Emulation of IEC 60870-5-104 Communication in Digital Secondary Substations*

Filip Holik[0000−0001−6595−0419], Doney Abraham[0000−0002−5760−5241], and Sule Yildirim Yayilgan[0000−0002−1982−6609]

Norwegian University of Science and Technology, 2815 Gjøvik, Norway
filip.holik@ntnu.no

Abstract. This paper describes two methods of emulation of digital secondary substations and their communication to the control center via the IEC 60870-5-104 protocol. The first method describes use of Mininet network emulator, which omits certain minor networking features, but can create the topology very efficiently. The second method describes use of virtual machines, which can be interconnected to achieve the full functionality including router devices and VPN connections.
An open source library libIEC60870-5 is used for communication between substations and the control center. The library is analyzed and compared to real traffic provided by Norwegian National Smart Grid Laboratory. Based on found differences, the paper provides information of how to modify the library in order to create messages identical to the real traffic. These messages can be used to verify the substation behavior, or for security penetration testing by creating messages with spoofed temperature or multimeter sensor values.

Keywords: Communication emulation · Digital secondary substations · IEC 60870-5-104 · libIEC60870-5 · Smart grid.

1 Introduction

Digital transformation of electricity grid continues as more areas are being updated in order to support smart grids. One of the most important elements in an electricity distribution network to undergo this transformation was digital substations (DS). It brought advantages such as effective real-time monitoring, higher resiliency, infrastructure simplification and cost reductions. The same process is now reaching secondary substations.

These secondary substations have much smaller scope in terms of transformed voltage, number of equipment and served area; but on the other hand, their number is an order of higher magnitude. Their digital transformation is therefore essential in order to create resilient and efficient smart grid [10].

* This work was funded by the Research Council of Norway, Innovation Project for the Industrial Sector - ENERGIX program, project number 296381 (Security of supply in smartgrids with interacting digital systems).

The main contribution of this paper is to propose and analyze methods to develop a model of a digital secondary substation (DSS) using real data communication. The term *emulation* is used for this method. Unlike *simulation* which simplifies the behavior. Emulated traffic uses real-like messages which can be appropriately recognized and handled by all networking devices. Such a model can then be used for behavior analysis and to verify security, which is one of the most important research question as the DSS different structure means new challenges and brings unexpected issues. Especially considering that these substations are enclosed in a single relatively compact block and therefore much easier to access by an attacker than DS with several access restriction techniques.

1.1 Digital Secondary Substations

Digital secondary substations (DSS) are transforming medium voltage to low voltage and are therefore most often located between DS and energy consumption (additional transformers can scale the voltage even further) [10]. DSS network topology is very simple when compared to DS as it is shown in Fig. 1.

The topology contains one gateway router connected to a remote terminal unit (RTU), which connects several transformer temperature sensors (TTS), low-voltage switchboard multimeters (LSM) and a door sensor. These devices can be connected via various protocols such as Modbus and the RTU acts as an interface between these protocols and communication to the control center realized by the IEC 60870-5-104 protocol (IEC104) [3]. No time-critical messages are being used to protect the grid function and the communication part within the DSS does not provide redundancy as in the case of a DS [10].

The connection out of the substation is typically redundant and is realized over a public network of an ISP. VPN is being used for encryption. The connection can use optic fiber or some form of wireless communication (most often cellular).

2 Related Work

Current work in the area of emulation approaches of DSS communication is very limited. To our best knowledge, only the work [13] presents a relevant model of limited parts of the communication between SCADA and RTU. Authors used Mininet for network emulation and Scapy tool for generation of selected IEC104 messages. Authors are working on extending the work to cover the entire IEC104 protocol and they are planning to publish the framework as open-source.

The only other relevant research is focused on the IEC104 protocol itself. Its behavior in terms of data flows, distribution of different message types and traffic frequency was analyzed in [7]. Description of communication scenarios with traffic measurements was analyzed in [9]. Protocol's security aspects were researched in [1, 12]. Authors in [1] proposed a multivariate Intrusion Detection System for anomalies detection which can signalize a Man-in-the-Middle attack as the protocol does not use any encryption. Paper [12] proposes a Coloured

