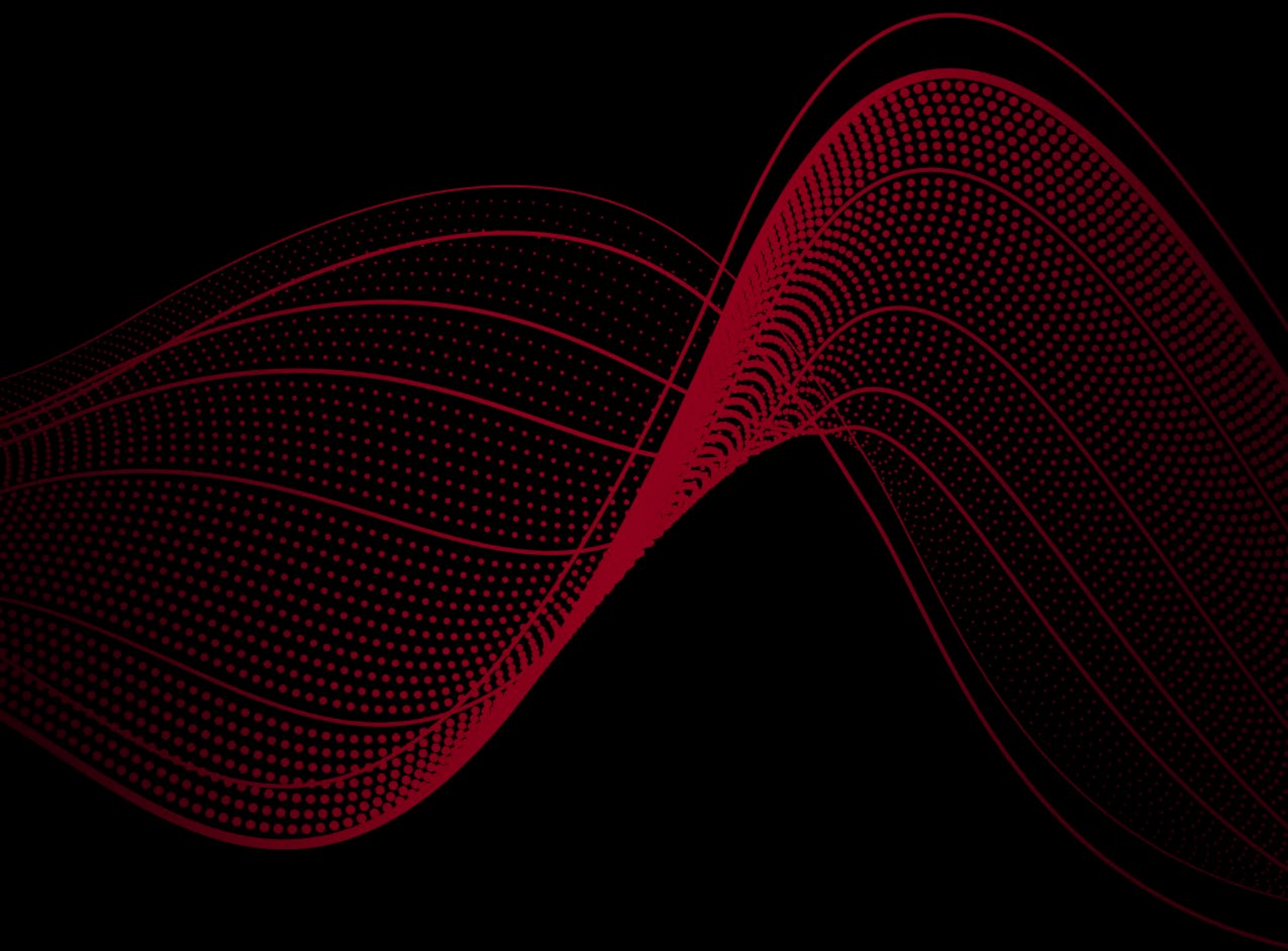


BLACK BOOK

Network Security



Your Feedback is Welcome

Our goal in the preparation of this Black Book was to create high-value, high-quality content. Your feedback is important to help guide our future books.

If you have comments regarding how we could improve the quality of this book, or suggestions for topics to be included in future Black Books, contact us at ProductMgmtBooklets@ixiacom.com.

Your feedback is greatly appreciated!

Table of Contents

How to Read this Book.....	vii
Dear Reader.....	viii
Network Security Overview.....	3
Ixia BreakingPoint	27
Test Methodologies for Known Vulnerabilities and Malware.....	29
Test Case: Measuring Security Effectiveness of Intrusion Prevention Engines	35
Test Case: Measuring Security Effectiveness of Intrusion Prevention Engine with Evasions.....	55
Test Case: Measuring the Security Effectiveness of Anti-Virus Engine	61
Test Methodologies for DoS and DDoS	81
DoS/DDoS: Methods of Attack	87
Test Case: Mitigation of UDP Flooding and TCP SYN Flooding Attacks	97
Test Case: Mitigation of Sophisticated Application Layer Attacks.....	119
Test Methodologies for IPsec VPN	145
Test Case: IPsec—Real Application Mix Forwarding Performance	159
Test Methodologies for Traffic Fuzzing and Negative Testing.....	185
Test Case: Fuzzing	185
Test Methodologies for Data Leakage or Data Loss Prevention	201
Test Case: Validating Basic Effectiveness of DLP Engines	203
Data Leakage Test: Lawful Intercept Labs.....	223
Test Methodologies for Virtual Environment Security	231
Test Case: Lateral Threat Propagation within Software Defined Data Center	233
Test Case: Performance Acceleration with Intel DPDK Support.....	249
Test Methodologies for Regenerating Production Network Traffic.....	257
Test Case: Pre-Deployment Validation with Production Application Mixes.....	259
Test Methodologies for Encrypted Traffic Inspection	283
Test Case: Measuring Encrypted Traffic Inspection Capabilities of SSL Proxy Devices	285
Appendix A: Configuring a Virtual Router.....	311

How to Read this Book

The book is structured as several standalone sections that discuss test methodologies by type. Every section starts by introducing the reader to relevant information from a technology and testing perspective.

Test case structure may follow this structure:

Overview	Provides background information specific to the test case.
Objective	Describes the goal of the test.
Setup	An illustration of the test configuration highlighting the test ports, simulated elements and other details.
Step-by-Step Instructions	Detailed configuration procedures using Ixia test equipment and applications.
Test Variables	A summary of the key test parameters that affect the test's performance and scale. These can be modified to construct other tests.
Results Analysis	Provides the background useful for test result analysis, explaining the metrics and providing examples of expected results.
Conclusions	Summarizes the result of the test.

Typographic Conventions

In this document, the following conventions are used to indicate items that are selected or typed by you:

- **Bold** items are those that you select or click on. It is also used to indicate text found on the current GUI screen.
- *Italicized* items are those that you type.

Dear Reader

Ixia's Black Books include network, application, and security test methodologies that will help you become familiar with new technologies and the key testing issues associated with them.

The Black Books are primers on technology and testing. They include test methodologies to verify device and system functionality and performance. The methodologies are universally applicable to any test equipment. Step-by-step instructions use Ixia's test platforms and applications to demonstrate the test methodology.

Our library of Black Books includes twenty-two volumes that cover key technologies and test methodologies:

- | | |
|--|--|
| Volume 1 – Network Security | Volume 12 – Network Convergence Testing |
| Volume 2 – Application Delivery | Volume 13 – Ethernet Synchronization |
| Volume 3 – QoS Validation | Volume 14 – Advanced MPLS |
| Volume 4 – Voice over IP | Volume 15 – MPLS-TP |
| Volume 5 – Video over IP | Volume 16 – Ultra Low Latency (ULL) Testing |
| Volume 6 – LTE Access | Volume 17 – Network Impairment |
| Volume 7 – LTE Evolved Packet Core | Volume 18 – Test Automation |
| Volume 8 – Carrier Ethernet | Volume 19 – 802.11ac Wi-Fi Benchmarking |
| Volume 9 – IPv6 Transition Technologies | Volume 20 – SDN/OpenFlow |
| Volume 10 – Converged Data Center | Volume 21 – Audio Video Bridging |
| Volume 11 – Converged Network Adapters | Volume 22 – Automotive Ethernet |

These Black Books are available in Ixia's online [Resources Library](#).

We are committed to helping our customers build and maintain networks that perform at the highest level, ensuring end users get the best application experience possible. We hope this Black Book series provides valuable insight into the evolution of our industry, and helps customers deploy applications and network services—in a physical, virtual, or hybrid network configurations.

Network Security

Test Methodologies

Network security is essential for homes, government organizations, and enterprises of all sizes. It is both a strategy and the provisions designed to protect network infrastructure and the data traversing it. The number and types of attacks are enormous and the devices used to defend against them are necessarily complex. This book provides an overview of network security and covers test methodologies that can be used to validate the effectiveness, accuracy, and performance of network security devices, policies, and process. By combining a realistic mix of legitimate application traffic emulation and comprehensive set of threat vectors, these methodologies are designed to help network operators and security professionals find the right balance between application performance and security in today's hyper-scale, dynamic world.

Network Security Overview

Network Security Overview

Security is a discipline concerned with protecting networks and computer systems against threats such as exploits, malware, data leakage, spam, and DoS attacks, as well as ensuring trusted access through mechanisms like IPsec or SSL. To defend against threats, and to prevent unintended data leakage, enterprises have deployed security devices of all types.

Network security devices have one or more security functions, including firewall, Intrusion Prevention (IPS), Data Leakage Prevention (DLP), and content security filtering functions (anti-spam, anti-virus, URL-filtering), SSL decryption and inspection, and VPN access. Devices that combine more than one of the technologies are commonly referred to as Next-Generation Firewalls (NGFW) or Unified Threat Management Systems (UTMS). Each type of security function above requires continuous testing to ensure that the devices are effective, accurate, productive, and up to date

Securing a network is essential for all users of the Internet. The number and types of attacks continue to grow at an alarming rate and the devices used to defend against them can be complex.

This book provides an overview of network perimeter security and covers test methodologies that can be used to assess the effectiveness, accuracy, and performance of such devices while they are inspecting legitimate and malicious traffic.

Test cases pertaining to the security of virtual network functions (VNF) and virtual network functions infrastructure (VNF*i*) are addressed in the Virtual Environment Security section of this book. Tests in this section focus on testing of virtual security appliances for the prevention of lateral threat propagation as well as optimizing performance by accelerating security testing using Intel data plane development kit (DPDK).

The Current State of Network Security

There has been an explosion of security threats and large-scale data breaches at some of the world's largest corporations, as well as smaller more indiscriminate attacks on the general Internet community.

Security solution provider Panda Labs identified 227,000 malware samples per day in the first quarter of 2016¹:

¹ <http://www.pandasecurity.com/mediacenter/news/pandalabs-study-q1/>

Network Security Overview

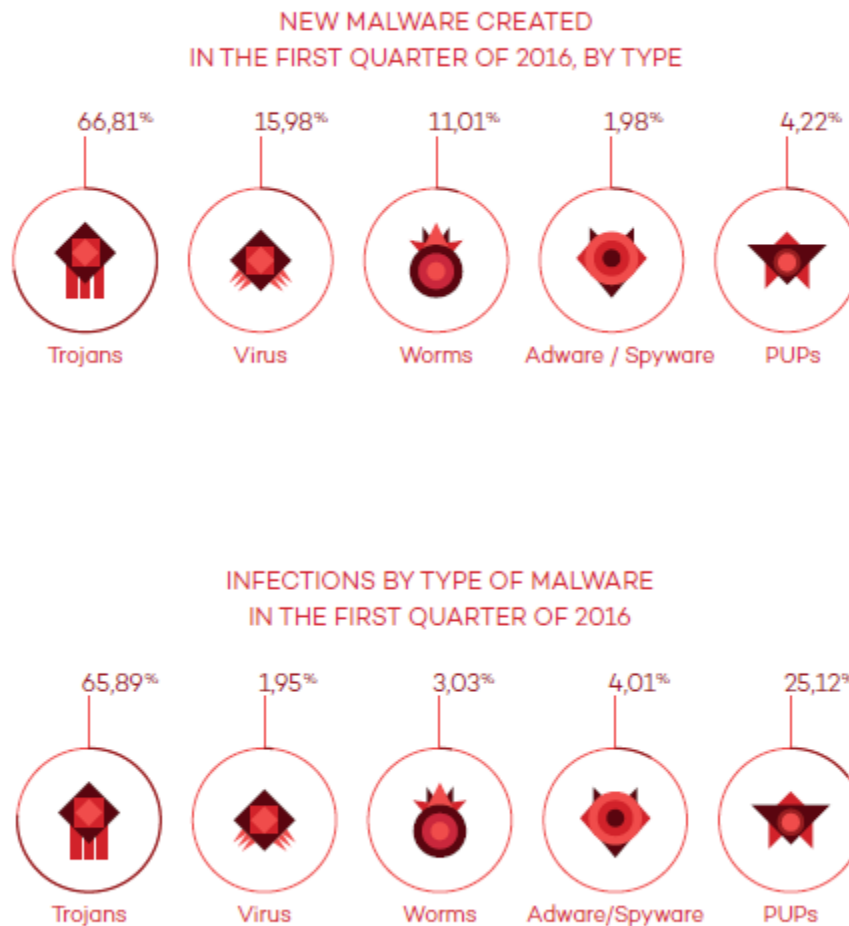


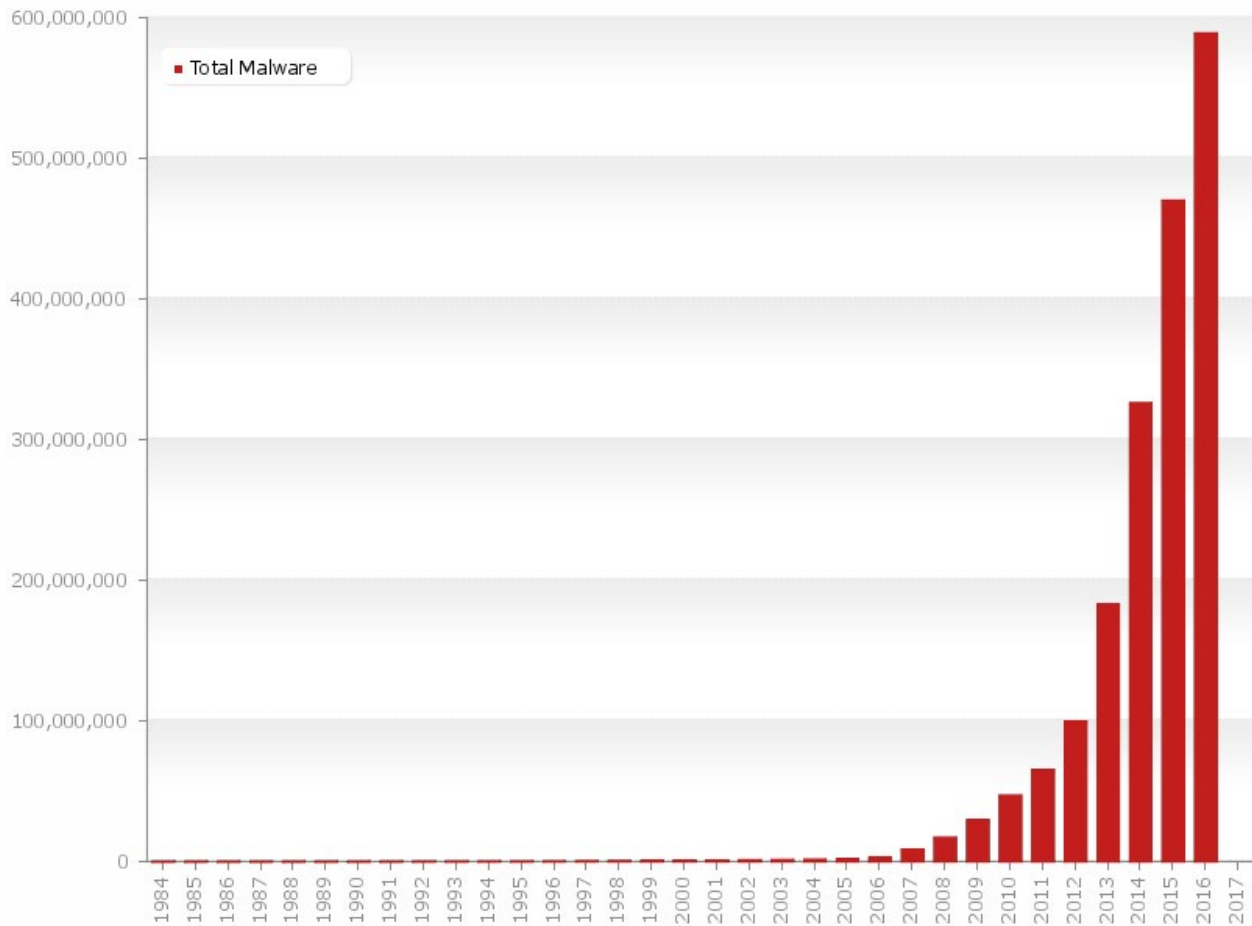
Figure 1. Breakdown of malware types created and the infection by type of Malware in first Q of 2016 (according to Panda Labs)

The purpose of most malware is to either steal data or encrypt it and demand ransom payment to unlock it. Email and web remain significant attack vectors, however social networking platforms, peer-to-peer (P2P), and mobile devices are increasingly targeted. In addition, the prevalence of Internet of things (IoT) increases the potential for cyberattacks against connected cars, homes, medical equipment, and wearables.

AV-Test, a company that provides independent comparative tests and reviews for antivirus software and anti-malware tools, has reported that there has been an exponential growth in the number of malware samples in the last 5 years². This, of course, was boosted by the rise of ransomware, enabling cyber criminals to make quick money by scamming unsuspecting Internet users.

² <https://www.av-test.org/en/statistics/malware/>

Network Security Overview



Last update: 11-28-2016 15:39

Copyright © AV-TEST GmbH, www.av-test.org

Figure 2. AV-Test report showing total malwares rapidly increasing over the past several years.

During the last few years, the cumulative number of vulnerabilities has also followed similar trends. What's surprising is that even after steady education on cyber-security and patch management, hackers have still been able to compromise systems using old known vulnerabilities at a steady rate. The below figure from Verizon's 2016 Data Breach Investigations Report³ shows the count of CVEs exploited in 2015 by publication date, confirming that hackers are still exploiting unpatched systems with known vulnerabilities dating back as far as 1998.

³ <http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/>

Network Security Overview

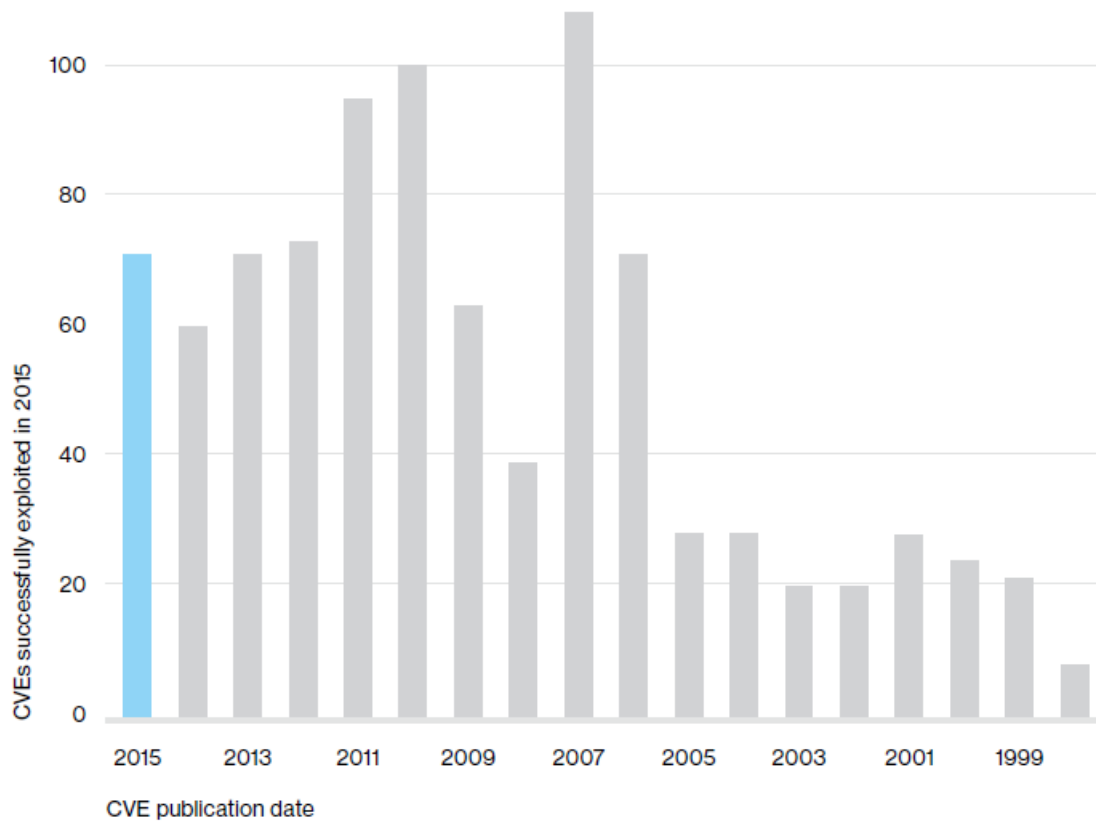


Figure 3. Count of CVE's exploited in 2015 by CVE publication date.

The number of discovered vulnerabilities in applications is far greater than the number discovered in operating systems. This makes sense, as there has also been an exponential growth in the number of applications and their reachability to masses, while at the same time operating system and core application developers have adopted more secure coding practices. The most popular applications for exploitation tend to change over time because the rationale for targeting a particular application often depends on factors like prevalence or the inability to effectively patch. Browsers and client-side applications seem to be consistently targeted, taking advantage of the current trend wherein trusted websites are converted into malicious servers.

Worldwide, there has been a significant increase in the number of people discovering zero-day vulnerabilities, as measured by multiple independent teams discovering the same vulnerabilities at different times. Zero-day vulnerabilities are those for which there is no patch available and no widely-spread knowledge.

Additionally, one of the most important factoids that has come up in recent times is that every major industry is a potential victim, with none of them being spared and almost all organizations are vulnerable to security attacks.

Industry	Total	Small	Large	Unknown
Accommodation (72)	362	140	79	143
Administrative (56)	44	6	3	35
Agriculture (11)	4	1	0	3
Construction (23)	9	0	4	5
Educational (61)	254	16	29	209
Entertainment (71)	2,707	18	1	2,688
Finance (52)	1,368	29	131	1,208
Healthcare (62)	166	21	25	120
Information (51)	1,028	18	38	972
Management (55)	1	0	1	0
Manufacturing (31-33)	171	7	61	103
Mining (21)	11	1	7	3
Other Services (81)	17	5	3	9
Professional (54)	916	24	9	883
Public (92)	47,237	6	46,973	258
Real Estate (53)	11	3	4	4
Retail (44-45)	159	102	20	37
Trade (42)	15	3	7	5
Transportation (48-49)	31	1	6	24
Utilities (22)	24	0	3	21
Unknown	9,453	113	1	9,339
Total	64,199	521	47,408	16,270

Figure 4. Verizon’s 2016 Data Breach Investigations Report shows the number of confirmed security incidents by victim industry and organization size

The Source of the Vulnerabilities

But, who is to blame for the vulnerabilities that malware uses in an attack? The Internet is something that we all want—the ability to publish and find information, to buy and sell products, to communicate with others. However, this vast interconnection made possible by the Internet also lowers the bar for malicious action.

The major avenue of attack is through flawed software; that is, mistakes made in software that leads to exploitation. A typical example is a buffer overflow, in which a programmer reads unsanitized input, but does not compare the length of the incoming data against the amount of storage set aside for the response. An overly long malicious response can be used to crash the software, or cause it to execute arbitrary computer code—code designed to steal data or embed other attacks.

Vulnerabilities are classified as either known, unknown, or zero-day. Known vulnerabilities are typically first disclosed to or discovered by the authors of the software in question, allowing them to fix issues before public disclosure. Unknown vulnerabilities and zero days are those not yet discovered or disclosed to the creator of the software. Zero-day vulnerabilities are more harmful, as they are known to only a few and can be exploited with minimal risk of discovery. Such vulnerabilities may be abused for years until patched.

Misconfigured networks, servers, and clients offer another avenue for hacking. Network elements, such as routers and IoT devices, come with a default administrator password, which is often never changed. A hacker with access to a router can cause all traffic to be proxied through their own server, allowing 'man-in-the-middle'(MiTM) attacks.

Similarly, misconfigured servers can allow hackers to disable or modify web sites, inserting code of their own choosing. Such code is usually intended to steal data from associated databases. Easily guessed administrator passwords, shared private keys for access, and SQL injection attacks are often means of gaining access.

Finally, many users are simply to blame. Creating complex passwords or enabling two-factor authentication is not a major concern for most users. We are often not careful enough—gullible or too trusting—allowing attackers to get us to cooperate with their plans.

The Damage

The damage from successful network security attacks can take many forms:

- **Loss of data.** This often consists of financial data, such as credit card numbers, but also includes customer lists, intellectual property, and product development and marketing plans.
- **Loss of time.** It can take a great deal of time to recover from a security breach. Data may need to be recovered or reconstructed and systems extensively checked and network security reworked.
- **Loss of money.** This is often preceded by the theft of data.
- **Disabled or crippled services.** Protesters and some governments may seek to disable offending websites. Hackers also may be purely malicious in their intent.
- **Legal exposure.** Any of the previous items may expose an organization to lawsuits for loss of data or money entrusted to them.

Classification of Security Attacks

User-Involved Attack Mechanisms

Computer users are the primary avenue used in security attacks. The most frequent methods include the following:

- **Email.** In addition to spam, emails may contain attachments that are malicious executable programs or links to infected websites. Waves of targeted email attacks, often called spear-phishing, are exploiting client-side vulnerabilities in commonly used programs such as Adobe® PDF Reader®, QuickTime®, Adobe® Flash® and Microsoft® Office. This is currently the primary initial infection vector used to compromise computers that have Internet access.
- **Web.** Those same client-side vulnerabilities are exploited by attackers when users visit infected websites. Because the visitors feel safe downloading documents from the trusted sites, they are easily fooled into opening documents, music, and video that exploit client-side vulnerabilities. Some exploits do not even require the user to open documents. Simply accessing an infected website is all that is needed to compromise the client software. Websites can be dangerous in several ways:
 - Masquerading as valid websites collecting financial and personal information.
 - Infecting through content injected from associated websites. The average commercial web page contains content from more than 100 sources—advertising, tracking, and content. One or more of those sources may have been compromised and may insert code that is used to collect and send data to a third party.
 - Presenting false information. For example, a web page advertisement might suggest that a user's computer is infected with a virus, inviting the user to click on a virus scanning program, which actually infects the computer.
- **Instant Messaging (IM).** IM programs now provide mechanisms for passing executable programs and web links, providing a means of infecting computers and revealing information.
- **Peer-to-peer (P2P).** P2P environments are often used to share software, which may be similarly infected.
- **Gaming.** Social interaction with other players may invite email or IM communications. Games themselves, when executed on a user's computer, may be the source of infections. Games that must be run in administrator mode or use ActiveX or JavaScript are especially suspicious.
- **Software updates.** Software vendors are increasingly updating their software over the Internet, using web pages, or dedicated resident programs. Malicious parties may substitute their own software, or infect the updates before they are downloaded.
- **People.** End-users are frequently at fault for the following reasons:
 - **Using poor passwords.** Using easy-to-guess passwords or reusing the same set of passwords over and over again.

- **Inconsistently updating their software.** Many attacks take advantage of known operating system and application vulnerabilities. Software vendors usually offer software updates that plug these vulnerabilities, but they must be applied by the end-user.
- **Getting too personal.** Online groups often ask for personal information, for example, spouse, children, and pet names. This information may be used for identity theft or password guessing.
- **Being too trusting.** Friends and other acquaintances may send us software or websites, and we frequently trust them because we know them. They may have been duped or the message may have been falsified.
- **Inconsistent application of security software.** Computer security can be confusing for a computer user, including personal and corporate firewalls, anti-virus software, anti-spam software, and browser and email protection. All types of protection must be applied.
- **Engaging in wishful thinking:** "It will not happen to me."

Web vulnerabilities comprise 49% of the total number of those reported. The cumulative number of reported web vulnerabilities is more than 20,000. Attacks against web applications constitute more than 60% of the total attack attempts observed on the Internet. These vulnerabilities are being exploited widely to convert trusted websites into malicious websites serving content that contains client-side exploits.

Network-Level Attack Mechanisms

Many attacks are mounted without user involvement. The Internet depends on a number of services accessible to everyone: Web, DNS, FTP, SMTP, POP, IMAP, and SIP to name just a few. The server software used for these services, plus the many plug-ins that are used in conjunction with the services, are an attractive target for hackers. All software has vulnerabilities, and hackers are able to find them and exploit them for theft or other nefarious purposes.

Sites that offer their users remote access may rely solely on user passwords. Automated 'robots' may try long lists of possible passwords to gain access. They may use information that the user has provided to others; for example, see the *Getting Too Personal* earlier.

DDoS attacks are another network-level threat in which the attacker sends a large volume of malicious traffic to a web- or other-server. The purpose of the attack is to disable access to or from the service.

Vulnerabilities

As discussed, Vulnerabilities are a result of software flaws, flaws that fail to anticipate all possible conditions, especially unusual user input. Software flaws exist in all software; most are innocuous, but many provide the means for security penetrations. Three types of vulnerabilities predominate:

- **Cross-site scripting (XSS).** This type of exploit inserts HTML or other web content into web pages before they are displayed to the user. Such code is often used to steal personal information or to direct the viewer to a different website.
- **SQL injection.** This type of exploit extracts information from a database. For example, users might be prompted for their account information; the web application may be expecting a simple answer such as John Smith and use that name in an SQL query of the form:

```
statement = "SELECT * FROM users WHERE name = " + userName + ";;"
```

However, a hacker answering by typing:

```
' or '1'='1
```

will generate the following query statement:

```
SELECT * FROM users WHERE name = " OR '1'='1';
```

The result would be that the statement would return information for all users in the database. Proper care in programming would prevent SQL injection attacks, but many websites are still vulnerable.

- **File includes.** This vulnerability is similar to SQL injection in that it takes advantage of unchecked user input. Such input may be used with websites that use PHP or Java. The unchecked user input is used to include additional code from a hacker's site using file include facilities in the web language.

The number of published vulnerabilities have hovered between 5000 to 7000 mark in the last few years according to the data from National Vulnerability Database (NVD)⁴. Figure 4 shows the trend of the number of vulnerabilities from 2010 –, 2016.

⁴ <https://web.nvd.nist.gov/view/vuln/search>

Network Security Overview

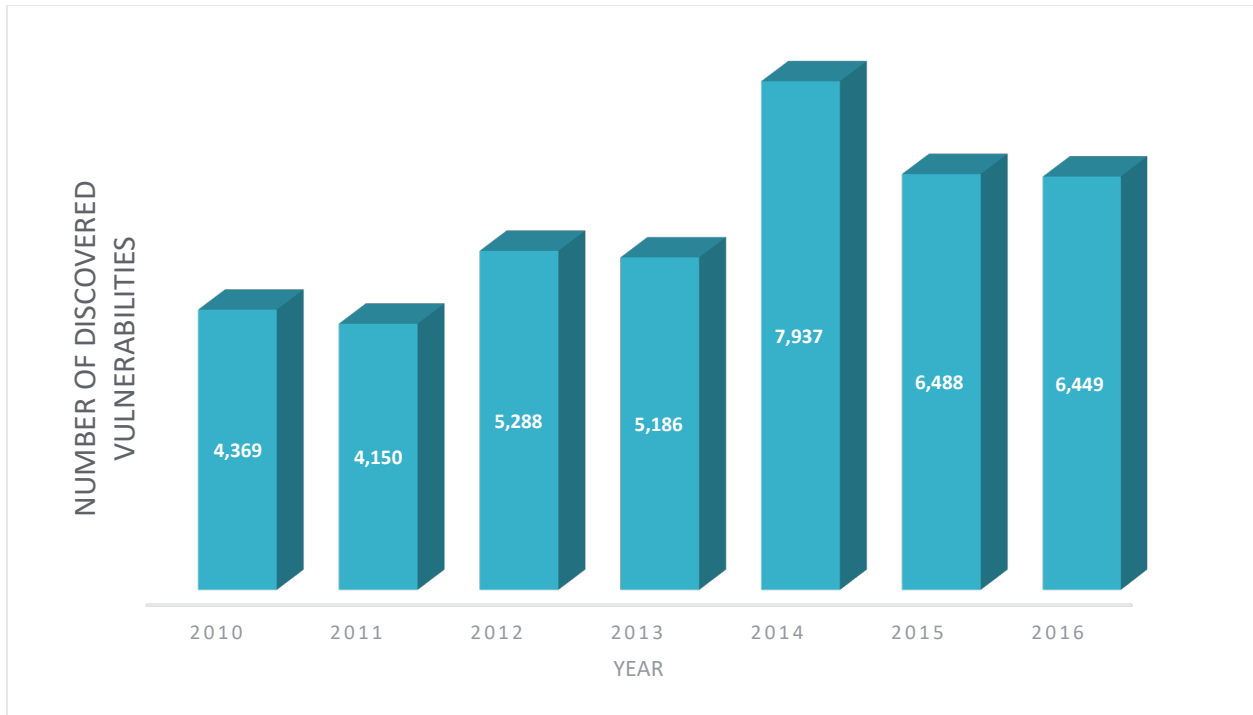


Figure 5. Number of vulnerabilities per year from 2010 – 2016-May – Source: National Vulnerability Database

In 2016, third-party applications were the most important source of vulnerabilities, with around 78% of the reported vulnerabilities. Operating systems were only responsible for 18% of vulnerabilities and hardware devices for 3%.

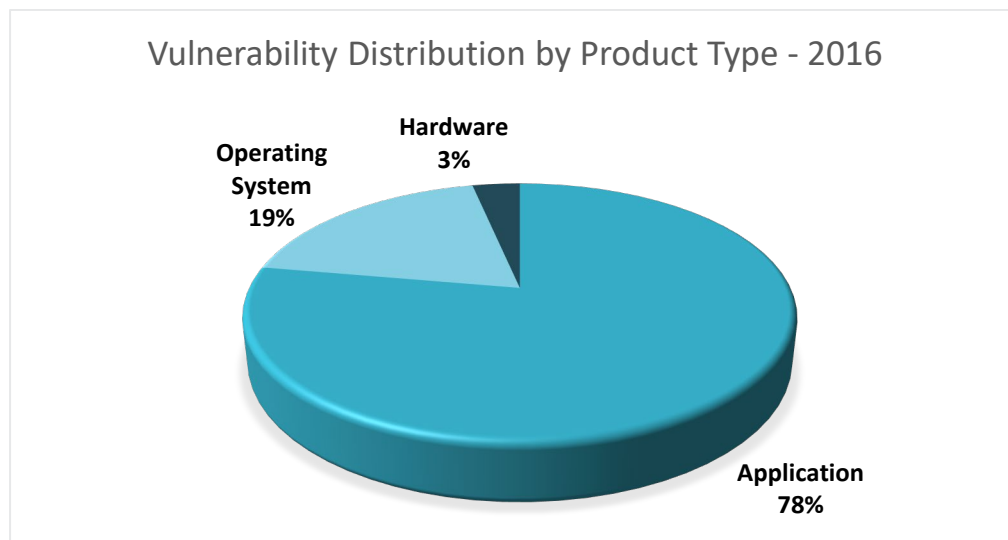


Figure 6. Vulnerability by product type – 2016 - Source: National Vulnerability Database

In the virtual world, vulnerabilities in the hypervisor can be critical, leaving the guest OS exposed to various forms of exploits. One such vulnerability discovered in 2015 was the venom vulnerability that impacted Xen, KVM, and QEMU hypervisors. This vulnerability allowed one guest OS access to the host, thus affecting every guest OS running on the same hypervisor.

With increased focus on virtualization and software defined data center (SDDC), it is crucial that hypervisor-level vulnerabilities are detected and blocked by security measures deployed in the SDDC.

Malware

Malware is the term used to describe the entire gamut of malicious software. For this discussion, we will break them down into these categories:

- Viruses
- Worms
- Trojans
- Drive-by Exploits
- Rootkits
- Exploit Kits
- Spyware
- Ransomware

Although we can distinguish these types, modern malware is very often hard to categorize—blending multiple types of attacks. You can look at this list as types of behaviors that malware can exhibit and some malware may fit more than one of the following categories.

Viruses

The term virus is often used instead of malware, but actually refers to a computer program that infects a computer and is spread with minimal user action. A virus typically attaches itself to another program. User actions include email and physical distribution through CD, DVD, or USB drive. Viruses often establish themselves on shared network file systems that may be accessed from multiple computers.

Worms

Worms are self-spreading programs that take advantage of security vulnerabilities. They spread themselves to other network nodes without any user interaction. Worms typically stand alone and do not need to attach themselves to other programs. They may consume bandwidth, or corrupt or modify files. The first significant worm was the Morris worm, which was released on November 2, 1988. It took advantage of known vulnerabilities in Unix sendmail, finger, and rsh/rexec, as well as weak passwords. It was originally intended to measure the size of the Internet, but a coding error caused it to infect computers multiple times, rendering them too busy to be useful. Other historically notable and highly damaging worms include the Witty Worm, SQL Slammer, and Conficker.

Trojans

Trojans are programs that appear harmless, but hide malicious functions. These functions are often remotely controlled by central computers. They are particularly insidious because they may do nothing for long periods of time, or only intermittently. They may do the following:

- Make the computer available as part of a network of remotely controlled computers, called a botnet
- Steal personal user information, either by scanning local information or by injecting code into web forms and email
- Install other malware
- Download or upload files, wasting computer storage space and network bandwidth
- Modify or delete files
- Log keystrokes to discover passwords and other information
- Transmit the contents of a user's screen

Drive-by Exploits

Drive-by Exploits, also known as drive-by downloads, refer to injection of malicious code by the HTML of websites that exploit vulnerabilities in web browsers. They are increasingly used by attackers to target web browser plugins such as Java, Adobe Reader, and Adobe Flash. These malware attacks can infect your computer simply by your visiting a website that is running malicious code. Most of the time these are legitimate websites that have been compromised to redirect you to another site controlled by the hackers. Once infected, the malware does what it was designed to do, which is mainly to make money for its masters.

The malware known as Zbot can access your email or bank accounts. Another type of payload called ransomware can hold your files hostage until you pay to have them released.

Rootkits

Rootkits are self-obscuring programs, hiding as normal operating system files. In doing so, they disable security software packages that might discover them. As part of the operating system, they can perform any number of functions. For example, in Unix-based systems, they can hide as the *login* program, capturing valuable user passwords for later use. Rootkits exist for a wide variety of operating systems, including Microsoft Windows, Linux, Mac OS, and Solaris.

Exploit Kits

Exploit kit is a toolkit that automates the exploitation of client-side vulnerabilities, targeting browsers and programs that a website can invoke through the browser. A key characteristic of an exploit kit is the ease with which it can be used even by attackers who are not IT or security experts. The attacker doesn't need to know how to create exploits to benefit from infecting systems. Further, an exploit kit typically provides a user-friendly web interface that helps the attacker track the infection campaign.

Some exploit kits offer capabilities for remotely controlling the exploited system, allowing the attacker to create an Internet crimeware platform (Malware-as-a-service) for further malicious activities like delivering other payloads. These may include a bot, a backdoor, spyware, or another type of malware.

Spyware

Spyware is a type of hidden malware that collects and forwards user and computer information. It can be used to collect various types of personal information, such as Internet surfing habits. Spyware can also interfere with user control of their computer in other ways, such as installing additional software and redirecting web browser activity. Spyware has been known to change computer settings, resulting in slow connection speeds, altered home pages and loss of Internet connectivity, or loss of functionality of other programs.

Ransomware

In the simplest term, Ransomware is a malicious software designed to encrypt user data or block access to contents within a user's system until a certain amount of money is paid. 2016 saw the meteoric rise of Ransomware as cyber criminals found a way to extract money from individuals as well as large corporations, thereby expanding their economy. Ransomware has used all the usual tricks from mutation to multi-level-evasions to ensure deeper penetration. Although authorities have issued regular warnings to users to not heed to ransomware demands. However, users' affection to their data and the fear of losing it ensured that they pay enough to contribute to the continuous growth of Ransomware.

Spam

Spam includes any type of unwanted message. Spam is usually delivered by email but these days it can be text messages, instant messaging, and Internet telephony-based spams. In the recent past, the email-based spams have been more of annoyance than a major threat, largely due to sophisticated spam filters. However, the emergence of snowshoe spam has put spam back on the map. Snowshoe spam, unlike regular spam, is not sent from one computer, but from thousands of users, each sending messages in low volume. It is easy to block spam coming from one location, but when it comes from many, it becomes difficult for anti-spam software to keep up. Even more problematic is the fact that a lot of snowshoe spam has been tied to legitimate bulk email addresses, such as the ones that send you newsletters that you voluntarily signed up for.

While spam is usually harmless, there is always the threat that snowshoe spam might evolve into a gateway for a large, harder to detect phishing attack—or even as a method to spread malware without setting off alarms.

Malicious Adware/Scareware

These are scams that trick a user into downloading and executing software, which may or may not contain malware. For example, the screen shown below invites the user to download a free scan program, which actually is malware.



Figure 7. Example scareware advertisement

Phishing

Phishing scams are fraudulent attempts by cybercriminals to obtain private information. Phishing scams often appear in the guise of email messages or websites designed to appear as though they are from legitimate sources. For example, the message would try to lure you into giving your personal information by pretending that your bank or email service provider is updating its website and that you must click on the link in the email to verify your account information and password details.

Botnets

A botnet is a group of computers or devices (especially in the context of IoT) connected to the Internet that have been compromised by a hacker using a computer virus or Trojan horse. An individual computer in the group is known as a zombie or a bot that communicates to the bot master, which directs them to perform malicious activities (command & control). Botnets have evolved from single purposes, such as spamming, distributing malware, or conducting DDoS attacks, to a multi-purpose army that can be rented by interested parties.

DDoS

DDoS attacks are an attempt to make a computer resource such as a web site or web service unavailable to users. One of the most common methods of attack involves saturating the target (victim) machine with external communications requests. The machine then cannot respond to legitimate traffic, or the response becomes so slow that it is effectively unavailable. The DDoS attacks are often launched by networks of zombie computers or botnets. This kind of attacks do not target directly the confidentiality or integrity of the information resources of a target, but they can result in significant financial and reputation loss.

The volume of traffic is critical when it comes to creating an effective DDoS, where the largest attack to date topped at near terabit levels. Also, application-based DDoS attacks (HTTP, DNS, SMTP, NTP, or SIP), including those leveraging reflection or amplification techniques are becoming more prevalent than simple stateless flood based attacks (UDP, ICMP, or TCP SYN).

Advanced Persistent Threats (APTs)

APTs are a type of targeted attack aimed at companies or institutions. Behind these attacks are usually organizations that invest huge sums of money to ensure that the target attack goes undetected for a long time. Often the objective of targeted attacks is either data exfiltration or gaining persistent access and control of the target system. This kind of attack consists of an information gathering phase and the use of advanced techniques to fulfil the attacker's goals. The first phase can possibly involve specially crafted emails (spearphishing), or infected media and social engineering techniques. The second phase involves advanced and sophisticated exploitation techniques.

Data Leakage/Breach

Compromising of confidential information refers to data breaches that occurred via intentional or unintentional information disclosure performed by internal or external threat agents. This threat targets sensitive information from various sectors such as the public health sector, governmental organizations, small-medium businesses (SMBs), large organizations. Data breaches are usually realized through some form of hacking, incorporated malware, physical attacks, social engineering attacks, and misuse of privileges.

Network Security Testing

Network security is a critical concern for enterprises, government agencies, and organizations of all sizes. Today's advanced threats demand a methodical approach to network security. In many industries, enhanced security is not an option. U.S. federal regulations such as Sarbanes-Oxley, HIPAA, GLBA, and others require organizations such as financial institutions, health care providers, and federal agencies to implement stringent security programs to protect digital assets.

The layered approach represents the best practice for securing a network. It is based on maintaining appropriate security measures and procedures at five different levels within a network:

1. Perimeter
2. Network
3. Host
4. Application
5. Data

Network security professionals speak in terms of 'work factor,' which is an important concept when implementing layered security. Work factor is defined as the effort required by an intruder to compromise one or more security measures, which in turn allows the network to be successfully breached. A network with a high work factor is difficult to break into, while a network with a low work factor can be compromised relatively easily. If hackers determine that a

Network Security Overview

network has a high work factor, which is a benefit of the layered approach, they are likely to move on and seek networks that are less secure.

Figure 86 details the accepted security levels, along with the types of security tools used at each level. Ixia tests products and software at the perimeter and network levels, which will be the subject of this document.

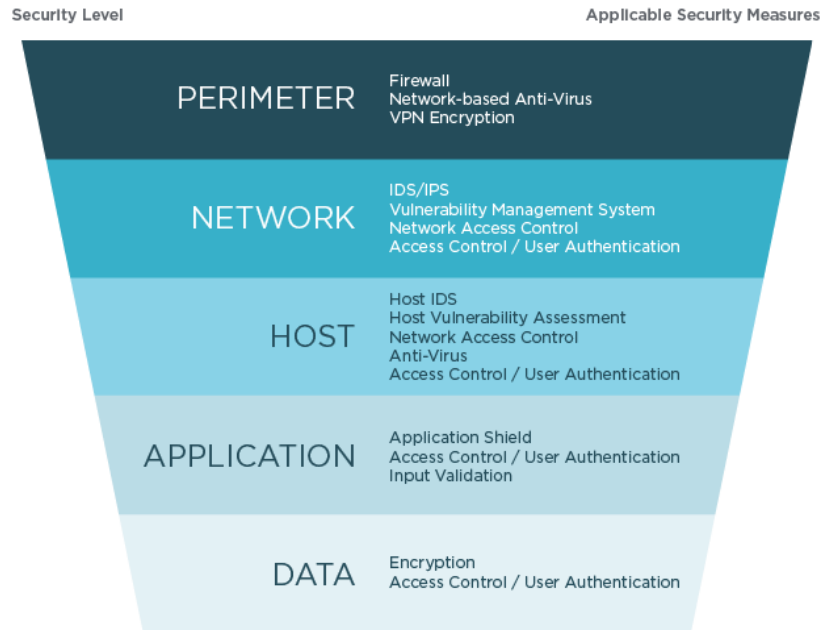


Figure 8. Security levels

Network Security Devices

The figure below is a simplified diagram of an enterprise network, complete with security devices.

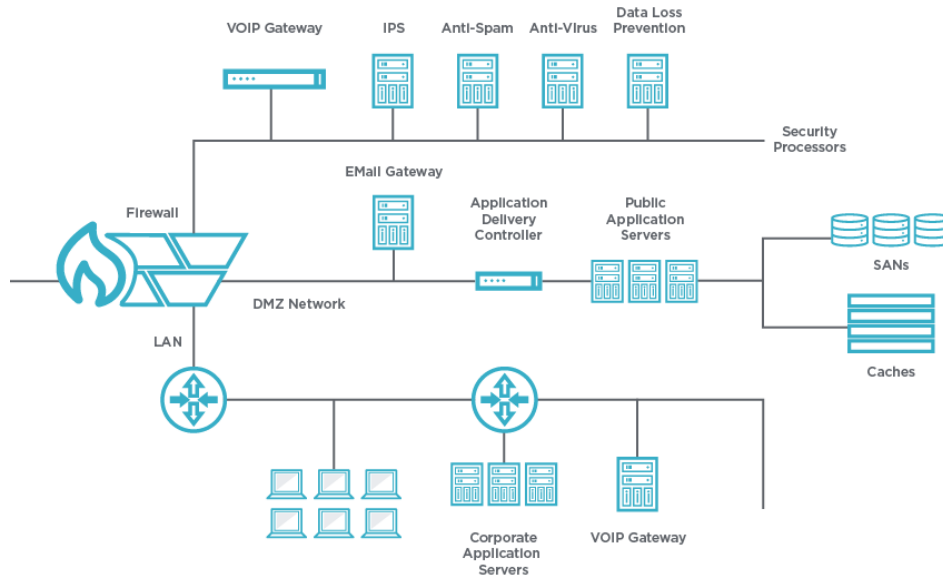


Figure 9. Simplified secured enterprise network

Network Security Overview

Network security devices have evolved over the years from a firewall to unified threat management (UTM) devices to NGFWs. Some security devices are for a specific purpose (anti-spam, anti-virus), while others integrate multiple different functionalities into one device. In this section, we describe some of the individual components that are used in network security devices.

The security components that will be discussed in the following sections include:

- Firewall
- VPN gateway
- IDS/IPS URL filtering
- Anti-virus
- Anti-spam
- Data loss/leakage prevention
- DDoS mitigation
- Advanced threat protection

The security processors, when they are not integrated into a UTM device, are normally connected to a private network connected to the firewall. Servers that offer public services, such as email and web are kept on a private network called the demilitarized zone (DMZ). These private networks serve to isolate them from the local area network (LAN) users.

Firewalls

Firewalls were the first independent security devices used with external network connections. The purpose of the original firewalls was to ensure that only required connections were allowed into the enterprise network. This typically includes services offered to the public: email, web, FTP, DNS, and a few others. Firewalls are also used to limit the types of services that internal computers may access outside the enterprise. This serves to somewhat limit malware from contacting external servers.

