BreakingPoint
Powering Accelerated Development

Simulating Distributed Denial of Service with BreakingPoint

April 30, 2009

Dustin D. Trammell
<dtrammell@breakingpoint.com>

Todd Manning
<tmanning@breakingpoint.com>

BreakingPoint Labs
http://www.breakingpoint.com

Contents

List of Figures

Chapter 1

Foreword

This paper and the accompanying test cases are intended to illustrate how the BreakingPoint products can be used to simulate Distributed Denial-of-Service[1] (DDoS) attacks using some of the product's various components.

The test cases provided for the scenarios described herein are meant to be provided as example material and further tailored to suit the user's specific needs. Note that the accompanying test cases were developed for the BreakingPoint Elite product. As such, import and use of these tests with the BreakingPoint Appliance products (BPS 1K and BPS 10k) is not likely to be compatible.

It is also important to note that when modeling DDoS attacks, large numbers of individual hosts are involved as well as large numbers of repetitive network packets or protocol sequences. While the AppSim component's User Interface is well-suited for creation of normal traffic scenarios, due to the time involved in manually configuring DDoS scenarios with these large numbers of hosts and AppSim actions from within the AppSim component User Interface, many of the test cases provided were created by utilizing TCL scripting within the BreakingPoint TCL command interface. Individual test cases that have been created this way will be accompanied by the script that created them, and a general introduction to BreakingPoint TCL scripting can be found in Appendix A.

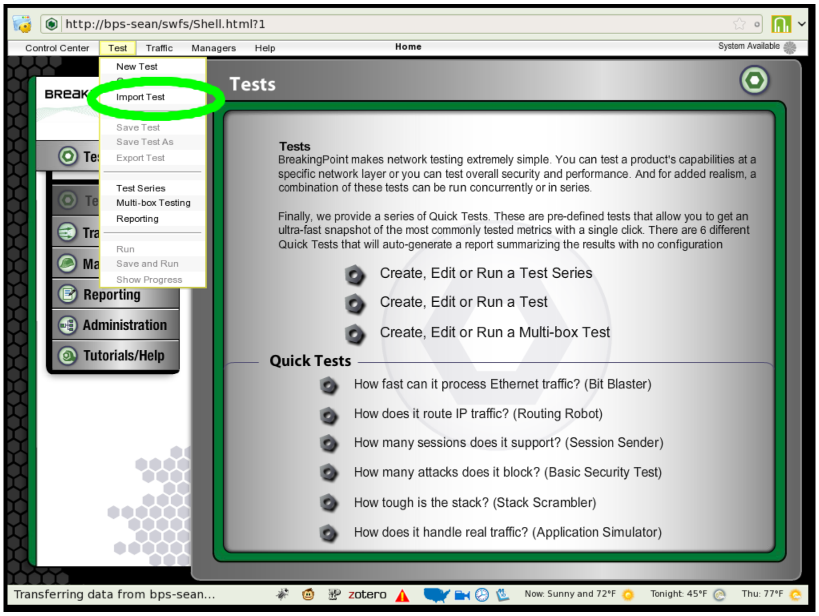
---

[1] A Distributed Denial-of-Service attack occurs when a large collection of multiple networked systems flood the bandwidth or resources of a targeted system with individually small or insignificant packets or network sessions.[1] When aggregated, the resultant traffic overwhelms the target.

Chapter 2

BreakingPoint System Preparation

Accompanying this document are a number of additional files. All of the files found within the "test-cases" sub-directory ending with the ".bpt" extension are BreakingPoint Test files suitable for import into your BreakingPoint Elite system. These files should be imported via the "Import Test" menu option (Figure 2.1). Importing these files will set up the various Network Neighborhoods, SuperFlows, App Profiles, and Tests which accompany the attack scenarios presented throughout this document. It is expected that the reader inspect each test case in the BreakingPoint product while reading the related section about its associated attack scenario to achieve full comprehension of the scenario discussed and how the test case accomplishes simulation of such.



Tests
BreakingPoint makes network testing extremely simple. You can test a product's capabilities at a specific network layer or you can test overall security and performance. And for added realism, a combination of these tests can be run concurrently or in series.

Finally, we provide a series of Quick Tests. These are pre-defined tests that allow you to get an ultra-fast snapshot of the most commonly tested metrics with a single click. There are 6 different Quick Tests that will auto-generate a report summarizing the results with no configuration

- Create, Edit or Run a Test Series
- Create, Edit or Run a Test
- Create, Edit or Run a Multi-box Test

Quick Tests

- How fast can it process Ethernet traffic? (Bit Blaster)
- How does it route IP traffic? (Routing Robot)
- How many sessions does it support? (Session Sender)
- How many attacks does it block? (Basic Security Test)
- How tough is the stack? (Stack Scrambler)
- How does it handle real traffic? (Application Simulator)

Figure 2.1: Import Test Menu

Also accompanying this document are files in the "bps-user-files" sub-directory. These files should be uploaded to your BreakingPoint Elite system for use by the various AppSim SuperFlows. This can be accomplished by browsing to the following URL on your BreakingPoint System's web interface:

```
http://bps/Import.html?resource
```

Replace “bps” in the URL above with your system’s hostname or network address.

Chapter 3

Configuring the Network Neighborhood

Unlike normal network traffic, DDoS traffic has some unique and significant properties. The general traffic profile of a DDoS consists of an arbitrarily large number of network sources directing traffic at a single or small group of targets. In addition to this attack traffic, there is also likely to be an arbitrary amount of legitimate traffic traversing the same network.

As an example, if a web server is under a DDoS attack, while the attack traffic may be all directed at port 80 and be consuming the web server's resources so that it cannot accept or respond to legitimate requests, those legitimate requests may still be traversing the network destined for the web server. In addition, there is likely traffic on the network between other services on the web server and other network systems, or among other network systems unrelated to the web server that is under attack.

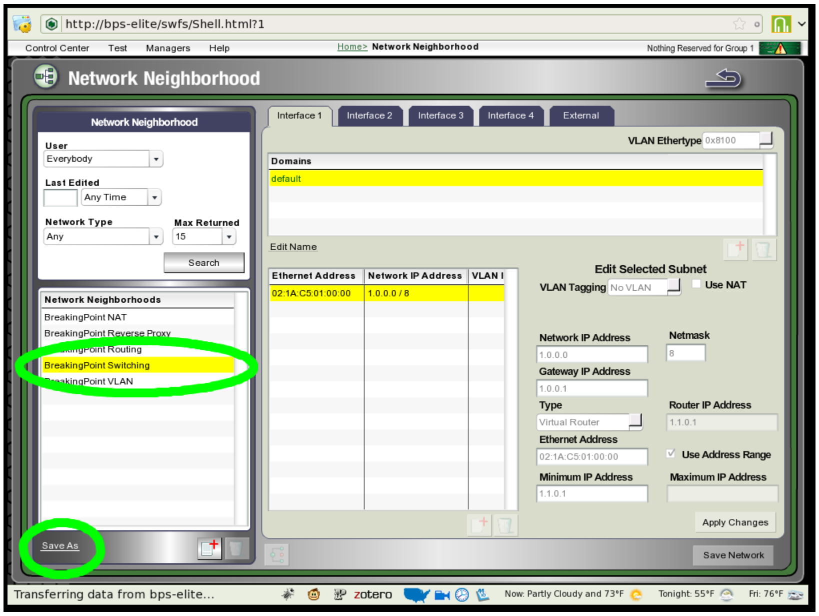
To accomplish simulating this traffic profile, a customized Network Neighborhood is required. This customized Network Neighborhood will be used by all of the test cases for the attack scenarios described herein.

