Effect of DDOS attacks on simple plant-controller networks

Ricardo Estrada
Computer Science
CSU Monterey Bay
restrada@csumb.edu

Graduate Mentors:
Blaine Nelson
Saurabh Amin
Suzanna Schmeelk

July 30, 2010

TRUST Research Experiences for Undergraduates (TRUST-REU)
In Cyber Security and Trustworthy Systems 2010

Department of Electrical Engineering and Computer Sciences
College of Engineering
University of California, Berkeley

ABSTRACT

In this paper, We are emulating routing and traffic associated with the Abilene network topology (known as Internet2) in order to determine the effects of an inside attack on critical infrastructure components. In our case, our critical infrastructure component is a plant/controller infrastructure. This architecture is simulated at a smaller scale and mimics the Abilene backbone without massive traffic actually being observed. The project was divided across several groups, each one responsible for one of the following: infrastructure development, software development and attack generation. The type of attack used for this experiment is in the form of a distributed denial of service (DDoS) attack. The goal is to exploring how a DDoS attack works and how its attack can cut the communication link between critical infrastructure components and their controllers. The principal result is to create a successful emulation of the Abilene topology along with its routing and traffic. Once this topology was in place, a DDoS attack was carried out in order to deny the service from plant to controller.

Introduction

The DETER testbed, implemented as a mesh of cluster of experimental nodes[5] located at USC ISI and UC Berkeley, is intended to provide an experimental infrastructure to support the development and demonstration of next-generation information security technologies[1][3]. DETER provides a medium-scale facility for safe, repeatable, security-related experimentation.[1] The goals of the project is to study the interaction and strategies for attack and defense of control systems in the emulated environment provided by the DETER cluster. An emulated environment provides real applications, protocols and operating systems to a synthetic network environment [2]. Using DETER, we hope to be able to accomplish the following goals:

1. Construct an emulation of the Abilene (Internet 2) topology backbone and the control system behavior communicating over it.
2. Implement realistic attacks in that emulated environment which cause the control system to fail, ultimately leading to plant failures across the system.
3. Explore defenses that make the control & learning systems more resilient to these attacks.

Network control systems (NCS) extend the distributed control of the physical world beyond distance barriers [4]. The setup consists of a computer system and a remote controller for monitoring and controlling a process [6][8]. They integrate sensing, processing, and actuation tasks that enable remote monitoring and control of the physical world. A few of the applications include water treatment and distribution, wastewater collection and treatment, oil and gas pipelines, electrical power transmission and distribution, and building structure control. Successful design and implementation of NCSs necessitate the existence of simulation tools that allow verification, validation, and evaluation of different control and network algorithms. A sensor will sample the values of physical quantities, writes them in a packet, and sends the packet over the network to the controller. The controller examines the received sample to generate a control signal that is then sent over the network to the actuator.

These industrial control systems are key infrastructure components that can be subject to malicious attacks both from external and internal threats. This paper focuses on a malicious attack (DDoS attack) from within the network. A distributed denial of service attack (DDoS) is a

coordinated attempt, via infected computers, to exhausts victims resources, thus making it unavailable to its intended users. There are various types of DDoS attacks but all follow the same guidelines. An attacker must first choose a method of scanning in order to find vulnerable machines. There are several methods, to list a few:

- Random scanning: infected machines probe IP addresses randomly and find vulnerable machines that it will try to infect. This spreads very quickly and creates large amount of traffic.
- Hit-list scanning: Attacker first collects a list of large numbers of potentially vulnerable machines before start of scanning. Once a machine is found, attacker infects it and splits the list, giving half of the list to the compromised machine. This same procedure is carried out on infected machines. All machines are compromised within a short interval time without generating significant amounts of traffic.
- Topological scanning: Uses information contained on the victim machine in order to find new targets. Looks for URLs in the disk of a machine that it wants to infect.
- Local subnet scanning: Acts behind a firewall and looks for targets in its own local network. This method creates large amount of traffic.
- Permutation scanning: All machines share a common pseudorandom permutation list of IP addresses. Based on certain criteria, it starts scanning at some random point or sequentially.

After the attacker has found vulnerable machines, he must now choose a method to propagate his malicious code. Here are several methods to accomplish this:

- Central source propagation: Copies code onto victim then the victim will find another victim and copy the same code onto that victim and the process will be repeated all over with new machines. Commonly uses HTTP, FTP and RPC protocols to propagate the malicious code.
- Back-chaining propagation: Finds a vulnerable machine, copies an attack toolkit onto that machine and sends the systems identity to the hacker. TFTP is used to propagate the malicious code.
- Autonomous propagation: Transfers the attack toolkit to the newly compromised system at the exact moment that it breaks into that system.

