SEER: A Security Experimentation EnviRonment for DETER

Stephen Schwab   Brett Wilson   Calvin Ko   Alefiya Hussain

SPARTA, Inc.
El Segundo, CA

Abstract

Configuring a security experiment can be tedious, involving many low level and repetitive configuration tasks. In order to make DETER's capabilites accessible to users at all skill levels, we have designed and implemented a Security Experimentation EnviRonment (SEER) that provides security researchers the ability to create, plan, and iterate through a large range of experimental scenarios with relative ease. SEER integrates various tools for configuring and executing experiments and provides a user-friendly interface for experimenters to use the tools. SEER aims to support a wide range of experimentation requirements such that many researchers will prefer to interact with DETER through it, fostering collaboration within the security research community.

1 Introduction

Over the years, we have been facing increasingly serious Internet security threats such as worms, spyware, Distributed Denial-of-Service (DDoS) attacks, botnets, etc. 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. Moreover, it is important that the scale of these network tests be sufficiently large to inform our understanding of how cyber attacks and defenses will evolve at Internet scale. In security research, repeatable experimentation is crucial in understanding the behavior of large-scale threats, validating conclusions of studies, and in evaluating and comparing different potential solutions. Similar to the networking and operating system communities, repeated research in cyber security is vital to the scientific advancement of the field.

The DETER testbed [1] aims to facilitate network security experimentation by providing an environment for researchers to perform experiments within, in a secure, isolated fashion. DETER runs a tailored configuration of the Emulab software developed at Utah [2]. It allows a security researcher to obtain exclusive use of a subset of the testbed machines, configure them into a specified topology, and access them through a firewall from across the Internet. Nevertheless, even with the availability of large-scale testbeds such as DETER, performing a security experiment is cumbersome. The process involves configuration of a complex environment to support a relevant network topology and infrastructure services (e.g., DNS), generation of attack and background traffic, deployment of defense mechanisms, instrumentation of the network for data collection, and finally data analysis and visualization. Typically, an experimenter needs to repeat the configuration for a wide set of experiments in order to systematically evaluate a defense.

In this paper we describe SEER – A Security Experimentation EnviRonment – that enables an experimenter to configure security experiments offline and provides fine-grain control and monitoring support during experiment execution. SEER is the on-going evolution of the technology that originated as the DDoS security experimenter's workbench [3]. SEER consists of a Java-based front-end GUI and a set of services that collectively give researchers the ability to create, plan, and execute their experiments. SEER interacts with the DETER control plane to identify and (if required) request the resources for the experiment. It then integrates a set of tools within the experiment context through which researchers create, configure, and control their experiments. Further, it enables repeatability by facilitating experimenters to rerun experiments with minimum effort, either verbatim or with controlled variation of parameters when exploring several dimensions of the evaluation space.

SEER evolved from our experience in developing an experiment methodology [4,5] for evaluating DDoS defense systems. SEER builds on the facilities in DETER and greatly enhances an experimenter's efficiency in performing security research. We expect many, and perhaps most researchers, will interact with DETER through SEER while DETER will continue to provide direct lower-level interfaces for use by the testbed community. SEER will foster innovation and collaboration at a higher level of abstraction, while supporting advanced users that wish to extend SEER with custom agents that directly control tools running on, or features of, DETER.

The rest of the paper is organized as follows. Section 2 provides an overview of SEER and the approach taken to assist security experimenters. Section 3 describes the implementation of SEER. In Section 4 we illustrate the use of SEER by describing an example of how SEER configures a simple experiment. Section 5 discusses future work and our conclusions.

2 Overview

Setting up and running a security experiment requires substantial configuration effort. For instance, just running a worm and observing its behavior demands setting up some real network services (e.g., a mail worm requires real mail servers). In addition, many network infrastructure services (e.g., DNS) have to be set up. For some worms that spread via user activities (e.g., users clicking on a website), we need to simulate activities of real users. In addition, some nodes may need to mimic one or more subnets or an autonomous system in the Internet. In simulating a Distributed Denial-of-Service attack, the experimenters also need to control the timing (e.g., when does the attack start and end).

Our goal is to provide security experimenters with an easy-to-use integrated environment to perform large-scale repeatable experiments with cyber attacks and defenses. The following summarizes the design objectives and how SEER achieves them.

