.

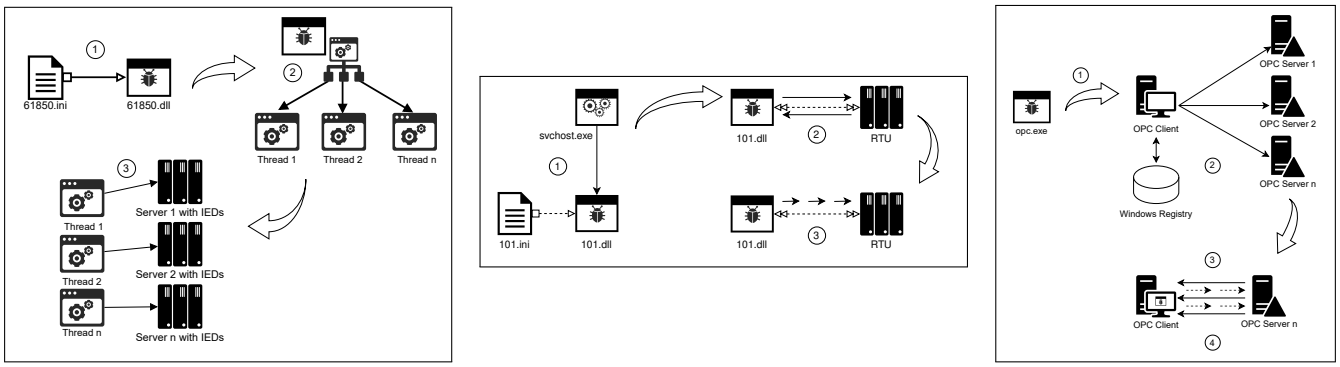
4.2. Attack Patterns of Industroyer 1 & 2

Being a modular industrial control malware, Industroyer 1 can carry out multi-pronged and multi-target attacks against different processes and parts of the power grid operation by using four payload modules, each targeting a specific power grid control protocol: IEC 101, IEC 104, IEC 61850, and OPC Data Access (OPC-DA). In contrast, Industroyer 2 is a stand-alone executable that uses a single protocol to perform its attack: IEC 104. Each supported protocol also reveals the potential industrial control targets the attackers could compromise. Using Industroyer 1, an adversary could launch an attack against an OPC-DA server, RTUs via IEC-101 or IEC-104, and substation IEDs using IEC-61850. As for Industroyer 2, the targets are RTUs supporting IEC-104.

Figure 3 shows the overall process each payload executes to accomplish its task. In the next section, we will describe each payload in detail, but here, we briefly overview the attack patterns.

Targets. This is the step where the payloads differed the most. In some cases, the attacker had a lot of information about the specific targets and baked the targets into the executable (e.g., Industroyer 2); in other cases, the malware expected a specific target (e.g., IP addresses, ASDU Common Address, and specific IOAs) but this information was provided via a configuration file read at execution time (e.g., Industroyer 1's IEC 104 payload), finally, in some cases the malware did not have nor expected any information about the target; instead, the malware executed commands to perform a reconnaissance of potential targets (e.g., Industroyer 1's OPC-DA payload).

Blocking. Most payloads prevented a legitimate process from interfering. If the legitimate control process retains a communications channel with the target, it can prevent or override any malicious command. Each payload attempts to avoid this by terminating the legitimate process (e.g., the IEC-104 payload terminates the running SCADA software communication agent) or locking the required resource within the operating system (e.g., the IEC-101 payload blocks all serial ports).



(a) IEC-61850 payload (b) IEC-101 payload (c) OPC-DA payload

(d) IEC-104 payload (e) Industroyer 2 payload

Figure 3: Overall malicious process for each payload

Connection. Most payloads use a TCP-based protocol to send malicious commands to the targets. These payloads create a separate thread to handle each TCP connection (e.g., IEC-104) or occupy the serial port (IEC-101).

Exploit. In this step, the malware sends malicious commands to devices in the power grid. Some payloads send specific commands to each target (e.g., the IEC-61850 payload sends 35 commands to a CSWI). In contrast, others enter an infinite loop in which they periodically send malicious commands (e.g., the IEC-104 payload of Industroyer 1 keeps repeating a configurable sequence of commands forever).

Regardless of the specific payload nuances, the main objective of all payloads is to change the status of specific circuit breakers controlled by RTUs or IEDs. We now discuss the general overview of the payloads.

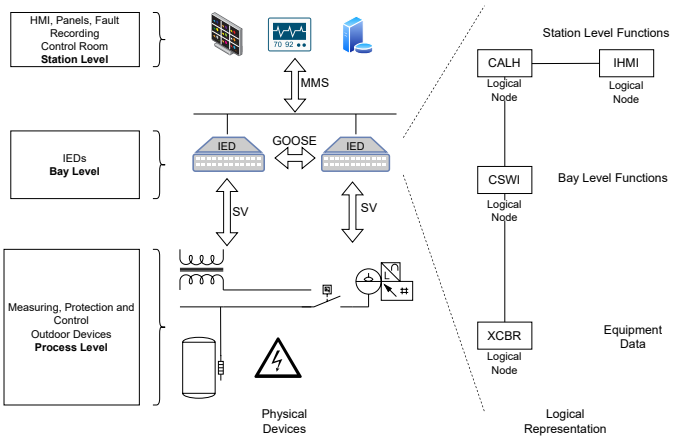


Figure 4: Industrial Network in a Substation.

4.3. IEC 61850 payload

Industroyer 1 targets MMS, a protocol used to connect with IED devices. To observe the malware interacting with its target, we used the iec61850bean open-source server [24] as a target.

Fig. 4 illustrates a general IEC 61850 network in a substation. The 61850 payload has three primary stages, as shown in Fig. 3a:

① *Initialization and configuration:* The payload's entry point may have a configuration (.ini) file with a list of target IP addresses. In this stage, the payload reads the configuration file and identifies targets with open port 102 (MMS service); otherwise, it scans the local network for viable targets.

② *Establishing communication:* In this stage, the payload creates an individual thread for each IP address with an open 102 port to establish the connection and launch the attack. It sends a TCP connection request, followed by a Connection-Oriented Transport Protocol (COTP) request. If the target accepts the COTP request, the malware sends an MMS connection initiation request. The target confirms the

connection initiation by sending an MMS initiation response and establishing a successful connection.

③ *Malicious control loop:* In the final stage, the payload enters into a loop of write requests. The goal of the 61850 payload is to locate the control switches (CSWIs) exposed by the target. The malware targets these switches and changes the values of their control variables.

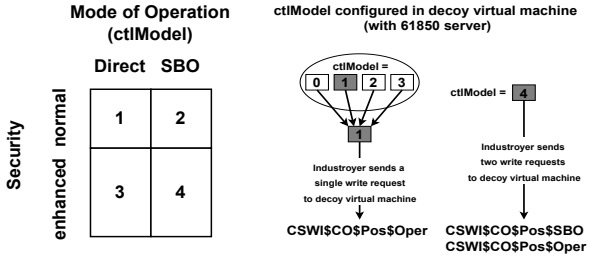
Devices (IEDs) that support IEC 61850 usually come pre-configured with *data models* from the manufacturer. These data models summarize what the device is capable of. For example, if an IED can Protect a circuit from over-current, it will have a logical node named `PTOC` (a logical node beginning with a "P" means it is for protection). Similarly, a device with over-voltage protection will have a logical device named `PTOV`.

Industroyer 1 uses the command `getNameList` to find the logical devices of a given IP address supporting IEC 61850. In particular, we find that Industroyer 1 asks if the device has logical nodes named `CSWI`. This logical node is used for control (logical names starting with "C" means that they are control devices). These control elements can operate circuit breakers (`XCBR`), isolators (`XSWI`), or other process equipment located in a merging unit in the switchyard (the process level in Fig. 4). Logical node names starting with an "X" means that these devices are switchgear–and are located in the switchyard.

In the analysis of the 61850 payload, we observed that Industroyer 1 tries to identify the following CSWI objects:
stVal: This variable indicates the status of the CSWI. It can assume several possible values, among which 0x40 is for position ***OPEN*** and 0x80 for position ***CLOSED***. In our analysis, we found that any value that is not 0x80 is (incorrectly) considered as ***OPEN*** by Industroyer 1.
ctlModel: Defines the mode in which the CSWI operates. In mode 0, it only reports its status; in mode 1, it can be controlled directly by a single command; in mode 2, it is configured with **Select Before Operate (SBO)**, where (for safety reasons) the switch first needs a *select* command indicating the intention to change its status, and only after it receives a second command (operate) it changes its value. In mode 3, it can be controlled by a single command, and the IED sends back a confirmation that the physical device changed its position as requested (or not). Finally, in mode 4, the switch is in SBO mode, and in addition, it sends the confirmation command. These modes are summarized in Figure 5a. We configured the IED server to reply with all of these possible values but found that Industroyer 1 considered `ctlModel` replies of 0, 1, 2, or 3 as if the server was configured with a `ctlModel` of 1, as illustrated in Figure 5b (i.e., Industroyer 1 sends an Operate command without a Select beforehand). This is one of the many ways Industroyer 1 deviates from the standards, as we will show later in the paper.

