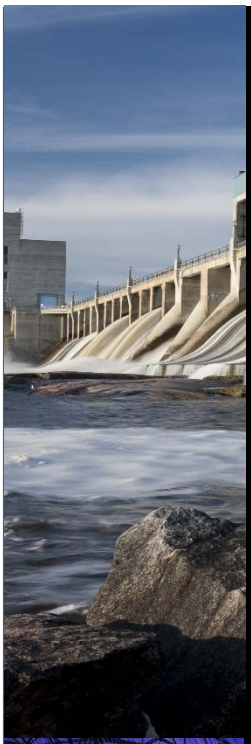
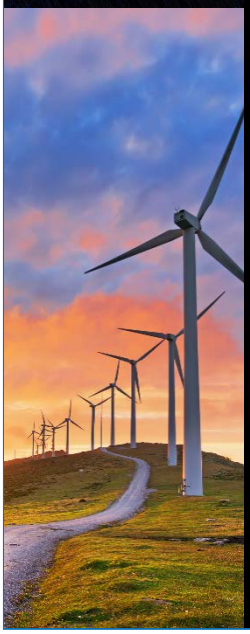




Rev: 26th October 2019

Using GTNET and NS-3 for Cyber-Physical Simulation

RTDS Technologies

Contents

1. Introduction to Cyber-security simulation for Power systems 3
1.1. Example scenario 3
1.2. Installing NS-3 5
2. System modelling 6
2.1. RTDS simulation system model 6
2.2. NS-3 Communication Network Simulation model 8
2.3. Interfacing RTDS with NS-3 9
2.4. Peak shaving application 10
2.5. Running Denial of Service (DoS) and Man-In-The-Middle attacks (MITM) 11
3. Running the case CyberSecSimPeakShaveUDP in RTDS 13
3.1. Changes needed in the Draft 14
3.2. Running the case in Runtime 15
4. Running the UDP Cyber-security simulation in NS-3 16
5. Running the case CyberSecSimPeakShaveTCP in RTDS 18
6. Running the TCP Cyber-security simulation in NS-3 19
6.1. Modifying the iptable rules 20
6.2. Make the rules permanent 20
7. Using NS-3 TAP to modify DNP3 messages 21
7.1. Example scenario: Control with DNP3 21
7.2. Running the simulation in NS-3 22
7.3. Stating the new values to be assigned for DNP3 23
7.4. Running the NS-3 DNP3 simulation 23
7.5. Setting and Monitoring DNP3 values 23
8. Changes made in NS-3.29 24
8.1. Writing an Application to Create a TCP server and Client 24
8.2. Writing an Application to Create a TCP SYN flood attack 26
8.3. Writing an Application to change TCP and UDP packets on the Wire 27
8.4. Modifications made in arp-l3-protocol to enable ARP spoofing 27
9. Summary 27
10. References 28

1. Introduction to Cyber-security simulation for Power systems

The traditional electric grid consisted of generation plants, transformers, tripping devices and feeders. The generation was mostly centralized, and the control was manual. The emergence of smart grid enabled two-way communication between the controller and assets in real time. This made it possible to include intermittent renewable energy resources such as wind and solar energy. Furthermore, it also enabled secondary energy markets, where the customer are paid for the limited energy they generate and the time shifting of their power consumption. However, this connectivity of controllers and assets through computer networks created the possibility of cyber-attacks on the power systems. The readers can refer [1] for a detailed survey of the possible attacks. In this report, we demonstrate how to simulate MITM attack and a DoS attack in an open source network simulator called NS-3. Then we show how to monitor the effects of these attacks in the power system using the RTDS™ with the aid of an example scenario. In this scenario, the Distribution system operator (DSO) makes use of secondary energy markets to carry out peak shaving.

1.1. Example scenario

In this report we are using a scenario that was presented in [2] at the Cyber-physical security for low-voltage grids website.

The Distributed energy resources (DERs) are customers that provide flexibility to the Distribution system operator (DSO) by mean of time shifting the load or providing small amounts of power. However, it is not feasible for the DSO to contract these DERs directly. Therefore, they make use of an entity called the Aggregator. The aggregators contract a portfolio of DERs. These aggregators in turn have contracts with the DSO. Whenever the DSO needs to shed some load these aggregators are informed of the shedding needed. The

aggregator operates the technical infrastructure to communicate with the DER units. These aggregators are financially responsible to the DSO in case the level of load shedding could not be delivered.

In this report, we are simulating a smart grid where the peak load of the distribution transformer is controlled by means of load shedding or increased generation at the DERs. In this scenario, the communication network consists of the following network connections:

- A remote terminal unit (RTU) and the DSO.
- DSO and the Aggregator
- The Aggregator and the DERs.

The RTU communicates data on the load level at the distribution transformer to the DSO. This communication network is modelled in NS-3. We can also model it in DeterLab which is a testbed containing actual computers. The power network related to this scenario is modelled in the RTDS.

Similar co-simulation setups have been used in the recent literature. In [2], Liu et al. conducted a co-simulation using RTDS and NS-3. In this simulation they made the state information of an IEEE 14-bus system available using the phasor measurement unit (PMU) and based on the information they performed closed loop control actions. Then the effect of DoS attacks of different magnitudes and MITM attacks were observed. In [3], Chen et al. analyzed the effects of cyber-attacks on the power system transient stability of bus voltage which uses a static VAR compensator. Here, they use an 11-bus test system modelled in RTDS. The cyber network was modelled in OPNET. Hahn et al. in [4] analyzed the cyber-physical impacts of malicious breaker tripping at generators observing the synchronicity of generator rotor angle, DoS attacks on the DNP3 servers and coordinated cyber-attacks. Here, the power system is modelled in RTDS and the cyber system is modelled in the PowerCyber testbed of the Iowa state university.

The remainder of this report is organized as follows. First, we introduce our system model which consists of the power system and the cyber-network with brief explanations on the

attacks that we are running. Next we explain how to connect the RTDS simulator to NS-3. Then we show how to simulate the cyber-physical system using the NS-3 simulator to model the network and attacks and RTDS simulator to model the power system

1.2. Installing NS-3

Follow the steps stated below to install NS-3 in you Ubuntu Linux PC.

1. In an Ubuntu Linux PC clone the git repository
   https://github.com/chamara84/ns3_cybersec.git

2. Installing the pre-requisites:
   - sudo apt-get install g++ python3
   - sudo apt-get install g++ python3 python3-dev pkg-config sqlite3
   - sudo apt-get install python3-setuptools git
   - sudo apt-get install qt5-default mercurial
   - sudo apt-get install python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev ipython
   - sudo apt-get install gir1.2-goocanvas-2.0 python-gi python-gi-cairo python-pygraphviz python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython ipython3
   - sudo apt-get install openmpi-bin openmpi-common openmpi-doc libopenmpi-dev
   - sudo apt-get install autoconf cvs bzr unrar
   - sudo apt-get install gdb valgrind
   - sudo apt-get install uncrustify
   - sudo apt-get install doxygen graphviz imagemagick
   - sudo apt-get install texlive texlive-extra-utils texlive-latex-extra texlive-font-utils dvipng latexmk
   - sudo apt-get install python3-sphinx dia
   - sudo apt-get install gsl-bin libgsl-dev libgsl23 libgslcblas0
   - sudo apt-get install tcpdump
   - sudo apt-get install sqlite sqlite3 libsqlite3-dev
   - sudo apt-get install cmake libc6-dev libc6-dev-i386 libclang-6.0-dev llvm-6.0-dev automake pip
   - python3 -m pip install --user cxxfilt
   - sudo apt-get install libgtk-3-dev
   - sudo apt-get install vtun lxc uml-utilities
   - sudo apt-get install libboost-signals-dev libboost-filesystem-dev
   - sudo apt install python3-pip
   - sudo apt install cmake

- pip3 install pygccxml
- pip3 install PyBindGen
- sudo apt-get install libxml++2.6-dev

3. export CXXFLAGS="-Wall"
4. Go to ./ns3_cybersec/ns-allinone-3.29
5. ./build.py

2. System modelling

The scenario is modelled in two parts. The first part is the power system model in the RTDS. The second is the communication network model in the NS-3. We are providing the details of these two models in the next two subsections. The power system model is shown in Figure 1.