3.1 Attack Traffic

In order to simulate a DDoS of an arbitrary number of sources directing traffic at a single or small group of targets, our custom Network Neighborhood will need a specially configured Domain to represent this, which we will name the DDoS Domain. Specifically, the two networks configured in the DDoS Domain as the Client (source) and Server (destination) sides of the sessions must be configured so that the BreakingPoint system's Network Processor is restricted to the appropriately sized pools of available network addresses when generating traffic.

The simplest way to create a new custom Network Neighborhood is to copy the BreakingPoint Neighborhood that is most applicable to the Device Under Test (DUT). For the purposes of this paper, a copy of the *BreakingPoint Switching* Neighborhood will be used, named *DDoS Switching*. Browse to the Network Neighborhood configuration screen by selecting the *Network Neighborhood* option from the *Control Center* menu. If you prepared your BreakingPoint system as described in Chapter **??**, the *DDoS Switching* Neighborhood should already exist.

If you are creating this Network Neighborhood yourself, select the *BreakingPoint Switching* Neighborhood from the list and then click *Save As* (Figure 3.1). Save the new Network Neighborhood as *DDoS Switching*.



User: Everybody

Last Edited: Any Time

Network Type: Any

Max Returned: 15

VLAN Ethertype: 0x8100

Domains
default

Ethernet Address	Network IP Address	VLAN I
02:1A:C5:01:00:00	1.0.0.0 / 8	

VLAN Tagging: No VLAN

☐ Use NAT

Network IP Address: 1.0.0.0

Netmask: 8

Gateway IP Address: 1.0.0.1

Type: Virtual Router

Router IP Address: 1.1.0.1

Ethernet Address: 02:1A:C5:01:00:00

☑ Use Address Range

Minimum IP Address: 1.1.0.1

Maximum IP Address:

Figure 3.1: Network Neighborhood

Next, under both the *Interface 1* and *Interface 2* tabs, select the "New Domain" button and name the new Domain "DDoS" (Figure 3.2). Interface 1 will be used as the source of the DDoS attack traffic and Interface 2 will be used as the target, or destination.

Configure the DDoS Domain on Interface 1 with both a large network as well as a large Available Address Range (Figure 3.3). This Available Address Range should reflect the size of the DDoS attacker's botnet[1] that will be simulated.

Configure Interface 2 to have the same network as Interface 1, however restrict its Available Address Range to a single or small number of addresses (Figure 3.4). This Available Address Range should reflect the simulated attack target(s). Due to the restricted Available Address Range in the DDoS Domain of Interface 2, when configuring hosts for SuperFlows, only the intended DDoS target host(s) should be attached to the "Server" interface (Figure 3.5), and all other hosts should be attached to the "Client" interface (Figure 3.6).

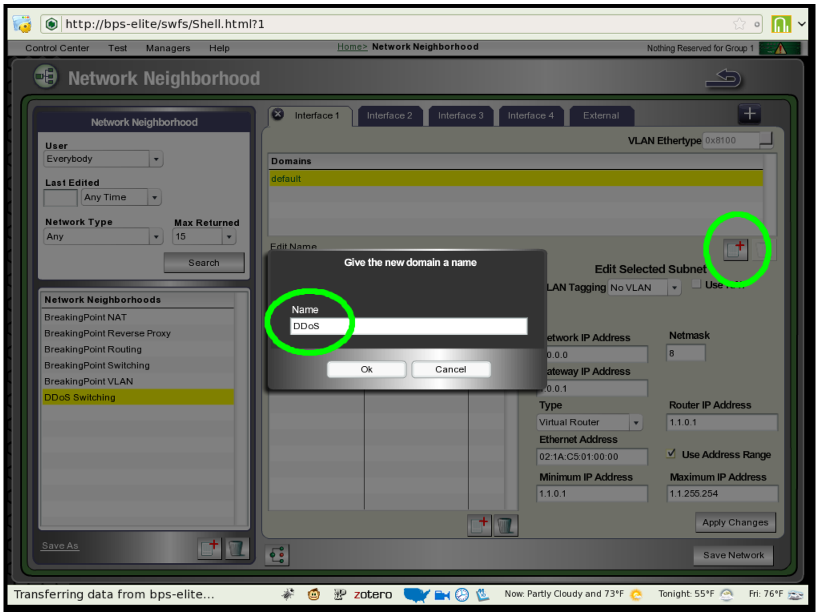
Finally, all test cases described in this paper will use the *DDoS Switching* Neighborhood, and the Interfaces of such tests must be configured to use the *DDoS* Domain (Figure 3.7). The *default* Domains were not reconfigured for use as the DDoS traffic scenario so that they may be used in some tests to simulate background or blended traffic mixes.

3.2 Duration and Throughput

The parameters relating to ramp-up, sustained, and ramp-down data rates, number of simultaneous sessions, and so forth, can be configured via the Test screen for each individual test case (Figure 3.8). Most test cases accompanying this paper use the default parameters. These are intended to be fine-tuned by the user to suit the specific DDoS data rates desired.

---

[1] A botnet is a distributed but coordinated collection of compromised networked systems that can be commanded and controlled by a central management system or network. Botnets are frequently employed to launch DDoS attacks.



http://bps-elite/swfs/Shell.html?1

Give the new domain a name

Name: DDoS

User: Everybody

Last Edited: Any Time

Network Type: Any

Max Returned: 15

VLAN Ethertype: 0x8100

Domains: default

VLAN Tagging: No VLAN

Netmask: 8

Type: Virtual Router

Router IP Address: 1.1.0.1

Ethernet Address: 02:1A:C5:01:00:00

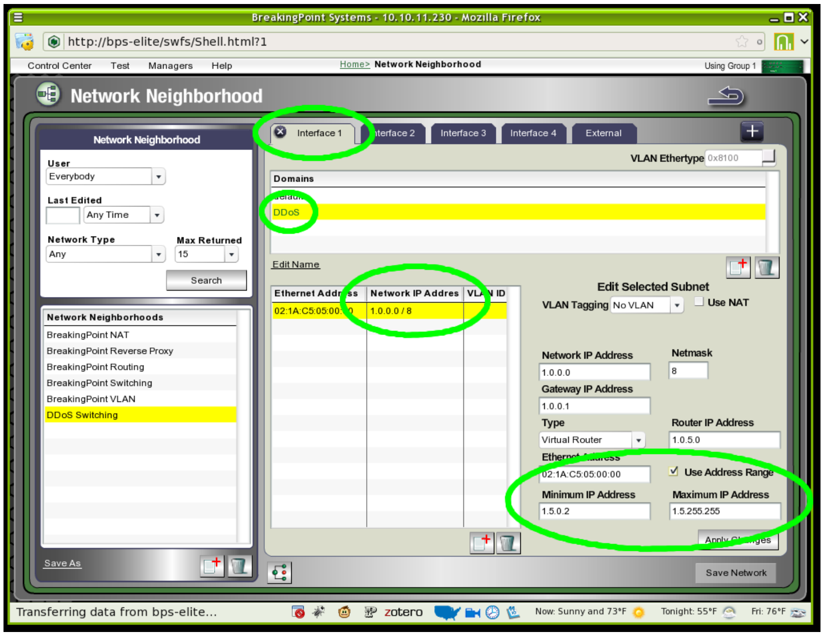
☑ Use Address Range

Minimum IP Address: 1.1.0.1

Maximum IP Address: 1.1.255.254

Transferring data from bps-elite...