After the initial reconnaissance is completed, Industroyer 1 delivers its malicious commands by sending ***write requests*** based on the `ctlModel` value. As illustrated in Fig. 5b, if `ctlModel=4`, then it sends two write requests: (1) **SBO** to select the device before sending Operate command, and (2) **Oper** to change the control value of CSWI. For any other value of `ctlModel`, it sends only one write command, **Oper** (which, as we mentioned before, violates the standard). We infer that the machine targeted by Industroyer 1 was only either in a `ctlModel` mode of 1 or 4, and that is the reason the malware developers did not create an accurate payload for the reply if the server replies with a `ctlModel` of 0, 2, or 3.

Another interesting observation is that the malware sends a sequence of 36 commands, each time alternating the status of the circuit breaker. This repeated opening and closing of the circuit breaker may be meant to damage the circuit breaker or associated device (in addition to trying to create a blackout).



(a) CSWI Operation Modes (b) Requests from Industroyer 1

Figure 5: Industroyer 1 sends same write requests for `ctlModel` = 0, 1, 2, 3. Only two distinct modes (1 and 4) of operation by malware.

Fig. 6 shows the logic of Industroyer 1 for changing the status of a circuit breaker. The first step is to ***get details of the target:*** after Industroyer 1 sends the first read request `getNameList`, the server gives the list of logical devices (IEDs) in the server. The second read request is logical device-specific, to obtain the list of logical nodes (Control switches, Circuit Breakers, etc.) in the specified logical device. The server responds with all the logical nodes and their corresponding data attributes. Then Industroyer 1 scans for the CSWI objects described above. If the variables `stVal` and `ctlModel` are not present, then it closes the connection. Industroyer 1 sends read requests to get these values if these variables are present. The response from the server may be 0x40 or 0x80 for `stVal`, which represents that the switch is either OPEN or CLOSED, respectively. For `ctlModel`, the server can respond with either value from $0-4$.

Experiments with IEDs: We now test how Industroyer 1 interacts with real-world devices.

Fig. 7 shows the devices used: *Siemens Siprotec 4 7sj82* (Fig. 7a); *Siemens Siprotec 5 6MU85* (Fig. 7d left); *ABB ref 620* (Fig. 7d right); and a Raspberri PI running the LibIEC61850 library (Fig. 7e). Figures 7b and 7c show the light indicator and the single-line diagram graphical representation of the protection mechanisms (circuit breakers and isolators).

While performing tests against different IEDs, we noticed the brittleness of Industroyer 1 toward the data model provided by an IED. Attempts against the Siemens Siprotec 4 family yielded mixed results depending on the testing mode setting. Whenever the IED is under regular operation, Industroyer 1 executes the attack without affecting the system's physical state, sending the commands in testing mode. However, if the device operates in testing mode, Industroyer

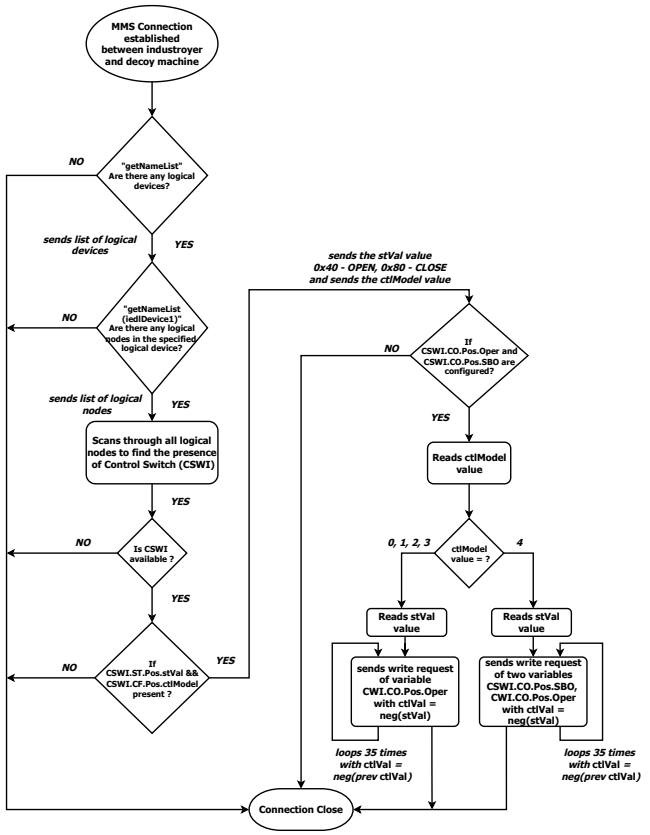
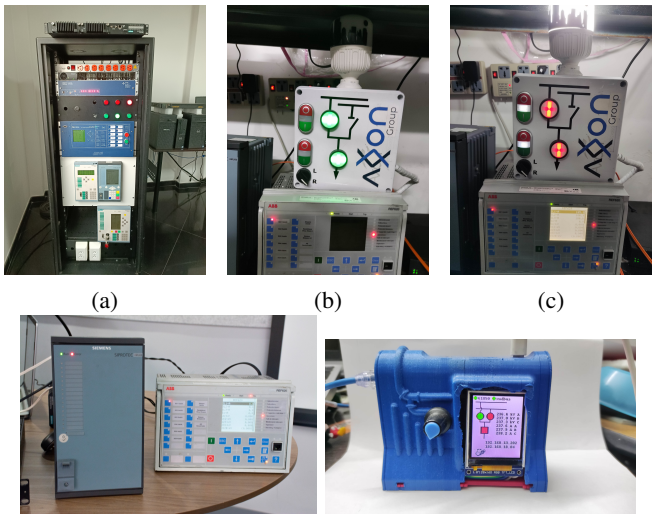


Figure 6: Flowchart of execution of 61850 payload. Industroyer 1 sends requests. Decoy machine with 61850 server sends responses.



(a) (b) (c)

(d) (e)

Figure 7: IEDs and Testbed

1 fails to interpret the device's data model and does not launch the attack. As for the Siprotec 5 family, it always fails to interpret the data model. Overall, our observations on SIEMENS devices reveal that Industroyer 1 receives the Logical Devices information but fails to inspect the respective logical devices; the malware creates a `GetNameList` command of a logical device with a blank name.

That was not the case for the utilized ABB device: when we chose the Testing Mode, the malware was able to successfully modify the positions of both Circuit Breaker and Isolator as shown in figures 7b and 7c, effectively compromising any physical state controlled by the target device. This change is because it sends the command to the control switch in the IED, which is the logical node responsible for actuating upon both the breakers and isolators in these devices.

We confirmed these results with physical and simulated devices using the same data model in both scenarios. Industroyer 1 behaved similarly whenever it tried to attack a simulated version of the data model of the Siemens and ABB devices. This behavior leads to the conclusion that the malware was tailored-made to attack a specific data model of certain manufacturers using testing mode (this was either because Industroyer 1 was configured to attack devices in a lab or because the target substation had left a device operating in testing mode and the adversaries knew about this). Our analysis confirms that the targets of Industroyer 1 were IEDs manufactured by ABB, as the malware did not work reliably with other manufacturers.

4.4. IEC-101 Payload

The IEC-101 payload communicates with the target RTU via a serial port in the compromised host. As shown in Fig. 3b, the overall process has three primary stages: ① Initialization and configuration, ② Establishing the communication and resetting the serial link, and ③ malicious control loop.

This payload requires a configuration file containing the serial port to use, the remaining serial ports to occupy, the name of the legitimate process, the RTUs' Common Address, and the target IOA ranges. In the first execution stage, the payload reads the configuration file, terminates the legitimate process, and occupies the configured serial ports.

The second stage involves establishing a proper connection with the target RTU. The payload sends a "Reset link" command encoded in FT 1.2 format to the RTU through the serial port, re-establishing a clean communications channel with the target and resetting the send-receive counters related to the IEC-101 protocol.

In the final stage, the payload enters an infinite loop, periodically alternating between sending a single (C_SC_NA_1) and a double (C_DC_NA_1) command with the 'OFF' value. This payload follows the Select-Before-Operate paradigm by sending two frames per command, one for selecting the information object and the other to execute the command.

It then continuously sends an 'OFF' value in an attempt to open the circuit breakers in the target substation. For each circuit breaker in the configured IO range, the payload will send a single and a double command, infinitely iterating this range. An educated guess as to why attackers alternate between single and double commands is that they were unsure which type of command the various IOs in the target substations support; thus, they send both.

