An Attack-Trace Generating Toolchain for Cybersecurity Study of IEC61850 based Substations

Partha P. Biswas*, Yuan Li*, Heng Chuan Tan*, Daisuke Mashima*, Binbin Chen†*

*Advanced Digital Sciences Center †Singapore University of Technology and Design
{partha.b, yuan.li, hc.tan, daisuke.m}@adsc-create.edu.sg, binbin_chen@sutd.edu.sg

***Abstract*—The digitisation of modern power grid substations allows them to better support various advanced operations. However, it also poses greater risk of cyber-related attacks. There is an array of cybersecurity solutions (e.g., intrusion detection systems) available in the market to prevent, detect, or respond to cyberattacks. This calls for the creation of datasets and test cases for the validation of those cybersecurity solutions. In our recent work, we have generated a synthesized dataset for testing of cybersecurity solutions of IEC 61850 based substations. Our dataset contained traces for some typical attack-free disturbance scenarios and cyberattack scenarios in a substation. In this work, we present the toolchain we have developed to allow easy generation of such traces. We discuss the design considerations behind our toolchain and provide step-by-step guide to potential users on how to create customized trace files for specific scenarios using our toolchain. By open-sourcing the project for the broad community, we hope our toolchain will enrich the body of testing datasets for substation cyber security solutions.**

I. Introduction

Modern smart grids are more prone to cyberattacks than ever due to large scale deployment of information technologies, e.g., the use of IEC 61850 protocols [1] in modern substations. In recent years, we see considerable amount of efforts in studying various types of cyberattack scenarios to power grids and the development of cybersecurity solutions to mitigate those attacks [2], [3], [4]. In particular, electric substations, which convert voltage from one level to another, are important entities in the power grid. The growing dependence on digitization in electric substations brings forth challenges to the engineers and researchers to develop advanced security solutions (such as intrusion detection systems) and the IEC 62351 standard [5] for securing such modern substations. To evaluate these solutions in a comprehensive manner, we need to create and continuously refine the set of attack scenarios so that we can assess the effectiveness of the solutions against emerging attack vectors. To our best knowledge, however, there is still a lack of realistic substation datasets, at least in the public domain.

In our recent work [6], we made a systematic effort to generate a benchmark dataset for IEC 61850 communication-based substation cybersecurity testing. We specifically considered the Generic Object Oriented Substation Events (GOOSE) protocol as it was a key enabler in substation automation, protection, and control. Datasets for attack-free normal operations, one with disturbances (mainly the protection operation), and attack-induced scenarios have been carefully developed and made publicly available. However, the operational processes of substations are complex and may evolve as new use cases are developed. Furthermore, one substation's specific operational process may differ from the other ones. Hence, we envision that it will require continued efforts to generate a tailored dataset for each setting and configuration.

Motivated by this need, we have developed a toolchain that aims to enable users (e.g., power grid operators) to easily create customized datasets for the validation of cybersecurity solutions for IEC 61850 communication-based substations. Our toolchain consists of a set of modular software tools that work together to handle different inputs (e.g., substation configurations, attack configurations, and simulation settings) and carry out the necessary processing steps needed for generating the customized datasets. In this paper, we discuss the various components of the toolchain and how the toolchain can be utilized to generate customized datasets for test, research and validation purposes.

We hope that, by making the source codes of our toolchain available in public domain (in Github repository[1]), we can contribute to the enrichment of the body of datasets for designing, training, and evaluating smart grid cybersecurity solutions, which could in turn lead to more trustworthy power grids. Open-sourcing allows us (and the community) to further extend the toolchain capability. For example, our current toolchain implementation focuses on substation systems that do not support the IEC 62351 standard [5] (as most of today's systems fall under this category). When IEC 62351 compliant devices become more widely deployed, the open-source nature of our toolchain will allow it to be extended by the community to generate new attack traces for IEC 62351 compliant systems.

II. Related Work

Efficient energy management in the modern smart grid requires fast and secure transfer of information among various entities. Moreover, with the large scale deployment of distributed energy resources, real time control and monitoring of the network becomes necessary. A resilient and reliable communication network can fulfil the requirements of the smart grid. Thus, interdependence and close association of

[1]https://github.com/smartgridadsc/IEC61850ToolChain

power and communication network in smart grid require co-simulation platforms for simulation studies. Among recent works on co-simulation testbeds, Montenegro *et. al* [7] performed analytical simulation of a standard IEEE distribution network using OpenDSS in real-time hardware in the loop (RT-HIL) environment. RT-HIL was suggested as an efficient, scalable and flexible simulator that could generate signals by interacting with the external equipment. Ciraci *et. al* [8] proposed a framework to co-simulate both transmission and distribution level grids with the communication network. It integrated GridLAB-D, PowerFlow and NS3 into the framework, and also reduced the time overhead with a speculative synchronization strategy. A testing platform named as DSSnet was developed in [9] for planning and evaluation of the smart grid. The DSSnet combines a software-defined network (SDN) emulator with the distribution system simulator, ingrained with an efficient mean of time-synchronization. The co-simulation framework OOCoSim [10] combined OPNET and OpenDSS for dynamic co-simulation of both the constituents on the same computing platform. The OOCoSim was utilized to simulate the integrated scenario of network and power systems on control operation of the smart grid. A similar co-simulation platform based on OpenDSS and OPNET was tested on an IEEE standard bus system in [11]. The discrete-event network simulator NS3 for communication and MATLAB for power system were used to develop the co-simulation platform of smart grid cyber-physical system in [12]. In [13], efforts were also made to develop a framework for real time simulation linked to a graphical interface emulating the energy control center of a distribution network with distributed energy sources.