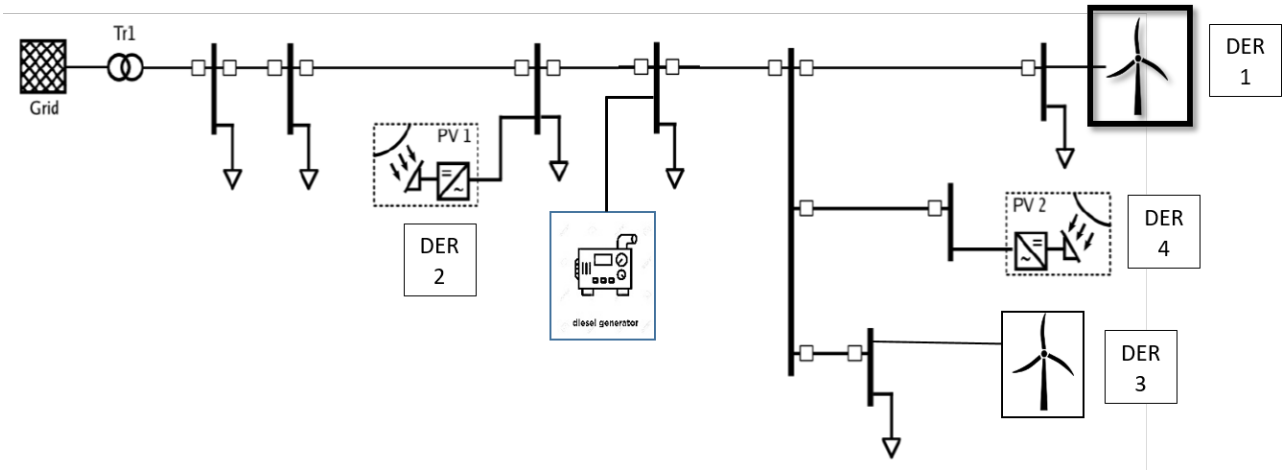


Figure 1 Power system model

2.1. RTDS simulation system model

Here, we are modelling a power system having four DERs. Each DER is represented with a Dynamic source and a Dynamic load. Then the grid transmission is modelled as a power source of 4kV. The Distribution transformer steps it down to 120V. The power factor was assumed to be 0.8 and we needed to keep the load of the distribution transformer at 6MVA. This load level is measured in real time using the P & Q meter model in the RSCAD.