4.5. IEC-104 Payload

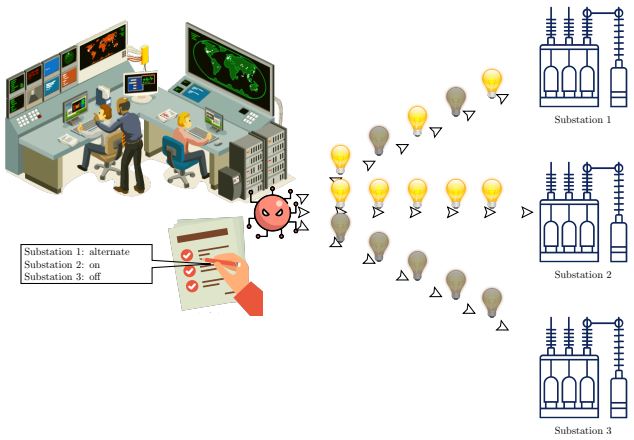


Figure 8: Options for IEC 104 in Industroyer 1.

This payload requires a configuration file containing several target parameters: IP addresses, ASDU Common Addresses, IO Address ranges, TCP port, and timeouts. In addition to these parameters, the attacker can fine-tune the next stage with additional parameters that control the qualifier of the IEC-104 command, whether the malware will alternate the value of each command, and the initial value to use.

In this first execution stage, the payload reads the configuration file and identifies all the target RTUs with their corresponding attack parameters. After this, a second stage involves ending the legitimate connections by terminating the legitimate process, if any. In the third stage, the malware creates a thread for every target RTU, prepared with all the necessary parameters to establish a connection with its target and the corresponding attack. Given the configuration, each thread will vary the actions performed during the malicious loop according to the attack parameters.

Following the attack configuration, the payload enters an infinite loop in the final stage. This payload loosely follows the IEC-104 standard in that it only sends the minimum necessary messages to communicate with the targets without strictly following an expected procedure of a legitimate connection. Moreover, this payload disregards any messages the target sends, continuing the attack even if the target RTU replies with an error frame. The attack itself varies according to how the adversary configured it. It can send the same value (ON or OFF) in every IEC-104 frame or alternate between values. This is the only payload that allows that level of configuration. All other payloads have a fixed set of instructions for the circuit breakers: (send repeated open commands, such as IEC 101, or alternate between open and close 36 times as IEC 61850). With the IEC 104 payload, the attacker can select to send continuous open commands (to keep overriding a manual close command), send repeated close commands, or send an alternating set of commands (open, followed by close) in an infinite loop.

Overall, the IEC-104 payload had the most advanced configuration file. As shown in Fig. 8, the attacker could create different configurations that allowed the malware to interact differently with each device and each substation. For example, it could ask one device in a substation to alternate its status and another in the same (or a different substation) to remain open.

5. OPC Payload

The OPC payload of Industroyer 1 aims to find manipulable devices using known standard COM interfaces and close and immediately open enumerated circuit breakers to cut off power. Based on our analysis, we believe that the malware attacked the OPC client station by initially retrieving the registered OPC servers in its Windows registry. It obtains all items in these servers and finally looks for items with a specific string identifying the targeted circuit breaker. Once it finds those items, it uses the exact commands to select and execute a disconnection from the power grid. As shown in Fig. 3c, the execution of the OPC payload is as follows:

① *Initialization and configuration:* The OPC module's entry points call a single primary function that handles the entire logic of the executable. Without requiring a configuration file as some of the other modules, the OPC's payload primary function starts by enumerating all OPC servers in the Windows registry by looking for the OPC Server 2.0 CATID, a type of globally unique identifier (GUID) that distinguishes between types of interfaces.

② *Establishing communication:* The module uses the Component Object Model (COM) interface `IOPCBrowseServerAddressSpace` to find locally registered OPC servers. The module needs to create an OPC group for each server to interact with and manipulate items. The function invokes the `IOPCServer::AddGroup` method to achieve this. Then, to enumerate the OPC items in each server, Industroyer 1 leverages the `IOPCItemMgt` interface.

③ *Retrieving OPC items:* The OPC module specifically looks for items containing one of the following strings: `ctlSelOn`, `ctlOperOn`, `ctlSelOff`, `ctlOperOff`, and `stVal`. According to ESET [4], these are strings associated with OPC items for *ABB* devices, and as shown later, these items are used as abstractions of the circuit breakers. During the enumeration of servers, the main function logs the name of each server. Assuming all the strings are present, the function will begin by logging the name, quality, and value of `\Pos.stVal`. The OPC item `\Pos.stVal` holds the current status of the circuit breaker, which can have one of four integer values: **0 for intermediate-state, 1 for off, 2 for on, and 3 for bad-state**. The rest of the items are functions that execute specific commands in the physical breaker.

④ *Writing OPC items:* The module uses the `IOPCSyncIO` interface to write `0x01` bytes to `ctlSelOn` and `ctlperOn` to close the circuit breaker the given item refers to. The module then logs the new values and uses the same interface to write `0x01` bytes to `ctlSelOff` and `ctlOperOff`, opening the circuit breakers. The final status after the last write is also retrieved and logged by the payload.

To understand the dynamic behavior of this payload, we simulated an OPC server in our virtual environment with

Matrikon [25]. In particular, we leveraged the Matrikon server to simulate a circuit breaker and the items the malware is trying to retrieve during the reconnaissance stage.

TABLE 3: MatrikonOPC simulation server configuration with fake items and values. Industroyer 1 writes the values of the last 4 items with 1 and logs the value of *stVal* in each step

Item	Value	After ON	After OFF
Pos.stVal	2	2	2
ctlOperOn	0	1	1
ctlOperOff	0	0	1
ctlSelOn	0	1	1
ctlSelOff	0	0	1

In normal scenarios, to close a circuit breaker (thereby activating it), one would have to first select the circuit breaker by sending a 0x01 byte to the ctlSelOn item. Then execute the task by similarly sending a 0x01 byte to the ctlOperOn item. To open a circuit breaker (which would be the objective of a malicious actor trying to cut off power), one would have to send a 0x01 byte to ctlSelOff and ctlOperOff. Industroyer 1 OPC module performs these actions automatically, as seen in Table 3, where the values of ctlOperOn, ctlOperOff, ctlSelOn and ctlSelOff are overwritten to 1.

```
[*ServerName: Matrikon.OPC.Simulation.1*]
 [State: Before]
  OPCItem name : Bucket Brigade.\Pos.stVal
 Quality: 192 value: 2
[*ServerName: Matrikon.OPC.Simulation.1*]
 [State: After ON]
  OPCItem name : Bucket Brigade.\Pos.stVal
 Quality: 192 value: 2  # Should change in real scenario
[*ServerName: Matrikon.OPC.Simulation.1*]
 [State: After OFF]
  OPCItem name : Bucket Brigade.\Pos.stVal
 Quality: 192 value: 2  # Should change in real scenario
Process Close
```

Figure 9: Log output generated by OPC module after executing against our test server *Matrikon.OPC.Simulation.1*

The module writes a log file during its operation as the one shown in Fig. 9, and prints the status of the *stVal* item, which, as we previously mentioned, could represent the resultant circuit breaker status. The payload logs the values of the *stVal* item in different stages of its execution: The initial value is logged under *State: Before*; The second value after setting ctlSelOn and ctlOperOn under *State: After ON*; and the third value after setting ctlSelOff and ctlOperOff under *State: After OFF*. According to our tests, the logging and value writing will still work in case some items are missing; however, we believe that the malware requires all of these items to be present in order to impact the physical circuit breakers.

5.1. Industroyer 2

Unlike its predecessor, Industroyer 2 does not use a custom configuration file through the command line to determine the targets and nature of the attacks. Instead, the malware has hard-coded configurations that specify the different targets. Having the targets hard-coded means the attacker must re-compile the malware whenever they need to change the attack targets. Moreover, since the adversary needs to have this information at compilation time, we assume that the attackers had prior detailed knowledge about the target infrastructure (IP addresses of target RTUs and specific IOA addresses of the circuit breakers).

We summarize the overall process in the same four basic steps, shown in Fig. 3e: ① the malware parses the configuration from the hard-coded strings, ② it kills the legitimate process and any active connections with the targets, ③ it launches an individual thread for each target RTU, and ④ it executes the malicious process over a newly established connection with each target RTU.

Upon execution, the malware first checks the command line arguments to determine the value of two possible arguments: "-t" and "-o".

Including the "-t" flag will set a timed start of the process. If the attacker includes this argument as part of the command line, the malware will take the following token in the command line and use it to determine when the process begins. The minute specified by the parameter determines this time. For instance, if the adversary runs the malware using the value "-t 40", the malware will wait until the next 40th minute in an hour. If the attacker runs the command, say at 02:35, the process will begin at 02:40; if s/he runs it at a minute greater than the specified, it will wait until the next occurrence of the specified minute within the following hour. For instance, if s/he runs it at 02:55, the process will begin at 03:40.

As for the second possible argument, the meaning is relatively straightforward; the "-o" parameter determines the filename of a log file the malware will use to store the output. If this argument is not specified, the malware prints any output on the standard output.

