Generating attacks on simple plant-controller networks

Ricardo Estrada
Capstone Advisor: Sathya Narayanan
CST401 Capstone Planning
Spring 2011
Cal State Monterey Bay
School of Information Technology and Communication Design
May 23, 2011
Table of Contents:
Abstract......................................................................................................................................................... 3
Executive summary........................................................................................................................................ 4
Project description ....................................................................................................................................... 5
Solution description ...................................................................................................................................... 8
Methodology ............................................................................................................................................... 10
Changes in project....................................................................................................................................... 11
Deliverables ................................................................................................................................................ 11
Budget ........................................................................................................................................................ 12
Time line ..................................................................................................................................................... 12
Evaluation .................................................................................................................................................. 15
  I.  Functional testing ............................................................................................................................ 15
  II.  Usability testing ............................................................................................................................ 16
  III.  Evaluation plan ............................................................................................................................. 17
  IV.   Results of testing ......................................................................................................................... 17
Collaboration statement .......................................................................................................................... 19
Final documentation ................................................................................................................................ 20
Abstract

In this paper, we address the vulnerability that a supervisory control and data acquisition (SCADA) system has against the threat of a distributed denial of service attack. This project takes an emulation based approach to mimic the routing and traffic associated with a local network topology between a controller and plant, in order to determine the effects of an inside attack on a critical infrastructure component. The goal is to develop several attack scripts and explore how a Distributed Denial of Service (DDoS) attack can affect the communication link between critical infrastructure components and their controllers. This capstone project produced several attack scripts that disrupt the communication channel between these components and causes instability with the plant, eventually leading to failure. This capstone project contributes to a group research project working on developing a set of algorithms designed to stop a denial of service attack. The content produced will enable future research with the DEfense Technology Experimental Research laboratory (DETER lab) testbed to further the exploration of concepts related to cyber security.
Executive summary

The objective of the research project that this capstone is a part of is to secure network control systems from the threat of cyber attacks. Network control systems monitor an entire system in real time, allowing the automation of industrial control systems. These network control systems, often referred to as SCADA systems, are composed of two main components. One component is the plant, which is responsible for running the industrial system itself. The other component referred to as controller, is responsible for receiving the data from sensors that monitor the plant. The controller then manages this data and controls the working state of the plants. This controller can manage multiple plants that may be spread across multiple geographical locations. Plants are still required to communicate their data to the controllers and must access the Internet to do so. It is at this point that these systems become vulnerable to the threat of cyber attacks.

To approach this problem, a research group with participants from multiple universities was set up to work on the various components necessary to help develop a set of Ordinary Differential Equations (ODE). These ODE’s will be collected in to an ODE library and will serve as a mechanism to detect when erratic behavior is occurring on the network and take appropriate action. The ODE library will take a sample of the network with normal traffic and use this as a baseline to determine when erratic behavior occurs on the network. Once the ODE library is employed on a test network, it will be able to detect abnormal traffic activity, which can be an indicator of a cyber attack; this will lead to rapid identification and reaction to an attack and decreasing the amount of damage an attack could have caused.

Since this project is a collaborative project with multiple researchers, each researcher is assigned a specific task to contribute to the project’s overall objective, which is to help create the set of ODEs for the ODE library. This capstone project will contribute scripts that will model cyber attacks in order to test the efficiency of the ODE algorithms. The first modeled cyber attack will consist of a cut link attack, which is designed to instantaneously cut the
communication link between plant and controllers. The next attack will be a flood link attack, which will flood the communication channel, this attack is intended to flood the channel and deny legitimate traffic from reaching its destination. The last script is designed to use several computers to emulate a DDoS attack by slowly increasing the amount of traffic on the communication channel of the intended target. It is designed to effectively cut the communication channel between the plant and controller by flooding the link and not allowing legitimate traffic from reaching its destination, resulting in a denial of service. These scripts will be tested by using the Defense Technology Experimental Research laboratory testbed (DETER lab), and will test the efficiency of the ODE library. The overall objective of this research effort is to develop and effectively employ these attack scripts and make them available for future use by researchers.

