Automating DDoS Experimentation

Jelena Mirkovic
University of Delaware
Newark, DE

Brett Wilson
SPARTA, Inc.
El Segundo, CA

Alefiya Hussain
SPARTA, Inc.
El Segundo, CA

Sonia Fahmy
Purdue University
West Lafayette, IN

Peter Reiher
UCLA
Los Angeles, CA

Roshan Thomas
SPARTA, Inc.
Centreville, VA

Stephen Schwab
SPARTA, Inc.
El Segundo, CA

***Abstract*— While the DETER testbed provides a safe environment and basic tools for security experimentation, researchers face a significant challenge in assembling the testbed pieces and tools into realistic and complete experimental scenarios. In this paper, we describe our work on automating experimentation for distributed denial-of-service attacks. We developed the following automation tools: (1) the Experimenter's Workbench that provides a graphical user interface, tools for topology, traffic and monitoring setup and tools for statistics collection, visualization and processing, (2) a DDoS benchmark suite that contains a set of diverse and comprehensive attack scenarios, (3) the Experiment Generator that combines chosen AS-level and edge-level topologies, legitimate traffic and a set of attacks into DETER-compatible scripts. Jointly, these tools facilitate easy experimentation even for novice users.**

I. Introduction

The DETER testbed [1] allows security researchers to replicate threats of interest in a secure environment and to develop, deploy and evaluate potential solutions. DETER's sister project EMIST [9] provides a collection of tools for traffic generation, statistics collection, analysis and visualization. Jointly, DETER and EMIST [9] facilitate reconstruction of numerous security scenarios, where every element of the scenario is customizable by the researcher. However, the tasks of (1) choosing realistic traffic and topology settings for experimentation, and (2) setting up and controlling experiments in an automated fashion remain open problems.

In this paper, we describe our work[1] on automating DDoS experimentation via three toolkits: (1) The *Experimenter's Workbench*, which provides a set of traffic generation tools, topology and defense libraries and a graphical user interface for experiment specification, control and monitoring, (2) The *DDoS benchmarks* that provide a set of comprehensive topology, legitimate and attack traffic specifications, and (3) The *Experiment Generator* that receives an input either from a user via the Workbench's GUI or from the benchmark suite, and glues together a set of selected topologies, legitimate and attack traffic into a DETER-ready experiment. This experiment can be deployed and run from the Workbench at the click of a button. Jointly, our tools facilitate easy DDoS experimentation even for novice users by providing a point-and-click interface for experiment control and a way to generate realistic experiments with minimal effort. Each component of an experiment, such as topological features, legitimate and attack traffic features, and performance measurement tools can also be customized by a user, facilitating full freedom for experimentation without the burden of low-level code writing and hardware manipulation.

[1]This material is based on research sponsored by the Department of Homeland Security under contract number FA8750-05-2-0197, and by the Space and Naval Warfare Systems Center, San Diego, under contract number N66001-07-C-2001. The views and conclusions contained herein are those of the authors only.

II. The Experimenter's Workbench

The Experimenter's Workbench enables even a novice experimenter to reproduce complex scenarios by selecting various experimental elements from a pre-defined palette, using an intuitive graphical user interface. The palette includes: (1) a library of legitimate and attack traffic generators, (2) a library of DDoS defenses that are deployed in the DETER testbed, (3) a library of experiment statistics collection tools and performance measures that operate on these statistics (e.g., tcpdump collection). The GUI enables easy deployment of palette elements to an existing or a new DETER experiment, experiment control (start, stop, restart) and traffic monitoring and visualization. The workbench also contains a library of topologies along with routing configurations that can be used in experiments.

Figure 1 shows the main GUI window (Controls tab, below the menu bar) with the palette on the left. Experimentation starts with a user attaching an existing DETER experiment to the GUI or using the GUI to start a new experiment (Experiment item on the menu bar). We now describe the current contents of the palette but note that these can easily be extended by adding new elements. The currently supported **legitimate traffic generators** are: (1) A replay tool that replays traffic from