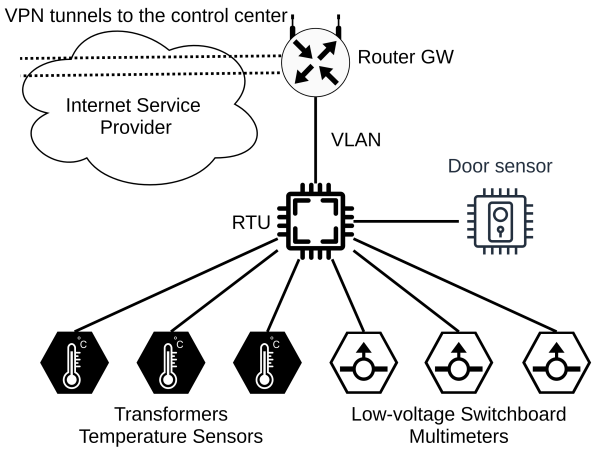

Fig. 1. Topology of a Digital Secondary Substation

Petri Net threat model and risk assessment for four types of attacks (ordered from the most to the least dangerous): unauthorized access, Denial of Service, Man-in-the-Middle, and traffic analysis.

The lack of further research confirms the importance and usefulness of emulation models described in this work.

3 The IEC 60870-5-104 Protocol

The IEC 60870-5-104 protocol (IEC104) [3] is mostly used for communication between the control center and DS (both primary and secondary) and is therefore build on top of a reliable TCP communication.

3.1 Message Types

The protocol uses the *Type* field to define three basic types of messages [4]:

1. Type U (0x03) - a fixed length message used for communication control. It has three subtypes: START (for connection initialization), STOP (for connection termination) and TEST (for verification of an active connection).
2. Type I (0x00) - a variable length message used for data transfer. It contains a payload in form of an Application Service Data Unit (ASDU).
3. Type S (0x01) - a fixed length message used for supervision and sending of acknowledgments.

3.2 Application Service Data Unit (ASDU)

ASDU contains payload of type I messages. The structure can vary based on the data type which is defined in the *TypeID* field. ASDU can be divided based on the communication direction [4]:

1. From control to monitors - includes two types of messages with the same *TypeID* (C_XX_XX_X): process information and system information. Process information includes single and double commands, step positions and set points. System information includes interrogation commands, reset, test, read and time synchronization.
2. From monitors to control - also includes process information (M_XX_XX_X) such as status values, measurements, step positions, etc.; and system information (M_EI_NA_1).
3. Bidirectional - includes control direction parameters (P_XX_XX_X) for modifying deadbands and a type for file transfers (F_XX_XX_X).

4 Emulation of Digital Secondary Substations Topology

This section describes two methods for creating an emulated topology of a DSS. For illustration purposes, examples show topologies of two DSS and a simplified control center.

4.1 Mininet Emulation

Mininet [8] is an open-source tool for creating a virtual network, which can contain hosts (end devices), switches and software-defined network controllers. Mininet uses a lightweight virtualization where all devices share the kernel with the host system. For this reason, Mininet is available only for Linux-based operating systems. This requirement can be avoided by installing Mininet within a Linux-based virtual machine (VM). This also allows an easy export of the entire model - including the DSS topology, the libIEC60870-5 library and any other software tools. This method is recommended and shown in Fig. 2.

Fig. 2 also shows mapping of DSS equipment to Mininet elements. RTU and routers are represented by switches, while sensors are represented by hosts.

The main advantage of the Mininet approach is low demand on computational resources [5]. Even a network topology with hundreds of devices can easily run on an average laptop. The modeled system can also be easily exported in form of a Python script. A script for this topology is provided on GitHub [2].

The main disadvantage of Mininet is simplification of certain aspects, mostly the inability to emulate the router functionality. Another problem is VLAN configuration on software switches. Solutions to these issues are described below.

Routing Simulation Routing can be ignored if all the topology devices are located within the same subnet. If the routing behavior is required, it can be simulated with the use of software-defined networking (SDN). An SDN controller can instruct switches of how to handle incoming messages.