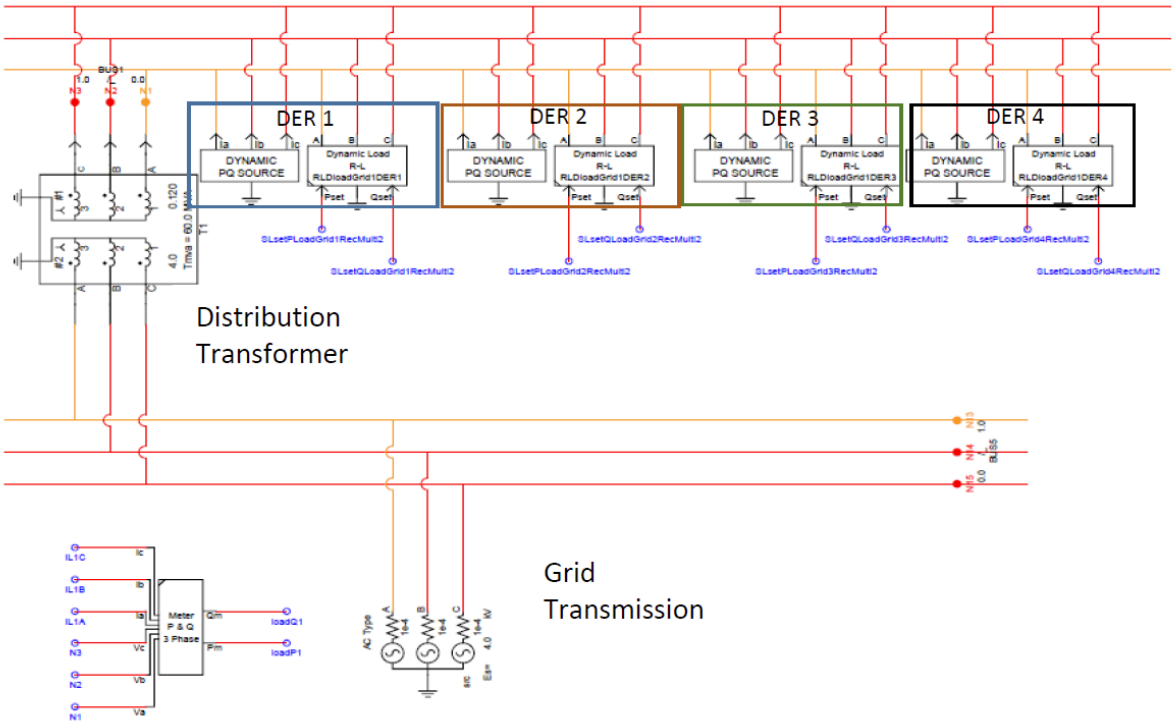


Figure 2 Power system simulation model

2.2. NS-3 Communication Network Simulation model

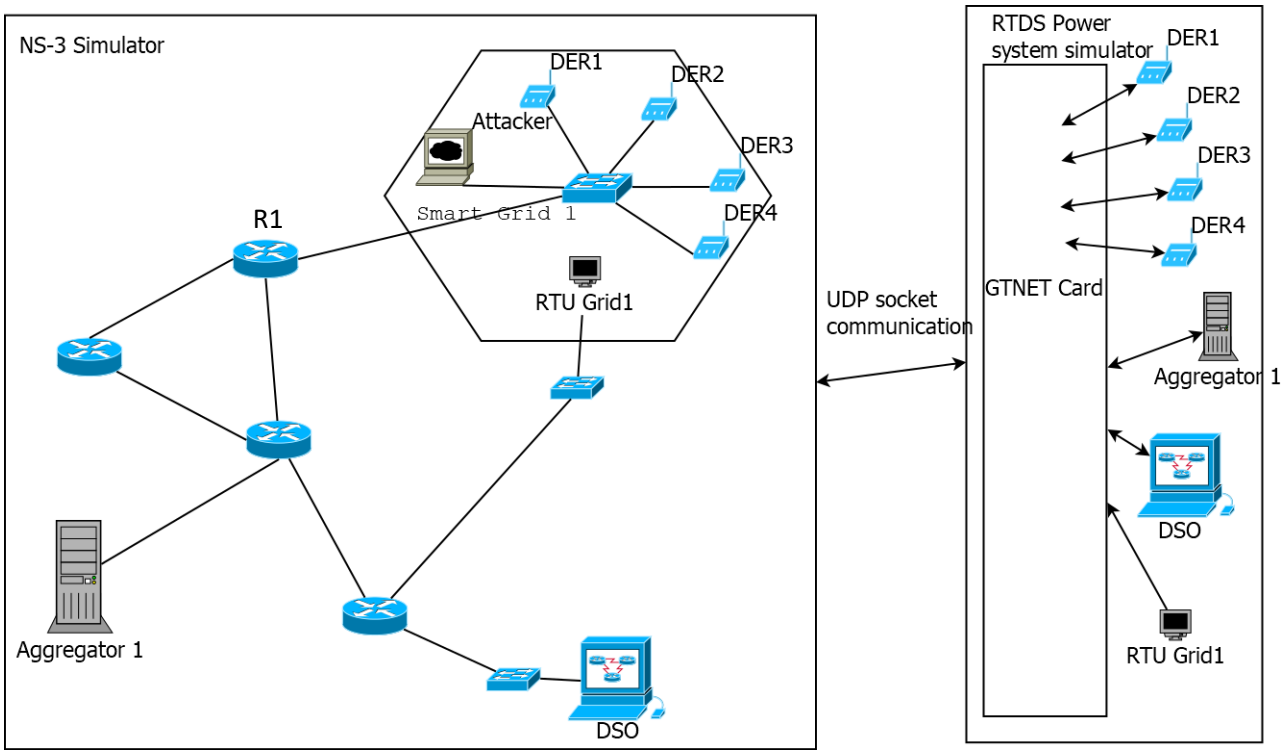


Figure 3 Network simulation model

We model all the network components using the Node type in the NS-3 network simulator. Then all the connections between nodes other than the DERs, the Attacker and the Switch in the Smart Grid1 are modelled as point-to-point links. The connection between DERS, Attacker and the Switch are modelled as a CSMA link. CSMA link is the closest possible connection type to the Ethernet connection in the NS-3. This Ethernet type connection is required to carryout the Man-In-The-Middle attacks using ARP spoofing. Router R1 acts as the gateway for the DERs to reach the Aggregator 1. We used the UDP sockets as the transport layer protocol for the communication. As one can see in Figure 3, the nodes DER1, DER2, DER3, DER4, Aggregator1, DSO and the RTU Grid1 exists in both NS-3 and the RTDS simulator. This is because the data is actually generated and consumed at the RTDS Simulator. NS-3 is just there to make the packets under go simulated network conditions before they reenter the RTDS simulator. We have made some modification to the original NS-3 version 3.29 so that it is able to carry out attacks such as ARP spoofing, TCP SYN flood, Capturing the packet at IP level and making modifications. These modifications are explained in the Section 7 and the source code can be made available for those who are interested. In the

Networking simulation conducted, we have used the emulated networking interface of NS-3.

2.3. Interfacing RTDS with NS-3

The NS-3 simulator need to run on a Linux machine, which has two or more network interface cards. The network connectivity between the RTDS and this Linux machine is achieved through the GTNET card of the RTDS simulator. The connection setup is shown in Figure 4.

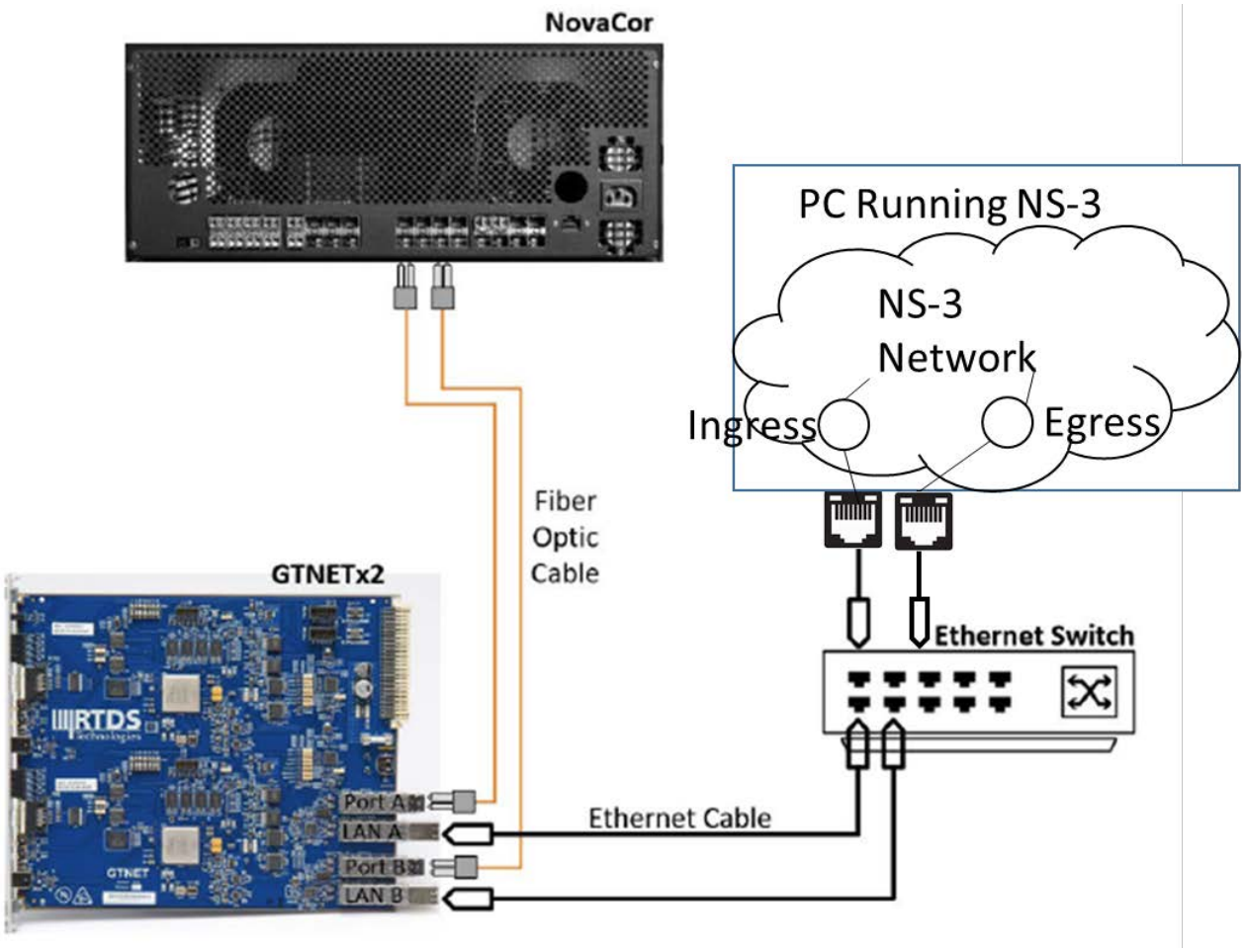


Figure 4 Interfacing the RTDS with NS-3

Here, as shown on Figure 4, the simulated nodes Ingress and egress are directly connected to each of the Ethernet ports on the machines. These nodes acts as the entry and exit points

to and from the simulated network. For an example if we want the data to traverse from DER1 to the Aggregator, the flow of data happens in the following order:

1. The Data is calculated at the RTDS simulator
2. GTNET interface corresponding to DER1 sends data to the Ingress interface
3. Ingress node forwards the data to the simulated DER1 Node
4. The data travers the network and gets to the simulated Aggregator node
5. Simulated aggregator node sends the data to the Egress node
6. Egress node forwards the data to the GTNET interface corresponding to the Aggregator node at RTDS simulator.

The other communication between the nodes happen in a similar fashion.