The Linux based testing platform for cyber-physical system (CPS) in [14] allowed physical computing for the cyber presence, while permitting offline simulation and computation for the physical world. Gunathilaka *et. al.* [15] presented a software-based testbed for evaluating security solutions of remote control interface for smart grid substations. For greater interest of the research community, the Iowa State University (ISU) cyber-physical system security testbed has been made remotely accessible [16]. The article [16] described the architecture, capabilities and challenges faced in allowing remote access of such testbed for CPS cybersecurity study. Precise timing of interactions between electrical and information network had been the focus in CPS modeling in [17]. A semantic analysis framework based on cyber and physical characteristics of the smart grid was suggested in [18] to detect malicious commands. Similar hybrid (i.e. both cyber and physics rule-based) network intrusion detection systems to detect attacks on digital relays [19] and automated power distribution systems [20] were also studied. Remotely issued control commands to substations were authenticated by incorporating a delay [21] and using the latency to simulate the power system dynamics [22] on execution of the commands.

In summary, the simulation of cyber-physical power system behavior performed in literature requires substantial resources and complex modeling, which may affect the usability acceptability, and reproducibility of the process in all scenarios. Our toolchain described in this article is more light-weight in terms of modeling complexity as it is intended for typical attack scenarios in a smart substation. The toolchain utilizes IEC 61850 standardized configurations as inputs, and its modular design makes it expandable for higher number of devices and various operating scenarios. The processes involved in the software are easy to understand, use and configure as needed for automatic generation of network traffic. However, in this tool, the impact on the electrical network is pre-defined for an attack. The generated attack-induced traces, including the resulting change in physical measurement, remain unchanged even if the attack packet is blocked by intrusion prevention system (IPS) functionality of the toolchain.

A major effort to secure IEC 61850 based substations is the IEC 62351 standard. The standard advocates the use of digital signature for authenticity and integrity of GOOSE messages in a IEC 61850 communication network. However, due to the challenge of today's cryptosystems on industrial control system devices to meet the delay requirements in time-critical substation automation operations [23], [24], the adoption of IEC 62351 remains a work-in-progress. Hence, our current toolchain implementation focuses on generating attack traces for systems that do not support IEC 62351. We envision our toolchain can be easily extended to support the IEC 62351 standard in the future. Indeed, our toolchain is capable to generate attack traces that bypasses some of IEC 62351 standard's recommended security controls, for example, the recommended checking regarding the value of the state number (stNum field) in GOOSE packets.

III. Review of Datasets for Cybersecurity Study of IEC 61850 based Substations

In [6], we consider a realistic 66/11kV distribution level substation as our model substation, based on which the datasets have been generated for cybersecurity study. A total of 18 IEDs (intelligent electronic devices), a minimum of one in each feeder, control and monitor the substation. The IEDs are interconnected via ethernet cables and communicate using IEC 61850 GOOSE protocol. All IEDs are in the same multicast group i.e., an IED can listen to the message broadcast by any other IED. IEC 61850 GOOSE communication has been the focus of the work as GOOSE is predominantly used for substation automation processes. In developing the network trace files, we have considered the following scenarios in the substation.

- Normal operation - In this scenario, the substation operates normally with all devices in service.
- Disturbance - This scenario does not account for any attack; however, the substation operation is not normal as one or more protection systems of the substation can trigger in response to faults. We have developed datasets separately for each of breaker failure, busbar protection, and under frequency loadshedding scenarios.
- Cyberattack - The datasets for this part consider cyberattack scenarios in the substation. Common GOOSE

TABLE I
COMMON CYBERATTACKS

Attacks	Tactics
DoS.1	• Flood bogus frames
MS.1	• Inject GOOSE sequence number (SqNum)
MS.2	• Inject GOOSE status number (StNum)
DM.1	• Modify current measurements reported by the merging units
DM.2	• Inject modified Boolean value of circuit breaker
DM.3	• Replay a previously valid message
CM.1	• Inject modified Boolean value of circuit breaker (from HMI)

related attack scenarios such as Denial of Service (DoS), Message Suppression (MS), Data Manipulation (DM) and Control Manipulation (CM) are framed in generating the attack trace files. Table I enumerates the sub-categories of attacks for which the attack trace files have been made available in the generated datasets.

IV. Toolchain Framework and Implementation

Fig. 1 shows the basic framework of the toolchain. It primarily consists of two parts - one part generates the attack-free network traces, while the other part generates the attack-induced network traces. The processes inside the blue-colored dotted rectangular box form the *Attack-Free Trace Generator*, whereas those inside the red-colored rectangular box are for *Attack-Induced Trace Generator*. Certain processes are common to both the generators, enclosed inside the black rectangle. The *Trace Generator* structural details are provided in Fig. 2. The framework is further elaborated in the following sub-sections.

A. Automatic Generation of Attack-Free Traces

The first step of generating any trace is to create meaningful IED models. The *SCL Conversion Tool*, written in JAVA language, parses the substation configuration language (SCL) which is the language and representation format specified by the IEC 61850 standard. Typically, such SCL files can be retrieved from the configured IEDs in the system. This process ensures generation of meaningful *IED models* and subsequently, the publication of logic codes in C programming language. The *Publish Logic Code* block is a code snippet to control the system publishing GOOSE messages. The snippet is embedded in the *Attack-Free Trace Generator*.