Ease of use. Making security experimentation easy to conduct is important because security researchers can then focus their attention on the unique, core aspects of the security research problem at hand. In particular, security researchers should be able to create and run experiments without performing low level configuration (e.g., editing configuration files on the experiment nodes, initializing routing tables) of the network. SEER provides a click-and-choose GUI for an experimenter to set up and run experiments. For instance, SEER allows a user to describe at a high level the configuration of the network and its infrastructure services and performs all the necessary low-level configurations automatically. In addition, SEER includes a visualization tool for researchers to observe the execution of the experiments, which is valuable in managing and debugging the experiments.

Support for automated repeatable experiments. In performing security research, one often needs to repeat the experiment with systematically varying parameters, such as using different background traffic, or a different type of attack or defense. SEER provides an intuitive scripting interface for advanced users to write scripts to run and control multiple experiments in a batch mode. It also automates the collection of data and provides common analysis tools for comparison of results between experimenters.

Support for collaboration and reuse. DETER is intended for the security research community to help each other in developing and evaluating cyber security defenses. SEER facilitates collaboration by allowing security researchers to share and reuse their components within and across experiments. In addition, an experiment configuration can be saved and readily reused by other researchers.

Extensible. It is impossible to anticipate all possible components required by future experiments. Therefore, SEER provides a systematic way for security researchers to add new components (e.g., new attack tools, new background traffic generators) to the framework as new agent types. Our vision is that a researcher will choose to use SEER because of its wide range of supported features, and will add their own novel components to SEER because they find it convenient and productive to do so. This will lead, we hope, to a virtuous cycle in which the growing set of community developed features attracts even more users to the testbed.

Security. As SEER and the DETER testbed are used for experimentation with attacks or malicious code, it is important to mitigate the potential risk and assure that the attacking code will not be accidentally released to the Internet. In particular, experiment nodes cannot exchange IP packets directly with the Internet. A DETER user must logon to the DETER control plane server (users) in order to logon to an experiment node. SEER employs secure communication between the users' desktop machine and the DETER server. In performing experiments with malware, security measures are needed to contain the malware on the experiment nodes, verifying that no residual malware remains after the experiment is terminated, and ensuring that potentially infected files are cleaned or encrypted before being released to the experimenter.

Tool Integration

Over time, many different tools (e.g., attack tools, traffic generation tools, network configuration tools) for supporting various security and network experiments have been developed. As these tools are developed by different researchers for their experiments, it will take significant effort for other researchers to utilize them. SEER aims to integrate a wide range of existing and future tools, tying them to a unified framework and providing common interfaces for the user to conduct experiments using the tools.

In particular, we identify the following building blocks of a security experiment.

- Network infrastructure configuration
- Generation of background traffic and activity in the test network
- Simulation of attacks
- Deployment and management of defense mechanisms
- Instrumentation and data collection
- Data analysis and visualization

A security experiment may need some or all of these building blocks. Currently SEER contains a library of traffic generation tools, attack tools, network configuration tools, and data collection and presentation tools. In the next section, we will describe the implementation of SEER in detail.

3 Implementation Detail

SEER is implemented using a variety of technologies. Figure 1 describes the high level structure of SEER. It contains three layers of software that interact with each other.