Figure 3.2: Network Neighborhood: New Domain



BreakingPoint Systems - 10.10.11.230 - Mozilla Firefox

http://bps-elite/swfs/Shell.html?1

User: Everybody

Last Edited: Any Time

Network Type: Any

Max Returned: 15

VLAN Ethertype: 0x8100

Domains: default, DDoS

Ethernet Address	Network IP Address	VLAN ID
02:1A:C5:05:00:00	1.0.0.0 / 8	

VLAN Tagging: No VLAN

☐ Use NAT

Network IP Address: 1.0.0.0

Netmask: 8

Gateway IP Address: 1.0.0.1

Type: Virtual Router

Router IP Address: 1.0.5.0

Ethernet Address: 02:1A:C5:05:00:00

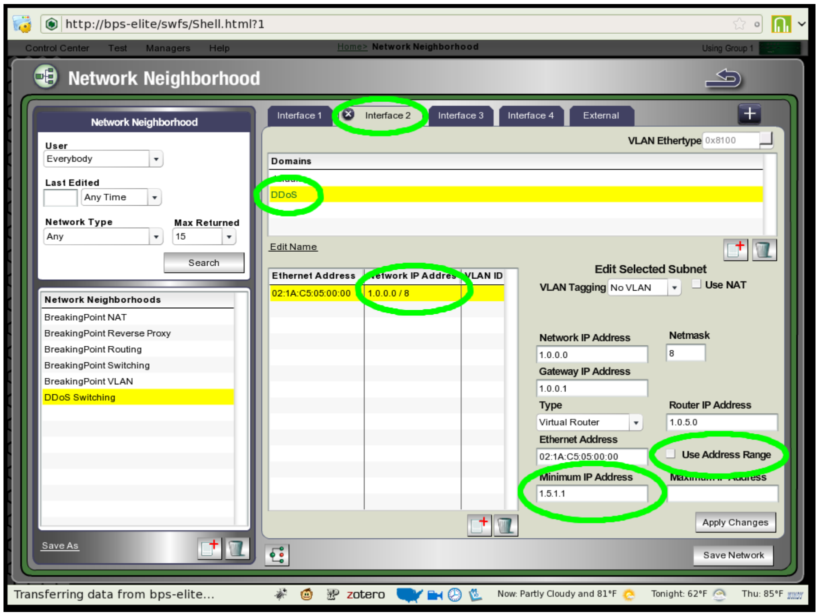
☑ Use Address Range

Minimum IP Address: 1.5.0.2

Maximum IP Address: 1.5.255.255

Transferring data from bps-elite...

Figure 3.3: Network Neighborhood: Source Addresses



Network Neighborhood

User: Everybody

Last Edited: Any Time

Network Type: Any

Max Returned: 15

Network Neighborhoods
BreakingPoint NAT
BreakingPoint Reverse Proxy
BreakingPoint Routing
BreakingPoint Switching
BreakingPoint VLAN
DDoS Switching

VLAN Ethertype: 0x8100

Domains
DDoS

Ethernet Address	Network IP Address	VLAN ID
02:1A:C5:05:00:00	1.0.0.0 / 8	

VLAN Tagging: No VLAN

☐ Use NAT

Network IP Address: 1.0.0.0

Netmask: 8

Gateway IP Address: 1.0.0.1

Type: Virtual Router

Router IP Address: 1.0.5.0

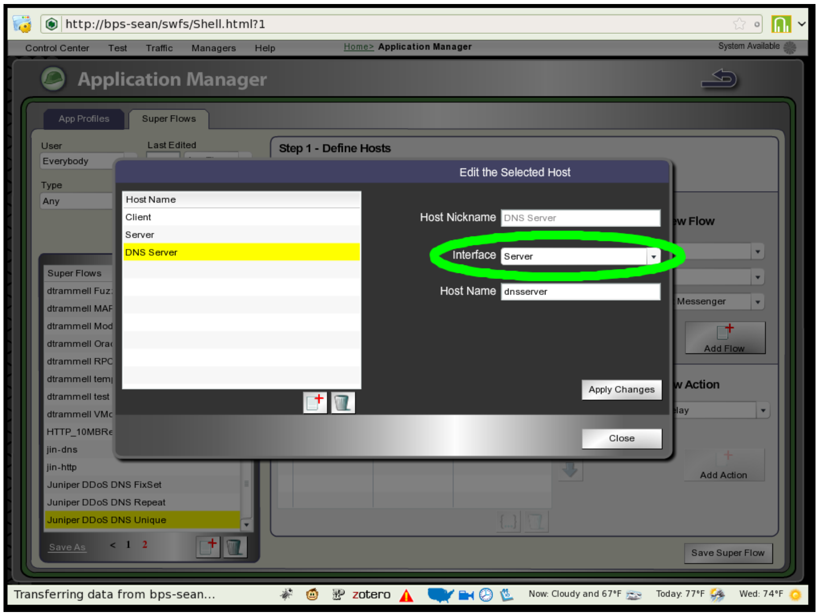
Ethernet Address: 02:1A:C5:05:00:00

☐ Use Address Range

Minimum IP Address: 1.5.1.1

Maximum IP Address:

Figure 3.4: Network Neighborhood: Target Address



User: Everybody

Type: Any

Host Name
Client
Server
DNS Server

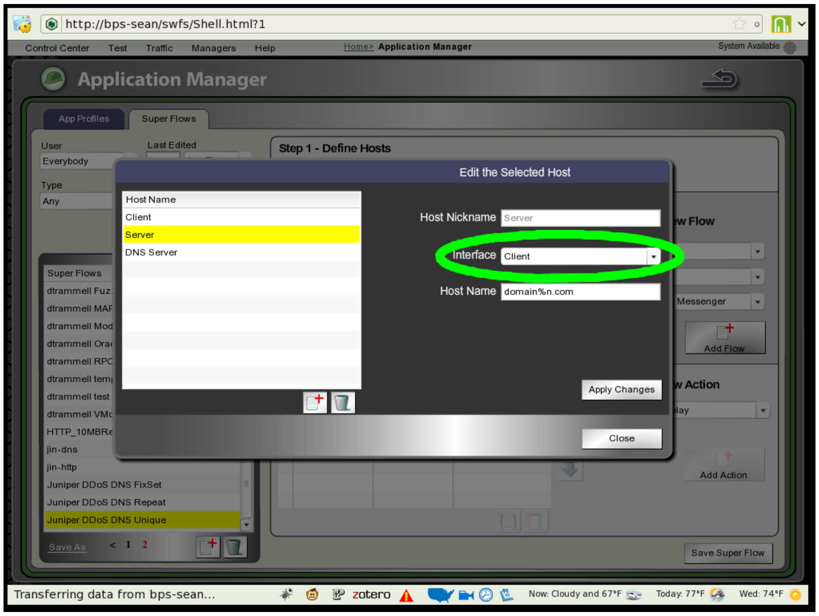
Host Nickname: DNS Server

Interface: Server

Host Name: dnsserver

Super Flows
dtrammell Fuz
dtrammell MAF
dtrammell Mod
dtrammell Orac
dtrammell RPC
dtrammell temp
dtrammell test
dtrammell VMc
HTTP_10MBRe
jin-dns
jin-http
Juniper DDoS DNS FixSet
Juniper DDoS DNS Repeat
Juniper DDoS DNS Unique