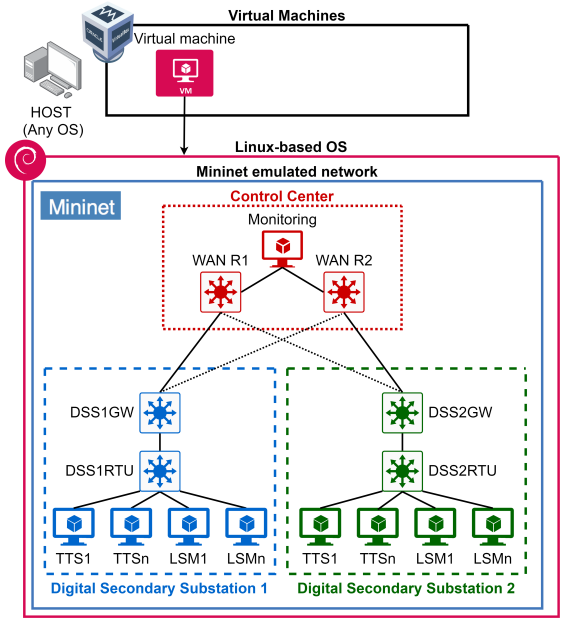

Fig. 2. Mininet Emulation of Digital Secondary Substations

VLAN Configuration Mininet does not support VLAN configuration commands for software switches. Configuration must be set outside the Mininet environment. The following example shows assignment of VLAN 10 to port 1 on switch 1:

```
sudo ovs-vsctl set port S1-eth1 tag=10
```

4.2 Full Emulation

This approach uses virtual machines for emulation of RTUs, routers and the monitoring control center host. Each device is implemented in one virtual machine and a virtual network is created for their interconnection. A hosting platform for this approach can use any virtualization tool (Oracle VM VirtualBox, VMware Workstation, OpenStack, etc). Oracle VM VirtualBox [11] was used in this work. The emulation schema is shown in Fig. 3. Sensor and multimeter devices are omitted from VMs as they would unnecessary increase the topology complexity. Their messages are generated by the RTU (on the figure shown with dash lines).

The main advantage of this approach is a possibility to fully emulate routers, which can be implemented as Linux-based hosts with appropriate tools, or as general boxes with router operating systems (for example pfSense, VyOS, OpenWrt and its variants). In case of the DSS emulation, these routers have to be configured with the following features:

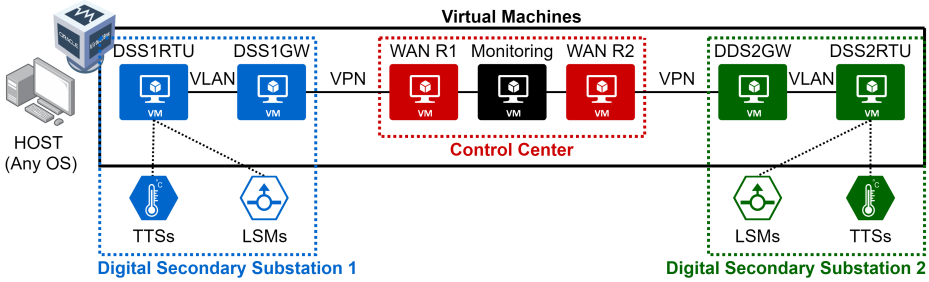

Fig. 3. Full Emulation of Digital Secondary Substations

- Routing - to provide connectivity, routers must have information about remote networks. This can be achieved by setting up a dynamic routing protocol (for example OSPF or BGP) or by static routing configuration.
- VLAN - a dedicated VLAN has to be created on the gateway router and assigned to the interface leading to the RTU.
- VPN - to emulate encrypted communication between DSS routers and a WAN routers, a site-to-site IPsec VPN must be set up.

The main disadvantage of full emulation are significantly more demanding computational resources as each device uses full virtualization. This approach is therefore more suitable for smaller topologies.

Network Emulation The network emulation varies based on the used virtualization technology. In Oracle VM VirtualBox, the *internal network adapter* option should be used to interconnect neighboring devices (for example the RTU to the router gateway). Name of the network must be the same on both devices. These interfaces then have to be configured within virtual machines (to set up theirs IP addresses, network masks and default gateways).

Optionally, an additional network adapter can be used for administration (*host-only adapter*) or for external access (*NAT*). The host-only network can be configured under the *File/Host Network Manager*, where an IP address of the host and a DHCP server can be set.

5 Communication Emulation

The library libIEC60870-5 [6] is used for communication emulation in both emulation methods. The library is written in C language and supports all major operating systems including Linux, macOS and Windows. This library must be installed on end nodes which varies based on the used emulation method. In both methods, the library can be downloaded from the official webpage [6].

5.1 Library Installation

Mininet Emulation End nodes (hosts) in the Mininet tool share the file system with the host linux-based operating system. The library must therefore be downloaded and decompressed only on the host system. All the Mininet hosts will then be able to access and use the library.

Full Emulation Emulated end nodes (RTUs and the control center) must use a supported operating system of the libIEC61850 library. This operating system should be lightweight, well documented and up to date. The library must be separately downloaded and decompressed on each emulated device.