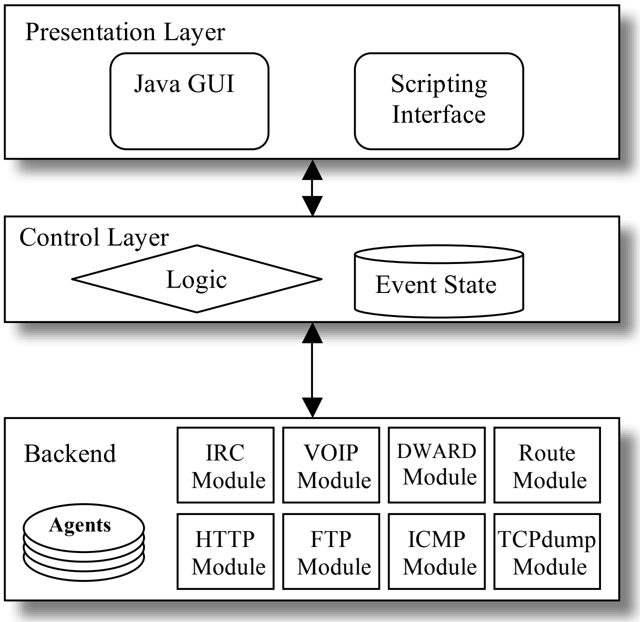


Figure 1: The 3-tier Structure of SEER

The top presentation layer consists of interface tools for users to interact with the experiment. Currently, it has a Java-based GUI and a Perl-based scripting interface. The SEER GUI provides a click-and-choose interface for users to configure various components in the experiment, either in real-time or capture and written to a script for later use.

The bottom layer consists of software agents running on every experiment node. The agent is responsible for performing the configuration required at the node on which it runs. The functionality of an agent can be extended through the loading of modules. We classify modules into the following types: traffic generation modules, attack modules, defense system modules, data collection & analysis modules, and network service modules.

The middle layer runs on a special *control node* in the experiment. It listens for high-level commands received from the presentation layer and sends low-level commands to the agents. This logic could be run on a testbed server such as users, but an extra control node in each experiment is used instead to offload the processing, storage, and network traffic. It is also necessary to run within the confines of the experiment if containment is required.

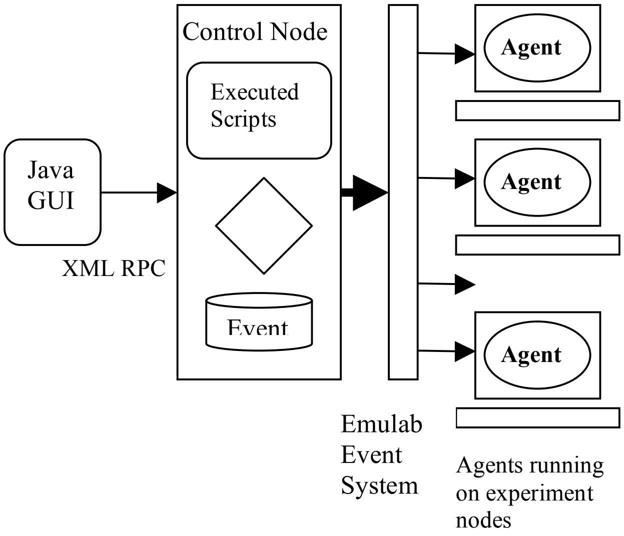


Figure 2: Runtime Architecture

Figure 2 depicts the runtime architecture of SEER. When the experiment starts in DETER, every node in the network will run an agent, which listens to commands from the control layer. The control node communicates with the agents using the Emulab event system.

An agent's state is represented as a set of variables that can be set, changed, or deleted. In addition, an agent performs actions in response to an event. Agents are categorized based on their object type and their group name. Each node in the experiment maintains the same agent state. When an action is to be performed on an agent group, every node belonging to that group will perform the action. Variables such as NODES, CLIENTS, and SERVERS instruct an agent as to its role in the group, and what the necessary operations are to be performed. For example, a node that is not listed as a client or a server in a HTTP background traffic generation group will simply perform a no-op. If the client or server variables change, the new clients or servers will already be aware of any other variables that were set previously.

The SEER GUI can run on the end user's desktop anywhere in the Internet. It employs XMLRPC to allow the GUI running on a remote host to communicate with the control node in the experiment. In the DETER environment, a SSH connection is used as the XMLRPC transport to the DETER control plane by logging in to the users.isi.deterlab.net server. From users.isi.deterlab.net, the XMLRPC is proxied to the internal DETER XMLRPC server or the SEER experiment control node, depending on the function called.

Currently SEER consists of a library of background traffic modules, a suite of network service modules, and a library of DDoS attack tools. The background traffic generation module supports generation of various traffic types including ICMP Echo, DNS, HTTP, FTP, SSH, IRC, and VOIP traffic. The frequency of the client service requests, the data sizes of the requests and the replies, and duration of the connection can be specified using common statistical distributions including the Pareto, Gamma, or Exponential distributions. In addition, the suite includes Harpoon, an Internet traffic generation tool that can produce background traffic. The replay agent allows a captured traffic dump, when available, to be replayed within the experimental network.