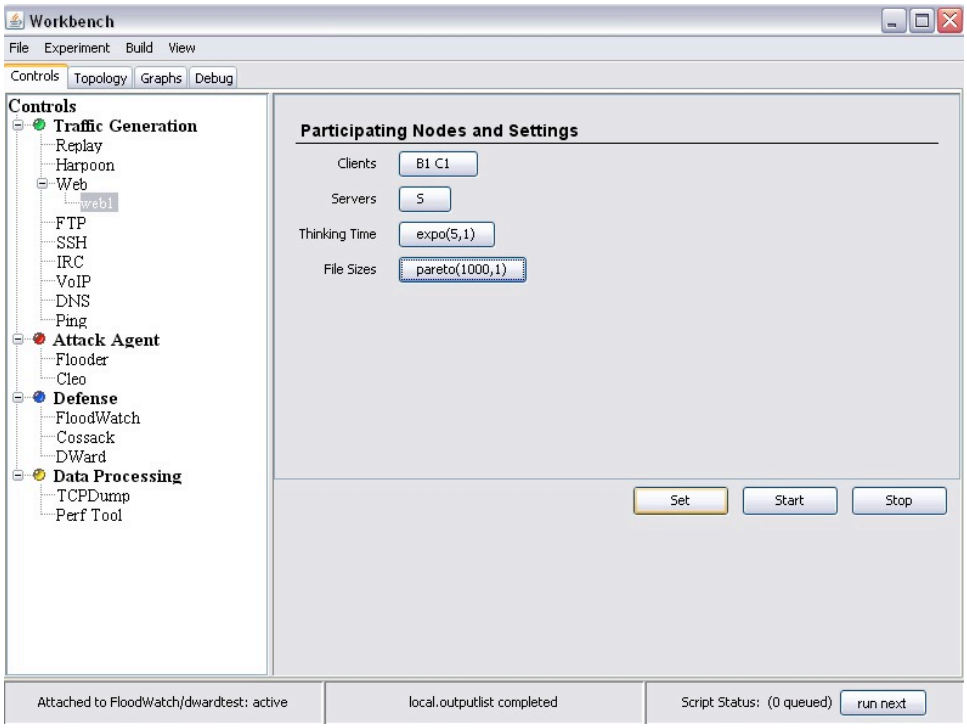

Fig. 1. Main GUI window with the palette

a tcpdump trace, (2) Harpoon [10], a flow-level traffic generator, which generates synthetic traffic whose statistical properties are mined from an input Netflow trace, (3) Several application traffic generators that generate traffic using real client and server applications according to user-specified distributions. Supported application traffic types are: Web, FTP, SSH, IRC, VoIP, DNS and Ping. For the replay tool and Harpoon, the user supplies the trace file and specifies the set of nodes whose communication should be replayed from the file, along with a few additional, tool-specific parameters such as number of client and server sessions for Harpoon. For application traffic generators, the user specifies the distribution of request and reply sizes and the request interarrival time distribution. The supported distributions are Paretto, Gamma, Exponential and MinMax. During the experiment, traffic generator tools may not be able to generate traffic according to these predefined parameters because of testbed limitation or congestion caused by attacks. To amend the first problem we allow for request and response scaling so that the generated traffic can be supported by limited resources while still maintaining timing and message size distribution selected by the user. We view the second phenomenon, modification of traffic because of attacks, as a desired feature in DoS experimentation, and do not try to amend it.

There are two currently supported **attack traffic generators**: (1) UCLA's Cleo tool and (2) SPARTA's Flooder tool. Both tools generate flooding attacks, with a customizable protocol, packet sizes, packet rates, spoofing and rate dynamics (flat, pulse, ramp up and ramp down). The Flooder tool provides additional customizable parameters such as TCP flag settings, ICMP codes, pulses with ramp up and ramp down features, etc. The Cleo tool provides an option of organizing attack machines into a master-slave architecture.

The palette currently supports automatic placement and configuration of three **defense systems**: (1) SPARTA's FloodWatch defense [13] that detects and filters attacks based on the entropy measure and Chi-square statistic of various traffic parameters, (2) USC/ISI's Cossack defense [8] that forms a multicast group of collaborating, distributed Snort modules that communicate to detect and suppress attacks and (3) UCLA's D-WARD defense [5] that is deployed at the source-end to prevent network participation in attacks. Parameters specific to each defense and defense deployment points can be specified from the palette.