Figure 3.5: Host Configuration: Target

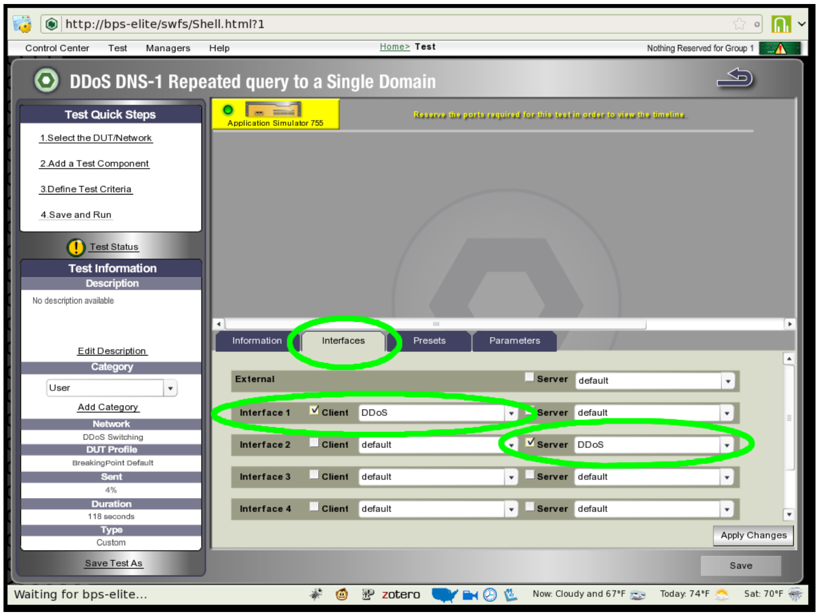


Edit the Selected Host

Host Name
Client
Server
DNS Server

Host Nickname: Server
Interface: Client
Host Name: domain%n.com

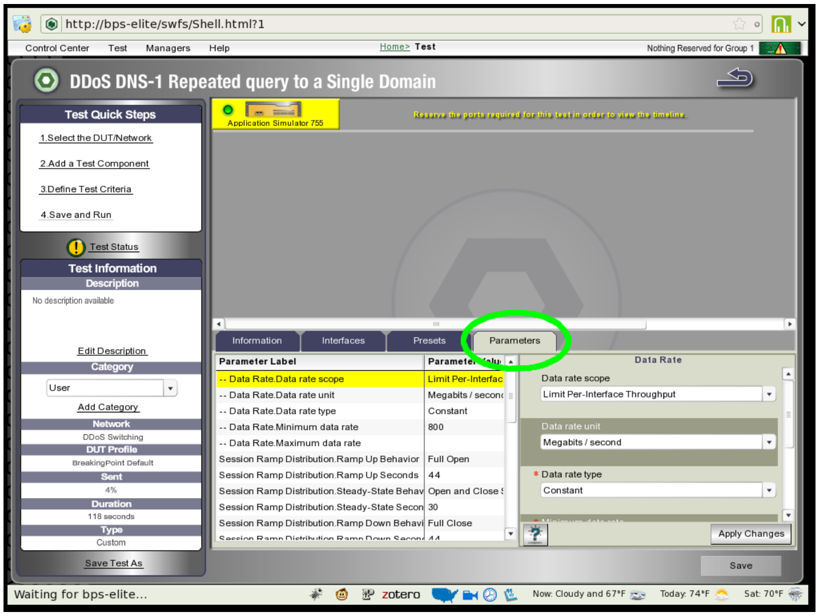
Figure 3.6: Host Configuration: Server



DDoS DNS-1 Repeated query to a Single Domain

External			☐ Server	default
Interface 1	☑ Client	DDoS	☐ Server	default
Interface 2	☐ Client	default	☑ Server	DDoS
Interface 3	☐ Client	default	☐ Server	default
Interface 4	☐ Client	default	☐ Server	default

Figure 3.7: Test: Interfaces



Parameter Label	Parameter Valu
-- Data Rate.Data rate scope	Limit Per-Interfac
-- Data Rate.Data rate unit	Megabits / seconc
-- Data Rate.Data rate type	Constant
-- Data Rate.Minimum data rate	800
-- Data Rate.Maximum data rate	
Session Ramp Distribution.Ramp Up Behavior	Full Open
Session Ramp Distribution.Ramp Up Seconds	44
Session Ramp Distribution.Steady-State Behav	Open and Close
Session Ramp Distribution.Steady-State Secon	30
Session Ramp Distribution.Ramp Down Behavi	Full Close
Session Ramp Distribution.Ramp Down Second	44

Data rate scope: Limit Per-Interface Throughput

Data rate unit: Megabits / second

* Data rate type: Constant

Figure 3.8: Rate Settings

3.3 Mixing Attack Traffic with Legitimate Traffic

There are two potentially desirable blended traffic scenarios including both attack traffic and legitimate traffic in regard to a Distributed Denial of Service. The first scenario consists of blending legitimate traffic destined for the target of the DDoS attack with the attack traffic itself. The second scenario consists of blending legitimate background network traffic between unrelated hosts on the same network with the attack traffic destined for the DDoS attack's target.

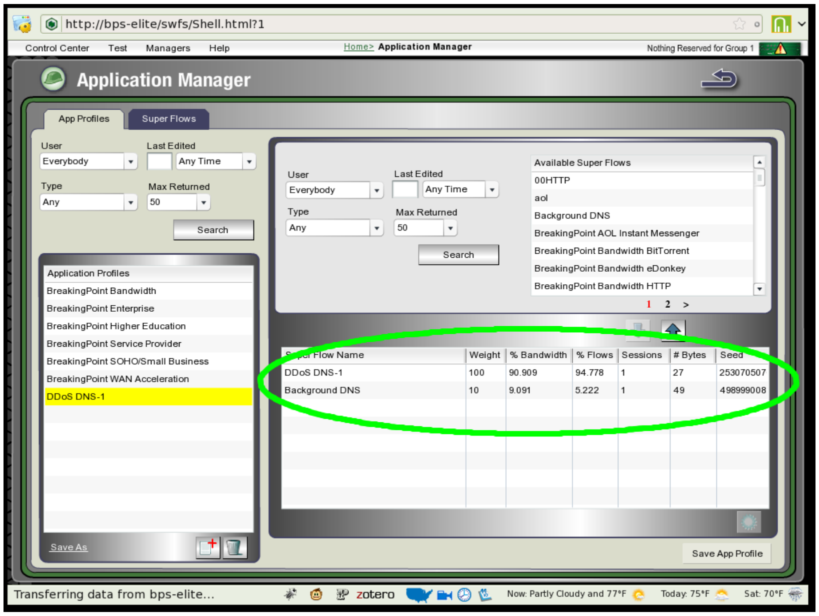
3.3.1 Same Destination

To accomplish a blended mix of legitimate and attack traffic all destined for the same host, simply include legitimate traffic SuperFlows alongside your attack traffic SuperFlow in the App Profile (Figure 3.9). Each SuperFlow included in the App Profile can then be weighted to define its presence in the resultant traffic. Being a DDoS scenario, the attack traffic should far outweigh the legitimate traffic.

Note that due to all of this traffic being destined for the same host, the legitimate traffic chosen should be appropriate for the simulated target, such as legitimate traffic of the same application or protocol as the attack traffic, or core network protocols like DNS.

The creation of a SuperFlow and accompanying App Profile to simulate legitimate background DNS traffic was accomplished through the use of the BreakingPoint TCL interface. The script used to generate this SuperFlow can be found in Appendix C. This SuperFlow can be used as described here.

