Towards a High-Fidelity Network Emulation of IEC 104 SCADA Systems

Luis Salazar
UC Santa Cruz
luedsala@ucsc.edu

Neil Ortiz
UC Santa Cruz
nortizsi@ucsc.edu

Xi Qin
UC Santa Cruz
xqin9@ucsc.edu

Alvaro A. Cardenas
UC Santa Cruz
alacarde@ucsc.edu

ABSTRACT

With the rise of malware targeting industrial control systems, researchers need more tools to develop a better understanding of the networks under attack, the potential behavior of malware, and design possible defenses. One of the most important protocols used in practice today is IEC 104, which is used to monitor and control the Power Grid of several countries, as well as to monitor and control other critical infrastructures such as gas, oil, and water systems. In this paper we present our preliminary results in implementing the IEC 104 industrial protocol standard in Python and integrate it to a network emulation tool supported by Mininet.

CCS CONCEPTS

• **Computing methodologies** → **Simulation environments**; **Simulation tools**; • **Security and privacy** → *Domain-specific security and privacy architectures*; • **Networks** → *Cyber-physical networks.*

KEYWORDS

network emulation, mininet, simulation framework, power grid simulation, SCADA systems

ACM Reference Format:
Luis Salazar, Neil Ortiz, Xi Qin, and Alvaro A. Cardenas. 2020. Towards a High-Fidelity Network Emulation of IEC 104 SCADA Systems. In *2020 Joint Workshop on CPS&IoT Security and Privacy (CPSIOTSEC'20), November 9, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3411498.3419969

1 INTRODUCTION

Many critical infrastructures, such as power systems, have existed for over a century; however, it is only in the past two decades that remote monitoring and control of these systems migrated from serial communications to IP compatible networks, supporting various industrial control protocols such as IEC 60870-5-104 (IEC 104), DNP 3.0, Modbus/TCP, ICCP, and IEC 61850. Some protocols focus on substation automation (IEC 61850), others on communications between control centers (ICCP), and others on remote monitoring and control of large-scale systems (IEC 104 or DNP 3.0).

Critical infrastructures are also at an increased risk of cyberattacks. In particular, the power grid has not only received significant attention in academia [1, 2, 19] but has also been the subject of real-world cyberattacks, such as Ukraine's 2015 and 2016 blackouts [23].

One of the particular industrial protocols targeted by Industroyer (also known as Crashoverride), the malware related to the Ukraine2016 cyber attacks, was IEC 104. However, we found few tools supporting this protocol, and even if they support it, their support is partial and unable to dissect all types of packets in the protocol entirely (e.g., the IEC 104 packet dissector of Wireshark is not able to correctly parse some of the packets we have obtained from real-world power systems).

We propose a new network emulation framework to support experimentation with IEC 104 networks to address this concern. Our goal is to use our framework to teach engineers about IEC 104 networks, help security engineers test their solutions, and help malware analysts run industrial malware in our framework to evaluate new malware dynamically. This paper summarizes our first efforts to achieve these goals.

In particular, our contributions are as follows: (1) we develop a new implementation of IEC 104 and deploy it in a highly modular and flexible network emulation software (Mininet). (2) To illustrate how researchers can use our framework to study malware, we develop a proof-of-concept malware that uses IEC 104 to open the circuit breakers in RTUs. Finally, (3) to illustrate how researchers can develop defenses using our framework, we develop a simple clustering algorithm using deep-packet inspection features from IEC 104 to detect anomalies and test in our system. Our solution is available to the public at https://github.com/Cyphysecurity/NEFICS.

The current state of our research is in preliminary work, so there are some limitations. The most significant limitation of our study is the crude power system under consideration in this paper. We plan to study how to integrate our network emulation tool to more robust power system simulation tools such as PowerWorld in future work.

The rest of the document is organized as follows: in section 2 we present some related works to our research, in section 3 we describe the IEC 104 protocol and showcase some relevant items of this protocol, in section 4 we present our framework design considerations and approach, in section 5 we discuss and demonstrate our proof-of-concept malware running against the framework, in

CPSIOTSEC'20, November 9, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8087-4/20/11.
https://doi.org/10.1145/3411498.3419969

section 6 we present a second use case involving deep packet inspection defenses, finally, we present our conclusions and future work considerations.

2 RELATED WORK

The need to implement realistic and scalable experimental platforms with different capabilities has increased significantly in the past decade. In particular, previous work has focused on creating power grid testbeds through simulators, hardware, hybrid platforms (integrating hardware and software).

Table 1: Communication Protocols used in Smart Grid Testbeds [5].

Protocols	Percentage
DNP3.0	22%
C37.118	22 %
IEC 61850	20%
Modbus	18%
OPC UA	6%
FIPA	4%
OPC DA	2%
Others	3%

The most comprehensive survey we found on smart grid testbeds [5] shows many diverse interests, such as demand response, distributed energy resources, energy storage, electric transportation, advanced metering infrastructure, distribution grid management, Network communications, Wide-Area situational awareness, and Cyber-Security. The prevailing trend shows that 40% of testbeds have centered in these two last areas.

Cyber-Security is a field that has drawn particular interest since the infamous attacks against Ukraine's power grid. In 2015, during Christmas eve, hackers compromised the Supervisory Control and Data Acquisition (SCADA) system of the energy distribution systems and successfully disrupted the electricity supply. Following that attack, in 2016, attackers used the virus Industroyer to perform a similar attack using legacy protocol IEC 104 (section 3) to execute their command injection attacks on the communication network. Events extensively analyzed on [4] and [18].

However, few testbeds offer implementations for IEC 104, which was used by Industroyer to perform its attack. As Table 1 shows, a big percent of testbeds support DNP3.0 and C37.118 (22% each), followed by IEC 61850 (20%) and Modbus(18%), but not one of them supports IEC 104 [5].

Therefore, the primary goal of this study is to propose a framework that uses power grid simulation (as explained in 4.5) and emulating a network communication (based on SDN as described in 4.3) to fill that gap.