Firewalls initially operated by filtering connections based on a 5-tuple, as shown:

- TCP or UDP
- Source IP address
- Source port number
- Destination IP address
- Destination port number

Network Security Overview

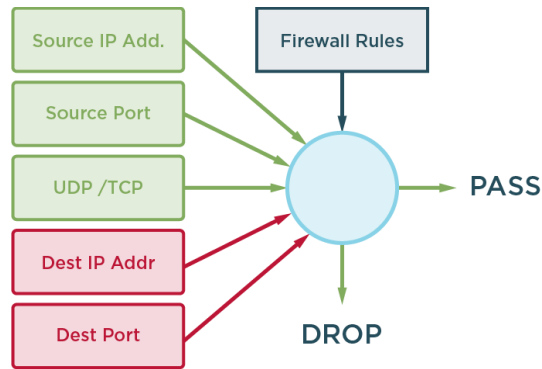


Figure 10. Basic firewall operation

Firewall rules are applied against connections attempted through the firewall, either inbound or outbound, to determine whether the connection is allowed or not. This worked well for a number of years, but as services and their protocols multiplied and applications began to use HTTP's port 80 as their transport mechanism, the ability of firewalls to meaningfully control traffic diminished.

With the explosion of applications on top of TCP, rules based on just the 5-tuple was no longer enough. With the increase in applications, there was also an increased threat of hackers misusing the applications for attack purposes. Further, attackers became more sophisticated and used evasion techniques to frustrate the traditional firewall rules. To handle this, most firewalls today are application aware. They detect the application being carried in the IP packet by using deep packet inspection (DPI) and can detect an application regardless of the underlying TCP or UDP port being used. This allows IT administrators to set up granular policies to allow or block certain applications (allow YouTube but block Facebook, for example) or certain aspects of some applications (allow Facebook chat but disallow Facebook upload, for example).

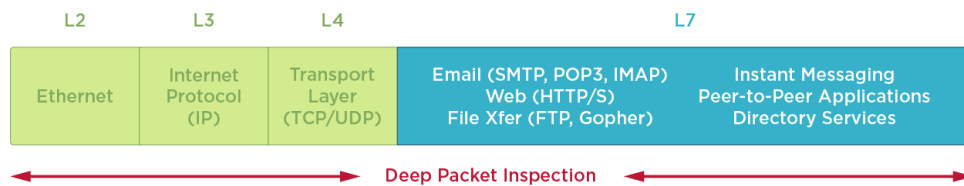


Figure 11. Deep packet inspection

Another feature of firewalls today is the ability to inspect SSL traffic. Now, a growing number of sites ranging from social media to unified communications use SSL by default. Moreover, these sites are now using SSL on all pages rather than selected pages deemed appropriate for encryption. However, hackers can also use SSL to obscure attacks or use SSL to deliver malicious payloads. Attacks hidden inside SSL can't be subject to firewall rules or protection unless the SSL is decrypted first. Thus, most next generation firewalls today also provide SSL inspection capabilities. They decrypt the SSL traffic, apply the inspection rules and then encrypt the traffic in real time to provide protection even for SSL traffic.

VPN Gateway

VPN gateways are used to securely connect multiple sites within an enterprise, remote and roaming employees, and business partners. Two protocols are commonly used:

- **SSL**. This protects and encrypts traffic, while providing a web-based interface for information access.
- **IPsec**. This is network-level security that encapsulates and encrypts all traffic between the gateways, as shown.

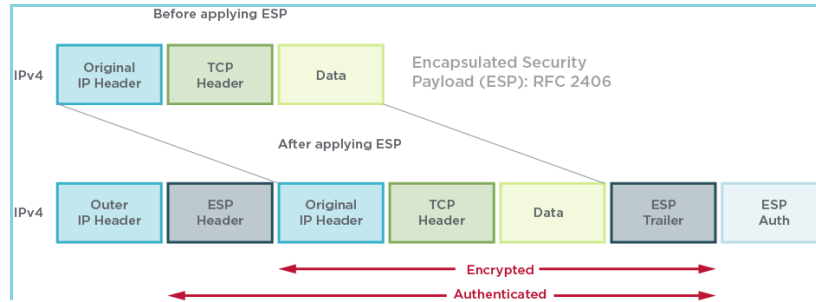


Figure 12. IPsec encapsulation

The original packet is encapsulated within a new packet that includes an additional encapsulated security payload (ESP) header. The header and additional trailers, and an optional authentication header (AH), serve to ensure that the source of the packet can be validated.

IPsec is used when multiple sites wish full, transparent access to each other's networks.

Intrusion Detection and Prevention Systems (IDS/IPS)

Intrusion detection systems (IDS), use technology that passively monitors network traffic, looking for particular malicious patterns, such as repeated attempts to log on to an account. When they notice a pattern, they send alerts to administrators and sometimes modify firewall rules to restrict access from the offending IP address.

Intrusion prevention systems (IPS) are similar to IDS, except they are logically in line with traffic. That is, all traffic from the network is routed through the IPS. It is responsible for identifying and stopping suspected traffic. Specific IPS rules and signatures are used to control how many flows are watched and for how long, to ensure that the IPS does not significantly diminish the overall traffic flow. IPSs are complex systems, attempting to minimize the number of false positives.

URL Filtering

URL filtering seeks to keep users away from a restricted set of websites. These sites are generally classified as follows:

- Offensive content: pornography or other objectionable material
- Harmful content: containing malicious code
- Inappropriate content: pages deemed not proper to view at work, such as games or sports

The list of websites used with the first two categories is often distributed as a service from a security vendor, based on the experience of all its customers. IT managers create and maintain the last category, often based on lists from the security vendor.

Anti-Virus

Network anti-virus software, located on the firewall or UTMS, serves to identify and filter all forms of malware. It does this by looking at the network connections associated with protected services: email, web, IM, and others. The data within the stream is examined using a number of techniques that identify malware. Depending on the particular software, the connection or transfer may be aborted or the offending malware removed from the stream.

Each vendor has a set of proprietary techniques that they use to identify malware. A common technique is the use of signatures, which are particular unique sequences or bits of data that identify the malware.

Anti-Spam

Anti-spam network software has a great deal in common with anti-virus software, and is often bundled together. Spam is a growing problem, with more and more sophisticated, customized messages being delivered. List-based approaches often miss such messages. Users must remain skeptical and vigilant with respect to 'special' offers.

Data Loss and Leakage Prevention

DLP is different than other security precautions in that it looks at outbound versus inbound information. DLP seeks to keep company and client proprietary and confidential information from leaving the organization, either innocently or maliciously.

Outbound information flows, such as email, web form data, FTP, DNS, IM, and other channels are inspected. A list of rules, keywords, and policies are applied to determine whether the communication should be rejected or allowed. Such filtering is very tricky. For example, a brokerage company might disallow any account numbers to be sent to a customer, which may be frustrating for the broker and customer.

DDoS Mitigation

DDoS Mitigation is a defense mechanism designed to eliminate or minimize downtime from a DDoS attacks. There are dedicated DDoS mitigation devices and also a built functionality within the NGFWs, application delivery controllers and other networking devices.

DDoS mitigation requires correctly identifying legitimate traffic from users versus high volume of malicious traffic from bots and hijacked web browsers. Some of the DDoS mitigation techniques include comparing signatures and examining different attributes of the traffic including IP reputation, bandwidth per IP, cookie variations, HTTP redirect, and Javascript challenge.

Advanced Threat Protection

As mentioned APTs operate covertly and are difficult to detect, months can pass with no visible compromises to the organization quietly under attack. Therefore, APT prevention and protection is a systematic strategy and plans that will need to mitigate full spectrum attack vectors

discussed previously as well as unknown malware and zero-day exploits. Sandboxing technique is used to directly observe the behavior of the unknown malicious malware and zero-day exploits. In addition, it is critical to have a good visibility into the baseline of your organizations network traffic profile so that you can pinpoint suspicious activities.

Evasion Techniques

Security devices have a tough job: operating on large traffic volumes and keeping up with an ever-changing set of threats.

An additional complication is the ability of hackers to disguise their attacks through evasion techniques. A few examples are as follows:

- **URL obfuscation** – URL filtering may be confused by the use of backslashes instead of forward slashes, or the use of % escape characters instead of 'normal' letters.
- **Fragmentation** – IP packets are broken up into many smaller pieces, making for more difficult identification
- **Stream segmentation** – An attack taking place over one connection, email for example, might be interspersed with other traffic, potentially over a long period of time. Security appliances may need to stop looking at the original connection for lack of space.
- **Application evasion** – An attack or exploit is hidden in the application payload or header.

Testing Security Devices

Testing of network security devices requires a number of techniques, which will be discussed in the next few sections:

- Known vulnerabilities
- Data leakage testing
- Distributed denial of service
- Protocol robustness
- Line-rate blended applications mix traffic
- Encrypted traffic

Known Vulnerability Testing

Known vulnerability testing is the cornerstone of network security device testing. Attacks are mounted against the security device by using a large database of known malware, intrusions, and other attacks. A number of organizations exist to maintain this list. One leading organization is the U.S. National Vulnerability Database maintained by the National Institute of Standards and Technology (NIST). The Mitre Corporation provides access to this database, called the CVE—Common Vulnerabilities and Exposures. As of September 2016, there are about 78000+ known CVE IDs.

Proper security testing requires that a number of known vulnerabilities be applied to security devices at a significant percentage of line rate. The device under test (DUT) should properly reject all such attacks, while maintaining a reasonable rate of transmission of 'good' communications.

In addition, known vulnerabilities must be applied using the wide variety of evasion techniques. The combination of thousands of known vulnerabilities and dozens of evasion techniques requires that a subset of all possibilities be used for testing. Test tools offer representative samples, including special cases for newly published vulnerabilities.

Data Leakage Testing

Data leakage testing involves transmission of data from the 'inside-out' to determine if data loss prevention devices will detect the leakage of proscribed information. All outbound means must be tested, including email, email attachments, web-based mail, web form data, FTP, and IM.

Enterprises must create test cases for each of the rules, keywords, and policies that they use in the security device, including tests that should not be flagged. Network equipment manufacturers (NEMs) have a more difficult job—requiring a more extensive set of test cases that exercise each type of rule and policy, along with a sampling of keywords.

Distributed Denial of Service

DDoS attacks often use large numbers of computers or devices that have been taken over by hackers. Those computers use dozens of attack techniques designed to overload network and security devices. This type of testing requires test equipment capable of simulating thousands of computers.

The DUT must be tested to ensure that none of the denial of service attacks, singly or in combination, can disable the device. In addition, the ability of the DUT to accept new legitimate connections and provide an acceptable level of performance must be measured.

Protocol Robustness

There are literally hundreds of protocols associated with modern Internet systems. Each operating system vendor and NEM implements each protocol in its own way. Many protocols are used in the deployment of end-user services. For example, the following shows some of the protocols used in a voice over IP (VoIP) call.

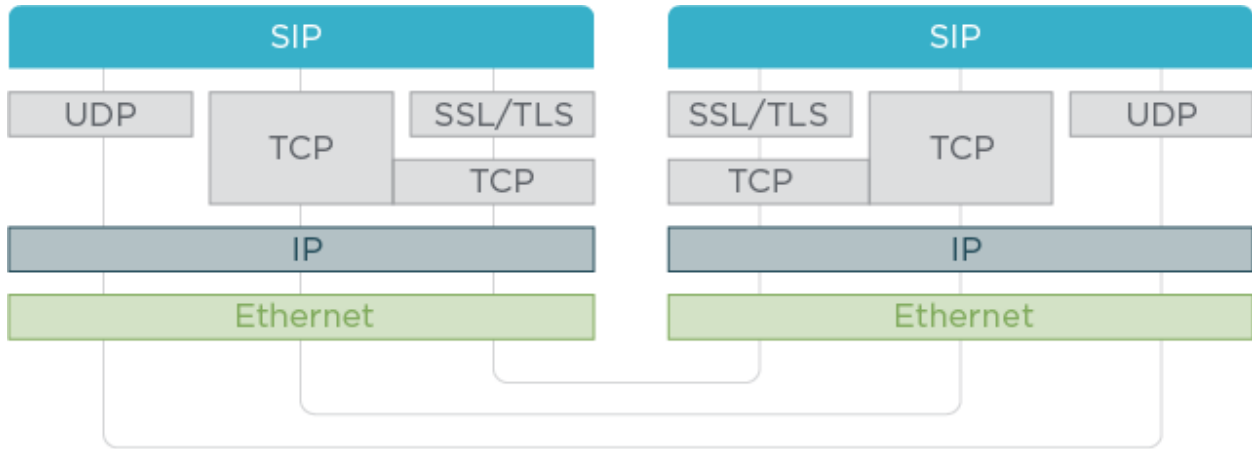


Figure 13. VoIP Protocols

Each and every protocol implements a complex state machine, complete with multiple state transitions and options handling. A perfect protocol implementation will properly handle each legal and illegal input.

These protocol implementations are generally tested for conformance to standards and proper functionality, but seldom extensively tested. Extensive testing requires that all corners of the protocols' implementation be tested. This type of testing is called protocol robustness/resilience testing, measuring the ability of a network device to handle unusual and malicious input. This type of testing finds design, configuration, and implementation flaws.

These types of flaws are often called 'zero-day' flaws, because they remain undiscovered until the first day of their deployment. These flaws can be particularly expensive; a newly offered service cannot be removed without loss of reputation or revenue. Some flaws have remained unpatched for years.

In principle, such testing could be accomplished by long sequences of random input, but the sophistication of today's protocols would require too long a period of time. The technique used for this type of testing is referred to as intelligent 'fuzzing.'

Intelligent fuzzing understands a protocol's state machine and the fields in the protocol that represent options. The fuzzing test machine uses this understanding to exercise a protocol's state machine, taking it through all normal legal transitions while trying illegal inputs along the way. In addition, all plausible options are attempted along the way. During testing, fuzzing checks for proper protocol behavior by monitoring the network connection.

Line-Rate Blended Applications Mix Traffic

Not only must security devices fend off attacks, but they must pass non-malicious traffic at the same time. To ensure this, it is necessary to test for defense against attacks with a background of real-world blended application mixes that replicates your network traffic profile. That is, a mix of social networking, P2P, media, data, and other services that constitute normal traffic should be applied to the DUT such that the sum of the malicious and normal traffic is the maximum for the device's interfaces.

The quality of experience for each of the normal services must be measured to ensure that the end users' satisfaction will not be sacrificed. For example, voice over IP requires very little bandwidth, but latency and jitter impairments are immediately heard by the human ear.

Encrypted Traffic

As enterprises move to connect their multiple sites and mobile and remote users together into a corporate VPN, SSL data encryption is becoming increasingly important. Data encryption ensures both privacy and authentication of the sending party using certificates or other techniques.

The process of establishing an encrypted link, and then subsequent encryption and decryption can be a significant load for a security device. It is essential that a realistic mix of encrypted traffic be mixed with clear text traffic during performance testing. This enables data center operators to understand the impact of SSL on their networks and find the right balance between security and end-user quality of experience. This is time-critical as traffic is moving toward total encryption, with the adoption of HTTP/2, 2/4k keys, and ECC ciphers.

Virtualization

Today's networks need to adapt quickly, and facilitate change. Strategies like network functions virtualization (NFV) and software defined networking (SDN) provide powerful flexibility gains by moving traditional application and security functions—like application delivery, load balancing, deep packet inspection (DPI), firewall, intrusion prevention system (IPS), and sandbox components—off dedicated hardware onto virtualized servers. Virtualized tools need to deliver the same or better performance and similar security efficacy than the traditional hardware appliances. Without a way to properly test these virtualized application and security devices, customer quality of experience (QoE) is at risk.

Ixia BreakingPoint

Ixia's BreakingPoint® application and security test solution enables users to validate the performance, stability, and accuracy of application-aware network security devices and the overall security posture of an organization's network. This Network Security test methodology Black Book details how BreakingPoint can be used to solve network security testing challenges.

BreakingPoint's unique design recreates every aspect of a realistic network, including scale and content creation—good, bad, and ugly network traffic simultaneously. Good refers to real-world legitimate application traffic with complete content flexibility. Bad traffic refers to the malicious traffic such as DDoS, exploits and malware. Finally, ugly refers to malformed traffic generated using fuzzing techniques. Combining all these with full control of the load capacity and detailed per-simulated host reporting makes BreakingPoint the ideal simple-to-use and repeatable testing ecosystem for modern network security testing.

BreakingPoint enables users to:

- Maximize security investments with onsite, network-specific, proof-of-concept (PoC) validation
- Optimize NGFW, IPS, and other security devices
- Ensure the always-on user experience in the midst of complexity and exploding traffic volume

Ixia BreakingPoint Virtual Edition (VE) delivers the same unique capabilities and workflow to validate network security and application performance of virtual infrastructure. The same methodologies can be extended to validate the scale, performance, and security effectiveness of your cloud and virtual network functions (VNF) without compromising security.

Test Methodologies for Known Vulnerabilities and Malware

Vulnerabilities represent flaws in a product that may allow malicious users to take control over the victim's computer, compromise data on it, allow remote execution of malicious code, or gain unauthorized access level.

The test methodologies covered by this section are trying to address the effectiveness, accuracy, and performance impact of devices that can protect the network against known, published vulnerabilities. Such devices include IPS/IDS systems, UTMS, and NGFW.

The key metrics that needs to be addressed while testing security devices include the following:

- Security effectiveness
- Resistance to evasion
- Detection accuracy
- Performance

Security Effectiveness

Security effectiveness refers to the ability of a network security device to detect and prevent threats. As detailed in the introductory section of the book, threats can consist in known vulnerabilities, unknown vulnerabilities, DoS and DDoS attacks, and malware.

Effectiveness measurements shall target the following:

- Effectiveness based on the security policy
- Effectiveness by attack vector
- Effectiveness by vulnerability's published date
- Effectiveness by attack source
- Effectiveness based on threat type

Effectiveness Based on Security Policy

Network security devices can be tuned to achieve maximum security protection, but usually, the elevated security comes at a cost by impacting the performance. To achieve the best balance between the security risk and the cost of security solution, measurements shall be conducted against different security policies. Many of today's security devices include a default security policy. Those should not be taken for granted, as the effectiveness of a default policy may be significantly different among products. Some vendors tuned their default policies to achieve the highest performance while reducing the protection level, while others provided highest protection with the cost in performance. Additionally, each deployment environment may require custom policies. Therefore, understanding the associated cost for a given security policy is important.

Effectiveness by Attack Vector

The largest number of known vulnerabilities target software that is used by a large number of users. Popular vendors like Microsoft, Adobe, Apple, and their applications are the main targets because they own large market share.

Assessing the effectiveness of the device should be determined in relation with the attack vector that exploits vulnerabilities specific to the environment where the IPS/IDS/UTM device is deployed.

The attack vector can be a set of vulnerabilities related to a vendor (for example, Microsoft, Apple, Adobe), or specific to an application (for example, Microsoft Internet Explorer, Mozilla Firefox).

Effectiveness by Vulnerability's Published Date

New vulnerabilities are disclosed daily, while software vendors address many of them in new versions of their software. Regardless of the availability of a patch, vulnerable software continues to be used. The older attacks can still be effective if they are targeting the right vulnerable software. Therefore, they continue to be relevant and protection against them shall be considered.

Effectiveness by Attack Source (Target Initiated vs. Attacker Initiated)

Attacks can be classified as target initiated or attacker initiated based on the source of the attack.

Attacker initiated attacks usually start by scanning the network perimeter of the victim, understanding the open ports and applications and operating systems used. After vulnerable software is found, the attacker executes the attack against the application and operating system. By exploiting the vulnerability, the attacker can execute the code remotely on the victim's computer. The attacker can also get root access, thereby gaining full control over the victim's computer. In this type of attack, the attacker controls when the attack is initiated.

The target initiated attacks are targeting client-based vulnerabilities. As an example, such an attack can be launched by a user who visits links that may exploit vulnerabilities in the browser, or open documents that are specially crafted to exploit vulnerabilities of application opening the document. This type of attacks relies on the victim's computer to initiate the attack. The time when the attack is initiated cannot be controlled by the attacker.

Effectiveness by Threat Type

There are many types of cyberattacks. No one single solution is effective against all types of attacks. A resilient network is expected to protect against many attacks types, including:

- Spear phishing
- Botnets
- DDoS
- Web application attacks, cross-site scripting, SQL
- Malwares

Resistance to Evasion

Not only do inline security devices detect network attacks in real-time, but they also have to normalize both network traffic and the content transmitted across. Many attacks can be obfuscated by taking advantage of protocol and application features and even undefined behavior. These techniques are collectively referred to as evasions or obfuscations. Many times, they can be layered on top of each other, making it very difficult to detect an attack. Failing to detect when one of those evasion techniques is being used gives attackers the opportunity to use multiple attack vectors leveraging the same evasion technique. Evasion techniques play a critical role in understanding the security risks present in your network and they should be a mandatory part of evaluations of inline network devices.

Some evasion technique examples are listed below:

Network- and Transport-Level Evasions

The TCP/IP stack has features that make stream reassembly particularly difficult. On the IP layer, packets can be fragmented and overlap across byte boundaries, be sent repeatedly and out of order. Similarly, on the TCP layer, segments can also be sent in small sizes, out of order, and repeatedly. TCP control flow flags (e.g., SYN/ACK/URG/FIN/RST) can be manipulated to confuse state machines.

Presentation- and Session-Layer Evasions

Many protocols have extensions and features that create a larger surface to hide and evade detection. Some simple examples are present in HTTP. Differing compression methods, characters and URL encoding, and mixed-case headers can all evade poorly written regular expressions. But HTTP is not the only protocol that enables these types of evasions. Some popular ones include SMTP, FTP, POP, and SIP. Another example, Remote Procedure Call (RPC) provides several features that can be used by attackers as an evasion technique: fragmentation support for application layer, several ways to represent the same data, option to create multiple bindings with a single request, and context alteration.

A common category gaining increased adoption is cryptographic protocols like SSL/TLS, which can make real-time analysis and detection challenging, if not impossible.

Application-Layer Evasions

Application-level evasions reside within the content delivered to users. Frequent examples include obfuscating malicious content within documents such as RTFs, PDFs, and Microsoft Office documents. Multiple obfuscating methods are deployed for scripting languages like Microsoft macros, powershell, and javascript. Document compression and packaging tools, such as LZMA, gzip, and tar are used with frequent success.

Some of these techniques require that more-advanced tooling be put in place to detect the malicious content, such as application proxies, sandboxes, inline A/V engines, and cloud-based threat feeds.

Detection Accuracy

Network security devices such as NGFW, IPS/IDSs and UTMs are placed inline to block internal and external attacks. To distinguish legitimate traffic from malicious traffic, such devices include complex techniques for traffic analysis and detection, which may include deep packet inspection, statistical and behavioral analysis, fingerprinting, signature dictionaries, regular expressions, and partial document matching. By filtering all the incoming and outgoing network traffic, valid connections may end up being blocked by the device, causing a denial of service. Therefore, the strength of the detection engine directly correlates with the detection accuracy.

Testing for accuracy is critical in ensuring that a solution has no false positives or false negatives.

Performance Impact

One of the most common effects when additional devices are placed inline is the increased latency. End to end latencies exceeding 150 ms will start affecting the quality of VoIP calls. Excessive network latency can also cause applications to spend a large amount of time waiting for responses from its remote peer, resulting in lower bandwidth usage. Different security policies impact differently the performance. Another variable is introduced by the type of traffic. Parsing SIP traffic compared with HTTP traffic may result in a larger processing effort, therefore impacting the performance differently. Lastly, the presence of malicious traffic results in additional processing operations that the device needs to take care of, potentially impacting the performance.

Benchmarking network security devices should start with baseline tests to assess the raw forwarding performance of the device. In those tests, all the security services must be disabled and the device must act as a simple forwarding element.

Test cases must cover raw performance for UDP and TCP protocols

- UDP - RFC 2544 Throughput Measurements
- TCP - Maximum Concurrent Connections
- TCP - Maximum Connections Rate
- TCP - Maximum Throughput

In the second step, the same test cases are repeated, but this time the security policies are enabled on the device. For each security benchmarking, the test cases must be repeated.

Because NGFW, IPS/IDS and UTM devices relies on deep packet inspection, the following set of DPI test cases should be covered:

- Max DPI Capacity and Performance with HTTP
- Maximum DPI Capacity and Performance with real world application mix

The application mix should match as close as possible the traffic mix seen in the deployment network. The traffic characteristics are different and the DUT's performance can be impacted differently. Profiles covering traffic patterns inspected by the IPS/IDS/UTM devices deployed by

Test Methodologies for Known Vulnerabilities and Malware

universities, enterprises, data centers, service providers, financial, and government organizations are good examples.

After the baseline performance numbers are established, the tests must be repeated in the presence of malicious traffic. An assessment of the security effectiveness must be conducted while generating traffic at different capacities. Recommended values include 25 percent, 50 percent, 75 percent, 90 percent, and 99 percent of the capacity determined by using the baseline test cases.

Test Case: Measuring Security Effectiveness of Intrusion Prevention Engines

Overview

Network-based Intrusion Prevention Systems (IPS) are playing an essential role in any enterprise and datacenter security solutions. This test determines the security effectiveness of a network-based Intrusion Prevention System against published vulnerabilities targeting both client and server applications. Breaking Point System includes various strikes that can be used to replicate the communication between the attackers and vulnerable targets. Further, Ixia's Application and Threat Intelligence (ATI) program releases new exploits and malware every two weeks to keep customer's up to date with the latest attacks.

To baseline the security effectiveness, we recommend that you send the attacks sequentially, at lower rates without any additional traffic. While the presence of additional benign traffic may impact the security effectiveness, we recommend that this type of test be executed upon identifying the list of attacks that are successfully blocked by the IPS, and use attacks that have been previously detected and blocked to assess any impact that legitimate application traffic may add.

Objective

This test measures the security effectiveness of network-based IPS against attacks targeting published vulnerabilities on client and server applications. This test uses the predefined "critical strikes" in BreakingPoint as an example (Strikes that exploit vulnerabilities with a CVSS score of 10.0), but you can run the test using user-defined lists (custom list of attacks).

Setup

The setup requires at least two test ports – one acting as an initiator and the other as a responder.

Test Methodologies for Known Vulnerabilities and Malware

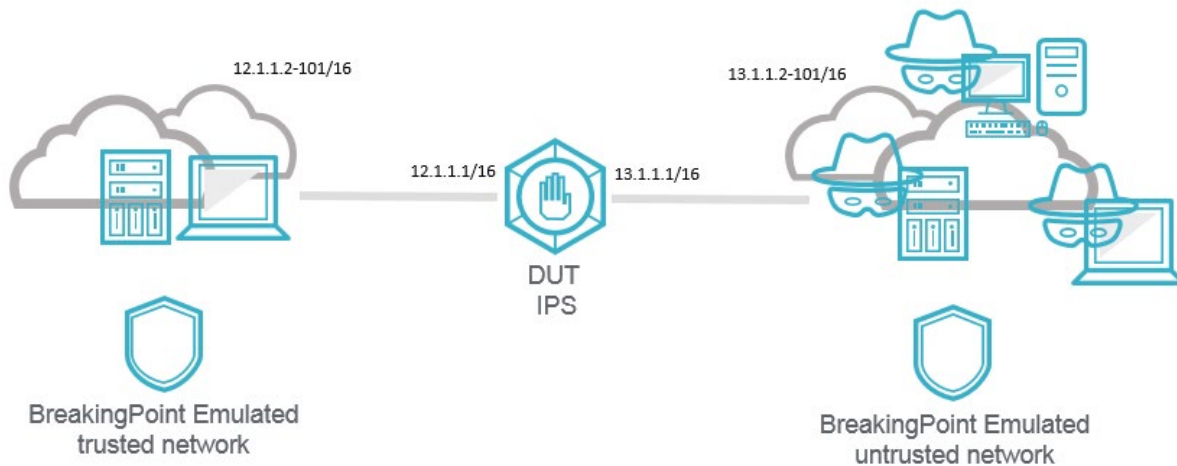
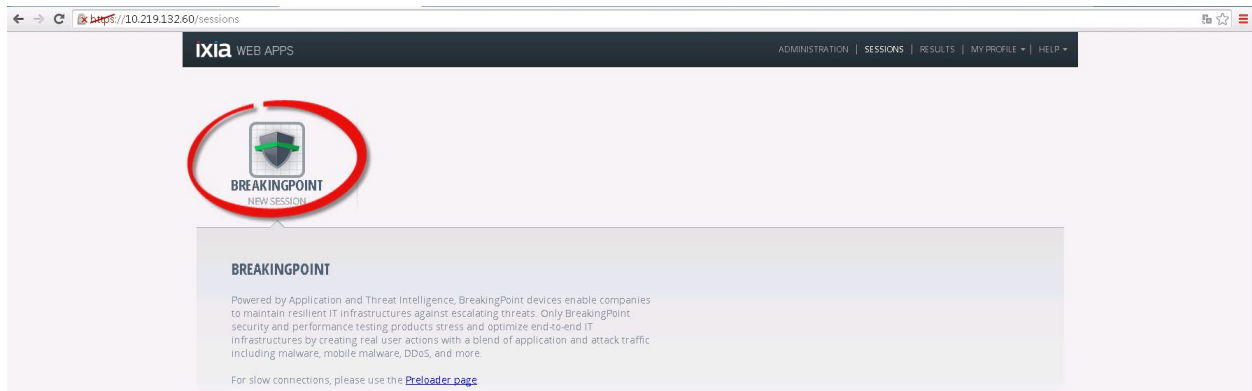


Figure 14. Test Setup

Configure the policy of the device to allow both inbound and outbound communication for any traffic/protocol on any port (allow ANY to ANY). Configure the IPS to provide the maximum protection against exploits targeting published vulnerabilities.

Step-by-Step Instructions

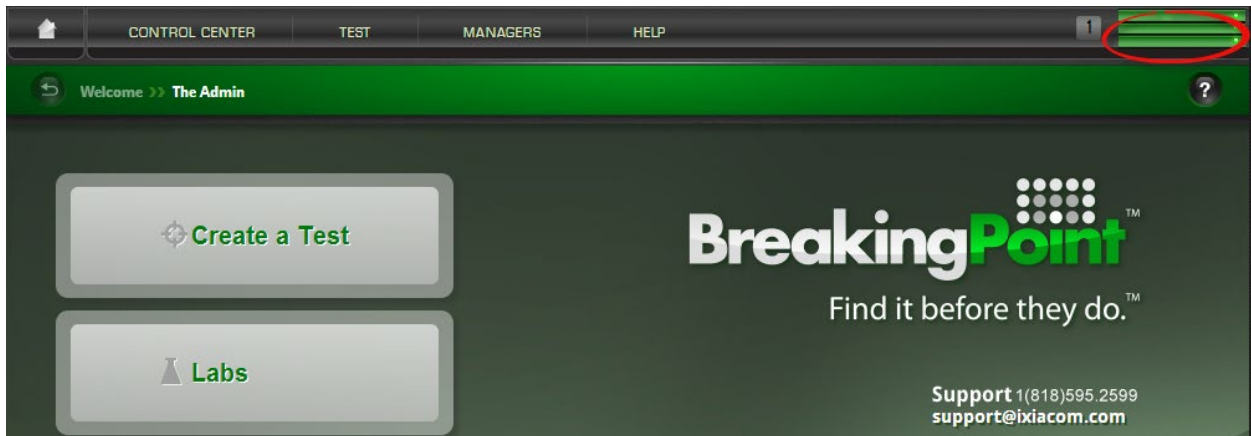
1. Use a web browser connected to the Ixia Web Apps GUI and start a new BreakingPoint Web Session:



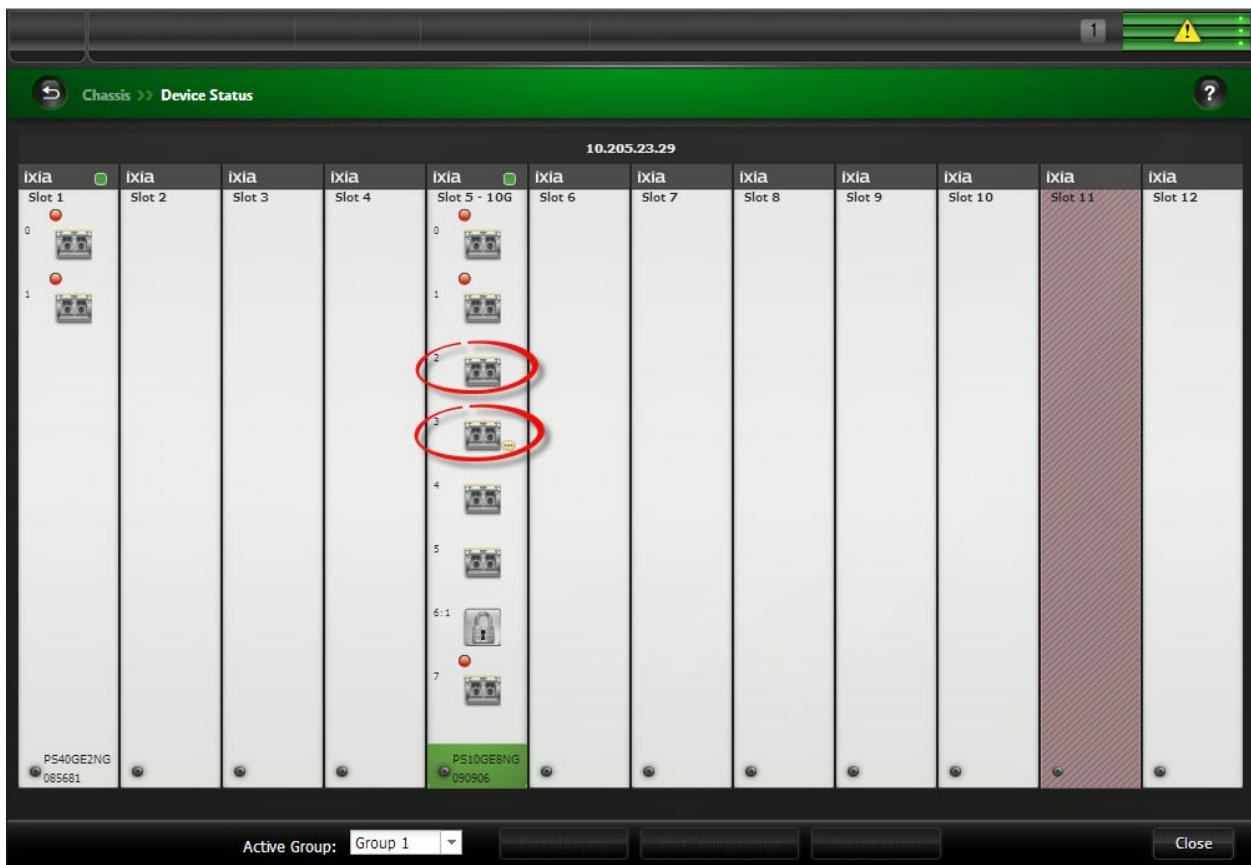
2. Once the BreakingPoint web home page loads, reserve the test ports to be used in the physical test setup to generate/receive traffic:

Test Methodologies for Known Vulnerabilities and Malware

- a. Click on the Device Status button located on the upper right corner:



- b. In the new screen select the physical ports that are to be used in the test:



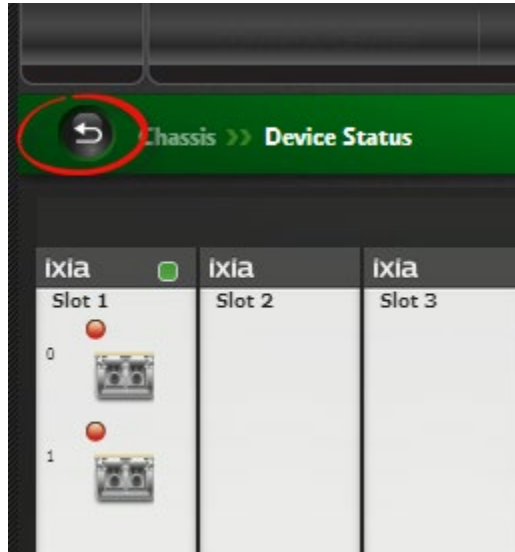
In this example, we will use ports 2 and 3 from the blade located in slot number 5.

Note: An important thing to remember when reserving your ports is the order in which you reserve them. Whenever you reserve a port, the system will automatically map that port to an interface on the chassis.

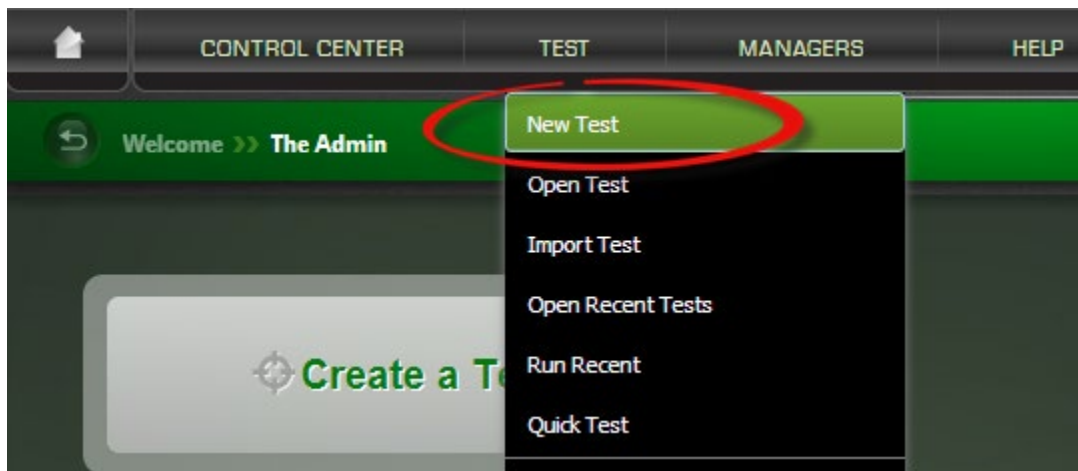
Note: The resources of each blade are allocated in proportion to the number of ports reserved on the blade. In some cases, it may be necessary to reserve additional ports to secure enough

resources for the test being performed. The sessions and bandwidth available to a test may be insufficient to adequately perform the test if too few ports have been reserved.

- c. Once the proper test ports have been selected click on the back arrow to return to the initial screen:



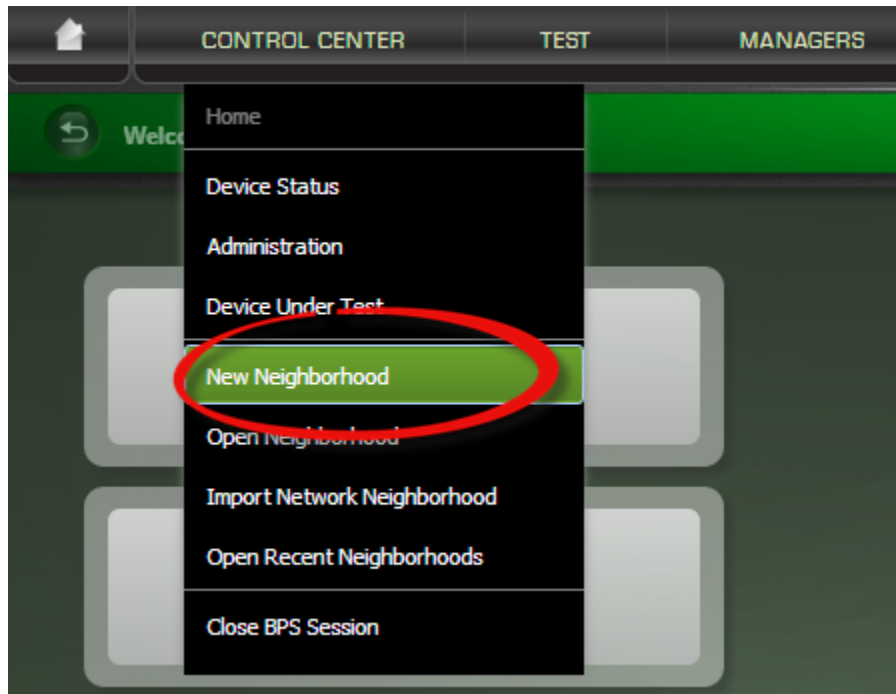
3. Next, select **Test** -> **New Test** option from the upper menu bar to start with configuring the test:



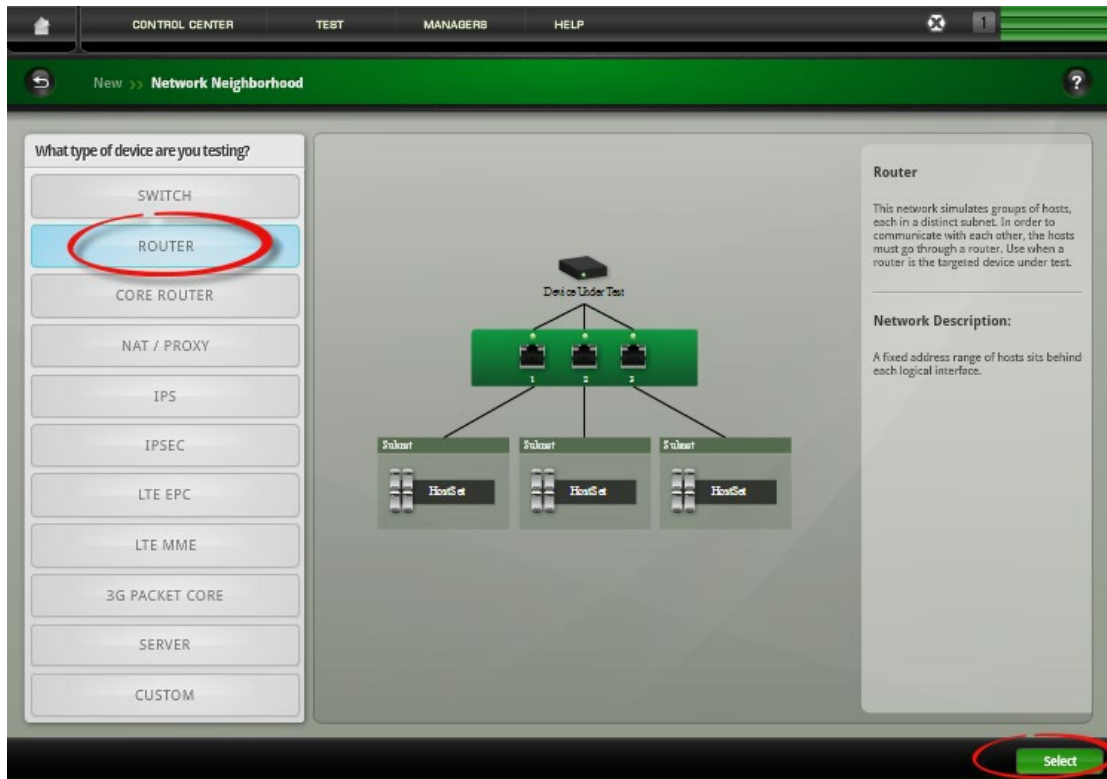
4. Since we already reserved the physical test ports (and, if applicable extra ports for more resource reservation) now we need to configure the MAC and IP layer parameters like MAC address VLANs, IP addressing scheme, MTU, etc. All these settings, among others are controlled/defined from the **Network Neighborhood**:

Test Methodologies for Known Vulnerabilities and Malware

- a. From the upper menu bar select **Control Center** -> **New Neighborhood**:

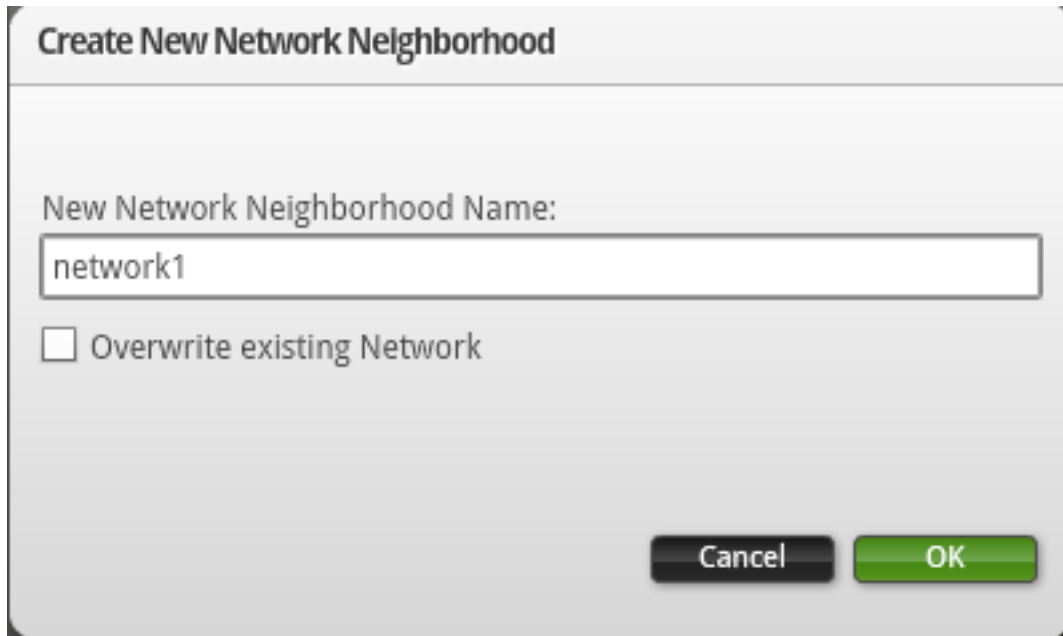


- b. Select the DUT type used for the test and a corresponding default Network Neighborhood will be created:

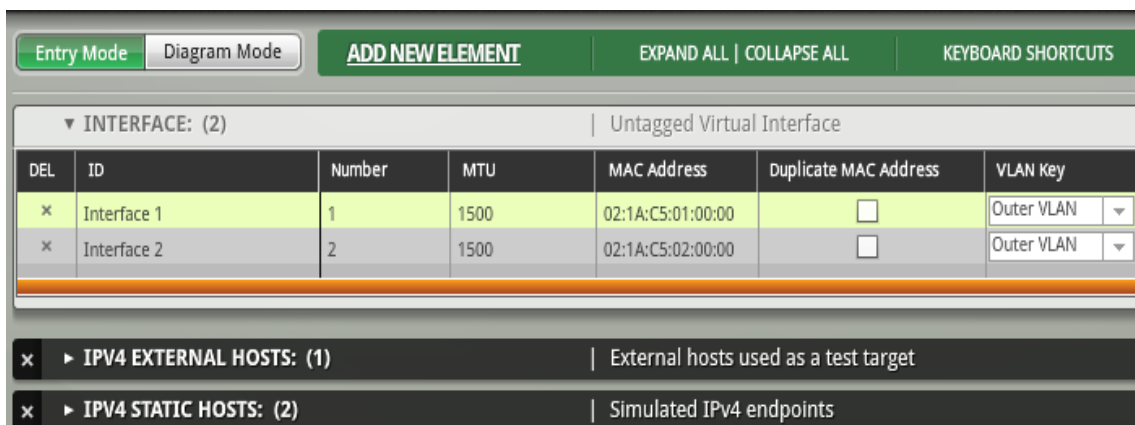


For this test, we will use *ROUTER* as DUT type.

- c. Enter an easy-to-remember name for the new Network Neighborhood and click **OK**:

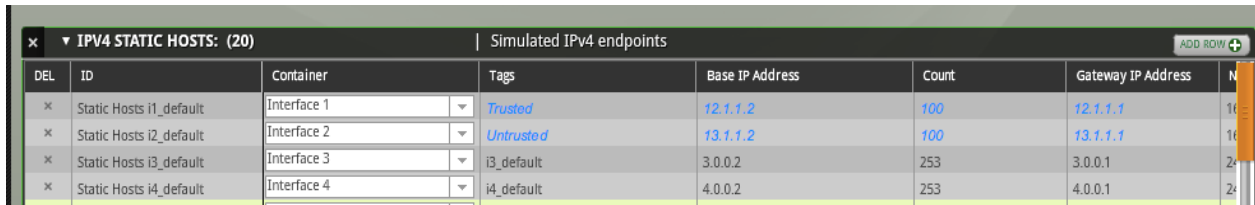


- d. According to the DUT type we previously selected, a default Network Neighborhood will be created. Since we chose *ROUTER* DUT type, the following elements are automatically created:
- i. Two INTERFACES: Provides access to interface level parameters like MTU, MAC, Impairments, and Packet Filters.
 - ii. One IPv4 EXTERNAL HOSTS: configures the IP address of the external end hosts/servers. This element is not required for IPS devices that are Pass-Through. It is only need for devices that are terminating the TCP Connection (e.g. Server Load Balancers).
 - iii. Two IPv4 STATIC HOSTS: Provides access to IP related parameters of the BreakingPoint emulated hosts. The Static Hosts represents the BreakingPoint emulated attackers and target hosts.



Test Methodologies for Known Vulnerabilities and Malware

- iv. For this test, under IPv4 static hosts we will use the following:
- Change the first entry (with ID as Static Hosts i1_default) to have a tag of “Trusted”, Base IP Address as 12.1.1.2, Count of 100 and Gateway IP address as 12.1.1.1.
 - Change the second entry (with ID as Static Hosts i2_default) to have a tag of “Untrusted”, Base IP Address as 13.1.1.2, Count of 100 and Gateway IP address as 13.1.1.1.