As a result of the static analysis we performed on the sample, we determined that an array of strings comprises this hard-coded configuration, each string containing the necessary values to execute an attack against a targeted RTU. As an example, the configuration string located at offset 0x9818 is u"192.168.122.2 2404 2 0 1 1 PService_PPD.exe 1 \"D:\\OIK\\DevCounter\" 0 1 0 0 1 0 0 8 1104 0 0 0 1 1 1105 0 0 0 1 2 1106 0 0 0 1 3 1107 0 0 0 1 4 1108 0 0 0 1 5 1101 0 0 0 1 6 1102 0 0 0 1 7 1103 0 0 0 1 8 ".

Each configuration string starts with global values regarding the target and the configuration format. The first two values ('192.168.122.2' and '2404'), the IP address, and the TCP port are self-explanatory. We traced the remaining values through the execution path of the binary during our static analysis. The malware uses the third value ('2') while building the IEC-104 frames as the ASDU "common address". The fourth value ('0') determines how the malware parses the configuration string, hence the operation mode.

Depending on the configured operation mode, the malware determines the target circuit breaker set via an IOA range (value '1') or a configured IO target list (value '0'). We found it curious that the subroutine parsing the configuration can customize the attack this way, as the configuration is hard-coded, and the attackers had prior knowledge of the intended targets when they compiled the malware. An

educated guess regarding the decision to hard-code the configuration is that attackers rushed the active usage of the malware due to the rapid evolution of the political and military situation at the time.

Suppose the operation mode is set to an IOA range (fourth value set to '1'). In that case, the malware uses the fifth and sixth values as the first and last "information object addresses" for the IEC-104 frames, iterating between them to generate the necessary frames. Aside from this, the remaining configuration value positions are shifted by 2.

The following value ('1') is a boolean flag that triggers the parsing of 9 additional optional values following this flag. If this flag is '1', the first optional value ('1') seems innocuous, as we found no execution branch associated with that value throughout the sample.

As part of the optional parameters, the following values target the legitimate process. First comes the executable name of the process that the malware will terminate upon execution ('PService_PPD.exe'). Next is a flag that determines whether or not the malware will rename the target executable ('1'), potentially preventing future executions of the legitimate process. Finally, a value specifies the path corresponding to the directory in which the target executable resides ('\"D:\\OIK\\DevCounter\"').

The following four optional values trigger delays in the malware's execution ('0', '1', '0', and '0'). We surmise that these values allow the attackers to tweak the timing of the malicious commands. The value that holds a '1' seems to be a delay between sent frames, according to the execution path we analyzed in the sample. After these timing values, another seemingly innocuous optional value has no use throughout the sample ('1'), concluding the additional optional values.

The next value ('0') determines what value the malware will use as the default value for the malicious commands. The malware uses this value in every IEC-104 information object unless altered by a specific flag. After this value, a flag ('0') triggers an additional IEC-104 command with an inverted default value by default, resulting in two commands per target information object.

The malware parses the remaining parameters only if the operation mode is '0'. The next and final global value is the target information object amount ('8'). After this, a set of sub-configurations corresponds to each target information object. As an example, the first target configuration is as follows: `1104 0 0 0 1 1`.

The first value ('1104') corresponds to the information object address the malware will add to the malicious ASDU. The second value ('0') determines whether the malware will build an IEC-104 single command (C_SC_NA_1, if the value is '1') or a double command (C_DC_NA_1 if the value is '0'). The next value ('0') determines if the malware will send one ('execute') or two ('select' and 'execute') frames. The next value ('0') determines if the malware will invert the default state from the global configuration. The next value ('1') seems to alter the order in which the malware sends the frames, prioritizing the targets. Finally, the last value ('1') is the current index of the target within the set.

In the sample we analyzed, the malware targets three IP addresses, each with a specific list of targeted information objects (target circuit breakers). The target systems exist within at least two subnetworks, one of which could be a subnetwork with a 23-bit prefix (Fig. 10).

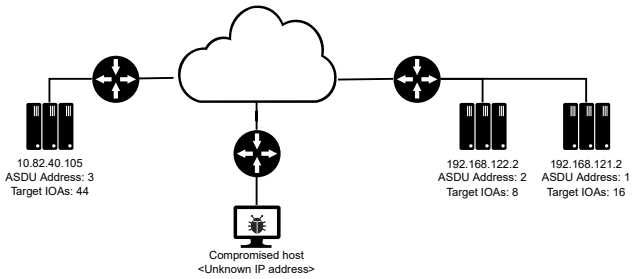


Figure 10: Targets of Industroyer 2.

Table 4 reveals the specific targets. We infer the attacker targeted three RTUs (presumably three different substations). In the first substation, it attempted to open 16 circuit breakers; in the second substation, it attempted to open 8 circuit breakers; and in the third substation (the largest substation with 44 circuit breakers), it attempted to open 28 circuit breakers, and then it attempted to change the status of 16 circuit breakers from open and then to close. Opening this large amount of circuit breakers in three different transmission substations can significantly negatively impact the power grid.

TABLE 4: Industroyer 2 configured targets

Target IP	ASDU Address	Default command	Additional inverted	Target IOAs	Single Command	Double Command	Select Before Operate	Invert default
192.168.121.2	1	0	0	16	0	16	0	0
192.168.122.2	2	0	0	8	0	8	0	0
10.82.40.105	3	0	0	44	28	16	0	28

Notice that the attacks included always disconnecting the circuit breaker, always connecting the circuit breaker, and connecting and disconnecting repeatedly. This last attack appears similar to the Aurora generator attack [26], where connect and disconnects can be used to apply maximum torque.

5.2. Post blackout

Industroyer 1 and 2 also attempted to wipe out systems after launching an attack opening circuit breakers. This could lead to longer blackouts that are harder to recover because the machines that could be used to recover the electrical grid would be wiped out and unbootable.

The entry point for the wiper in Industroyer 1 is called Crash, and it calls three subroutines. One subroutine tries to make the system unbootable by writing to the ImagePath values, the other attempts to delete critical files and the third subroutine terminates all processes and crashes the system.

TABLE 5: Attack target configuration.

	Autonomous	Optional	Required	Hard-coded
OPC	●			
IEC-61850	●	●		
IEC-101			●	
IEC-104			●	
Industroyer 2				●

For our OT analysis, we noticed that subroutine 2 attempts to delete the following file extensions: `.scl`, `.cid`, and `.scd`. These are substation configuration files: `.scl` files define the functions of IEC 61850 devices, interfaces, and systems; `.cid` is a configured IED description, and `.scd` is a Substation Configuration Description file. Deleting these files from a substation can mean that even if the devices are powered on again, the configuration and the rules for how they should interact together would be lost. This would prevent a quick recovery of a substation and make the remote operation of the breakers impossible, requiring manual reconnections.

5.3. Summary

Table 5 summarizes how the malware targets devices. This process goes from a fully automated attack to a hard-coded attack defined within the binary.

Autonomous: Both the OPC and the IEC-61850 payload can autonomously acquire their targets and begin the attack. In the case of the OPC payload, it will examine the infected host's registry for a server and then request the potential targets. As for IEC-61850, if it does not have a configuration file, it will scan the broadcast domain to find potential targets.

Optional: The IEC-61850 payload can discover targets automatically, but it can also be executed with a configuration file. The configuration file is simple in this case, as it only has a list of target IP addresses.

Required: Both IEC-101 and IEC-104 payloads require a configuration file. While the formats vary, the contents are relatively similar because the variables define the attack's behavior. Each configuration file holds variables defining the legitimate process to kill, the communication mechanism (e.g., serial ports or target IP address), protocol requirements (e.g., ASDU Common Address), and attack definition (e.g., IOAs or command patterns).

Hard-coded: Industroyer 2 has the configuration hard-coded in the binary. It defines a structure similar to the original IEC-104 in that it also has some general variables determining the communication, variables involving the legitimate process, and variables defining the particular targets and attack method.

Once it connected to each target device, it attempted to change the status of circuit breakers. The options can be to open, close, or alternate between opening and closing them.

The timing of these commands is different among different payloads. We look at the frequency at which the different payloads send commands toward their targets. Since the primary objective is to disrupt operations, the malware attempted to send commands fast enough to prevent a response by human operators. While this behavior makes sense for an attack, it is also a giveaway for any intrusion detection system.

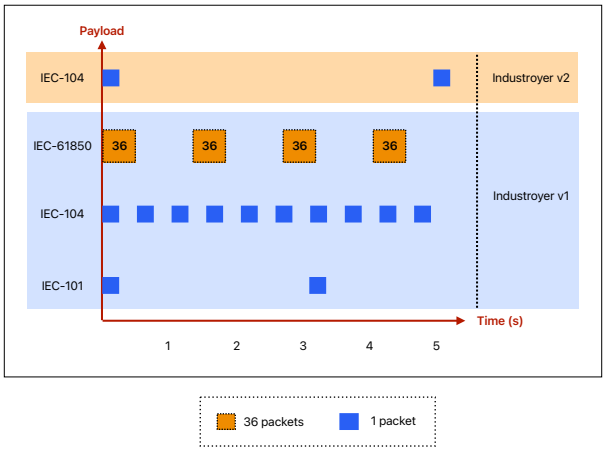


Figure 11: Average delay between commands, frames, or packets.