Once the above steps are accomplished, the attacker will coordinate when and what type of attack he wants to carry out on a system.

The security of these SCADA-based systems has come into question as they are increasingly seen as extremely vulnerable to cyber warfare/cyber terrorism attacks. Some of the consequences that may result from such DDoS attack on a network control system can range from bringing an infrastructure down, disrupting the flow of operations, cause financial and physical damage on equipment or ultimately deny service to customers.

Methodology

The Abilene network architecture for this research is simulated at a smaller scale and mimics the Abilene backbone without massive traffic actually being observed. Our approach to

performing a successful DDoS attack on this SCADA system requires a series of steps, shown in the order below:

Phase 1: Use DETER to build NS file. The DETER testbed gives us a GUI editor that allows us to simply drag and drop nodes and switches to create a basic scenario. After this basic scenario is created, an NS file is generated with TCL scripting language, this NS file will allow for later modification and additions to our experiment.
Phase 2: TCL scripting allows us to manually perform the following changes in our newly created NS file:

```
Set node0 [$ns node]
Set node1 [$ns node]
```

This TCL script will add an additional node with the name "node0" to our script. To add additional nodes, simply copy code and replace "node0" with "node#" were number is the number of our next node.

```
tb-set-node-os $node0 FC6-STD
tb-set-node-os $node1 FC6-STD
```

Set the operating system for node0 to be FC6-STD. The values that can change are the "$node0" which can be the name of the node we would like to assign our operating system to and "FC6-STD", which can change to other available operating systems on DETER.

```
set link0 [$ns duplex-link $node0 $node1 100000.0kb 0.0ms DropTail]
```

Set the communication link (link0) between our node0 and node1 to be a duplex-link at 100Mbs with 0.0ms of latency with a queue type of "Drop Tail".

```
tb-set-node-startcmd [set $node0] "sudo python /share/seer/v160/experiment-setup.py Basic"
```

Call on a start command to start up node0 when the experiment is swapping in by executing the "experiment-setup.py" script.

```
tb-set-node-tarfiles $node0 /bin/ /proj/TRUST-REU/tarfiles/controller.tar
tb-set-node-tarfiles $node2 /bin/ /proj/TRUST-REU/tarfiles/plant.tar
```

This code will tar the contents of controller.tar file located in the path listed on to the /bin/ directory of node0. The code below will do the same action for for the plant.tar onto node2.

```
set prog0 [$node0 program-agent -command "sudo python /bin/controller.py -p 50000"]
```

Create a program agent (prog0) that will execute the given command in quotations, on port 50000 of node0. The controller.py will give us a controller server in our experiment.

```
set prog1 [$node2 program-agent -command "sudo python /bin/plant.py -s 10.1.1.2 -p 50000"]
```

Create a program agent (prog1) that will execute the given command in quotations on node2, and request to establish a connection on port 50000 of listed IP address. The plant.py will give us our plant in our experiment.

$ns at 45.0 "$prog0 start"	$ns at 90.0 "$prog1 start"

Above is code for a timed event to execute our program agent scripts.
Script on left will start our prog0 at 45 seconds
Script on right will start our prog1 at 90 seconds

$ns rtproto Static

Set our routing protocol to static

Phase 3: The script above is swapped in and allowed to run, in order to ensure that our plant and controller software are running. DETER will create our experiment based on the inputs we made with TCL scripting on the NS file.
Phase 4: In this phase we will use various tools such as SSH secure shell, SEER & TCPDump to conduct and verify results of our experiments.

- SSH secure shell played a major role in conducting our experiments. It was used to manually pull log files, troubleshoot node problems, test connectivity and most importantly capture traffic by utilizing TCPDump.
- TCPDump is a tool that will allow you to view in real time or capture the traffic going through a specified ethernet port.
- SEER provides security researchers the ability to create, plan, and iterate through a large range of experimental scenarios with relative ease[7]. It integrates various tools for configuring and executing experiments and provides a user friendly interface for experimenters to use the tools[7]. The experiments conducted through SEER ranged from creating simple HTTP traffic on the DETER network, to performing packet flooder attacks and viewing live graphs of network activity.