**Project description**

A system that collects data from various sensors at a factory, plant or in other remote locations and then sends this data to a central computer, which then manages and controls the data is known as a Supervisory Control and Data Acquisition (SCADA) system. This system has two components, the plant/factory, and the central computer (controller). The central computer manages the incoming data and controls the working states of the plants. The controller uses the received data to perform other tasks, but also to maintain stability within that plant; For example, a plant may be having some software malfunction, the plant will send its current state to the controller, the controller will use this data to instruct the plant to perform some event in order to prevent failure of the plant. Some examples of where SCADA systems can be found in the industry range from water management, power, electric, traffic signals, mass transit, environmental control, and manufacturing systems. Since controllers tend to manage multiple plants/factories, they tend to be outside of the network of the plant. Therefore, plants will have to send their data over the internet to reach the controller, and it is at this communication link that these systems are vulnerable to cyber attacks.
These systems and their networks are vulnerable to cyber attacks and little attention has been paid to investigating or developing algorithms for mitigating such attacks. Some of the systems at the controller stations are using Windows or Linux operating systems, making them vulnerable to operating system or application exploits. IT administrators are aware that there are constant updates that come up for these operating systems and a worry is that installing these updates on such critical machines could make them unstable or potentially cause that system to crash. A crash on these systems cannot be afforded and IT administrators may not install that update, however, leaving that system un-patched and out-dated will leave it vulnerable to a cyber attack that the patch would have been able to protect against. If an attack where to successfully bring down one of the mentioned critical systems that you and I use on a daily basis (water, electricity, gas, etc...), then those who use that service, which are many, would be left without it. If power was to be attacked and the attack caused catastrophic failure, then the schools, hospitals, emergency services, you, and I would feel the effects of this attack. The threat of cyber attacks towards these SCADA systems is the issue we are trying to address with this project, since they threaten the services that we rely on to function in society.

There has been similar research projects carried out before that addresses the security vulnerabilities of industrial systems. However, previous research projects addressed this problem using either a mathematical model or a simulation approach to get the results needed. The goal of the research effort is to take an emulation based approach to address this problem. Emulation is a hybrid approach that subjects real applications, protocols, and operating systems to a synthetic network environment. This approach is a blend of physical machines with a wide range of operating systems and the usage of real protocols, all carried out in a simulated network environment. This adds to the realism of the experiments and also gives us much more precise results when compared to mathematical or simulation results.

The objective of this capstone project is to produce a series of attack scripts that can be applied towards simulating an attack on another system. These scripts have been developed
and tested, using the DETER lab testbed to emulate the events scripted. Once the ODEs have been developed, their efficiency can be tested against these attack scripts.
Solution description

The overall group project provides a solution to the threat of a denial of service attack by using a set of algorithms to detect when the attack is occurring. The objective will be to produce a series of Ordinary Differential Equations (ODE) that will be used to monitor the network traffic of the plants and controllers and use it to detect when a Distributed Denial of Service attack occurs on the network. The implementation of this ODE algorithm is to have it detect the attack while its occurring and react on the first sign it detects some sort of “erratic” behavior. A sample of normal network activity would be captured and the equation would use this baseline to determine when the network is being erratic, such as an increase of traffic during a time when traffic should not be increasing; for example, when everyone is out to lunch but network activity is steadily climbing, this may be a sign of a cyber attack and could potentially bring down your network and in the long run, cost you time and money.

The goals of the group project are split into several sections that are assigned to several students from multiple universities. Each researcher is assigned a specific task that will contribute to the group’s overall objective of producing a working set of Ordinary Differential Equations. These tasks range from the creation of the test Abilene network along with its routing algorithms, links and added latency to simulate traffic delay. While one researcher works on creating this network, another researcher is working on creating or using software that will generate traffic based on the Abilene distributional model. Meanwhile, another researcher, who is a PhD student, will develop the set of Ordinary Differential Equations that will be implemented to stop my attacks from causing any damage to the industrial control systems (plants and the controller).

The contribution of this capstone project is to develop attack scripts on the DETER lab testbed in order to target the modular scripts, which were designed by another researcher to mimic plant and controller behavior. The objective of the attack scripts is to deny the legitimate traffic (data) from reaching its destination and eventually causing a denial of service. This
research is aimed at contributing to the project's main goal, which is to help test the efficiency of the ODE algorithms against the threat of a cyber attack.