The creation of power system datalog is another parallel operation required for trace generation. The *Simulation Configuration* file stores launch time, published interface, and other high-level descriptions of attack-free scenario models. The *Simulation Configuration* file is also read by the independent *Power System Simulator*. The measurement and status of each IED at a certain timestamp are recorded in separate *Power System Data Log* files based on the simulation configuration. Each row of the log file represents a data vector of current, voltage, position (of breaker), etc., for a timestamp. The order of the stored power system data in a log file (corresponding to an IED) is kept consistent with the respective SCL file.

TABLE II
TYPES OF ATTACKS

Sl. No.	Attack Type	Description
1	Packet Modification	Intercept and modify attack-free packet payload and send out.
2	Malicious Packet Injection	Insert malicious packets into attack-free traces
3	Denial-of-Service	Launch DoS attack among attack-free traces

TABLE III
WAYS TO TRIGGER ATTACKS

Sl. No.	Method	Description
1	Time	Attack is triggered by program time
2	Packet parameter	Attack is triggered by certain packet parameters (SqNum & StNum)
3	Payload value	Attack is triggered by certain payload values

The *Attack-Free Trace Generator* calls the the code snippets (i.e., logic codes) and publishes the updated payloads by reading data from the *Power System Data Log*. The attack-free traces are automatically generated following the above-mentioned processes.

B. Automatic Generation of Attack-Induced Traces

The process of generating attack-induced traces is quite similar to the attack-free trace generation. IED models converted from SCL, *Publish Logic Code* block, and *Simulation Configuration* are necessary input files required by the *Trace Generator* like in the attack-free trace generation. Besides, an *Attack Scenario Configuration* is also loaded, which contains attack scenario settings such as the type and time of the attack. This toolchain supports three types of attacks - Packet Modification, Malicious Packet Injection and Denial-of-Service attacks, as described in Table II. Furthermore, there are three ways to trigger attacks - by time, by certain packet parameters, or by certain payload values. These are summarized in Table III. An attack is launched when the *Trace Generator* detects certain matched conditions.

C. Trace Generator Module

Trace Generator module is the heart of the toolchain. It loads the configurations and log files to generate and publish GOOSE messages. As shown in Fig. 2, there are multiple IED simulators inside the *Trace Generator* module. Each IED simulator compiles one given *Publish Logical Code* and loads specific IED model and *Power System Data Log* to behave like a real IED, ready to broadcast. Besides, each IED loads its own *Attack Scenario Configuration* to control the attack process. If this file is empty, the IED behaves as an attack-free IED; otherwise, it should simulate a specific attack behaviour based on the attack scenario configuration. The *Initialization* module launches all the IED simulators synchronously. It also handles configurations such as MAC addresses and broadcast interfaces for different IED simulators, depending on the input simulation configurations. Each IED simulator generates its

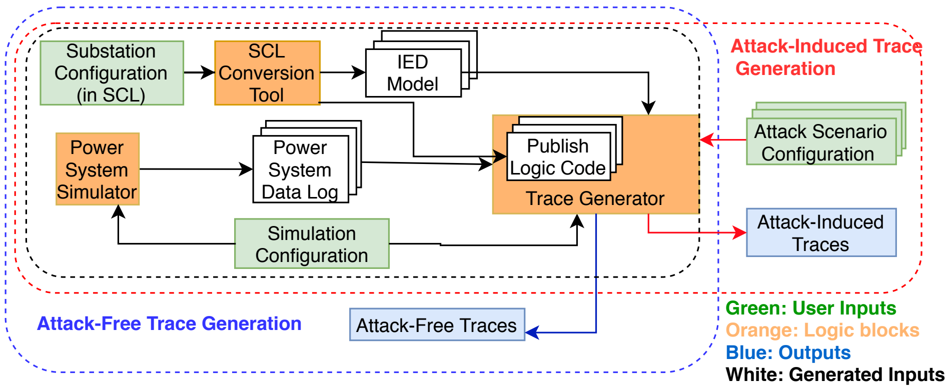


Fig. 1. Framework for generating attack-free and attack-induced traces

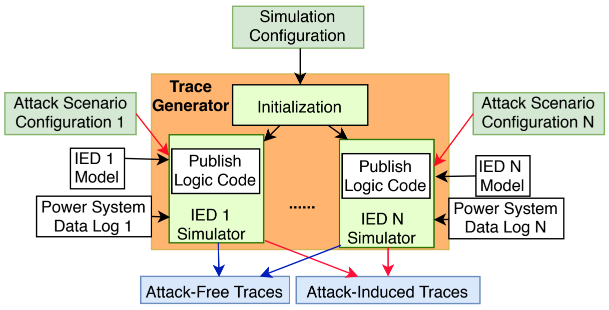


Fig. 2. Trace Generator structure

own attack-free or attack-induced GOOSE packets and the *Trace Generator* merges the packets together.