A test case has been included along with this paper to demonstrate this blended traffic scenario, entitled "DDoS Blended Traffic Same Destination", which makes use of the aforementioned Background DNS SuperFlow.



http://bps-elite/swfs/Shell.html?1
Control Center
Test
Managers
Help
Home> Application Manager
Nothing Reserved for Group 1
Application Manager
App Profiles
Super Flows
Search
Search
1
2
>
Save As
Save App Profile
Transferring data from bps-elite...
zotero
Now: Partly Cloudy and 77°F
Today: 75°F
Sat: 70°F

User: Everybody
Last Edited: Any Time
Type: Any
Max Returned: 50

Application Profiles
- BreakingPoint Bandwidth
- BreakingPoint Enterprise
- BreakingPoint Higher Education
- BreakingPoint Service Provider
- BreakingPoint SOHO/Small Business
- BreakingPoint WAN Acceleration
- DDoS DNS-1

User: Everybody
Last Edited: Any Time
Type: Any
Max Returned: 50

Available Super Flows
- 00HTTP
- aol
- Background DNS
- BreakingPoint AOL Instant Messenger
- BreakingPoint Bandwidth BitTorrent
- BreakingPoint Bandwidth eDonkey
- BreakingPoint Bandwidth HTTP

Super Flow Name	Weight	% Bandwidth	% Flows	Sessions	# Bytes	Seed
DDoS DNS-1	100	90.909	94.778	1	27	253070507
Background DNS	10	9.091	5.222	1	49	498999008

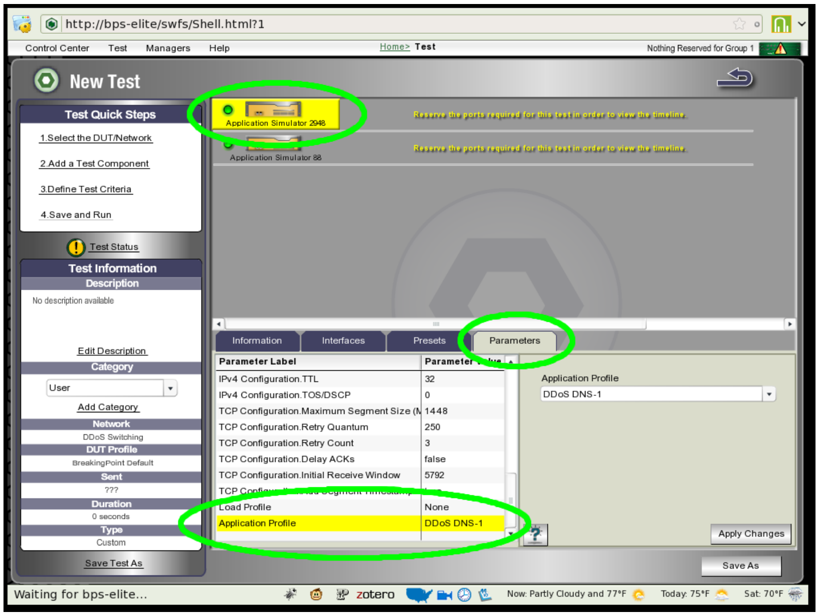
Figure 3.9: Same Destination Blended Traffic App Profile

3.3.2 Multi-Destination

To accomplish generating background traffic that is not destined for the single or small number of network addresses provided by the pool of available addresses in the *DDoS Switching* Network Neighborhood's *DDoS* Domain, the *default* domain which has no such restrictions must be used with a second AppSim component in the test.

When creating a test, in addition to the first AppSim component which is set to your attack traffic's Application Profile (Figure 3.10) and the *DDoS* Domain, also include a second AppSim component. This second AppSim component should have its Interfaces set to the *default* Domains (Figure 3.11) and its App Profile set to the desired App Profile, such as the BreakingPoint Enterprise profile, which will provide the legitimate background traffic. The AppSim Component's "Data Rate" parameters will need to be adjusted so that the interfaces that they are attached to do not become oversubscribed (Figure 3.12).

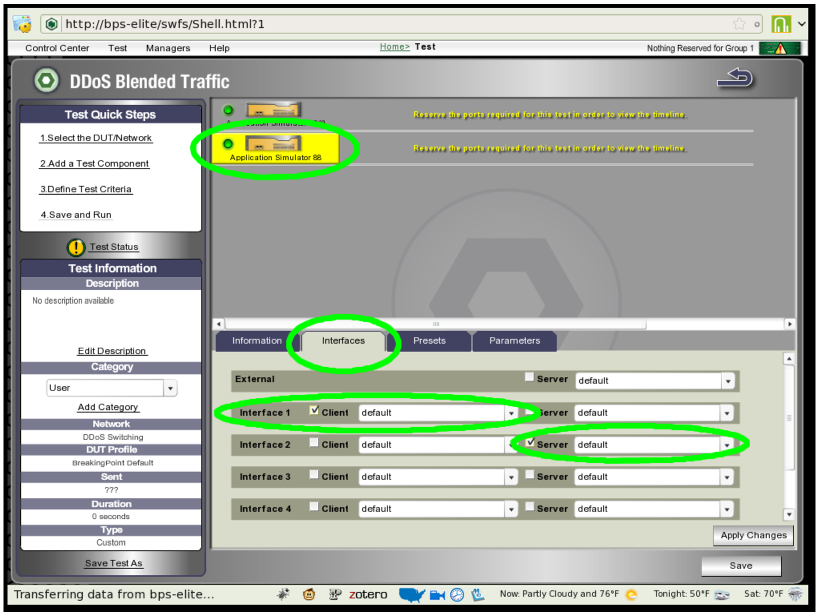
A test case has been included along with this paper to demonstrate this blended traffic scenario, entitled "DDoS Blended Traffic Multi-Destination", which makes use of the BreakingPoint Enterprise traffic profile.



Parameter Label	Parameter Value
IPv4 Configuration.TTL	32
IPv4 Configuration.TOS/DSCP	0
TCP Configuration.Maximum Segment Size (M	1448
TCP Configuration.Retry Quantum	250
TCP Configuration.Retry Count	3
TCP Configuration.Delay ACKs	false
TCP Configuration.Initial Receive Window	5792
TCP Configu[illegible]	[illegible]
Load Profile	None
Application Profile	DDoS DNS-1

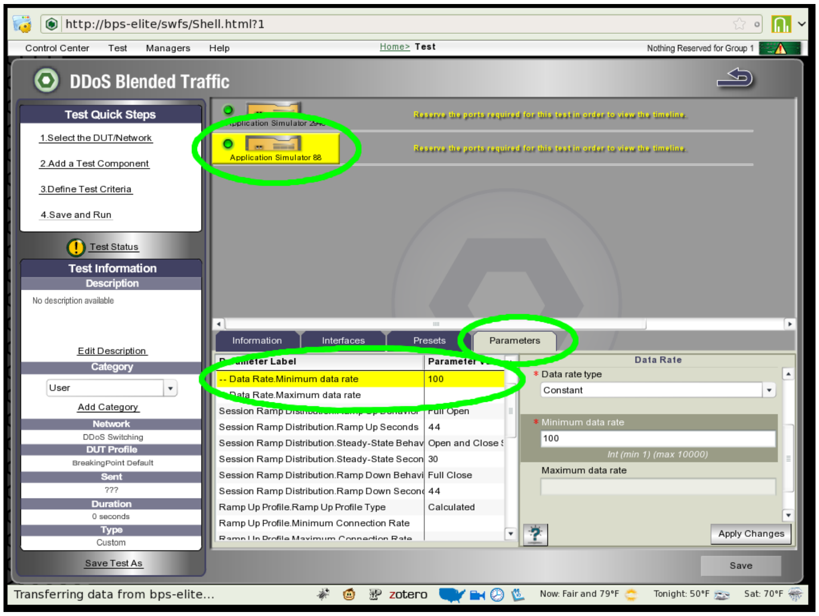
Application Profile: DDoS DNS-1

Figure 3.10: Test Application Profile Setting



External ☐ Server default
Interface 1 ☑ Client default ☐ Server default
Interface 2 ☐ Client default ☑ Server default
Interface 3 ☐ Client default ☐ Server default
Interface 4 ☐ Client default ☐ Server default

Figure 3.11: AppSim Component Interfaces



Reserve the ports required for this test in order to view the timeline.

Reserve the ports required for this test in order to view the timeline.