Some simulation software frameworks, such as Mosaik [17], simulate various scenarios in large-scale smart grids. According to the smart grid network topology, its core function is to create a communication framework among the network endpoints. With this communication framework, all the components, including the control centers and Remote Terminal Unit (RTU), can exchange the control commands and the measurement data in the corresponding physical control system events. Mosaik configures TCP socket connections for the simulated power grid components to connect to Mosaik modules. However, it does not emulate the configurations in the network layers of the OSI model. Thus, there are no setting options for users to set industrial protocols and other network configurations.

On the other hand, as a co-simulation framework, Mosaik requires other established simulators (e.g., Matlab Simulink) to model the smart grid components. To access these models, it provides APIs to connect to the external power grid simulator. In contrast, our framework integrates the smart grid models as built-in components and creates a seamless design between the physical control system and the communication network.

3 IEC 104 DESCRIPTION

As we mentioned before, one of the most recent cyberattacks against power grid systems occurred in 2016 in Ukraine, where attackers issued control commands via the IEC 104 protocol to cause a power outage that affected more than 200,000 consumers [15].

The International Electrotechnical Commission (IEC) originally developed IEC 104 in 1995 in its original version, IEC 60870-5-101 (IEC 101). Then, in 2000, it was extended to IEC 104. It is widely used in Europe and Asia to monitor and control large geographical areas. IEC 104 encapsulates telecontrol messages into Application Protocol Data Units (APDUs) over TCP/IP using port 2404, as shown in Fig. 1.

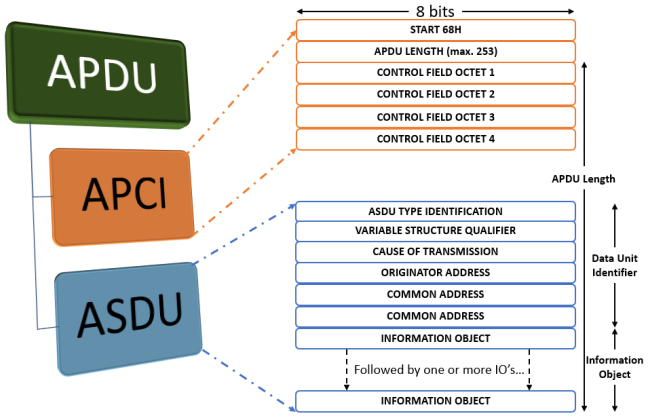


Figure 1: IEC 104 APDU Octets Structure

The contents of this TCP payload have one or more APDUs. The **Application Protocol Control Information (APCI)** is the first part of an APDU, which acts as the header of the message. The **Application Service Data Unit (ASDU)** is the second part and carries the sensor values and control messages between the controlled station (e.g., **Remote Terminal Unit (RTU))** and the control station (e.g., SCADA systems).

There are three types of APDUs:

- **I-Format** APDUs carry sensor and control data between endpoints.
- **S-Format** APDUs acknowledge after a specific (but configurable) number of I-format APDUs received.

- **U-Format** APDUs provide three connection control functions: start transmission (`STARTDT act/con`), stop transmission (`STOPDT act/con`) and keep-live connection request (`TESTFR act/con`).

An ASDU is a I-Format APDUs that comprised of a data unit identifier (DUI) and an information object (IO), as shown in Fig. 1. DUIs always maintain the same structure for all types of ASDUs, while the structures of IOs vary for different types of ASDUs. DUIs involve "what" type of data or command is being sent (type ID) and "why" it was sent (cause of transmission). Each IO is in correspondence with a specific field device, and the device has a unique ID known as the Information Object Address (IOA). IOs encapsulate the actual measurement data values or control commands within their IOAs.

For instance, figure 2a shows an example of an ASDU for measurement data. Marked in red boxes, the field "Value" stores the measurement value of 42105.3 (voltage) in the address of 1002 using type M_ME_TF_1 (36). In contrast, figure 2b shows an ASDU for commands, C_SC_NA_1 (45), where the field "SCO" specify the type of change to 'Execute' in a cicuit-breaker that in this case is to set it to be 'On' for the address 101. Another type of measurement data are status values for switches and circuit breakers in a substation, such as ASDU type 3, `M_DP_NA_1`. Figure 2c shows an example of the status of the circuit breaker corresponding to the address 101 where 'DPI' (Double Point Information) indicate its status.

In our simulation scenario, we use ON to determine the circuit breaker's state as "OPEN", and OFF to determine the state as "CLOSED". Some IOs also include the timestamps and inconsequential quality flags when reporting the data.

A common use of IEC 104 is to perform Automated Generation Control (AGC) to the generation units in a Generation plant. In Fig. 3, we show the process AGC extracted from a reference dataset captured from a real-world smart grid. The AGC process works so that the SCADA control center sends an expected amount of power generation command in ASDU type 50, C_SE_NB_1, (represented by the red square symbol in the figure) to one of the power generators based on the real-time global load analysis. The power generator acknowledges and confirms with the received expected power value (represented by the blue triangle in the figure). Each exchange of AGC requests and confirmations finishes within milliseconds.

In summary, IEC 104 is a comprehensive protocol for monitoring and controlling communications between control centre (SCADA) and substations (RTU) in power grids, which allows to know the current status of the grid and change their operation points as well as the topology of the network.

4 FRAMEWORK

4.1 Power systems overview

The primary purpose of a power grid is to supply the energy that users demand. However, because of energy in high amounts is not feasible to store it; the power of electricity must be consumed as it is generated. Therefore, power grids are systems that demand coordination between supervision and control, where SCADAs system performed an import role. However, SCADA systems are the central target for an attack.