2.4. Peak shaving application

In this application all the calculations are performed using a script running in the RSCAD. The data flows of this application are as follows:

1. The control system reads measurement data from Remote Terminal Units (RTU) in the field (e.g. in substations) and delivers the data to the distribution management system (DMS).
2. A state estimator in the DMS calculates power flow estimates for all grid assets. If any of the assets are loaded above the limit, the DMS calculates the inverted difference as a reference signal.
3. The DMS sends a reference signal to one or several aggregators. In the case where several Aggregators are jointly providing the service, the signal will be split and be sent to all contracted aggregators corresponding to each aggregator's proportional share in the installed capacity or service commitment.
4. The aggregators requests flexibility information from all DER units in its portfolio.
5. The DER units respond with a flexibility prognosis.

6. The aggregator performs an internal optimization of its portfolio, in order to be able to deliver the service in the cheapest and most optimal way.
7. The aggregator sends set-points to all connected units and requests flexibility updates.
8. The DER units respond with an updated flexibility prognosis.
9. Smart meters at the DER owner provide measurements to the DSO.

In the simulation case, we simulate a system with a single Aggregator. The following calculations are carried out using the runtime script in the RTDS simulator:

1. Calculation of the reference signal in step 2
    a. Here we collect the information of the current loading by reading a meter in runtime in the script
    b. Then we calculate the level of overloading using a set threshold in the script
2. Calculation of the flexibility for each DER at step 5
    a. Here, we use a set percentage for the increment in power output and load reduction
3. Calculation of the set-points at the aggregator in step 6

2.5. Running Denial of Service (DoS) and Man-In-The-Middle attacks (MITM)

We developed some capability into NS-3 to run attacks such as DoS and Man-In-The-Middle. The DoS attack can result in the server going unresponsive to the legitimate traffic. Furthermore, the receive queues of the nodes are limited in size. Therefore the legitimate traffic might get completely dropped or get delayed. In the case of MITM, the attacker changes the packets with a malicious intent in mind. In our simulated scenario, the DoS attacks on the Aggregator resulted in the distribution transformer being overloaded for an extended period of time. In the case of MITM attack, we assumed a scenario where the attacker work in favor of DER1. Therefore, the attacker made changes to the flexibility information of the

other DERs such that they do not have any flexibility. This resulted in DER1 unfairly getting to sell all of its flexibility and earning the maximum.

2.5.1. DoS attacks in NS-3

The construction of DoS attacks in NS-3 differs based on the transport layer protocol used. For the UDP transport layer, the DoS attack is just one or many nodes sending a lot of bogus UDP traffic at a given port of the server, which is listening on that port. This makes the server overwhelmed with the traffic. In the case of TCP transport layer, DoS attacks are created by sending many SYN packets with bogus source IPs. The SYN packet is the first packet sent to establish a TCP session. Then the server allocates resources and send the SYN ACK packet. However, since the SYN packet has a bogus source IP there is no one to accept the SYN ACK. Then the server keeps this session open for a given time and then close it. When there are a larger number of packets like this, the server can go out of resources for the legitimate traffic. This type of DoS attack is called a SYN flood attack.

2.5.2. MITM attacks in NS-3

In the testing carried out, we used ARP spoofing in order to carry out MITM attacks. In order for the ARP spoofing to work, both the attacker and the victim should be in the same sub-net. ARP protocol maps the IP addresses to the Medium access control (MAC) address. When a node needs to send a packet to an IP address which is in its own sub-net it requests the MAC address of the node which bears the given IP address. This request is called the ARP request. This request is broadcasted in the sub-net. Then the node bearing that IP replies with the MAC address. This is called the ARP reply. This mapping between the MAC address and the IP address is stored in the ARP table of this node. These entries have a time of expiry.
In the case of ARP spoofing, the attacker listens to these ARP requests and replies with its own MAC address. Then the victim sends the data packets to the attacker without knowing that this is a malicious user. Then the attacker can either extract information before sending it to the intended user or it can modify or drop that packet. Another way of achieving the same result is sending unsolicited ARP replies or ARP requests. Figure 5 shows the first scenario.

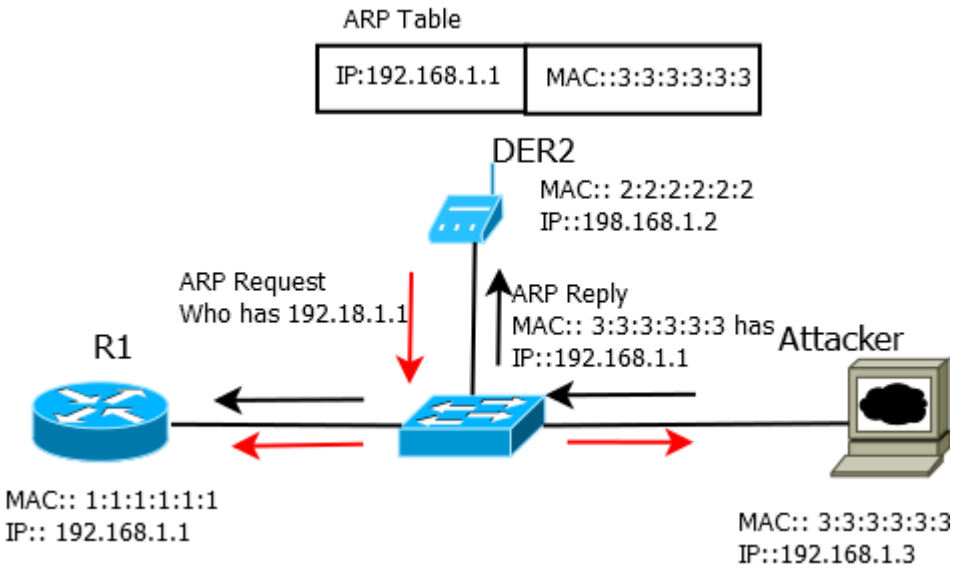
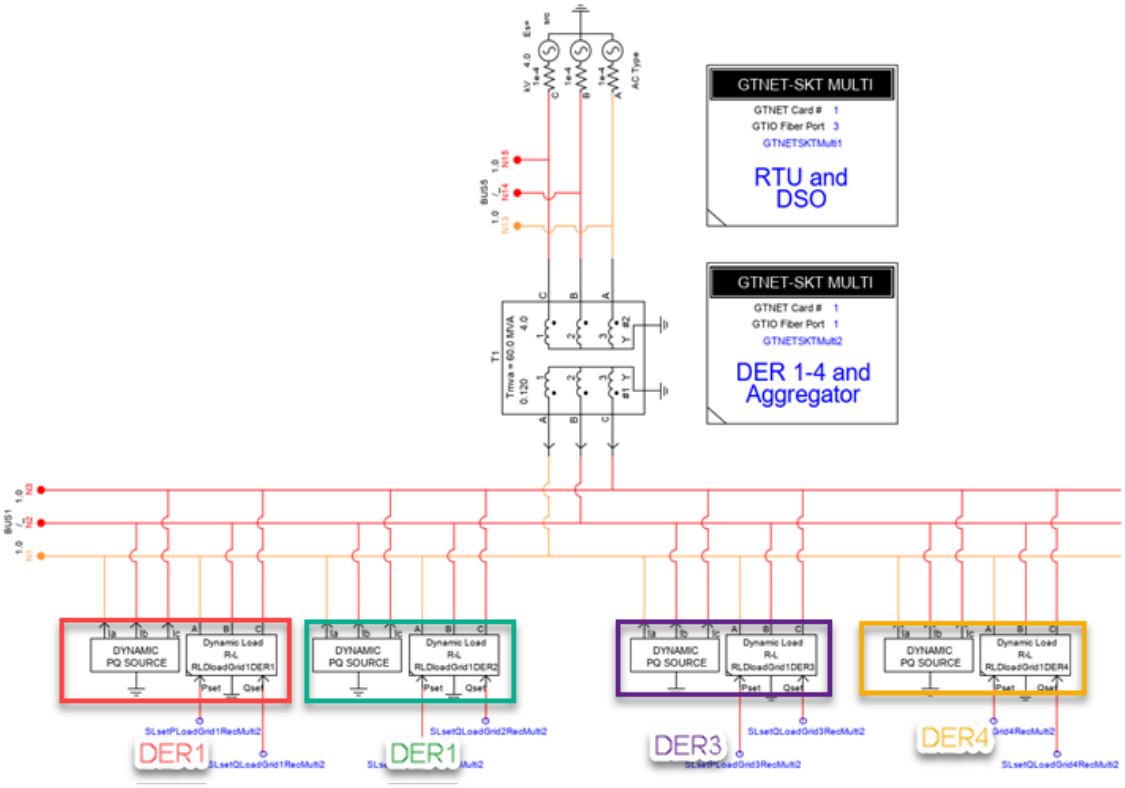