D. Implementation

The main processes of the toolchain, available in Github repository[2], have been implemented primarily in C and JAVA programming languages utilizing a third-party library called libIEC61850 [25]. To elaborate, the execution of toolchain processes requires calling several functions from libIEC61850. The *SCL Conversion Tool* is a JAVA program which generates *Publish Logic Code* and creates IED models in C language. The main module *Trace Generator* is a multi-threaded C program to simulate the behaviour of an IED. A python script is used in *Initialization* module to launch all IEDs synchronously and assign valid network interfaces based on *Simulation Configuration*, which is in XML format. *Attack Scenario Configuration* for an IED is also in XML format, while *Power System Data Log* is in CSV format. Wireshark is used to sniff and save attack-free and attack-induced traces. A virtual network interface card (VNIC) is employed to create a dedicated interface for each IED model.

V. Evaluation

In this section, we present a case study on how to use the toolchain to generate attack-free and attack-induced traces. We simulate a scenario where 50 IEDs are involved, 3 of which are compromised IEDs i.e., IEDs under the influence of different types of attacks such as injection attack, payload modification attack and DoS attack. The remaining 47 IEDs are operating normally (attack-free IEDs). The details of these IEDs, attack types and attack conditions are provided in Table IV.

[2]https://github.com/smartgridadsc/IEC61850ToolChain

A. Generation of Attack-Free Traces

As mentioned beforehand, a pre-defined SCL file is needed for automatically generating the traces. We take IED4 as an example here. The steps are to be repeated for all IEDs generating the attack-free traces. The IID file (instantiated IED description) of IED4 is fed into the SCL conversion tool to create the corresponding IED model for interfacing with the trace generator program. The steps observed to generate the attack-free traces are provided herein. The same shall be used as a guideline for attack-free trace generation.

1) *SCL Conversion Tool* is used to convert ’ied4.iid’ file into IED4 model. The output files obtained in the process are model.c and model.h.
2) IED4’s *Publish Logic Code* block is created also by *SCL Conversion Tool* to minimize hard coding. These code snippets are then embedded as source codes into the *Trace Generator* and compiled as an executable program.
3) Users must generate their own *Power System Data Log* file for each IED using any power system simulator (such as Power world [26] or Pandapower [27]). This file should be in CSV format, sample of which is available in the Github repository. The *Attack-Free Trace Generator* reads the data log file line by line and updates the GOOSE payload for each transmission. If this log is empty, *Attack-Free Trace Generator* will continue to send GOOSE packets with the default payload.
4) *Simulation Configuration* file specifies the network interfaces and port numbers for publishing GOOSE packets, including simulation start time and its duration.
5) After completion of aforementioned steps, we run the *Attack-Free Trace Generator* to generate the attack-free traces.

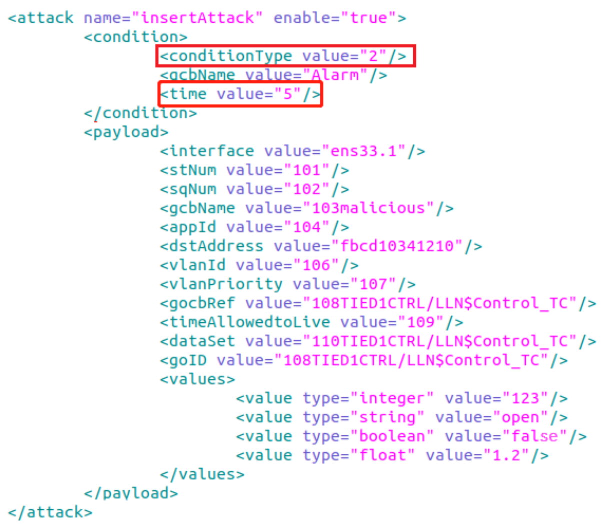
TABLE IV
CASE STUDY SCENARIO CONFIGURATION

IED No.	IED Type	Attack Type	Attack Condition
IED1	Compromised	Injection	Time=5s
IED2	Compromised	Modification	StNum=5, SqNum=0
IED3	Compromised	DoS	Fourth data in payload = 'true'
IED4-50	Attack-Free	–	–

B. Generation of Attack-Induced Traces

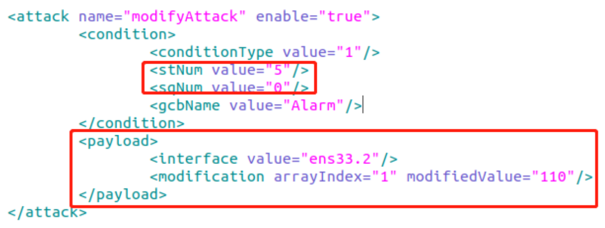
For generating the attack-induced traces, one needs to follow the same steps described in the previous section to generate the corresponding IED models for the compromised IEDs (i.e., IED1, IED2, and IED3). To incorporate the attack in traces, the user needs to define attack information in *Attack Scenario Configuration* for each of the compromised IEDs. The following presents step-by-step guidance for use of the toolchain to generate attack-induced traces.

1) **Define Injection attack on IED1:** As shown in Fig. 3, this attack is triggered at a certain time (e.g., at 5th sec.). The desired trigger condition is defined in 'condition' block of the configuration file. In this block, 'conditionType' indicates the way to trigger attack: value 1 - an attack triggered by StNum and SqNum; value 2 - an attack triggered by time; value 3 - an attack triggered by specific payload value. In addition, the GOOSE properties and related values for the inserted packets need to be defined in the 'payload' block. Those properties include state number (stNum), sequence number (sqNum), GOOSE control block name (gcbName), application identifier (appId), VLAN identifier (VlanId), VLAN priority (vlanPriority), GOOSE control block reference (gocbRef) and GOOSE identifier (goID) as per the standard.
2) **Define Modification attack on IED2:** This attack is triggered by specific StNum and SqNum values. The trigger conditions and modified values are provided in 'condition' and 'payload' blocks, respectively, as seen in Fig. 4.