Results

The purpose of our experiments was to conduct several DDoS attacks on the Abilene network. Our attacks were designed to cause instability between the controller and plant, ultimately leading to the failure of the plant. The plant is designed to send a value to the controller and the controller is designed to perform a calculation and send a value back to the plant. The plant will then use that value to maintain stability. We successfully conducted a cut link experiment along with two types of DDoS attacks to cut the communication link between our plant and controller. Below are the results of each attack:

- The first experiment we conducted was the usage of a cut link command in our NS file to instantaneously cut the link between the plant and controller. Upon cutting the communication between the plant and controller, the plant became unstable and crashed. The purpose of this cut link command was to determine the amount of time it took for our plant to become unstable and crash. This immediate crash is due to the coding in the plant script, unfortunately we were not able to address this issue with the amount of time remaining.

- Our flood link attack is designed to flood the communication link between the plant and controller. We select several nodes that will simultaneously carry out this attack at a specific time of the experiment on our victim (plant). Flood link sends huge amounts of traffic to the victim, filling up the available bandwidth and causing the router buffer to overflow and to start dropping packets. In this case, since the bandwidth is full of attack traffic and the router is now dropping packets, the communication between the plant and controller has been severed. The plant becomes unstable because the value it needs to maintain stability is not returned and therefore crashes.

- TCP ramp up was conducted using SEER because when implemented in the NS file by TCL scripting, it would not work correctly. Once the experiment was up and running, the SEER application had to be connected to the DETER network and the "packet flooder attack" was adjusted to perform a TCP rampup attack on our plant. This TCP rampup attack is designed to gradually (over a period of time depending on the size of the attack) flood the communication link between the controller and plant. Data being sent back and forth between the plant and controller will continue until the point when the link between them overflows the router with attack traffic.

Conclusion

The goal of a distributed denial of service attack is to exhaust the victim's resources. A successful DDoS attack can deny a service to the victim and its hosts by consuming the network bandwidth or exhausting the computing power. Although our attacks were targeting an Abilene network that had no security measures to deal with these forms of attacks, our attacks were simple DDoS attacks. There are much more sophisticated forms of attacks out there that are designed to penetrate networks by exposing a security weakness. A DDoS attack directed towards critical infrastructure components can cause disruptions, setbacks, physical and financial damages. The need to address the security issues surrounding these infrastructure components is critical. These network control systems control vital services that the public uses and is vulnerable to these forms of attacks by people with either curiosity or malicious intent. In order to effectively counter these emerging Internet threats, a thorough understanding of these threats and systematic evaluation of the potential defense in a realistic testing environment is required [7]. Further research should be addressed to ensure that these systems are secure and able to deal with the threats that come from being connected to the outside world (internet).

Future Work

My recommendation for future work would be to conduct a full scale DDoS attack on the emulated Abilene network. Further research should be conducted on DDoS and the methods to successfully carry out an attack. I find it beneficial to understand the methods and how to carry out an attack in order to be able to deploy methods to defend a network from such attacks. My approach for future work would consist of selecting a scanning method to scan for vulnerable machines. Once the vulnerable machines were found, the next step would be to select a method to propagate the malicious code to infect the vulnerable machines. After attaining a certain amount of infected nodes with malicious software installed on them, a DDoS

attack method should be conducted on the Abilene network. The various methods listed earlier should be combined and experimented with, in order to find the right DDoS attack that would penetrate the defenses of the network.

The next component after familiarizing and performing various types of DDoS attacks will be to look at methods for minimizing the threats (defending) that a network control system faces on a daily basis. The methods will range from implementation of firewalls, network/host intrusion detection systems at key locations, policies that will ensure nodes are up-to-date, have security software and are compliant, and monitoring the flow of traffic to observe any anomalies on the network. After implementing the recommended security features, researchers should test their acquired skills to carry out DDoS attack and see if weaknesses can be exposed on the secured Abilene network. I believe that this approach will allow researchers to find weaknesses on a more secured Abilene network, since the current Abilene network we used has no methods of security deployed to it.

Acknowledgements

This work is supported by the National Science Foundation under Award number CFF-0424422, via the Department of Electrical Engineering and Computer Sciences, College of Engineering located in Cory Hall at the University of California Berkeley. I would like to thank the Team for Research in Ubiquitous Secure Technology (TRUST), Dr. Kristen Gates, Larry Rohrbough, Ted Faber, Jelena Mirkovic, the DETERlab team. Additionally, I would like to thank my mentors: Blaine Nelson, Saurabh Amin and Suzanna Schmeelk. Special thanks also to my groupmates: Katherine Gabales – Chico State University and Kyle Marlin – Youngstown State University.