Figure 5 ARP spoofing

3. Running the case CyberSecSimPeakShaveUDP in RTDS

In this section, we describe the steps involved in running this case in the RTDS simulator. The sample case can be made available to those interested. This case uses the GTNET SKT-Multi firmware. Therefore, the SKT 1.19 firmware must be installed.



3.1. Changes needed in the Draft

We first discuss the changes needed in the GTNET card used by the RTU and the DSO. Here we used 2 channels. One used by the RTU and the other used by the DSO. For this communication, we did not use the NS-3 network as an intermediate hop. This communication was directly between the two GTNET channels where RTU is the client and the DSO is the server. This is not a limitation. We can use the NS-3 for this communication as well. We did this to lessen the complexity of the NS-3 simulation. Each of the channels of the GTNET card needs an assigned IP. These can be set at the Config file editor in the RSCAD main window. Let us assume that the IP address of the RTU is 172.24.9.249 and that of DSO is 172.24.9.250. Then, the remote address of the RTU should be set to the IP address of the DSO which is 172.24.9.250. The DSO is sending correction signal to the Aggregator. Therefore, the remote IP of DSO is set to the IP address of the Aggregator responsible for DSO. Let us call it Aggregator-DSO interface. Aggregator-DSO interface has the IP address 172.24.9.248 in the example case. The 3rd channel in the RTU and DSO GTNET interface is used to get the information on the time the distribution transformer stays overloaded. The information is captured in NS-3. The remote IP of this channel is set to the IP of the Ingress port which is 172.24.2.139 in the example case.

Now, let us look at the changes needed in the GTNET card used by the DERs and the aggregator. First the IP addresses should be set for the DERs. Let us assume we use the IPs 172.24.9.240-172.24.9.243 for the DERs. In order to be able to process this control scenario in the script we used five IP addresses for the Aggregator. Four of these are used for the communication with each DER and the other for the communication with the DSO. Had we did all this processing using a C++ code inside the aggregator node in NS-3 we could eliminate the need for all of these IPs since the aggregator node processing is no longer carried out using a script. The other way of doing this is aggregating all the DER flexibility. Let us assume the aggregator use 172.24.9.244 for the communication with DER1 (let us call it Aggregator-DER1), 172.24.9.245 for Aggregator-DER2, 172.24.9.246 for Aggregator-DER3, 172.24.9.247 for Aggregator-DER4 and 172.24.9.248 for Aggregator-DSO. Then let us assume that the IP address of the ingress interface of the machine running NS-3 is 172.24.2.139 and that of the egress interface is 172.24.2.102.
Since we need to send the traffic between the DERs and the Aggregator through the simulated network, we set the remote IPs of the DERs and Aggregator as follows:

- DER1 remote IP 172.24.2.139
- DER2 remote IP 172.24.2.139
- DER3 remote IP 172.24.2.139
- DER4 remote IP 172.24.2.139
- Aggregator-DER1 remote IP 172.24.2.139
- Aggregator-DER2 remote IP 172.24.2.139
- Aggregator-DER3 remote IP 172.24.2.139
- Aggregator-DER4 remote IP 172.24.2.139

The other parameters set in the GTNET multi are the variables received and transmitted. These are specified in the “From GTNET-SKT-x ” and “To GTNET-SKT-x” settings respectively. The ingress node is capable of finding out which DER or Aggregator it should forward the packet. This is made possible by including the DER index and the Aggregator index in the data sent over the GTNET card.

3.2. Running the case in Runtime

This case is run using the script provided. In the script we set the P and Q load and P and Q generation as shown in the below table.

Node name	P Load	P Generation	Q Load	Q Generation
DER 1	4.5 MW	3.0 MW	4.0 MVar	3.0 MVar
DER 2	4.5 MW	3.0 MW	4.0 MVar	3.0 MVar
DER 3	4.5 MW	3.0 MW	4.0 MVar	3.0 MW
DER 4	4.5 MW	3.0 MW	4.0 MVar	3.0 MVar

The P load at the Distribution transformer is maintained at 4.8MW and 3.6 MVar, using load shedding and increased generation at the DERs. The DSO send the correction signal to Aggregator to reduce the P load by 1.2 MW and Q load by 0.4 MVar. Each DER is assumed to be able to reduce the load by 80% and increase the generation by 20%. DERs send this information to the Aggregator. Then the aggregator calculates the set points and send them to the DERs. As mentioned before these calculations are carried out in the script. To repeat the process to get the average values we reset the values of the loads and generation of the DERs to the initial value after the script go through the loop twice in the RSCAD script. Furthermore, we pick DERs in random order in each iteration to use their flexibility. Therefore, every DER is able to sell their flexibility with equal opportunity. The time the transformer remain overloaded is indicated on the meter, timeOverloadPlot2, in runtime.

4. Running the UDP Cyber-security simulation in NS-3

We have created an example NS-3 scenario at “examples/RTDS-DoS-Simulatio/rtds-dos-simulation_UDP_BiDir.cc”. This scenario can be run by navigating to the “./ns-3-allinone/ns-3.29” folder and running the command below. However, remember that the IP addresses for DERs and Aggregators should be the ones that you set on the GTNET cards. The IP addresses for the Int1IP and the Int2IP are the IP addresses of the network interface cards of the PC running NS-3.

- NS_LOG="Icmpv4L4Protocol":"Ipv4Protocol" ./waf --run "rtds-dos-simulation_UDP_BiDir --stopTime=500 --DoSEnabled=false --ArpSpoofEnabled=false --IPDER1=172.24.9.10 --IPDER2=172.24.9.11 --IPDER3=172.24.9.12 --IPDER3=172.24.9.13 --IPAggreDER1=172.24.9.14 --IPAggreDER2=172.24.9.15 --IPAggreDER3=172.24.9.16 --IPAggreDER4=172.24.9.17 --Int1IP=172.24.2.101 --Int2IP=172.24.2.102 –

Int1MAC=08:00:27:48:57:c3 --Int2MAC=08:00:27:20:6d:04 --Gateway=172.24.0.1 -- Subnet=2 -- deviceName1=p1p1 -- deviceName2=p3p1”

Now, let us break this command into parts and see what each subpart does.