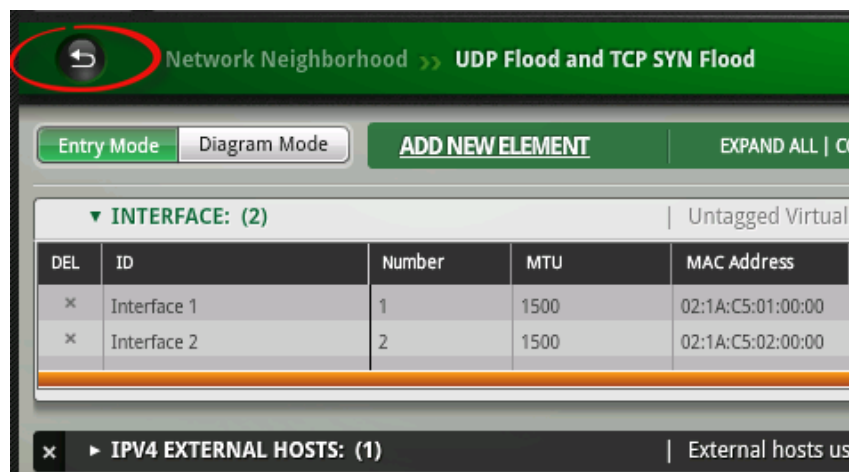


DEL	ID	Container	Tags	Base IP Address	Count	Gateway IP Address	M
x	Static Hosts i1_default	Interface 1	Trusted	12.1.1.2	100	12.1.1.1	16
x	Static Hosts i2_default	Interface 2	Untrusted	13.1.1.2	100	13.1.1.1	16
x	Static Hosts i3_default	Interface 3	i3_default	3.0.0.2	253	3.0.0.1	24
x	Static Hosts i4_default	Interface 4	i4_default	4.0.0.2	253	4.0.0.1	24

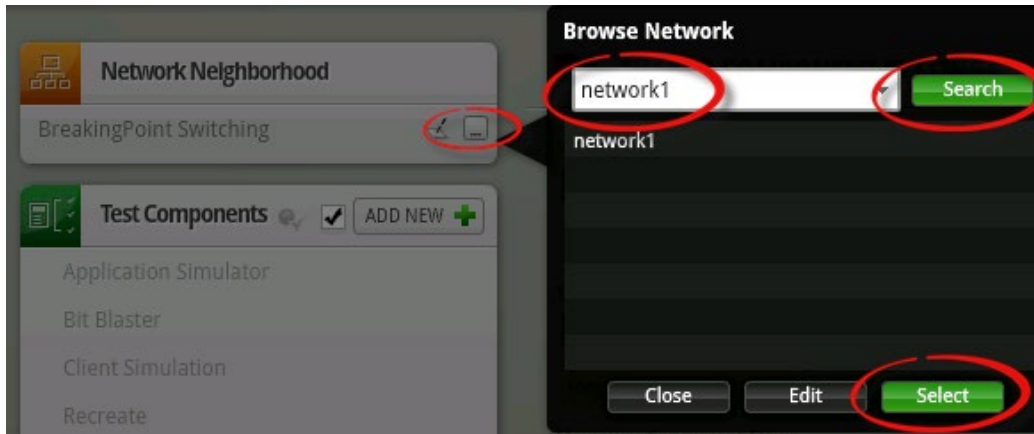
- e. Save the new Network Neighborhood configuration by clicking the **Save** button located on the lower right corner:



- f. Click on the back arrow to return to the main test screen:



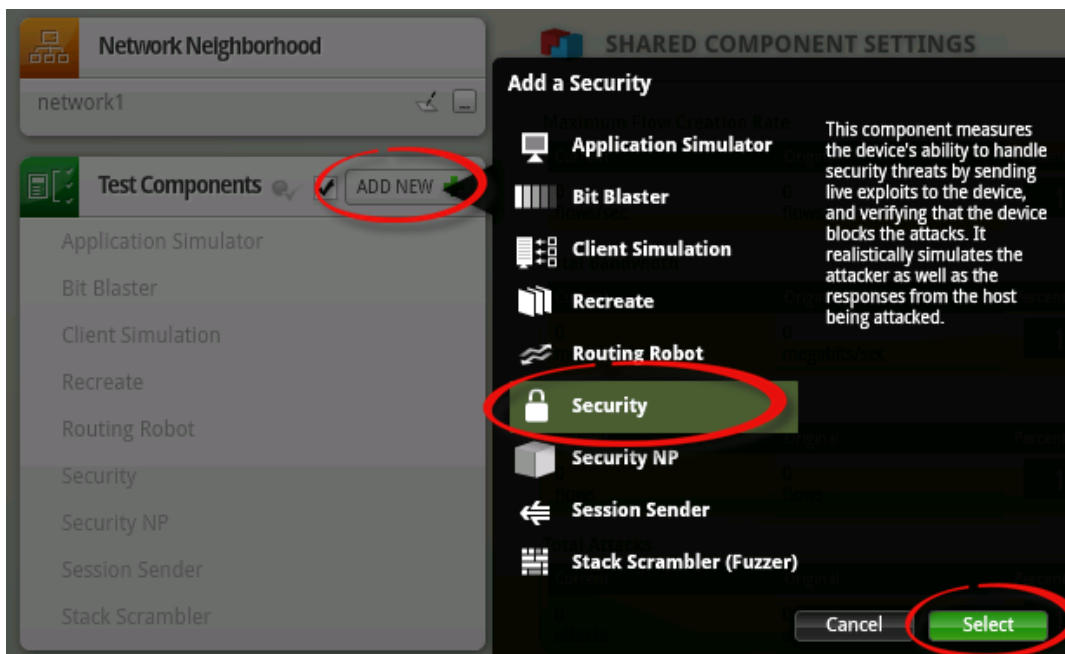
- g. From the main test screen click on the browse button from the **Network Neighborhood** section to search and select the above created Network Neighborhood:



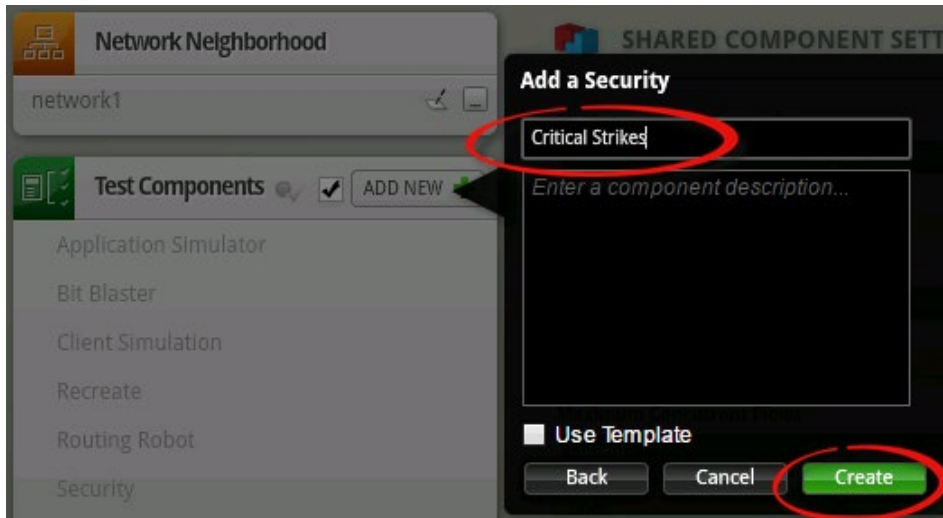
After configuring the MAC and IP layer parameters the actual traffic component needs to be created.

The Test Component is the entity that controls the traffic patterns. For this example, we will use the Security test component to emulate the actual attacks.

5. Click on the **ADD NEW** button from the **Test Components** section. Choose *Security* from the selection list and click on **Select** button:

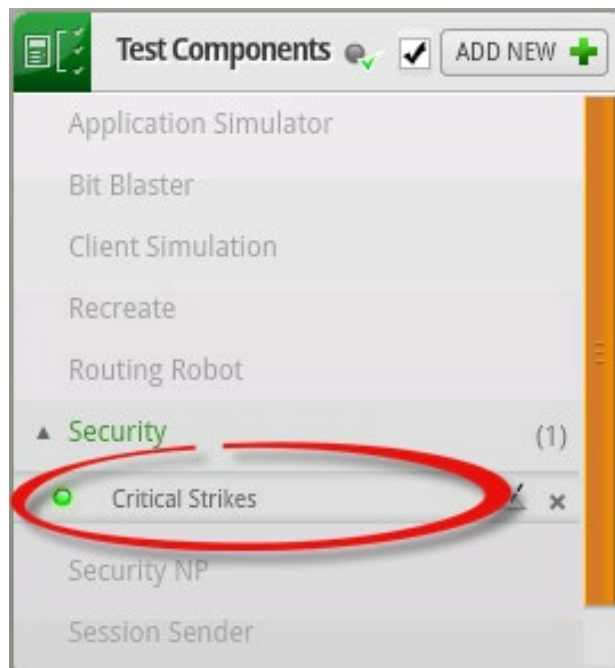


6. Rename the component from *Security_1* to something more meaningful if required and click Create button:



A new entry (i.e. *Critical Strikes*) will be created under **Security** Test Component.

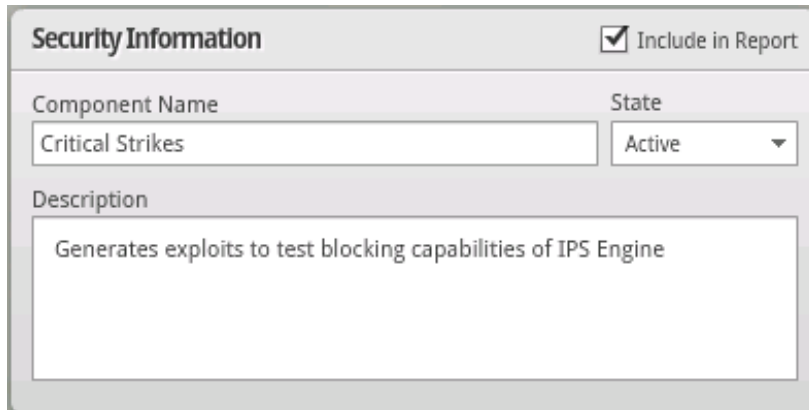
7. Click on the newly created component to edit its parameters:



8. In the next steps, we will configure the **Security** test component:

Test Methodologies for Known Vulnerabilities and Malware

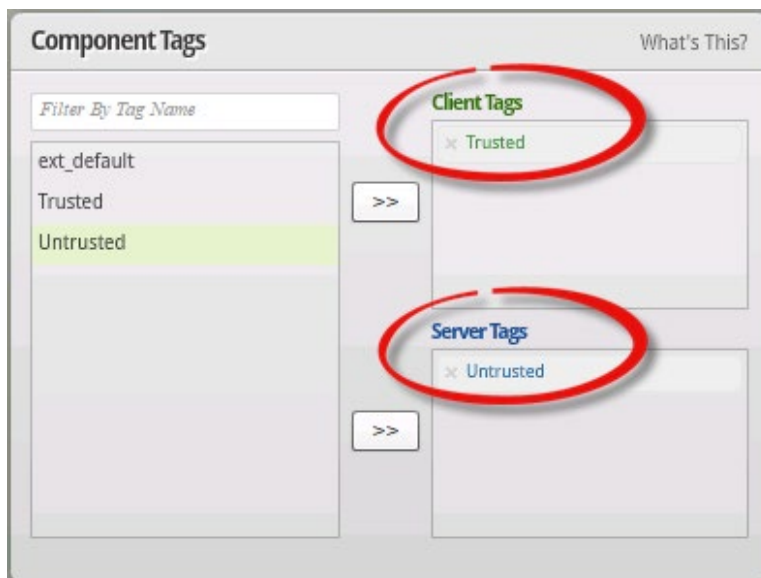
- a. A meaningful description can be added in the **Description** box for easy reference of what this particular component is configured for:



The screenshot shows a configuration window titled "Security Information" with a checked "Include in Report" option. It contains a "Component Name" field with the text "Critical Strikes" and a "State" dropdown menu set to "Active". Below these is a "Description" text area containing the text "Generates exploits to test blocking capabilities of IPS Engine".

- b. In the **Component Tags** section make sure to assign the proper interface tags. For the Client Tags, assign the tag corresponding to the IPv4 Static Host Network Neighborhood element emulating the Trusted side.

For the Server Tags, assign the tag corresponding to the IPv4 Static Host Network Neighborhood element emulating the Untrusted side:



The screenshot shows the "Component Tags" configuration window. On the left is a list of available tags: "ext_default", "Trusted", and "Untrusted", with "Untrusted" highlighted. On the right are two tag assignment areas: "Client Tags" and "Server Tags". The "Client Tags" area contains a "Trusted" tag, and the "Server Tags" area contains an "Untrusted" tag. Red circles highlight these assigned tags. A "What's This?" link is visible in the top right corner.

- c. The **Parameters** section contains multiple test component attributes that are grouped into category folders based on their functional purpose:

- i. **Maximum Attacks Per Second:** Configure the maximum attacks per second that you want to generate. For this test, set it to 10.
- ii. **Maximum Packets Per Second:** If you want to control the rate based on packets / sec instead of attacks / sec, configure it here. A value of 0 indicates it is not set. For

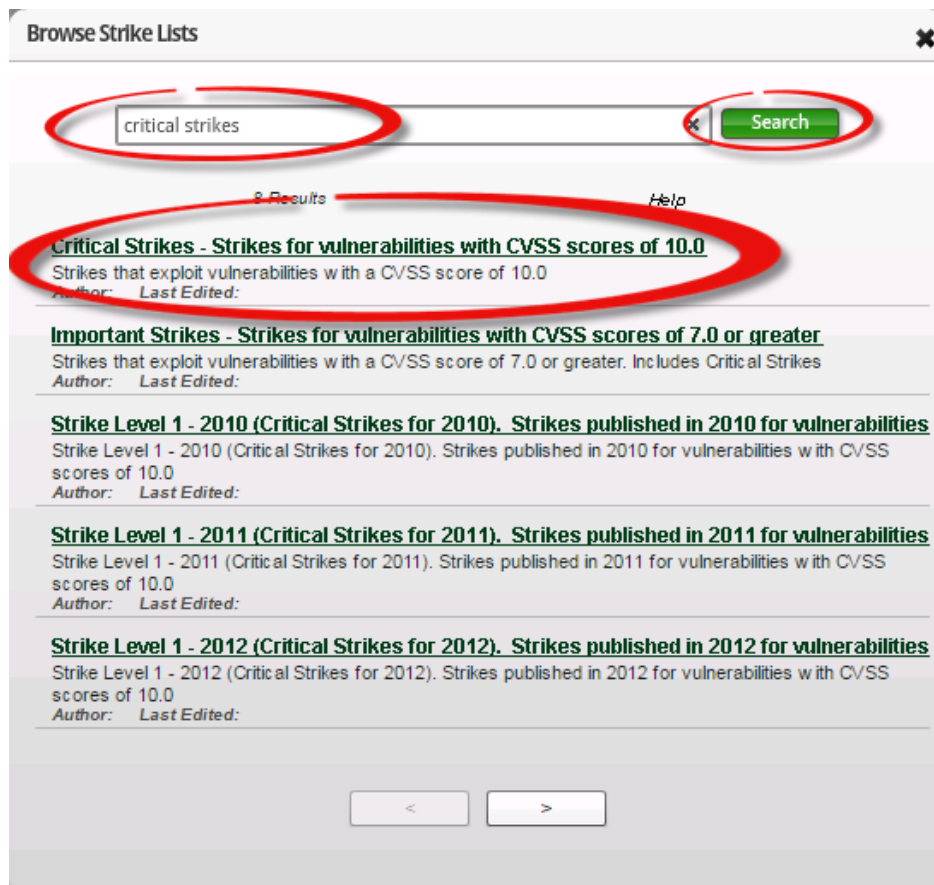
Test Methodologies for Known Vulnerabilities and Malware

this test, we will use attacks / sec instead of packets / sec so leave it at default value of 0.

- iii. **Attack Retries:** The number of times a particular action within a strike will be retried before considered blocked by the device. We'll use 3 for this test.
- iv. **Random Seed:** This controls "how" random data will be generated by the Strikes. Leave this at default value of 0 indicating that the seed will itself be randomly generated.
- v. **Delay Start:** If there needs to be a delay before the attacks are generated, configure the value here. Leave it at default value of 0.
- vi. **Strike List:** The strike list defines the list of attacks the test will generate. For this test, we will use the *critical strikes* (Strikes that exploit vulnerabilities with a CVSS score of 10.0). To do this, first select the Browse button.

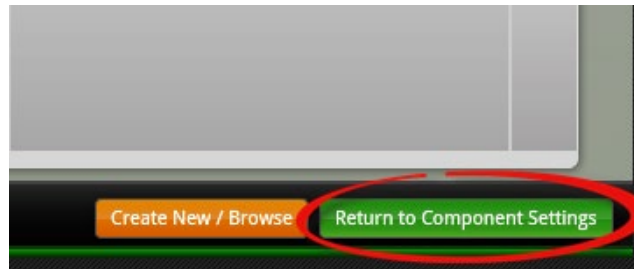


Following that, search for "critical strikes" and select the first result (*Critical Strikes*).

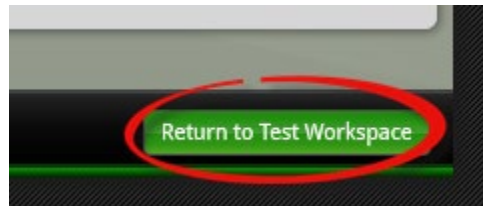


Test Methodologies for Known Vulnerabilities and Malware

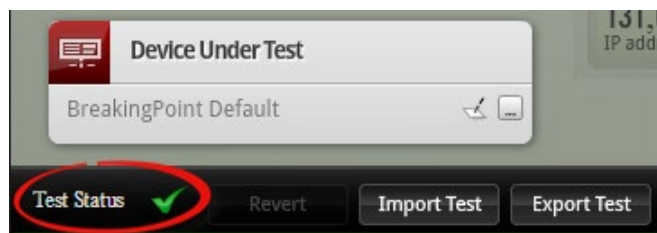
- vii. **Strike List Iterations:** This parameter controls how many iterations of the strike list the test should do. For this test, leave this at Default value of 1.
 - viii. **Strike List Iteration Delay:** This parameter configures the delay between iterations.
 - ix. **Evasion Profile:** This configures the different evasion techniques to apply on the exploit traffic. For this test, we will not configure any evasion profile and leave it at Default.
- d. Click on the **Return to Component Setting** button to go back to the Test Component configuration screen:



- e. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:



9. Make sure the **Test Status** indicated (on the lower left corner) has a green checkmark:



If there is not, determine what is wrong by selecting Test Status and viewing the errors.

10. Select **Save and Run** from the lower right corner:



11. If the test has not previously been saved, enter a name for the test and click **Save**



Run the test and while the test is running continue to monitor the DUT with respect to the exploits that are being detected. See the **Results Analysis** section for important statistics and diagnostics information.

Results Analysis

This section covers the key statistics and events that BreakingPoint provides for this type of test.

Real-Time Stats

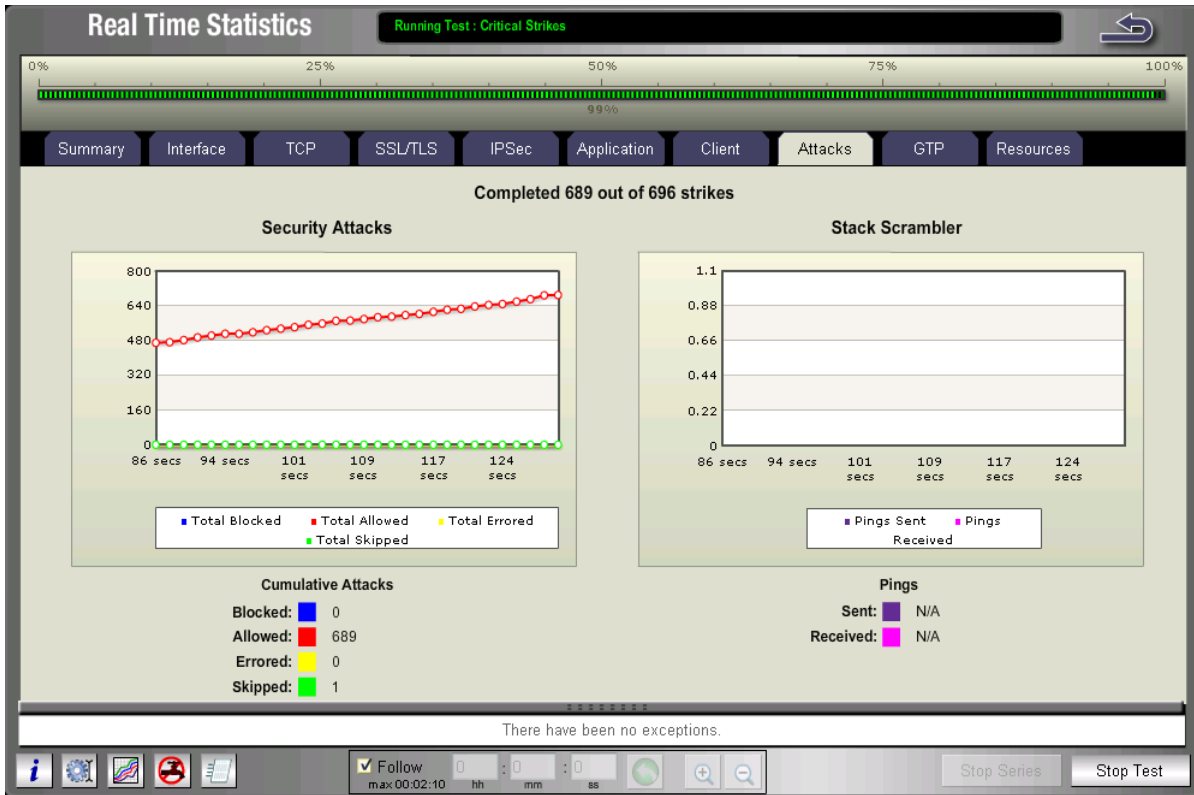
After the test is started and initialized, the screen will automatically switch to Real Time Statistics window.

The **Attacks** tab shows how many attacks will be run and how many of those have been completed thus far. At the bottom of this page are more detailed cumulative statistics, the most relevant being:

- **Blocked:** Of those attacks completed, how many have been blocked by the device
- **Allowed:** Of those attacks completed, how many have been allowed by the device

Also, some of the strikes can be *Skipped* which means that some of the attacks configured in the strike list are deprecated and therefore they will be skipped. Users that prefer to run them can create an evasion profile that will have *Allow Deprecated* option set. Skipped also will occur when there is an IPv6 network neighborhood configured to run a Strike that requires IPv4, and vice versa.

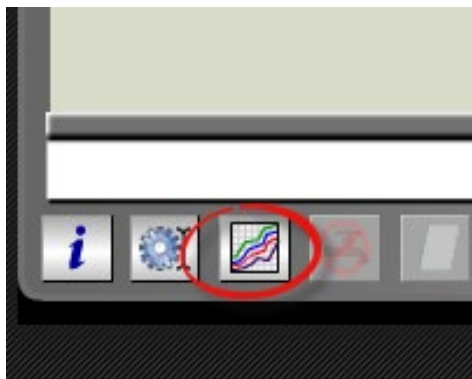
Test Methodologies for Known Vulnerabilities and Malware



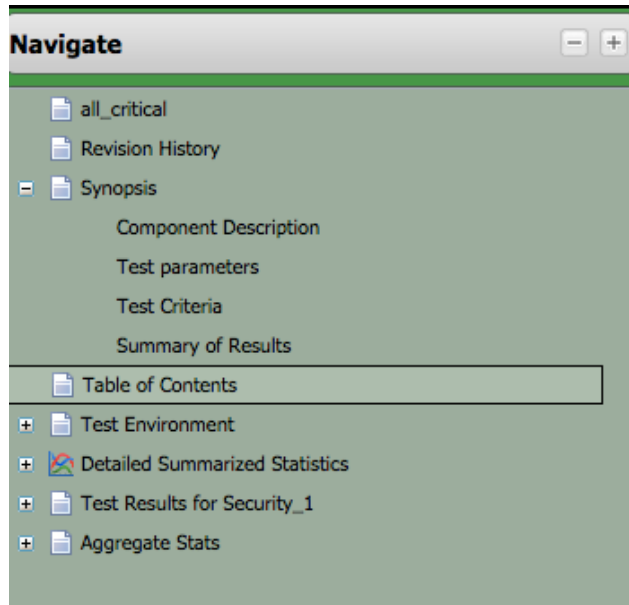
The test will stop once all the strikes have been completed.

Report

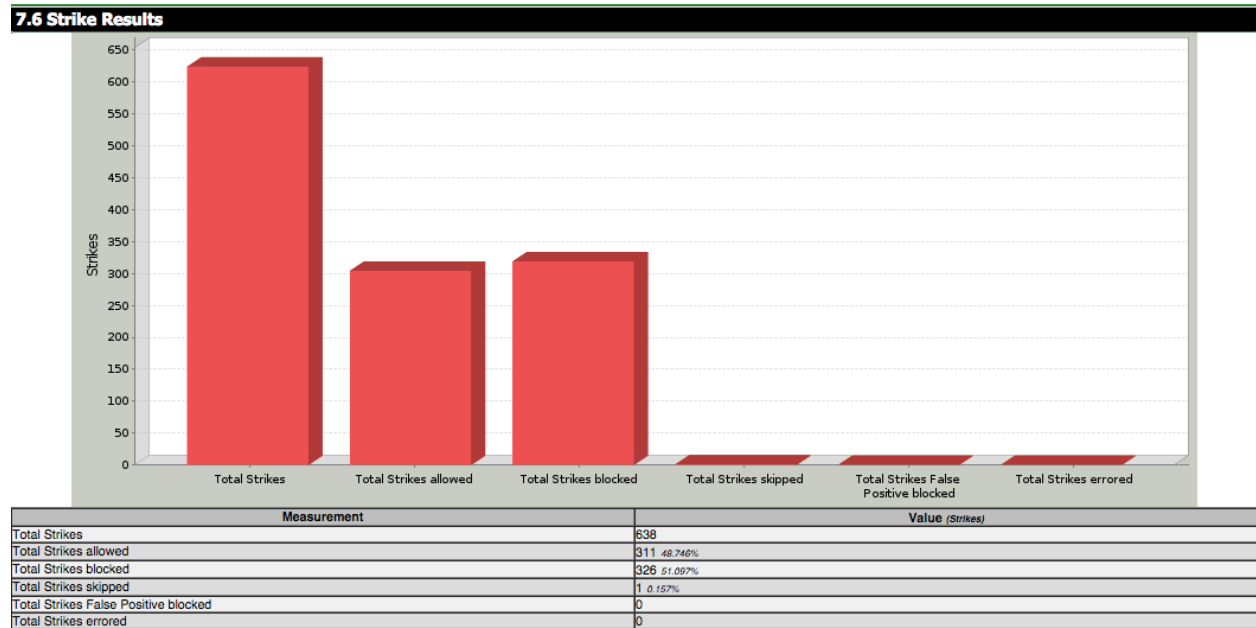
A detailed report can be generated selecting the graph button from the lower left corner of the Real Time Statistics window. This will open the results in a new browser window.



Once a test report is generated, click on the table of contents from the left pane



Navigate to Section 7.6 (Test Results for *Critical Strikes* -> *Strike Results*) to view the strike Results that shows the percentage of strikes that were allowed and blocked.



Section 7.7.1 in the below report (Strike Category Assessment) has the distribution of allowed and blocked by category

Test Methodologies for Known Vulnerabilities and Malware

7.7.1 Strike Category Assessment					
Strike Category	Strikes Blocked	Strikes Passed	Strikes Errored	Strikes Skipped	Strikes False Positive Blocked
			<i>Strikes</i>		
Denial of Service: SNMP	1	0	0	0	0
Exploits: RTSP	1	0	0	0	0
Exploits: SOCKS	1	0	0	0	0
Exploits: SSL	1	0	0	0	0
Exploits: Apache	1	0	0	0	0
Exploits: VNC	1	0	0	0	0
Exploits: NNTTP	1	0	0	0	0
Exploits: RADIUS	1	0	0	0	0
Exploits: ICMP	0	0	0	1	0
Exploits: WINS	1	0	0	0	0
Exploits: Miscellaneous	100	62	0	0	0
Generic: IXIA	50	39	0	0	0
Exploits: Web Application PHP Include	0	27	0	0	0
Exploits: Browser	32	24	0	0	0
Exploits: Webserver	5	19	0	0	0
Exploits: Telnet	2	15	0	0	0
Exploits: Web Application Command Execution	12	15	0	0	0
Exploits: SCADA	11	10	0	0	0
Denial of Service: Miscellaneous	12	10	0	0	0
Exploits: Clientside	11	9	0	0	0
Exploits: Clientside Microsoft Office	0	8	0	0	0
Exploits: Web Application Malformed Request Headers	2	7	0	0	0
Exploits: Backup Appliance	4	7	0	0	0
Exploits: SMB	21	6	0	0	0
Exploits: Web Application Directory Traversal	0	4	0	0	0
Denial of Service: Browser	2	3	0	0	0
Exploits: PHP	1	3	0	0	0
Exploits: Oracle	1	3	0	0	0
Exploits: Web Application File Upload	5	3	0	0	0
Exploits: Browser FTP	0	3	0	0	0
Exploits: Microsoft IIS	3	3	0	0	0
Exploits: SunRPC	1	2	0	0	0
Exploits: SMTP	5	2	0	0	0
Exploits: LPD	0	2	0	0	0
Exploits: POP3	0	2	0	0	0
Exploits: DCERPC	11	2	0	0	0
Backdoors: All	1	2	0	0	0
Exploits: Unix r Service	0	2	0	0	0
Exploits: SSH	0	1	0	0	0
Exploits: IDS	1	1	0	0	0
Exploits: Clientside Microsoft	0	1	0	0	0
Exploits: Web Application Information Disclosure	0	1	0	0	0

To understand better which strikes were allowed, click on section 7.7.3 (Allowed Strike List), which will provide details of the category and the strike that was allowed by the IPS device.

Test Methodologies for Known Vulnerabilities and Malware

7.7.3 Allowed Strike List		
Timestamp	Strike Category	Strike Name
37.9865685	Exploits: Miscellaneous	HP OpenView Network Node Manager parameter overflow
4.19061565	Exploits: Backup Appliance	Veritas Netbackup bjava-msvc Format String Attack (OS X)
4.19164419	Generic: IXIA	Castle Rock Computing SNMPc Network Manager Community String Stack Buffer Overflow attack
4.19412518	Exploits: Telnet	Solaris in.telnetd TTYPROMPT Buffer Overflow (uucp)
4.19502068	Exploits: Web Application PHP Include	Headline Portal Engine page.dmoz.show.php3 HPEInc Parameter PHP File Include Variant 2
4.19698763	Exploits: SCADA	AzeoTech DAQFactory Stack Buffer Overflow
4.19941044	Exploits: SMTP	Sendmail headers.c Address Field Overflow
4.88184547	Exploits: Web Application PHP Include	Headline Portal Engine news.xmlphp.php3 HPEInc Parameter PHP File Include Variant 3
4.88660431	Exploits: SCADA	3S Smart Software Solutions CoDeSys Gateway Server Filename Directory Traversal
4.89833498	Exploits: Miscellaneous	HP OpenView nnmRptConfig.exe schd_select1 memory corruption
5.34999609	Denial of Service: Browser	Internet Explorer AxDebugger.Document DoS
5.36084652	Exploits: Browser	Microsoft Internet Explorer Install Engine SetCffFile Overflow
5.80781002	Generic: IXIA	Microsoft Internet Explorer HTML Decoding Memory Corruption attack
5.91965055	Exploits: Browser	Microsoft IE VML Object Handling Use-After-Free Vulnerability
5.9228878	Exploits: Web Application Cookie	HP OpenView Network Node Manager AcceptLang Buffer Overflow
6.08971071	Exploits: SMB	Microsoft Server Service NetpwPathCanonicalize Overflow (Generic)
6.27733469	Exploits: Web Application Directory Traversal	Rocket Servergraph Admin Center fileRequest Arbitrary File Creation
6.90234184	Exploits: Web Application Malformed Request Headers	HP OpenView Network Node Manager ovalarm.exe CGI Buffer Overflow
6.97220087	Exploits: Miscellaneous	Fujitsu Systemcast Wizard PXE buffer overflow
7.09237862	Exploits: Miscellaneous	WANem v2.3 Unauthorized Remote Root Access
7.2126298	Backdoors: All	ManageEngine Desktop Central DCPluginServlet addPluginUser Policy Bypass
7.31677437	Exploits: SCADA	RealFlex RealWin SCADA SPCP_INITIALIZE and SPCP_INITIALIZE RF Buffer Overflow
8.07975578	Exploits: Miscellaneous	HP Data Protector Backup exec setup Command Execution
8.08254433	Exploits: Telnet	Solaris in.telnetd TTYPROMPT Buffer Overflow (random username)
8.31361198	Exploits: Telnet	Solaris in.telnetd TTYPROMPT Buffer Overflow (root)
8.51102352	Exploits: ClientSide Microsoft Office	Microsoft Powerpoint 2003 Heap Overflow (POP3)
8.98993683	Denial of Service: FTP	Microsoft IIS 7.5 FTPSVC Telnet IAC DoS
9.12600708	Exploits: Miscellaneous	DATAC Control RealWin SCADA System Packet Handling Buffer Overflow
9.12893772	Exploits: Miscellaneous	HP Data Protector Backup Command Execution
9.29023457	Generic: IXIA	Ipswitch IMail IMAP STATUS Command Buffer Overflow attack
9.77002716	Generic: IXIA	Microsoft WordPad Font Conversion Buffer Overflow attack
10.5203934	Exploits: Miscellaneous	HP Operations Agent coda.exe Stack Buffer Overflow
18.5614758	Exploits: Miscellaneous	D-Link DAP-1160 Authentication Bypass
18.564436	Denial of Service: Miscellaneous	Apple OS X QuickDraw GetSrcBits32ARGB Memory Corruption Denial of Service (POP3)
19.7811985	Exploits: Telnet	Solaris in.telnetd -fuser Authentication Bypass (root)
22.5136299	Exploits: Unix r Service	UNIX rlogind Service -froot Authentication Bypass
23.0963497	Generic: IXIA	Adobe Acrobat Reader eBook Format String Vulnerability attack
23.1023903	Exploits: Miscellaneous	HP LoadRunner directory disclosure
23.7298679	Generic: IXIA	CA License Software PUTOLF Buffer Overflow attack
23.8248768	Exploits: Browser	Firefox 3.6.16 Object mChannel Use After Free
25.1270657	Exploits: ClientSide	Adobe Reader/Acrobat ToolButton Object Use-after-free
25.4723606	Exploits: Web Application PHP Include	Headline Portal Engine news.xmlbi.php3 HPEInc Parameter PHP File Include Variant 1
27.3682194	Exploits: Webserver	HP Universal CMDB JMX Console Authentication Bypass
27.903677	Exploits: DCERPC	Trend Micro ServerProtect CMON_NetTestConnection Overflow
28.4086533	Exploits: Miscellaneous	CA RightStor Discovery Service Overflow

Clicking on any name, will show more details regarding the strike like the CVE number associated with it

Time of strike	Strike Name	Strike Result	Strike Reference
4.898335	HP OpenView nnmRptConfig.exe schd_select1 memory corruption-16	Allowed	CVE 2011-0269 BID 45762 CVSS-Critical
6.972201	Fujitsu Systemcast Wizard PXE buffer overflow-43	Allowed	CVE 2009-0270 BID 33342 OSVDB 51486 CVSS-Critical
7.0923786	WANem v2.3 Unauthorized Remote Root Access-44	Allowed	BID 55485 OSVDB 85345 http://itsecuritysolutions.org/2012-08-12-WANem-v2.3-multiple-vulnera http://www.exploit-db.com/exploits/21190/ CVSS-Critical
8.079756	HP Data Protector Backup exec setup Command Execution-48	Allowed	CVE 2011-0922 BID 46234 OSVDB 81759 CVSS-Critical
9.126007	DATAC Control RealWin SCADA System Packet Handling Buffer Overflow-55	Allowed	CVE 2008-4322 BID 31418 CVSS-Critical
9.128938	HP Data Protector Backup Command Execution-53	Allowed	CVE 2011-0923 BID 46234 OSVDB 81759 CVSS-Critical
10.520393	HP Operations Agent coda.exe Stack Buffer Overflow-60	Allowed	CVE 2012-2020 BID 54362 OSVDB 93874 CVSS-Critical
16.99458	HP Data Protector Cell Request Service Opcode Z11 Buffer Overflow-2	Blocked - client	CVE 2013-2333 BID 60309 OSVDB 93867 CVSS-Critical
18.561476	D-Link DAP-1160 Authentication Bypass-64	Allowed	OSVDB 66164 http://www.lcysilence.org/wp-content/uploads/IS-2010-005_D-Link_DAP-1160_Authentication_Bypass.pdf CVSS-Critical
20.47425	HP Data Protector Cell Request Service Opcode 264 Buffer Overflow-29	Blocked - client	CVE 2013-2327 BID 60302 OSVDB 93861 CVSS-Critical
21.920315	D-Link HNPAP HTTP POST Content Stack Buffer Overflow-5	Blocked - client	CVE 2014-3936 BID 67651 OSVDB 107049 http://www.devty0.com/2014/05/hacking-the-d-link-dsp-w215-smart-p CVSS-Critical
22.383524	HP OpenView Network Node Manager memory Corruption-9	Blocked - client	CVE 2010-1550 CVSS-Critical
23.10239	HP LoadRunner directory disclosure-74	Allowed	CVE 2013-4798 BID 61443 OSVDB 95642 CVSS-Critical

Test Methodologies for Known Vulnerabilities and Malware

All these statistics should be compared with the logs from the IPS device to make sure that the blocked strikes were properly identified and classified and no other traffic drops of other nature (other than properly identifying the strikes as exploits) caused the strike to get blocked.

Additionally, using all this information the IPS configuration can be re-visited to understand why the allowed strikes were permitted to pass through even though they were critical.

Test Variables

Use the following test configuration parameters to repeat the test.

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
IP Version	IPv4	IPv6, IPsec, DSLite, 6rd, selected Mobility stacks, etc.
Concurrent Strikes	Default	100
Strike List	Critical Strikes	User Defined
Benign Traffic	None	Add benign traffic using AppSim test component to stress the IPS. The traffic should be a mix that matches the end customer deployment production network. The objective of such a new test should be to find the maximum performance of the device with same level of detection accuracy reached in the initial test.
Evasion Techniques	Disabled	IP Fragmentation, Application Evasion Profiles, etc. Run test with attacks that have been previously blocked/detected by the IPS to validate its resilience to such evasions.

Conclusions

This configuration covered the main parameters of the security component in BreakingPoint using a practical example allowing the user to baseline the security effectiveness of an Intrusion Prevention System.

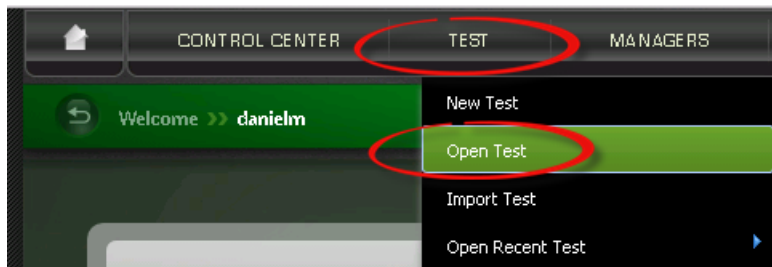
Test Case: Measuring Security Effectiveness of Intrusion Prevention Engine with Evasions

Objective

This test is a variant of the previous test where the same exploits are sent with evasion enabled. Evasion techniques can be used to bypass traditional IPS prevention mechanisms used to block the exploits. In this test, we will use the TCP Split Handshake evasion. The TCP Split Handshake evasion technique is designed to confuse state machines, allowing vulnerability exploits or similar threats to bypass detection from network-based security devices. The evasion technique works by modifying the standard TCP 3-way handshake in a way that can confuse a decoder state machine. The connection starts with the SYN from client to server. The server then sends two packets, one with a SYN bit set, the other with an ACK bit set. The client will then respond with a SYN/ACK packet instead of the server. Typically, IPSs will look at the SYN and ACK bits to determine the direction of the session and apply the security policies accordingly (for example client to server exploits will be checked for in the direction of client to server, not vice versa), but the split handshake can confuse the IPS and thus have it lose session state and let the traffic flow, thus enforcing an incorrect security control. Thus, exploits that were blocked by the IPS before will be allowed to go through.

Step-by-Step Instructions

1. Open the test configured in the previous test case by choosing **Test** -> **Open Test** from the upper menu bar:

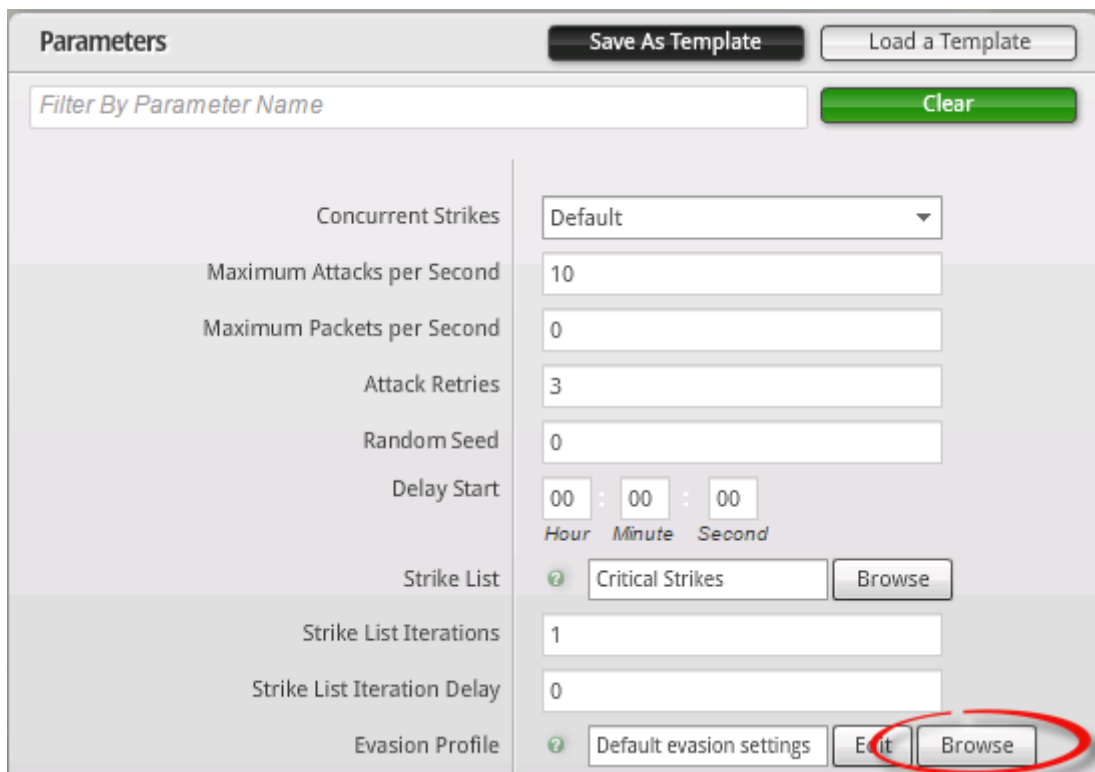


Test Methodologies for Known Vulnerabilities and Malware

2. In the new window enter the name under which the previous test was saved and click on the Search button:

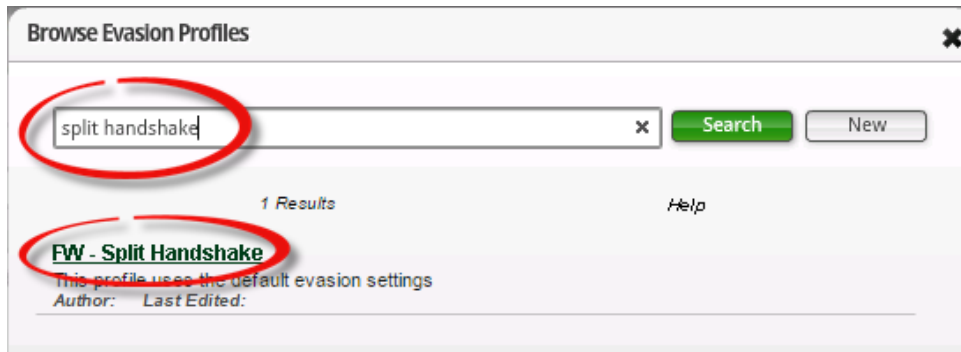


3. Select on the corresponding test name and once the test configuration loads, click on the security component to open its settings.
4. Click on the **Browse** button for the evasion profile

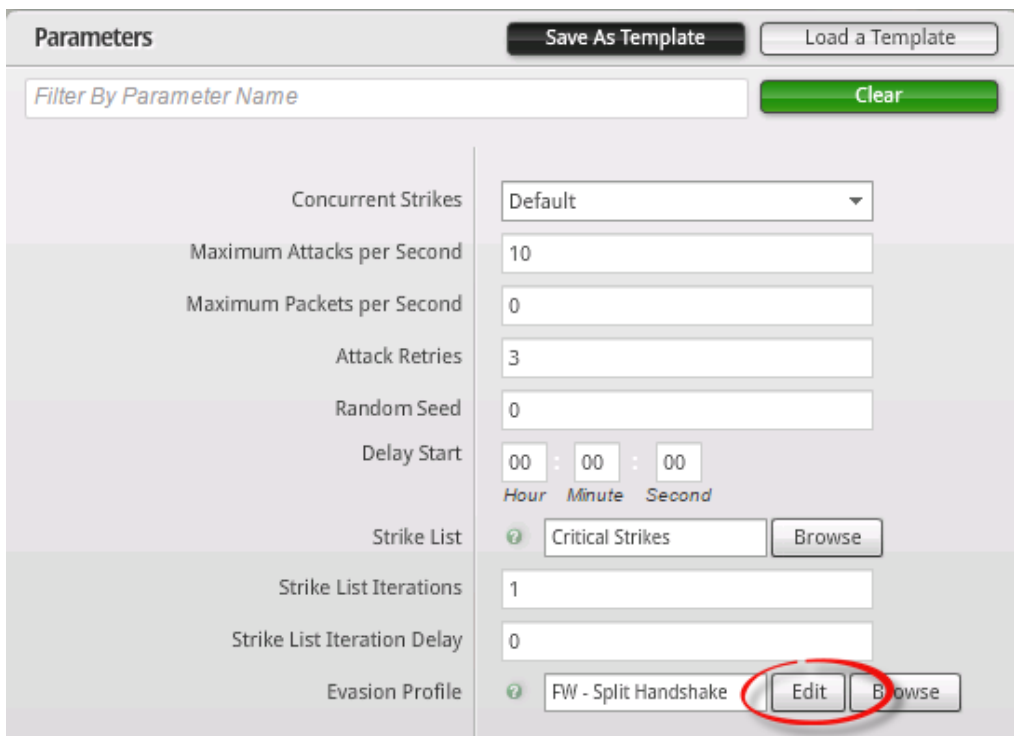


Test Methodologies for Known Vulnerabilities and Malware

5. Search for Split handshake and select the evasion profile

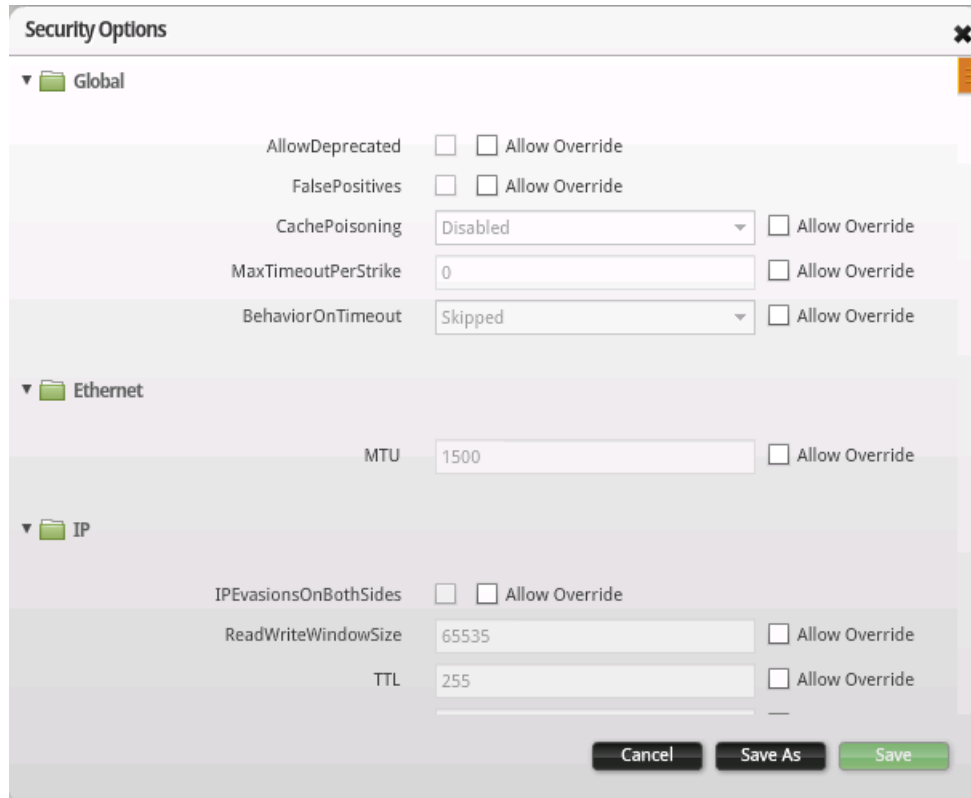


6. To see the actual settings of a particular evasion profile, click on the **Edit** button once the relevant evasion profile has been selected:



Test Methodologies for Known Vulnerabilities and Malware

7. A new window with all the security options for that particular evasion profile will be displayed:

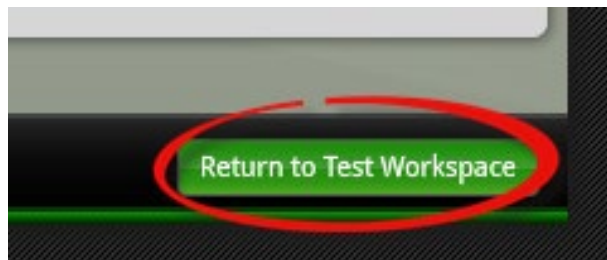


Various security options and parameters can be configured to create custom evasion profiles.

8. Click on **Cancel** button to return to the security component settings window:



9. Click on the **Return to Test Workspace** button to return to the main test screen:



10. Click on **Save and Run** as before to run the test

Results Analysis

The real time and report results for this test are similar to the previous test, so refer to the results section of the previous test for the details. The important observation to make here is whether the number of allowed strikes remain the same or **increases** when evasion is enabled. If the total number of allowed strikes increases, it means that the IPS is not effective against protection when this evasion is enabled. Further, by looking at section 7.7.3 of the report (allowed strikes section) and comparing to the previous test that was run without evasions, the strikes that were blocked before but were allowed when evasion was enabled can be identified.