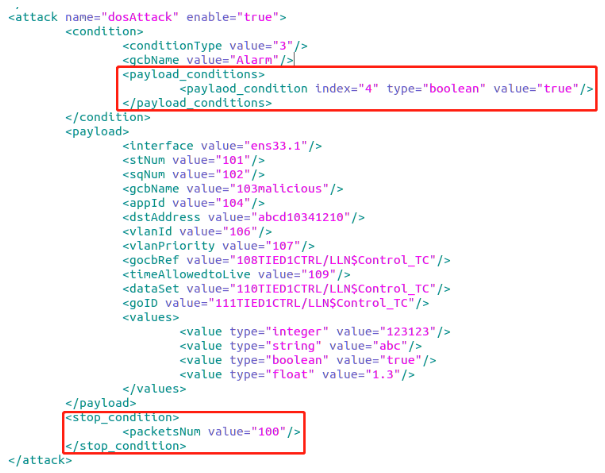
```
<attack name="insertAttack" enable="true">
        <condition>
                <conditionType value="2"/>
                <gcbName value="Alarm"/>
                <time value="5"/>
        </condition>
        <payload>
                <interface value="ens33.1"/>
                <stNum value="101"/>
                <sqNum value="102"/>
                <gcbName value="103malicious"/>
                <appId value="104"/>
                <dstAddress value="fbcd10341210"/>
                <vlanId value="106"/>
                <vlanPriority value="107"/>
                <gocbRef value="108TIED1CTRL/LLN$Control_TC"/>
                <timeAllowedtoLive value="109"/>
                <dataSet value="110TIED1CTRL/LLN$Control_TC"/>
                <goID value="108TIED1CTRL/LLN$Control_TC"/>
                <values>
                        <value type="integer" value="123"/>
                        <value type="string" value="open"/>
                        <value type="boolean" value="false"/>
                        <value type="float" value="1.2"/>
                </values>
        </payload>
</attack>
```

Fig. 3. Example of Attack Scenario Configuration for Injection attack



```
<attack name="modifyAttack" enable="true">
        <condition>
                <conditionType value="1"/>
                <stNum value="5"/>
                <sqNum value="0"/>
                <gcbName value="Alarm"/>
        </condition>
        <payload>
                <interface value="ens33.2"/>
                <modification arrayIndex="1" modifiedValue="110"/>
        </payload>
</attack>
```

Fig. 4. Example of Attack Scenario Configuration for Modification attack



```
<attack name="dosAttack" enable="true">
        <condition>
                <conditionType value="3"/>
                <gcbName value="Alarm"/>
                <payload_conditions>
                        <paylaod_condition index="4" type="boolean" value="true"/>
                </payload_conditions>
        </condition>
        <payload>
                <interface value="ens33.1"/>
                <stNum value="101"/>
                <sqNum value="102"/>
                <gcbName value="103malicious"/>
                <appId value="104"/>
                <dstAddress value="abcd10341210"/>
                <vlanId value="106"/>
                <vlanPriority value="107"/>
                <gocbRef value="108TIED1CTRL/LLN$Control_TC"/>
                <timeAllowedtoLive value="109"/>
                <dataSet value="110TIED1CTRL/LLN$Control_TC"/>
                <goID value="111TIED1CTRL/LLN$Control_TC"/>
                <values>
                        <value type="integer" value="123123"/>
                        <value type="string" value="abc"/>
                        <value type="boolean" value="true"/>
                        <value type="float" value="1.3"/>
                </values>
        </payload>
        <stop_condition>
                <packetsNum value="100"/>
        </stop_condition>
</attack>
```

Fig. 5. Example of Attack Scenario Configuration for DoS attack

3) **Define DoS attack on IED3:** This attack is triggered when the system detects a certain payload value (when the fourth data value in 'Alarm' packet equals to 'true', see Fig. 5). Besides trigger condition, DoS packet format and quantity are also specified. In this example, 100 packets will be broadcast to flood the network ('packetsNum' value = 100). Further, similar to the injection attack discussed earlier, the packets shall contain GOOSE properties and values in 'payload' block, as depicted in Fig. 5.
4) Post completion of the above steps, we execute *Attack-Induced Trace Generator* to create attack-induced traces.

The above attacks are generated following the exact protocol specification about stNum and sqNum transitions. To elaborate, for every GOOSE transmission, the sqNum is incremented by '1' when there is no new event. The stNum is updated only when there is a new event in the GOOSE packet, in which case, the sqNum is reset to '0'. Noted that even for IEC 62351 standard-compliant IEDs, if an attacker can bypass the digital signature checks (e.g., by stealing a key), the generated attack trace can actually bypass the standard's proposed checking for such manipulation.

C. Performance Evaluation

We evaluate the performance of our toolchain by simulating different numbers of IEDs (up to 50 IEDs) and observing the CPU utilization and memory usage statistics. A virtual

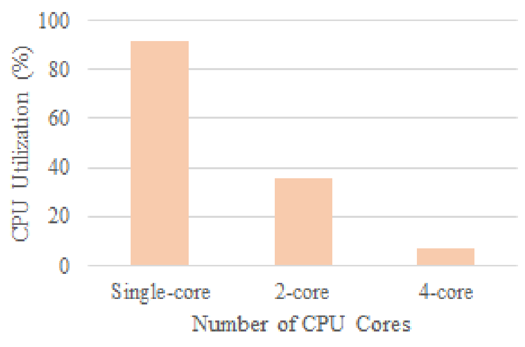


Fig. 6. CPU usage for 50 IEDs in multi-core machines

machine with Ubuntu 18.04.2 LTS operation system, single-core CPU and 13 GB RAM is used for this evaluation. Under same experimental set-up, i.e., for a specific number of IEDs, the CPU utilization and memory usage remain almost constant over time.