- NS_LOG="Icmpv4L4Protocol":"Ipv4Protocol" : This shows all the information on the Icmpv4 protocol and IPv4 protocol when NS-3 simulation enters the code related to those protocols
- ./waf --run "rtds-dos-simulation_UDP_BiDir: This part starts the simulation stated in the C++ code rtds-dos-simulation_UDP_BiDir.cc
- --stopTime=500 : These are the options. This option set the stop time of the simulation to be 500 seconds
- --DoSEnabled=false : When true this starts a DoS attack at 100 seconds and ending at 200 seconds.
- --ArpSpoofEnabled=false: When true this enables ARP spoofing and we set the flexibility of all the DERs except DER1 to 0.0
- --IPDERx=172.24.9.10 : Here x can be 1,2,3 or 4. This make the egress node send the data directed to DERx to this IP address.
- --IPAggreDERx=172.24.9.14: Here x can be 1,2,3 or 4. This make the egress node send to data directed to Aggregator-DERx to this IP address
- --Int1IP=172.24.2.101: This sets the IP address of the ingress node to this IP
- --Int2IP=172.24.2.102: This sets the IP address of the egress node to this IP
- --Int1MAC=08:00:27:48:57:c3: This sets the MAC address of the ingress node to the given MAC
- --Int2MAC=08:00:27:20:6d:04: This sets the MAC address of the egress node to the given MAC
- --Gateway=172.24.0.1 : This sets the gateway of the second interface
- --Subnet=2: This sets the subnet mask for the two NS-3 emu interfaces

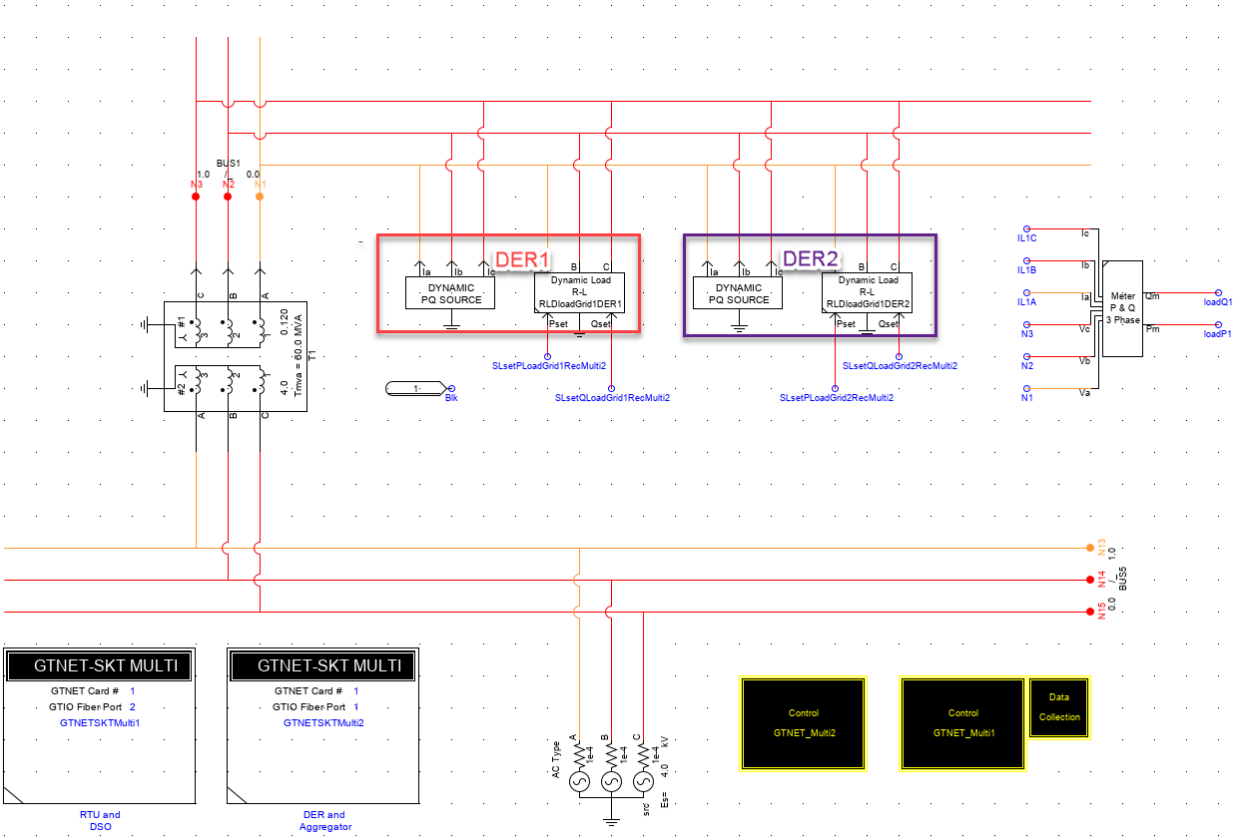
Running this command starts the ns-3 simulation. However, before this command is run both the network interfaces in the Linux machine running NS-3 should be in the promiscuous mode. This can be run by running the following Linux command.

- sudo ifconfig <interface_name> promisc

The interfaces can be listed using the command: ifconfig -a

5. Running the case CyberSecSimPeakShaveTCP in RTDS

This case is similar to the case where we used the UDP transport layer protocol. In this case, we used TCP only for the communication between DERs and the Aggregator, since we are planning to disrupt this communication inside the simulated NS-3 network using cyber attacks. However, when running TCP it was difficult to use a single IP address for both reception and transmission. Therefore, we reduced the case to have only two DERs. Each one of these DERs has one IP to listen to incoming connections and the other for initiating a connection. Similarly, we have two IPs for the Aggregator-DER1 interface and Aggregator-DER2 interface.



This case is run using the script provided. In the script we set the P and Q load and P and Q generation as shown in the below table.

Node name	P Load	P Generation	Q Load	Q Generation
DER 1	6.0 MW	3.0 MW	4.5 MVar	3.0 MVar
DER 2	6.0 MW	3.0 MW	4.5 MVar	3.0 MVar

The P load at the Distribution transformer is maintained at 2.4 MW and 1.8 MVar using load shedding and increased generation at the DERs. The DSO send the correction signal to Aggregator to reduce the P load by 3.6 MW and Q load by 1.2 MVar. Each DER is assumed to be able to reduce the load by 80% and increase the generation by 20%. The other details are the same as the case having the UDP transport layer protocol.

6. Running the TCP Cyber-security simulation in NS-3

We have created an example NS-3 scenario at "examples/RTDS-DoS-Simulation/rtds-dos-simulation_TCP_BiDir.cc". This scenario can be run by navigating to the ./ns-3-allinone/ns-3.29 folder and running the command below. However, remember that the IP addresses for DERs and Aggregators should be the ones that you set on the GTNET cards. The IP addresses for the Int1IP and the Int2IP are the IP addresses of the network interface cards of the PC running NS-3.

- NS_LOG="Icmpv4L4Protocol":"Ipv4Protocol" ./waf --run "rtds-dos-simulation_TCP_BiDir --stopTime=500 --DoSEnabled=false -- ArpSpoofEnabled =false --IPDER1C=172.24.9.10 --IPDER1S=172.24.9.11 --IPDER2C=172.24.9.12 --IPDER2S=172.24.9.13 --IPAggreDER1S=172.24.9.14 --IPAggreDER1C=172.24.9.15 --IPAggreDER2C=172.24.9.16 --IPAggreDER2S=172.24.9.17 --Int1IP=172.24.2.101 --Int2IP=172.24.2.102 --InterSynTime=1e-6 --maxParallelSessions=100 --Int1MAC=08:00:27:48:57:c3 --Int2MAC=08:00:27:20:6d:04 --Gateway=172.24.0.1 --Subnet=2 -- deviceName1=p1p1 --deviceName2=p3p1"

Here, the IPDERxS, where x is 1 or 2, is the server interface of DERs 1 and 2. The IPDERxC is the client interface of the DERs. IPAggreDERxS is the interface IP of the aggregator which listens to packets from the DERs. The IPAggreDER1C is the interface which sends packets to the DERs. The option InterSynTime defines the time between two consecutive SYN packets when running the SYN flood DoS attack. The option maxParallelSessions is the number of TCP sessions that can be served by the node concurrently. All the other options are same as the scenario with UDP transport layer protocol.