No description available

Category: User

Network	DDoS Switching
DUT Profile	BreakingPoint Default
Sent	???
Duration	0 seconds
Type	Custom

Parameter Label	Parameter V
-- Data Rate.Minimum data rate	100
Data Rate.Maximum data rate	
Session Ramp Distribution.Ramp Up Behavior	Full Open
Session Ramp Distribution.Ramp Up Seconds	44
Session Ramp Distribution.Steady-State Behav	Open and Close
Session Ramp Distribution.Steady-State Secon	30
Session Ramp Distribution.Ramp Down Behavi	Full Close
Session Ramp Distribution.Ramp Down Second	44
Ramp Up Profile.Ramp Up Profile Type	Calculated
Ramp Up Profile.Minimum Connection Rate	
Ramp Up Profile.Maximum Connection Rate	

* Data rate type: Constant

* Minimum data rate: 100

Int (min 1) (max 10000)

Maximum data rate:

Transferring data from bps-elite...

Figure 3.12: Data Rate Parameters

Chapter 4

Test Cases

With all attack scenarios discussed herein, default configurations and settings are used for their associated test cases unless otherwise noted. It is assumed in most cases that since these are Denial of Service scenarios that the interest is in the request and or flood traffic while the responses are rather irrelevant, or expected to be missing due to the target of the attack being too overwhelmed to actually respond. Except in specific test cases where the responses are relevant, the server traffic has usually been omitted.

4.1 TCP

TCP-1 SYN Flood

This test case consists of a Session Sender test component. All parameters are default with the following exceptions:

Test Parameters

1. The *Destination Port . Port distribution type* is set to "Constant"
2. The *Destination Port . Minimum port number* is set to "80"
3. The *Session Ramp Distribution . Ramp Up Behavior* is set to "SYN Only"
4. The *Session Ramp Distribution . Steady State Behavior* is set to "Open and Close with Reset Response"
5. The *Session Ramp Distribution . Ramp Down Behavior* is set to "Reset"
6. The *TCP Session Duration (segments)* is set to 0
7. The *Application Profile* is set to "DDoS TCP-1"

These parameters will produce a flood of "half-open" TCP connections (SYN packets only) directed at port 80 on the target(s).

The creation of this test case was accomplished through the use of the BreakingPoint TCL interface. The script used to generate this test can be found in Appendix D.

4.2 DNS

DNS-1 Query for a Single Domain Name

This test case consists of an AppSim component configured to use an App Profile which contains a single SuperFlow containing three hosts; the default *Client* and *Server* hosts, and a new third host called *DNS Server*. The *Server* host is configured to a static hostname, "example.com" and bound to the "Client" interface. This is the host that will provide the target hostname for the DNS Query. The *DNS Server* host is configured to a static hostname, "dns.example.com", and bound to the "Server" interface; this host will be the target of the DDoS attack traffic. The DNS AppSim is used between the *Client* and *DNS Server* hosts to query for the *Server* host. All parameters are default with the following exceptions:

Hosts Settings

1. Server
   (a) *Interface* is set to "Client"
   (b) *Host Nickname* is set to "example.com"
2. DNS Server
   (a) *Interface* is set to "Server"
   (b) *Host Nickname* is set to "dns.example.com"

Action Settings

1. Client Query Action
   (a) *Transaction Flag* is set to "StartEnd"
   (b) *Host* is set to "Server"

Test Parameters

1. *Application Profile* is set to "DDoS DNS-1"

DNS-2 Queries to a Fixed Set of Domain Names

This test case consists of an AppSim component configured to use an App Profile which contains a single SuperFlow containing the two hosts, *Client* and *DNS Server*, as well as 50 randomly named hosts in the form of "<randomstring>" with hostnames set to "<randomstring>.com". The DNS AppSim is used between the *Client* and *DNS Server* hosts to query for the randomized hosts. All parameters are default with the following exceptions:

Hosts Settings

1. DNS Server
   (a) *Interface* is set to "Server"
   (b) *Host Nickname* is set to "dns.example.com"
2. 50 Randomized Servers

    (a) *Hostname* is set to "<randomstring>"
    (b) *Interface* is set to "Client"
    (c) *Host Nickname* is set to "<randomstring>.com"

Action Settings

1. 50 Client Query Actions (one per randomized server)
    (a) *Transaction Flag* is set to "StartEnd"
    (b) *Host* is set to host "<randomstring>"

Test Parameters

1. *Application Profile* is set to "DDoS DNS-2"

The creation of this test case was accomplished through the use of the BreakingPoint TCL interface. The script used to generate this test can be found in Appendix E.

DNS-3 Queries to Random Domain Names

This test case consists of an AppSim component configured to use an App Profile which contains a single SuperFlow containing three hosts: the default *Client* and *Server* hosts, and a new third host called *DNS Server*. The *Server* host is configured to a randomized hostname of the form "domain%n.com" where the %n token will be interpolated with a unique number, and the host is set to the "Client" interface. This host configuration provides the random domain names that the DNS Query targets. The DNS AppSim is used between the *Client* and *DNS Server* hosts to query for the *Server* host. All parameters are default with the following exceptions:

Hosts Settings

1. Server
    (a) *Interface* is set to "Client"
    (b) *Host Nickname* is set to "domain%n.com"
2. DNS Server
    (a) *Interface* is set to "Server"
    (b) *Host Nickname* is set to "dns.example.com"

Action Settings

1. Client Query Action
    (a) *Transaction Flag* is set to "StartEnd"
    (b) *Host* is set to "Server"

Test Parameters

1. *Application Profile* is set to "DDoS DNS-3"

Note, while the above domain name is effectively randomized, not every request on the wire will be unique. The way the BreakingPoint system's Network Processor streams data is by requesting that an AppSim generate traffic, then it plays that traffic to the wire repeatedly while a new batch is generated. Once the new batch is generated it is swapped in to replace the old batch. As such, traffic with identical randomized domain name queries will appear on the wire.

4.3 HTTP

4.3.1 Resource-intensive Target URLs

Some DDoS attacks are accomplished not by traffic amplification but by causing a large amount of CPU or memory resource consumption by the target by making specific, carefully chosen requests. Some such requests involve search or database queries, authentication attempts that prompt use external back-end services such as a RADIUS or Directory server, or requests that return an overly large amount of data relative to the request that must be either dynamically generated or processed in order to be returned.

HTTP-1 Search Query

This test case consists of an AppSim component configured to use an App Profile which contains a single SuperFlow containing the two default hosts, *Client* and *Server*. The HTTP AppSim is used to simulate a single GET request from *Client* to *Server*. The usual HTTP 200 response has been omitted to simulate an overwhelmed target not being able to respond. The GET request simulates a request to a search function with an overly generic search term. All parameters are default with the following exceptions:

Action Settings

1. Client GET Action
    (a) *Transaction Flag* is set to "Start"
    (b) *Request path* is set to "/search?q=s"

Test Parameters

1. *Application Profile* is set to "DDoS HTTP-1"

HTTP-2 Authenticated Request

This test case consists of an AppSim component configured to use an App Profile which contains a single SuperFlow containing the two default hosts, *Client* and *Server*. The HTTP AppSim is used to simulate a single authenticated GET request from *Client* to *Server*. The usual HTTP response, whether properly authenticated or not, has been omitted to simulate an overwhelmed target not being able to respond. The GET request simulates a request to resources that require authentication on a server that communicates with a back-end authentication authority to vet credentials. All parameters are default with the following exceptions:

Action Settings

1. Client GET (authenticated) Action
    (a) *Transaction Flag* is set to "StartEnd"

    (b) *HTTP Authentication scheme to use* is set to “Digest Authentication”

Test Parameters

1. *Application Profile* is set to “DDoS HTTP-2”

HTTP-3 Large Response

This test case consists of an AppSim component configured to use an App Profile which contains a single SuperFlow containing the two default hosts, *Client* and *Server*. The HTTP AppSim is used to simulate a single GET request from *Client* to *Server*. The GET request simulates a request to resources that return relatively large amounts of data in comparison to the request. This data is potentially dynamically generated at the time of request, or is static but extremely large, causing a disproportional amount of processing expenditure by the server in comparison to the client. All parameters are default with the following exceptions:

Action Settings

1. Client GET Action
    (a) *Transaction Flag* is set to “Start”
2. Server Response 200 (OK) Action
    (a) *Transaction Flag* is set to “End”
    (b) *Random response min length* is set to “50000000” (50 Megabytes)
    (c) *Random response max length* is set to “100000000” (100 Megabytes)

Test Parameters

1. *Application Profile* is set to “DDoS HTTP-3”

4.4 SMTP

SMTP-1 Backscatter Flood

An SMTP backscatter flood is usually the resultant side-effect of a SPAM mailer sending an email campaign with the source email address of the messages spoofed to appear as originating from an innocent third party, commonly referred to as a “Joe Job” attack[2]. A percentage of the messages in the SPAM campaign will be addressed to nonexistent addresses which causes the receiving mail servers to respond to the messages with Delivery Service Notifications (DSNs) indicating the nonexistent address. This flood of response DSN messages directed back at the innocent third party’s mail server is considered a specific type of DDoS attack when the volume becomes large enough.