This capstone project will handle the tasks of generating three attack scripts that will target plants or controllers. Success is determined by causing the plants or controllers to become unstable and crash. Each attack script has a purpose; for example, the purpose of the cut link script is to instantaneously cut the communication link between the plant and controller. We are using this attack to get a baseline to determine how long it takes for our plant to become unstable and crash. The flood link attack is designed to use several compromised computers to instantaneously flood the communication link with a specified amount of continuous traffic with the goal of disrupting traffic between the plant and controller. The final attack is designed to use several attacker computers and slowly increase the amount of traffic being directed towards a specific plant or controller. Ultimately, the devices that handle the routing of data (routers) will not be able to handle the amount of data flowing; causing router overflow on these devices that will lead to failure of the plants.

One of the goals for this project is to make this research available for future researchers. The potential future researchers will have a good start off point when attempting to emulate attack scenarios on the DETER lab testbed. These researchers can continue to develop these attack scripts and further the study of mitigating cyber security threats.
Methodology

This project's complexity is based on the amount of research that goes into carrying out the objectives that were assigned. Everything from designing, creating, testing, and debugging, of an experiment requires research in learning how to use the tools available to carry out those objectives. In order to carry out a good implementation plan, we must be proficient in conducting research because time will be wasted and this may lead to producing very little results if you don’t know where to look for. The first step taken to approach this task is to conduct research about the task. This research process involves reading through research papers, searching for current works done with DETER lab, manuals for using DETER lab, and the available tools, tutorials, and learning how to use scripting language TCL. Following this approach will let us approach our experiments with some knowledge about the subject and have an idea of how to use the tools learned in the research.

The integration of this research with technology is the usage of emulations that will run on a cluster of computers that are located at UC Berkeley and USC. DETER lab assigns physical machines to simulate each computer in the experiment, then simulates the traffic, routing algorithms, and other network activities that were scripted into a file. Normally, simulations are carried out, within a single machine that will handle the entire simulation. However, this integration is a mix of many physical computers that will be used as actual computers in the simulation. That primary server that handles the simulation portion is, now a lot more efficient and does not have to worry about simulating various computers that would be performing their own computation as well.

There are very little resources that are needed to conduct this research. A laptop is needed along with SEER application software; SSH secure shell, Wireshark, and internet connectivity. DETER lab is the other resource that is provided to the researchers and is hosted at USC and UC Berkeley. Access to this lab is possible through the labs website and using a secure shell login. Any technical assistance for the DETER lab comes from the USC DETER lab staff.
Changes in project

The research group at Berkeley has put the project aside for future research because the researchers have moved on from the school setting. The biggest change that comes from this is that the attack scripts may not be used to test the ODE library until this project is continued by future researchers interested in this topic. This will not affect the portion of my project, since the goal is to produce attack scripts and make them available for future use. Some of the portions that were to be contributed by other researchers into the group project will not be included in my project. The portion to generate traffic based on the Abilene distributional model and the set of ODE’s will not be included in this project. The portion to create an Abilene topology will not be used during this experiment; instead a smaller local area topology will be created to test out the efficiency of the attack scripts on the control system.

Deliverables

This capstone project will be delivering several scripts that will be modeling cyber attack behavior against the industrial control systems in the experiment. The function of these scripts is explained in the recommended solutions section. The attack scripts and research portion will be implemented with the research group in order to carry out the main experiment. The attacks will carry out the most important part of the experiment as they will be testing the efficiency of the ODE algorithms. The other deliverable is a technical report explaining how the scripts work and how to recreate the experiment if someone wanted to replicate it.
Budget

There are few resources needed to successfully execute this project. All that is needed is a laptop to log on to the DETER lab to run the experiments. Below is the budgeting for the laptop and the amount a security researcher for academia gets paid (according to www.salary.com).

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sony Vaio Laptop</td>
<td>$600.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Services</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Security researcher (academia)</td>
<td>$35.00 per hour (Approx $67,000 per year)</td>
</tr>
</tbody>
</table>

Time line

August 2010:
- Read through research papers, manuals for DETER lab
- learning how to use the available software tools
- learning how to use scripting language TCL
- Got familiarized with the previous research that had been done on DETER lab
- Learned how to use the experimental lab from the available manuals
- Met all the members of the team and got a briefing about the project.