6.1. Modifying the iptable rules

The iptables in Linux is the firewall of the Linux machine. It is possible for it to block incoming tcp ports. Furthermore, it will send RST packets or icmp port unreachable packets when a SYN packet arrives to ports ns-3 nodes are listening to (ex: tcp port 7001). Therefore, the following commands should be executed (ignore the statements starting with #).

#Dropping RST

sudo iptables -I OUTPUT -p tcp --tcp-flags RST RST -j DROP

Accept the packets sent to tcp port 7001

sudo iptables -A IN_public_allow -p tcp -m tcp --dport 7001 -m conntrack --ctstate NEW,UNTRACKED -j ACCEPT

6.2. Make the rules permanent

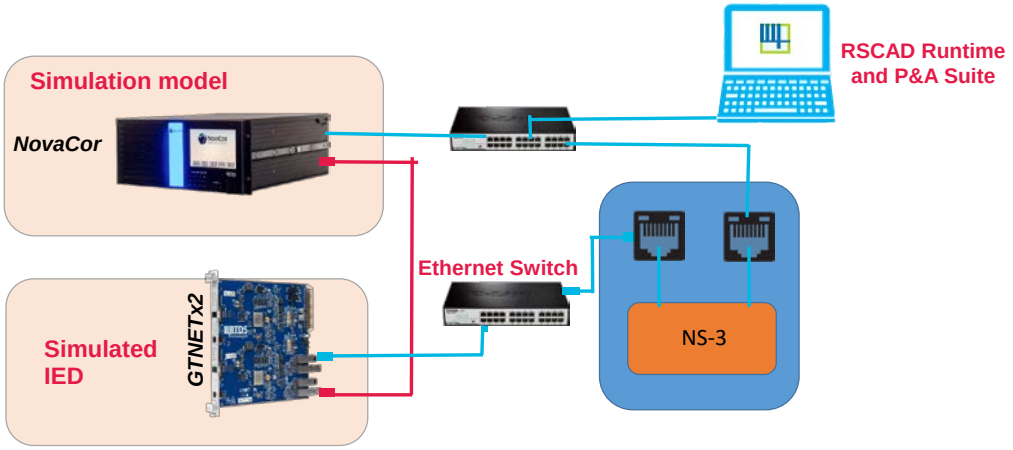
Install the iptables-save package to save the iptable rules to make them permanent using the following command.

sudo apt-get install iptables-persistent

After making changes to the iptables enter the following command to make the rules permanent.

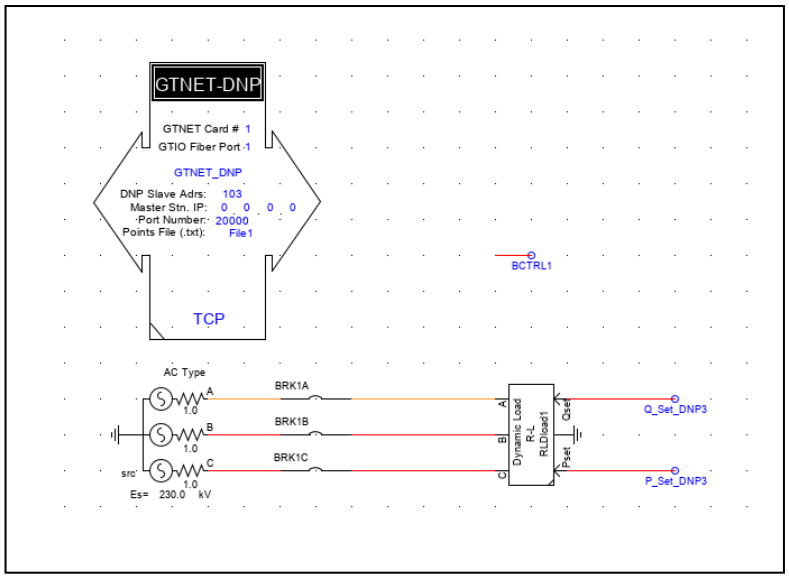
sudo iptables-save

7. Using NS-3 TAP to modify DNP3 messages



We use the above network topology when connecting the GTNET card to the P&A suite or any other controller. Here, the NS-3 simulation creates a simulated network between the GTNET card and the controller or the monitor. We can run Man-In-The-Middle attacks and Denial of service attacks within the NS-3 network.

7.1. Example scenario: Control with DNP3



Here, we have a simple network with a power source and a dynamic load with a breaker in

the middle. The user can control the breaker status, and the set active power and reactive power in the dynamic load using DNP3. Here we use the P&A suite to control these parameters.

7.2. Running the simulation in NS-3

This simulation is run using the TAP interface of NS-3. First we need to create two TAP interfaces and two bridge interfaces in the Linux machine running NS-3. Let the tap interfaces be tap0 and tap1 and the bridge interfaces be br00 and br01. Then, we bridge one of the physical Ethernet interfaces (say eth0) with tap00 using br00. Then we bridge the other physical interface (say eth1) with tap01 using br01. The set of commands to be entered are given below:

```
sudo brctl addbr br00
sudo brctl addbr br01
sudo ip tuntap add mode tap tap00
sudo ip tuntap add mode tap tap01

sudo ifconfig tap00 hw ether 00:00:00:00:01:20
sudo ifconfig tap00 0.0.0.0 promisc up
sudo ifconfig tap01 hw ether 00:00:00:00:01:21
sudo ifconfig tap01 0.0.0.0 promisc up

sudo ifconfig eth0 promisc
sudo ifconfig eth1 promisc

sudo brctl addif br00 tap00
sudo brctl addif br00 eth0
sudo ifconfig br00 up
sudo brctl addif br01 tap01
sudo brctl addif br01 eth1
sudo ifconfig br01 up
```

7.3. Stating the new values to be assigned for DNP3

The modifications to the DNP3 data are included in a file “ns3.conf”. This file should be located in /etc/ns3 in the Linux machine. The format of the text are given below

protocol dnp3

<function_code> <Group> <Variation> <Index> <value>

Here the file parser uses the protocol dnp3 to identify that the following data are for dnp3 packet modification. The next lines contain information on what to modify.

Function_code - The Function code of the dnp3 message of interest

Group - Is the DNP3 group number of the parameter

Variation - Is the DNP3 variation of the group of the parameter

Index - Is the index of the variable belonging to the stated group and variance

Value - Is the new parameter value to be assigned

7.4. Running the NS-3 DNP3 simulation

To run the NS-3 Tap simulation to modify DNP3 data, first navigate to the ns-3-allinone/ns-3.29. Then execute the following command:

NS_LOG="Icmpv4L4Protocol":"Ipv4Protocol" ./waf --run "rtds-Tap-ICS-Mod-One_Net“

This will start the NS-3 simulation.

7.5. Setting and Monitoring DNP3 values

In the runtime of the RSCAD simulation you are able to see the current that travels in the bus and the set active and reactive power of the dynamic load. Then in the P&A suite you are able to connect to the DNP3 server hosted on the GTNET cards. Furthermore, you can modify and monitor the parameters using DNP3 communication. If there is a man-in-the-middle attack on DNP3 communication, there will be a disparity between what you see in the runtime and in the P&A suite.