Test Variables

This test used an evasion technique at the TCP layer. BreakingPoint supports multiple different evasion techniques, and more than one can be applied simultaneously to create more sophisticated evasions.

Test Case: Measuring the Security Effectiveness of Anti-Virus Engine

Overview

Network based Anti-Virus (AV) or Anti-Malware systems play an essential role in any enterprise security solutions. These systems scan incoming traffic for known malware signatures and actively block them preventing them from infecting end user machines inside the enterprise. This test determines the security effectiveness of a network-based antivirus against various types of malware. Breaking Point System includes various strikes that are known malware. Further, Ixia's Application and Threat Intelligence (ATI) program releases new malware samples every month to keep customers up to date with the latest malware.

Objective

This test measures the security effectiveness of network-based anti-malware in blocking various malware strikes. This test uses the predefined “canned malware” in BreakingPoint as an example.

Setup

The setup requires at least two test ports – one acting as an initiator and the other as a responder.

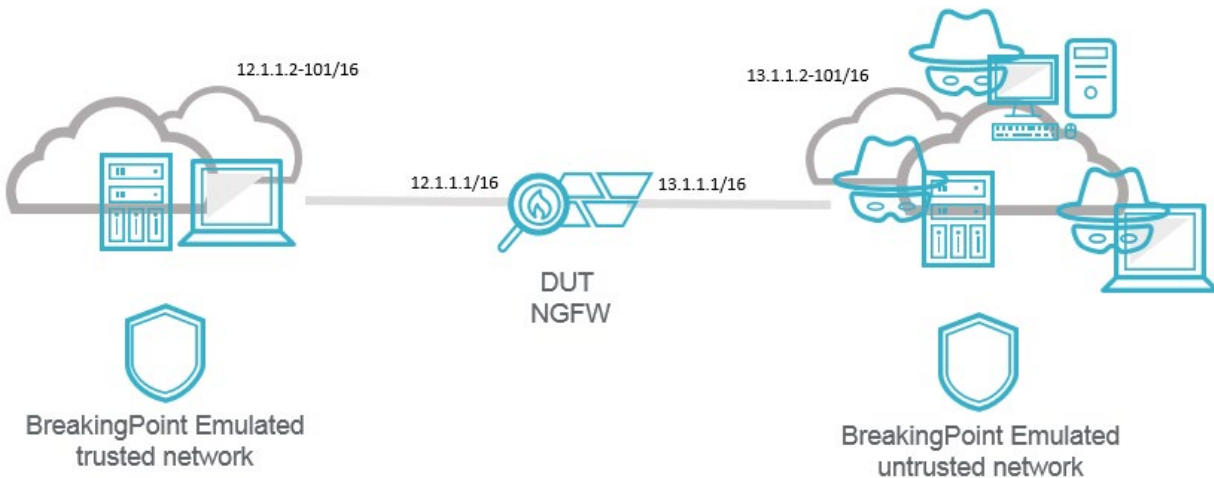
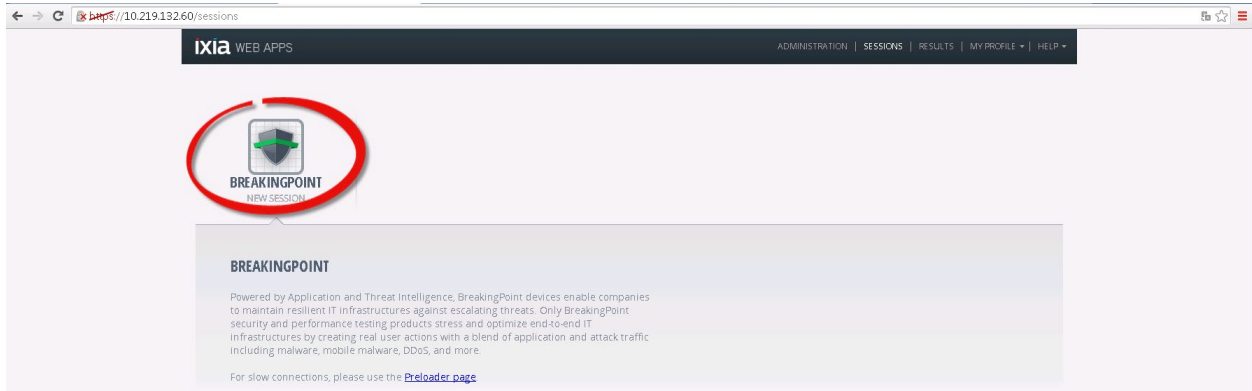


Figure 15. Test Setup

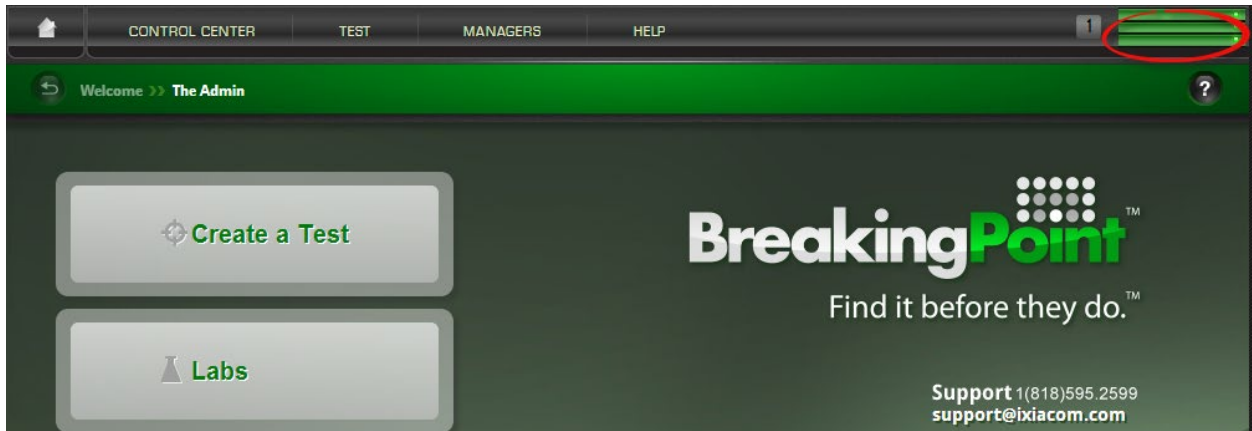
Configure the policy of the device to allow both inbound and outbound communication for any traffic/protocol on any port (allow ANY to ANY). Enable the antivirus engine to provide the maximum protection against various types of malware.

Step-by-Step Instructions

1. Use a web browser connected to the Ixia Web Apps GUI and start a new BreakingPoint Web Session:

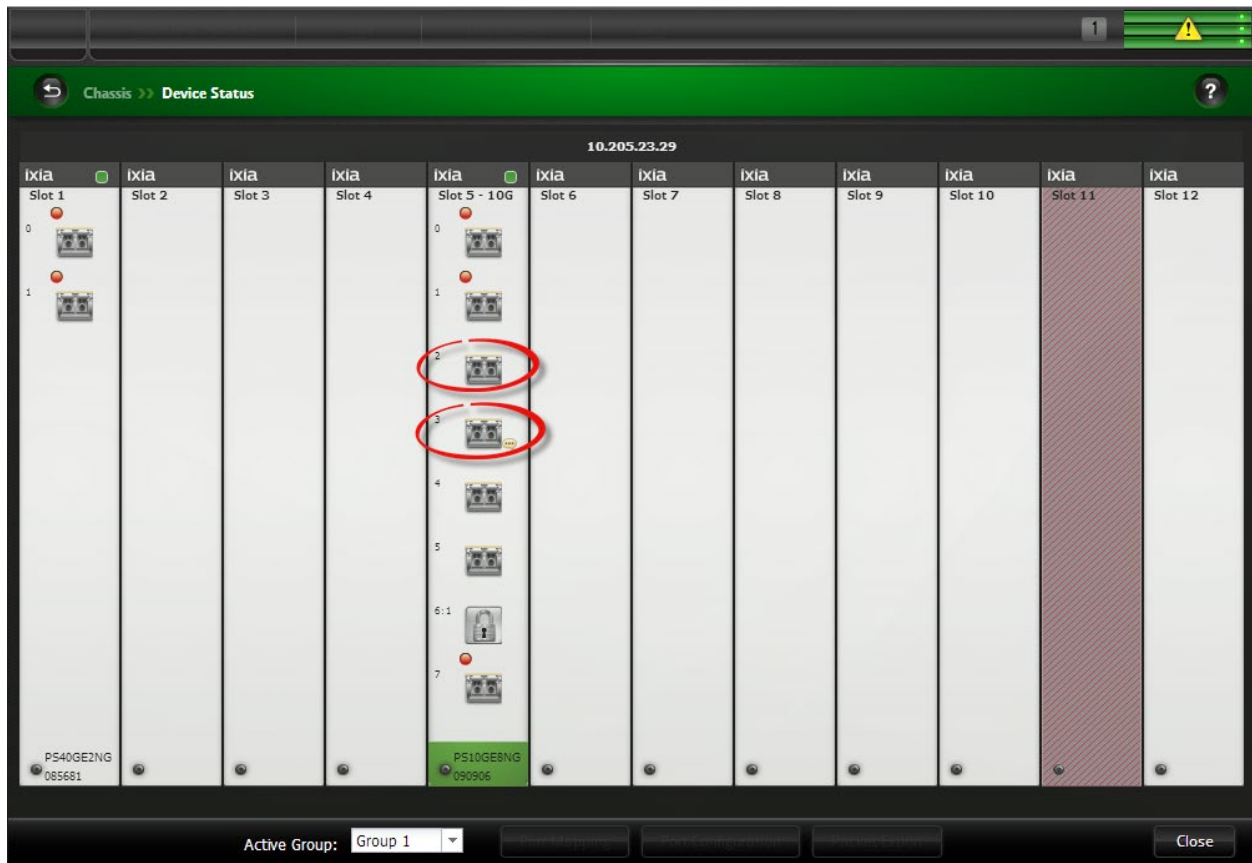


2. Once the BreakingPoint web home page loads, reserve the test ports to be used in the physical test setup to generate/receive traffic:
 - a. Click on the Device Status button located on the upper right corner:



Test Methodologies for Known Vulnerabilities and Malware

b. In the new screen select the physical ports that are to be used in the test:

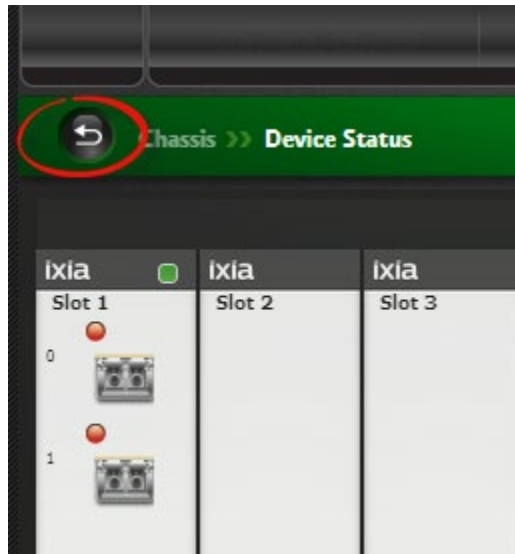


In this example, we will use ports 2 and 3 from the blade located in slot number 5.

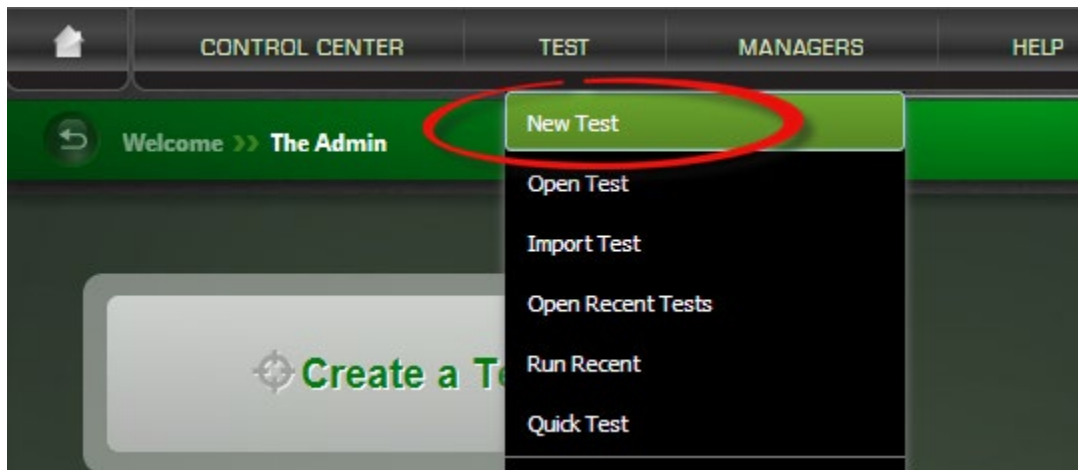
Note: An important thing to remember when reserving your ports is the order in which you reserve them. Whenever you reserve a port, the system will automatically map that port to an interface on the chassis.

Note: The resources of each blade are allocated in proportion to the number of ports reserved on the blade. In some cases, it may be necessary to reserve additional ports to secure enough resources for the test being performed. The sessions and bandwidth available to a test may be insufficient to adequately perform the test if too few ports have been reserved.

Once the proper test ports have been selected click on the back arrow to return to the initial screen:



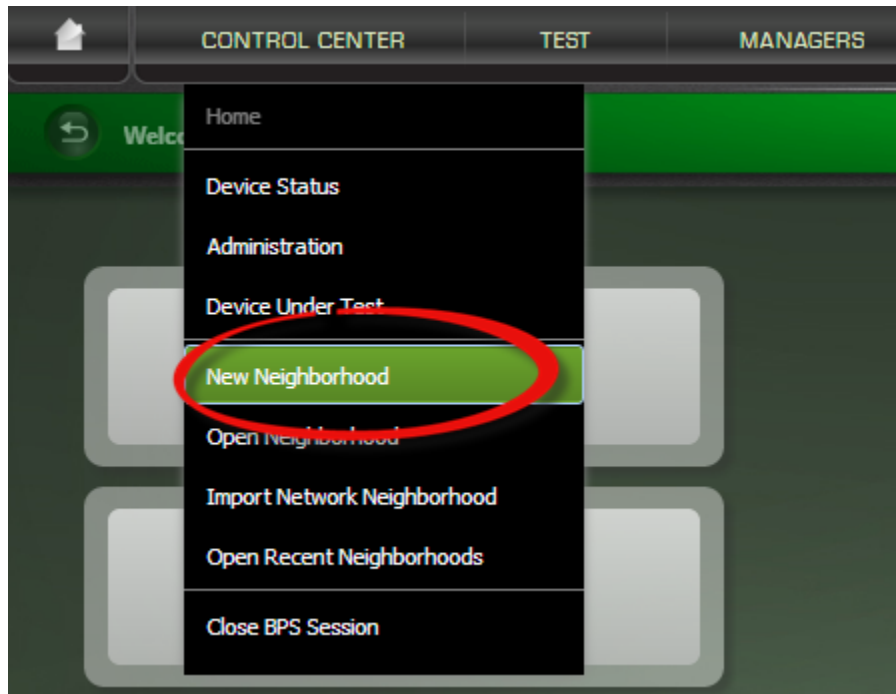
3. Next, select **Test** -> **New Test** option from the upper menu bar to start with configuring the test:



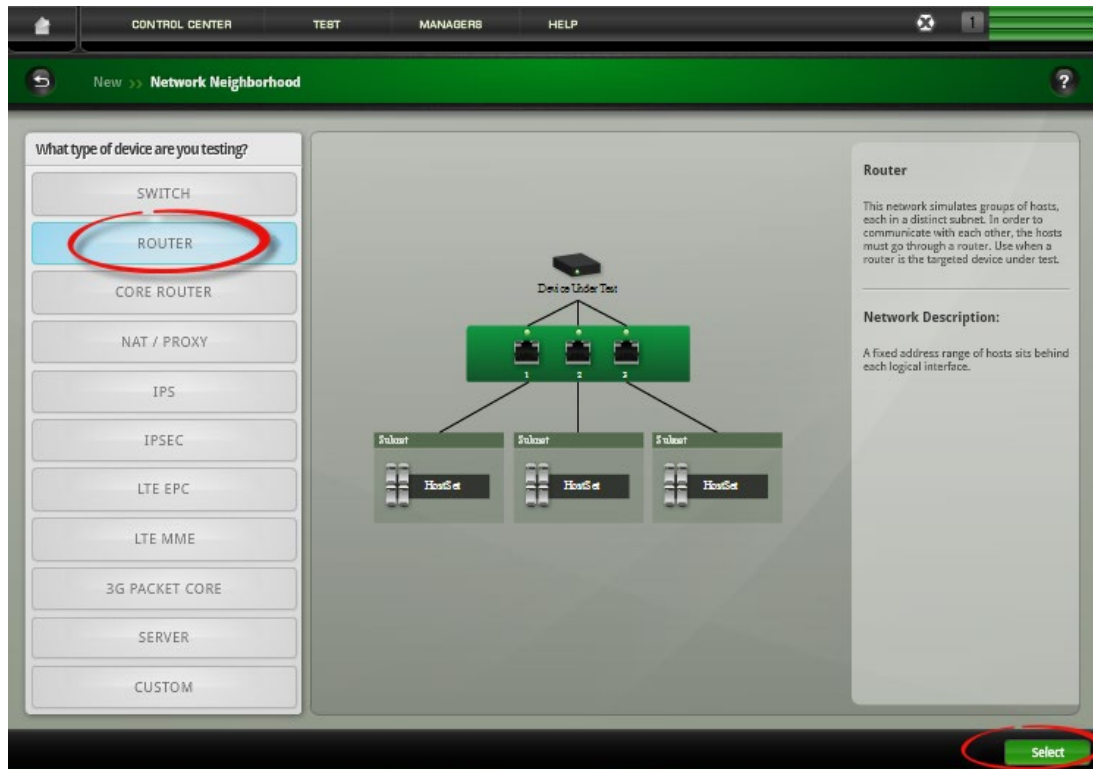
4. Since we already reserved the physical test ports (and, if applicable extra ports for more resource reservation) now we need to configure the MAC and IP layer parameters like MAC address VLANs, IP addressing scheme, MTU, etc. All these settings, among others are controlled/defined from the **Network Neighborhood**:

Test Methodologies for Known Vulnerabilities and Malware

- a. From the upper menu bar select **Control Center** -> **New Neighborhood**:

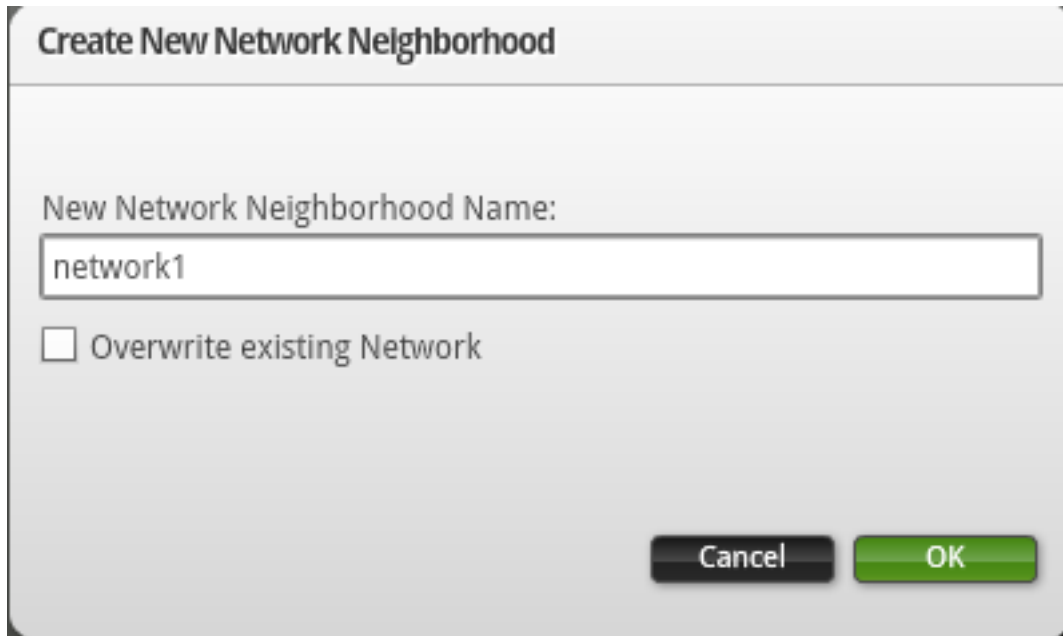


- b. Select the DUT type used for the test and a corresponding default Network Neighborhood will be created:

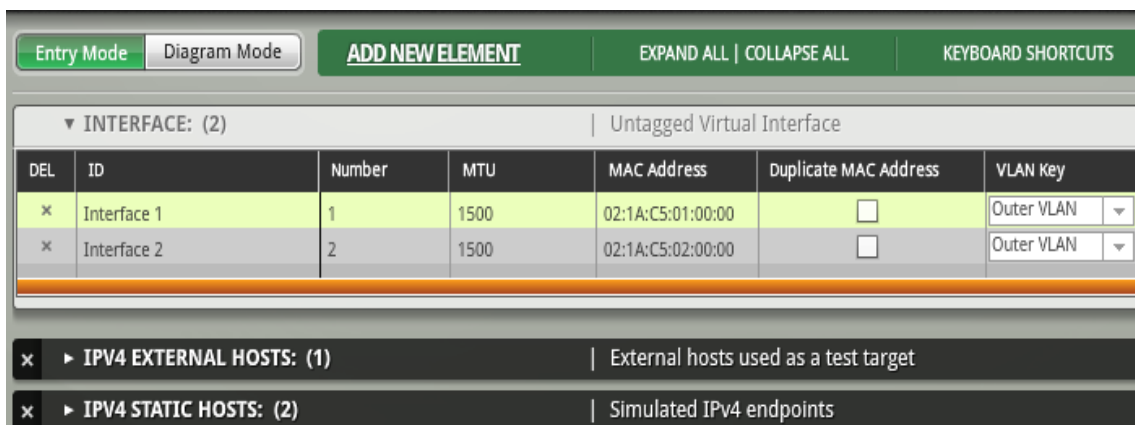


For this test, we will use *ROUTER* as DUT type.

- c. Enter an easy-to-remember name for the new Network Neighborhood and click **OK**:



- d. According to the DUT type we previously selected, a default Network Neighborhood will be created. Since we chose *ROUTER* DUT type, the following elements are automatically created:
- i. Two *INTERFACES*: Provides access to interface level parameters like MTU, MAC, Impairments, and Packet Filters.
 - ii. One *IPv4 EXTERNAL HOSTS*: configures the IP address of the external end hosts/servers. This element is not required for NGFW devices configured as Pass-Through. It is only need for devices that are terminating the TCP Connection (e.g. Server Load Balancers).
 - iii. Two *IPv4 STATIC HOSTS*: Provides access to IP related parameters of the BreakingPoint emulated hosts. The Static Hosts represents the BreakingPoint emulated attackers and target servers.



Test Methodologies for Known Vulnerabilities and Malware

For this test, under **IPv4 Static Hosts** we will use the following:

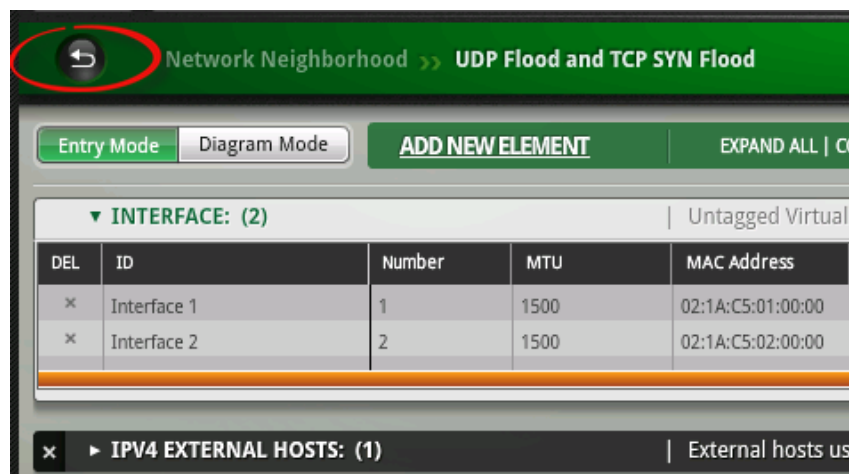
- Change the first entry (with ID as *Static Hosts i1_default*) to have a tag of “*Trusted*”, Base IP Address as 12.1.1.2, Count of 100 and Gateway IP address as 12.1.1.1
- Change the second entry (with ID as *Static Hosts i2_default*) to have a tag of “*Untrusted*”, Base IP Address as 13.1.1.2, Count of 100 and Gateway IP address as 13.1.1.1

DEL	ID	Container	Tags	Base IP Address	Count	Gateway IP Address
x	Static Hosts i1_default	Interface 1	Trusted	12.1.1.2	100	12.1.1.1
x	Static Hosts i2_default	Interface 2	Untrusted	13.1.1.2	100	13.1.1.1
x	Static Hosts i3_default	Interface 3	i3_default	3.0.0.2	253	3.0.0.1
x	Static Hosts i4_default	Interface 4	i4_default	4.0.0.2	253	4.0.0.1

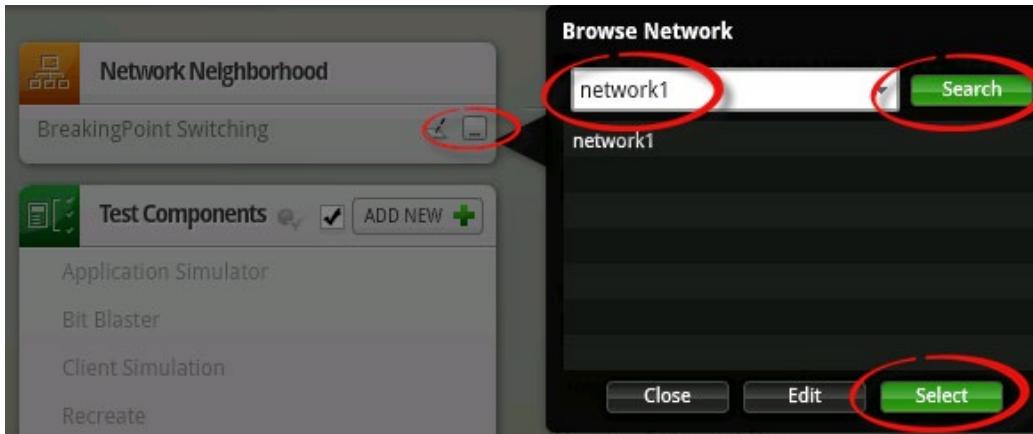
- e. Save the new Network Neighborhood configuration by clicking the **Save** button located on the lower right corner:



- f. Click on the back arrow to return to the main test screen:



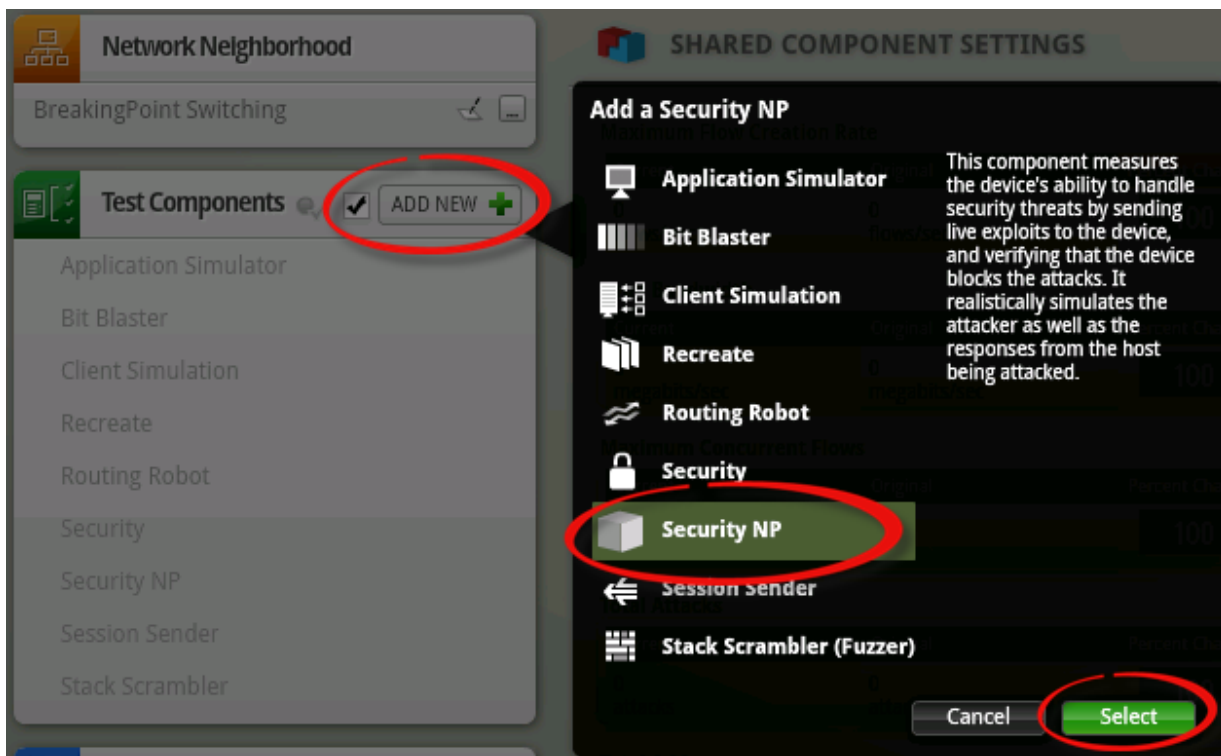
- g. From the main test screen click on the browse button from the **Network Neighborhood** section to search and select the above created Network Neighborhood:



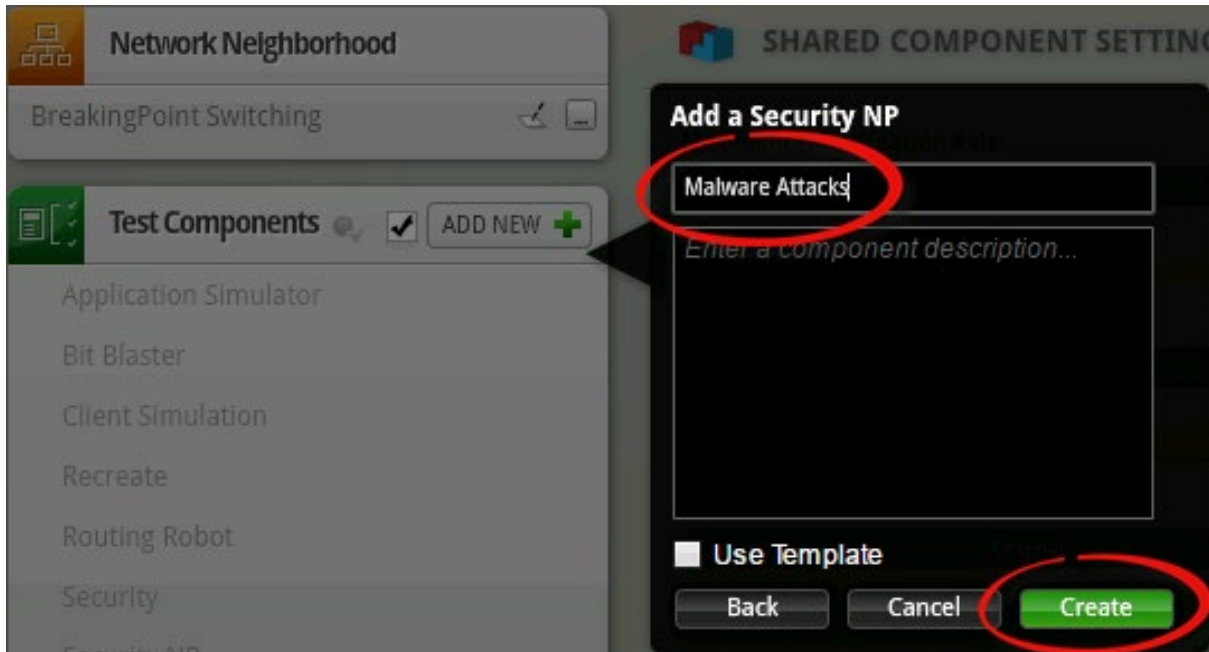
After configuring the MAC and IP layer parameters the actual traffic component needs to be created.

The Test Component is the entity that controls the traffic patterns. For this example, we will use the Security test component to emulate the actual attacks.

5. Click on the **ADD NEW** button from the **Test Components** section. Choose *Security NP* from the selection list and click on **Select** button:

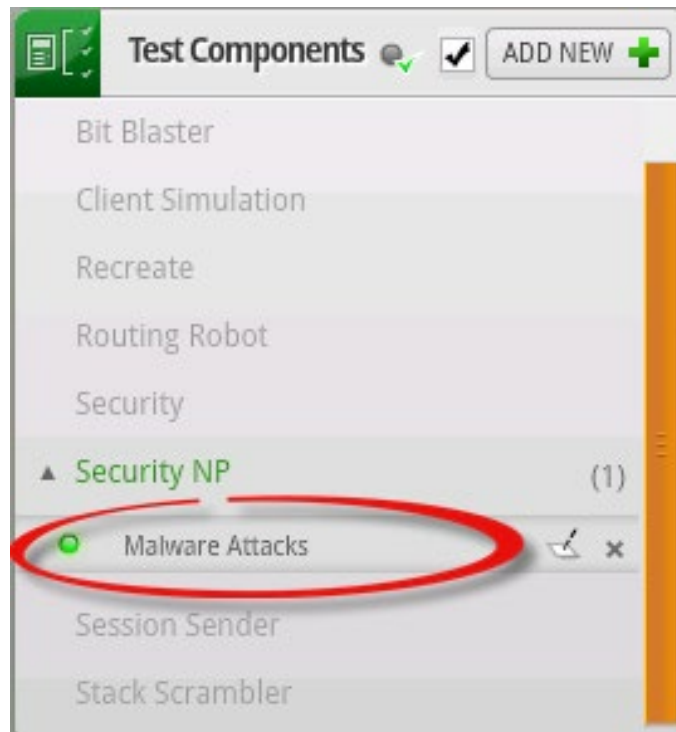


6. Rename the component from *SecurityNP_1* to something more meaningful if required and click Create button:



A new entry (i.e. *Malware_Attrcks*) will be created under **Security NP** Test Component.

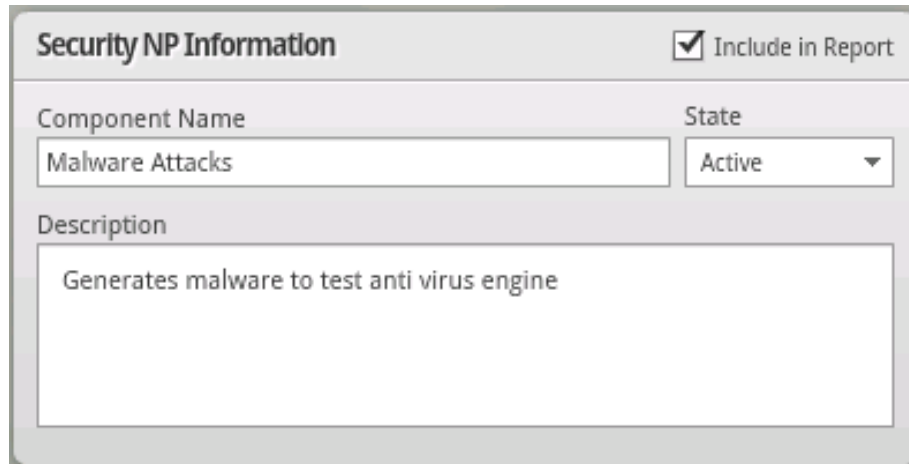
7. Click on the newly created component to edit its parameters:



8. In the next steps, we will configure the Security NP test component:

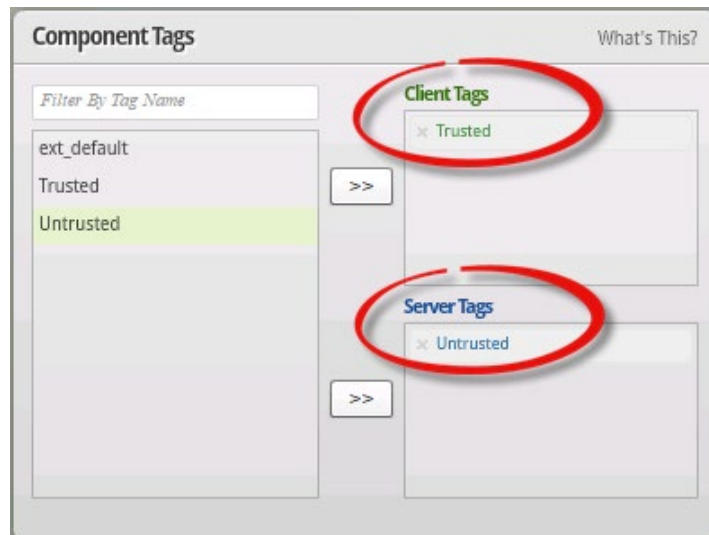
Test Methodologies for Known Vulnerabilities and Malware

- a. A meaningful description can be added in the **Description** box for easy reference of what this particular component is configured for:



The screenshot shows a configuration window titled "Security NP Information" with a "What's This?" link in the top right. A checkbox labeled "Include in Report" is checked. Below the title bar, there are two input fields: "Component Name" with the text "Malware Attacks" and "State" with a dropdown menu set to "Active". A large text area labeled "Description" contains the text "Generates malware to test anti virus engine".

- b. In the **Component Tags** section make sure to assign the proper interface tags. For the **Client Tags**, assign the tag corresponding to the IPv4 Static Host Network Neighborhood element emulating the Trusted side. For the **Server Tags**, assign the tag corresponding to the IPv4 Static Host Network Neighborhood element emulating the Untrusted side:



The screenshot shows a "Component Tags" configuration window with a "What's This?" link in the top right. On the left, there is a list of tags: "ext_default", "Trusted", and "Untrusted". The "Untrusted" tag is highlighted in green. In the center, there are two right-pointing arrow buttons (">>"). On the right, there are two panels: "Client Tags" and "Server Tags". The "Client Tags" panel has a red circle around it and contains a tag labeled "Trusted". The "Server Tags" panel has a red circle around it and contains a tag labeled "Untrusted".

- c. The **Parameters** section contains multiple test component attributes that are grouped into category folders based on their functional purpose.

The screenshot shows a 'Parameters' configuration window. At the top, there are buttons for 'Save As Template' and 'Load a Template'. Below these is a search bar labeled 'Filter By Parameter Name' with a 'Clear' button. The main area is divided into two sections: 'Data Rate' and 'Security Configuration'. The 'Data Rate' section includes: 'Data Rate Unlimited' (unchecked checkbox), 'Data Rate Scope' (dropdown menu set to 'Limit Per-Interface Throughput'), 'Data Rate Unit' (dropdown menu set to 'Frames / Second'), 'Data Rate Type' (dropdown menu set to 'Constant', marked with a red asterisk), 'Minimum Data Rate' (text input field set to '1', marked with a red asterisk), and 'Maximum Data Rate' (text input field set to '1'). The 'Security Configuration' section includes: 'Maximum Simultaneous Attacks' (text input field set to '256'), 'Maximum Attacks Per Second' (text input field set to '256'), 'Delay Start' (time picker set to '00 : 00 : 00' with labels 'Hour', 'Minute', 'Second' below), and 'Attack Retries' (text input field set to '0').

Data Rate Section:

- i. **Data Rate Unlimited:** Enabling this option will instruct the engine to send data as fast as possible. Typically, this is enabled for a very high throughput test. For this test, leave it at default (unchecked)
- ii. **Data Rate Scope:** This option can take two values: Limit per interface or aggregate throughput. Leave this at default value.
- iii. **Data Rate Unit:** The data rate limits can be applied either on frames/second or on Mbps. Leave this at default.
- iv. **Data Rate Type:** This option defines whether a constant data rate is requested or a range. If a random or range is selected, the minimum and maximum bounds of the

Test Methodologies for Known Vulnerabilities and Malware

data rate are configured by the below two parameters. For this test, leave this as default.

- v. **Minimum Data Rate:** This configures the minimum data rate value. If data rate type is constant, this is the value for the constant data rate.
- vi. **Maximum Data Rate:** This option is enabled only when data rate type is Range or Random.

Security Configuration Section:

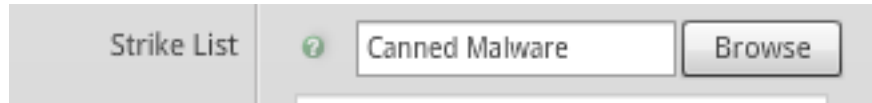
The screenshot shows a web interface for 'Security Configuration'. The settings are as follows:

Parameter	Value
Maximum Simultaneous Attacks	256
Maximum Attacks Per Second	256
Delay Start	00 : 00 : 00 (Hour, Minute, Second)
Attack Retries	0
Random Seed	0
Strike List	Canned Malware (with a 'Browse' button)
Strike List Iterations	1
Strike List Iteration Delay	0
Evasion Profile	Default evasion settings (with 'Edit' and 'Browse' buttons)

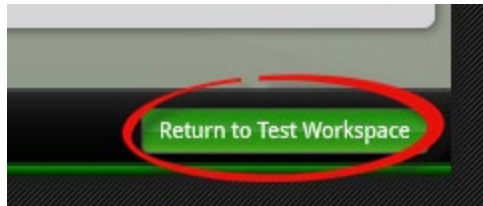
- i. **Maximum Simultaneous Attacks:** This parameter controls how many simultaneous attacks will be generated.
- ii. **Maximum Attacks Per Second:** This parameter controls the rate of the attack generation.
- iii. **Delay Start:** If there needs to be a delay before the attacks are generated, configure the value here. Leave it at default value of 0.
- iv. **Attack Retries:** The number of times a particular action within a strike will be retried before considered blocked by the device.
- v. **Random Seed:** This controls how random data will be generated by the Strikes. Leave this at default value of 0 indicating that the seed will itself be randomly generated.

Test Methodologies for Known Vulnerabilities and Malware

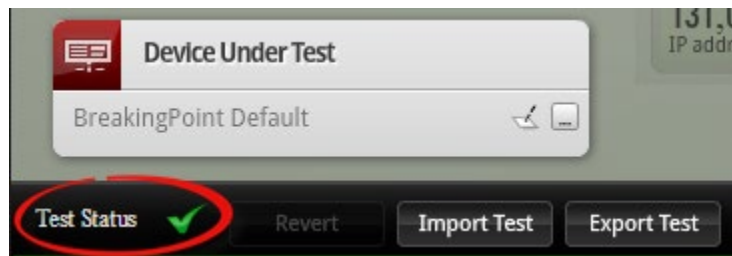
- vi. **Strike List:** The strike list defines the list of attacks the clients will generate. For this test, we will use the *Canned Malware*. These are malware strikes that are pre-installed on the system



- vii. **Strike List Iterations:** This parameter controls how many iterations of the strike list the test should do. For this test, leave this at Default value of 1.
- viii. **Strike List Iteration Delay:** This parameter configures the delay between iterations.
- ix. **Evasion Profile:** This configures the different evasion techniques to apply on the malware traffic. For this test, we will not configure any evasion profile and leave it at Default.
- x. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:

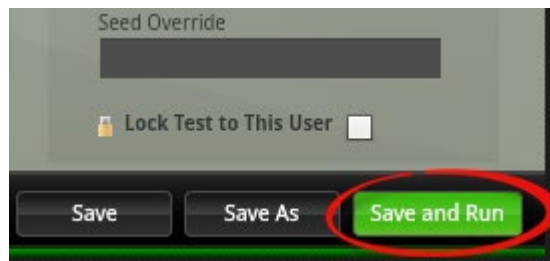


- 9. Make sure the **Test Status** indicated (on the lower left corner) has a green checkmark:



If there is not, determine what is wrong by selecting Test Status and viewing the errors.

- 10. Select **Save and Run** from the lower right corner:

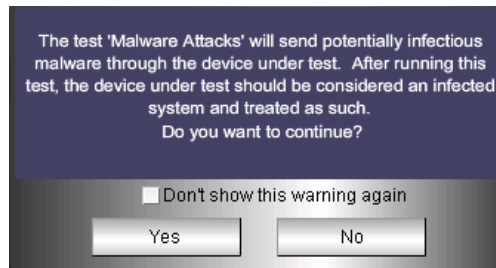


11. If the test has not previously been saved, enter a name for the test and click **Save**



Run the test and while the test is running continue to monitor the DUT with respect to the malware that is being detected. See the Results Analysis section for important statistics and diagnostics information.

Note: Since the malware attacks that will be generated by BreakingPoint also contain real infectious samples a relevant warning message will be displayed before the actual test execution will start:



Results Analysis

This section covers the key statistics and events that BreakingPoint provides for this type of test.

Real time Stats

After the test is started and initialized, the screen will automatically switch to Real Time Statistics window.

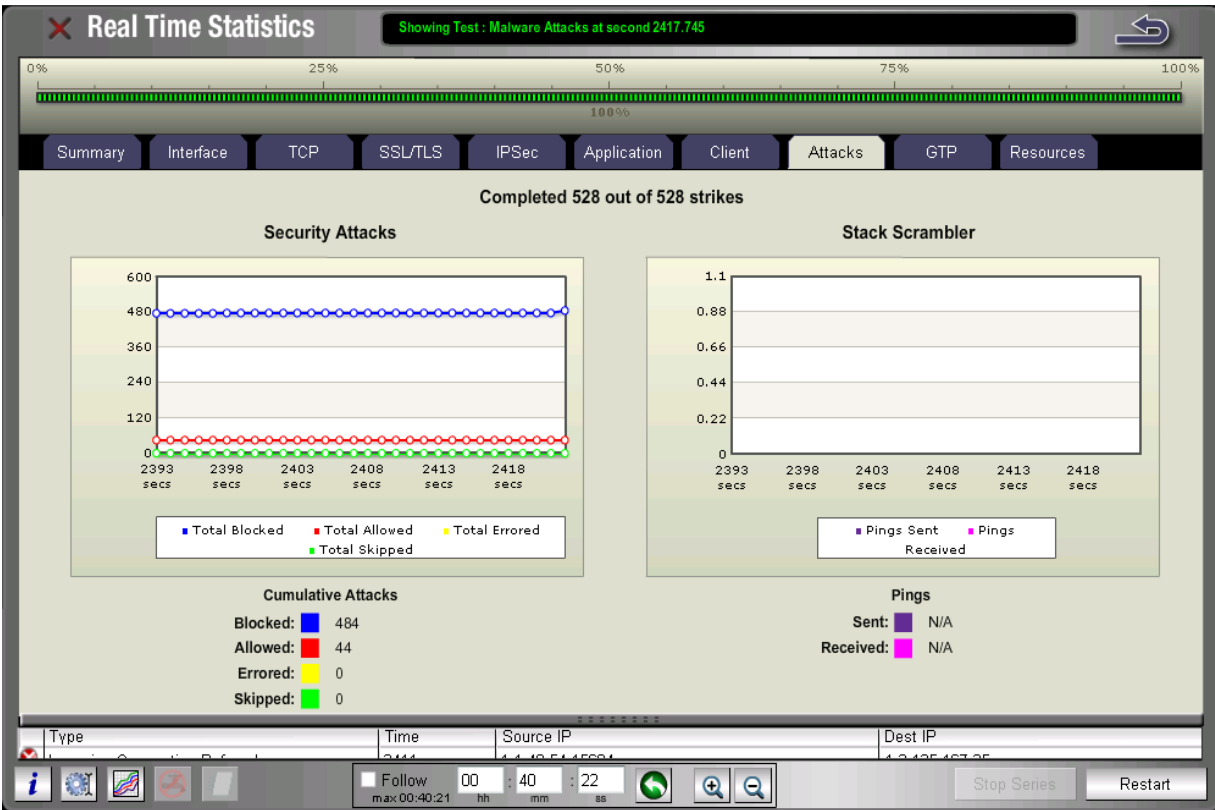
The **Attacks** tab shows how many attacks will be run and how many of those have been completed thus far. At the bottom of this page are more detailed cumulative statistics, the most relevant being:

Blocked: Of those attacks completed, how many have been blocked by the device

Allowed: Of those attacks completed, how many have been allowed by the device

Also, some of the strikes can be **Skipped** which means that some of the attacks configured in the strike list are deprecated and therefore they will be skipped. Users that prefer to run them can create an evasion profile that will have **Allow Deprecated** option set. Skipped also will occur when there is an IPv6 network neighborhood configured to run a Strike that requires IPv4, and vice versa.

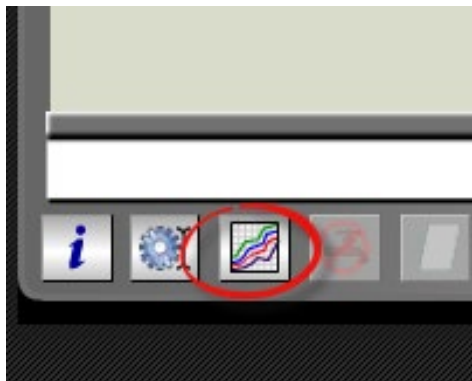
Test Methodologies for Known Vulnerabilities and Malware



The test will stop once all the strikes have been completed.