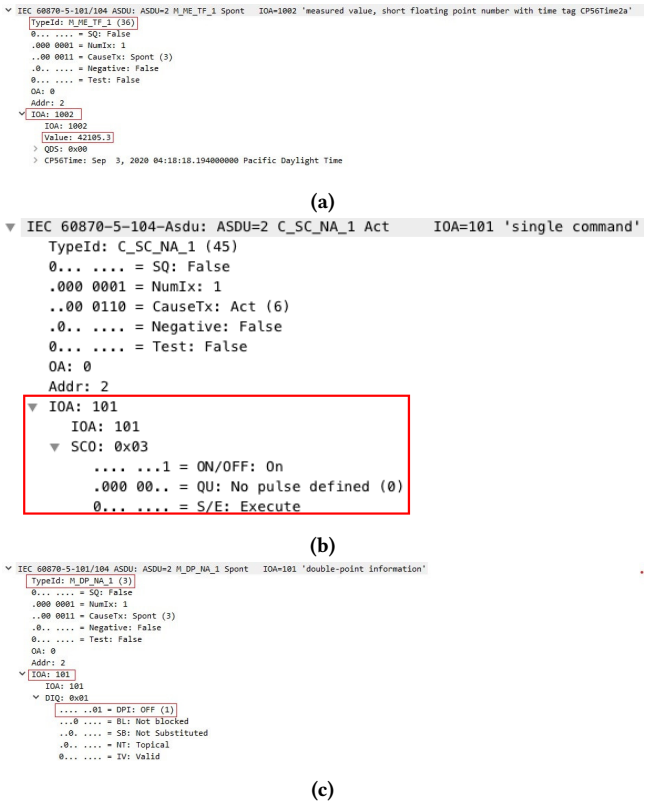


Figure 2: (a) Voltage measurement using ASDU type 36. (b) Command data using ASDU type 45 for opening a circuit breaker. (c) Status measurement using ASDU type 3 for a circuit breaker.

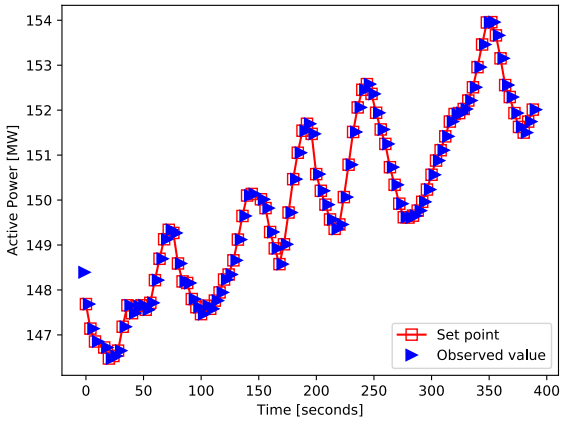


Figure 3: Example of an AGC command operation from a real-world smart grid dataset. Command data ASDU type 50. The time series view of the field device reacts to the set point control command in a prompt manner.

A complete power grid comprises three parts: generation, transmission, and distribution, Fig. 4. In generation are the power plants and the substations that increase the voltage to transmission level. In transmission, there are mainly the power lines and intermedia substations that help to keep the voltage level during the long journey to the load. Finally, the distribution systems are the substation that decreases the voltage to consumption level and distributes the electricity between end-users.

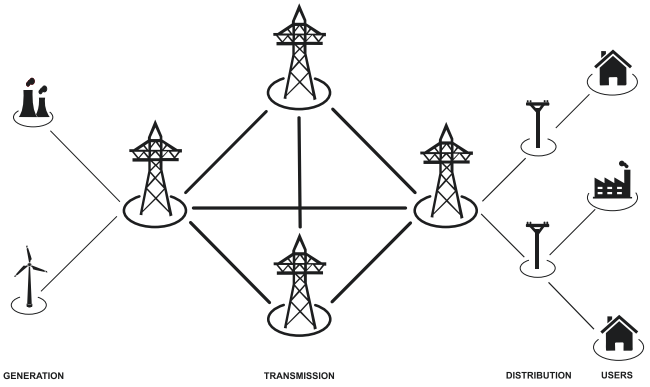


Figure 4: Power Grid Anatomy

SCADAs system is in charge of supervising and controlling the operation of the power grid (generation, transmission, and distribution) within acceptable levels for the elements in the power grid. However, the power grid operation is still a complex task that required human decisions in cases of contingency. SCADA are systems whose inputs on one side are measurements (such as voltage, currents values, and breaker positions) and output the operation state of the system. On the other hand, inputs are commands that produce topology change actions over the system as input. Fig. 5 shows a representation of this concept.

A RTU is an electronic device that interfaces the power grid with the SCADA. These devices collect the measurements in the substations (such as voltage, currents values, and breaker positions) and send to SCADA. Besides, they execute the commands from the SCADA. All the communication between SCADA and RTU is carrying out under severals standards among them the standard IEC 104 [6].

Figure 5: SCADA's role in Power Grid. Concept representation

4.2 IEC 104 Implementation

We use Python to implement our RTU simulation, using SCAPY packet manipulation tools for computer networks [3] to build the packets. The simulator has three modules: Parse Module (PM), the Communication Module (CM), and the Actuator Module (AM). The PM is the main core of the simulator and is in charge of packing and unpacking IEC 104 packets. It receives the floating value of current and voltage, or integer value for the breaker status from the substation, and encapsulated in an IEC 104 packet, either ASDU Type 36 or 3, then give to the CM for transmission. On the other hand, it unpacks the IEC 104 command packets received from the SCADA and passes the command to AM for its execution. The CM contains the state machine rules that determine the RTU behavior, define in [6]. The AM is where the voltage, current, and Req calculations perform along with commands executions. Fig. 6 illustrates the general concept of the RTU simulator. We plan to release the RTU simulator as open-source in the future once we complete the protocol implementation.