For normal system running (i.e., without the toolchain), the single-core CPU uses 5.83% of its capacity for operating system and other basic related tasks. When 50 IEDs are simulated in the toolchain, CPU utilization reaches 91.34%. With the increase in the number of IEDs, the CPU overhead rises super-linearly due to frequent switching of the CPU from one process to another. In terms of memory usage, the system uses 2158 MegaByte (MB) of memory before we start the toolchain simulation. When all the 50 IEDs send traffic in the toolchain, memory usage increases to 2331 MB. From our study on memory usage, we conclude that the memory usage increases almost linearly with the number of IEDs with each IED occupying around 3.4 MB of memory.

To demonstrate the scalability of our toolchain, we further repeat our experiments on multi-core processor settings, in particular, 2-core and 4-core CPUs. The CPU utilization rates for simulating 50 IEDs under these systems are 35.4% and 7.15%, respectively, as depicted in Fig. 6. We observe a drastic reduction in CPU utilization with the increase in the number of cores. This can be attributed to the fact that each core handles less number of IEDs and thus, the switching overhead of the CPU for simulating different IEDs reduces significantly.

D. Case Study Results

In this sub-section, we discuss the results of our case study. A 'pcap' file which contains traces of all the IEDs is the final output of this toolchain. The trace file accommodates both attack-free and attack-induced traces. Each IED is configured to send 3 GOOSE packets every second for 1 minute, as defined in its SCL model.

Fig. 7 shows a part of the resulting pcap file for injection attack. In Fig. 3, a malicious packet with defined payload (123, 'open', false, 1.2) is injected at 5.26 sec by IED1. This is triggered according to the 5 second condition as specified in IED1's *Attack Scenario Configuration*. Fig. 8 shows the results of the modification attack by IED2. The original packet with stNum=5 and sqNum=0 is modified at the first value in payload to '110' as shown in Fig. 4. The pcap file for DoS attack can be observed in Fig. 9. After the appearance of a packet with the fourth value as 'true' (as defined in Fig. 5), a stream of bogus packets is published to the network in a short time to achieve the DoS attack via IED3.

From the detailed evaluation of our toolchain performance, it is evident that the GOOSE packets generated are duly compliant with IEC 61850 standard. Further, the tooclhain is lightweight and simple to use. Given the SCL files for a substation/network, the user needs to define *Simulation Configuration* and generate *Power System Data Log* files to obtain attack-free traces. The attack trace generation requires defining additional *Attack Scenario Configuration* file in XML format. By changing these few inputs, the toolchain can be utilized to generate traces for desired scenarios.

VI. Conclusion

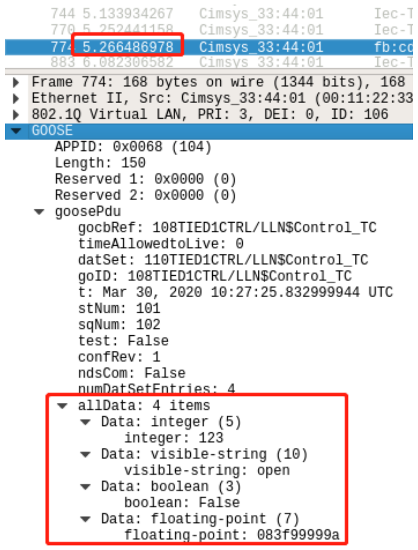
In this work, we present a toolchain to generate datasets for cybersecurity study of a smart substation. We provide a step-by-step guide on how to use our toolchain in generating datasets for IEC 61850 based substations under normal operations, under electrical disturbances, and under cyberattack scenarios. We consider some specific attack scenarios and describe the corresponding output obtained out of the toolchain. A substation may have a variety of configurations for normal operation. It may also experience various disturbances with the trigger of electrical protection and cyberattacks. Therefore, we believe that an open-soruce, easy-to-customize toolchain, as described in this work, will enable users to generate datasets according to specific scenarios of their concern. Our evaluations of the flexibility and performance of the toolchain demonstrate its capability of handling different scenarios and dozens of IEDs efficiently.

Acknowledgement

This research is supported in part by the National Research Foundation, Prime Minister's Office, Singapore under the Energy Programme and administrated by the Energy Market Authority (EP Award No. NRF2017EWT-EP003-047), and in part by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