5.2 Library Launch

The library contains an *examples* folder with pre-prepared scripts for emulating the traffic. To launch these scripts, executable files must be firstly created with the *make* tool. This tool can be simply launched with the command `make` executed either separately for each script file, or for the entire library in the root folder.

5.3 Devices Emulation

Sensors and Multimeters Scripts In a real network, RTUs create TCP/IP messages based on data received from the grid sensors. The `simple_server.c` script can be used to generate these messages. The script will start listening for a client connection on the default TCP port 2404. When a client connects, the script will start sending IEC104 messages. The following command starts the script (must be launched from the library folder):

```
sudo ./lib60870-C/examples/cs104_server/simple_server
```

Control Center Scripts A control center can be simulated by collecting the messages generated in substations. The `simple_client.c` script can be used. The following command starts the script (with up to two optional parameters for IP address and PORT number - they must correspond to the server):

```
sudo ./lib60870-C/examples/cs104_client/simple_client IP PORT
```

6 Library Evaluation

The topology presented in Fig. 3 was implemented according to the description in Sect. 5. The IEC104 traffic was then transmitted and analyzed by the Wireshark tool [15]. Emulated traffic was compared to real traffic provided by Norwegian National Smart Grid Laboratory [14]. This traffic is stated as "real traffic".

6.1 Default Script Analysis

Default library scripts (cs104_client/server) have a fixed sequence of messages, but the total number of sent messages can vary based on the synchronization mechanism and cycle sleep times. The entire communication takes around 9 seconds. A detailed time analysis and comparison of traffic was not conducted as IEC104 messages are not time critical (they use slower TCP) and they serve mostly for monitoring and control. Emulation of real latency on the WAN would not be possible as it varies case by case and in time. The following list shows the sequence:

1. Connection establishment - U messages (ACT and CON) are exchanged.
2. M_ME_NB_1 I messages are being sent with an S message acknowledgment returned after every 8th message. Messages contain a single scaled value element (starts with a value which is increased by 1 in every consequent message).
3. C_IC_NA_1 messages - Act and ActCon are exchanged.
4. M_ME_NB_1 I message is sent, but with "Inrogen" *CauseTx* instead of "Per/Cyc" and with 3 measured values elements (-1, 23 and 2300).
5. M_SP_NA_1 I message with two single-point information elements (0x01 and 0x00) is sent.
6. M_SP_NA_1 I message with eight single-point information elements (alternating 0x01 and 0x00) is sent.
7. M_BO_NA_1 I message with 32-bits string of value 0xaaaa0000 is sent.
8. C_IC_NA_1 I ActTerm interrogation message is sent.
9. M_ME_NB_1 I messages are sent in one second intervals. An S message acknowledgment is sent after the first message. Messages continue with the sequentially increasing single scaled value.
10. TCP closes the connection (FIN, ACK).

Real Traffic Behavior Real data traffic is exchanged based on a polling mechanism. Typically, an S message (can be accompanied by an I message) is sent to the monitoring device and the device then sends a measurement I message back. Transmission is not bounded by time as it runs indefinitely.

6.2 Messages Analysis

Both real traffic and default library scripts use all types of messages - U, I and S. The U, S, and control I messages are identical as they have the same format and same values. The only difference between real traffic and library scripts are monitoring I messages, which use different *TypeIDs*. Default library scripts use M_ME_NB_1 (measured scaled values), and M_BO_NA_1 (bitstring values), which were not present in the real traffic. The only common monitoring messages were M_SP_NA_1 (single-point information).

The following messages are transmitted by the real traffic, but they are not present in the default library scripts:

1. M_ME_NC_1 - short floating point number.
2. M_ME_TF_1 - short floating point number with a CP56 timestamp.
3. M_SP_TB_1 - single-point information with a CP56 timestamp.
4. M_IT_TB_1 - integrated totals with a CP56 timestamp.

The following section describes how to emulate these messages (and the common M_SP_NA_1 message) by modification of the default library scripts.

6.3 Real Traffic Messages Emulation

This section describes modifications of the default library script in order to create messages equal to the real traffic. The complete script is provided on GitHub [2].