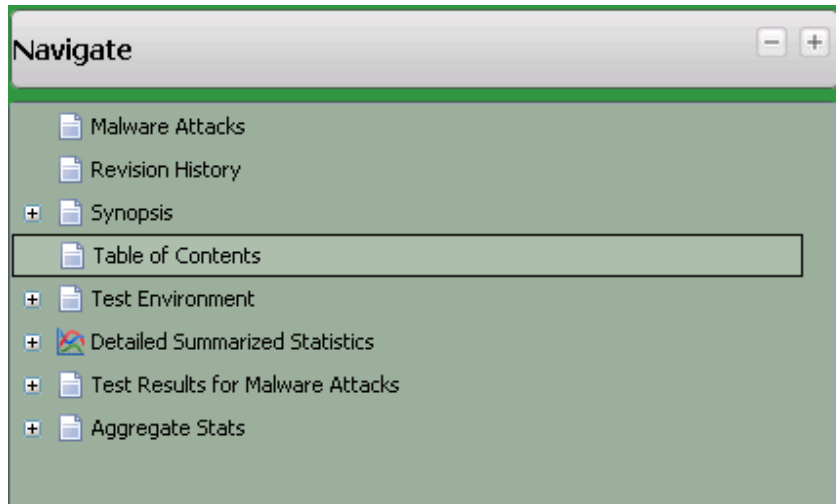
Report

A detailed report can be generated selecting the graph button from the lower left corner of the Real Time Statistics window. This will open the results in a new browser window:

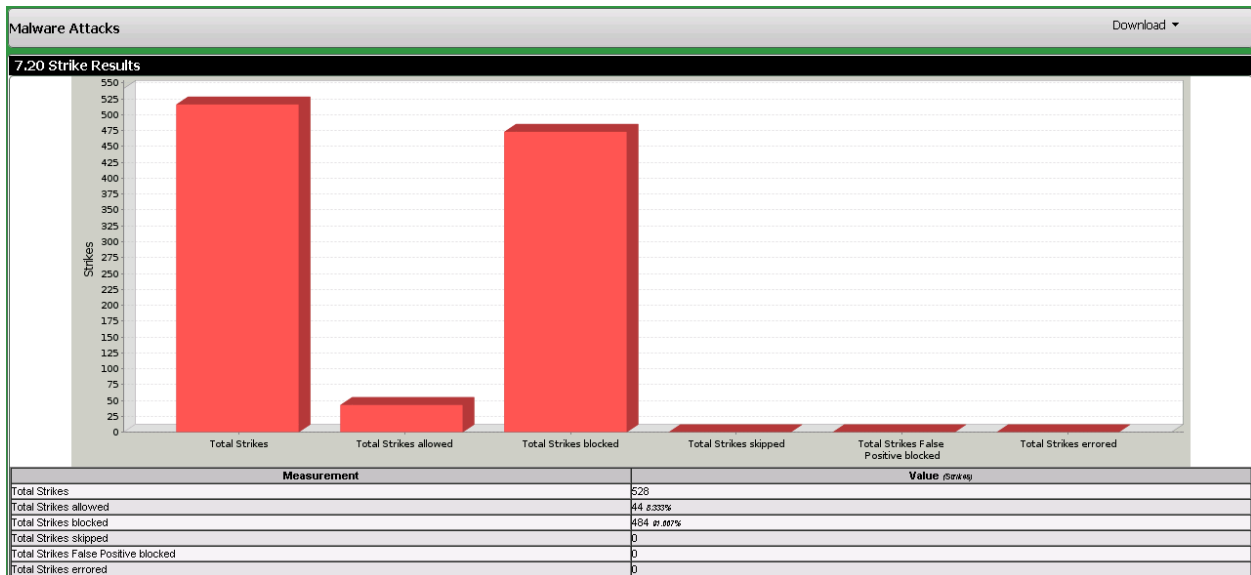


Test Methodologies for Known Vulnerabilities and Malware

Once a test report is generated, click on the Table of Contents from the left pane to visualize all the report sections:



Navigate to Section 7.20 (Test Results for Malware Attacks -> Strike Results) to view the malware Results that shows the percentage of strikes that were allowed and blocked:



Section 7.21.27 (Component Detection Assessment -> Strikes -> Malware) has the distribution of allowed and blocked by category which will show the aggregate details in with the relevant categories: Click on the Section name (malware mobile and malware all to see further details details):

Malware Attacks Download ▾

7.21.27.1.1 Malware: All 484/528

Strike Section	Blocked	Allowed	FalsePositiveBlocked	Skipped	Errored
Malware: Mobile	14	0	0	0	0
Malware: All	30	0	0	0	0

Test Methodologies for Known Vulnerabilities and Malware

Section 7.21.27.1.1.1 gives details regarding malware mobile. Each entry will specify whether the malware was blocked or allowed and the corresponding malware reference:

Malware Attacks				Download ▾
7.21.27.1.1.1 Malware: Mobile 157/171				
Time of strike	Strike Name	Strike Result	Strike Reference	
1,723.0	Live Malware Mobile Symbian OS Blankfont.A Trojan	Blocked	http://virscan.org/report/02f3b031655bd6d2e76dc9118097a56.html http://www.novusord.ro/index.php?menu=ven&mal=Worm.SymbOS.Blankfont.a	
1,723.0	Live Malware Mobile Symbian OS CReadMe Trojan	Blocked	http://contajodump.blogspot.com/2011/03/take-sample-leave-sample-mobile-malware.html http://www.fortiguard.com/encyclopedia/virus/symbos_creadme.atr.html	
1,754.0	Backdoor.AndroidOS.KungFu.ki	Allowed		
1,760.0	Live Malware Mobile Android OS Dougalek.A Trojan	Blocked	https://www.virustotal.com/file/b0d98c2d4581aa100bc7ed1dae681711d95af1afa6f76e52bba6a5606f2041d1/analysis/	
1,760.0	Live Malware Mobile Android OS FakePlayer.C Trojan	Blocked	http://www.virustotal.com/file-scan/report.html?id=2a7a2a1ac2172eb9819bd18b2d7161bd89da2d2b332c3c5f0d73f75500bc1-1301408197	
1,761.0	Live Malware Mobile Symbian OS Cabir.J Worm	Blocked	http://www.virustotal.com/file-scan/report.html?id=9c3326c815844c7e58779f1fa5050b775163c39b50a8c50f19f5e493935f19b-1308775515	
1,761.0	Live Malware Mobile Symbian OS Cabir.J Worm	Blocked	http://www.novusord.ro/index.php?menu=ven&mal=Worm.SymbOS.Cabir.J https://www.virustotal.com/file-scan/report.html?id=af0f995308e459d09b7d69d425d4a1abcf52f75cac1567210d1fbab54e78ebe-1308760676	
1,811.0	Live Malware Mobile Symbian OS Skudoo.A Trojan	Blocked	http://www.f-secure.com/w-doscs/skudoo_a.shtml http://www.virustotal.com/file-scan/report.html?id=ce2120b3db7e6efb3f0cae441499e9e550b61693fb75fce20d1e61ffcd2bc2-1309290978	
1,811.0	Live Malware Mobile J2ME Konov.b SMS Trojan	Blocked	http://www.virustotal.com/file-scan/report.html?id=1f82a0d52a18f3bd287c505b413f38cd06c3a7827450d02528e4e3b9a51dd4-1301198873	

Similarly, Section 7.21.27.1.1.2 gives details regarding malware all. Each entry will specify whether the malware was blocked or allowed. The reference will have the CVE number if present:

Malware Attacks				Download ▾
7.21.27.1.1.2 Malware: All 327/357				
Time of strike	Strike Name	Strike Result	Strike Reference	
131.0	Live Malware CVE-2008-3005.XLS.00	Allowed	CVE 2008-3005	
143.0	Trojan.PSW.Win32.Tepfer.fdmk	Blocked		
143.0	Live Malware CVE-2010-1297.PDF.03	Blocked	CVE 2010-1297	
156.0	Trojan.PSW.Win32.Tepfer.hkdb	Blocked		
156.0	Adware.Win32.Hotbar	Blocked		
160.0	Dorkbot	Blocked		
178.0	Adware.Clikpotato.gen3	Blocked		
178.0	Gen.Variant.Kazy.94410	Blocked		
221.0	Win32.Trojan	Allowed		
226.0	Live Malware CVE-2010-0188.PDF.13	Blocked	CVE 2010-0188	
226.0	not-a-virus:Downloader.Win32.Pds.q	Blocked		
233.0	Trojan.Win32.AVKill.hq	Blocked		

To better understand which malware was allowed, click on section 7.21.26 (Allowed Strike List), which will provide details of the category and the exact malware that was not detected by the AV engine.

Malware Attacks				Download ▾
7.21.26 Allowed Strike List				
Timestamp	Strike Category	Strike Name		
131.0	Malware: All	Live Malware CVE-2008-3005.XLS.00		
221.0	Malware: All	Win32.Trojan		
290.0	Malware: All	Live Malware CVE-2009-4324.PDF.05		
394.0	Malware: All	Win32.Simda		
473.0	Malware: All	Live Malware CVE-2010-0188.PDF.05		
509.0	Malware: All	Trojan.Win32.Inject.FitF		
561.0	Malware: All	Backdoor.Agent.ABEW		
561.0	Malware: All	Trojan.Win32.Generic		
629.0	Malware: All	Hoax.Win32.ArchSMS.ovki		
629.0	Malware: All	ArtemisCS460957FB34		
851.0	Malware: All	Medfos.FBIR		
851.0	Malware: All	Win32.Alp.ToDown.B		
746.0	Malware: All	Live Malware CVE-2009-4324.PDF.02		
778.0	Malware: All	Live Malware CVE-2010-0188.PDF.07		
785.0	Malware: All	Live Malware CVE-2010-0188.PDF.02		
806.0	Malware: All	Live Malware CVE-2009-0927.PDF.04		
884.0	Malware: All	Backdoor.Win32.Simda.gen		
894.0	Malware: All	Adware.Win32.Hotbar		
894.0	Malware: All	Live Malware CVE-2009-0658.PDF.02		
1,040.0	Malware: All	JDS: DangerousObject.Multi.Generic		
1,213.0	Malware: All	HEUR:Trojan.Downloader.Win32.Generic		
1,278.0	Malware: All	Adware.Win32.Hotbar		
1,597.0	Malware: All	Gen.Variant.Zusy.Etrob.1632		
1,597.0	Malware: All	Rancos.Trojan		
1,694.0	Malware: All	Live Malware CVE-2010-0188.PDF.21		
1,754.0	Malware: Mobile	Backdoor.AndroidOS.KungFu.ki		
2,016.0	Malware: Mobile	Live Malware Mobile Symbian OS Cabir.C Worm		
2,253.0	Malware: Mobile	Live Malware Mobile Symbian OS Cabir.D Worm		
2,283.0	Malware: Mobile	Live Malware Mobile Symbian OS SteaWar.C Trojan		
2,283.0	Malware: Mobile	Live Malware Mobile Symbian OS Bootton.A Trojan		
2,297.0	Malware: Mobile	Live Malware Mobile J2ME Smmer.Trojan		
2,311.0	Malware: Mobile	Live Malware Mobile Symbian OS Skulls.D Trojan		
2,319.0	Malware: All	Adware.Clikpotato.gen3		
2,327.0	Malware: Mobile	Live Malware Mobile Android OS Adrd.A Trojan		
2,335.0	Malware: Mobile	Live Malware Mobile Android OS Getmiml.A Trojan		
2,338.0	Malware: Mobile	Live Malware Mobile Symbian OS Cabir.G Worm		
2,339.0098	Malware: Mobile	Live Malware Mobile Symbian OS CReadMe.Trojan		
2,353.0	Malware: Mobile	Live Malware Mobile Symbian OS Drevor.C Trojan		
2,356.0	Malware: Mobile	Live Malware Mobile Symbian OS Mabtal.A Trojan		
2,356.0	Malware: All	Neis.Z		
2,356.0	Malware: All	Trojan.Dropper.Win32.Dapato.btyr		
2,356.00195	Malware: Mobile	Live Malware Mobile Android OS DroidDream Bowlingqtime.Trojan		
2,374.00293	Malware: All	Trojan.Malfinder.Win32.Agent.akw		
2,388.00195	Malware: All	Live Malware CVE-2010-0188.PDF.09		

Test Methodologies for Known Vulnerabilities and Malware

All these statistics should be compared with the logs from the NGFW device to make sure that the blocked malware are properly identified and classified and no other traffic drops of other nature (other than properly identifying the strikes as malware) caused the strike to get blocked.

Additionally, using all this information the NGFW configuration can be re-visited to understand why the allowed malware was permitted to pass through.

Test Variables

Use the following test configuration parameters to repeat the test.

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
IP Version	IPv4	IPv6, IPsec, DSLite, 6rd, selected Mobility stacks, etc.
Data Rate	Limited	Unlimited
Strike List	Critical Strikes	Live Malware (see appendix below)
Maximum Attacks Per Second	256	Higher (1024)
Benign Traffic	None	Add benign traffic using AppSim component to stress the AV engine. The traffic should be a mix that matches the end customer deployment production network. The objective of such a new test should be to find the maximum performance of the device with same level of detection accuracy reached in the initial test

Test Methodologies for Known Vulnerabilities and Malware

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
Evasion Techniques	Disabled	<p>To use the evasion with malware, please use the Security component (not the Security NP component). The rest of the steps will remain the same. Malware supports the following evasion sections in an evasion Profile.</p> <p>IP Section</p> <p>TCP Section</p> <p>Malware Section: Of particular interest is the Transport Protocol under the malware section. This enables the malware to be transported using a variety of different transport protocols. If the anti-virus is scanning for malware only for a particular protocol, this evasion will be successful in bypassing the anti-virus protection</p>

Conclusions

This configuration covered the main parameters of the Security NP component in BreakingPoint using a practical example allowing the user to baseline the security effectiveness of an anti-virus engine.

Test Methodologies for DoS and DDoS

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are the oldest methods of attacking IP networks. While those methods are well-known and have been studied for years, they continue to remain one of the most effective ways to restrict access to a network, service, or application for legitimate users.

DoS versus DDoS

By definition, the intent of a DoS/DDoS attack is to deny access of legitimate users to resources provided by a victim's network, computer, or service. When this attempt is initiated from a single host, the attack is called a DoS attack. While some of the DoS attacks can be successful by using a single host with limited resources—compared with the victim's computer—the majority of the attacks require a group of malicious hosts to flood the victim's network by generating an overwhelming amount of attack packets. This type of attack is called DDoS.

According to Internet World Stats⁵, the worldwide Internet population at the end of November 2015 exceeded 3.3 billion users. Many Internet users browse the Internet without appropriate security software, or by using operating systems and software that are not properly patched. Those systems are vulnerable to attacks, allowing attackers to use automated techniques to discover such systems and use known vulnerabilities to install DDoS tools on their system. Such infected computers are named zombies or bots. Through automation, attackers exploit a large number of vulnerable computers, infecting them with malware software that gives attackers control to those systems.

WORLD INTERNET USAGE AND POPULATION STATISTICS NOVEMBER 30, 2015 - Update						
World Regions	Population (2015 Est.)	Population % of World	Internet Users 30 Nov 2015	Penetration (% Population)	Growth 2000-2015	Users % of Table
Africa	1,158,355,663	16.0 %	330,965,359	28.6 %	7,231.3%	9.8 %
Asia	4,032,466,882	55.5 %	1,622,084,293	40.2 %	1,319.1%	48.2 %
Europe	821,555,904	11.3 %	604,147,280	73.5 %	474.9%	18.0 %
Middle East	236,137,235	3.3 %	123,172,132	52.2 %	3,649.8%	3.7 %
North America	357,178,284	4.9 %	313,867,363	87.9 %	190.4%	9.3 %
Latin America / Caribbean	617,049,712	8.5 %	344,824,199	55.9 %	1,808.4%	10.2 %
Oceania / Australia	37,158,563	0.5 %	27,200,530	73.2 %	256.9%	0.8 %
WORLD TOTAL	7,259,902,243	100.0 %	3,366,261,156	46.4 %	832.5%	100.0 %

Figure 16. Internet Usage and World Population Statistics (November 30, 2015)

A zombie computer reports back to a Command & Control center (C&C) by attempting a login session. After zombie computers are logged on, they become a part of a botnet that allows the

⁵ <http://www.internetworldstats.com/stats.htm>

attacker to control them. The most common C&C servers are Internet Relay Chat (IRC) or Web Gateways.

Relying on hundreds to thousands of computers that have been previously infected with worms or trojans, large DDoS attacks can be coordinated. Larger botnets can consist of millions of zombie computers, which can generate aggregated traffic of hundreds of Gbps. One of the largest DDoS attacks reported in 2016 clocked in at over 600Gbps according to <https://krebsonsecurity.com>⁶. Also, based on Arbor's 2016 Worldwide Infrastructure Security report⁷ the largest DDoS reported attack in 2015 was 500Gbps compared to only 8Gbps in 2004:

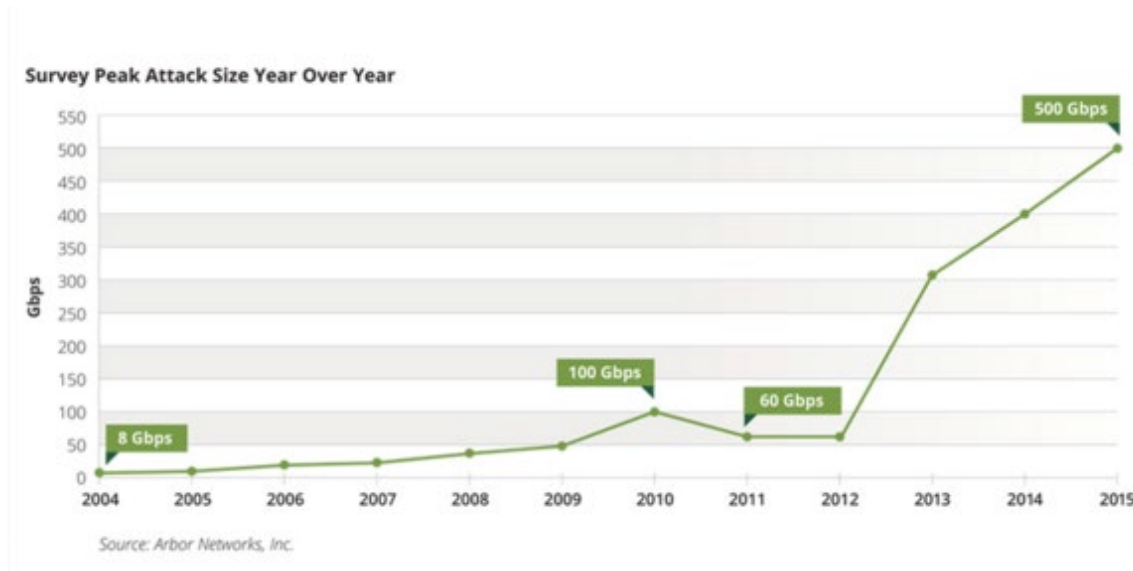


Figure 17. Largest DDoS attack reported per year

Along with the dramatic increase in throughput and volume, the nature of the attacks is becoming more complex: Arbor's Network's same report reveals that application-layer attacks have continued to increase, being reported by 93 percent of service provider respondents, compared to 90 percent in 2014 and 86 percent in 2013. Another increasing factor is the multi-vector attack type, seen by 56 percent of service provider respondents from 42 percent one year before. Also of note is that aside from the actual targeted system, over a half of organizations (enterprise, government, and education respondents) had their firewall or IPS devices experience a failure or contribute to an outage during a DDoS attack.

To increase the effect of the attack, servers are also targeted to be part of the botnet. Server machines give the advantage of having better computing resources and their bandwidth is usually higher. Additionally, the attack traffic will be generated from trusted datacenter IPs.

⁶ <https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/>

⁷ https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf

Also of note is the increasing usage of “Reflected” and “Amplification” Denial of Service attacks. Reflected DDoS attacks abuse connectionless protocols that can be used to both mask the botnets initial source location and increase the amount of data sent to the victim. Protocols typically used for this attack include DNS, NTP, and SSDP.

Motivation for DoS/DDoS attacks

DoS attacks are illegal activities. Regardless, such attacks continue to be frequently seen. They exist because they are easy to implement and the attack source is difficult to detect. A large number of tools are available on the Internet. The most common ones include Mirai, LOIC (Low Orbit Ion Cannon), R-U-Dead-Yet, DAVOSET, Tribe Flood Network (TFN) and its newer version TFN2K, Trinoo (Trin00), Stacheldraht, myServer, Mstream, Omega, Trinity, Plague, and Derivatives.

As an example, a dissection of the command and control protocol used by the increasingly common Mirai botnet is available on the Ixia blog site - [Mirai: A Botnet of Things](#)⁸.

Revenue driven/Monetary Gain

Motivated by monetary gain, many attackers advertise their service for DDoS attacks using underground forums. According to Verisign’s Q4 2014 DDoS Trends Report⁹, the prices of such services are extremely low and the cybercriminals offering them (so called “booters”) had become real professionals. There is actually competition between such DDoS service providers, which drove the prices as low as USD 5 per hour or USD 30 per day.

⁸ <https://www.ixiacom.com/company/blog/mirai-botnet-things>

⁹ <http://www.verisigninc.com/assets/report-ddos-trends-Q42014.pdf?cmp=LK-RPT-TRENDSQ1-1>

Test Methodologies for DoS and DDoS

Service Name	Service Pricing (USD)
Xakepy.cc	1 hour starts at \$5 24 hours starts at \$30 1 week starts at \$200 1 month starts at \$800
World DDoS Service	1 day starts at \$50 1 week starts at \$300 1 month starts at \$1,200
King's DDoS Service	1 hour starts at \$5 12 hours starts at \$25 24 hours starts at \$50 1 week starts at \$500 1 month starts at \$1,500
MAD DDoS Service	1 night starts at \$35 1 week starts at \$180 1 month starts at \$500
Gwapo's Professional DDoS Service	1-4 hours at \$2 per hour 5-24 hours at \$4 per hour 24-72 hours at \$5 per hour 1 month at \$1,000 fixed
PsyCho DDoS Service	1 hour for \$6 1 night for \$60 1 week for \$380 1 month for \$900
DDoS Service 911	1 night for \$50
Blaiz DDoS Service	1 day for \$70 1 week starts at \$450
Critical DDoS Service	1 day starts at \$50 1 week starts at \$300 1 month starts at \$900
No. 1* DDoS_SERVICE	1 day starts at \$50 1 week starts at \$300 1 month starts at \$1,000

Figure 18. Example as shown in Verisign's Q4 2014 DDoS Trends Report

To prove the size of the botnet owned, demonstrations of DDoS attacks are done by DDoS service providers to attract their customers. In many cases, many victims are randomly picked for such demonstrations. Competitive situations where the buyer rents DDoS services to cause loss in competitor's sales or to affect their reputation may be a strong motivator.

Cases of extortion using DDoS were reported as well. According to Sophos¹⁰, in 2006, an arrested group of Russian cyber-criminals made USD 4 million from blackmailing online gambling and casino websites.

Payback/Revenge

On October 26, 2007, a UK based company, MoneyExpert.com¹¹, experienced a DDoS attack, which put their site down during the weekend. During the day of the attack, the website planned to launch their payment protection insurance (PPI) reclaiming campaign, a massive new campaign to help many people recover thousands of pounds back on mis-sold insurance on

¹⁰ <http://www.sophos.com/pressoffice/news/articles/2006/10/extort-ddos-blackmail.html>

¹¹ <http://www.moneysavingexpert.com/site/moneysavingexpert.com-ddos-attack>

loans, credit cards, store cards, and mortgages. Suspicions that the attack was the payback of the banks suffering the potential loss remains unproved. Another plausible cause could be the legitimate attempt to use the website from the large number of people that could benefit from the reclaiming campaign.

Unexpected Peak Hours

DDoS attacks can be the result of an overwhelming number of legitimate users trying to access websites announcing hot news or events that interest millions of users in a short time interval. One of the most publicized examples is where Google¹² mistook the millions of search queries for a distributed DoS attack. Google search volume index chart depicted next shows the peak time was at 15:00 PDT.

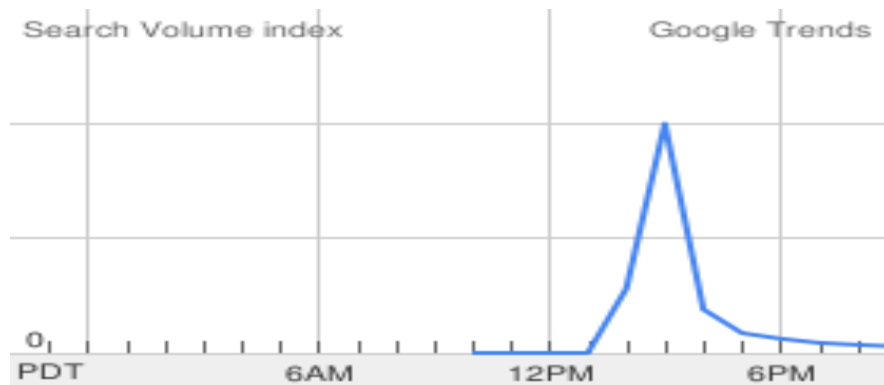


Figure 19. Google Trends - search for "Michael Jackson died"

To filter legitimate traffic, Google prompted the users with an error page displaying a CAPTCHA field to let the users continue their query.

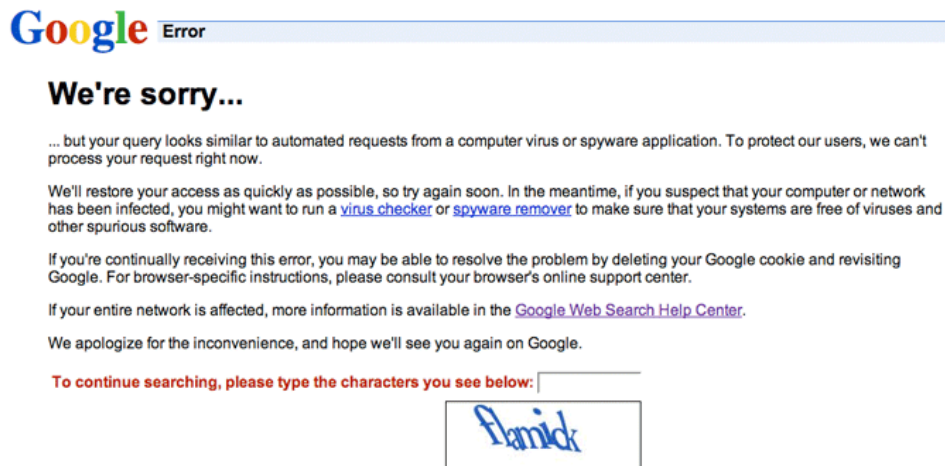


Figure 20. Google returned message to a valid search

¹² <http://www.christian-kalmar.com/google-michael-jacksons-ddos-attack/>

Collateral damage

According to McAfee¹³, on August 6, 2009, a DDoS attack targeted social media sites hosting the account of a pro-Georgian blogger. As a result, the attack took down Twitter for several hours and significantly slowed down Facebook.

Hactivism

Various organizations are using DDoS attacks as means of protesting for ideological reasons. Such attacks are launched using both zombies (machines that have been infected without owner knowledge) and users who are consciously participating in the attack—adhering therefore to the same ideological beliefs. Although typically these attacks are announced before the actual launch, they are extremely efficient, and few organization are able to successfully protect themselves.

Miscellaneous

In many cases, the reason behind a DDoS attack remains unknown. Those attacks may simply have the intent of practicing an attack, being initiated 'for fun' or because of nihilism and vandalism. Many video tutorials are available online accompanied by links to download the toolsets that can carry out the attacks.

¹³ <http://www.avertlabs.com/research/blog/index.php/2009/08/07/collateral-damage/>

DoS/DDoS: Methods of Attack

A large variety of DoS attacks can be attempted. Based on their intent, they can be classified as follows:

- Resource starvation
- Alteration or destruction of system configurations
- Hardware damage

The most common denial of service methods intend to flood the victim's computer or network with useless data that result in overutilization of the following:

- Network bandwidth
- CPU utilization
- Memory consumption
- Disk and storage

Because of the limited nature of such resources on any system, they represent an easy and common target in DoS attacks. Usually, the attacks have a temporary effect, and availability of resources come back once the DoS attack stops.

Based on the resources targeted, the attacks can be further classified as follows:

- Bandwidth consumption

One of the easier ways to deny access to a resource is by consuming the bandwidth available between the ISP and the victim's network or within the victim's network itself. Bandwidth can be easily consumed with any garbage data—UDP, ICMP, or TCP-based traffic.

By consuming the entire bandwidth available, traffic from legitimate sources may result in connection drops. DoS attacks targeting bandwidth consumption require a higher bandwidth than the victim's bandwidth or they are relying on amplification techniques. A basic example of DoS that uses amplification is the Smurf attack, which floods the victim's network by using spoofed ICMP messages sent to a broadcast address.

- System resource starvation

These attacks focus on consuming system resources such as CPU time and memory. CPU time is usually consumed with packets initiating new connections (for example, TCP SYN Flooding, HTTP GET Flooding, SIP INVITE Flooding attacks) or packets targeting non-existing sessions (for example, TCP FIN Flooding, SIP ACK flooding attacks).

Memory starvation can be achieved with legitimate connections that are maintained active after a connection is established.

By consuming these resources in an excessive manner, they become unavailable to legitimate users and systems.

- DoS attacks targeting protocol and software flaws

These attacks attempt to exploit design flaws in software. Those attacks do not require a large botnet to be effective; a single host machine can be used to send packets at low rates and lead to DoS by crashing the victim's computer, causing the computer to stop responding, or rebooting it.

Some examples of such DoS attacks that take advantage of the protocol's inherent design include PING of Death or Land Attacks.

- Storage

As a general rule, anything that allows data to be written to disk can be used to execute a DoS attack, assuming that no protection is set on the amount of data that can be written. As an example, an intruder may attempt to consume disk space by simulating actions that generate error messages on the victim's computer, errors that are logged and stored to the disk. Other examples may include massive amounts of unsolicited email messages or upload of useless data in unprotected locations (for example network shares or ftp accounts).

- Alteration or destruction of system configurations

These types of attacks require access to the victim's computer. Exploits based on known vulnerabilities in the operating system or application itself may allow attackers to gain root access to the system. By altering key configuration aspects of the server—routing tables, network configuration, user passwords, registry keys—or by destroying certain data (for example, the information stored in a database), an intruder may prevent users from accessing the compromised computer or network.

Routing-based DoS attacks target modification of the routing table, preventing the victim from properly sending or receiving legitimate traffic.

To simplify the use of network addressing, name systems such as Domain Name Servers (DNS) provide a way to map the user-friendly name for a computer or service to the IP address associated with that name. DNS is a hierarchical naming system and has the root domain on top of the hierarchy. An intruder who gains access to a DNS server can alter the cached data to direct legitimate traffic to wrong Internet (IP) addresses. This results in either flooding victim's network or preventing the victim from sending or receiving any traffic.

- Hardware damage

Attackers who gain root access to systems may destroy the hardware. As an example, attempting to update the firmware of a device with a corrupted image may result in permanent damage.

Common DoS/DDoS Attacks

TCP SYN Flooding

TCP SYN is one of the most common DDoS attacks. A typical TCP connection requires a three-way handshake in which the client computer requests a new connection by sending a TCP SYN

packet to its remote peer. In response, the TCP SYN/ACK packet is sent by the remote peer and the TCP connection request is placed into a queue, continuing to wait for the TCP ACK packet, which completes the handshake.

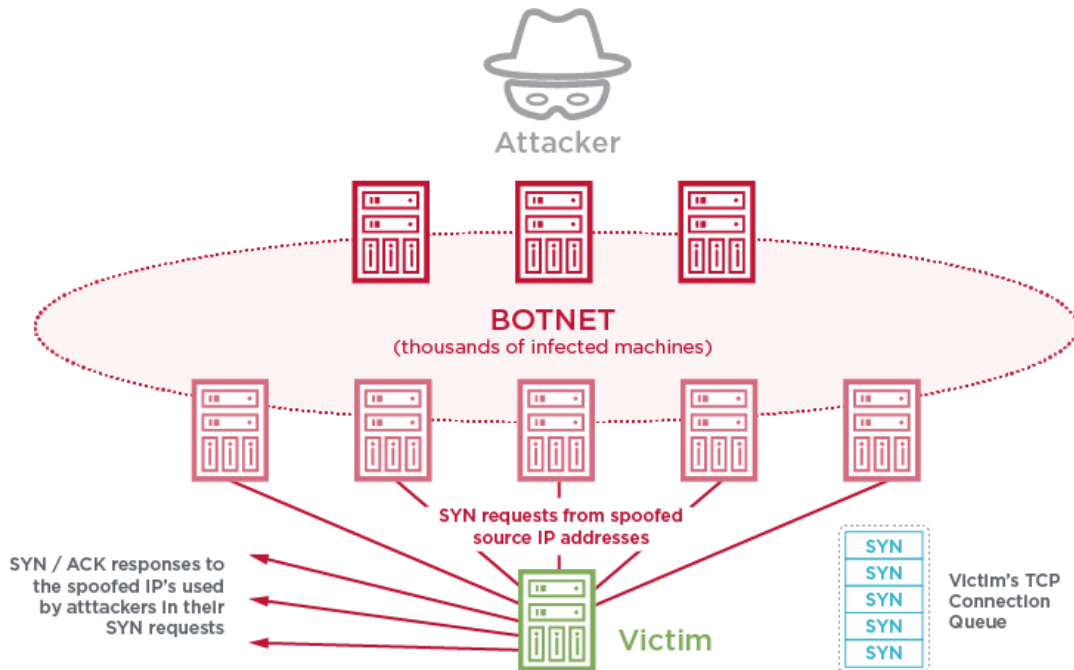


Figure 21. SYN Flooding DDoS attack

The attacker sends to victim's IP a storm of TCP SYN packets initiated from a large number of spoofed IP addresses, causing the victim to open a huge number of TCP connections and respond to them with SYN/ACK. Because the attacker never ACKs the SYN/ACK packet, the victim ends up in a state in which it cannot accept new incoming TCP connections—even if they are coming from legitimate users.

Fake Session Attack

This attack attempts to fake a complete 3-way TCP handshake. It is designed to bypass network defense tools that monitor traffic in a single direction, not relying on returned server traffic, as seen in common asymmetric routed networks. The attack sends a crafted SYN packet, multiple ACK packets and then one or more FIN/RST packets creating the appearance of valid unidirectional TCP session. There are two versions of the fake session attack: one can be simulated by generating multiple SYN packets, followed by multiple ACK packets, and then one or more FIN/RST packets. The other version skips the initial SYN packet, and starts by sending multiple ACK packets, then one or more FIN/RST packets. This attack is typically harder to detect than SYN flood but both have similar goals, exhausting target system resources.

UDP Flooding

A UDP flooding attack relies on a large number of attackers sending multiple UDP packets to the victim's computer, saturating its bandwidth with useless UDP packets. The attack packets can target open and closed ports. When the packets are targeting ports on which the victim's

computer is not listening, ICMP Destination Unreachable (Port Unreachable) packets may be replied by the victim to the spoofed IP included with each UDP packet. This will result in additional processing time and an additional storm of UDP packets destined to other computers.

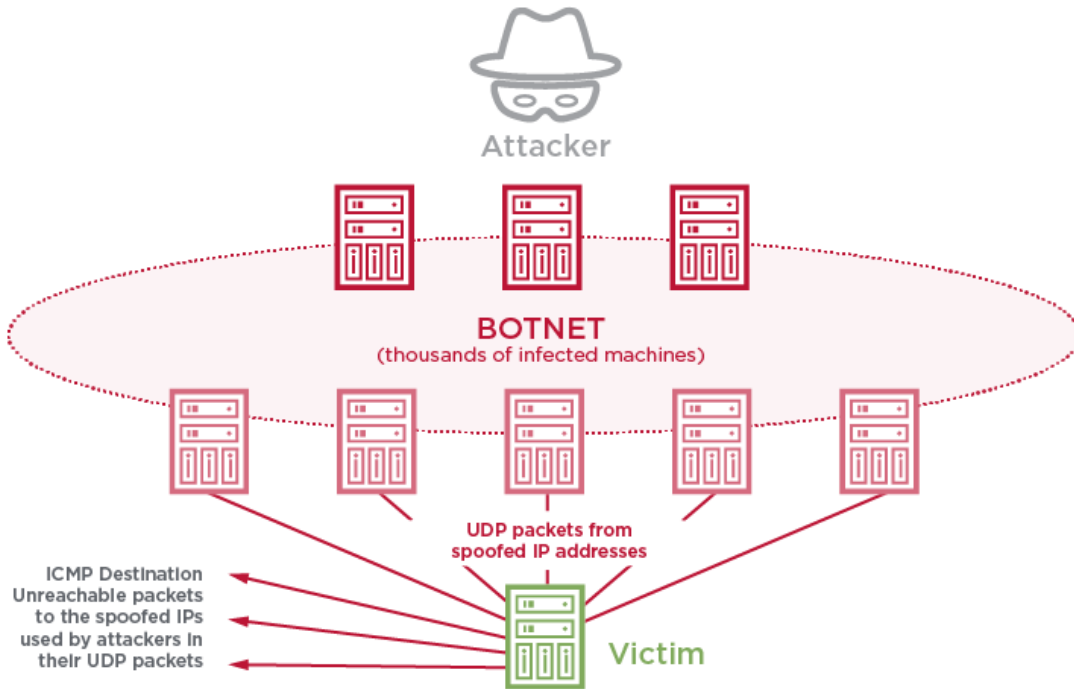


Figure 22. ICMP Flooding

PING Flooding attack

This threat floods the victim (a server or an end user) with multiple ICMP Echo request packets (PING), thus saturating its bandwidth. This is a very standard attack that can be done with utilities, such as PING, included with any operating system.

Smurf Attack

Smurf is yet another ICMP Echo request (PING) type of attack.

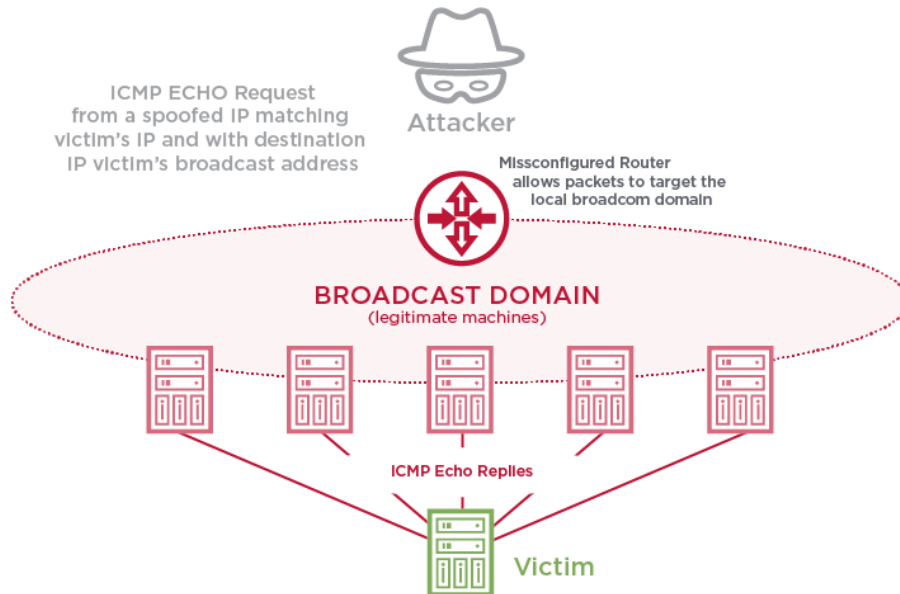


Figure 23. Smurf DoS attack

The attack exploits improperly configured networks, which allow external packets coming from the Internet to have the destination as an IP broadcast address. By sending a storm of ICMP Echo request packets with the address spoofed with the intended victim's address, all the ICMP Echo requests are reflected back to all computers of the local network, resulting in an amplified number of replies destined to the victim's computer. The effect of this attack is the same as with the PING Flooding attack.

Fraggle Attack

A Fraggle attack uses the same technique as described in a Smurf attack, except that the packets sent are UDP echo instead of ICMP Echo packets. The targeted services are *echo* (port 7) and *chargen* (port 19).

Defined by RFC 862, the ECHO service is an Internet protocol that listens on port 7 for either TCP or UDP. On receipt of a packet, the ECHO protocol sends back a copy of data that it receives.

Defined by RFC 864, Character Generator (CHARGEN) is an Internet Protocol that listens on port 19 for UDP and TCP packets. On receipt of a UDP packet, a random number of characters are returned as a UDP response packet. On receipt of TCP packets, random characters are sent to the connecting host until the TCP connection is closed by the host.

ICMP 'Destination Unreachable'

On receipt of an ICMP 'Destination Unreachable' packet, the recipient will drop the corresponding connection immediately. This behavior can be exploited by an attacker by simply sending a forged ICMP Destination Unreachable packet to one of the legitimate communicating hosts. The DoS attack is achieved by breaking the communication of the legitimate hosts involved in communication.

ICMP 'Host Unreachable'

The ICMP Host Unreachable packet is another ICMP packet type that can be used to break the communication of two hosts. The DoS is achieved as described for ICMP 'Destination Unreachable', except that the packet type is ICMP 'Host Unreachable'.

ICMP 'Time Exceeded'

The Time Exceeded Message is an ICMP message that is generated by a gateway to inform the source of a discarded datagram because of the *time to live field* reaching zero. A time exceeded message may also be sent by a host if it fails to reassemble a fragmented datagram within its time limit.

This type of ICMP packet can also be used to break the communication of two hosts. The DoS is achieved as described for ICMP 'Destination Unreachable', except that the packet type is ICMP 'Time Exceeded.'

Teardrop Attack

This is a fragmented message where the fragments overlap in a way that destroys the individual packet headers when the victim attempts to reconstruct the message. This may cause the victim to crash or stop responding.

FIN Flood Attack

This threat floods a user-specified target with TCP packets from randomized, spoofed addresses, where the FIN (final) flag has been turned on. The FIN flag is sent by a user to designate that it is no longer sending packets. This attack is an attempt to flood the target with erroneous packets to hinder the performance and cause a slowed response to legitimate traffic and possibly DoS.

RST Attack

This vulnerability could allow an attacker to create a Denial of Service condition against existing TCP connections, resulting in premature session termination. Because an attack uses a random IP as the source IP, it is possible that the source IP or computer (if it exists) will send a reset packet (RST/ACK) back to the server that says it did not make the connection request. More likely, the IP address does not correspond to an active connection (because it is a random number); the server will keep trying to initiate a connection by resending SYN/ACK, and then RST/ACK (because it did not get any ACK back) packets back to the bogus source IP address. All this creates incomplete or half-open connections.

SYN-ACK Flood Attack

This attack generates spoofed SYN-ACK packets toward the target server at a very high packet rate, which are normal TCP responses toward clients from initial SYN packet when initiating a 3-way handshake. This flood of SYN-ACK packets will exhaust the server CPU and memory resources, causing poor system performance or shutdown.

Xmas Tree Flood Attack

A Christmas Tree attack sends TCP packets with either all or various combinations of the TCP flags being set (like FIN, URG and PSH). Flooding the victim with these kinds of packets can be very effective since the unusual flag combination requires more processing than regular, legitimate packets.

Application Level DoS and DDoS

Flaws in software implementations can be exploited to cause buffer overflow, consume all memory and CPU, crash the application stack, make the computer stop responding, or reboot the computer.

Another group of DoS attacks relies on brute force, flooding the target with an overwhelming flux of packets depleting the target's system resources. Brute force attacks at application level flood the victim with the legitimate-looking application requests that initiate transactions at application level. Examples of such attacks include HTTP GET/POST Flooding, SIP INVITE Flooding, DNS Flooding, and many others.

HTTP GET Flooding attack

This attack is achieved by sending an overwhelming number of HTTP GET or HTTP POST requests to the targeted HTTP Server, depleting the victim's resources. The requests have legitimate contents and are originated over valid TCP connections. By serving those requests as normal requests, the server ends up exhausting its resources.

By asking for large files stored on the server, the attack is amplified as a single legitimate request that can keep the server busy for a longer duration. A large HTTP GET Flooding attack was seen in U.S. and Korea in July 2009. The targets were websites of major organizations, news media, financial companies, and several government websites.

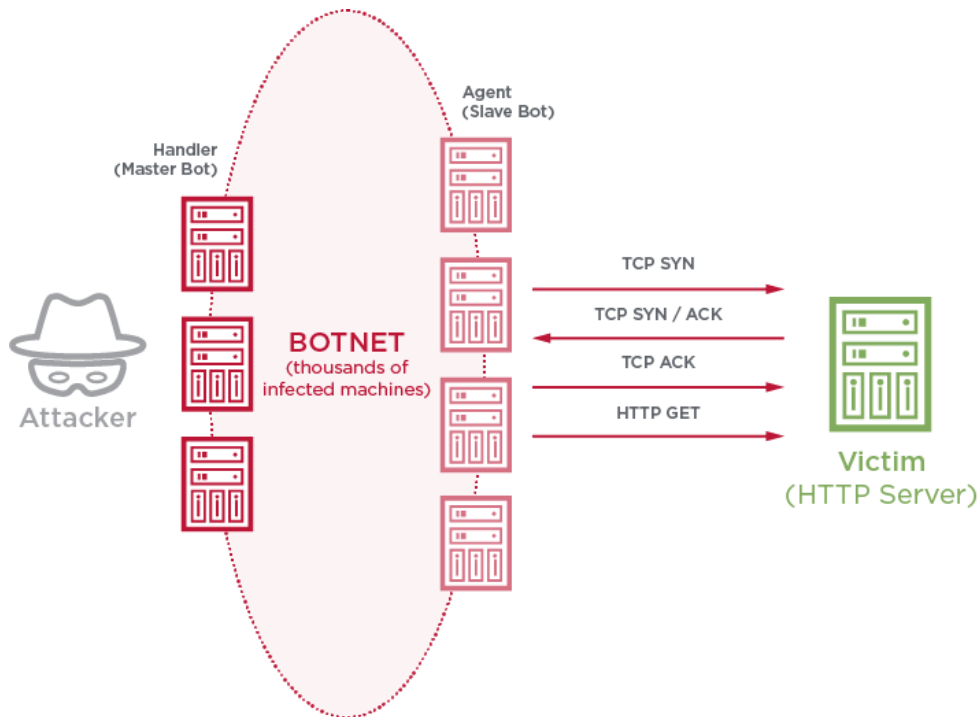


Figure 24. HTTP DDoS attack

DNS Flooding Attack

This threat attacks a DNS server by sending a high number of DNS requests that look like they are initiated from the victim's IP address. The small queries sent by the zombie computers are amplified by the recursive DNS Servers that are used as intermediaries to resolve the domain. This generates response to larger UDP packets, overwhelming the victim's computer. Another type of DNS Flooding attack can overwhelm a DNS Server by sending legitimate DNS queries to resolve random domain names, forcing the DNS Server to resolve them by initiating further queries to root servers and authoritative name servers. The storm of DNS queries may lead to resource depletion, therefore causing the DoS effect.

Slow Loris / Partial Header Attack

This is a very effective, low-bandwidth attack that opens TCP connections to a target server and holds these TCP sessions open for long periods of time. Partial GET requests in the form of subsequent headers are sent at long regular intervals, but never close the TCP connection. These connections consume server memory and CPU resources and remain open indefinitely.

RUDY

Another type of low and slow attack, similar to Slow Loris is R-U-Dead-Yet? (RUDY). It can be difficult to detect and very effective by opening and keep sessions alive as long as possible using never-ending POST requests. These requests usually contain one byte of data and are sent at regular or variable (to make detection even more difficult) intervals. Continuously accumulating this kind of sessions eventually results in exhausting the target server's connection table.

SIP Flooding Attacks

This class of attacks floods the victim with a significant number of SIP messages, including REGISTER, INVITE, OPTIONS, MESSAGE, BYE, SUBSCRIBE, NOTIFY, ACK, and PING. The messages are sent from spoofed IP addresses and target depletion of victim's resources by forcing the victim to process useless SIP messages.

0 Byte Receive Window

This attack combines in an unexpected manner an application layer transaction with some TCP layer parameters. In particular, an attacker can craft HTTP GET requests but having a TCP receive window set to 0. Typically, a TCP receive window set to 0 is used to notify sender that receiver is temporarily busy and cannot accept more incoming traffic (i.e. receiving buffer is filled up). This way a large number of TCP sockets can potentially stay open for extremely long time durations.

DNS Reflection

One of the most efficient way of generating large volumetric DDoS attacks is by using amplification techniques. DNS is one of the most exploited protocols to perform such attacks.

In a DNS reflection attack, the attacker abuses publicly accessible open DNS resolvers by sending fake DNS requests with the source IP address spoofed to the target address. The DNS servers will then flood the target with real DNS response traffic. An important aspect to this type of attack is the amplification factor defined as the ration between the response and the query size. Various methods can be used to amplify such an attack like sending spoofed queries with type "ANY" hence having the server reply with all available information related to a DNS zone.

NTP Reflection

Another type of amplification attack that is gaining traction among cybercriminals and has an even higher amplification factor than DNS is the NTP Reflection attack. Network Time Protocol (NTP) is a long-existing UDP based protocol, which offers time synchronization services. A particular functionality of NTP (i.e. MONLIST) allows administrators to query NTP servers and retrieve a list with the last (up to 600) endpoints previously connecting to the server. The nature of this response creates substantially larger traffic volumes when compared to the initial request. Abusing internet-open NTP Servers with this functionality enabled, attackers can spoof MONLIST requests (using the victim source IP address) and generate a sheer volume of network traffic.

Test Solution for Dos and DDoS

Ixia offers a broad and comprehensive testing solution to validate and assess DDoS mitigation solutions. Leveraging its unmatched flexibility, BreakingPoint can run DDoS attacks using virtually all test components. It offers solutions ranging from network-based L2/4 attacks (like ARP flood, Ping flood, etc.) all the way up to L7 application attacks (like Slow Loris, Rudy and SIP-related attacks, etc.), and even attacks based on the transmission of invalid traffic.

Test Methodologies for DoS and DDoS

The next sections describe some of the most common DDoS scenarios and explain how to validate related network security protection solutions using a series of test cases.