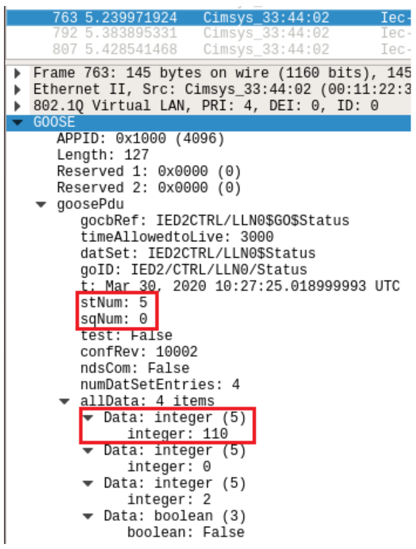
References

[1] "IEC 61850 - communication networks and systems in substations," May 2019. [Online]. Available: https://webstore.iec.ch/
[2] A. Gupta, A. Anpalagan, G. H. Carvalho, L. Guan, and I. Woungang, "Prevailing and emerging cyber threats and security practices in iot-enabled smart grids: A survey," *Journal of Network and Computer Applications*, vol. 132, pp. 118–148, 2019.
[3] H. He and J. Yan, "Cyber-physical attacks and defences in the smart grid: a survey," *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, no. 1, pp. 13–27, 2016.
[4] H. C. Tan, C. Cheh, B. Chen, and D. Mashima, "Tabulating cybersecurity solutions for substations: Towards pragmatic design and planning," in *2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*. IEEE, 2019, pp. 1018–1023.
[5] "IEC 62351:2020 - power systems management and associated information exchange - data and communications security - all parts," May 2020. [Online]. Available: https://webstore.iec.ch/publication/6912



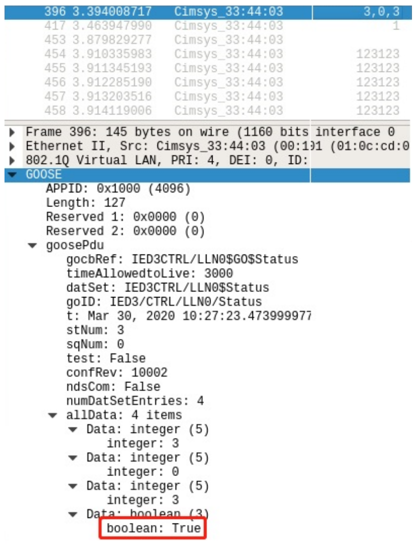
```
    744 5.133934267   Cimsys_33:44:01     Iec-T
    770 5.252441158   Cimsys_33:44:01     Iec-T
    774 5.266486978   Cimsys_33:44:01     fb:cc
    883 6.082306582   Cimsys_33:44:01     Iec-T
▸ Frame 774: 168 bytes on wire (1344 bits), 168
▸ Ethernet II, Src: Cimsys_33:44:01 (00:11:22:33
▸ 802.1Q Virtual LAN, PRI: 3, DEI: 0, ID: 106
▾ GOOSE
     APPID: 0x0068 (104)
     Length: 150
     Reserved 1: 0x0000 (0)
     Reserved 2: 0x0000 (0)
   ▾ goosePdu
        gocbRef: 108TIED1CTRL/LLN$Control_TC
        timeAllowedtoLive: 0
        datSet: 110TIED1CTRL/LLN$Control_TC
        goID: 108TIED1CTRL/LLN$Control_TC
        t: Mar 30, 2020 10:27:25.832999944 UTC
        stNum: 101
        sqNum: 102
        test: False
        confRev: 1
        ndsCom: False
        numDatSetEntries: 4
      ▾ allData: 4 items
         ▾ Data: integer (5)
              integer: 123
         ▾ Data: visible-string (10)
              visible-string: open
         ▾ Data: boolean (3)
              boolean: False
         ▾ Data: floating-point (7)
              floating-point: 083f99999a
```

Fig. 7. Case study result: Injection attack



```
    763 5.239971924   Cimsys_33:44:02     Iec-
    792 5.383895331   Cimsys_33:44:02     Iec-
    807 5.428541468   Cimsys_33:44:02     Iec-
▸ Frame 763: 145 bytes on wire (1160 bits), 145
▸ Ethernet II, Src: Cimsys_33:44:02 (00:11:22:3
▸ 802.1Q Virtual LAN, PRI: 4, DEI: 0, ID: 0
▾ GOOSE
     APPID: 0x1000 (4096)
     Length: 127
     Reserved 1: 0x0000 (0)
     Reserved 2: 0x0000 (0)
   ▾ goosePdu
        gocbRef: IED2CTRL/LLN0$GO$Status
        timeAllowedtoLive: 3000
        datSet: IED2CTRL/LLN0$Status
        goID: IED2/CTRL/LLN0/Status
        t: Mar 30, 2020 10:27:25.018999993 UTC
        stNum: 5
        sqNum: 0
        test: False
        confRev: 10002
        ndsCom: False
        numDatSetEntries: 4
      ▾ allData: 4 items
         ▾ Data: integer (5)
              integer: 110
         ▾ Data: integer (5)
              integer: 0
         ▾ Data: integer (5)
              integer: 2
         ▾ Data: boolean (3)
              boolean: False
```