The following code explains how to create and send a message and it is therefore common for all following specific messages.

```
/* Create an ASDU */
CS101_ASDU newAsdu = CS101_ASDU_create(alParams, false,
   CS101_COT_INTERROGATED_BY_STATION, 0, 1, false, false);

/* Create an Information Object and add it into the ASDU */
io = (InformationObject) ... // Varies based on the message type
CS101_ASDU_addInformationObject(newAsdu, io);

/* Send the message and deallocate */
InformationObject_destroy(io);
IMasterConnection_sendASDU(connection, newAsdu);
CS101_ASDU_destroy(newAsdu);
```

1. M_ME_NC_1 One of the real traffic messages has short floating point number of value 29.25 and IOA of 3. The following command shows an example of how to reproduce an equal message using the library. Important variables (IOA and the value) are shown in blue.

```
io = (InformationObject) MeasuredValueShort_create(NULL,
   3, 29.25, IEC60870_QUALITY_GOOD);
```

2. M_ME_TF_1 Includes the same information as in the case of M_ME_NC_1 messages, but adds the CP56Time2a timestamp. The current timestamp in miliseconds can be generated by calling the function `Hal_getTimeInMs()` located in the file: `/src/hal/time/unix/time.c`

Another difference is that the real traffic uses "Spont" (3) *CauseTx* while the library uses "Inrogen" (20) by default. This can be changed by replacing the ASDU function parameter. All of the available parameters are listed in the file: `/src/iec60870/cs101/cs101_asdu.c`

The following commands show, how to recreate the message:

```
/* Create the timestamp */
CP56Time2a timestamp = CP56Time2a_createFromMsTimestamp(NULL,
   Hal_getTimeInMs());

/* Create the ASDU with "Spont" CauseTx */
newAsdu = CS101_ASDU_create(alParams, false,
   CS101_COT_SPONTANEOUS, 0, 1, false, false);

io = (InformationObject)
   MeasuredValueShortWithCP56Time2a_create(NULL, 3, 29.25,
   IEC60870_QUALITY_GOOD, timestamp);
```

3. M_SP_TB_1 Uses the same CP56Time2a timestamp and "Spont" CauseTx. The following command shows the information object creation (with IOA 11).

```
io = (InformationObject) SinglePointWithCP56Time2a_create(NULL,
   11, false, IEC60870_QUALITY_GOOD,timestamp);
```

4. M_IT_TB_1 Uses the timestamp and data values formatted as "binary counter". The library implements *BinaryCounterReading* class located in the file: `/src/iec60870/cs101/cs101_bcr.c` The following commands show how to create a binary counter object and insert it into the information object with IOA 9:

```
/* BCR parameters: self, value, SQ, CY, CA, IV */
BinaryCounterReading bcr = BinaryCounterReading_create(NULL, 0,
   10, false, false, false);

io = (InformationObject)
   IntegratedTotalsWithCP56Time2a_create(NULL, 9, bcr,timestamp);
```

5. M_SP_NA_1 This message is created in the default library script, with 8 IOAs. The following command shows the modification needed to emulate the real traffic (IOA 10, SIQ 0x00):

```
CS101_ASDU_addInformationObject(newAsdu, io = (InformationObject)
   SinglePointInformation_create(NULL, 10, false,
   IEC60870_QUALITY_GOOD));
```

6.4 Discussion

The aforementioned library script modifications can create identical messages to real traffic. This has been proven by comparing the traffic in the Wireshark tool. Fig. 4 shows comparison of M_ME_TF_1 messages - real traffic and emulated using the script. It can be seen that the IEC104 messages contain same values

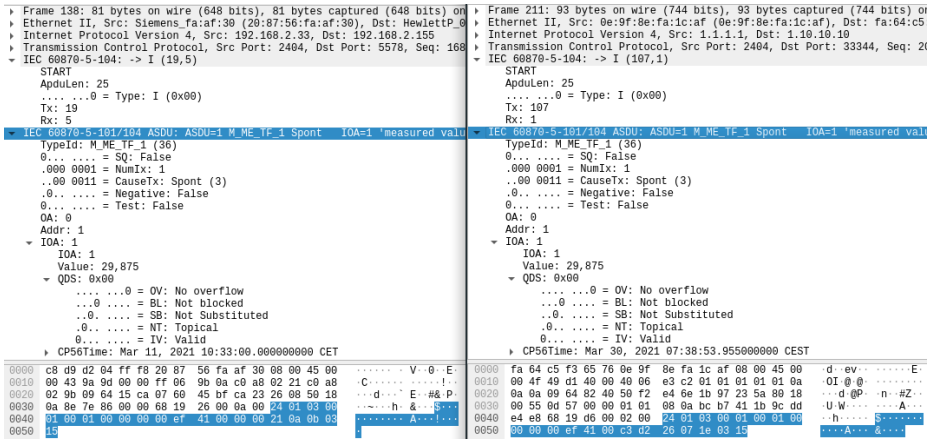

Fig. 4. Comparison of real (left part) and emulated (right part) traffic

except in variable fields such as Tx, Rx (depends on previously sent messages) and the timestamp.