There are two **statistics collection** tools: (1) The tcpdump tool and (2) The Perf tool, which calculates several denial-of-service measures [6] we developed during our benchmark work. Tool-specific parameters and monitor deployment locations can be specified from the palette. Note that tcpdump may fail to collect all packets if the host CPU is too busy, a situation not unusual in DoS experiments. To amend this problem we have developed

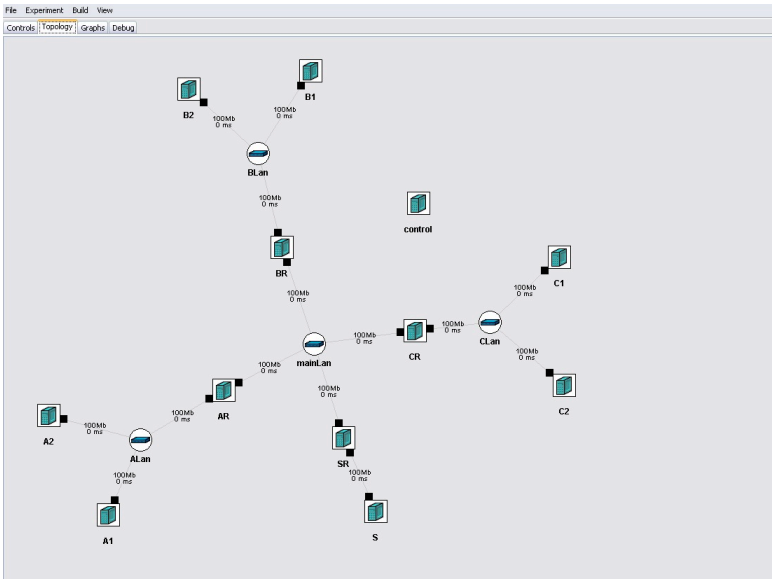

Fig. 2. Topology tab with a sample topology

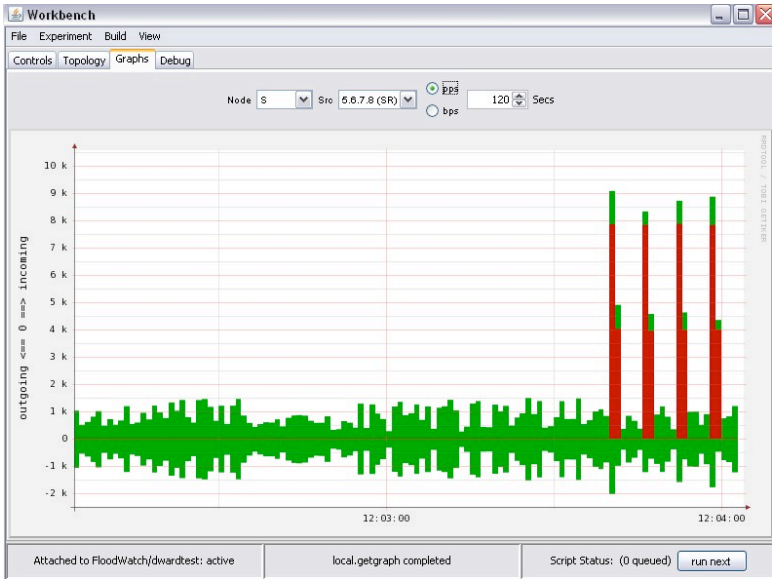

Fig. 3. Traffic visualization at the node S

two device drivers that are integrated with the Click modular router [3]. The device drivers provide direct access to Click's kernel memory from a user application, hence bypassing the default IP stack. Thus, the task of tcpdump is split into two parts. The filtering part resides in kernel-level Click, while the packet writes to disk are in the user-level application. Our simple test application was able to read packets from Click's buffer at over 800 KPackets/s. Writing the packets to disk was achieved at a rate of 220 KPackets/s. Thus, it is now possible to filter and capture packets of interest to disk at very high rates, increasing the fidelity of the measurements.

The palette enables composable experimentation. Each element can be instantiated as many times as needed, with various parameter settings. For example, to generate a mixture of Web traffic from two clients to a common server, where one client's requests follow an exponential distribution while the other client's requests arrive uniformly, two Web elements would need to be instantiated, one for each parameter setting (distribution). If both clients communicated with the server (or a set of servers) with the same distribution of request interarrival times, request size and reply size, a single Web element could be used to generate their traffic. In Figure 1 at the

right part of the window we show instantiation of a Web element called web1 for the topology shown in Figure 2. Client and server nodes are selected by the user from a pop-up menu, showing all the nodes in the experiment's topology. In our example we selected nodes B1 and C1 to act as clients, node S as a server, and specified an exponential distribution of the request interarrival times and a Paretto distribution of the reply sizes. The second argument in the distributions is the multiplicative factor, used to scale samples up or down.