Each node in the experiment may play the role of an entire subnet, defined by network address and mask. The traffic generation tools will automatically recognize the subnet address ranges, and use them as the source and destination for their traffic, selecting addresses in the subnet at random. Other methods for selecting addresses from the subnet range will also be supported in future versions.

SEER contains a library of DDoS attack tools that can reproduce a repertoire of DDoS attacks commonly occurring on the Internet. In the future, we will augment these tools with a spyware and adware library, as well as a collection of botnet code for conducting experiments with spyware, adware, and botnets.

3.1 Scripting Support

The agent framework provides a very simple abstraction to the event transmission and agent groups in Perl. Using this abstraction, the experimenter can specify an experiment in a Perl script using any of the language constructs available. The experimenter creates a reference to an agent group using the idiom:

```
$var=Agent::New($txobject,group_type,group_name);
```

The returned variable can be used to set variables and send events to the given group of agents with the `Set(hash)` and `Event(string)` functions. Shortcut functions `Start()`, `Stop()` and `SetLocation()` are also available. The experimenter is responsible for understanding what variables can be set and what they do. This information is available at http://seer.isi.deterlab.net.

Scripts are executed in the experiment environment and simply inject events into the event system. Therefore any application or script that is capable of injecting events could potentially be used. Future work may include other language interfaces as well as additional pre-defined interfaces specific to each agent type.

4 Example

This section illustrates how a security researcher configures and runs a simple DDoS experiment using SEER. The experiment has a simple topology consisting of four nodes connected to each other.

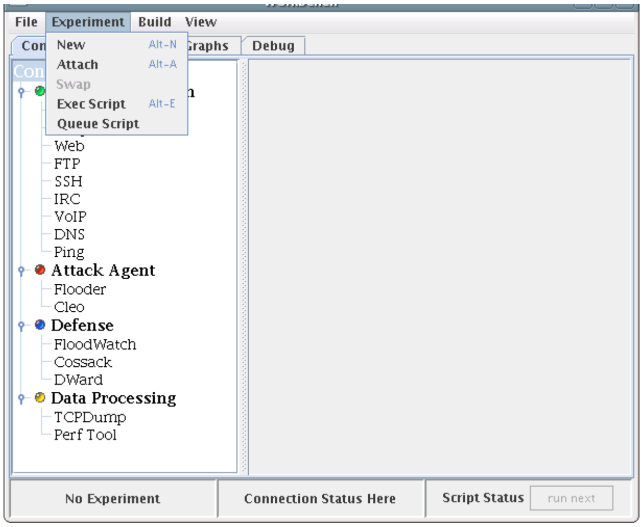
Figure 3 presents a DETER topology definition that specifies the network topology as well as the necessary initialization for SEER. The syntax and semantics of the definitions are based on the topology definition in ns2. Lines 6-13 describes the nodes (V, R, A1, A2, and control), including the name of the node, the machine type, the operating system the node will run, and the DETER group they belong too. *Control* is the control node that runs the SEER control software. It is only connected to the control plane. Line 10 instructs DETER to load the SEER agent software on every node. Lines 15-17 define the links between the nodes. Lines 22-24 instruct DETER to start the SEER agent when the node boots up.

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

#Create the topology nodes
set proggroup [$ns event-group]
foreach node { V R A1 A2 control } {
   set $node [$ns node]
   tb-set-node-os $node FC4-STD
   tb-set-hardware $node pc3060
   tb-set-node-tarfiles $node
           /share/workbench/wb-fc4-20070522-dev.tgz
   set $node-prog [$node program-agent]
   $proggroup add $node-prog
}
#Create the topology links
set linkRV [$ns duplex-link $V $R 100Mb 3ms DropTail]
set linkRA1 [$ns duplex-link $R $A1 50Mb 3ms DropTail]
set linkRA2 [$ns duplex-link $R $A2 50Mb 3ms DropTail]
$ns rtproto Static