September 2010:
- Researched and learned how to script sequentially based events
- Learned to use 'Prog-agent' commands to create sequentially based events
- Tested of see if it was possible to execute TCL commands in a bash script
- Began researching various methods to generate attack traffic on Deter lab

October 2010:
- Continued research on traffic generation by reading through the DETER lab manual
- Started testing and trouble shooting of the sequentially based events of plant/controller scripts
• Testing with TCL to determine if it was possible to execute TCL commands and manipulate a live experiment
• Testing to see if it was possible to launch plant/controller scripts using batch scripts

November 2010:

• Colleague completed a newer version of the plant/controller scripts
• Attempted to try once again and automate these scripts, yet we were still unsuccessful
• Decided to go with the option to manually log into the nodes during the experiment and manually start separate plant/controller scripts
• Learned basic Python in order to troubleshoot some minor issues with the controller script
• Researched how to use Constant Bit Rate (CBR) generators

December 2010:

• Slow month due to the holidays, not much work was done during this time period

January 2011:

• Group project got set aside by the Berkeley mentors due to other commitments
• Research continued on my end and I was able to develop a simple attack using 'tevc -e'
• Worked on developing a bash script that would incorporate the 'tevc -e' command

February 2011:

• Created a scenario that tested out my newly formed bash scripts that executed an attack on the plant
• Developed a dynamic script for the above scenario, that would allow for quick manipulation of an experiment

March 2011:

• Researched 'trace and monitoring', used to capture packets during an experiment

April 2011:

• Researched how to implement a random function to incorporate into attack script
• Added a random function to the attack script that mimics a distributed denial of service attack
• Perform the coding, testing, and debugging process of the scripts
• Collected statistics and data

May 2011:

• Project objective completed, to generate attack scripts and successfully disable the plant/controller system
• Finished capstone paper
• Finished capstone presentation
• Finished capstone poster
Evaluation

I. Functional testing

This was the first time that the group had used an emulation based approach to model cyber security attacks on network control systems. It was also the first time that the group had used the Deter lab to model cyber security attacks. The process of developing a successful attack script starts by developing a test plan that will guide me through the entire process of developing my attack scripts. It begins with getting familiarized with this area of research by reading through research papers, tutorial and available documentation. During this process, the experimenter will gain familiarization of the function of Deter lab, understand the basics of its scripting language (TCL) and understand how to develop an experiment on the system.

DETER lab will read a filed called network simulator (NS), this NS file uses a scripting language called TCL and this scripting language allows us to create network topologies of our design. In this file we have timed based events that will launch scripted events at our predetermined time. The only way to ensure that this NS file is working, is by starting the experiment, waiting for the DETER lab to build out our scenario and observing to see if the events that we scripted takes place at the predetermined time that we specified.

To use the Deter lab, the experimenter will script all the events that he would like to execute during the experiment; for example, how many nodes he wants to use in the experiment, what type of operating system to load in the nodes, what type of network will be used, network latency, when the attack will begin and so on. In this file, we script every event that we want to happen during our experiment. Once the file has been developed, it is ‘swapped’ into Deter, Deter then assigns a physical node for each node in the experiment and will then begin to execute the events that we scripted in the NS file.

Once all the basics have been covered, the next step is to develop an experiment that will serve as the template for all future experiments. This template is composed of a simple network with various nodes assigned the role of controller, plant and attacker. The next stage is to start developing the attack scripts that would carry out the attack portion of the experiment. After much research, it was found that shell batch scripts can also execute the TCL code from within a node, during the execution of an experiment. This was the best approach to take, since the goal was to figure out how to make the attacker nodes execute an attack during the experiment. The bash shell script uses a series of TCL code to manipulate the amount of traffic it sends to the victim. Once this bash script is working without any issues, it is then dropped into an experiment directory, where it is pulled and installed on the attack node when the
experiment is being swapped in. This bash script, which will be copied onto the temporary directory of our attacker node, will then be executed by our NS file at a specified time during the experiment.