8. Changes made in NS-3.29

NS-3.29 is an open source network simulator. Therefore, it does not have built in support to run cyber attacks. However, it has all the necessary infrastructure to carryout some network level cyber attacks such as DoS attacks and MITM attacks. In this section, we discuss the additions we included in NS-3.29.

8.1. Writing an Application to Create a TCP server and Client

Although there are examples in the NS-3.29 on UDP client and server, there is no example on the TCP client and server communication. Here, we describe how an application involving TCP Client and server can be developed. A general guide on how to create new applications can be found in https://www.nsnam.org/wiki/HOWTO_make_and_use_a_new_application. In our current application, we are creating an application called MyApp which can accept, process and send TCP packets. The source code can be found in /examples/rtds-dos-simulation_TCP_BiDir.cc. The three way hand shake involved in a client sending a packet is handled automatically by the TcpSocketFactory. The only part we have to worry about is the acceptance of a packet in the server side. Every application has to implement three functions by default.

1. The Setup function
2. The StartApplication function
3. The StopApplication function

In the Setup function, we initialize the class variables. In the StartApplication function we run all the functions that are needed for the application to do its job. These functions continue to run until the function StopApplication is called. These two functions are called by the event scheduler when we set the start time and the stop time for the application at the caller. In addition to these functions, we have the following functions for the TCP application to process the packet data.

1. HandleAcceptRequest function
2. HandleAccept function
3. HandlePeerClose function
4. HandlePeerError function

5. HandleClose function
6. PrintTraffic function
7. pktProcessingIngressNode
8. pktProcessingEgressNode
9. pktProcessingAggregatorNode
10. giveParsingString

8.1.1. HandleAcceptRequest

This function is called when the TCP server gets a SYN request. Usually we do not need to do anything since the TCP protocol handles it.

8.1.2. HandleAccept

This function is called after the three way hand shake is done and we need to call the function to process the packet inside this function using callbacks. Furthermore, we can make the server drop the connection if the number of sessions are above a certain threshold.

8.1.3. HandlePeerClose

This function is called when the client send the FIN packet to close the socket connection.

8.1.4. HandlePeerError

This function is called when the client is in an error situation. We close the socket at the server at this situation.

8.1.5. HandleClose

This function is called when the server closes the connection.

8.1.6. PrintTraffic

This is one of the functions that processes the TCP data. This function is used to forward the data as it is the simulated Aggregator node in the NS-3 simulation or the actual DER IP at the RTDS simulator.

8.1.7. pktProcessingIngressNode

This function processes TCP data at the ingress node. The main requirement of this function is to figure out the simulated node to forward the data. This is achieved by processing the first three integers in the packet data. The first one is the message type, the send is the index number of the DER or the aggregator the message was initialized, and the third is the index number of the DER or the Aggregator the message is destined to. If the Message type is eight, the message carries the flexibility information. Then the second integer is the ID of the DER and the last integer is the ID of the Aggregator. If the message type is 7, the message carries the setPoints. The second integer is the Aggregator index and the third is the DER index the setPoints are destined to.

8.1.8. pktProcessingEgressNode

This function forward the data to the correct GTNET IP the packets are destined. If the message type is 7, these messages are sent to the correct DER GTNET IP based on the third integer entry in the packet. If the message type is 8, it is forwarded to the correct Aggregator GTNET IP based on the third integer in the packet.

8.1.9. pktProcessingAggregatorNode

This function forwards the setPoints to the correct simulated DER based on the second integer in the packet.

8.1.10. giveParsingString

This function returns a string array of the information presented in each 4 byte block of the data area of the SKT application based on the message type.

8.2. Writing an Application to Create a TCP SYN flood attack

The source code for this SYN flood attack can be found in /src/applications/model/tcp-syn-flood.cc. Here we provide a brief explanation on how the code works. As explained earlier, in SYN flood attacks the client send TCP session initialization messages using fabricated source IPs. As any NS-3 application, this application has the tree functions that initiate, start and stop the application. The respective functions are: Setup, StartApplication and StopApplication. In the Setup function, we initialize the NS-3 node the application is installed, the IP of the victim, the actual IP of the node, the TCP port that is exploited and the time between two consecutive SYN packets. In the StartApplication function, we create the socket and

schedule an event to call the SendSyn function. The SendSyn function is the function that uses the socket to send the SYN packets. Then we have the StopApplication function which deallocates the socket and clear the data.

In the SendSyn function, we generate a bogus source IP and create the TCP header and the IPv4 header and send the packet to the IP address of the victim.

8.3. Writing an Application to change TCP and UDP packets on the Wire

These modifications enabled MITM type of attacks. The source code for this application can be found in /src/applications/model/attack-app.cc. In the Setup function, we initialize the NS-3 node the application runs in, the network device, the IPv4 interface, the actual IP address of the node interface, the victims IP address and the victims MAC address. In the StartApplication function, we use the device receive callback function to get the packets on the wire. We set the callback to the function NonPromiscReceiveFromDevice. Then when ever a packet is received in that device this function gets called. In this stop application function, we stop this application.

Inside the NonPromiscReceiveFromDevice function we call the ReceiveFromDevice function. Here, we remove the headers and modify the data such that only DER1 has flexibility.

8.4. Modifications made in arp-l3-protocol to enable ARP spoofing

The source code that implement the ARP protocol is in /src/internet/model/arp-l3-protocol. This code in ns-3.29 does not accept unsolicited ARP replies. Therefore we modified the code to accept a flag m_spoofARP which enables the device to reply to any ARP request with its own MAC address.

9. Summary

In this short report, we discussed how to run a cyber-physical scenario using the RTDS simulator for the physical system and the NS-3 open source network simulator for the cyber system. We created a smart grid Distributed Energy resource scenario to simulate a case where the grid used these DERs to reduce the peak load on the network. This scenario used the SKT protocol to communicate the data and the control signals. The RSCAD draft files and the NS-3 source code can be made available to anyone who is interested in the topic.

10. References

[1] Elmrabet, Z., Ghazi, H.E., Kaabouch, N., & Ghazi, H.E. (2018). Cyber-Security in Smart Grid: Survey and Challenges. Computers & Electrical Engineering, 67, 469-482.

[2] M. Korman, E. Mathias, O. Gehrke, A. M. Kosek . D1.1 Smart grid scenarios, version 1.1. Retrieved from Cyber-physical security for low-voltage grids website: http://www.salvage-project.com/uploads/4/9/5/5/49558369/salvage_d1.1_v1.1.pdf

[3] R. Liu, C. Vellaithurai, S. S. Biswas, T. T. Gamage and A. K. Srivastava, "Analyzing the Cyber-Physical Impact of Cyber Events on the Power Grid," in IEEE Transactions on Smart Grid, vol. 6, no. 5, pp. 2444-2453,Sept.2015. doi: 10.1109/TSG.2015.2432013

[4] B. Chen, K. L. Butler-Purry, A. Goulart and D. Kundur, "Implementing a real-time cyber-physical system test bed in RTDS and OPNET," 2014 North American Power Symposium (NAPS), Pullman, WA, 2014, pp. 1-6. doi: 10.1109/NAPS.2014.6965381

[5] A. Hahn, A. Ashok, S. Sridhar and M. Govindarasu, "Cyber-Physical Security Testbeds: Architecture, Application, and Evaluation for Smart Grid," in IEEE Transactions on Smart Grid, vol. 4, no. 2, pp. 847-855, June 2013. doi: 10.1109/TSG.2012.2226919