Fig. 8. Case study result: Modification attack



```
    396 3.394008717   Cimsys_33:44:03          3,0,3
    417 3.463947990   Cimsys_33:44:03              1
    453 3.879829277   Cimsys_33:44:03
    454 3.910335983   Cimsys_33:44:03         123123
    455 3.911345193   Cimsys_33:44:03         123123
    456 3.912285190   Cimsys_33:44:03         123123
    457 3.913203516   Cimsys_33:44:03         123123
    458 3.914119006   Cimsys_33:44:03         123123
▸ Frame 396: 145 bytes on wire (1160 bits interface 0
▸ Ethernet II, Src: Cimsys_33:44:03 (00:1)1 (01:0c:cd:0
▸ 802.1Q Virtual LAN, PRI: 4, DEI: 0, ID:
▾ GOOSE
     APPID: 0x1000 (4096)
     Length: 127
     Reserved 1: 0x0000 (0)
     Reserved 2: 0x0000 (0)
   ▾ goosePdu
        gocbRef: IED3CTRL/LLN0$GO$Status
        timeAllowedtoLive: 3000
        datSet: IED3CTRL/LLN0$Status
        goID: IED3/CTRL/LLN0/Status
        t: Mar 30, 2020 10:27:23.473999977
        stNum: 3
        sqNum: 0
        test: False
        confRev: 10002
        ndsCom: False
        numDatSetEntries: 4
      ▾ allData: 4 items
         ▾ Data: integer (5)
              integer: 3
         ▾ Data: integer (5)
              integer: 0
         ▾ Data: integer (5)
              integer: 3
         ▾ Data: boolean (3)
              boolean: True
```

Fig. 9. Case study result: DoS attack

[6] P. P. Biswas, H. C. Tan, Q. Zhu, Y. Li, D. Mashima, and B. Chen, "A synthesized dataset for cybersecurity study of iec 61850 based substation," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2019, pp. 1–7.

[7] D. Montenegro, M. Hernandez, and G. Ramos, "Real time opendss framework for distribution systems simulation and analysis," in *2012 Sixth IEEE/PES Transmission and Distribution: Latin America Conference and Exposition (T&D-LA)*. IEEE, 2012, pp. 1–5.

[8] S. Ciraci, J. Daily, J. Fuller, A. Fisher, L. Marinovici, and K. Agarwal, "FNCS: a framework for power system and communication networks co-simulation," in *Proceedings of the symposium on theory of modeling & simulation-DEVS integrative*, 2014, pp. 1–8.

[9] C. Hannon, J. Yan, D. Jin, C. Chen, and J. Wang, "Combining simulation and emulation systems for smart grid planning and evaluation," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 28, no. 4, pp. 1–23, 2018.

[10] H. Kim, K. Kim, S. Park, H. Kim, and H. Kim, "Cosimulating communication networks and electrical system for performance evaluation in smart grid," *Applied Sciences*, vol. 8, no. 1, p. 85, 2018.

[11] X. Sun, Y. Chen, J. Liu, and S. Huang, "A co-simulation platform for smart grid considering interaction between information and power systems," in *ISGT 2014*. IEEE, 2014, pp. 1–6.

[12] Z. Pan, Q. Xu, C. Chen, and X. Guan, "Ns3-matlab co-simulator for cyber-physical systems in smart grid," in *2016 35th Chinese Control Conference (CCC)*. IEEE, 2016, pp. 9831–9836.

[13] H. Dharmawardena and G. K. Venayagamoorthy, "An on-line electric power distribution system simulator," in *2018 Clemson University Power Systems Conference (PSC)*. IEEE, 2018, pp. 1–8.

[14] C. Hannon, J. Yan, Y.-A. Liu, and D. Jin, "A distributed virtual time system on embedded linux for evaluating cyber-physical systems," in *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 2019, pp. 37–48.

[15] P. Gunathilaka, D. Mashima, and B. Chen, "Softgrid: A software-based smart grid testbed for evaluating substation cybersecurity solutions," in *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, 2016, pp. 113–124.

[16] A. Ashok, S. Krishnaswamy, and M. Govindarasu, "Powercyber: A remotely accessible testbed for cyber physical security of the smart grid," in *2016 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2016, pp. 1–5.

[17] J. Li, Y. Chen, and Y. Xia, "An event-driven modeling and simulation method for cyber-physical power system," in *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*. IEEE, 2017, pp. 1–6.

[18] H. Lin, A. Slagell, Z. Kalbarczyk, P. W. Sauer, and R. K. Iyer, "Semantic security analysis of SCADA networks to detect malicious control commands in power grids," in *Proceedings of the first ACM workshop on Smart energy grid security*, 2013, pp. 29–34.

[19] G. Koutsandria, V. Muthukumar, M. Parvania, S. Peisert, C. McParland, and A. Scaglione, "A hybrid network IDS for protective digital relays in the power transmission grid," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2014, pp. 908–913.

[20] M. Parvania, G. Koutsandria, V. Muthukumary, S. Peisert, C. McParland, and A. Scaglione, "Hybrid control network intrusion detection systems for automated power distribution systems," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 774–779.

[21] D. Mashima, P. Gunathilaka, and B. Chen, "Artificial command delaying for secure substation remote control: Design and implementation," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 471–482, 2017.

[22] D. Mashima, B. Chen, T. Zhou, R. Rajendran, and B. Sikdar, "Securing substations through command authentication using on-the-fly simulation of power system dynamics," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2018, pp. 1–7.

[23] "P1646 - standard communication delivery time performance requirements for electric power substation automation."

[24] E. Esiner, D. Mashima, B. Chen, Z. Kalbarczyk, and D. Nicol, "F-pro: A fast and flexible provenance-aware message authentication scheme for smart grid," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2019, pp. 1–7.

[25] "LibIEC61850/Lib60870-5: Open Source Libraries for IEC 61850," April 2019. [Online]. Available: https://libiec61850.com/libiec61850/about/

[26] "PowerWorld Simulator Overview," April 2020. [Online]. Available: https://www.powerworld.com/products/simulator/overview

[27] L. Thurner, A. Scheidler, F. Schäfer, J.-H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510–6521, 2018.