II. Usability testing

Some of the procedures that have been developed in order to verify that events in our experiment have taken place involve using a few tools. The first tool that is very valuable to conducting experiments is the Secure Shell (SSH) tool; this allows you to interact with your working space provided to you on the server. With this tool you can login to the computers on your experiments and once you are logged in, you can verify that the events that you scripted take place and did execute as you wanted to. For example, if testing to observe that a communication link was disconnected at a specific time, we would have to log into one of the experimental nodes, and see if it’s possible to send pings to another node that would be on the other end of the communication link. If the node sending pings does not receive any replies, then the scripted event did work. If it did not work, then the process of looking into the script and debugging begins.

Another great tool to use is SEER, which is a tool that attaches to your running experiment and provides a graphical user interface (GUI) that allows you to view your designed topology. This tool allows you to view live graphs by capturing the traffic flowing from the nodes of your choosing. SEER gives you the ability to view some of the events that are taking place by displaying the flow of packets traveling on the experimental network. A great advantage of this tool is the ability to create and inject events by manipulating the settings provided by the SEER GUI. You can create events such as different types of traffic events (i.e. HTTP traffic, video streaming) and you can also launch various types of cyber attacks on an experiment. These tools are very valuable since all the events are taking place inside the DETER lab and the only way to observe that our scripted experiments have taken place, is by using these tools to observe that our events did execute as we planned them too.
III. Evaluation plan

Testing of research is done at all stages to ensure that the experiment works as it was intended before progressing to extend and add more events. The testing started off small in order to verify that things are working and then continued to expand from there. The attack scripts were developed for their intended purposes, which is to test to see if the ODE algorithms can successfully stop an attack. Other users who may use my scripts for future research should be able to successfully implement my experiments since they will be easily available and easy to understand. Since the scripts I am developing will be written using NS files with TCL scripting code, other future researchers will find it easy to interact with the product and find that it is very easy to implement with other experiments that may be addressing the cyber vulnerabilities of another system.

IV. Results of testing

Success for this capstone project is based on the delivery and efficiency of the developed cyber attacks. This capstone project was tasked with developing three attack scripts that would disable the communication channel of the intended victim. The first script was the easiest and was successfully developed to instantaneously cut the communication channel between the plant and controller at a specified time during the experiment. The actual command to carry out this task, $ns at 120.0 "$link0 down", is a TCL code that is executed at a specified time during the experiment, 120 seconds after start of experiment for this example code. The purpose of this script was to determine how long would it take for the plant to become unstable once its communication channel was cut. The plant became unstable and went down immediately upon cutting the communication link.

The second script was designed to successfully flood the communication channel of the victim by having the attacker nodes send a predetermined amount of traffic at a specified time. This script has various attacker nodes sending large amounts of traffic to the victim. A flooded communication channel is the end result of multiple attackers sending large amounts of traffic
to the victim. The plant (victim) cannot effectively send its traffic to the controller, this results in instability with the plant and eventually failure.

The third script was designed to slowly increase the amount of traffic the attacker sends to the victim over a longer period of time. This was the most complicated script since it involved integrating bash scripts with TCL commands and getting it to work with Deter lab. The bash script works by taking a loop to manipulate a TCL command and continuously increase the amount of traffic being sent to the victim. A random function is incorporated into the loop; this random function produces a random number that is then multiplied by a predefined number and the result of that is then added to the new rate of traffic that the attacker will be sending. Each time the loop executes, the sum increases the amount of traffic that is sent to the victim. This attack is carried out by several nodes and the time to successfully disable the communication channel between the plant and controller varies because of this random function.
Collaboration statement

**Darrel Tyler** – Student/researcher (Humboldt State University)

Contributed to the project by developing the plant and controller modular scripts. These scripts were written using python programming language. The purpose of these scripts is to load them onto a node in the experiment and have them resemble plant and controllers by sending values back and forth on the communication channel during the execution of the experiment.

**John Mela** – Student/researcher (Youngstown State University)

Contributed to project by designing the Abilene topology (internet 2) model for use with the DETER lab.

**Sathya Narayanan** – Capstone advisor

Guided me through the capstone process and also gave recommendations for direction of capstone project. He took over as capstone client after project got put aside with the Berkeley mentors.

**Berkeley researchers** – Mentor advisors

The Berkeley researchers provided the direction of the project and gave advice.

**Deter lab technical staff** –

Gave technical support with any issues encountered while using Deter.
Final documentation

Below is the code for generating a distributed denial of service attack using DETER lab and available tools.