The Topology tab displays the experimental topology, as shown in Figure 2. The black squares represent network interfaces. During experimentation, interfaces that relay traffic are colored green or red, depending if the traffic is legitimate or attack. The size of the colored portion relative to the square size is proportional to the bandwidth consumption at that interface.

Left clicks on the interfaces or nodes open the graphs representing traffic statistics (packets or bytes per second) in the Graph tab. Multiple graphs can be opened. We show one sample graph in Figure 3, for an experiment with the topology shown in Figure 2. Legitimate Web and FTP traffic is generated from B1, B2, C1 and C2 to S. Additionally, A1 and A2 send a high-volume, pulsing UDP flood to S. The graph shows incoming (top portion) and outgoing (bottom portion) traffic in packets per second at the node S, with legitimate traffic colored green and attack traffic colored red.

The Workbench is a stand-alone JAVA application, and is easily run on a variety of operating systems. It runs locally on the user's desktop, and communicates with the DETER's experiment server using xmlrpc to send commands to each node in the topology. In addition to the GUI, the workbench also provides support for experiment automation and repeatability via a Perl-based scripting interface. For example, the following script sets up the same web1 traffic generator shown in Figure 1, and runs the traffic for 120 seconds.

```
use Agent;
use EventTx;

$tx = EventTx::New();

# Create Web traffic
$web1 = Agent::New($tx, 'HTTP', 'web1');
$web1->SetLocation('B1 C1 S');
$web1->Set(
        servers=>'S',
        think=>'expo(5,1)',
        sizes=>'pareto(1000,1)'
);

# Start traffic
$web1->Start();
$tx->wait(120);

# Stop traffic
$web1->Stop();
```

The scripting language allows an experienced user to rapidly execute a large set of experiments in the batch mode.

III. DDoS Benchmarks

In the previous section we discussed the challenge of setting up and controlling large, distributed experiments. Another challenging task in any experiment is choosing realistic and comprehensive scenarios, and reproducing them in a testbed or in a simulation. In case of DDoS experimentation, these scenarios consist of three dimensions: (1) **Attack traffic** — features describing a malicious packet mix arriving at the victim, and the distribution and activities of machines involved in the attack. (2) **Legitimate traffic** — features describing communication patterns of the target network. (3) **Network topology and resources** — features describing the target network architecture.

Our work on DDoS benchmarks addresses this challenge by developing test scenarios with realistic topologies and legitimate traffic patterns, and with a comprehensive suite of attacks. To develop these scenarios we designed a collection of tools that harvest traffic and topology samples from the Internet.

A. Legitimate traffic

The *LTProf* tool produces legitimate traffic models that describe communication between a set of active clients and a network that is the target of a DDoS attack. The tool collects legitimate traffic samples from public traces by creating a communication profile for each observed subnet and deriving relevant traffic feature distributions from these profiles. These distributions then serve as an input to the Workbench's traffic generators.

We build subnet models by first identifying /24 and /16 subnets in a traffic trace anonymized in a prefix-preserving manner. For each subnet, we identify the total traffic received by it and select the largest receivers to act as target networks in our scenarios. We then identify subnets that send a significant percentage of traffic to this target network and model their sending behavior.

We model separately a sender's outgoing traffic for each well-known port number. Within the selected traffic mix, we identify individual sessions between two IP addresses and extract the distributions of the number and length of service requests, the reply length and the request inter-arrival time. These distributions are used during an experiment to drive the traffic generation. For example, the outgoing traffic from the anonymized network 0.3.117.0/24 in the CAIDA's OC48 traffic trace consists of traffic to port 53 and port 80, with the characteristics shown in Table I. The *LTProf* tool automates this traffic modeling and produces for each target network a set of outgoing traffic models for its most active client subnets. These models can be fed directly into the Workbench's traffic generators.

port	traffic feature	distribution
53	Requests per second	Poisson(1.828)
	Requests per host	Paretto(1.1,2.17)
	Requests size	Paretto(32.74,2.5)
	Reply size	Paretto(117.5,3.1)
80	Requests per second	Poisson(94.147)
	Requests per host	Paretto(10,2.315)
	Requests size	Paretto(287,2.35)
	Reply size	Paretto(259,2.028)

TABLE I
OUTGOING TRAFFIC MODELS FOR 0.3.117.0/24

B. Topologies

For DDoS experimentation, we are interested in modeling topologies of the target network and its Internet Service Provider. We will refer to these as *end-network* topology and *AS-level* topology.