It is of high importance when validating DDoS mitigation solution to use a real application traffic mix, as seen in the customer setup. It is also important to initiate attacks that might be relevant to the system types present in the production network.

Regardless of the exact DDoS mitigation solution that is being considered, the following metrics needs to be evaluated:

1. Number and type of DDoS attacks blocked (relevant only in accordance with the end user system types)
2. Time to detect and mitigate
3. DDoS effectiveness, composed of two parts:
 - Amount of DDoS traffic blocked
 - Impact on real-user traffic (measured with failed transactions and increased traffic latency)

Test Case: Mitigation of UDP Flooding and TCP SYN Flooding Attacks

Overview

This test methodology walks you through a configuration that uses a combination of volumetric attacks such as UDP flooding and TCP SYN flooding attacks to measure the mitigation capabilities of an anti-DDoS solution.

Objective

The goal of this test case is to measure DUT's capabilities to detect and mitigate UDP flooding and TCP SYN flooding DDoS attack. Incremental tests should to be considered where application traffic is layered with the DDoS attacks to assess the impact on legitimate users of various services like web, voice or video.

Setup

The current setup consists of two PerfectStorm test ports connected directly to the device/system under test.

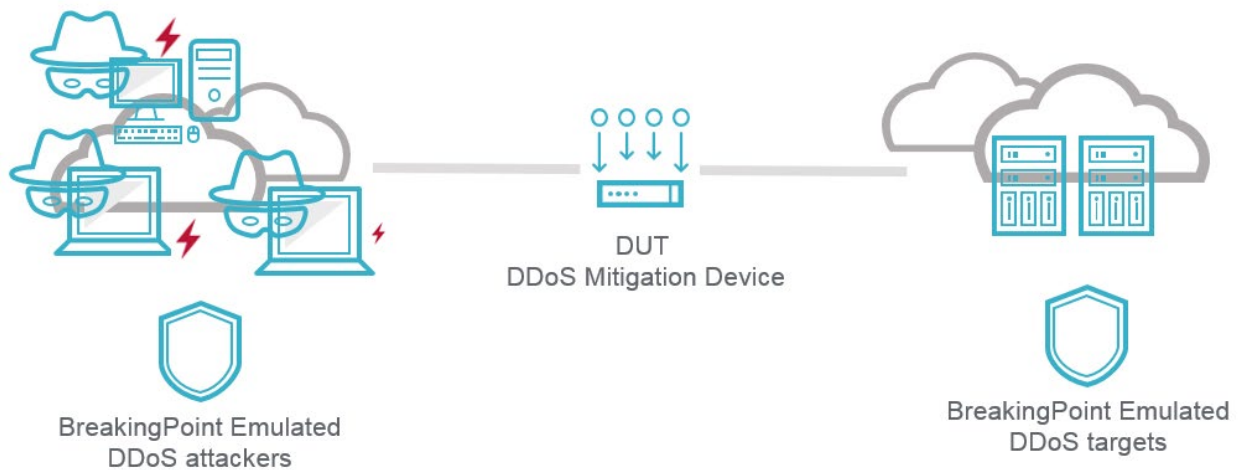


Figure 25. Test topology

In this test topology, BreakingPoint emulates:

- On port1:
 - BOTNET consisting of 1000 DDoS Attackers
- On port2:
 - Target network by placing 10 Servers

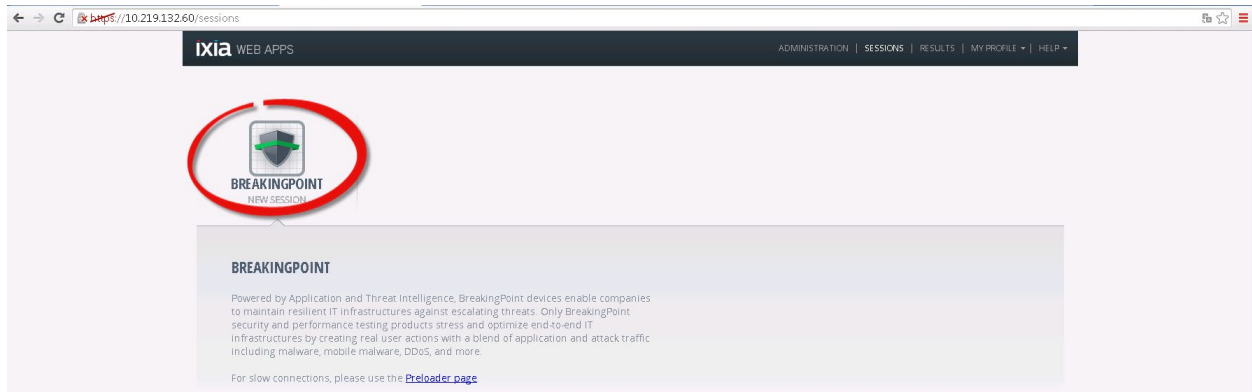
Test Methodologies for DoS and DDoS

This topology can be used to test intermediate devices such as dedicated anti-DDoS solutions, firewalls, and unified thread management systems. The Target Server is optional and can be replaced with a real external target, such as an Apache Web Server.

Step-by-Step Instructions

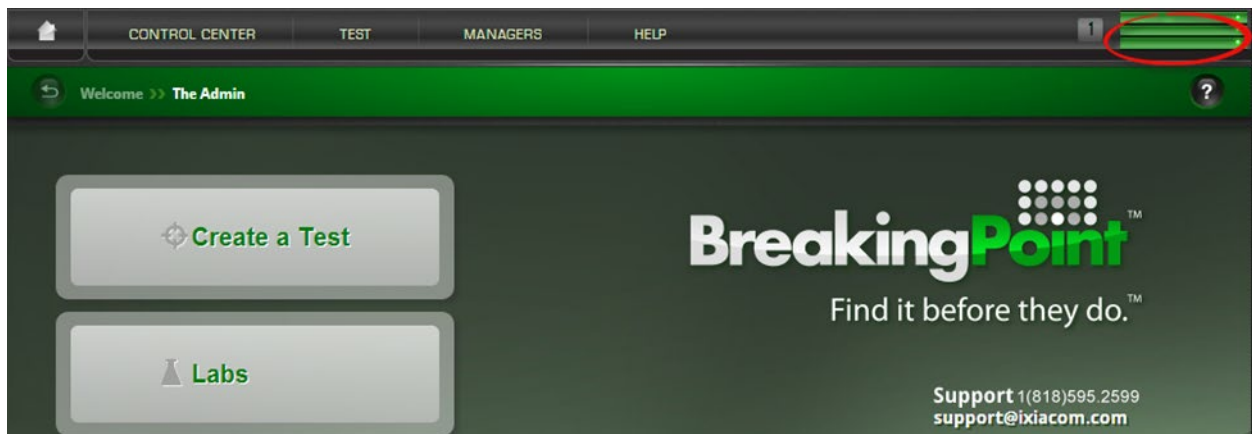
This section provides step-by-step instructions to execute this test using the BreakingPoint tool.

1. Use a web browser connected to the Ixia Web Apps GUI and start a new BreakingPoint Web Session:



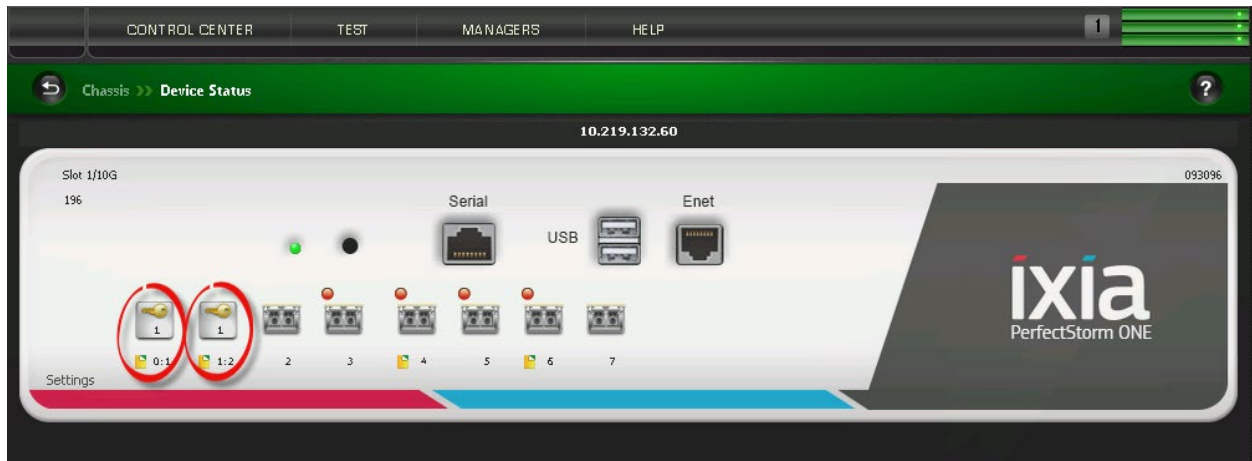
2. Once the BreakingPoint web home page loads, reserve the test ports to be used in the physical test setup to generate/receive traffic:

- a. Click on the Device Status button located on the upper right corner:



Test Methodologies for DoS and DDoS

- b. In the new screen select the physical ports that are to be used in the test:

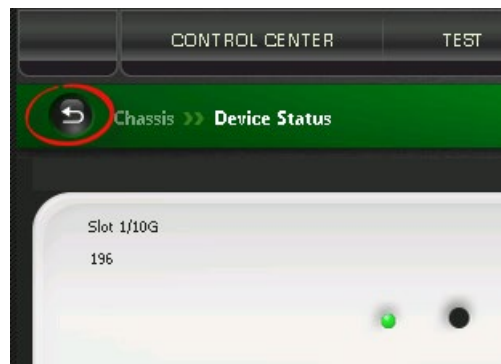


In this example, we will use physical ports 0 and 1 from the PerfectStorm One appliance.

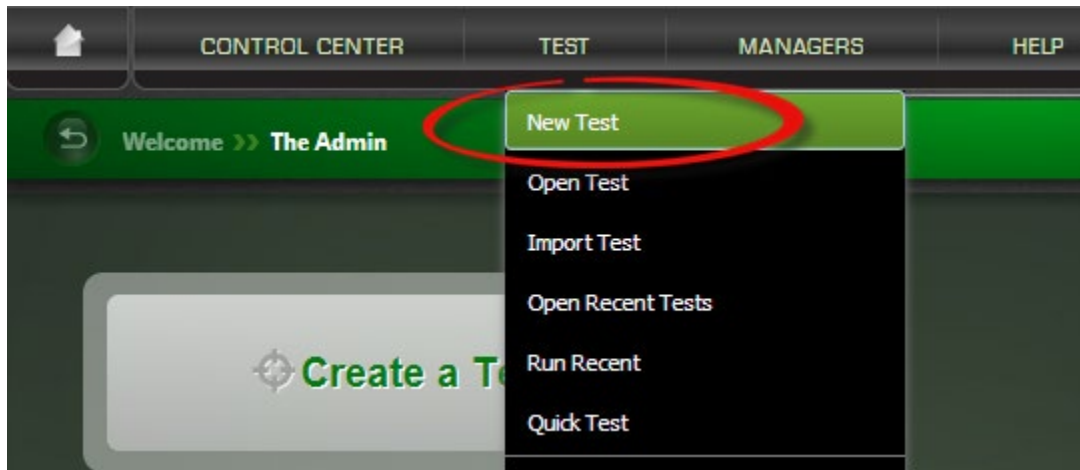
Note: An important thing to remember when reserving your ports is the order in which you reserve them. Whenever you reserve a port, the system will automatically map that port to a logical interface: first reserved port will be Interface1, the second reserved port will be Interface2, and so on. The logical interface parameters (MAC and IP address along with other parameters) will be configured as part of the Network Neighborhood, as we will see below.

Note: The resources of each blade are allocated in proportion to the number of ports reserved on the blade. In some cases, it may be necessary to reserve additional ports to secure enough resources for the test being performed. The sessions and bandwidth available to a test may be insufficient to adequately perform the test if too few ports have been reserved.

- c. Once the proper test ports have been selected, click on the back arrow to return to the initial screen:

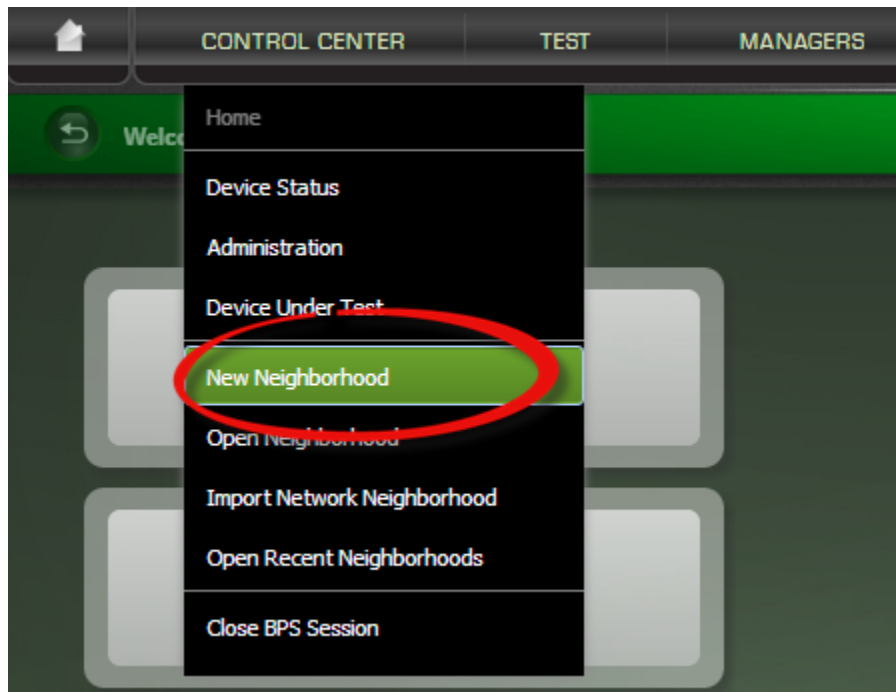


- Next, select **Test** -> **New Test** option from the upper menu bar to start configuring the test:



- Since we already reserved the physical test ports (and, if applicable extra ports for more resource reservation) now we need to configure the MAC and IP layer parameters like MAC address, VLANs, IP addressing scheme, MTU, etc. All these settings, among others are controlled/defined from the **Network Neighborhood**:

- From the upper menu bar select **Control Center** -> **New Neighborhood**:



Test Methodologies for DoS and DDoS

- b. Select the DUT type used for the test and a corresponding default Network Neighborhood will be created:



For this test, we will use *ROUTER* as DUT type.

- c. Enter an easy-to-recall name for the new Network Neighborhood and click **OK**:

The dialog box is titled "Create New Network Neighborhood". It contains a text input field for "New Network Neighborhood Name:" with the text "UDP Flood and TCP SYN Flood" entered. Below the input field is a checkbox labeled "Overwrite existing Network" which is currently unchecked. At the bottom right, there are two buttons: "Cancel" and "OK".

Test Methodologies for DoS and DDoS

- d. According to the DUT type we previously selected, a default Network Neighborhood will be created. Since we chose *ROUTER* DUT type, the following elements are automatically created:
- i. Two *INTERFACES*: Provides access to interface level parameters like MTU, MAC, Impairments, and Packet Filters.
 - ii. One *IPv4 EXTERNAL HOSTS*: configures the IP address of the external end hosts/servers. We will not use this element for our specific test scenario.
 - iii. Two *IPv4 STATIC HOSTS*: Provides access to IP related parameters of the BreakingPoint emulated endpoints. The Static Hosts represents the BreakingPoint emulated DDoS attackers and target servers. In this example, we will use the following:
 - One element for the DDoS attackers with 1000 IP addresses
 - One element with a 10 IP addresses for the emulated servers.

The screenshot shows the configuration interface for a Network Neighborhood. It includes a top navigation bar with 'Entry Mode' and 'Diagram Mode' buttons, and a green bar with 'ADD NEW ELEMENT', 'EXPAND ALL | COLLAPSE ALL', 'KEYBOARD SHORTCUTS', and a lock icon for 'Lock Network Neighborhood to This User'.

The main content area is divided into three sections:

- INTERFACE: (2)** | Untagged Virtual Interface

DEL	ID	Number	MTU	MAC Address	Duplicate MAC Address	VLAN Key	Ignore Pause Frames	Description
×	Interface 1	1	1500	02:1A:C5:01:00:00	<input type="checkbox"/>	Outer VLAN	<input type="checkbox"/>	
×	Interface 2	2	1500	02:1A:C5:02:00:00	<input type="checkbox"/>	Outer VLAN	<input type="checkbox"/>	
- IPv4 EXTERNAL HOSTS: (1)** | External hosts used as a test target
- IPv4 STATIC HOSTS: (2)** | Simulated IPv4 endpoints

DEL	ID	Container	Tags	Base IP Address	Count	Gateway IP Address	Netn
×	Static Hosts i1_default	Interface 1	DDoS Attackers	100.0.0.2	1000	100.0.0.1	8
×	Static Hosts i2_default	Interface 2	Target	200.0.0.2	10	200.0.0.1	8

Note: Make sure to configure the IP addresses according to the physical test setup address assignment scheme. In our example:

- DDoS attackers will be emulated starting with IP address 100.0.0.2, 100.0.0.3, ... 100.0.0.233 (1000 attackers). The gateway used to reach the target network will be 100.0.0.1 (typically the IP address of the public/entry point of the Device/System Under Test). For easy reference configure the Tags field to "DDoS Attackers"
- The attack targets will be emulated starting with IP address 200.0.0.2, 200.0.0.3, ... 200.0.0.11 (10 Servers). The gateway used to reach the public network will be 200.0.0.1 (typically the IP address of the private interface of the Device/System Under Test). For easy reference configure the Tags field to "Target"

Note: In this test case, we are using 1000 IP address as DDoS Attackers just as an example. To emulate a large botnet with tens or hundreds of thousands of zombies the DDoS Attackers count should be increased accordingly. Aside from increasing the actual count, users should also add a Virtual Router Network Neighborhood element to emulate a router in front of all these DDoS attackers, which will provide the following benefits:

Test Methodologies for DoS and DDoS

- Avoid an ARP storm on the interface connected to the DUT/SUT (which can be caused in case many emulated IP addresses are directly connected to the DUT/SUT without a Virtual Router)
- DDoS attackers are not generally directly attached to the DUT/SUT, and they can spawn broad subnet sets

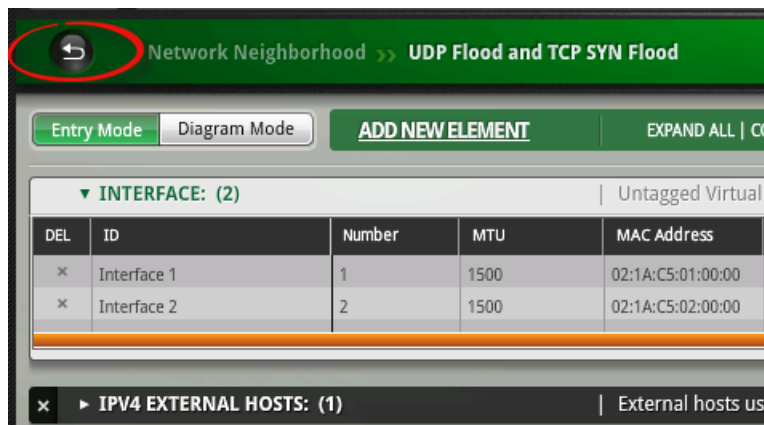
For more details on how to add and configure a Virtual Router Network Neighborhood element, please refer to Appendix A.

Note: When emulating certain volumetric DDoS attacks (like UDP Flood, TCP SYN Flood, TCP ACK Flood, etc.) together with legitimate application traffic, to emulate a real environment as close as possible it is recommended to use an External Host network neighborhood element (containing one or a small subset of the same IP addresses used as the destination of the legitimate application traffic) as the destination of the DDoS attack traffic (i.e. in the Component Server tags).

- e. Save the new Network Neighborhood configuration by clicking the **Save** button located on the lower right corner:

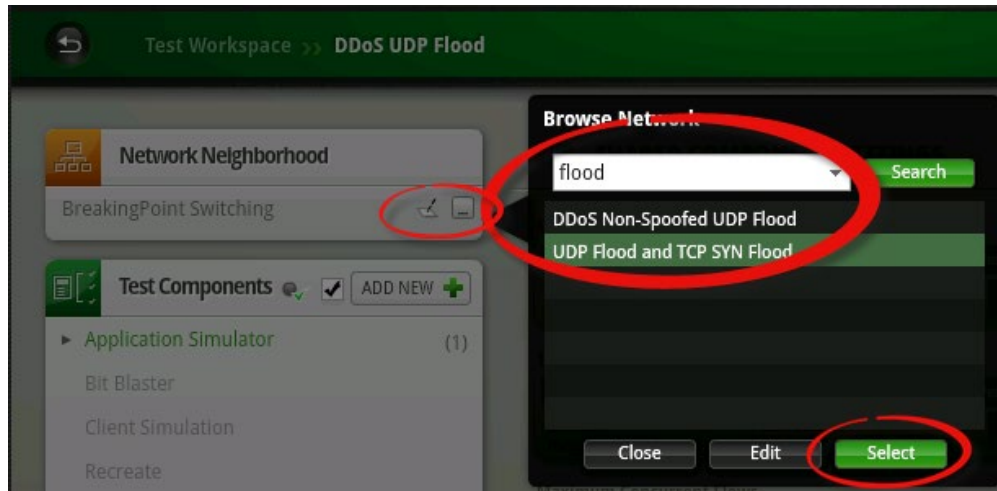


- f. Click on the back arrow to return to the main test screen:



Test Methodologies for DoS and DDoS

- g. From the main test screen click on the browse button from the **Network Neighborhood** section to search and select the above created Network Neighborhood:



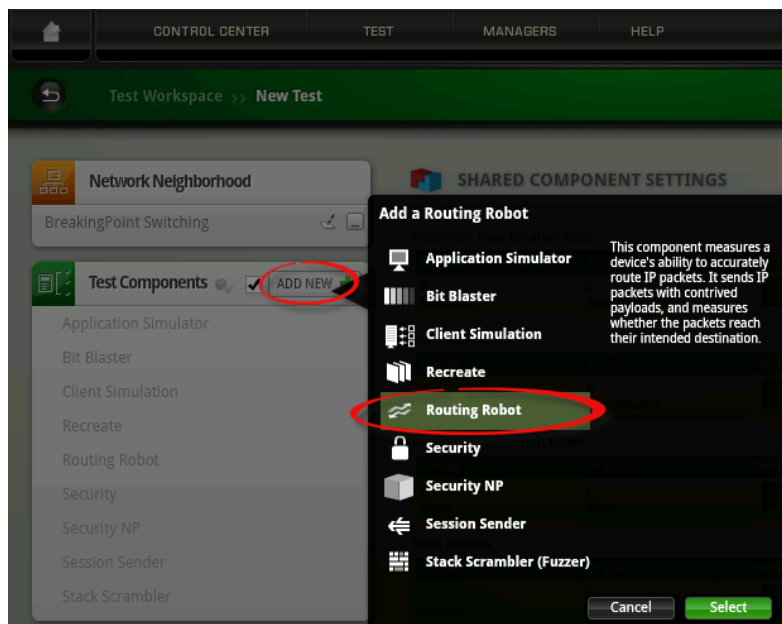
After configuring the MAC and IP layer parameters the actual traffic test component needs to be created.

The Test Component is the entity that controls the traffic patterns. Due to its extreme flexibility, similar traffic patterns can be emulated using different Test Components, each with its own set of advantages. For this example, we will use the following test component:

Routing Robot to generate UDP Flood

Session Sender to generate TCP SYN Flood

5. Click on the **ADD NEW** button from the **Test Components** section. Choose *Routing Robot* from the selection list and click on **Select** button:



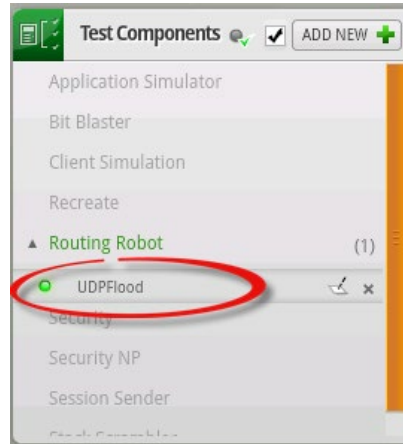
Test Methodologies for DoS and DDoS

6. Rename the component from *RoutingRobot_1* to *UDPFlood* and click Create button:

A new entry (i.e. *UDPFlood*) will be created under **RoutingRobot** Test Component.

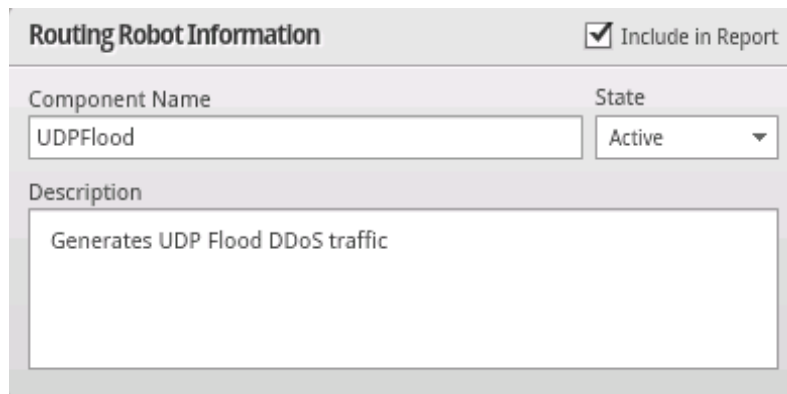
Routing Robot Test Component can be used to generate network based L2/4 DDoS attacks, and since it is leveraging dedicated hardware to generate the traffic, it can easily reach line rate. It features pre-defined packet templates (UDP, TCP, ICMP Echo Request, ICMP Echo Replay, TCP Syn Flood), as well as Packet Mix Distribution, User Defined Fields, and many others.

7. Click on the newly created component to edit its parameters:



8. In the next steps, we will configure the Routing Robot test component:

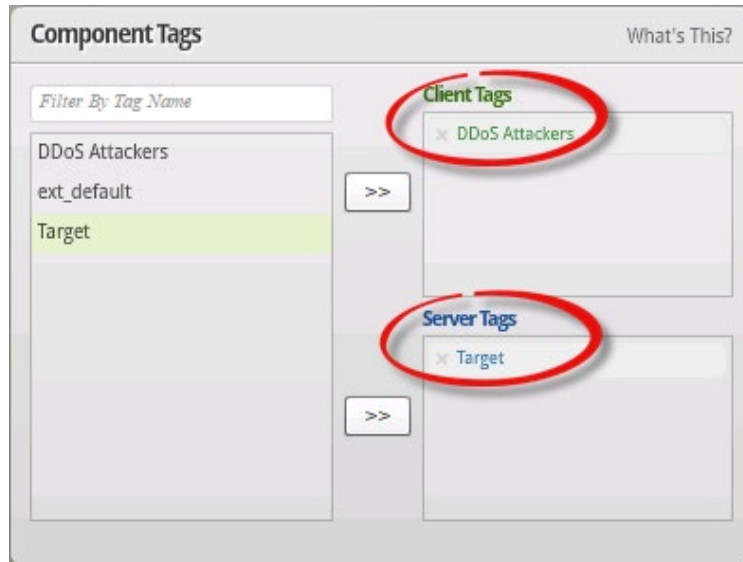
- a. A meaningful description can be added in the **Description** box for easy reference of what this particular component is configured for:

A screenshot of a configuration window titled "Routing Robot Information". It has a checkbox labeled "Include in Report" which is checked. Below this, there are two fields: "Component Name" with the value "UDPFlood" and "State" with a dropdown menu showing "Active". Below these is a "Description" field with the text "Generates UDP Flood DDoS traffic".

- b. In the **Component Tags** section make sure to assign the proper interface tags.

For the Client Tags remove the existing tags and assign the tag corresponding to the Attackers as configured in the Network Neighborhood. For the Server Tags remove the existing tags and

assign the tag corresponding to the Target, emulating the victims as configured in the Network Neighborhood:

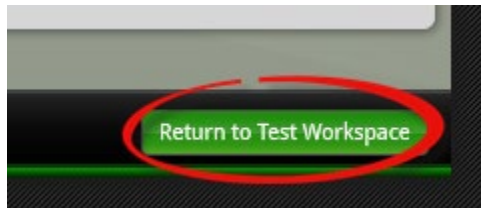


- c. The **Parameters** section contains multiple test component attributes that are grouped into category folders based on their functional purpose. For the scope of this test the following parameters will be modified:
 - i. **Test Duration Measure by a Time Interval:** configure the duration of the UDP Flood traffic to 10 minutes.
 - ii. **Data Rate Type:** set it to *Constant* to get a fixed traffic level.
 - iii. **Minimum Data Rate:** configure the UDP Flood data rate at the required value according to the particularities of your tested environment. In this example, we will set it to 500 Mbps.
 - iv. **Size Distribution:** the parameters in this section define frames/packet size generated by this component (including IMIX distributions if needed). For this test, we will configure a *Constant* frame size of 512 Bytes.
 - v. **Packet Templates:** set the packet definition as a quick method to generate traffic with several different packet types:
 - **Type:** UDP. The other available packet templates are: TCP, ICMP Echo Request, ICMP Echo Reply, TCP SYN.
 - **Delay Start:** useful when the DDoS traffic starts after a period of running legitimate baseline traffic. For this test, we can configure 0s as delay.
 - **Source Port Modifier:** *Random*. This will help have a wide distribution of the Attackers Source Port.

- **Destination Port Modifier:** *Random*. Configuring a random destination port will help in getting an unpredictable destination port which can cause additional overhead on some DDoS mitigation solution. Another option is to have the destination port configured to a fixed value, mapped to a real DNS service available in the target network (in this case configure the **Destination Port Modified** to *Constant* and the actual DNS port in the **Destination Port** field)
- **Slow Start:** *Unchecked*. Specifies whether the component can send a small amount of traffic to the DUT before ramping up to the full rate of the test. This allows switching devices to identify which port to send test traffic.
- **Maximum Stream Count:** *1000*. (Can be thought of as the total number of IPs). This override parameter sets the minimum and maximum number of streams to use for this component. If requested MAC/IP addresses are not symmetric, the number of streams can exceed the Maximum Stream Count.
- IP Address Algorithm: *Random*

For this test, the remaining parameters can be left to their default values.

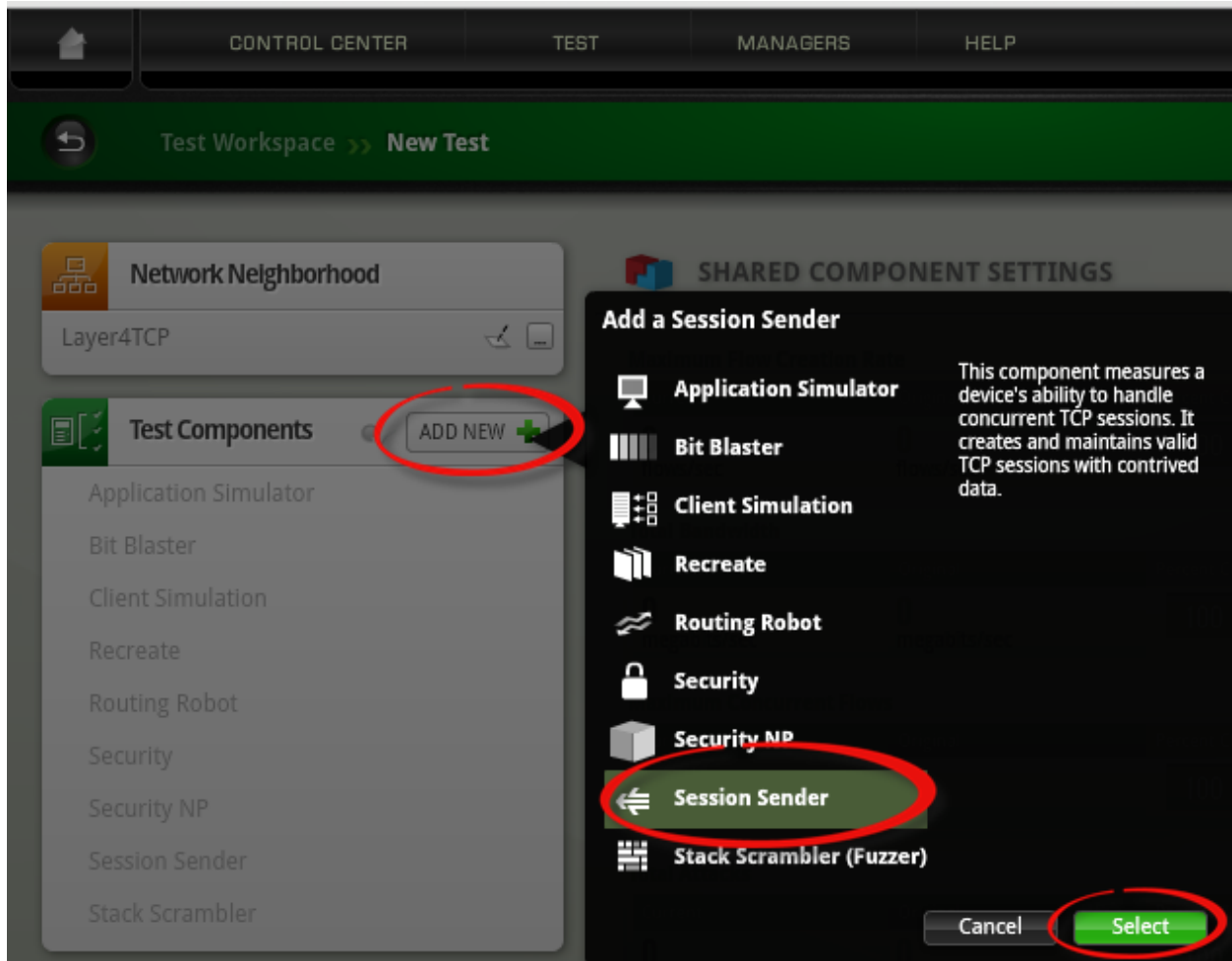
- d. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:



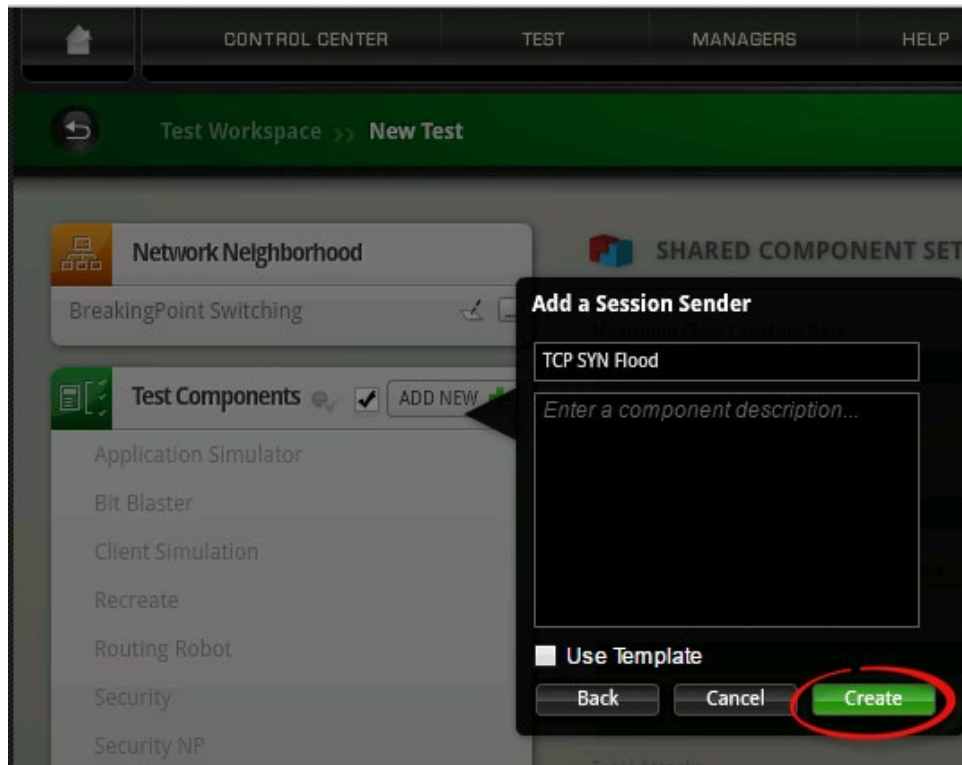
Now that we configured the component to generate UDP Flood traffic we can configure another test component to generate the TCP SYN flood attack.

Test Methodologies for DoS and DDoS

9. Click on the **ADD NEW** button from the **Test Components** section. Choose *Session Sender* from the selection list and click on **Select** button:

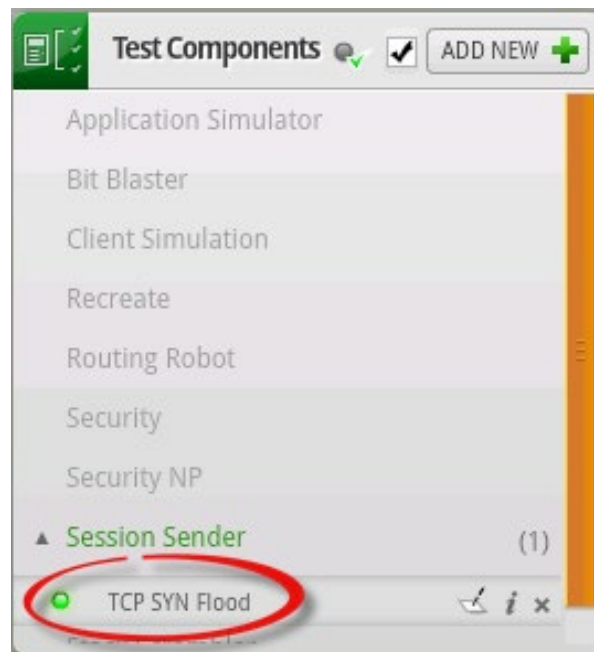


10. Rename the component from *Session Sender_1* to *TCP SYN Flood* and click Create button:



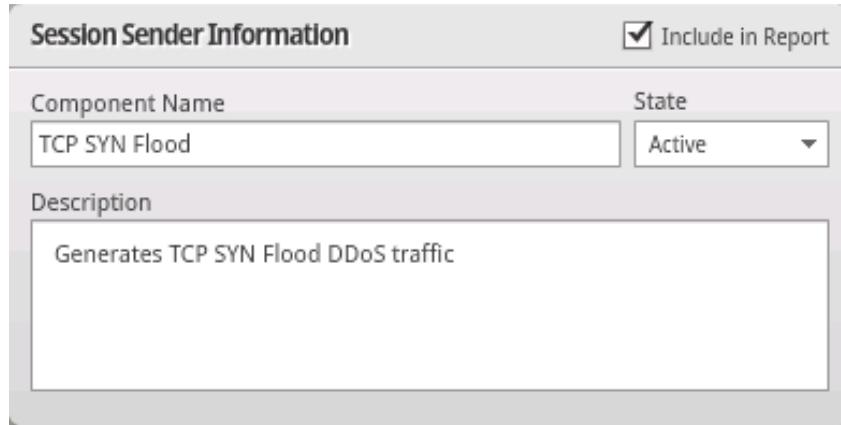
A new entry (i.e. *TCP SYN Flood*) will be created under **Session Sender** Test Component.

11. Click on the newly created component to edit its parameters:



12. In the next steps, we will configure the TCP SYN Flood test component:

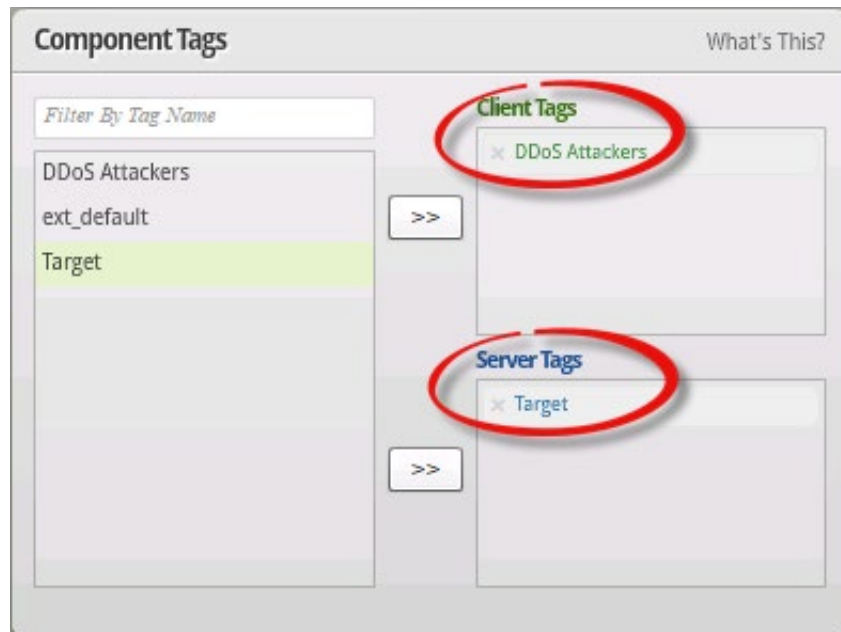
- a. A meaningful description can be added in the **Description** box for easy reference of what this particular component is configured for:



The screenshot shows a configuration window titled "Session Sender Information" with a checked "Include in Report" option. It contains a "Component Name" field with "TCP SYN Flood" and a "State" dropdown menu set to "Active". Below these is a "Description" text area containing the text "Generates TCP SYN Flood DDoS traffic".

- b. In the **Component Tags** section make sure to assign the proper interface tags.

For the Client Tags assign the tag corresponding to the TCP SYN flood attackers as configured in the Network Neighborhood. For the Server Tags assign the tag corresponding to the Target emulating the victims as configured in the Network Neighborhood:



The screenshot shows the "Component Tags" configuration window. On the left, there is a list of available tags: "DDoS Attackers", "ext_default", and "Target". The "Target" tag is highlighted. On the right, there are two sections: "Client Tags" and "Server Tags". The "Client Tags" section contains the tag "DDoS Attackers", and the "Server Tags" section contains the tag "Target". Red circles highlight the "Client Tags" and "Server Tags" sections.

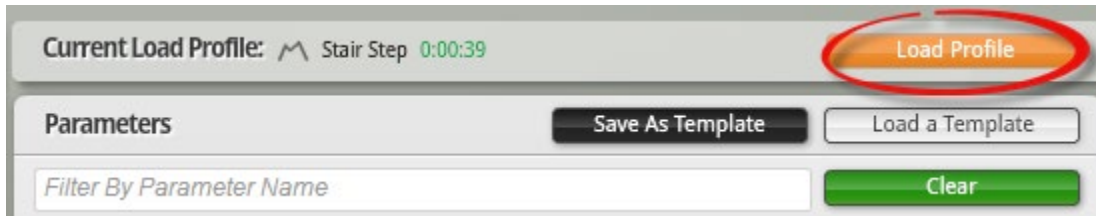
- c. The **Parameters** section contains multiple test component attributes that are grouped into category folders based on their functional purpose. For the scope of this test the following parameters will be modified:
- i. Transport: *TCP*
 - ii. **Data Rate**: check the **Unlimited Data Rate** option so that the test will not be limited by the amount of generated throughput. Instead the traffic load will be controlled by the number of TCP SYN per second as configured below.
 - iii. **Maximum Simultaneous Superflows**: Configure this value to the total number of TCP SYN sent by the end of the test. If we want to generate 10000 TCP SYNs per second and have this test running for 5 minutes than the total number of TCP SYN initiated would be $300 \text{ (seconds)} \times 10,000 \text{ (TCP SYN)} = 3,000,000$.
 - iv. **Maximum Super Flows Per Second**: set the value to the maximum desired value (for this test we will set it to *10,000*).
 - v. **Source Port**: *Random* with a Minimum and Maximum Port Number to *1024* to *65535*. This will help have a wide distribution of the Attackers Source Port.
 - vi. **Destination Port**: *Constant* with a Minimum Port Number to *80* (or a known open port on the device).
 - vii. TCP Configuration – **Retry Count**: 0. It will disable the TCP Retransmission so a very accurate TCP SYN packet rate will be generated.
 - viii. **Raw Flags**: -1. For this test, we will disable this functionality, however it is a very important parameter that provides flexibility to generate a large number of TCP-related attacks by allowing the user to force TCP flags to a particular value, such as a decimal number. Different combinations like TCP SYN, TCP ACK, TCP SYN+FIN, TCP SYN+ACK and many others can be configured.
 - ix. **Payload Packets per Session**: change the value from 20 to 0 (Specifies how many data packets are sent during an open session, which for a TCP SYN only test won't be the case).
 - x. **Delay Start**: 5 min. Usually Delay Start is extremely helpful when testing with real application data so that the TCP SYN Flood attack would start after a period of time. However, in this case we will use Delay Start to have the initial part of the test running only UDP flood and after 5 min into the test, the TCP SYN flood will be

Test Methodologies for DoS and DDoS

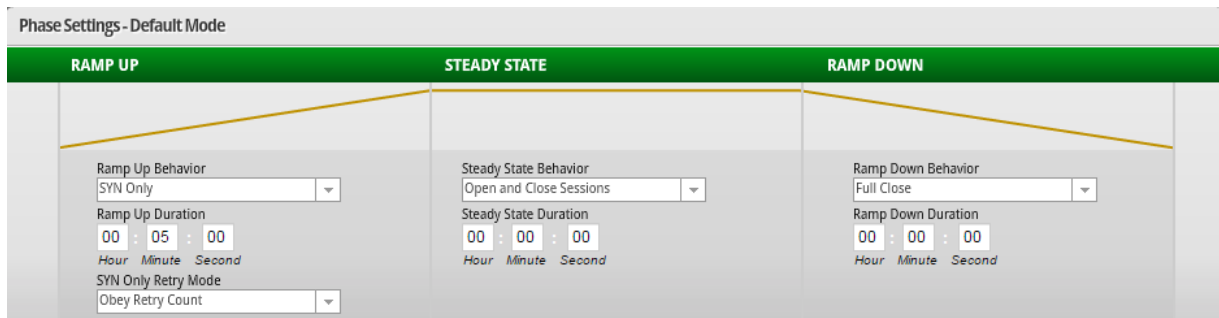
triggered as well. It will help validated the anti-DDoS solution capability of coping with a layered, offset DDoS mixed type.

For this test, the remaining parameters can be left to their default values.

- d. Configure the test traffic pattern using the Load Profile section:
 - i. Click on the **Load Profile** button:

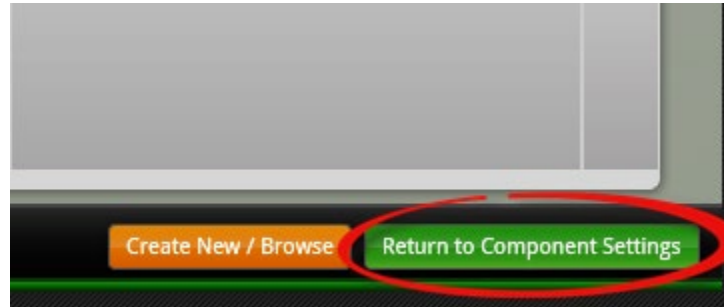


- ii. Change the Ramp Up Behavior to SYN Only
 - iii. The Ramp Up Duration is the actual duration of the test (since only TCP SYN packets would be generated). Enter a value of 5 Minutes.
 - iv. Steady State Duration needs to be configured to 0s since we are not running into TCP data exchange stage.
 - v. Similarly, Ramp Down Duration needs to be configured to 0s as well for the same reasons.

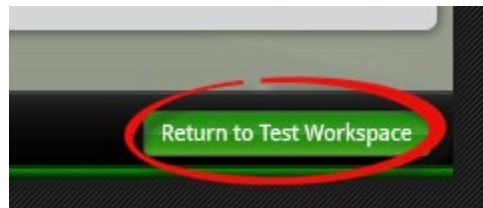


In this example, the Session Sender component will behave in a uniform manner, generating 10,000 TCP SYN packets per second for the entire Ramp Up Duration (i.e. 5 Minutes).

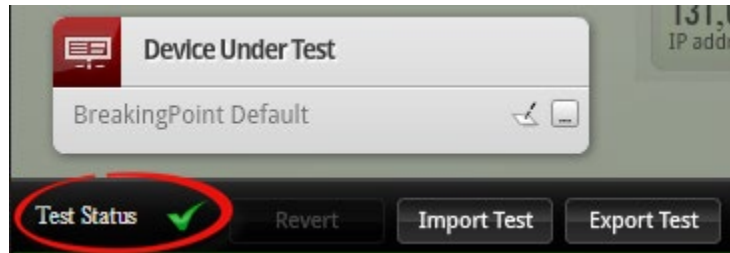
- e. Click on the **Return to Component Setting** button to go back to the Test Component configuration screen:



- f. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:



13. Make sure the **Test Status** indicated (on the lower left corner) has a green checkmark:



If there is not, determine what is wrong by selecting Test Status and viewing the errors.

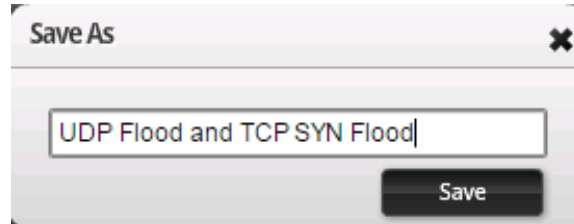
Note: Depending on the previously configured test objectives and the number of physical test interfaces selected, a warning sign (⚠️) might appear next to the Test Status. This might indicate that:

- Not all test interfaces are selected therefore not all test blade resources will be available for the test.
- Total sending bandwidth capacity exceeded (and Interfaces being oversubscribed) caused by the fact that “unlimited” data rate was selected for the test component. This should not represent an issue for this type of tests.