These results show that the libIEC60870-5 library has a potential to create messages unrecognizable from real DSS messages. This fact can be used for various verifications before an expensive deployment within a real DSS. It also demonstrate how easily messages can be injected by an attacker if an access to a DSS is gained. Injection of messages with specific values can damage the grid equipment and can cause a local blackout.

The only limitation of the presented script is the need of Tx and Rx fields modification to correspond to the legitimate traffic in case of an attack simulation where the control center should not detect any suspicious activity.

7 Conclusion

The analysis of the libIEC60870-5 library has shown that the tool can be used to easily recreate messages with values corresponding to the real DSS or to purposefully create messages with a potential to cause damage. This can be used to verify security mechanisms or to test any other communication behavior without a need to use a real DSS network.

The paper did not cover any performance measurements as these are irrelevant in DSS. IEC104 messages are mostly used for monitoring purposes and not for real-time grid adjustments as is the case of GOOSE messages in DS. In our future work, we would like to target emulation of these messages including theirs performance comparison.

References

1. Grammatikis, P.R., Sarigiannidis, P., Sarigiannidis, A., Margounakis, D., Tsiakalos, A., Efstathopoulos, G.: An anomaly detection mechanism for iec 60870-5-104. In: 2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST). pp. 1–4 (Sep 2020). https://doi.org/10.1109/MOCAST49295.2020.9200285
2. Holik, F.: DSS (2021), https://github.com/filipholik/DSS, last accessed 26 Aug 2021
3. IEC 60870-5-104:2006: (2016), https://webstore.iec.ch/publication/25035, last accessed 01 Mar 2021
4. Introduction to the IEC 60870-5-104 standard: (2021), https://www.ensotest.com/iec-60870-5-104/introduction-to-the-iec-60870-5-104-standard/, last accessed 22 Mar 2021
5. Lantz, B., Heller, B., McKeown, N.: A network in a laptop: Rapid prototyping for software-defined networks. p. 19 (01 2010). https://doi.org/10.1145/1868447.1868466
6. libIEC61850 / lib60870-5: (2020), https://libiec61850.com/libiec61850/about/, last accessed 01 Mar 2021
7. Mai, K., Qin, X., Ortiz Silva, N., Cardenas, A.A.: Iec 60870-5-104 network characterization of a large-scale operational power grid. In: 2019 IEEE Security and Privacy Workshops (SPW). pp. 236–241 (May 2019). https://doi.org/10.1109/SPW.2019.00051
8. Mininet: (2018), http://mininet.org/, last accessed 25 Feb 2021
9. Musil, P., Mlynek, P.: Overview of communication scenarios for iec 60870-5-104 substation model. In: 2020 21st International Scientific Conference on Electric Power Engineering (EPE). pp. 1–4 (Oct 2020). https://doi.org/10.1109/EPE51172.2020.9269173
10. Omerovic, A., Vefsnmo, H., Gjerde, O., Ravndal, S.T., Kvinnesland, A.: An industrial trial of an approach to identification and modelling of cybersecurity risks in the context of digital secondary substations. In: Kallel, S., Cuppens, F., Cuppens-Boulahia, N., Hadj Kacem, A. (eds.) Risks and Security of Internet and Systems. pp. 17–33. Springer International Publishing, Cham (2020)
11. Oracle VM VirtualBox: (2021), https://www.virtualbox.org/, last accessed 05 Mar 2021
12. Radoglou-Grammatikis, P., Sarigiannidis, P., Giannoulakis, I., Kafetzakis, E., Panaousis, E.: Attacking iec-60870-5-104 scada systems. In: 2019 IEEE World Congress on Services (SERVICES). vol. 2642-939X, pp. 41–46 (July 2019). https://doi.org/10.1109/SERVICES.2019.00022
13. Salazar, L., Ortiz, N., Qin, X., Cardenas, A.A.: Towards a High-Fidelity Network Emulation of IEC 104 SCADA Systems. In: Proceedings of the 2020 Joint Workshop on CPS;IoT Security and Privacy. p. 3–12. CPSIOTSEC'20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3411498.3419969, https://doi.org/10.1145/3411498.3419969
14. Smart Grid Laboratory - SINTEF: (2017), https://www.sintef.no/en/all-laboratories/smartgridlaboratory/, last accessed 15 Mar 2021
15. Wireshark: (2021), https://www.wireshark.org/, last accessed 15 Mar 2021