AS-level topologies consist of router-level connectivity maps of selected Internet Service Providers. They are collected by the *NetTopology* tool, which we developed. The tool probes the topology data by invoking traceroute commands from different servers, performing alias resolution, and inferring several routing (e.g., Open Shortest Path First routing weights) and geographical properties. This tool is similar to *RocketFuel* [14], and was developed because *RocketFuel* is no longer supported.

We further developed tools to generate DETER-compatible input from the sampled topologies: (i) *RocketFuel-to-ns*, which converts topologies generated by the *NetTopology* tool or *RocketFuel* to DETER `ns` scripts, and (ii) *RouterConfig*, a tool that takes a topology as input and produces router BGP and OSPF configuration scripts.

A major challenge in a testbed setting is the scale-down of a large, multi-thousand node topology to a few hundred nodes available on DETER [1], while retaining relevant topology characteristics. The *RocketFuel-to-ns* tool allows a user to select a subset of a large topology, specifying a set of Autonomous Systems or performing a breadth-first traversal from a specified point, with specified degree and number-of-nodes bounds.

The *RouterConfig* tool operates both on (a) topologies based on real Internet data, and on (b) topologies generated from the GT-ITM topology generator [18]. To assign realistic link bandwidths in our topologies, we use information about typical link speed distribution published by the Annual Bandwidth Report [16].

Since many end-networks filter outgoing ICMP traffic, the *NetTopology* tool cannot collect end-network topologies. To overcome this obstacle, we analyzed enterprise network design methodologies typically used in the commercial marketplace to design and deploy scalable, cost-efficient production networks. An example of this is Cisco's classic three-layer model of hierarchical network design that is part of Cisco's Enterprise Composite Network Model [7], [17]. This consists of the topmost core layer which provides Internet access and ISP connectivity choices, and a middle distribution layer that connects the core to the access layer and serves to provide policy-based connectivity to the campus. Finally, the bottom access layer addresses the design of the intricate details of how individual buildings, rooms and work groups are provided network access, and typically involves the layout of switches and hubs. We used these design guidelines to produce end-network topologies with varying degrees of complexity and redundancy. One such topology is shown in Figure 4.

C. Attack traffic

To generate comprehensive attack scenarios we sought to understand which features of the attack interact with the legitimate traffic, the topology and the defense. We first collected information about all the known DoS attacks and categorized them based on the mechanism they deploy to deny service. We then selected for further consideration only those DoS attacks that require distribution. These are packet floods and congestion control exploits. Packet floods deny service by exhausting some key resource. This resource could be bandwidth (if the flood volume is large), router or end host CPU (if packet rate of the flood is high) or tables in memory created by the end host operating system or application (if each attack packet creates a new record in some table). Packets in bandwidth and CPU exhaustion floods can belong to any transport and application protocol, as long as they are numerous, and may contain legitimate transactions, e.g., flash crowd attacks. An attacker can use amplification effects such as reflector attacks to generate large-volume floods. Examples of memory exhaustion floods are TCP SYN floods and random fragment floods.

In congestion control exploits the attacker creates the impression at a sender that there is congestion on the path. If the sender employs a congestion control

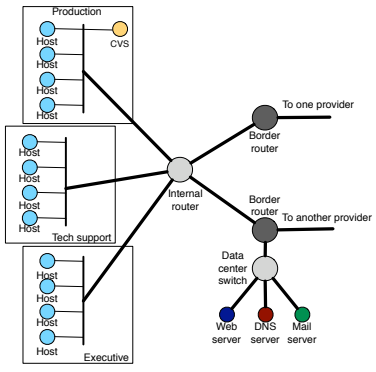

Fig. 4. A sample end-network topology

mechanism, it reduces its sending rate. One example of such attacks is the shrew attack with pulsing flood [4].

Table II lists all the attack types in the benchmark suite and their denial-of-service mechanisms. Although there are a few attack categories, they can invoke a large variety of DoS conditions and challenge defenses by varying attack features such as sending dynamics, spoofing and rates.