We executed every payload in the sandbox and took samples of the traffic we could capture or log. Within each traffic capture, we isolated the malicious commands sent by the payloads and measured the time between commands during an attack. Figure 11 presents the average delay between commands for every payload.

These average delays provide some insights regarding the intended target. For instance, the IEC-101 protocol travels via serial port at a 9600 baud rate towards a remote substation with limited bandwidth. It seems the attackers considered this limitation, as the payload is the slowest within the framework. In contrast, the IEC-61850 payload is the only attack meant to be launched over a LAN; hence, it is the fastest payload.

The most curious behavior, however, is how attackers evolved the delay between IEC-104 commands. The Industroyer 1 IEC-104 payload sent commands with a reasonable delay for its purposes, as it intended to contact remote substations via some private WAN. Nonetheless, when it comes to Industroyer 2, the attackers decided to increase this delay by ten times, making it the most conservative payload, regardless of the available bandwidth. This may have been a strategy to avoid triggering anomaly detection systems.

Industroyer 1 also had several non-standard behaviors. We will discuss in the next section how it did not follow the IEC 104 standard, but here we summarize some of the unusual behavior of the 61850 payload: (1) it did not interpret correctly Control Switch StVal or CtlModel values, (2) it send a fast alternating sequence of 36 commands to modify the position status of a circuit breaker may have been intended to cause not only a blackout but also equipment damage, (3) it was targeting devices on test mode. Despite all of this unusual behavior, it still manages to affect the intended ABB devices (but not the other devices we tested). So, while the malware can have general payloads that could

potentially be used against other targets, it appears it was only tested in equipment the attackers knew was used by the victim.

From a network security perspective, the nonstandard behavior of Industroyer 1 can be detected by various means, from unusual rates of the traffic being generated to detecting state transitions not allowed in the protocol standard. For example, to detect Industroyer's behavior on a network, we can use Snort IDS to log every command sent using Snort's inspector modules (e.g., "iec104"), and alter the `log` action to `alert` whenever an endpoint sends several commands in a short period of time with rate filters.

```
iec104 = { }
binder = {
 {
  when = { proto = 'tcp', ports = '2404'},
  use = { type = 'iec104' },
 },
}

log tcp any any -> any 2404 (
 msg:"IEC-104 Single Command";
 sid:1000001;
 rev:1;
 priority:10;
 service:iec104;
 iec104_asdu_func:c_sc_na_1;
);

rate_filter gen_id 1, sig_id 1000001, \
 track by_src, count 2, seconds 10, \
 new_action alert, timeout 1800
```

We can create similar anomaly detection rules for other payloads in Industroyer 1. Detecting Industroyer 2 would require going beyond protocol anomalies and studying the payload's semantics. For example, for Industroyer 2, one anomaly would be: Does it make sense to activate so many circuit breakers simultaneously in three different substations? These rules must balance operational needs (operators isolating parts of the grid for maintenance) vs. security risks.

6. Discussion

6.1. Malware Evolution

Evolution towards Industroyer 2: Industroyer 1 does not follow the standard in many ways. Not only did it disregard any incoming packets, but it also did not behave as a legitimate client. Industroyer 2 fixed these issues by following the correct behavior of the standard. While Industroyer 1 can be detected by monitoring the expected protocol behavior, Industroyer 2 would bypass this anomaly detection check.

Fig. 12 shows the expected behavior of a controlling station in IEC 104. Under normal circumstances, the controlling station would establish a TCP connection with the RTU, poll for the existing devices with an interrogation command, and send keep-alive frames to maintain the connection with the controlled station while receiving measurements.

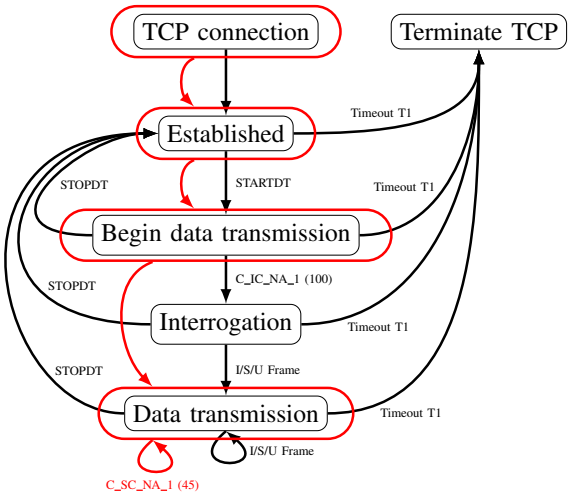


Figure 12: IEC-104 connection behavior. Industroyer 1 (red) does not follow the expected behavior of a legitimate connection (black).

The behavior of the IEC-104 payload of Industroyer 1 varies from the behavior of its successor. Industroyer 1 skips the interrogation stage, disregards any incoming frames and begins to send commands after starting the data transmission at a rate of 2 frames/s in an infinite loop for as long as the process executes. Industroyer 2, however, follows the standard more closely; not only does it verify every received frame as a valid IEC-104 frame, but it follows the legitimate control flow of sending an interrogation command prior to any other interaction with the controlled station after the data transmission begins, sending commands at a rate of 0.2 frames/s. Also, Industroyer 2 does not execute indefinitely like the 104 and 101 payloads (or 36 times in the case of the 61850 payload); it sends one (or two) commands to the configured targets and gracefully terminates the connection with the controlled station after sending the malicious commands.

Third Blackout: A report that was published after this paper was accepted [27] details yet another blackout in Ukraine, on October 2022, caused by a combination of cyber attacks and physical attacks (missile strikes targeting critical infrastructure across the country).

Instead of deploying their own custom malware, the attackers used their access to the SCADA software (a MicroSCADA server capable of directly sending IEC-104 commands to the target RTUs) by running a script to be interpreted by the MicroSCADA server. This change creates more stealthy attacks, as the commands to open circuit breakers come from a legitimate process, not a piece of malware. The attack consists of a script that runs a Supervisory Control Implementation Language (SCIL) executable provided by MicroSCADA to interpret the contents of a text file (which was unavailable during their analysis), presumably holding the SCIL commands.

While the method is different, our sandbox can also be used to analyze this attack. We would need to have the infected virtual machine running MicroSCADA. The MicroSCADA instance should have the necessary privi-

leges to interact with the simulated RTUs or IEDs in the sandbox. Assuming we acquire the malware samples and the specific SCIL commands the attackers used (currently, these commands are unknown), we could run the malicious script described in Mandiant's report in the infected virtual machine and then analyze the commands sent to the targeted RTUs or IEDs.

Takeaways: Looking at the evolution of these attacks, we see Sandworm constantly changing tactics in attempts to trip devices. First, Industroyer 1 was a Swiss army knife to attack the power grid. It was developed without any specific target, and it could deal with legacy connections (IEC 101), modern connections (IEC 104), and interoperability systems (OPC). In addition to working on SCADA computers in a central control room, Industroyer 1 could even be deployed in the computers at substations (IEC 61850). Second, Industroyer 1 was highly configurable. It expected configuration files telling it how and who to attack. It also attempted a limited reconnaissance of potential victim devices. Finally, it had several bugs; most notably, it did not follow the industrial protocol specifications, and its success may have depended on how tolerant victim devices were to non-standard industrial protocols.

Because Industroyer 1 was so adaptive, without the configuration files, we do not know the targets or the attacks launched (did they open and close the circuit breakers? Only some of them? Open and then close? etc.) In contrast, Industroyer 2 was a targeted strike; it did not read any configuration files, so attackers knew how and who to attack before deploying the malware. It also only targeted one industrial protocol (instead of four). Industroyer 2 was also fairly polished, following the expected behavior defined by the industrial protocol standards more precisely. These changes might mean that the attackers did not want to share all their capabilities with the forensic teams and that the malware was now polished enough to interact with most devices without errors. As time passes, attack tools may become more polished, and the teams using them will only use the pieces necessary for the attacks.

The most recent attack on MicroSCADA shows a paradigm shift. After several attacks, Ukrainian power utilities may develop stronger attack detection tools, and they can identify malicious processes or that the provenance of a control command is not the SCADA. Future defenses may prevent the execution of remote control commands originating from a script.

6.2. Attack Surface

All known attacks targeting the Ukrainian power grid compromised a Windows computer. This computer was either in a substation or the control room, both shown as ① in Fig. 13. However, these are not the only possible targets; there are several other embedded computers in the power grid, including RTUs (②) and IEDs (③). Any of these devices can also send commands to electrical equipment.