References

[1] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, S. Schwab. TridentCom2006. *Experience with DETER: A Testbed for Security Research*. 2nd IEEE Conference on testbeds and Research infrastructures for the Development of Networks and Communities".

[2] White, B., J. Lepreau, L.Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. Dec. 2002. *An Integrated experimental environment for distributed systems and networks*. In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI02). Pp. 255-270

[3] T. Benzel, B. Braden, T. Faber, J.Mirkovic, S. Schwab, K. Sollins and J. Wroclawski. *Current Developments in DETER Cybersecurity Testbed Technology*. Washington D.C: IEEE computer society, 2009.

[4] Ahmad T. Al-Hammouri, Michael S. Branicky, and Vincenzo Liberatore. *Co-simulation tools for Network Control Systems.* Berlin, Heidelberg: Springer-Verlag, 2008.

[5] T. Benzel, R. Braden, E. Fraser, A. Joseph, D. Kim, J. Mehringer, C. Neuman, R. Ostrenga, S. Schwab, Dan Sterne. *Design of the DETER Security Testbed.* USC Information Sciences Institute, University of California at Berkeley, McAfee Research, 2004.

[6] Vincenzo Liberatore. *Network Control Systems.* Cleveland: Case Western Reserve University, 2002.

[7] Stephen Schwab, Brett Wilson, Calvin Ko, Alefiya Hussain. *SEER: A Security Experimentation EnviRonment for DETER.* Berkeley, Ca: USENIX Association, 2007.

[8] Alvaro A. Cardenas, Saurabh Amin, Shankar Sastry. *Research Challenges for the Security of Control Systems.* Berkeley, Ca: USENIX Association, 2008.

Appendices

Below are the contents of an NS file manipulated with TCL scripting code to represent a 4 node topology with a switch. The purpose of each segment of code is listed in the commented out section above the code.

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

# Create nodes
set control [$ns node]
set node0 [$ns node]
set node2 [$ns node]
set node3 [$ns node]
set node4 [$ns node]

#Set OS of each node
tb-set-node-os $control FC6-STD
tb-set-node-os $node0 FC6-STD
tb-set-node-os $node2 FC6-STD
tb-set-node-os $node3 FC6-STD
tb-set-node-os $node4 FC6-STD

#Wake up nodes
tb-set-node-startcmd [set $control] "sudo python /share/seer/v160/experiment-setup.py
Basic"
tb-set-node-startcmd [set $node0] "sudo python /share/seer/v160/experiment-setup.py
Basic"
tb-set-node-startcmd [set $node2] "sudo python /share/seer/v160/experiment-setup.py
Basic"
tb-set-node-startcmd [set $node3] "sudo python /share/seer/v160/experiment-setup.py
Basic"
tb-set-node-startcmd [set $node4] "sudo python /share/seer/v160/experiment-setup.py
Basic"

#Create the link between nodes
set Switch1 [$ns make-lan "$node0 $node2 $node3 $node4" 100000.0kb 0.0ms]

#We will tar the files to the /bin directory
tb-set-node-tarfiles $node0 /bin/ /proj/TRUST-REU/tarfiles/controller.tar
tb-set-node-tarfiles $node2 /bin/ /proj/TRUST-REU/tarfiles/plant.tar
tb-set-node-tarfiles $node3 /bin/ /proj/TRUST-REU/tarfiles/plant.tar
tb-set-node-tarfiles $node4 /bin/ /proj/TRUST-REU/tarfiles/plant.tar

#setting up a program agent to process a command
set prog0 [$node0 program-agent -command "sudo python /bin/controller.py -p 50000"]
set prog1 [$node2 program-agent -command "sudo python /bin/plant.py -s 10.1.1.2 -p
50000"]
set prog2 [$node3 program-agent -command "sudo python /bin/plant.py -s 10.1.1.2 -p
50000"]
set prog3 [$node4 program-agent -command "sudo python /bin/plant.py -s 10.1.1.2 -p
50000"]

#starting our program agent command
$ns at 45.0 "$prog0 start"
$ns at 90.0 "$prog1 start"
$ns at 95.0 "$prog2 start"
$ns at 100.0 "$prog3 start"

$ns rtproto Static
$ns run
```