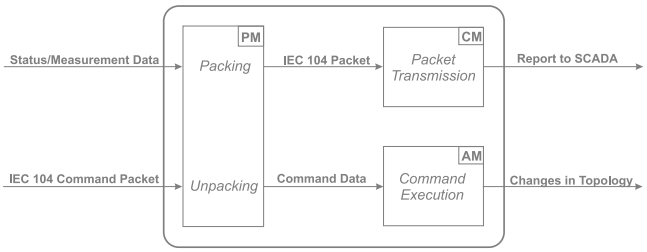


Figure 6: RTU simulator. Modular concept.

4.3 Network Emulation

Regarding the network emulation, there are two main elements in the simulation of our scenario. First, we need to recreate the network behavior of the power grid as a communications channel between its components. Also, we need to simulate the physical properties of the system. Since several components comprise these systems, and these components are communicating with each other, we need a mechanism to synchronize the physical state of the system upon any changes produced by logical commands. Because these components are not in the same endpoint, the different components must execute this physical synchronization via a different communications channel.

To be able to adjust the number of endpoints dynamically, we implement our solution under a software-defined network (SDN) environment, using an existing rapid prototyping network emulator called "Mininet."[20] Therefore, we implement simulated endpoints mimicking the behavior of both the SCADA and RTU endpoints and deploy these endpoints within different Mininet nodes, as shown in Fig. 7. This implementation provides an isolated network in which the different components of the simulated power grid can communicate with each other using the IEC 104 protocol.

Furthermore, each RTU must account for any interaction between the SCADA and other RTUs in the system. Thus, if the SCADA sends a command that alters the current state of any RTU in the system, the others must reflect this change as well. While a set of equations that describe the entire behavior of the system can model this, the scalability and distributed nature of the system would be lost in this type of model. Therefore, we decided to break

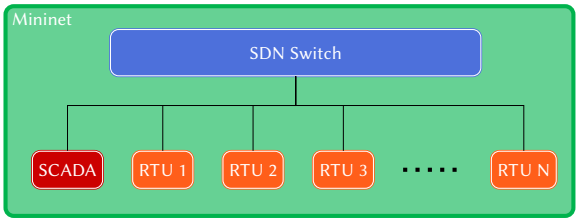


Figure 7: Deployment schema. Each mininet node simulates a component of the power grid. The communication between nodes flows through a mininet SDN switch.

down the model into single components that run within each RTU. Each component receives data from neighboring components and feeds back its results, making each RTU into a semi-autonomous mini-system. Moreover, the totality of the data exchange between the nodes fully describes the entire system.

To exchange information, the RTUs establish a secondary communications channel using a non-standard protocol to exchange these messages. We implemented this secondary channel using a UDP protocol. We decided on this because of the nature of the messages the RTUs are exchanging. Since the states are frequently updating, a missing packet will not adversely affect the simulation, as the polling frequency of the IEC 104 is much slower than the physical message exchange frequency. To establish a pseudo-directionality for the messages, these carry sender and receiver IDs corresponding to the IDs of each RTU. During the instantiation of each RTU, the IDs of the "physical neighbors" are provided to them so they can exchange messages with them and determine the state of their sub-system. As a result, if any RTU in the system executes a command sent by the SCADA, the resulting state is synchronized with the remaining RTUs far quicker than the next polling from the IEC 104. Thus, the next update received by the SCADA will reflect the changes made by the command that was issued before.

Because the data exchange between the different endpoints in the system only needs raw values (integers and floats) to determine the simulation status, we propose a simple protocol to exchange these messages based on the APDU ID of the endpoints. Our custom protocol packet holds the sender ID, receiver ID, and message ID as integers, and four slots for data (two integers, two floats), for a total of 28 bytes. Since our physical model is rather simple, we do not need to exchange overly-complicated packets. The message ID determines the kind of data and associates a particular data slot to the value. The endpoints run the data exchange in a continuous loop that updates each RTU's current status, resulting in multiple data exchanges per second. Following this raw data exchange, each RTU packs the IEC 104 I-frames –in a separate execution thread– with the status data at regular 1-second intervals, using our SCAPY-based IEC 104 module, after which it sends this data to the SCADA server using a standard TCP socket.

4.4 Supporting DPI Efforts

We want our framework to support the development of new Deep-Packet Inspection (DPI) defenses. In order to do this we design our system to support:

(1) **Parsing:** Parse the network traffic capture under IEC 104 protocol. We wrote parsers (in Python) for IEC 104 traffic (section 4.2) since the on-shelf parser such as Pyshark cannot dissect layers correctly. For this project, we write extra APIs to extract fields and to perform computation based on the parsed data.
(2) **Preprocessing:** Clean the parsed and restructured data extracted from network traffic capture, such as the physical measurement of power grid field devices.

4.5 Power Grid Simulation Scenario

Since our proposal's primary purpose is to analyze the cybersecurity component in power grids, we used a simple power system model shown in Fig 8. This kind of model allows fast power flow calculation along with sudden changes in topology. We used a resistive model. i.e., negligible capacitive, inductive effects, and not transient effect (stable state only) with no electrical failures (no protection elements). The grid is a radial topology that connects the generation side to the load side through two transmission substations (TS), including a generation and load substations (GS and LS). The system is a 500 kV grid that feeds a 3 GW load through a transmission line of 3 circuits, each circuit with a maximum loading capacity of 1000 MW (SIL), according to [12]. The line is 300 miles long, divided into three sections of 100 miles modeled as an equivalent resistance (Req) of 3 equal parallel resistance controlled by circuit-breakers. There is a circuit-breaker per line-circuit, located in each TS, in order to model the topology changes during the control operations. The circuit-breakers are the actuators that an attacker's main goal is to gain access and control over them to disconnect the load (produce a blackout in the system - case 4 Fig. 9). Last but not least, there is a RTU as the sensor unit per substation to supervised currents, voltages, and breaker's status and deliver commands from Supervisory Control and Data Acquisition (SCADA) to the grid. All communications between SCADA and RTU are under the protocol IEC 104 (section 3), where we use three types of IEC 104 packets: (2) measurement and (1) command packets, Table 2. There is one SCADA server that supervises and controls the system, which represents the Control Center. The SCADA is an external element in the system, which means that there is no electrical connection with the power grid, located in a remote place connected to the RTUs via ethernet.