This test consists of an AppSim component configured to use an App Profile which contains a single SuperFlow containing the two default hosts, *Client* and *Server*, representing a sending mail server and a receiving mail server, respectively. The SMTP AppSim is used to send a DSN-style email message from *Client* to *Server*. All parameters are default with the following exceptions:

Action Settings

1. Client Send HELO Action
    (a) *Transaction Flag* is set to "Start"
2. Client Send RCPT Action
    (a) *To Address* is set to "joe@example.com"
3. Client Send email (raw) Action
    (a) *Protocol To Address* is set to "joe@example.com"
    (b) *Envelope To Address* is set to "joe@example.com"
    (c) *Subject* is set to "Failure Notice"
    (d) *Raw message* is set to "dsn-email.txt"
4. Server 221 closing Action
    (a) *Transaction Flag* is set to "End"

Test Parameters

1. *Application Profile* is set to "DDoS SMTP-1"

4.5 VoIP

VoIP-1 Low-Rate Call Flood

Not every DoS is the result of massive resources overwhelming the processing or bandwidth capacity of its target. In some instances, such as a low-rate VoIP call flood, the target of the attack may not even be the technology itself. In this scenario, the target is a human, having to cope with a constantly ringing telephone. While a legitimate call may in fact get routed to the target telephony device along-side the malicious calls, in many cases a human will actually DoS themselves by simply taking the phone off-hook. These types of floods frequently fly below the radar as well due to their relatively low rate and the fact that they may not actually overwhelm the target device's bandwidth or resources.

This test case consists of an AppSim component configured to use an App Profile containing a single SuperFlow containing the two default hosts, *Client* and *Server*, representing two SIP devices acting as a UAC and UAS, respectively. The SIP AppSim is used to simulate a call setup procedure from *Client* to *Server*. Only the SIP AppSim is employed to create a flood of call-setup sequences which simulates causing the telephony device associated with the target user, via a specific telephone number, to ring. The SuperFlow does not setup a VoIP media channel or convey any audio; the SIP UAC immediately hangs-up the call upon answer by the UAS. This test case makes use of the timing parameters in the test to properly simulate the low-rate nature of this DDoS. All parameters are default with the following exceptions:

Flow Settings

1. *Caller Phone Number* is set to "(512)867-5309"

Test Parameters

1. *Session Configuration . Maximum Simultaneous Sessions* is set to "1"
2. *Session Configuration . Maximum Sessions Per Second* is set to "1"
3. *Application Profile* is set to "DDoS VoIP-1"

Appendix A

Introduction to Scripting with the BreakingPoint TCL Interface

BreakingPoint provides a custom TCL shell that provides the API for controlling your devices. This shell is used as the interpreter for TCL scripts, as well as an interactive shell. The interactive mode is a great way to explore and learn the API. Customers can also read chapter 14 of the BreakingPoint UserGuide, which can be downloaded from the BreakingPoint Systems Documentation page on StrikeCenter (authentication required). The examples in this section are intended for use with appliances running the 1.2.3 BreakingPoint software release.

BreakingPoint Interactive TCL Shell

To get started, download the BreakingPoint TCL shell from the system start page. When you run it from the command line (or from a script using the -i flag), you'll get a familiar-feeling '% ' prompt. To connect to your BreakingPoint box, issue something similar to the following:

```
% set host bpsbox
% set username elmo
% set password elmo
% set bps [bps::connect $host $username $password]
```

This sequence is so common that I have wrapped it inside a script I call harness.tcl, which parses its command-line arguments, makes the initial connection, and then drops you into the interactive shell.

Once you're connected, you can take a quick look at the commands and variables available for use by issuing:

```
% lsort -increasing [info commands]
% lsort -increasing [info vars]
```

The listed commands will show usage information when you issue them without arguments, so you can just experiment and see what each of the commands accept. Many variables are defined, but the variable we care about at this point is the $bps variable, which represents the connection we have established to the BreakingPoint appliance.

The $bps object gives commands to control many aspects of the device: tests, components, application profiles, attack series, load profiles, DUTs, reports, and device management.

Running a Security Test with TCL

So let's dive in and configure a simple test with a single Security component. The basic steps are:

1. Connect to the Chassis and reserve ports
2. Name the test
3. Add a security component
4. Set the test interfaces for the component
5. Save and run the test
6. Review the results

The code for this is pretty simple:

```
% set ch [$bps getChassis]
% $ch unreservePort 1 0
% $ch unreservePort 1 1
% $ch unreservePort 1 2
% $ch unreservePort 1 3
% $ch reservePort 1 0
% $ch reservePort 1 1
% set componentname security
% $bps configure -name "TCL Demo Test $componentname"
% set component [$bps createComponent $componentname tcldemo_$
{componentname}]
% $component setDomain server 1 default
% $component setDomain client 2 default
% $bps save -force
% $bps run
% set result [$component result]
% foreach p [$result values] { set v [$result get $p]; puts "$p == $v" }
% $bps exportReport -file TCL_Demo_Test_${componentname}_Result.pdf -
format pdf
```

This test will use the default DUT profile of 'BreakingPoint Default,' and the default Network Neighborhood of 'BreakingPoint Switching.' If your test environment requires something different like routing, you'll need to issue the appropriate *setDut* and *setNeighborhood* commands before adding the component. The two *setDomain* commands set the server and client sides of the security component, and specify the default Ethernet and IP addressing domains. After running the test, we display the summary results of the test. To get detail about what strikes were blocked, you need to view the report generated from the *exportReport* call.

The nice thing about this code is that it's general enough to run any component. You can set componentname in the first line to appsim, bitblaster, recreate, routingrobot, security, sessionsender, or stackscrambler. The list of available components can be found by issuing '$bps createComponent false false'. This will show both the canned (BreakingPoint defined) as well as custom (customer defined) components.

The automation afforded by using TCL has been invaluable in reducing the time it takes the BreakingPoint Labs team to test and deploy StrikePacks. I know it will benefit anyone that takes the time to learn it. Over the coming weeks, I'll be sharing hints and code that will help BreakingPoint customers leverage automation in their testing efforts. The next installment will focus on utilizing the Security component to test IPS signatures. In the meantime, please download the code archive for this post and get started with TCL!

The above content, and more about BreakingPoint system automation using TCL, can be found tagged as “tcl” at the BreakingPoint Systems Blog[3].