This is what the network will look like after it is swapped into Deter lab

![Network Diagram]

Code to generate this network

```tcl
source tb_compat.tcl
set ns [new Simulator]

# Create nodes for the experiment
foreach node { node0 node1 node2 nodeA nodeB A1 A2 A3 A4 control } {
    # Create new node
    set $node [\$ns node]

    # Define the OS image
    tb-set-node-os [\$node] FC6-STD

    # Have SEER install itself and startup when the node is ready
    tb-set-node-startcmd $node "sudo python /share/seer/v160/experiment-setup.py Basic"
}

# Create the links between nodes and set to duplex link
set link0 [\$ns duplex-link \$node1 \$node2 10Mb 0ms DropTail]
set link1 [\$ns duplex-link \$nodeA \$A1 10Mb 0ms RED]
```
set link2 [\$ns duplex-link \$nodeA \$A2 10Mb 0ms RED]
set link3 [\$ns duplex-link \$nodeB \$A3 10Mb 0ms RED]
set link4 [\$ns duplex-link \$nodeB \$A4 10Mb 0ms RED]

# Set lan0 and connect the nodes
set lan0 [\$ns make-lan "$\$node0 \$node1 \$nodeA \$nodeB" 10Mb 0ms DropTail]

# Trace the traffic during experiment on link0
# packet data is written to tcpdump (pcap) output files in /local/logs on the delay node
\$link0 trace

# un-tar the controller/plant files to the /bin directory of experimental nodes
\$\$tb-set-node-tarfiles \$node0 /bin /proj/TRUST-REU/tarfiles/controller.tar
\$\$tb-set-node-tarfiles \$node2 /bin /proj/TRUST-REU/tarfiles/plant.tar

# un-tar the attack batch scripts to the attackers /tmp/ directory
\$\$tb-set-node-tarfiles \$A1 /tmp /proj/TRUST-REU/tarfiles/stage2A.tar
\$\$tb-set-node-tarfiles \$A2 /tmp /proj/TRUST-REU/tarfiles/stage2B.tar
\$\$tb-set-node-tarfiles \$A3 /tmp /proj/TRUST-REU/tarfiles/stage2C.tar
\$\$tb-set-node-tarfiles \$A4 /tmp /proj/TRUST-REU/tarfiles/stage2D.tar

###### TRAFFIC GENERATORS #######
# Create a TCP agent (creates TCP traffic) and attach it to attack node A1
set tcp1 [new Agent/TCP]
\$\$ns attach-agent \$A1 \$tcp1

# Create a CBR traffic source and attach it to tcp1
# Constant Bit Rate (CBR) traffic generators are used to create the attack traffic
set cbr1 [new Application/Traffic/CBR]
\$cbr1 set packetSize_ 512
\$cbr1 set interval_ 0.005
\$cbr1 attach-agent \$tcp1

# Create another TCP agent and attach it to attack node A2
# Create a TCP agent and attach it to A2
set tcp2 [new Agent/TCP]
\$\$ns attach-agent \$A2 \$tcp2

# Create a CBR traffic source and attach it to tcp2
set cbr2 [new Application/Traffic/CBR]
\$cbr2 set packetSize_ 512
\$cbr2 set interval_ 0.005
\$cbr2 attach-agent \$tcp2

# Create another TCP agent and attach it to attack node A3
# Create a TCP agent and attach it to A3
set tcp3 [new Agent/TCP]
\$\$ns attach-agent \$A3 \$tcp3

# Create a CBR traffic source and attach it to tcp3
set cbr3 [new Application/Traffic/CBR]
\$cbr3 set packetSize_ 512
\$cbr3 set interval_ 0.005
\$cbr3 attach-agent \$tcp3
# Create another TCP agent and attach it to attack node A4
set tcp4 [new Agent/TCP]
$ns attach-agent $A4 $tcp4

# Create a CBR traffic source and attach it to tcp4
set cbr4 [new Application/Traffic/CBR]
$cbr4 set packetSize_ 512
$cbr4 set interval_ 0.005
$cbr4 attach-agent $tcp4