To compute the power flow, we used Ohm's law to calculate voltages and currents, Equation 1, in every substation. When a substation detects a topology change in the system, it will perform the following operations: first, compute its Req, Equation 2, according to its local circuit-breaker status (open or close), then calculate voltages and currents base on the information received by it closest neighbors. Finally it will report its new values to its neighbors, section 4.3.

$$V = IR \tag{1}$$

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} \tag{2}$$

An attacker could disrupt the system's regular operation by taking control of the circuit-breaker of any substation. By changing its

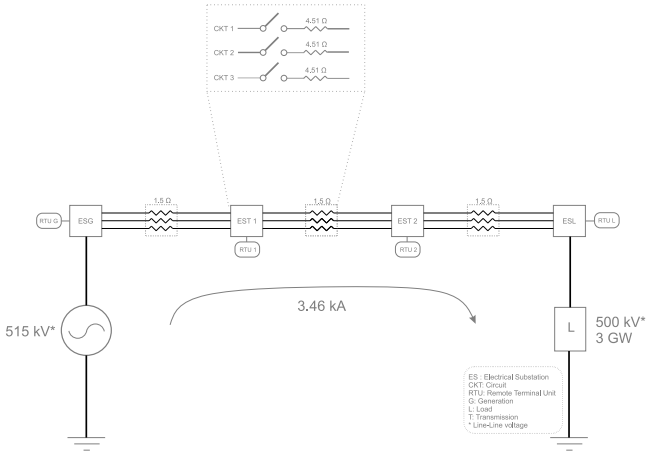


Figure 8: 500 kV, 3 GW power grid example used as a test case.

topology an attacker could lead the system to an unsafe state. Since the loading condition remains unchanged, and the grid is under maximum capacity, any unexpected change, e.g., a cyber-attacker, would put the system under stress conditions. There are three possible scenarios in our test case represented in the topological changes depicted in Fig. 9. Case 1 is the system in regular operation. Case 2 to 4 are the possibles scenarios resulting from an attack. Case 2 and 3 are scenarios where an attacker opens one and two of the line transmission circuits, respectively, leaving the grid under stress conditions, i.e., increasing the Req of the whole system; As consequence, the generator have to compensate the losses by producing more energy even more than its nominal capacity. Case 4 is the worst-case scenario, and the foremost goal of an attacker; produces a blackout, i.e., opening the three-line circuits and disconnecting the load from the grid.

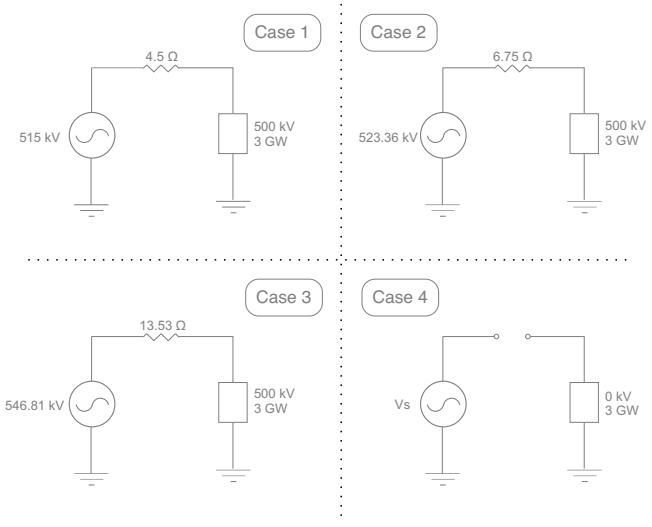


Figure 9: Possible topological changes due to cyber-attack in the power grid test case.

ASDU	Type	Data	Source
36	Short floating point	Voltage & current values	RTU
3	Double-point	Breaker status	RTU
45	Single command	Open/close commands	SCADA

Table 2: IEC104 packets used in the test case

4.6 Physical System

The typical design to simulate physical systems involves translating all the different physical components into some non-linear system described by several differential equations. After doing so, the simulation follows these equations to reflect the physical state of the system. As with every solution, this approach has both advantages and disadvantages. On the one hand, the simulator has complete knowledge of every variable in the system, as the equations characterize the system entirely. Moreover, as the simulator holds all the variables, any response is given without any relevant delay. On the other hand, the simulation is fixed to a specific scenario, 4.5, and does not have any flexibility to scale up without a significant update to the system's equation model. Moreover, the fact that the model describing the system is fixed means that all the information must be processed in the same simulation process, regardless of how many threads the simulator uses, which means a single end-point simulating the whole system.

Taking these advantages and disadvantages into consideration, we decided that our simulation should break apart the model into simpler mini sub-models that can work autonomously, regardless of the scenario's topology. By building the simulator, we mean to determine the simulation model of a single RTU and run it as a stand-alone simulator in a single end-point. This way, we do not need to know the topology beforehand, and the simulator can accommodate different physical topologies without changing the underlying model. However, the disadvantage comes with an added delay, as we need some mechanism for the separate RTUs to exchange messages regarding their current state.

Our simpler model has each RTU in the system depending only on its immediate neighbors' information to determine its current state. We simulate three different types of RTUs: a generation, a transmission, and a load. The generation and load RTUs are static, meaning that all the parameter definition takes place upon its instantiation. The transmission RTUs are the system's dynamic components, which can receive commands from the SCADA service to change their internal state. As their internal state is dynamic, they need to calculate the internal currents and voltages, dependent on the voltage input from the RTUs linked to the generation, and the load observed from the RTUs linked to the final load.