As the latest attack on MicroSCADA shows, attackers may be trying to become more stealthy, and attacking an embedded device with limited anti-malware support may be more attractive to future attackers. Furthermore, attacking an RTU or an IED might give attackers the same level of stealthiness as Stuxnet, as these embedded devices can send malicious control commands to electrical equipment while reporting back to the Human Machine Interface (HMI) that everything is working under normal operating conditions. While remaining stealthy might be difficult in an interconnected power system where attacks propagate and become visible to other power utilities, sabotaging specific devices (e.g., the Aurora attack) might be easier to keep under the radar.

The disadvantage of targeting these embedded devices is that the reach of the attack may be limited. By compromising a central control room, the attacker can reach multiple substations, while by compromising an IED or an RTU, the device can only send control commands to the substation where they are located. Then again, if the defender only focuses on protecting Windows desktops, they may motivate the attacker to jump to RTUs or IEDs.

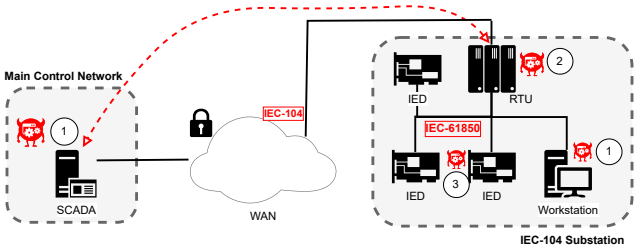


Figure 13: Attack surfaces.

6.3. Impact on the Power Grid

So far, we have analyzed the commands sent to circuit breakers in a substation, but we have not discussed how opening circuit breakers may cause a blackout. Furthermore, the impact of blackouts can be different depending on the magnitude of the attack; disconnecting a distribution system will keep the bulk of the power grid running, and reinstating power in the affected area will only require reconnecting the affected area to the bulk system. On the other hand, an attack that brings down the interconnected bulk system will affect a large geographical area (e.g., a country), and restoring power to all consumers can take several days.

In this subsection, we want to highlight that the impact of future malware attacks on the power grid will depend on (1) which devices the attacker can target and (2) how many devices the attacker can trip. Transmission owners operate different parts of the North American Bulk power grid; these include the American Transmission Company, Xcel Energy, Entenergy Arkansas, LLC, Wolverine Power Cooperative, etc. Each of these companies manages different portions and sizes of the power grid; for example, Xcel Energy manages 1,200 substations and over 20,000 miles of transmission lines, while the American Transmission Company manages 582 substations and over 10,081 miles. Attacking different companies may produce different results.

We leverage a high-fidelity cascade analysis tool we have used in previous work [28], [29], [30] and recontextualize our work with our new analysis. Our model considers a large-scale North American regional interconnection system

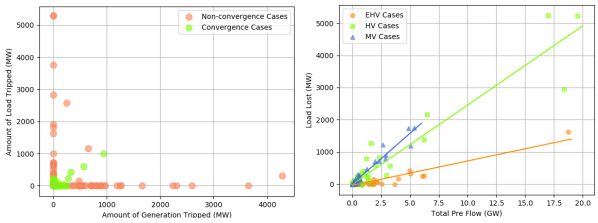
with over 5,000 buses. This model approximates a real-world system in North America operated by different power companies; in particular, each element in our dataset (e.g., substations, lines) has a tag that identifies the transmission owner of the element. This allows us to partition our power grid model into areas operated by different companies, which, in turn, allows us to model the case where malware like Industroyer can infiltrate the control room (or substation) of one of these transmission companies. As mentioned in Section 2, we call these companies TDSPs, and we consider an attacker that has access to one (or more) control rooms in that TDSP and can send commands for opening all or a specific subset of these devices.

Because the attack of Industroyer 2 was hard-coded, we know it may have been attacking three substations and opening 68 circuit breakers (and then closing 28 of them). A circuit breaker can de-energize various devices, including generators, loads, lines, transformers, and capacitors, and it is not clear which of these were attacked by Industroyer 1 and 2.

Our simulations show that a naive attacker opening all circuit breakers might not cause a blackout. Fig. 14a shows some cases under 1,000 MW (in generation or load tripped) where the points along the diagonal (in green) with the same amount of generation and load removed will balance out and will not cause a blackout. Thus, if the adversary targets a TDSP managing only a limited amount of power, the power imbalance caused by the initial contingency is critical to maximizing the chances of a system collapse. The attacker needs to know precisely which circuit breakers will disconnect a generator and which will disconnect a load to maximize the chances of a system-wide blackout. Our results also show that if an adversary prioritizes which TDSP to compromise, and the load imbalance of disconnecting all devices in both cases is the same, attacking the TDSP with more generation and disconnect generators will have a greater effect than disconnecting the load.

The types of lines disconnected can also affect the outcome of the attack. There are three main voltage classes: Extra High Voltage (EHV) which covers anything over 500kV; High Voltage (HV), ranging from 100kV to 500kV; and Medium Voltage (MV), ranging from 1kV to 100KV. A naive attacker might think disconnecting the EHV, and then the HV lines will maximize the chances of causing a load imbalance. In contrast, Fig. 14b shows that attacks on medium-voltage and high-voltage lines are more effective than those on Extra high-voltage lines if we want to cause load shedding. The reason for this is that EHV lines are highly interconnected networks, and the remaining EHV lines can easily handle the redistributed flows; on the other hand, MV lines are closer to the customers, and they are less redundant, so if you take just a couple of them, the consumer will not be connected to the electricity system. Having said that, attacking EHV lines is more likely to result in a complete system collapse, showing that load imbalance is not the only metric for a successful attack. In our simulations, attacking only 25 EHV lines can bring a complete system collapse (non-convergence case), and attacking more than 100 is a guaranteed system collapse.

If the attacker wants to be completely agnostic to the devices it is operating, our results show that if the attacker targets a company that operates more than 200 devices (and the attacker can trip them all), then the attack will most likely result in a complete system collapse; however, if the number is closer to 100 or less, it is more likely that the bulk system will survive the attack.



(a) Generation vs. Load. (b) Losses per line attacked.

Figure 14: Industroyer-like Attacks to the Power Grid.

In summary, we can see that different types of disconnection attacks can have different results in the power grid. Attacking the largest TDSP or targeting the largest number of EHV lines can cause system-wide blackouts. In contrast, if the adversary cannot bring the system down but wants to maximize the local blackouts (load shedding), it should target the largest number of MV lines it can. Finally, an attacker that wants to be strategic in attempting to create a systemwide blackout should not simply open any line, but (if possible) it should attempt to create the largest imbalance between generation and load. Even more, while an imbalance in either direction (more generation than load or more load than generation) can create system-wide blackouts, our results indicate that cutting generation will create higher chances of a systemwide collapse than targeting only loads.

From a defense perspective, a single compromised control room can take out as many devices as it has access to. We need to perform risk assessments that assume a control room has been compromised and then design alternatives (e.g., divide control of substations into smaller and different control rooms with a separation of privilege) or identify and respond to attacks (identify anomalous patterns of control commands sent to substations).

7. Related Work

The most well-known malware attacks to control systems are (1) Stuxnet [31], the first documented malware targeting industrial systems; (2) Industroyer, the first malware targeting the power grid and causing a blackout; and (3) Triton [32] (also known as TRYSIS) the first malware targeting the equipment that protects the safety of industrial systems. While each of these attacks has created physical damage (Stuxnet damaged centrifuges, Industroyer 1 created blackouts, and Triton shut down a petrochemical facility), they are very different in their targets and methods. Stuxnet and Triton are malware designed for embedded control systems called Programmable Logic Controllers (PLCs), while Industroyer interacts with RTUs and IEDs.

Academic research has also looked at the security of PLCs. There are several efforts to analyze the programs of PLCs [33], maintain the control flow integrity of PLC programs [34], perform PLC forensics [35] or reverse-engineer

PLC binaries [36]. There are new threat analysis tools that identify what attackers can do with a PLC program [37], PLC malware with basic physics of a process [38], or attacks exploring how to identify the effort to compile a PLC payload [39] or to automate PLC payloads [40]. Similarly, other efforts have examined static code analysis of robotic languages [41]. Our efforts complement these other papers, as we focus on malware that interacts with IEDs or RTUs using network protocols for the power grid, as opposed to focusing on PLCs.

In addition, we look at the malware's impact on real-world devices and the power grid at large. Other papers have shown the importance of understanding the operation of the power grid under attack [42]. One of the popular methods to understand the impact of attacks is using digital twins [43], [44]: high-fidelity software simulations often interfacing with real-world hardware. While there is an active community working on the security of the power grid [45], as far as we are aware, this is the first paper that looks jointly at program analysis of real-world malware samples against the power grid, their impact on several real-world IED devices, and study also the impact against a high-fidelity model of a large-scale power grid.