#Start the workbench backend on each node
set setupseq [$ns event-sequence {
     $proggroup run -command "sudo
            /usr/wb/bin/experiment-setup.pl"
}]
$ns at 0 "$setupseq start"
$ns run
```

Figure 3: A DETER Experiment Topology Definition

A user can access the nodes by swapping in the experiment. After that, the nodes are reserved and links between them are properly set up, and SEER agents are started on every node.

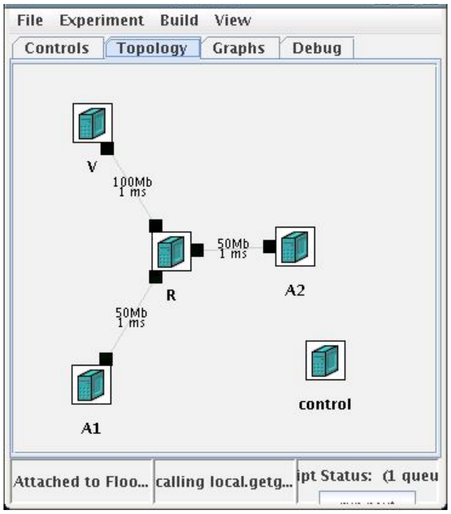
A user can start the SEER GUI to further configure the experiment. Figure 4 depicts the SEER GUI startup window. It shows the current available functions grouped in several categories. A user can attach to the experiment which was just swapped in using the attach command. The user needs to supply his username and account on the DETER server so that the SEER GUI can connect to the DETER server using SSH and execute XMLRPC functions.



No Experiment | Connection Status Here | Script Status

Figure 4: SEER GUI Main Window Frame

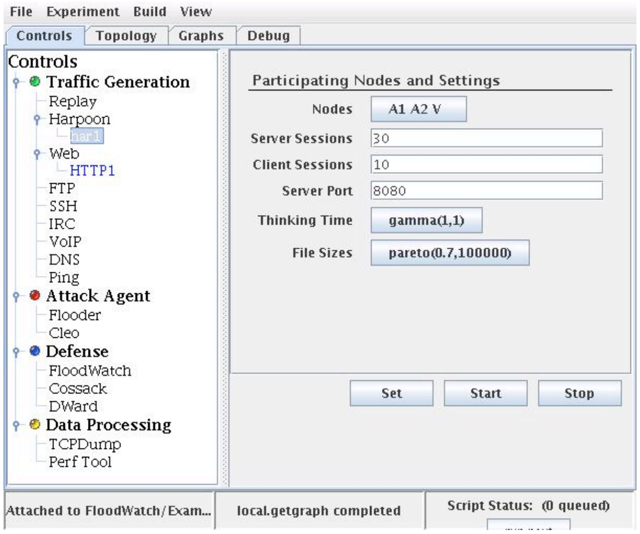
The SEER GUI allows the user to view the topology by clicking on the topology tab (See Figure 5). In addition, it allows the user to simply click on an interface of any node in the topology to visualize the incoming and outgoing packet and data transfer rates. The traffic is color-coded so that legitimate and attack traffic can be visually distinguished from each other.



Attached to Floo... | calling local.getg... | ipt Status: (1 queu

Figure 5: SEER GUI Topology Frame

For nodes that attempt to drop or limit forwarded attack traffic, the forward graph will also display legitimate and attack traffic that has been dropped. This significantly reduces the barrier to experimentation for a novice user.



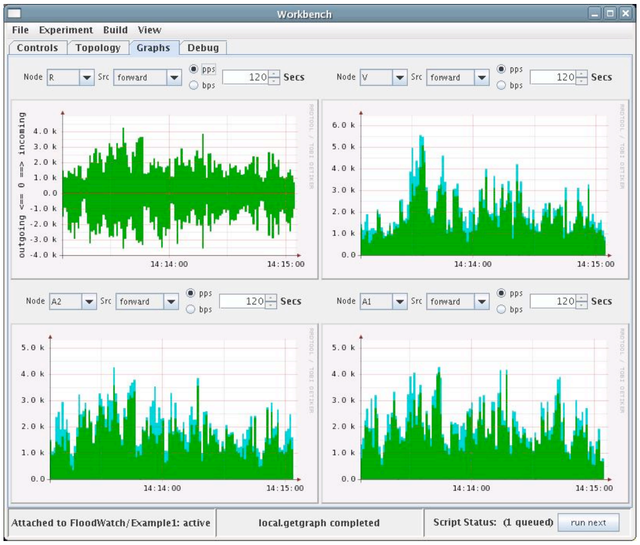
Participating Nodes and Settings

Nodes: A1 A2 V
Server Sessions: 30
Client Sessions: 10
Server Port: 8080
Thinking Time: gamma(1,1)
File Sizes: pareto(0.7,100000)

Attached to FloodWatch/Exam... | local.getgraph completed | Script Status: (0 queued)

Figure 6: Configuration of the Harpoon Traffic Generation Group

A researcher can generate background traffic, deploy and start the defense, and start and stop the attacks at appropriate times. Figure 6 depicts the configuration screen for a group (Har1) of background traffic generators using the Harpoon module. The SEER GUI allows the user to select the parameters, including the node participating in the traffic generation, the time between each web request, the size of the reply, etc. The user can start or stop the harpoon traffic using the start and stop button. Similarly, the user can create a Flooder attack group (a synthetic attack tool developed by SPARTA) to generate attack traffic. Finally, Figure 7 presents a typical graphical display in which an experimenter can monitor the status of several measurements simultaneously as the experiment scenario unfolds on the emulated topology.



Node R Src forward ● pps ○ bps 120 Secs

Node V Src forward ● pps ○ bps 120 Secs

Node A2 Src forward ● pps ○ bps 120 Secs

Node A1 Src forward ● pps ○ bps 120 Secs

Attached to FloodWatch/Example1: active

local.getgraph completed

Script Status: (1 queued)

Figure 7: SEER GUI Graphical Display of Traffic