As every RTU has a given ID, whenever the simulator instantiates a new RTU, it needs its ID and the physical neighbors' IDs linked to this RTU. The first message that every RTU sends via this protocol is a broadcast 'who-is' message, similar to the ARP protocol's behavior. The actual simulation of the RTU will not begin until it knows the IP address of every neighbor. In the limited tests we have made so far, this handshake phase takes a few seconds to initiate the system's full simulation. After this phase, the message exchange

begins. The main values exchanged by the RTUs are voltage and load.

5 USE-CASE 1: TESTING THE BEHAVIOR OF MALWARE

In the literature, there are different types of cyberattacks for power grids [13]. On one side, some attacks focus on damaging components in the systems, such as damaging generation machines in the Aurora case [22], while others attempt blackouts like the cyberattacks in Ukraine [18]. The blackouts that occurred in Ukraine during 2015 and 2016 exposed several of the vulnerabilities in the power grid operation. They also gave us insights on the practical steps attackers will take to launch their attacks. For instance, in 2015, the attackers performed the blackout by obtaining remote access to the SCADA system [9], while in 2016, the attacks were automated with malware [14]. However, in both attacks, the main goal of the attacker was to create power outages that affected a hundred thousand customers for several hours. In this work we implement a simulation of an attack similar to Industroyer [8], to model the attacks that happened in Ukraine on December 2016.

5.1 Industroyer

On December 17, 2016, Ukraine's power grid was affected by a cyberattack, which caused, to an extent, a power outage in its capital. A malware framework dubbed "Industroyer" is presumed to be the cause of this cyberattack. Several components comprise the malware framework: a backdoor component that allows the attacker to control the components of the framework, a launcher component used to execute the payloads, a wiper component that renders the infected system inoperable, and four payloads designed to interact with power grid substation systems via the standard protocols IEC 101, IEC 104, and IEC 61850 [4].

Our focus revolves around the IEC 104, described in section 3. The IEC 104 payload component of the malware framework uses a configuration file to determine the attack's properties. Within this file, the attacker includes, among other properties, the target RTU's IP address, TCP port, ASDU address, the process to stop in the infected system (SCADA service), and the IOA range.

Once the payload is running, its behavior is very straightforward: it establishes a connection with the target RTU, initiates a data transfer state, and sends a command to the target IOAs to cause an outage. The entire attack uses the information contained in the configuration file to ascertain the target's parameters.

This design has one considerable disadvantage: the attacker must have prior knowledge of the power grid s/he is attacking, including RTU's IP addresses, ASDU addresses, and IOA ID's. This disadvantage gives some validity to the claim that the attackers launched a highly targeted attack, as they must have known details of the target infrastructure beforehand.

5.2 Developing a new malware as a proof of concept

A command injection attack happens when an unauthorized party sends a command to an RTU. To execute this command injection attack, we take some ideas from the behavior of Industroyer to create malicious software that runs on a compromised device in the SCADA network. From this compromised device, the malware launches attacks in stages: The first stage corresponds to a reconnaissance state in which it scans the network to identify any live hosts and then identify which of those are RTUs. Following this initial reconnaissance stage, the malware connects to the identified RTUs posing as a legitimate SCADA server, which allows the malware to receive the polled measurements and status from all the identified RTUs. The malware extracts and accounts for all the IOAs present in each RTU by receiving the reported status data. After gathering enough information from the RTUs to map out all the circuit breakers in the system to each RTU, it executes the final stage of the attack by sending commands to each RTU with a circuit breaker that instructs the RTU to open the breakers, resulting in a massive blackout.

We use a Python script to simulate our malware. The script runs in an additional node in mininet representing the compromised device. Once the execution starts, the script achieves live hosts' identification by sending ARP probes destined to each host in the compromised machine's subnet. Once identified, the malware probes each live host with a stealth SYN scan (A half-open TCP connection that closes the socket before the three-way handshake is complete) for the specific TCP port 2404 designated for IEC104 communications. If a live host is an RTU, the TCP socket will receive a TCP SYN-ACK packet; otherwise, it will receive a TCP RST packet. With this simple technique, the malware can ascertain whether a particular endpoint is likely to be an RTU.

After this, the script opens a socket with each RTU and sends a 'STARTDT act' packet, allowing the malware to receive the IEC104 polled measurements and states, looking for any RTUs that report a type-3 ASDU, which corresponds to a circuit breaker. For each type-3 ASDU, the malware uses the same scapy-based IEC 104 module to extract the IOA corresponding to the particular circuit breaker the RTU is reporting, mapping out which RTUs have circuit breakers that could be compromised. After the malware finds no new circuit breakers in the network, it sends a type 45 ASDU for each identified breaker, instructing each corresponding RTU to open it. By doing so, and because our simulated power grid is a radial circuit, the load node is "physically disconnected," causing a simulated blackout.

The authors in [11] also propose a series of interesting deception attacks in an emulated power grid system. Although the assumed adversary shares the ARP poisoning technique to obtain network access in their and our attack scenarios, our attack design differs from theirs in terms of the industrial protocol, the power grid component, the threat model, and the consequences of the attack.

6 USE-CASE 2: DEVELOPING DPI DEFENSES

We focus on deep-packet inspection (DPI) of payload data units in the application layer of IEC 104 packets, particularly the physical device measurement values.

For our simulation, we consider two scenarios and collect the network traffic captures for each scenario. As a baseline, we place the initial capture during the regular operation of the system. The regular operation is that the SCADA system receives the measurements and status from the RTUs and sends commands provided by the operator. Then we conduct the second capture when the command injection attacks the system during the regular operation.