Reports on Industroyer: After the attack against the Ukrainian power grid in 2016, several security companies presented their findings regarding this malware [4], [5], [7], [8]. These reports discuss the malware's inner workings and primary goal, focusing on binary analysis, configuration files, payloads, and network protocols. Similar reports emerged addressing Industroyer 2 [9], [10], [11], [12]. While these reports present an overall analysis of the malware (Table 6), they do not consider the surrounding context of the attack. Instead, theirs is a more practical approach to detect the malware and improve existing commercial products to identify the relevant signatures. These reports fail to combine their findings with the relevant context into a comprehensive analysis of the malware that goes beyond a standard malware analysis and considers the challenges of dynamic analysis of malware with different libraries and devices, as well as using high-fidelity models of a power grid to understand the impact that the malware can have.

In particular, no other report has run dynamic analysis tests for all the industrial protocols targeted by the malware. For example, by using real-world IEDs, we discovered that Industroyer 1 could change not only the status of circuit breakers but also the status of isolators; we also discovered the brittleness of Industroyer 1 to specific IED models from different brands. These facts were not mentioned in any of the previous white papers.

To the best of our knowledge, we present the first in-depth study discussing the underlying inner workings of the malicious payloads and the ramifications of these attacks on the targets. Moreover, we discuss the adversaries' design decisions while developing these malware artifacts and the practicality of using one payload versus another.

State-Sponsored Operations: There are three well-known examples of malware with industrial protocol payloads causing real-world physical damage: Stuxnet, Industroyer, and Triton. While Stuxnet has been attributed to a collaboration between the US and Israel, the remaining two malware attacks have been attributed to two different

TABLE 6: Reports on Industroyer 1 or Industroyer 2.

Analysis	[4]	[5]	[7]	[8]	[9]	[10]	[11]	[12]	Ours
Static analysis	●	◐	●	○	●	◐	●	◐	●
Dynamic analysis	◐	○	○	○	◐	◐	●	●	●
Malware capabilities	●	●	●	◐	◐	◐	●	●	●
Malware configuration	●	●	○	○	◐	◐	●	●	●
Malware design discussion	○	○	◐	◐	○	○	◐	○	●
Attack stages	○	◐	◐	●	◐	○	◐	◐	●
Attack's ICS impact	◐	◐	○	●	○	○	○	◐	●
High-fidelity cascading analysis	○	○	○	◐	○	○	○	○	●
Attack timing analysis	○	○	○	○	○	○	○	○	●
Tests with real devices	○	○	○	○	○	○	○	○	●
Comparison of Industroyer 1 & 2	–	–	–	–	◐	◐	○	◐	●

○: no mention ◐: partially discussed ●: thoroughly analyzed –: not applicable

entities within the Ministry of Defense of Russia: the GRU and TsNIIKhM. TsNIIKhM, a subcontractor for the GRU, is believed to be the main entity behind Triton [46].

The GRU, on the other hand, has two main active groups: *Sandworm* and *Fancy Bear* [47]. Combined, these two groups are believed to be behind some of the most infamous cyber attacks in the last decade, including the cyber attacks against Ukraine's election in 2014, the US election in 2016, and the attacks against the power grid in Ukraine in 2015 and 2016 (Industroyer) [48]. The *Five Eyes* (the intelligence agencies of Australia, Canada, New Zealand, the UK, and the US) also believe they are behind NotPetya, the ransomware that was described by the White House in 2017 as *the most destructive and costly cyber-attack in history* and *part of the Kremlin's ongoing effort to destabilize Ukraine.* [49].

The 2022 Russian invasion of Ukraine has also been accompanied by various cyber operations [50], [51], [52], [53]. The Russian government has increasingly resorted to targeting Ukraine's power grid with a massive bombardment of power facilities, destroying more than 40% of the country's energy infrastructure [54]. These attacks intensified in the winter months, similar to when the cyber attacks against Ukraine's power grid happened in 2015 and 2016. The most recent cyber attack against the power grid, reported in 2023 [27], aligned with kinetic attacks against substations.

War is an act of force (physical harm or intimidation) for political purposes to motivate the enemy to do the attacker's will. Most nations who rely on their military for political purposes now consider cyberspace as an official theater of conflict (in addition to land, air, sea, and space), and this poses new societal threats as our dependence on computer-controlled physical infrastructures is targeted by sophisticated adversaries. We argue that we must be better prepared to analyze and respond to future cyber-attacks on critical infrastructure systems, given this new reality.

8. Conclusions

In this paper, we designed and tested a new industrial sandbox that enabled us to provide the first in-depth analysis of the only known malware family to attack a power grid. We believe our work fills the gap between industrial reports on Industroyer and the technical details that researchers need to understand the malware and how future attack vectors may develop and impact equipment and the grid. For example, our finding regarding the specificity of the IEC-

61850 payload affecting ABB devices in "testing" mode, is something not visible through static analysis only.

Through our analysis, we uncovered some of the implementation decisions the adversaries made during the development of the malware. For instance, our timing analysis revealed they fine-tuned their attack against the specific network target. Moreover, these timings consider the overall network delay and the time it takes to physically open or close a target circuit breaker. Furthermore, the fact that they considered these timings and the ability to tweak some of the payloads via configuration files suggests that they also considered the possibility of damaging the devices through different attack patterns.

As for the duration of the attacks, the way they coded the payloads provided some clues of a long-standing network intrusion. For both cases of IEC-104 usage, the adversaries had to obtain relevant IP addresses, common addresses, and information object addresses corresponding to the targeted substations and circuit breakers. Regarding IEC-61850, the fixed "invoke ID," and the hard-coded bytes within the binary suggest that the attackers had traffic captures of working substations and gathered the necessary commands from these captures. Access to these captures reveals potential device manufacturers via MAC addresses and specific configuration settings such as the "test mode" of the target devices.

Among future protections, 61850 allows operators to protect the data models, such as allowing reconnaissance commands (e.g., getNameList) to be executed only by specific devices or via password protection.

While it was not necessary for Industroyer 1 & 2, future malware attacking the grid may only execute in the right environment. To prepare for this, we plan to continue working on our framework and extend it with the use of symbolic execution. We also plan to adapt more protocols popular in the U.S., like DNP3.

Acknowledgements

We want to thank Kelvin Mai for the initial analysis of Industroyer 1's 104 payload and for finding the entry point in the binary. We also thank Dominic Lucchesi and Aviv Brook for their help in analyzing the OPC payload. This work was partially supported by NSF awards 1931573 and 1929410. The work of the fifth author was partially supported by the INTERSECT project, Grant No. NWA.1162.18.301, funded by Netherlands Organisation for Scientific Research (NWO). Any opinions, findings, conclusions, or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of NWO.

References

[1] K. Zetter. (2016, 3) Inside the cunning, unprecedented hack of Ukraine's power grid. [Online]. Available: https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/

[2] A. Greenberg. (2017, 6) Crash Override: The malware that took down a power grid. [Online]. Available: https://www.wired.com/story/crash-override-malware/

[3] P. Polityuk, O. Vukmanovic, and S. Jewkes. (2017, 1) Ukraine's power outage was a cyber attack: Ukrenergo. [Online]. Available: https://www.reuters.com/article/us-ukraine-cyber-attack-energy-idUSKBN1521BA

[4] A. Cherepanov, "Win32/industroyer a new threat for industrial control systems," 6 2017. [Online]. Available: https://www.welivesecurity.com/wp-content/uploads/2017/06/Win32_Industroyer.pdf

[5] Dragos, Inc., "Crashoverride: Analysis of the threat to electric grid operations," 6 2017. [Online]. Available: https://www.dragos.com/wp-content/uploads/CrashOverride-01.pdf

[6] M. Lesser, S. Fellows, and O. Cox, "Defending operational technology (OT) in kinetic, cyber, and hybrid warfare."

[7] J. Slowik, "Anatomy of an attack: detecting and defeating crashoverride," 2018. [Online]. Available: https://www.virusbulletin.com/virusbulletin/2019/03/vb2018-paper-anatomy-attack-detecting-and-defeating-crashoverride/

[8] ——, "Crashoverride: Reassessing the 2016 Ukraine electric power event as a protection-focused attack," 2019. [Online]. Available: https://dragos.com/wp-content/uploads/CRASHOVERRIDE.pdf

[9] ESET, "Industroyer2: Industroyer reloaded," 4 2022. [Online]. Available: https://www.welivesecurity.com/2022/04/12/industroyer2-industroyer-reloaded/

[10] N. N. Labs, "Industroyer2: Nozomi networks labs analyzes the iec 104 payload," 4 2022. [Online]. Available: https://www.nozominetworks.com/blog/industroyer2-nozomi-networks-labs-analyzes-the-iec-104-payload/

[11] V. Labs, "Industroyer2 and incontroller: In-depth technical analysis of the most recent ics-specific malware," 7 2022. [Online]. Available: https://www.forescout.com/resources/industroyer2-and-incontroller-report/

[12] D. Zafra, R. Leong, C. Sistrunk, K. Proska, C. Hildebrant, K. Lunden, and N. Brubaker, "Industroyer.v2: Old malware learns new tricks," 4 2022. [Online]. Available: https://www.mandiant.com/resources/blog/industroyer-v2-old-malware-new-tricks

[13] OPC Foundation. (2017, 6) What is OPC? [Online]. Available: https://opcfoundation.org/about/what-is-opc/

[14] O. Foundation. (2017, 6) Unified architecture. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/

[15] M. J. McDonald, G. N. Conrad, T. C. Service, and R. H. Cassidy, "Cyber effects analysis using vcse promoting control system reliability," 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:5012619

[16] D. Bergman, D. Jin, D. Nicol, and T. Yardley, "The virtual power system testbed and inter-testbed integration," 2009. [Online]. Available: https://www.usenix.org/legacy/event/cset09/tech/full_papers/bergman.pdf

[17] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga, and S. Hariri, "A testbed for analyzing security of scada control systems (tasscs)," in *ISGT 2011*, 2011, pp. 1–7.