14. Select **Save and Run** from the lower right corner:



15. If the test has not previously been saved, enter a name for the test and click **Save**:



BreakingPoint offers a broad suite of statistics to monitor various KPIs relevant for an extended set of different tests. Refer to the Result Analysis section to analyze the results for this test.

Results Analysis

The main objective of this test is to validate that the DDoS mitigation solution can detect and protect against various DDoS type of attacks. It is very important to read and interpret below statistics in the context of the entire system configuration, hence in accordance to the DUT/SUT type, mitigation capabilities, configured thresholds, etc.

Real-time Statistics

After the test is started and initialized, the screen will automatically switch to Real Time Statistics window.

The **Summary** tab presents basic metrics that provide a good understanding of the overall test progress.

Test Methodologies for DoS and DDoS



The most relevant stats for our test case being:

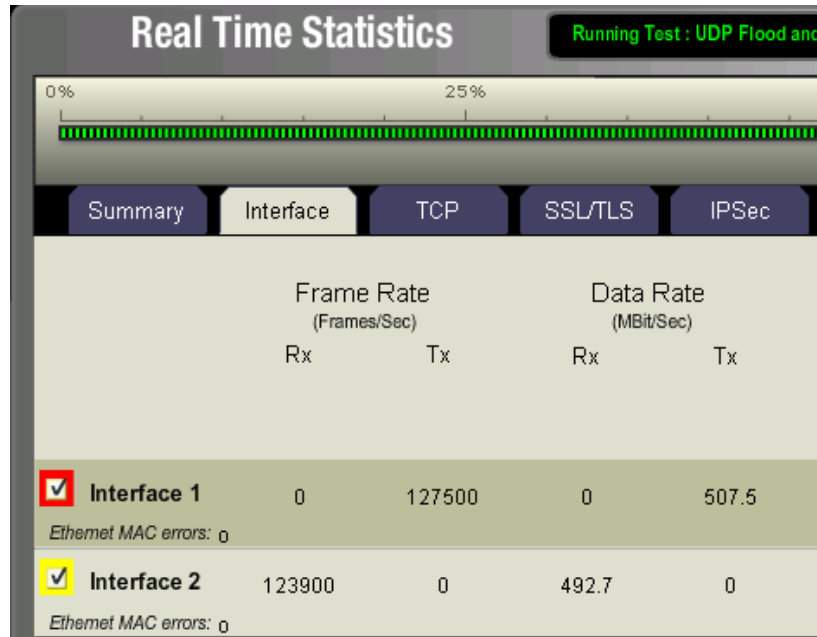
Frame Rate: As long as Tx is equal to Rx, it means that there is no traffic being dropped by any intermediary device. For the first 5 min of the test, since the TCP SYN flood attack did not start yet, these stats are recording only the UDP Flood traffic.

TCP Connection Rate is important for the TCP SYN flood attack. Five minutes into the test, we can see the TCP SYN flood attack initiated at a rate of 10,000 TCP SYNs per second:

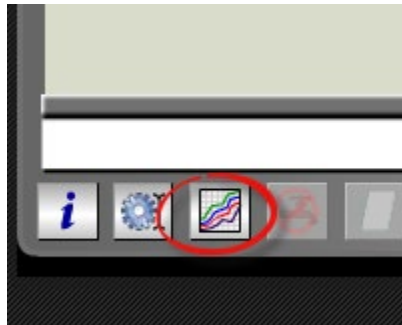
TCP Connection Rate	
Client	Server
Attempted: 10000	Established: 0
Established: 0	Closed: 0
Closed: 0	

Another important tab is the **Interface** tab: it helps investigate the amount of traffic generated in each direction and how much made it to the other side.

Test Methodologies for DoS and DDoS

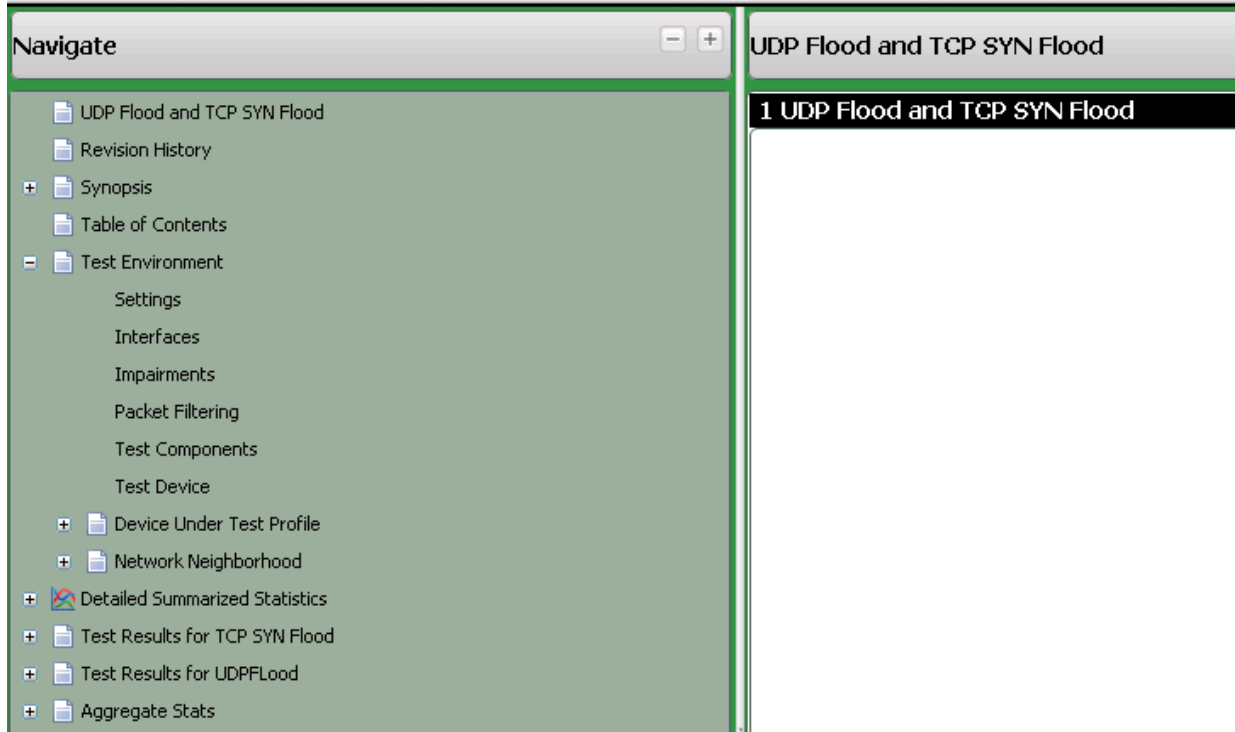


Beside the real-time statistics, detailed post-test analysis can be performed. In the lower left corner of the Real Time Statistics window, select the graph button to view detailed results. This will open the results in a new browser window.



Test Methodologies for DoS and DDoS

On the left side of the detailed report window is the navigation panel, where you can navigate and browse the results. The results and test information will be displayed on the right side of the browser:



Detailed statistics can be investigated based on each test component, TCP/UDP traffic type, transmit vs. receive traffic, latency values, etc.

Test Variables

BreakingPoint offers the following test configuration parameters that provides the flexibility to simulate a high number of different tests with various volumetric DDoS attack types to assess the DDoS mitigation solution capabilities and the impact over the legitimate traffic profile that the device would experience in a production network.

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
IP Version	IPv4	IPv6, IPsec, DSLite, 6rd, selected Mobility stacks, etc.
DDoS Attack type	UDP Flood, TCP SYN Flood	Ping Flood Attack, PSH-ACK Attack, SYN-ACK Flood Attack, Smurf Attack, UDP Fragmentation, Xmas Tree Flood, etc.

Test Methodologies for DoS and DDoS

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
Benign Traffic	None	Custom application mix that matches the traffic profile from the end customer deployment production network that needs to be validated.

Conclusions

This test methodology demonstrates how to configure BreakingPoint to determine the attack rate and attack throughput that a DDoS mitigation system such as a firewall, UTM, or dedicated anti-DDoS solution can mitigate, while the system under test is being flooded with UDP and TCP SYN attack traffic.

Test Case: Mitigation of Sophisticated Application Layer Attacks

Overview

DDoS attacks are getting more and more sophisticated, and the pressure on DDoS mitigation solutions is increasing accordingly, becoming more effective and much more difficult to detect. DDoS attacks have evolved into targeting legitimate application methods (e.g. SIP Invite) or corner case operations and interactions with such applications (e.g. Slow Post, Slow Loris, etc.).

Objective

The goal of this test case is to measure DUT's capabilities to detect and mitigate application-level DDoS attacks like Slow Loris. In this test, we will also add application traffic in addition to the DDoS attacks to assess the impact on legitimate users of various services like web, voice, or video.

Setup

The current setup consists of two PerfectStorm test ports connected directly to the system under test.

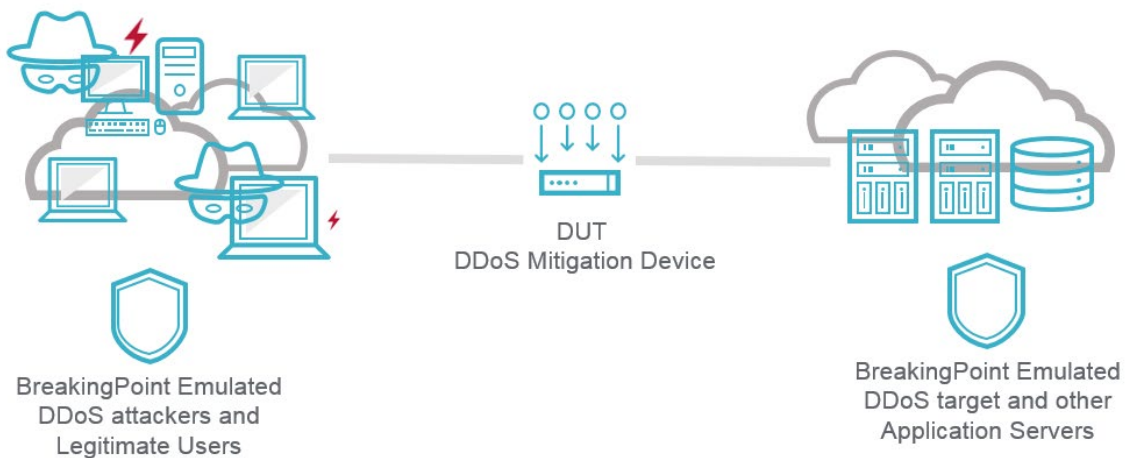


Figure 26. Test topology

In this test topology, BreakingPoint emulates:

- On Port1:
 - BOTNET consisting of 1,000 DDoS Attackers
 - Legitimate Users, consisting of 1 million users
- On Port2:
 - Target Server consisting of one application server under attack

Test Methodologies for DoS and DDoS

- Server Network by placing a small number of Application Servers

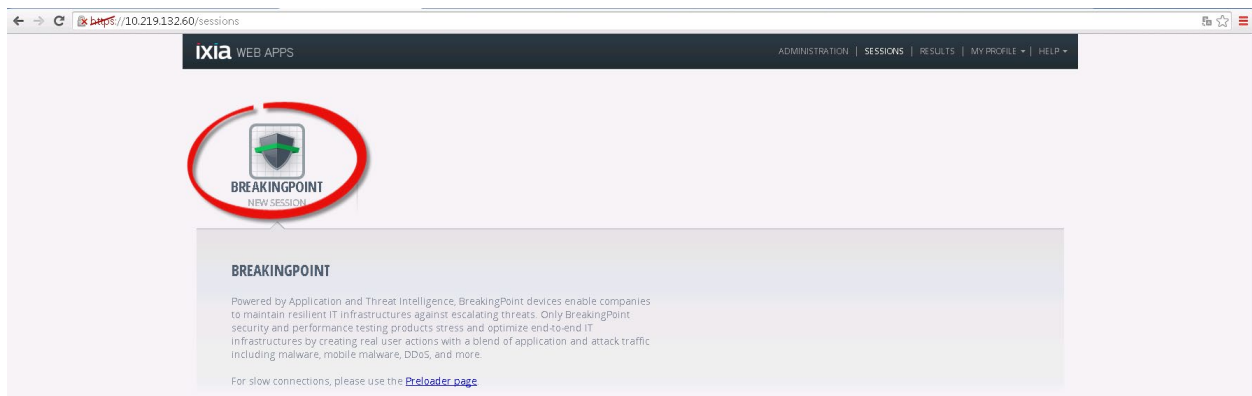
This topology can be used to test intermediate devices such as dedicated anti-DDoS solutions, firewalls, and unified threat management systems.

A real external Application Server can be used as the Target Server, which will further increase the test realism. However, in this situation special consideration should be taken when interpreting the results, since a number of statistics will not include the exhaustive Target Server contribution in the BreakingPoint test reporting.

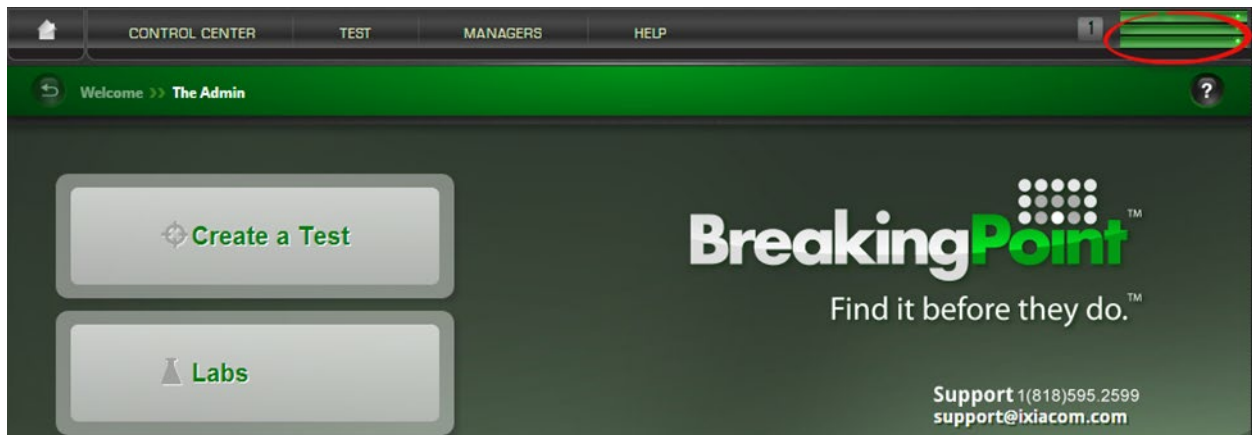
Step-by-Step Instructions

This section provides step-by-step instructions to execute this test using the BreakingPoint tool.

1. Using a web browser, connect to the Ixia Web Apps GUI and start a new BreakingPoint Web Session:

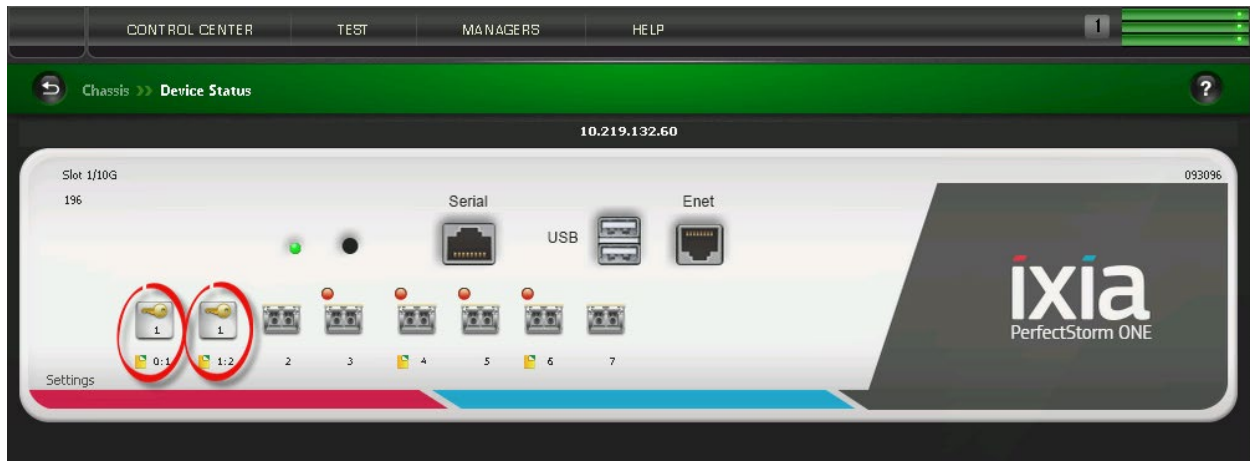


2. Once the BreakingPoint web home page loads, reserve the test ports to be used in the physical test setup to generate/receive traffic:
 - a. Click on the Device Status button located on the upper right corner:



Test Methodologies for DoS and DDoS

- b. In the new screen, select the physical ports that are to be used in the test:

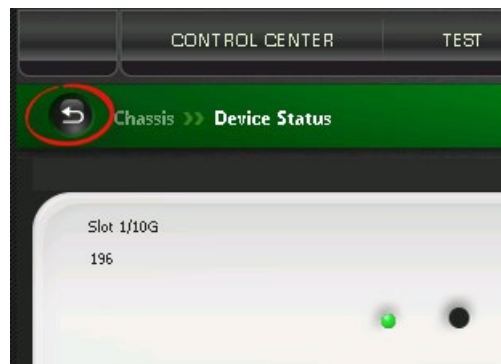


In this example, we will use physical ports 0 and 1 from the PerfectStorm One appliance.

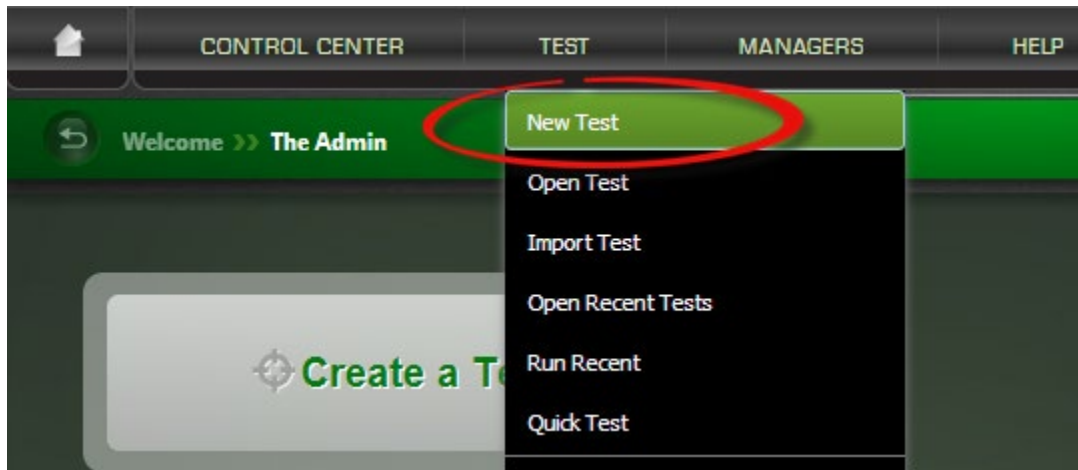
Note: An important thing to remember when reserving your ports is the order in which you reserve them. Whenever you reserve a port, the system will automatically map that port to a logical interface: first reserved port will be Interface0; the second reserved port will be Interface1, and so on. The logical interfaces parameters (MAC and IP address along with other parameters) will be configured as part of the Network Neighborhood as we will see below.

Note: The resources of each blade are allocated in proportion to the number of ports reserved on the blade. In some cases, it may be necessary to reserve additional ports to secure enough resources for the test being performed. The sessions and bandwidth available to a test may be insufficient to adequately perform the test if too few ports have been reserved.

- c. Once the proper test ports have been selected, click on the back arrow to return to the initial screen:

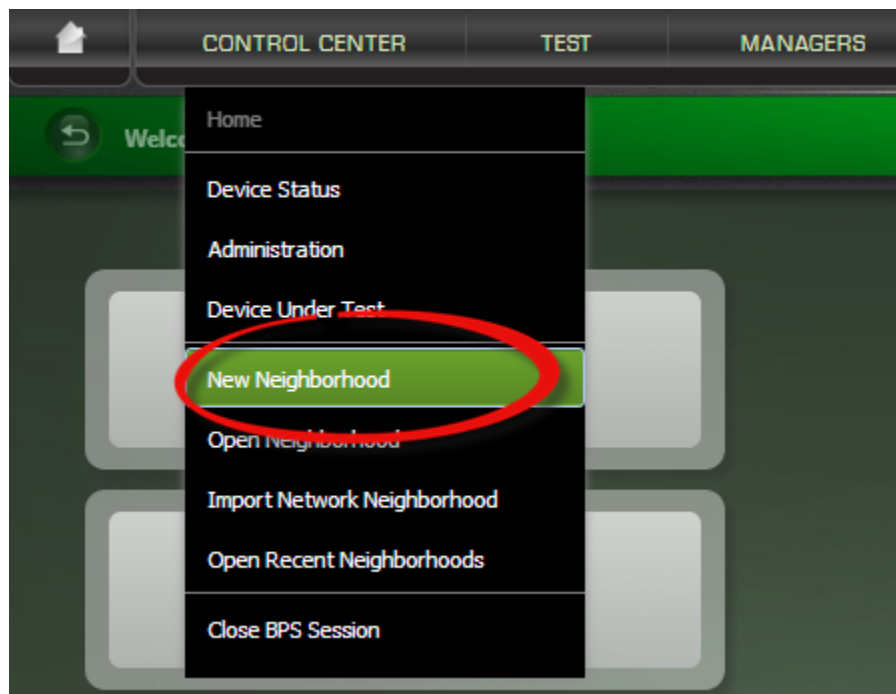


- Next, select **Test** -> **New Test** option from the upper menu bar to start configuring the test:



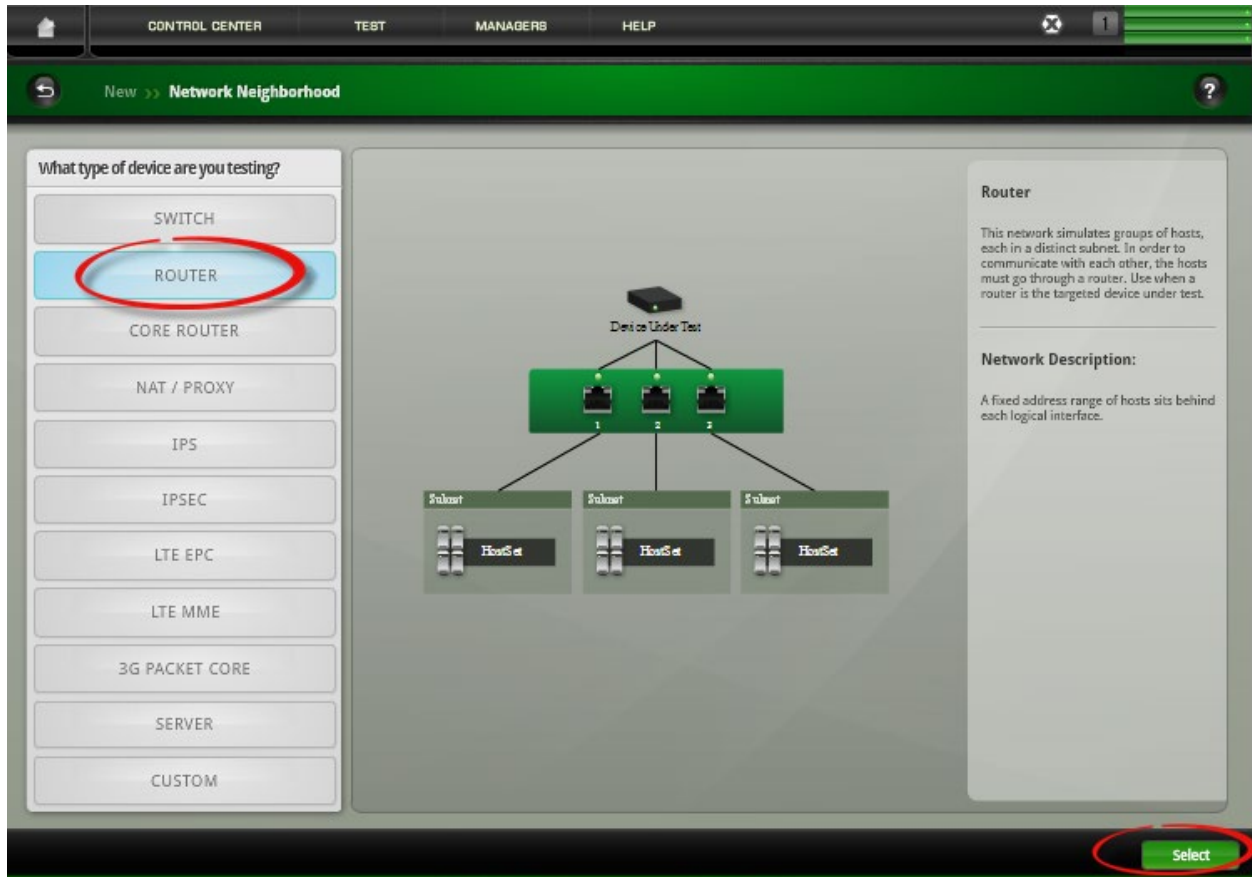
- Since we already reserved the physical test ports (and, if applicable, extra ports for more resource reservation), now we need to configure the MAC and IP layer parameters like MAC address, VLANs, IP addressing scheme, MTU, etc. All these settings, among others are controlled/defined from the **Network Neighborhood**:

- From the upper menu bar select **Control Center** -> **New Neighborhood**:



Test Methodologies for DoS and DDoS

- b. Select the DUT type used for the test and a corresponding default Network Neighborhood will be created:



For this test, we will use *ROUTER* as DUT type.

- c. Enter an easy-to-recall name for the new Network Neighborhood and click **OK**:

The dialog box is titled "Create New Network Neighborhood". It contains a text input field labeled "New Network Neighborhood Name:" with the text "Application DDoS" entered. Below the input field is a checkbox labeled "Overwrite existing Network" which is currently unchecked. At the bottom right, there are two buttons: "Cancel" and "OK".

Test Methodologies for DoS and DDoS

- d. According to the DUT type we previously selected, a default Network Neighborhood will be created. Since we chose *ROUTER* DUT type, the following elements are automatically created:
- i. Two INTERFACES: Provides access to interface level parameters like MTU, MAC, Impairments, and Packet Filters.
 - ii. One IPv4 EXTERNAL HOSTS: configures the IP address of the external end hosts/servers. We will not use this element for our specific test scenario.
 - iii. Four IPv4 STATIC HOSTS: Provides access to IP related parameters of the BreakingPoint emulated endpoints. The Static Hosts represents the BreakingPoint emulated DDoS attackers, legitimate users and target servers. In this example, we will use the following:
 - One element for the DDoS attackers with 1,000 IP addresses
 - One element for the legitimate traffic users with 1 Million IP addresses
 - One element with 10 IP addresses for the emulated legitimate servers.
 - One element with a single IP address for the target server (part of the same subnet as the target network).

DEL	ID	Number	MTU	MAC Address	Duplicate MAC Address	VLAN Key	Ignore Pause Frames	Description
×	Interface 1	1	1500	02:1A:C5:01:00:00	<input type="checkbox"/>	Outer VLAN	<input type="checkbox"/>	
×	Interface 2	2	1500	02:1A:C5:02:00:00	<input type="checkbox"/>	Outer VLAN	<input type="checkbox"/>	

DEL	ID	Container	Tags	Base IP Address	Count	Gateway IP Address	Netn
×	Static Hosts i1_default	Interface 1	DDoS Attackers	100.0.0.2	1000	100.0.0.1	8
×	Static Hosts i2_default	Interface 2	Target Server	200.0.0.12	1	200.0.0.1	8
×	ip_static_hosts 3	Interface 1	Legitimate Users	100.5.0.2	1000000	100.0.0.1	8
×	ip_static_hosts 4	Interface 2	Server Network	200.0.0.2	10	200.0.0.1	8

Note: Make sure to configure the IP addresses according to the physical test setup address assignment scheme. In our example:

- DDoS Attackers will be emulated starting with IP address 100.0.0.2, 100.0.0.3, ... 100.0.3.233 (1000 attackers). The gateway used to reach the target network will be 100.0.0.1 (typically the IP address of the public/entry point of the Device/System Under Test). For easy reference configure the Tags field to "DDoS Attackers"

Test Methodologies for DoS and DDoS

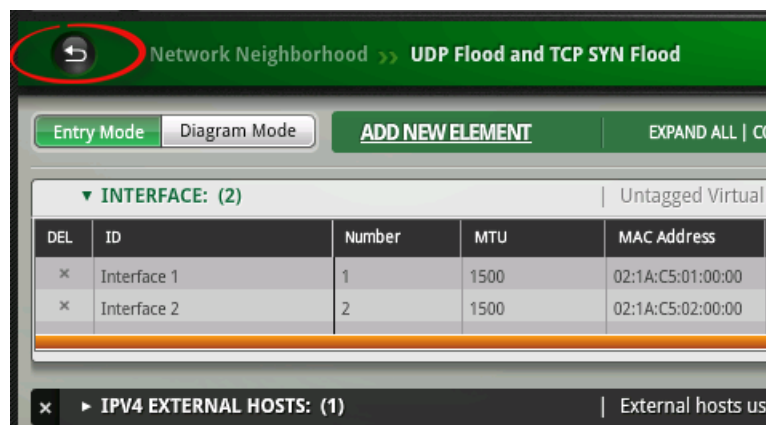
- The Targeted Server will be configured with IP address 200.0.0.12 (the number of target server can be increased to a value to match the particular deployment that is being validated for). The gateway used to reach the public network will be 200.0.0.1 (typically the IP address of the private interface of the Device/System Under Test). For easy reference configure the Tags field to “*Target Server*”
- Legitimate Users will be emulated starting with IP address 100.5.0.2, 100.5.0.3, ... 100.0.25.223 (1 Million legitimate users). The gateway used to reach the target network will be 100.0.0.1 (typically the IP address of the public/entry point of the Device/System Under Test). For easy reference configure the Tags field to “*Legitimate Users*”.
- The Server Network will be configured with IP address 200.0.0.2, 200.0.0.3, ... (10 Application Servers. The number of target server can be increased to a value to match the particular deployment that is being validated for). The gateway used to reach the public network will be 200.0.0.1 (typically the IP address of the private interface of the Device/System Under Test). For easy reference configure the Tags field to “*Server Network*”.

Note: Due to the large number of emulated IP addresses on the public side (1 Million legitimate users and 1,000 DDoS attackers) to avoid a large ARP storm a Virtual Router can be used. See Appendix A for details on how to configure a virtual router.

- e. Save the new Network Neighborhood configuration by clicking the **Save** button located on the lower right corner:

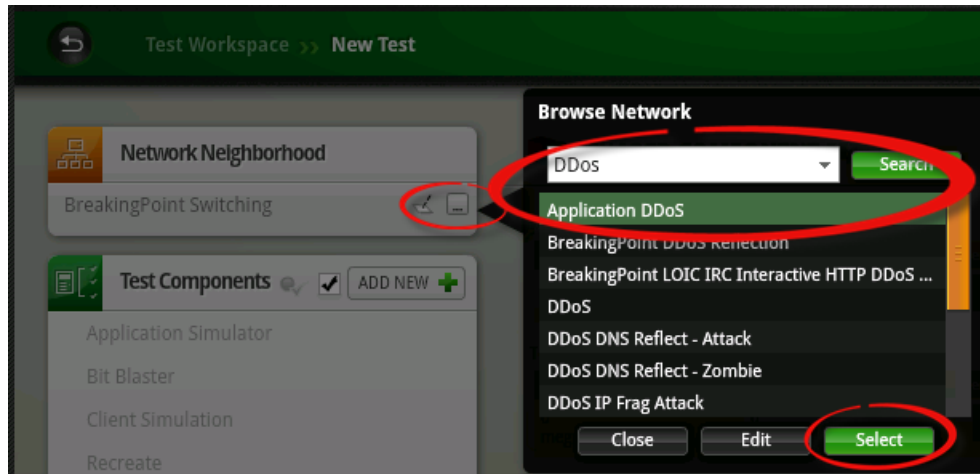


- f. Click on the back arrow to return to the main test screen:



Test Methodologies for DoS and DDoS

- g. From the main test screen click on the browse button from the **Network Neighborhood** section to search and select the above created Network Neighborhood:



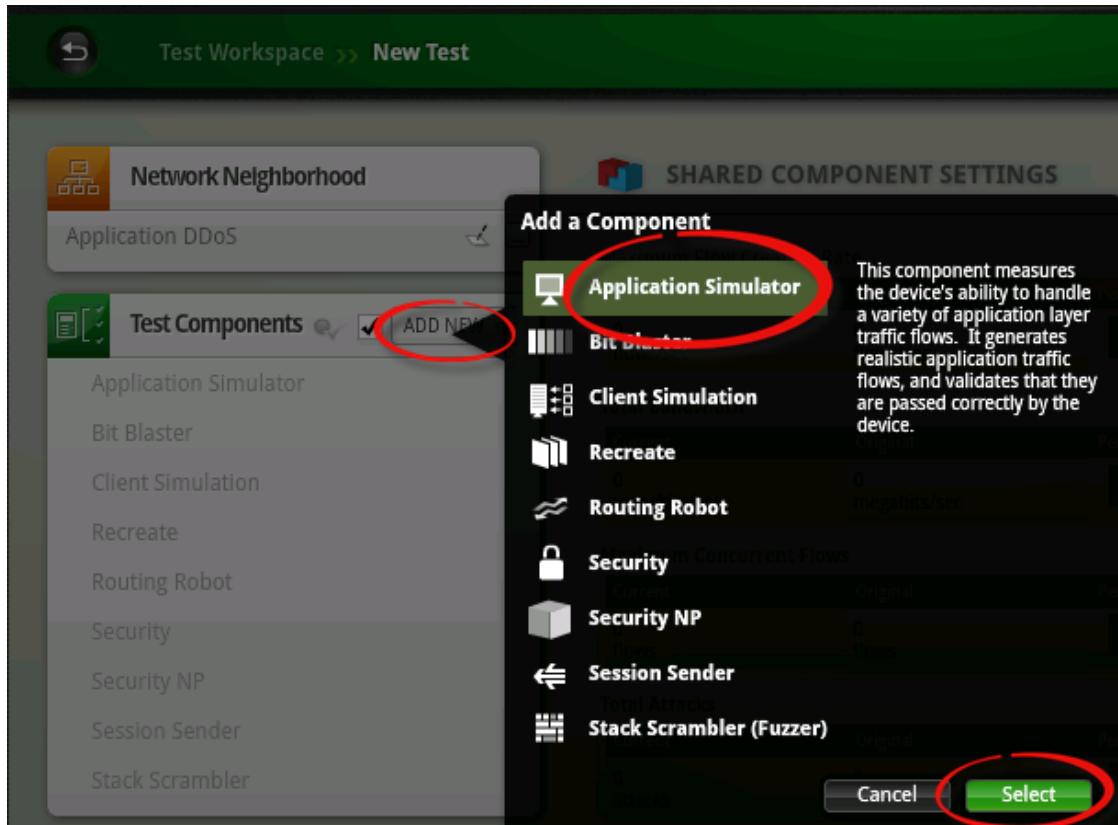
After configuring the MAC and IP layer parameters, the actual traffic test component needs to be created.

The Test Component is the entity that controls the traffic patterns. Due to its extreme flexibility, the same Test Component type can be used for various tasks. For example, in this test, since we are looking to emulate an application-layer DDoS-attack, we will emulate it using AppSim test component. Similarly, since it is highly important to use real application traffic in conjunction with the DDoS attack, we will use another AppSim component (configured in a different way, of course) to emulate the legitimate application traffic. Therefore, for this test methodology we will be using:

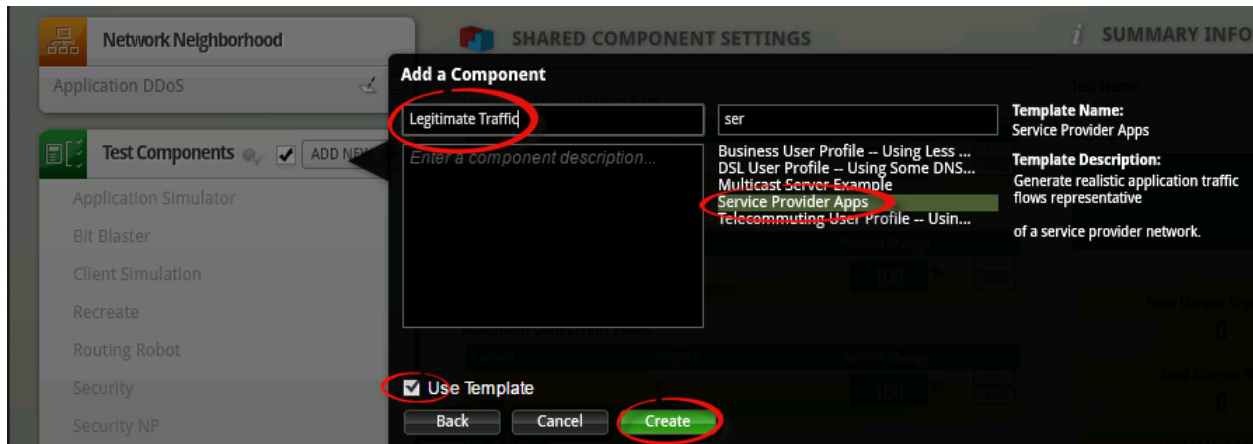
- AppSim1: to generate the legitimate application traffic
- AppSim2: to generate the DDoS attack

Test Methodologies for DoS and DDoS

- Click on the **ADD NEW** button from the **Test Components** section. Choose *Application Simulator* from the selection list and click on **Select** button:



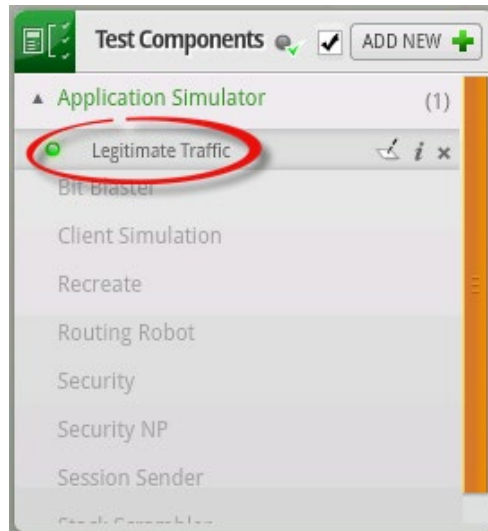
- Tick the **Use Template** checkbox, then select *Service Provider Apps* as a template. Optionally rename the component to something more meaningful like *Legitimate Traffic* and click **Create** button:



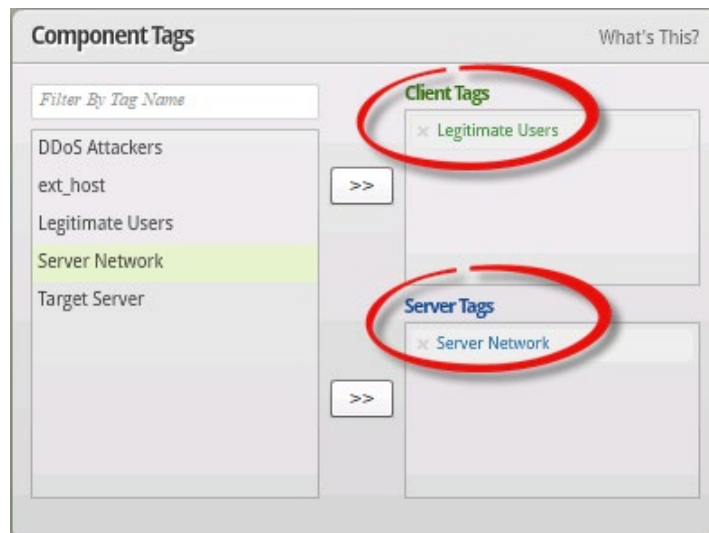
This pre-canned template generates realistic legitimate traffic representative of a service provider network. However, to increase the relevance of this test case the actual application profile should be configured (using the corresponding Superflows) to match as close as possible the traffic mix and distribution seen in the particular production deployment.

Test Methodologies for DoS and DDoS

7. A new entry (i.e. *Legitimate Traffic*) will be created under **Application Simulator** Test Component. Click on the newly created component to edit its parameters:



8. In the next steps, we will configure the AppSim test component:
 - a. In the **Component Tags** section, make sure to assign the proper interface tags:
 - i. For the Client Tags, assign the tag corresponding to the *IPv4 Static Host Network Neighborhood* element emulating the legitimate users.
 - ii. For the Server Tags, assign the tag corresponding to the *IPv4 Static Host Network Neighborhood* element emulating the Server Network:

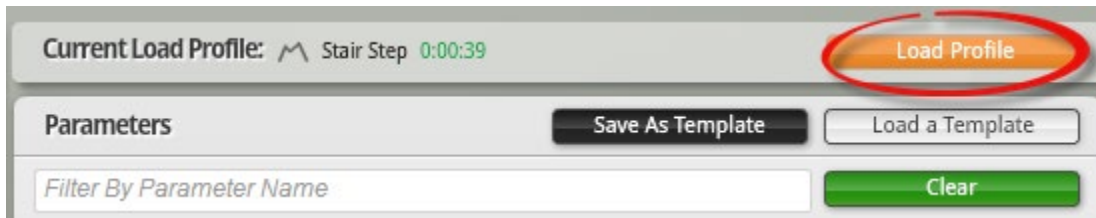


- b. The **Parameters** section contains multiple test component attributes that are grouped into category folders based on their functional purpose. For the scope of this test, since

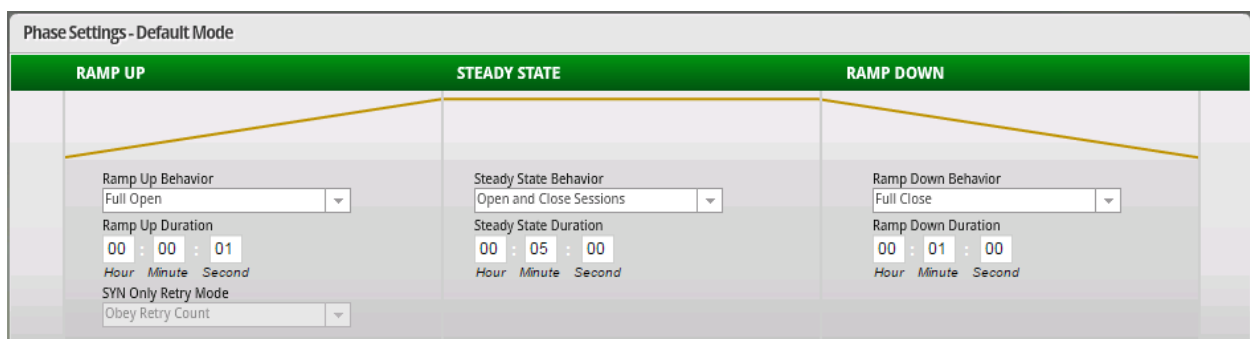
Test Methodologies for DoS and DDoS

we have used a template, most of the parameters can be left with their existing value except the following parameters:

- i. **Data Rate:** check the **Unlimited Data Rate** option so that the test will not be limited by the amount of generated throughput. Instead the traffic load will be controlled by the Maximum Super Flows per second as configured below.
 - ii. **Maximum Simultaneous Superflows:** Configure this value to the total number of simultaneous Superflows targeted to be achieved by the legitimate users. For this example, we will use 200,000.
 - iii. **Maximum Super Flows Per Second:** set the value to the maximum desired value (for this test we will set it to 10,000).
- c. Configure the legitimate traffic pattern using the Load Profile section:
- i. Click on the **Load Profile** button:



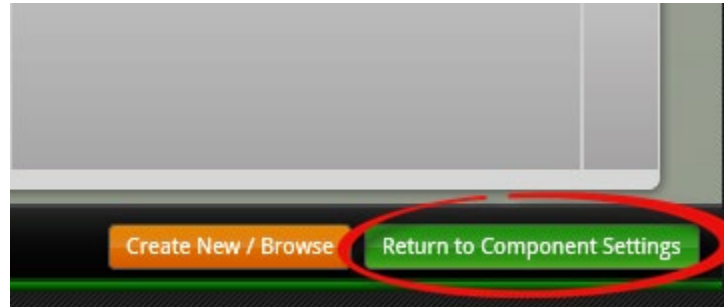
- ii. Change the Ramp Up Duration to 1 seconds.
- iii. Configure Steady State Duration to 5 minutes.
- iv. Configure Ramp Down Duration to 1 minute.



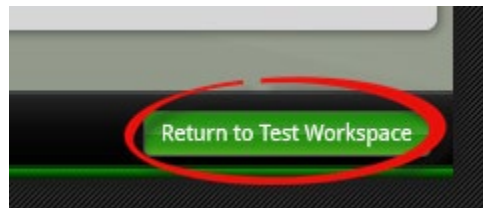
In this example, this AppSim component is configured to generate legitimate traffic for a duration of 6 minutes and 1 second.

Test Methodologies for DoS and DDoS

- v. Click on the **Return to Component Setting** button to go back to the Test Component configuration screen:

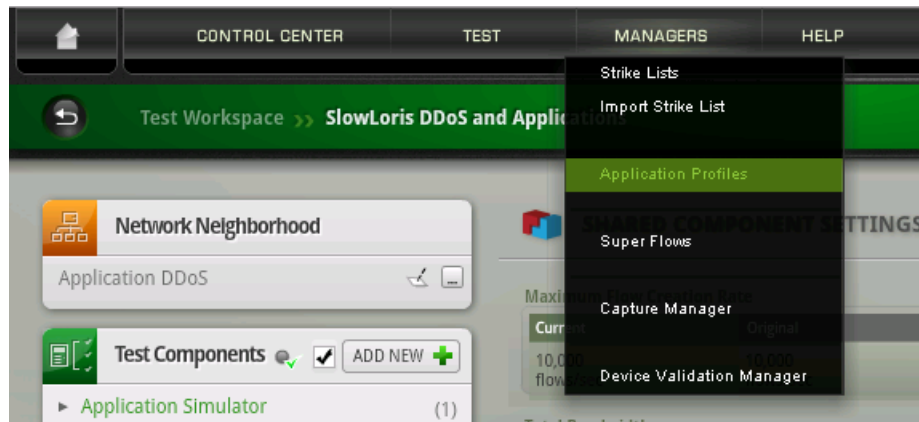


- vi. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:



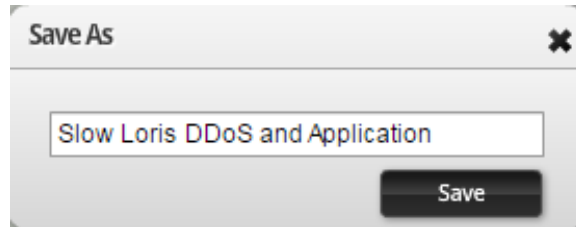
Now that we configured our legitimate application traffic we need to configure the application-level DDoS attacks, in our case Slow Loris. For this we need first to create an application profile containing the corresponding Slow Loris attack (by using a corresponding Superflow) and then generate that application profile using a second AppSim component

- 9. From the upper menu bar select Managers -> Application Profile:

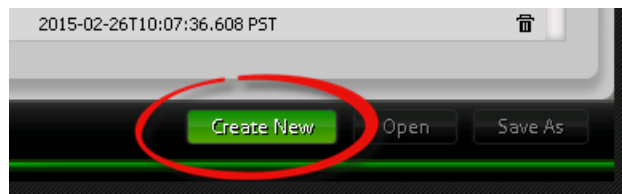


Test Methodologies for DoS and DDoS

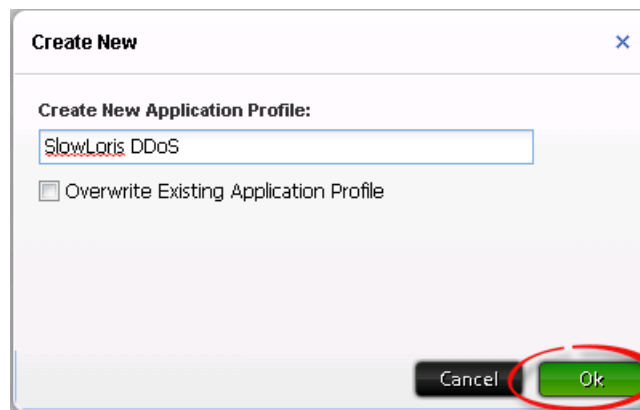
10. If the test has not previously been saved, a new pop-up will be generated to save the test. Hit “Yes” in the first dialog to save the test and then enter a name for the test and click Save:



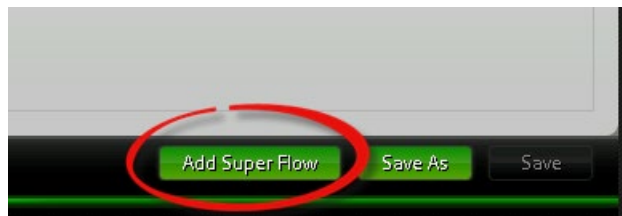
11. Click on Create New button form the lower right corner, to create a new Application Profile:




12. Enter an easy-to-remember name and press the Ok button (in our case, SlowLoris DDoS):



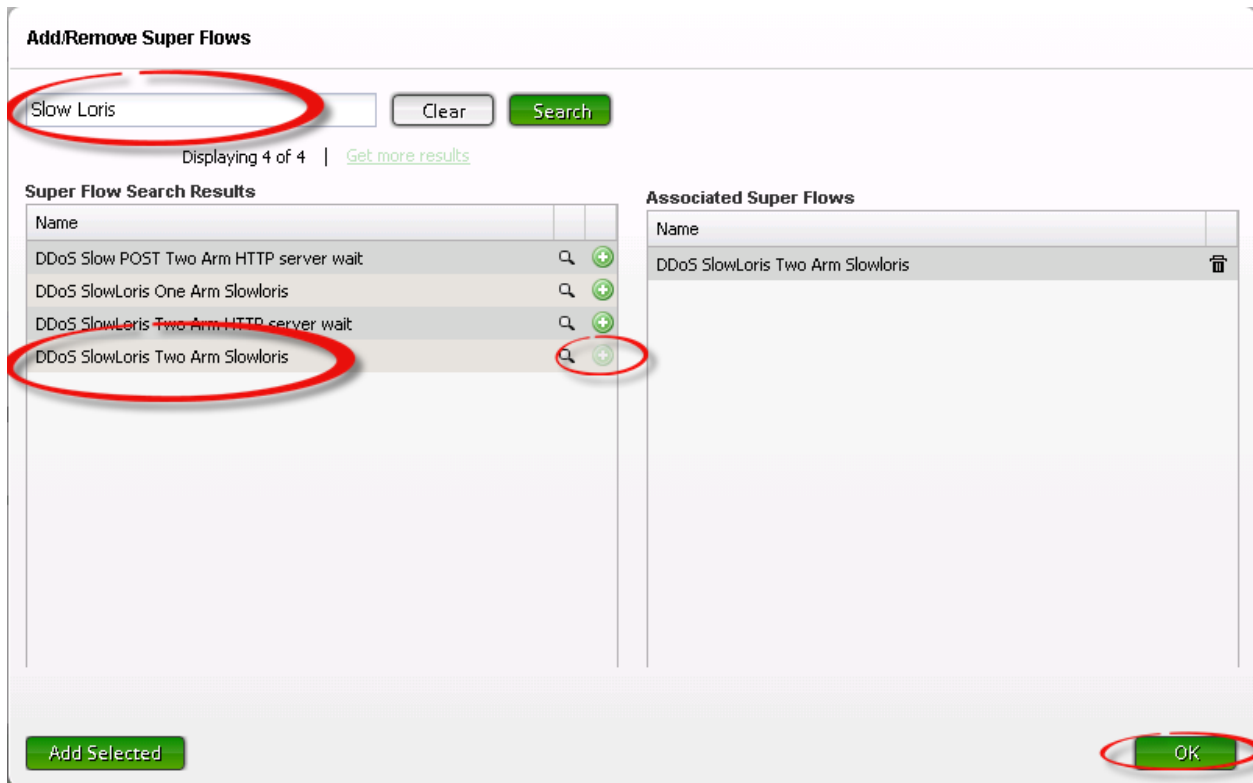
- In the new window click on “Add Super Flow” button form the lower right corner:



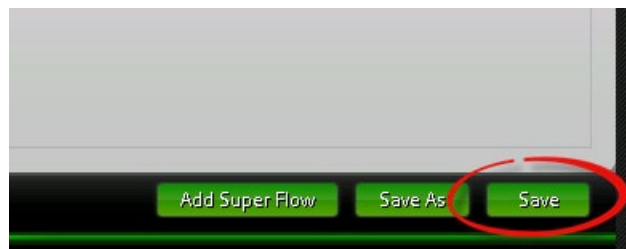
13. In the new pop-up window, we get access to all Superflows available on the system which can be combined in various ways to create a large diversity of custom Application Profiles. For our test, type “Slow Loris” in the search box and add the pre-canned “DDoS SlowLoris Two Arm Slowloris” Superflow by pressing the  button associated with this Superflow.