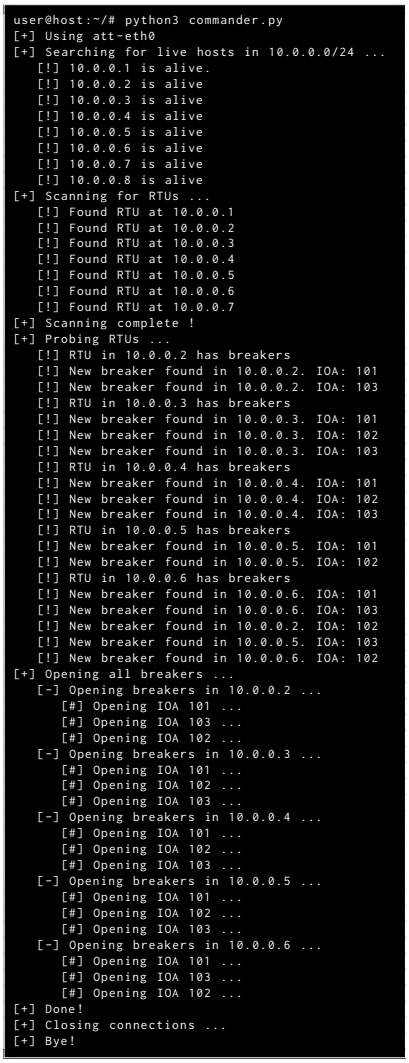


Figure 10: Execution of the simulated malware performing a command injection attack.

So the capture has mixed traffic traces in regular operation and under the attack. As illustrated in previous sections, we propose the attack scenario where the adversary uses a compromised device to scan and communicate with the RTUs. With these two datasets, we can measure and establish the baseline of normal operations, and analyze the influences in physical devices' measurements after the attack.

6.1 Feature Selection

For each APDU in the network traffic, we pick the payload data types as listed in Table 2 as categorical features and measurement values as a numeric feature. We use one-hot encoding to create the feature vectors, as shown in Table 3.

Table 3: Examples of Feature Vectors

	type3	type36	type50	measurement
command data	0	0	1	0
voltage data	0	1	0	5200

6.2 Anomaly Detection

We apply clustering algorithms to detect abnormal traffic from traffic capture. First, we train K-Means [10] and DBSCAN [7] algorithms on the traffic captured without attacks. Since the clustering result from both algorithms are similar, we present K-Means results only in this section. First, we use the Silhouette score [16] and Elbow method [21] to decide the number of clusters when using K-Means, as shown in Figure 11c, Figure 11a, Figure 11d and Figure 11b. If the Silhouette coefficient on the x-axis value is greater than 0.5, then it indicates that the data point is well matched to its cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If the majority of points have a low or negative value, then the clustering configuration may have too many or too few clusters. Elbow method can help us pick the cluster number at the elbow position with the least imprecise cluster assignments. Therefore, we conclude that 4 clusters are the best choice. In Figure 12a, we use compressed feature components to visualize the clustering results with principal component analysis (PCA). PCA compresses four feature components into two principal components and preserves 86% variance of the original feature vector. Each cluster composition is listed in Table 4.

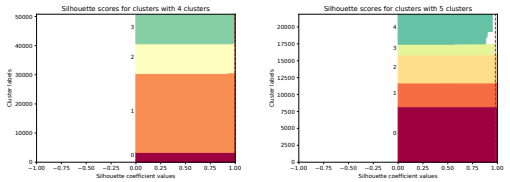
Table 4: Cluster composition under regular operation

cluster 0	circuit breaker status
cluster 1	current measurement around 40000 amps
cluster 2	control command confirms the close of breakers
cluster 3	voltage measurement around 5200 volts

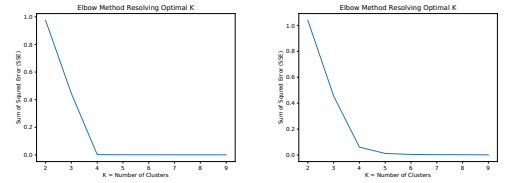
We then apply the same clustering algorithm on the traffic capture when the testbed is in the same operation configuration but under attack. From Fig. 11d, we have a cluster 4 successfully grouping all the abnormal values of current and voltage when they suddenly change to negligible values, in Figure 12b. Comparing the cluster groups in Figure 12a and Figure 12b, there is a new fifth cluster in Figure 12b. This new cluster successfully aggregates all the network traces containing the abrupt drop of the currents and voltages to negligible values. This phenomenon happens because the attacker opens all the circuit breakers in the transmission line. So the measurements of the field devices all drop to close to zero if the devices locate in the same transmission line as those circuit breakers. The operators can further investigate all the data points in cluster 4 to confirm the lines affected by the attack.

7 CONCLUSIONS AND FUTURE WORK

In this paper we implement the IEC 104 protocol and embed it in a Mininet implementation. Our work is ongoing and we plan to

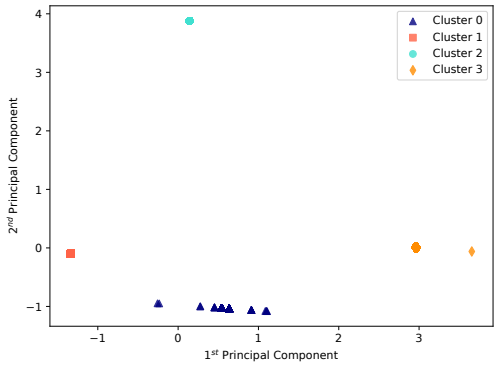


(a) normal operation (b) operation under attack

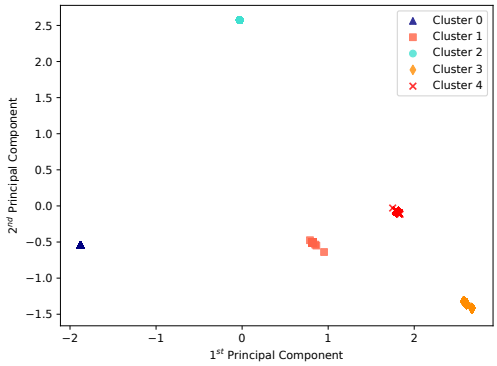


(c) normal operation (d) operation under attack

Figure 11: Choosing the proper number of clusters with K-Means.