[18] T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu, and R. Reddi, "A control system testbed to validate critical infrastructure protection concepts," *International Journal of Critical Infrastructure Protection*, vol. 4, no. 2, pp. 88–103, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874548211000266

[19] V. Salehi, A. Mohamed, A. Mazloomzadeh, and O. A. Mohammed, "Laboratory-based smart power system, part i: Design and system development," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1394–1404, 2012.

[20] A. Hahn, A. Ashok, S. Sridhar, and M. Govindarasu, "Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 847–855, 2013.

[21] M. Shahidehpour and M. Khodayar, "Cutting campus energy costs with hierarchical control: The economical and reliable operation of a microgrid," *IEEE Electrification Magazine*, vol. 1, no. 1, pp. 40–56, 2013.

[22] H. G. Aghamolki, Z. Miao, and L. Fan, "A hardware-in-the-loop scada testbed," in *2015 North American Power Symposium (NAPS)*, 2015, pp. 1–6.

[23] I. Grinberg, M. Meskin, and M. Safiuddin, "Test bed for a cyber-physical system (cps) based on integration of advanced power laboratory and extensible messaging and presence protocol (xmpp)," 06 2015.

[24] Beanit. (2021) Iec61850 bean. [Online]. Available: https://www.beanit.com/iec-61850/

[25] Matrikon, "Matrikon OPC server," https://www.matrikonopc.com/, 2021.

[26] M. Zeller, "Myth or reality—does the aurora vulnerability pose a risk to my generator?" in *2011 64th Annual Conference for Protective Relay Engineers*. IEEE, 2011, pp. 130–136.

[27] K. Proska, J. Wolfram, J. Wilson, D. Black, K. Lunden, D. Kapellmann, N. Brubaker, T. McLellan, and C. Sistrunk, "Sandworm disrupts power in Ukraine using a novel attack against operational technology," Nov 2023. [Online]. Available: https://www.mandiant.com/resources/blog/sandworm-disrupts-power-ukraine-operational-technology

[28] B. Huang, A. A. Cardenas, and R. Baldick, "Not everything is dark and gloomy: Power grid protections against {IoT} demand attacks," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1115–1132.

[29] Y. Wang and R. Baldick, "Interdiction analysis of electric grids combining cascading outage and medium-term impacts," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2160–2168, 2014.

[30] B. Huang, M. Majidi, and R. Baldick, "Case study of power system cyber attack using cascading outage analysis model," *IEEE PES GM, Portland OR*, 2018.

[31] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, p. 29, 2011.

[32] T. ICS-CERT, "Mar-17-352-01 hatman-safety system targeted malware (update b)," 2019.

[33] M. Zhang, C.-Y. Chen, B.-C. Kao, Y. Qamsane, Y. Shao, Y. Lin, E. Shi, S. Mohan, K. Barton, J. Moyne *et al.*, "Towards automated safety vetting of PLC code in real-world plants," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 522–538.

[34] A. Abbasi, T. Holz, E. Zambon, and S. Etalle, "ECFI: asynchronous control flow integrity for programmable logic controllers," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017, pp. 437–448.

[35] I. Ahmed, S. Obermeier, S. Sudhakaran, and V. Roussev, "Programmable logic controller forensics," *IEEE Security & Privacy*, vol. 15, no. 6, pp. 18–24, 2017.

[36] A. Keliris and M. Maniatakos, "ICSREF: a framework for automated reverse engineering of industrial control systems binaries," *Network and Distributed Systems Security Symposium NDSS*, 2018.

[37] J. H. Castellanos, M. Ochoa, A. A. Cardenas, O. Arden, and J. Zhou, "Attkfinder: Discovering attack vectors in PLC programs using information flow analysis," in *Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses*, 2021, pp. 235–250.

[38] L. A. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. Mohammed, and S. A. Zonouz, "Hey, my malware knows physics! attacking PLCs with physical model aware rootkit," in *Proceedings 2017 Network and Distributed System Security Symposium*. Internet Society, 2017. [Online]. Available: https://doi.org/10.14722/ndss.2017.23313

[39] B. Green, M. Krotofil, and A. Abbasi, "On the significance of process comprehension for conducting targeted ICS attacks," in *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*. ACM, 11 2017. [Online]. Available: https://doi.org/10.1145/3140241.3140254

[40] B. Green, R. Derbyshire, M. Krotofil, W. Knowles, D. Prince, and N. Suri, "Pcaad: Towards automated determination and exploitation of industrial systems," *Computers & Security*, vol. 110, p. 102424, 2021.

[41] M. Pogliani, F. Maggi, M. Balduzzi, D. Quarta, and S. Zanero, "Detecting insecure code patterns in industrial robot programs," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 2020, pp. 759–771.

[42] T. Shekari, A. A. Cardenas, and R. Beyah, "{MaDIoT} 2.0: Modern {High-Wattage}{IoT} botnet attacks and defenses," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3539–3556.

[43] M. Eckhart and A. Ekelhart, "Digital twins for cyber-physical systems security: State of the art and outlook," *Security and quality in cyber-physical systems engineering*, pp. 383–412, 2019.

[44] S. Hussain, S. S. Hussain, A. Iqbal, S. Zanero, and E. Ragaini, "A novel methodology to validate and evaluate combined cyber attacks in automated power systems using real time digital simulation," in *2021 IEEE 2nd International Conference on Smart Technologies for Power, Energy and Control (STPEC)*. IEEE, 2021, pp. 1–6.

[45] M. N. Nafees, N. Saxena, A. Cardenas, S. Grijalva, and P. Burnap, "Smart grid cyber-physical situational awareness of complex operational technology attacks: A review," *ACM Computing Surveys*, vol. 55, no. 10, pp. 1–36, 2023.

[46] A. Soldatov and I. Borogan, "Russian cyberwarfare."

[47] A. Greenberg, *Sandworm: A New Era of Cyberwar and the Hunt for the Kremlin's Most Dangerous Hackers*, 11 2019.

[48] A. Troianovski and E. Nakashima, "How Russia's military intelligence agency became the covert muscle in putin's duels with the west," *Washington Post*, vol. 28, 2018.

[49] A. Satariano and N. Perlroth, "Big companies thought insurance covered a cyberattack. they may be wrong," *The New York Times*, 2019.

[50] J. Bateman, "Russia's wartime cyber operations in Ukraine: Military impacts, influences, and implications," 2022.

[51] L. Franceschi-Bicchieraiarchive, "What's next in cybersecurity," https://www.technologyreview.com/2022/11/28/1063703/whats-next-in-cybersecurity/, 2022.

[52] A. Greenberg, "Russia's new cyberwarfare in Ukraine is fast, dirty, and relentless," 11 2022. [Online]. Available: https://www.wired.com/story/russia-ukraine-cyberattacks-mandiant/

[53] G. Corera, "Inside a US military cyber team's defence of Ukraine," 10 2022. [Online]. Available: https://www.bbc.com/news/uk-63328398

[54] V. GERA, Z. MILLER, and M. BALSAMO, "Poland: Russian-made missile fell on our country, killing 2," 11 2022.

Appendix A.
Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

A.1. Summary

This paper presents a detailed analysis of two industry malware samples, Industroyer 1 and 2, and evaluates the protocol-level behaviors with a sandboxing environment. It also uses a simulator to analyze the potential impact of an attack to today's power grid.

A.2. Scientific Contributions

- Independent Confirmation of Important Results with Limited Prior Research
- Creates a New Tool to Enable Future Science
- Provides a Valuable Step Forward in an Established Field

A.3. Reasons for Acceptance

1) This paper contributes a detailed analysis of real-world malware, with detailed, protocol-level behaviors. It describes the background in a very clear manner. This could potentially help other researchers to move into this area.
2) The sandboxing environment could be useful for researchers in this area, although there are noteworthy concerns about the novelty of this tool.
3) The paper is well written and provides background to an area where future research could be conducted.

A.4. Noteworthy Concerns

1) The sandboxing environment is interesting, but it would be stronger if there are more novel designs that went into that. It seems to be largely based on a containerized environment with Mininet, with a limited amount of customization for this particular problem setting.
2) The attack impact to the power grid is interesting, but somewhat loosely coupled to the rest of this paper.