##### TRAFFIC SINKS at VICTIM######
# Create four TCPSink agents for A1, A2, A3, A4 & attach to node2 (victim)
# This is were the traffic will be directed
set null1 [new Agent/TCPSink]
$ns attach-agent $node2 $null1

set null2 [new Agent/TCPSink]
$ns attach-agent $node2 $null2

set null3 [new Agent/TCPSink]
$ns attach-agent $node2 $null3

set null4 [new Agent/TCPSink]
$ns attach-agent $node2 $null4

##### Connect the traffic sources with the traffic sinks#####
$ns connect $tcp1 $null1
$ns connect $tcp2 $null2
$ns connect $tcp3 $null3
$ns connect $tcp4 $null4

# Setting up a prog-agent command to execute the command found in "quotations"
# prog-agent command will be executed during the experiment at a specified time by 'ns at'
# below are the bash scripts that will carry out the attack
set prog2 [SA1 program-agent -command " /tmp/stage2A.bash "]
set prog3 [SA2 program-agent -command " /tmp/stage2B.bash "]
set prog4 [SA3 program-agent -command " /tmp/stage2C.bash "]
set prog5 [SA4 program-agent -command " /tmp/stage2D.bash "]

# Start program agent command
$ns at 5.0 "$link0 trace stop"
$ns at 10.0 "$link0 trace start"
$ns at 20.0 "$prog0 start"
$ns at 40.0 "$prog1 start"
$ns at 80.0 "$prog2 start"
$ns at 85.0 "$prog3 start"
$ns at 90.0 "$prog4 start"
$ns at 95.0 "$prog5 start"
$ns at 110.0 "$link0 trace snapshot"
# Following will cut link0 and disable communications that link: This is the cutlink example
#$ns at 120.0 "$link0 down"
# Setting up routing
$ns rtproto Static

# Run the simulation
$ns run

#END OF CODE FOR EMULATION#
#below is the attack script that is installed in each designated attack computer#
#this script uses a random function inside a loop to generate random values and make our Distributed Denial of Service attack vary between each experiment. Each time the loop is ran, it will add that random value to the size of the attack traffic it’s sending to the victim.
#!/bin/bash
tevc -e TRUST-REU/MergeAttack2 now cbr1 start
tevc -e TRUST-REU/MergeAttack2 now cbr1 set rate_=500Kb
x=0
while [ $x -lt 30 ]
do
tevc -e TRUST-REU/MergeAttack2 +$time cbr1 set rate_=$rate
x=$(( x + 1 ))
time=$(( time + 5 ))
NUMBER=$(( RANDOM % 10 ) + 1)
rate=$(( rate + (3* NUMBER) ))
done

#End of attack bash script#
#END OF EXPERIMENT CODE#
#Below is code for implementing a cut link attack – This attack is intended to sever the communication channel
and is used as a baseline to determine how long it takes for a plant to go down when they have no connection to
the controller node.
#Note: Most of the code was taken out because it is the same as above experiment#
#code that produces cut link is bolded#

```
set ns [new Simulator]
source tb_compat.tcl

# Controller for SEER
set control [$ns node]
tb-set-node-os $control Ubuntu804-NS2
tb-set-node-startcmd [set $control] "sudo python /share/seer/v160/experiment-setup.py Basic"

#Controller for network
set controller [$ns node]
tb-set-node-os $controller Ubuntu804-NS2
tb-set-node-startcmd [set $controller] "sudo python /share/seer/v160/experiment-setup.py Basic"

set node1 [$ns node]
tb-set-node-os $node1 Ubuntu804-NS2
tb-set-node-startcmd [set $node1] "sudo python /share/seer/v160/experiment-setup.py Basic"

set node2 [$ns node]
tb-set-node-os $node2 Ubuntu804-NS2
tb-set-node-startcmd [set $node2] "sudo python /share/seer/v160/experiment-setup.py Basic"

# Links
set link0 [$ns duplex-link $controller $node1 100000.0kb 0.0ms DropTail]
set link1 [$ns duplex-link $controller $node2 100000.0kb 0.0ms DropTail]
set link3 [$ns duplex-link $node1 $node2 100000.0kb 0.0ms DropTail]

#following will cut the link between router and victim
#$ns at 120.0 "$link2 down"

$ns rtproto Static
$ns run
#End of code#
```