(a) normal operation



(b) operation under attack

Figure 12: K-Means results in two situation with the identical testbed configuration

release it as open-source in the future once we have a complete implementation of the protocol. We hope this software will help other researchers experiment with this industrial protocol and develop their own deep-packet inspection tools. With our framework, we can test various attacks and capture network traffic. We also illustrate how our framework can be used to implement simulated attacks similar to Industroyer, and then used preliminary machine learning techniques to detect this malware.

While our simulation of the power system is limited at this point, we are looking into how to integrate our network emulation software with high-fidelity power system simulators like PowerWorld, where we can develop and deploy the dynamics of more complex power systems. This integration will also help us in developing more realistic and useful anomaly detection tools.

8 ACKNOWLEDGMENTS

This work was performed under the financial assistance award 70NANB17H282N from U.S. Department of Commerce, National Institute of Standards and Technology (NIST), the National Science Foundation under award CNS-1929406, and by the Air Force Research Laboratory under agreement number FA8750-19-2-0010. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

REFERENCES

[1] Rafael R. R. Barbosa, Ramin Sadre, and Aiko Pras. 2012. A first look into SCADA network traffic. In *Proceedings of the 2012 IEEE Network Operations and Management Symposium*. IEEE, Maui, HI, USA, 518–521. https://doi.org/10.1109/NOMS.2012.6211945

[2] Shameek Bhattacharjee, Aditya Thakur, and Sajal K. Das. 2018. Towards Fast and Semi-supervised Identification of Smart Meters Launching Data Falsification Attacks. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (Incheon, Republic of Korea) *(ASIACCS '18)*. ACM, New York, NY, USA, 173–185. https://doi.org/10.1145/3196494.3196551

[3] Philippe Biondi and the Scapy community. [n.d.]. https://scapy.net/

[4] Anton Cherepanov. 2017. WIN32/INDUSTROYER A new threat for industrial control systems. https://www.welivesecurity.com/wp-content/uploads/2017/06/Win32_Industroyer.pdf

[5] Mehmet Hazar Cintuglu, Osama A. Mohammed, Kemal Akkaya, and A. Selcuk Uluagac. 2017. A Survey on Smart Grid Cyber-Physical System Testbeds. *IEEE Communications Surveys Tutorials* 19, 1 (2017), 446–464.

[6] Webstore International Electrotechnical Commission. 2006. Telecontrol equipment and systems - Part 5-104: Transmission protocols - Network access for IEC 60870-5-101 using standard transport profiles. https://webstore.iec.ch/publication/3746

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.

[8] Andy Greenberg. 2017. Crash Override Malware Took Down Ukraine's Power Grid Last December. https://www.wired.com/story/crash-override-malware/

[9] Wired Andy Greenberg. 2017. How an Entire Nation Became Russia's Test Lab for Cyberwar. https://www.wired.com/story/russian-hackers-attack-ukraine/

[10] A. K. Jain, M. N. Murty, and P. J. Flynn. 1999. Data Clustering: A Review. *ACM Comput. Surv.* 31, 3 (Sept. 1999), 264–323. https://doi.org/10.1145/331499.331504

[11] Amit Kleinmann, Ori Amichay, Avishai Wool, David Tenenbaum, Ofer Bar, and Leonid Lev. 2017. Stealthy deception attacks against SCADA systems. In *Computer Security*. Springer, 93–109.

[12] P. Kundur, N.J. Balu, and M.G. Lauby. 1994. *Power System Stability and Control*. McGraw-Hill Education. https://books.google.com/books?id=2cbvyf8Ly4AC

[13] Subhash Lakshminarayana, E Veronica Belmega, and H Vincent Poor. 2019. Moving-Target Defense for Detecting Coordinated Cyber-Physical Attacks in Power Grids. https://arxiv.org/pdf/1908.02392.pdf

[14] Donghui Park, Julia Summers, and Michael Walstrom. 2017. Cyberattack on Critical Infrastructure: Russia and the Ukrainian Power Grid Attacks. https://jsis.washington.edu/news/cyberattack-critical-\infrastructure-russia-ukrainian-power-grid-attacks/

[15] Donghui Park, Julia Summers, and Michael Walstrom. 2019. Cyberattack on Critical Infrastructure: Russia and the Ukrainian Power Grid Attacks. https://jsis.washington.edu/news/cyberattack-critical-infrastructure-russia-ukrainian-power-grid-attacks/

[16] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20 (1987), 53 – 65. https://doi.org/10.1016/0377-0427(87)90125-7

[17] Steffen Schütte., Stefan Scherfke., and Michael Sonnenschein. 2012. MOSAIK - SMART GRID SIMULATION API - Toward a Semantic based Standard for Interchanging Smart Grid Simulations. In *Proceedings of the 1st International Conference on Smart Grids and Green IT Systems - Volume 1: SMARTGREENS,*. INSTICC, SciTePress, Porto, Portugal, 14–24. https://doi.org/10.5220/0003950100140024

[18] Joe Slowik. 2019. CRASHOVERRIDE: Reassessing the 2016 Ukraine Electric Power Event as a Protection-Focused Attack. https://dragos.com/wp-content/uploads/CRASHOVERRIDE.pdf

[19] Rui Tan, Varun Badrinath Krishna, David K.Y. Yau, and Zbigniew Kalbarczyk. 2013. Impact of Integrity Attacks on Real-time Pricing in Smart Grids. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security* (Berlin, Germany) *(CCS '13)*. ACM, New York, NY, USA, 439–450. https://doi.org/10.1145/2508859.2516705

[20] Mininet Team. 2017. Mininet. http://mininet.org/

[21] Robert L. Thorndike. 1953. Who belongs in the family? *Psychometrika* 18, 4 (01 Dec 1953), 267–276. https://doi.org/10.1007/BF02289263

[22] Joe Weiss. 2016. Aurora generator test. *Handbook of SCADA/Control Systems Security* (2016), 107.

[23] Wired Kim Zetter. 2016. Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid. https://www.wired.com/2016/03/inside-cunning-unprecedented-\hack-ukraines-power-grid/