Attack type	DoS mechanism
UDP/ICMP packet flood	Large packets consume bandwidth, while small packets consume CPU
TCP SYN flood	Consume end-host's connection table
TCP data packet flood	Consume bandwidth or CPU
HTTP flood	Consume Web server's CPU or bandwidth
DNS flood	Consume DNS server's CPU or bandwidth
Random fragment flood	Consume end-host's fragment table
TCP ECE flood	Invoke congestion control
ICMP source quench flood	Invoke congestion control

TABLE II

ATTACK TYPES IN THE COMPREHENSIVE BENCHMARK SUITE

Attack traffic generated by the listed attacks interacts with legitimate traffic by creating real or perceived contention at some critical resource. The level of service denial depends on the following traffic and topology features: (1) Attack rate, (2) Attack distribution, (3) Attack traffic on and off periods in case of pulsing attacks, (4) The rate of legitimate traffic relative to the attack, (5) Amount of critical resource — size of connection buffers, fragment tables, link bandwidths, CPU speeds, (6) Path sharing between the legitimate and the attack traffic prior to the critical resource, (7) Legitimate traffic mix at the TCP level — connection duration, connection traffic volume and sending dynamics, protocol versions at end hosts, (8) Legitimate traffic mix at the application level — since different applications have different quality of service requirements, they may or may not be affected by a certain level of packet loss, delay or jitter.

If we assume that the legitimate traffic mix and topological features are fixed by inputs from our legitimate traffic models and topology samples, we must vary the attack rate, distribution, dynamics and path sharing to create comprehensive scenarios. Additionally, presence of IP spoofing can make attacks more challenging to some DDoS defenses. Table III lists the feature variations included in our benchmark suite for each attack type listed in Table II.

The benchmark suite contains a list of attack specifications that include: attack type (from table II), packet size (large or small), attacker deployment pattern (uniform or clustered), attack dynamics (flat rate, synchronous pulse and interleaved pulse), attack rate (low, moderate or large), attacker aggressiveness (whether the chosen attack rate is spread over a low, moderate or large number of attackers) and spoofing type. All combinations of attack features are explored, but those that are contradictory, e.g., spoofing with an application-level attack such as HTTP flood, are discarded.

Feature	Variation
Rate	Low, moderate and large
Attacker aggressiveness	Low, moderate and large
Dynamics	Continuous rate vs. pulsing (vary on and off periods) Synchronous senders vs. interleaved senders
Path sharing	Uniform vs. clustered locations of attack machines; legitimate clients are distributed uniformly
Spoofing	None, subnet, fixed IP and random

TABLE III

ATTACK FEATURE VARIATIONS THAT INFLUENCE DOS IMPACT

IV. EXPERIMENT GENERATOR

The Experiment Generator receives as input (1) AS-level and edge-network topologies from the topology library, (2) legitimate traffic models generated by the *LTProf* tool, and (3) list of attacks. It glues these elements together into an `ns` file containing topology specification and a collection of Perl scripts, one for each attack from the list and one script for legitimate-traffic-only testing.

The AS-level and the edge-network topology are glued together by the *TopologyMerge* script in the following manner:

1) We identify the nodes in the AS-level topology that have less than 9 neighbors, with 9 being the maximum number of network interfaces a node can have in DETER. Identified nodes become candidates for expansion.
2) We identify border routers in the edge-network topology and connect each to one candidate AS node, via a limited-bandwidth link.
3) One input parameter to our script is the desired number of edge networks in the final topology. This is relevant for experiments that test collaborative defenses and need several copies of the edge networks to deploy the defense at each copy. If the input number of copies is greater than 1 we create multiple copies of the edge network and repeat step (2) for each copy.
4) Each candidate AS-node is further expanded by attaching to it several newly created nodes to play the role of the external subnet communicating with the attack's target, or the role of an attacker. We call these new nodes *clones*. The number of clones to be attached to each AS-node is given as an input parameter to our script.
5) Each clone is assigned a /24 or /16 address range (an input parameter controls the size of the range)