Appendix B

TCL Libraray (tcl.lib)

```
proc randomString {length {chars ”
    ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz”}} {
   set range [expr {[string length $chars]−1}]

   set txt ””
   for {set i 0} {$i < $length} {incr i} {
      set pos [expr {int(rand()∗$range)}]
      append txt [string range $chars $pos $pos]
   }
   return $txt
}
```

Appendix C

TCL Script: 3.3.1

```
#!/usr/local/bin/bpsh
source lib.tcl

# ARG handling
if { [llength $argv] >= 3 } {
  set host [lindex $argv 1]
  set username [lindex $argv 2]
  set password [lindex $argv 3]

  set clientinterface 1
  set serverinterface 2
} else {
  set scriptname [lindex $argv 0]
  puts "Usage:␣$scriptname␣<BPS␣ipaddr>␣<username>␣<password>"
  exit -1
}

# make sure the device isn't in use by another user
if { [catch {set bps [bps::connect $host $username $password
    -checkversion false]}]} {
  puts "\[!]␣Error␣connecting:␣$errorInfo"
  exit -1
}
puts "\[*]␣Connected..."

set name "Background␣DNS"
puts "\[*]␣Creating␣${name}␣SuperFlow"
set sf [$bps createSuperflow -name $name]
set count 50

puts "\[*]␣Creating␣DNS␣Server␣host"
$sf removeHost "Server"
$sf addHost "DNS␣Server" target "dns.example.com"

puts "\[*]␣Creating␣${count}␣random␣hosts"
for {set i 0} { $i < $count} {incr i} {
  set length [expr int(rand()*6)+2]
  set hostname [randomString $length]
  set domainname "${hostname}.com"
  puts "\[+]␣$sf␣addHost␣$hostname␣origin␣$domainname"
  $sf addHost $hostname origin $domainname
}
$sf save -force

puts "\[*]␣Creating␣${count}␣DNS␣Queries␣between␣Client␣and␣DNS␣Server"
puts "\[+]␣Adding␣flow␣between␣Client␣and␣DNS␣Server"
set flowid [$sf addFlow dnsadv Client "DNS␣Server"]

dict for {hostid host} [$sf getHosts] {
  if {"Client"!=$hostid && "DNS␣Server"!=$hostid} {
    puts "\[+]␣Adding␣DNS␣request␣for␣$hostid"
    $sf addAction $flowid client dns_query -domainname $hostid
  }
}

puts "\[*]␣Saving␣SuperFlow"
$sf save -force

puts "\[*]␣Creating␣App␣Profile␣${name}"
set ap [$bps createAppProfile $name]
#$ap addSuperflow "${sf}" 100
$ap addSuperflow $name 100
$ap configure -name $name

puts "\[*]␣Saving␣App␣Profile"
$ap save -force

puts "\[*]␣Done!"
```

Appendix D

TCL Script: TCP-1

```
#!/usr/local/bin/bpsh

# ARG handling
if { [llength $argv] >= 3 } {
  set host [lindex $argv 1]
  set username [lindex $argv 2]
  set password [lindex $argv 3]

  set clientport 0
  set clientinterface 1
  set serverport 1
  set serverinterface 2
} else {
  set scriptname [lindex $argv 0]
  puts "Usage:␣$scriptname␣<BPS␣ipaddr>␣<username>␣<password>"
  exit -1
}

# make sure the device isn't in use by another user
if { [catch {set bps [bps::connect $host $username $password
    -checkversion false]}]} {
  puts "\[!]␣Error␣connecting:␣$errorInfo"
  exit -1
}
puts "\[*]␣Connected..."

set name "DDoS␣TCP-1␣SYN␣Flood"
puts "\[*]␣Creating␣test␣${name}"
$bps configure -name $name

puts "\[*]␣Adding␣SessionSender␣Component"
set ss [$bps createComponent sessionsender {DDoS TCP-1}]
$ss configure -dstPortDist.type constant -dstPortDist.min 80
    -packetsPerSession 0 -rateDist.min 400 -rateDist.max 1000
    -rateDist.type random -sessions.maxPerSecond 500000 -sessions.max
    5000000 -rampDist.upBehavior syn -rampDist.downBehavior rst
    -rampDist.steadyBehavior cycle+rst -rampDist.steady 120
```

```
puts "\[*]␣Setting␣Network␣Neighborhood␣to␣DDoS␣Switching”
$bps setNeighborhood "DDoS␣Switching”
puts "\[*]␣Setting␣interface␣Domains␣to␣DDoS”
$ss setDomain client $clientinterface "DDoS”
$ss setDomain server $serverinterface "DDoS”

puts "\[*]␣Saving␣Test”
$bps save −force

puts "\[*]␣Done!”
```

Appendix E

TCL Script: DNS-2

```
#!/usr/bin/env bpsh
source lib.tcl

# ARG handling
if { [llength $argv] >= 3 } {
  set host [lindex $argv 1]
  set username [lindex $argv 2]
  set password [lindex $argv 3]

  set clientport 0
  set clientinterface 1
  set serverport 1
  set serverinterface 2
} else {
  set scriptname [lindex $argv 0]
  puts "Usage:␣$scriptname␣<BPS␣ipaddr>␣<username>␣<password>"
  exit -1
}

# make sure the device isn't in use by another user
if { [catch {set bps [bps::connect $host $username $password
    -checkversion false]}]} {
  puts "\[!]␣Error␣connecting:␣$errorInfo"
  exit -1
}

puts "\[*]␣Connected..."

set name "DDoS␣DNS-2"
puts "\[*]␣Creating␣SuperFlow␣${name}"
set sf [$bps createSuperflow -name $name]
set count 50

puts "\[*]␣Creating␣DNS␣Server␣host"
$sf removeHost "Server"
$sf addHost "DNS␣Server" target "dns.example.com"
```

```
puts "\[*]␣Creating␣${count}␣random␣hosts"
for {set i 0} { $i < $count} {incr i} {
  set length [expr int(rand()*6)+2]
  set hostname [randomString $length]
  set domainname "${hostname}.com"
  puts "\[+]␣$sf␣addHost␣$hostname␣origin␣$domainname"
  $sf addHost $hostname origin $domainname
}
$sf save -force

puts "\[*]␣Creating␣${count}␣DNS␣Queries␣between␣Client␣and␣Server"
puts "\[+]␣Adding␣flow␣between␣Client␣and␣DNS␣Server"
set flowid [$sf addFlow dnsadv Client "DNS␣Server"]

dict for {hostid host} [$sf getHosts] {
  if {"Client"!=$hostid && "DNS␣Server"!=$hostid} {
    puts "\[+]␣Adding␣DNS␣request␣for␣$hostid"
    $sf addAction $flowid client dns_query -domainname $hostid
        -transflag startend
  }
}

$sf save -force

puts "\[*]␣Creating␣App␣Profile␣${name}"
set ap [$bps createAppProfile $name]
$ap addSuperflow $name 100
$ap configure -name $name
$ap save -force

puts "\[*]␣Creating␣AppSim␣${name}"
set appsim [$bps createComponent appsim $name]
$appsim configure -profile [$ap cget -name]

puts "\[*]␣Setting␣Network␣Neighborhood␣to␣DDoS␣Switching"
$bps setNeighborhood "DDoS␣Switching"
puts "\[*]␣Setting␣interface␣Domains␣to␣DDoS"
$appsim setDomain client $clientinterface "DDoS"
$appsim setDomain server $serverinterface "DDoS"

set name "DDoS␣DNS-2␣Repeated␣Queries␣to␣a␣Fixed␣Set␣of␣Domain␣Names"
puts "\[*]␣Creating␣test␣${name}"
$bps configure -name $name

puts "\[*]␣Saving␣Test"
$bps save -force

puts "\[*]␣Done!"
```

Bibliography

[1] Wikipedia. Denial-of-service attack - distributed attack. http://en.wikipedia.org/wiki/Denial-of-service_attack#Distributed_attack, February 2009.

[2] Wikipedia. Joe job. http://en.wikipedia.org/wiki/Joe_job, January 2009.

[3] Todd Manning. Tcl tagged blog posts. http://www.breakingpointsystems.com/community/blog/tags/tcl, January 2009.