5 Discussion and Future Work

Several efforts are closely related to SEER. The Emulab Experimentation Workbench [6] is designed for generic networking experimentation. It provides storage, recording, replay, and versioning. The complementary nature of SEER and the Emulab workbench provide the potential for synergy as DETER users can leverage some of the Emulab workbench facilities from within the SEER framework, rather than develop their own variants. The DDoS Benchmarking [3] effort aims at establishing benchmark experiments for evaluating and comparing DDoS defenses. The Benchmarking effort employs SEER to configure DDoS experiments.

SEER eases the process of conducting security experiments so that researchers can focus their attention on the core aspect of the research. Our ultimate goal is to support a wide range of security experiments. Currently, SEER consists of tools for DDoS experimentation. In the future, we will incorporate support for performing experiments with malware such as spyware, adware, and botnets. In particular, additional precautions are required to conduct experiments with malware that might propagate to other hosts. Under a related effort, DETER researchers are developing a Malware Containment capability to enable users across the Internet to access a controlled, isolated Malware experiment. SEER will be extended and tailored to support this class of experiments. A wiki located at http://seer.isi.deterlab.net contains directions for downloading and using SEER on the DETER testbed.

6 Acknowledgements

This material is based upon work supported by the Department of Homeland Security, and Space and Naval Warfare Systems Center, San Diego, under Contract No. N66001-07-C-2001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Department of Homeland Security for the Space and Naval Warfare Systems Center, San Diego.

7 References

[1] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experience with DETER: A Testbed for Security Research. In *Proceedings of the 2nd IEEE Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2006)*, Barcelona, SPAIN, March 2006.

[2] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integratede Experimental Environment for Distributed Systems and Networks. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (ODSI)*, pp. 255-270, Boston, MA, Dec. 2007.

[3] J. Mirkovic, S. Wei, A. Hussain, B. Wilson, R. Thomas, S. Schwab, S. Fahmy, R. Chertov, and P. Reiher. DDoS Benchmarks and Experimentation Workbench for the DETER Testbed. In *Proceedings of the 3rd IEEE Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2007)*, Orlando, FL, May 2007.

[4] A. Hussain, S. Schwab, R. Thomas, S. Fahmy, and J. Mirkovic. DDoS Experiment Methodology. In *Proceedings of the DETER Community Workshop on Cyber Security Experimentation*, Arlington, VA, June 2006.

[5] S. Schwab, B. Wilson, and R. Thomas. Methodologies and Metrics for the Testing and Analysis of Distributed Denial of Service Attacks and Defenses. In *Proceedings of IEEE MILCOM*, Atlantic City, NJ, 2005.

[6] E. Eide, L Stoller, and J. Lepreau. An Experimentation Workbench for Replayable Networking Research. In *Proceedings of the Fourth USENIX Symposium on Networked Systems Design and Implementation* (NSDI), Cambridge, MA, Apr. 2007.