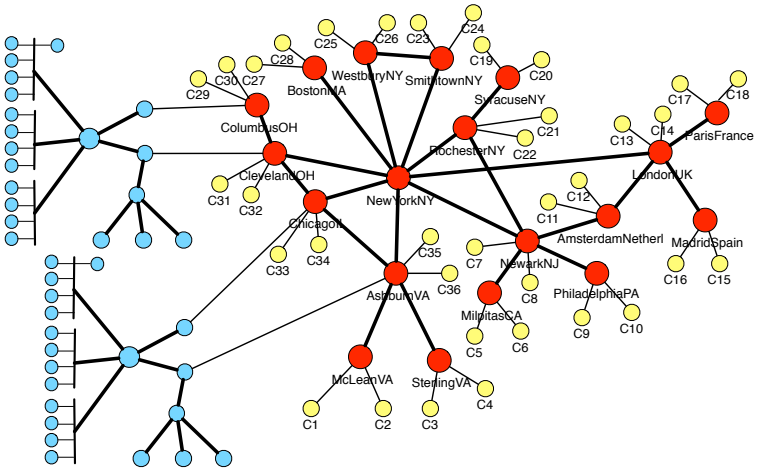

Fig. 5. Merging of the NTT America AS topology (red nodes) with two copies of the edge network (blue nodes) and two clones per candidate AS-node (yellow nodes).

and routing is set accordingly. We assign multiple addresses to clone nodes to generate sufficient source address diversity at the target — this is an important feature for some DDoS defenses that model entropy of source IP addresses.

Figure 5 illustrates merging of the NTT America's AS topology with two copies of the edge network shown in Figure 4 and with two clones per each candidate AS-node. The outcome of the merging is an `ns` file that can be used to create a new experiment on the DETER testbed, either manually or via the Workbench GUI.

We produce the Perl scripts for experimentation by feeding the topology file, the legitimate traffic and the attack traffic specifications into the *CreateScenario* script. The script identifies clone nodes in the topology and randomly chooses some to play the role of legitimate subnets. After parsing the legitimate traffic models, commands are generated to set up traffic generators at selected clone nodes. The remaining clone nodes are called *free clones*. For each attack description the script iterates through the following steps:

1) Print the standard Perl script preamble and legitimate traffic setup commands to a new output file.
2) Calculate attack rate in packets per second, taking into account the packet size. Small-packet attacks target CPU resources, while large-packet attacks target bandwidth. Since we know hardware capacities in DETER we can easily calculate the attack rate that exhausts CPU or bandwidth in our target topology.
3) Calculate the maximum capacity of an individual attacker (in packets or bytes per second) and, by dividing attack rate with capacity, obtain the minimum number of attackers.
4) For very aggressive attackers, the minimum number of attackers will be deployed. For non-aggressive attackers, all free clones become attack nodes. The number of attack nodes for moderately aggressive attackers lies in between these two cases, and is roughly one half of all free clones.
5) Select clone nodes to act as attackers according to the attacker distribution. This option only makes sense for very and moderately aggressive attackers, since non-aggressive attackers use each free clone node. For the uniform distribution, attack locations are selected at random among free clones. At the same time, to maximize path sharing, the script ensures that each AS-node that has a legitimate clone also has an attacker clone. For the clustered distribution, the first attack node is deployed at random. The remaining nodes are deployed iteratively: (a) Assign a probability to each free clone to be selected, with higher probabilities given to clones in the neighborhood of attack nodes, and (b) Flip a coin for each free clone biased by its selection probability and deploy new attack nodes on clones favored by the coin flip.
6) Generate attack setup commands for selected nodes, attack type, attack rate, attack dynamics and spoofing strategy.
7) Generate commands for starting and stopping legitimate traffic generators, attack traffic generators and traffic statistics collection.

The result of the *CreateScenario* script is a set of Perl scripts that a user can run manually or via the Workbench's GUI. We are also working on automated batch testing where the Workbench will run multiple selected scripts and store results for later user's review.

V. Related Work

For space reasons we provide a brief overview of work related to automated testing and benchmarking. In [11] Sommers et al. propose a framework for malicious workload generation called MACE. MACE provides an extensible environment for construction of various malicious traffic, such as intrusions, worms and DDoS attacks, but only a few attack generators are implemented. In DDoS realm, MACE only produces SYN flood and fragment flood attacks, with optional spoofing. Our generation tools provide a wider variety of attacks. Similarly, Vigna et al. propose automatic generation of exploits in [15], for testing of intrusion detection systems. Our work focuses on DDoS attacks.

In [12], Sommers et al. propose a traffic generation method for online Intrusion Detection System evaluation. This work relies solely on Harpoon for traffic generation, and contains a limited number of simple DoS attacks. We use a wider variety of traffic generators and attacks.

The Center for Internet Security has developed benchmarks for evaluation of operating system security [2], and large security bodies such as CERT and SANS maintain checklists of known vulnerabilities that can be used by software developers to test the security of their code. However, much remains to be done to define rigorous and representative tests for various security threats, and we tackle this problem for DDoS attacks.

VI. Conclusions

Various approaches for handling DDoS attacks have been hard to study and compare because of lack of a common facility and experimental methodology. The DETER testbed provides the necessary facility, and the work described in this paper provides the methodology. We have built a complete set of tools that will allow even novices to quickly run standardized experiments on various DDoS situations. Since the tools are common, experiments run by different groups will be more directly comparable than they have been in the past.

Our automation mechanisms are based on realistic choices of topologies and legitimate traffic patterns. Our attack tools are highly parameterizable and are based on observations of properties of real DDoS attacks. We have included three open source defense mechanisms for experimenters to work with, and plan to add more as they become available in sufficiently stable form. Experimenters can easily integrate their own defenses or new topologies and traffic generators into the Workbench, which is an open-source tool. We have provided both a full graphical interface and a powerful scripting capability for creating experiments that cover an extremely wide range of the possibilities. The Workbench is currently being transformed into the Security Experimentation EnviRonment (SEER). It is currently available in the DETER testbed (at `http://seer.isi.deterlab.net/`), and is being extended to support experimentation with other threats such as worms, routing attacks etc. The Experiment Generator is also integrated with SEER and we are looking into extending it for other experiments, beyond DoS.

Our tools will significantly ease the difficult problem of performing high quality experiments with DDoS attacks. They will be useful not only to researchers, but to students and educators who need to learn about denial of service. Our future work will focus on extending our collections of traffic generators, topologies, defenses and statistics collection tools, and on applying automation to experiments with other network security threats.

References

[1] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experiences With DETER: A Testbed for Security Research. In *2nd IEEE TridentCom*, March 2006.

[2] The Center for Internet Security. CIS Standards Web Page. `http://www.cisecurity.org/`.

[3] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[4] A. Kuzmanovic and E. W. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants). In *ACM SIGCOMM 2003*, August 2003.

[5] J. Mirkovic. *D-WARD: source-end defense against distributed denial-of-service attacks*. PhD thesis, UCLA, 2003.

[6] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher, R. Thomas, W. Yao, and S. Schwab. Towards User-Centric Metrics for Denial-Of-Service Measurement. In *Workshop on Experimental Computer Science*, 2007.

[7] P. Oppenheimer. *Top-Down Network Design*. CISCO Press, 1999.

[8] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan. COSSACK: Coordinated Suppression of Simultaneous Attacks. In *Proceedings of DISCEX*, pages 2–13, 2003.

[9] EMIST project. Evaluation methods for internet security technology. `http://www.isi.edu/deter/emist.temp.html`.

[10] J. Sommers, H. Kim, and P. Barford. Harpoon: A Flow-Level Traffic Generator for Router and Network Tests. In *ACM SIGMETRICS*, 2004.

[11] J. Sommers, V. Yegneswaran, and P. Barford. A Framework for Malicious Workload Generation. In *ACM Internet Measurement Conference*, 2004.

[12] J. Sommers, V. Yegneswaran, and P. Barford. Toward Comprehensive Traffic Generation for Online IDS Evaluation. Technical report, Dept. of Computer Science, University of Wisconsin, August 2005.

[13] Sparta, Inc. A Distributed Denial-of-Service Detection and Response System. `http://www.isso.sparta.com/research/documents/floodwatch.pdf`.

[14] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with RocketFuel. In *Proceedings of ACM SIGCOMM*, 2002.

[15] G. Vigna, W. Robertson, and D. Balzarotti. Testing Network-based Intrusion Detection Signatures Using Mutant Exploits. In *ACM Conference on Computer and Communication Security*, 2004.

[16] Websiteoptimization.com. *The Bandwidth Report*. `http://www.websiteoptimization.com/bw/`.

[17] R. White, A. Retana, and D. Slice. *Optimal Routing Design*. CISCO Press, 2005.

[18] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. of IEEE INFOCOM*, volume 2, pages 594 –602, March 1996.