Test Methodologies for DoS and DDoS

The new selected Superflow will be added to the Associated Superflows pane. Click ok to continue:



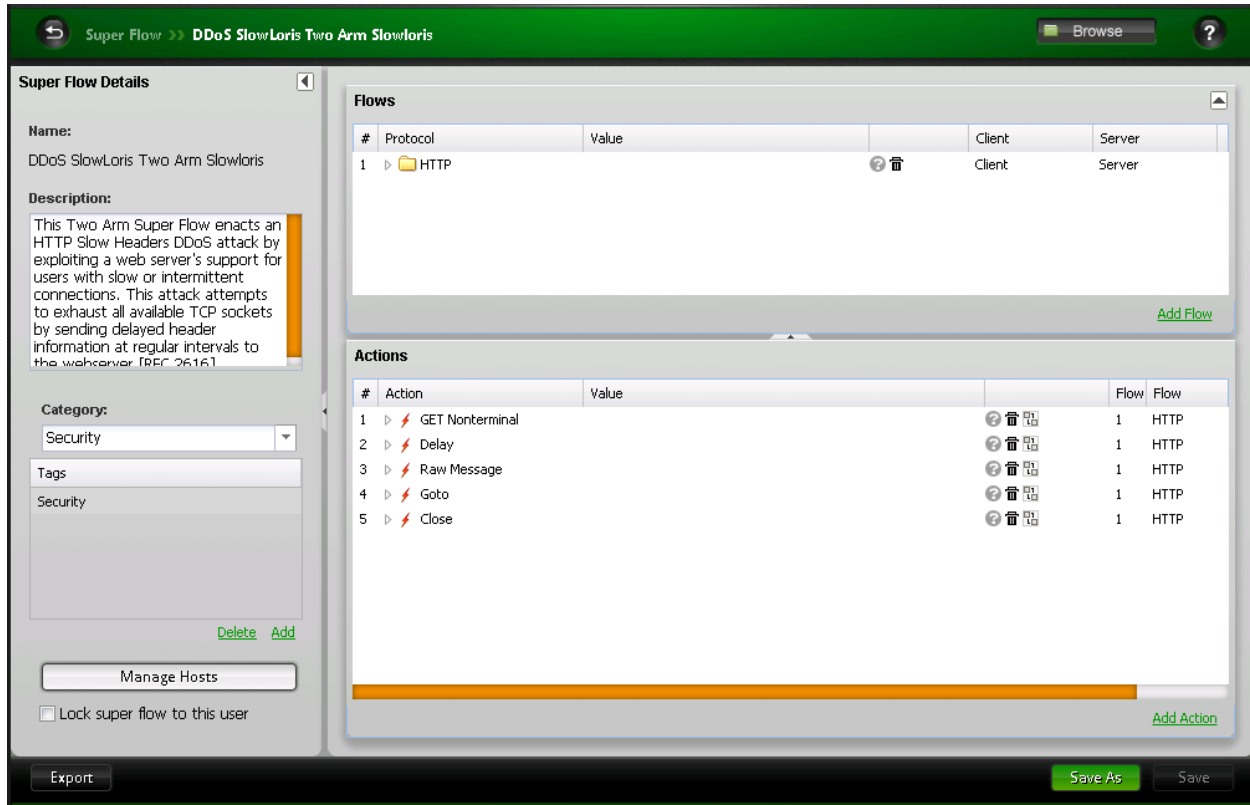
14. Now, the newly created “SlowLoris DDoS” application profile comprises the pre-canned “DDoS SlowLoris Two Arm SlowLoris” Superflow. Press save button from the lower right corner to save the application profile:



15. To see the exact actions that this pre-canned Superflow is executing press the  button:



16. As displayed in the actions pane, this pre-canned Superflow starts with an un-complete “GET” request. It is followed by a looped raw message simulating the transmission of delayed header information at regular intervals to the webserver. This simulates a DDoS attack that attempts to exhaust all available TCP sockets on the webserver.



There are a high number of such pre-canned such Superflows that are emulating various application-level DDoS attack:

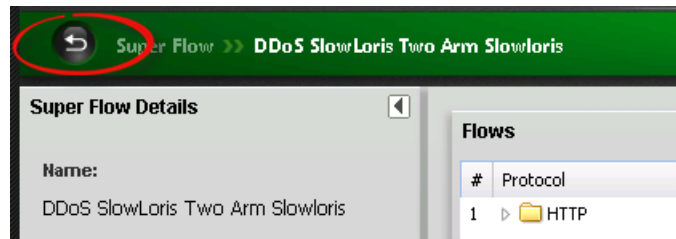
Aside from the canned attacks, BreakingPoint offers the flexibility to customize various DDoS attacks by defining the appropriate actions in custom created Superflows. For example a SIP Invite Flood attack can be easily built by creating a new Superflow with just a “SIP Invite” action.

BreakingPoint architectural approach empowers users` creativity to generate an extremely wide coverage of known application-level DDoS attacks as well as to try various options and variables to unveil new ones that are currently undiscovered.

Now that the application profile containing the DDoS attack has been created, we can return to the initial test pane.

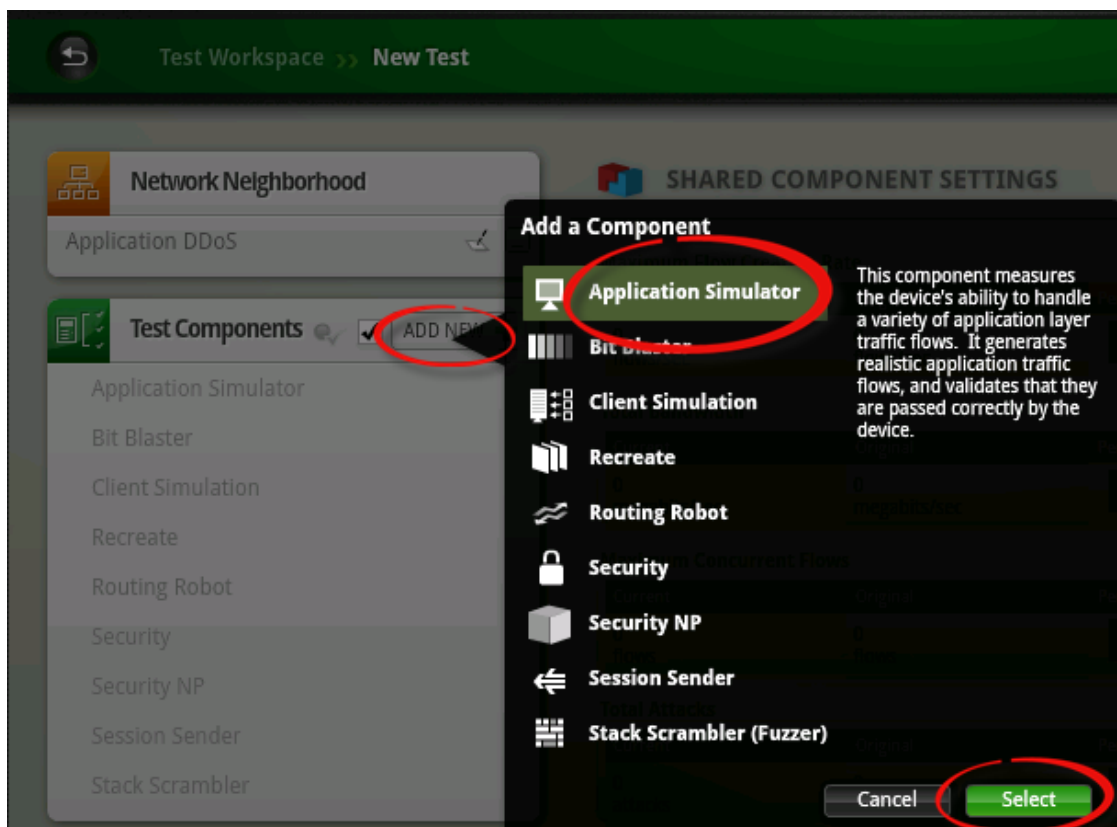
Test Methodologies for DoS and DDoS

17. Click on the back arrow multiple times until returning to the main test screen:

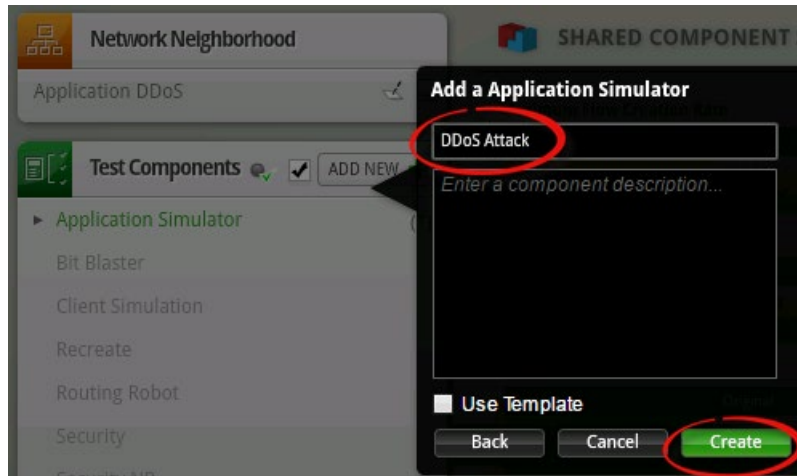


Next, the second Application Simulator test component that emulates the SlowLoris DDoS attack needs to be configured and added to the test:

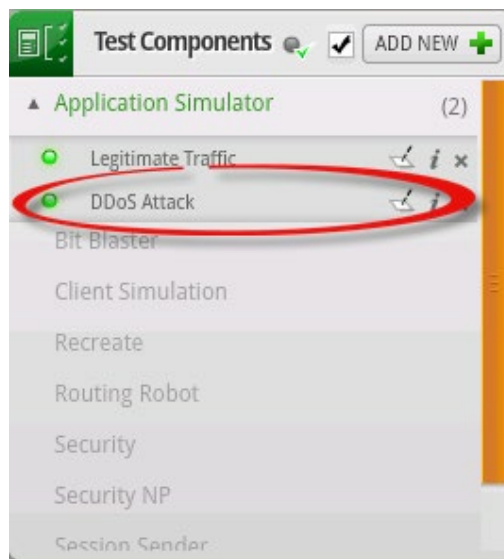
18. Click on the **ADD NEW** button from the **Test Components** section. Choose *Application Simulator* from the selection list and click on **Select** button:



19. Rename the component to something more meaningful like *DDoS Attack* and click **Create** button:



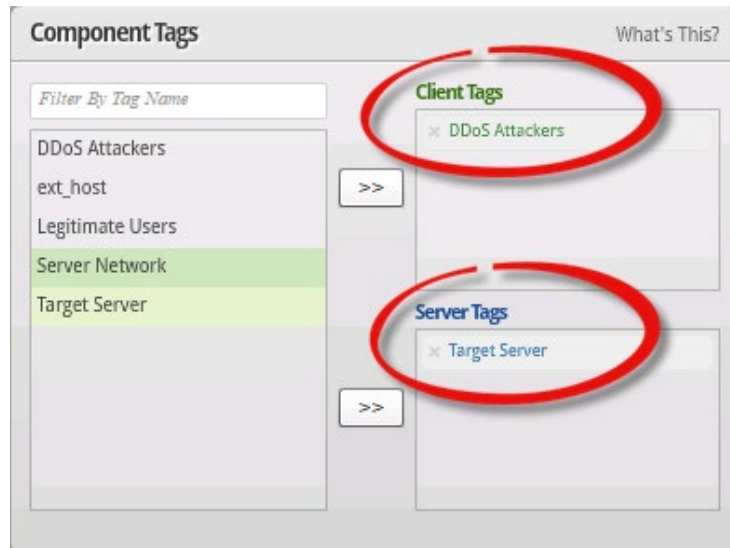
20. A new entry (i.e. *DDoS Attack*) will be created under **Application Simulator** Test Component. Click on the newly created component to edit its parameters:



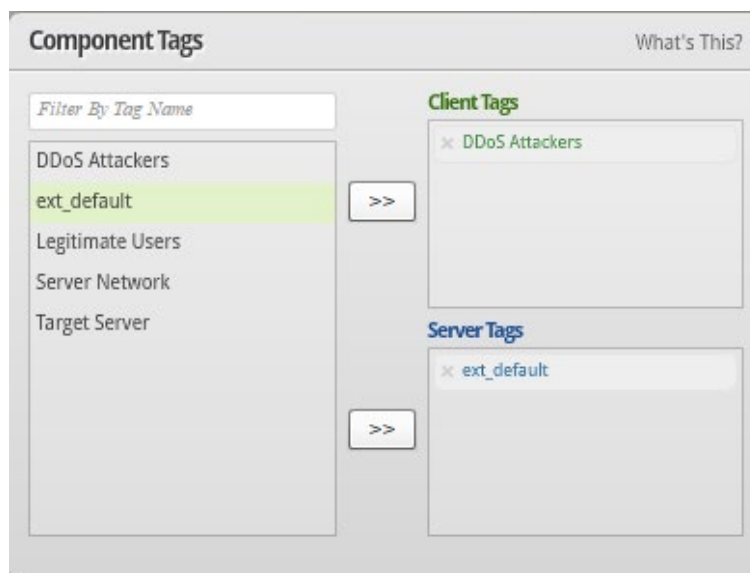
21. In the next steps, we will configure the *DDoS Attack* AppSim test component:

Test Methodologies for DoS and DDoS

- a. In the Component Tags section, make sure to assign the proper interface tags:
 - i. For the Client Tags assign the tag corresponding to the *IPv4 Static Host* Network Neighborhood element emulating the DDoS Attackers.
 - ii. For the Server Tags assign the tag corresponding to the *IPv4 Static Host* Network Neighborhood element emulating the Target Server:



Note: When emulating certain volumetric DDoS attacks (like UDP Flood, TCP SYN Flood, TCP ACK Flood, etc.) together with legitimate application traffic, in order to emulate an real environment as close as possible it is recommended to use an External Host network neighborhood element (containing one or a small subset of the same IP addresses used as the desination of the legitimate application traffic) as the destination of the DDoS attack traffic (i.e. in the Component Server tags), like in below example (will NOT be used in this test since we are emulating an application level DDoS attack):

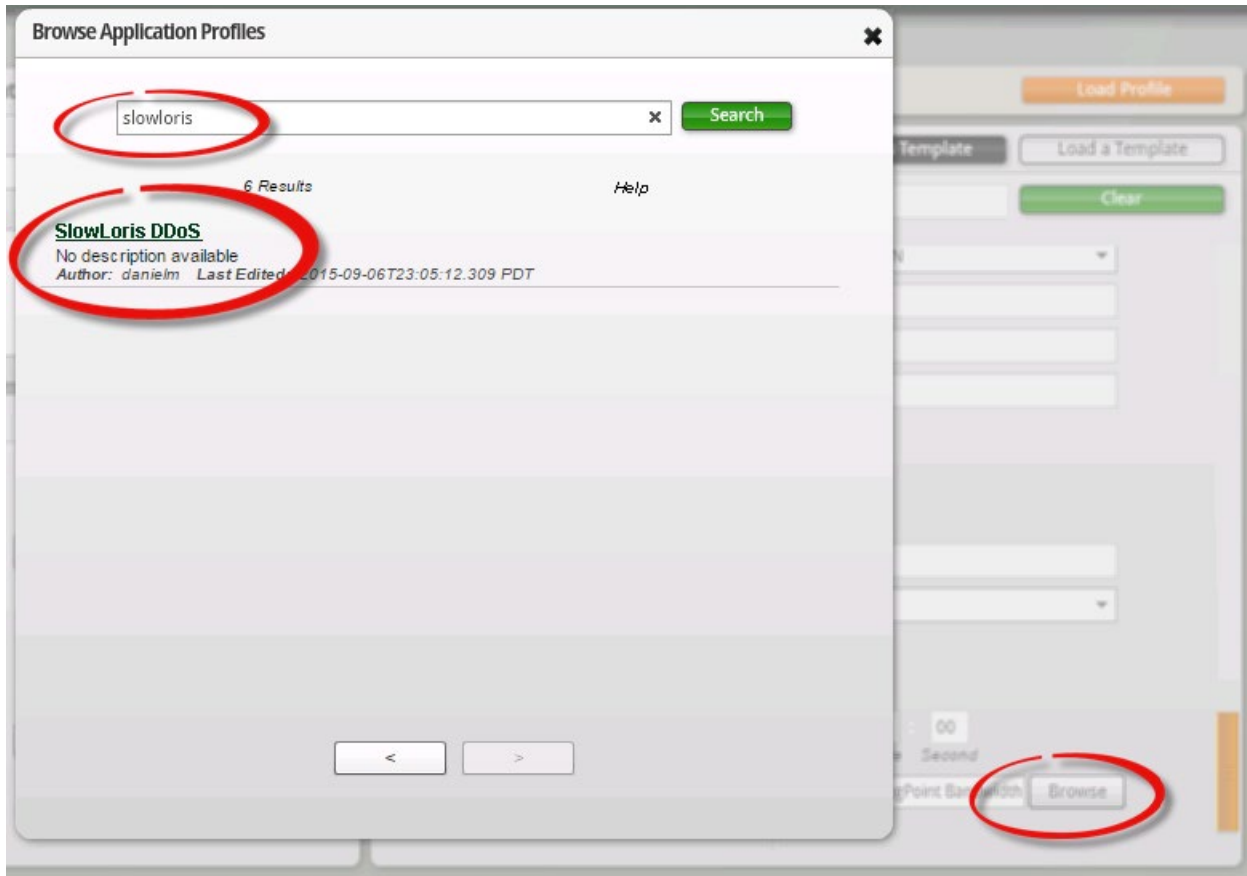


- b. The Parameters section contains multiple test component attributes that are grouped into category folders based on their functional purpose. For the scope of this test, since we have used a template, most of the parameters can be left with their existing value except the following parameters:
- i. **Data Rate:** check the **Unlimited Data Rate** option so that the test will not be limited by the amount of generated throughput. Instead the traffic load will be controlled by the number of Simultaneous Superflows and Superflows per second as configured below.
 - ii. **Maximum Simultaneous Superflows:** Configure this value to the total number of simultaneous Superflows targeted to be achieved by the DDoS Attackers. For this example, we will use 2,000,000.
 - iii. **Maximum Super Flows Per Second:** set the value to the maximum number of sessions per second that the DDoS attackers will generate (for this test we will set it to *100,000*).
 - iv. **Delay Start:** Configure it to 2 min. This will create a delayed offset until the DDoS attack will start.

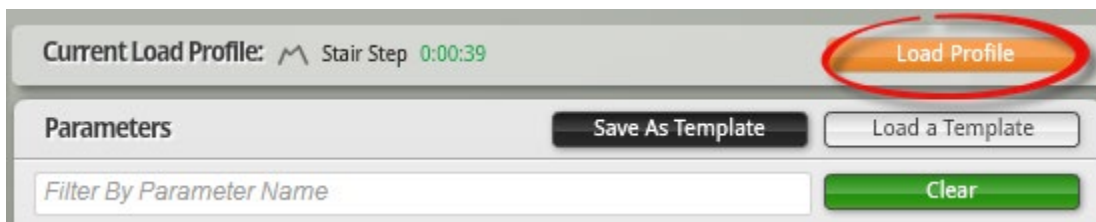
Some DDoS mitigation devices require a period of time passing legitimate traffic through them for the “learning” phase to create the baseline traffic and define/generate the thresholds before starting to perform DDoS mitigation. In such cases, the legitimate application mix traffic could be run for an even extended time period.

Application Profile: Click on Browse button from the lower right corner of the page to search and select the previously created SlowLoris DDoS application profile for the attacks:

Test Methodologies for DoS and DDoS

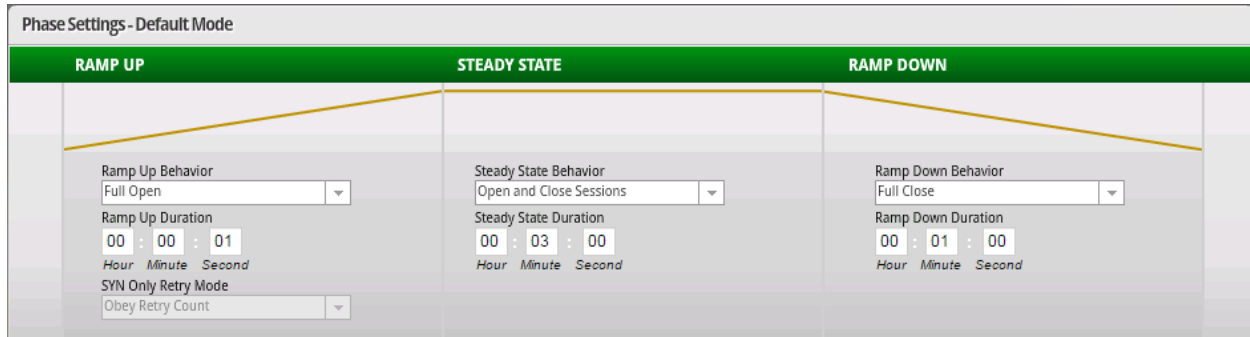


- v. Configure the legitimate traffic pattern using the Load Profile section:
 - Click on the **Load Profile** button:



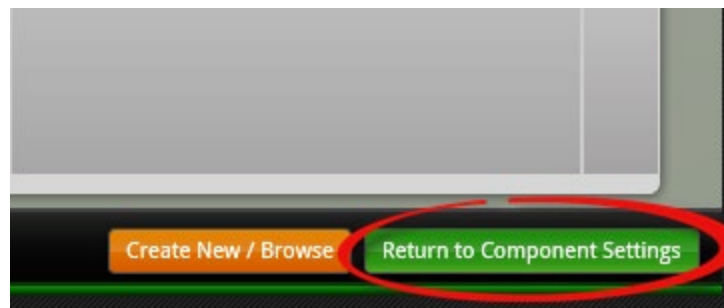
Test Methodologies for DoS and DDoS

- Change the **Ramp Up Duration** to *1 seconds*.
- Configure **Steady State Duration** to 3 minutes.
- Configure **Ramp Down Duration** to 1 minute.

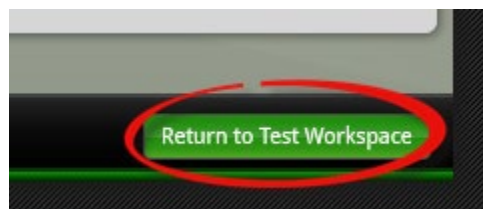


In this example, the *DDoS Attack AppSim* component is configured to generate Slowloris DDoS attacks for a duration of 4 minutes and 1 second, however it will start 2 min after the legitimate traffic begun (as configured above using the Delay Start parameter)

- Click on the **Return to Component Setting** button to go back to the Test Component configuration screen:

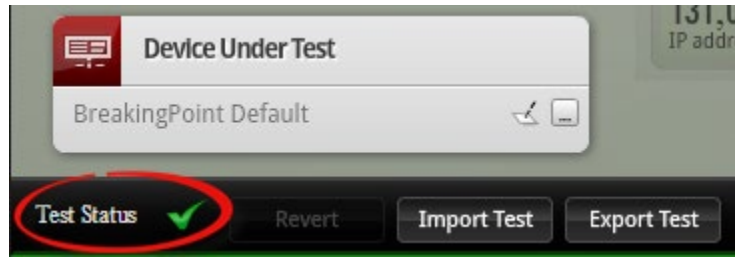


- vi. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:



Now both the legitimate application traffic as well as the DDoS Attack components have been configured and the test is ready to run.

22. Make sure the **Test Status** indicated (on the lower left corner) has a green checkmark:

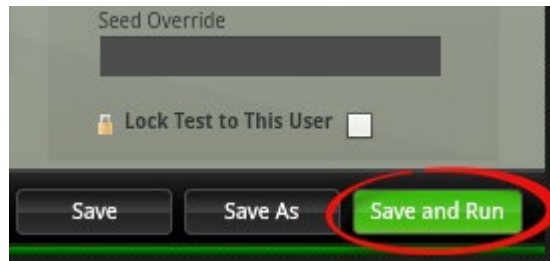


If there is not, determine what is wrong by selecting Test Status and viewing the errors.

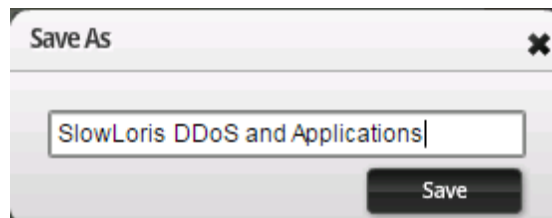
Note: Depending on the previously configured test objectives and the number of physical test interfaces selected, a warning sign (⚠) might appear next to the Test Status. This might indicate that:

- a. Not all test interfaces are selected therefore not all test blade resources will be available for the test.
- b. Total sending bandwidth capacity exceeded (and Interfaces being oversubscribed) caused by the fact that “unlimited” data rate was selected for the test component. This should not represent an issue for this type of tests.

23. Select **Save and Run** from the lower right corner:



24. If the test has not previously been saved, enter a name for the test and click **Save**:



BreakingPoint offers a broad suite of statistics to monitor various KPIs relevant for an extended set of different tests.

Result Analysis

It is always a good practice before running a layered DDoS test, to first run a baseline test with only the legitimate application traffic. This makes sure that the DUT is able to sustain the required traffic load.

The main objective of this test is to validate that the DDoS mitigation solution is able to detect and protect against various application-level DDoS attacks. It is very important to read and interpret below statistics in the context of the entire system configuration; hence in accordance to the DUT/SUT type, mitigation capabilities, configured thresholds, etc.

Real-time Statistics

After the test is started and initialized, the screen will automatically switch to Real Time Statistics window.

The **Summary** tab presents basic metrics that provide a good understanding of the overall test progress.



Since the DDoS attacks have not started yet (there is a 2-minute configured offset), the most relevant stats for the legitimate application traffic are:

TCP Connection Rate/Cumulative TCP Connections: The number of **Attempted** connections should be similar to the number of **Established** connections for the legitimate-only traffic period to certify that the DUT is able to sustain that configured load.

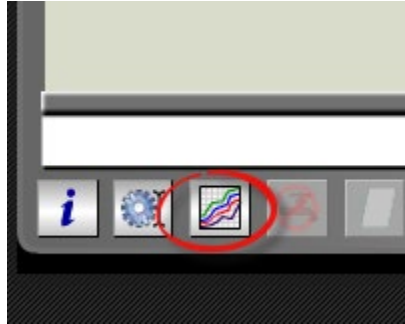
Bandwidth/Frame Rate/Cumulative Frames: As long as Rx is equal to Tx it means that there is no traffic being dropped by any intermediary device. For the first 2 minutes of the test (running only legitimate traffic), it is expected that these counters match.

Application Transactions: it is expected that no failed transactions are recorded for the legitimate traffic.

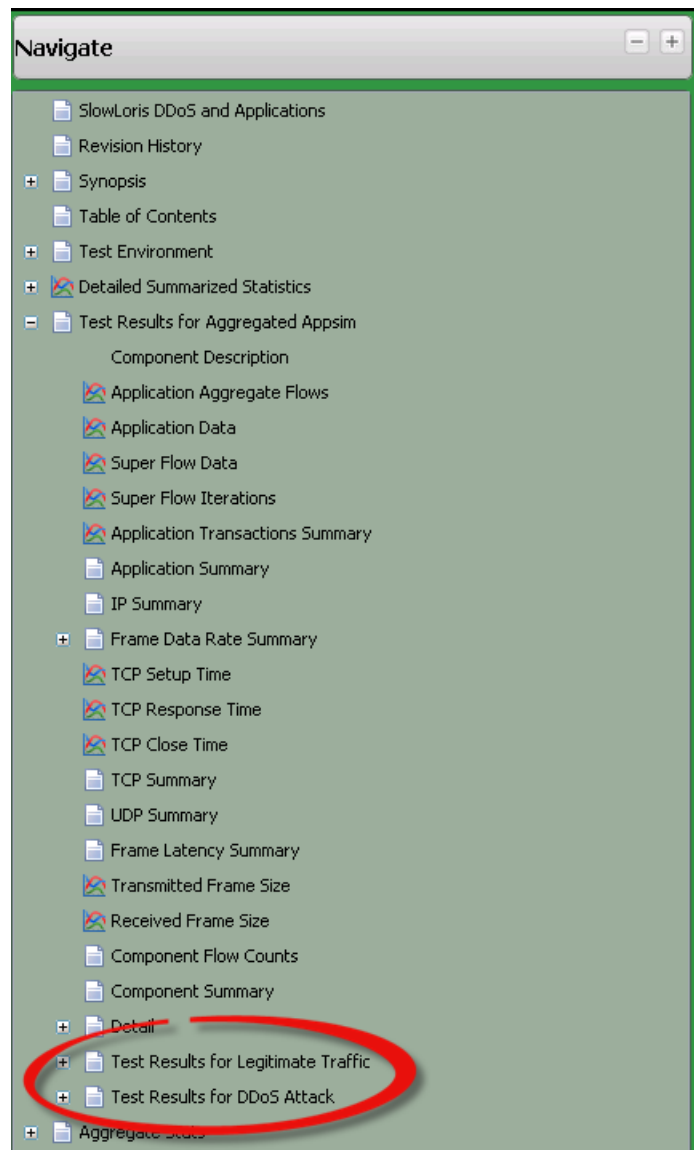
Once the DDoS traffic starts, to have a good understanding of what is happening with each of the two test components for legitimate and attack traffic, the test report can be inspected in real-time.

Test Methodologies for DoS and DDoS

In the lower left corner of the Real Time Statistics window, select the graph button to view detailed results. This will open the results in a new browser window.



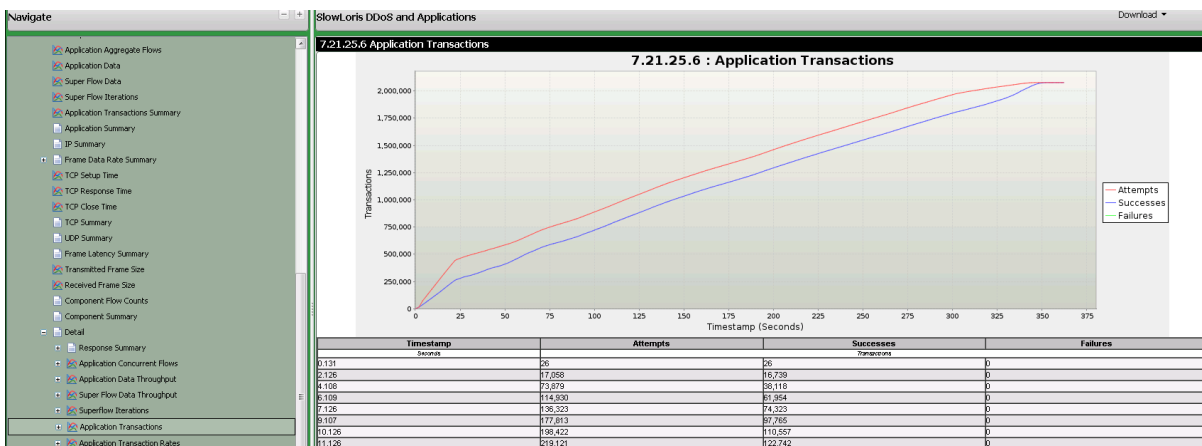
Under “Test Results for Aggregated AppSim” section, the two test components have their own set of stats:



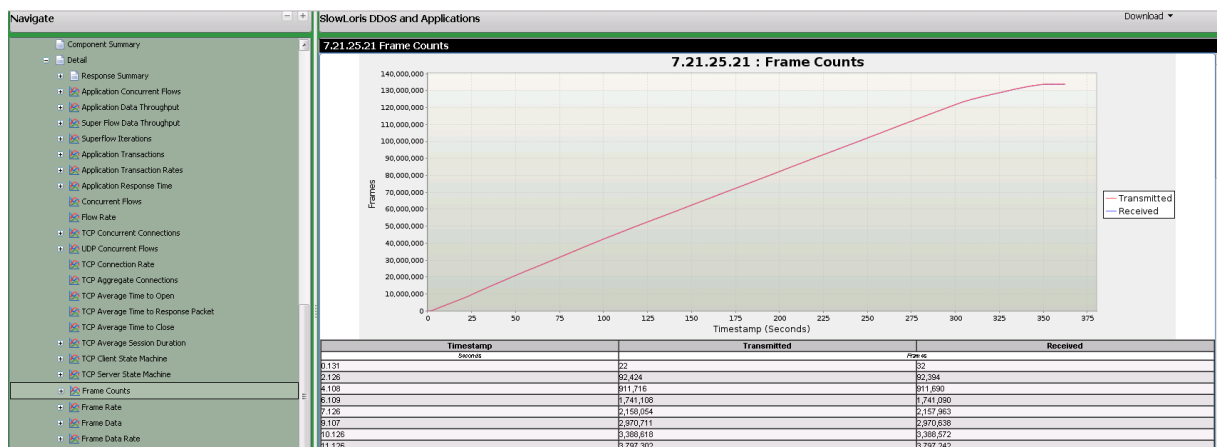
Test Methodologies for DoS and DDoS

Expanding the “Test Results for Legitimate Traffic” section provides details on what is happening with the legitimate traffic before and after the DDoS attack started.

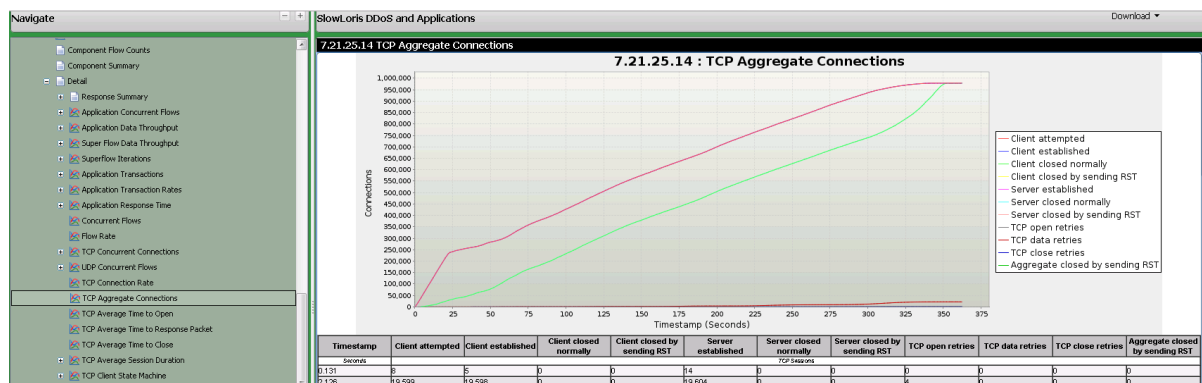
For example, under “Detail” section the “Application Transaction” time graph (as well as “Superflow Iterations,” “Application Transaction Rate,” etc.) can reveal if there are any application transaction failures beyond the point where the DDoS attack has been triggered:



Similarly, the cumulative “Frame Counters” section (and even “Frame Data” and “Frame Data Rate”) can pinpoint if there are sudden and abrupt packet losses at any point in time. These counters are very important in monitoring the DUT efficiently in mitigating the DDoS attacks.



Another important set of counters can be found under “TCP Aggregated Connection” section:



Latency-related statistics (like “Application Response Time,” “TCP Average Session Duration,” etc.) can also be leveraged to assess the impact of the DDoS traffic on the legitimate traffic.

Expanding the “Test Results for DDoS Attack” section provide details on what is happening with the DDoS traffic once it has been started.

Number of attempted vs established sessions, number of concurrent sessions throughout the test, and the amount of traffic generated vs received are important KPIs to validate if and how much of the DDoS traffic has been blocked.

All the above statistics need to be correlated with the DUT counters: the detected DDoS attack type, the amount of traffic being blocked vs the total amount of traffic, and others.

Test Variables

BreakingPoint offers the following test configuration parameters, which provide the flexibility to simulate a high number of different tests with various DDoS attack types. This assesses the DDoS mitigation solution capabilities and the impact over the legitimate traffic profile that the device would experience in a production network.

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
IP Version	IPv4	IPv6, IPsec, DSLite, 6rd, selected Mobility stacks, etc.
DDoS Attack type	Slow Loris	RUDY, Slow POST, Recursive GET, DNS Reflect, SIP Invite Flood, etc.
Benign Traffic	Pre-Canned Mix	Custom application mix that matches the traffic profile from the end customer deployment production network that needs to be validated.

Conclusions

This test methodology demonstrates how to configure BreakingPoint to determine the attack types and volume that a DDoS mitigation system such as a firewall, UTM, or dedicated anti-DDoS solution can mitigate, while the system under test is being attacked and forwarding legitimate application traffic at the same time.

Test Methodologies for IPsec VPN

IPsec Overview

The purpose of the IPsec and IKE protocols is to provide authentication, encryption and data integrity for network traffic traveling over an insecure network, such as the Internet. Two forms of IPsec usage normally apply:

Site to Site. This is shown in below figure. Two sites are connected through a pair of IPsec secure gateways. The LANs at each location are presumed to be secure and the insecure segment between the secure gateways is secured using the tunnel.

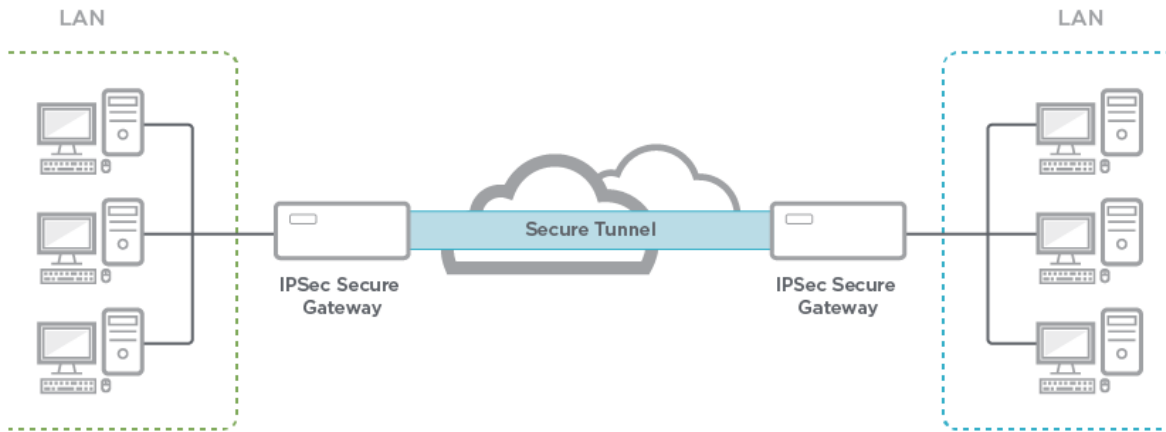


Figure 27. Site to site IPsec network

Remote Access. This is shown in below figure. In this case, the client is actually operating as its own secure gateway.

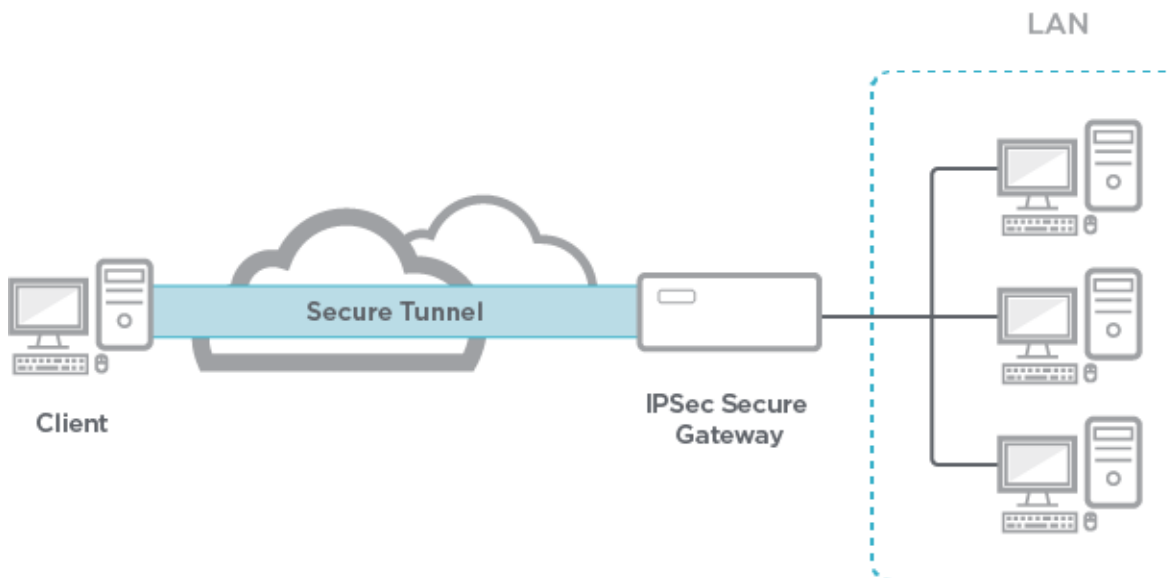


Figure 28. Remote access IPsec network

IPsec

IPsec ensures that network traffic is transmitted securely. It consists of a suite of protocols that ensure data integrity, data authenticity, data confidentiality and data non-repudiation at the IP layer. Two IPsec protocols, AH and ESP, add headers (and in the case of ESP, a trailer) to each packet:

AH: Authentication Header. The AH protocol uses a hashing algorithm over a portion of the packet to ensure that the packet has not been modified during transit. AH provides data authenticity, full data integrity and data non-repudiation, but not data confidentiality.

ESP: Encapsulated Security Payload. The ESP protocol introduces a portion of the original packet that has been encrypted, and adds a trailer to the end of the packet. ESP provides data integrity (however only for the ESP encapsulated data, not the entire packet) and data confidentiality by encrypting the upper-layer payload.

The headers can be used separately, or both can be used at the same time. The manner in which these headers are used is influenced by the two modes in which IPsec can operate:

Transport mode: In transport mode, an AH or ESP header is inserted between the IP header and the upper layer protocol header. See below figure.

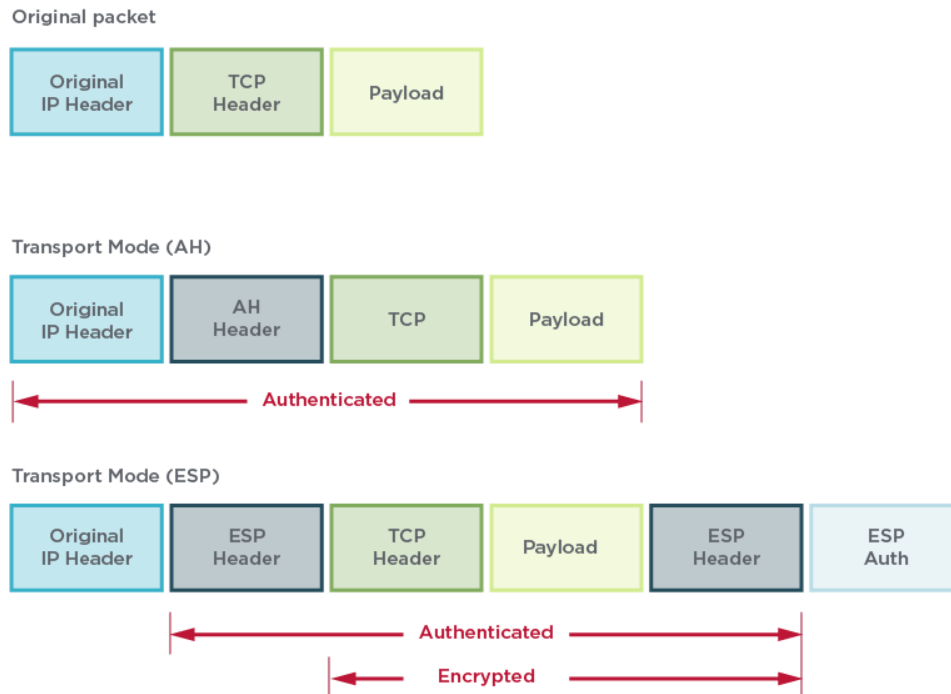


Figure 29. Transport mode packet format

The AH header includes a cryptographic checksum over the entire packet. The receiving end can verify that the entire packet was received without error or modification. The ESP header also includes a cryptographic checksum and, in addition, the packet's payload section is encrypted.

Test Methodologies for IPsec VPN

Transport mode is used only in remote access connections, where the source of the packets is also the crypto-endpoint or tunnel endpoint.

Tunnel mode: The original packet is encapsulated into a new packet that includes a new header and AH or ESP headers, as shown in below figure.

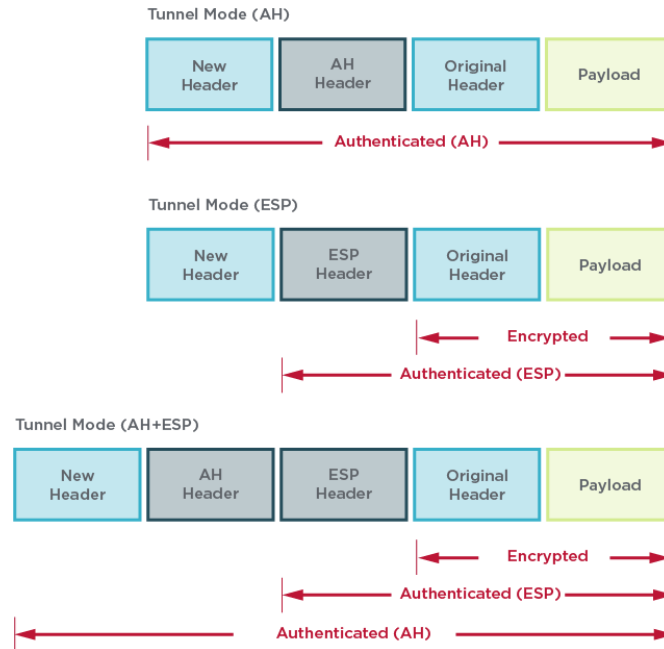


Figure 30. Tunnel mode packet format

The AH header is used to authenticate the entire packet. The ESP header is used to encrypt and authenticate the original packet. When used in combination, the original packet is encrypted and the entire packet is authenticated.

Tunnel mode is used when a security gateway is used to perform IPsec operation on behalf of a client computer, as is the case in LAN-to-LAN IPsec networks. The use of both AH and ESP headers provides maximum protection.

IKEv1

The purpose of the IKE protocol is to set up the parameters that allow two IPsec endpoints to communicate securely with each other. The set of parameters is called a security association (SA). SAs can be unidirectional or bidirectional.

The negotiation process between IPsec endpoints involves one party acting as an initiator and the other acting as a responder. Where parameters are being negotiated, the initiator offers the set of authentication, encryption, and other techniques that it is ready to use with the other endpoint. The responder tries to match this list against its own list of supported techniques. If there is any overlap, it responds with the common subset. The initiator chooses one combination of techniques from the responder and they proceed with the negotiated setting.

IKE negotiation is broken down into two phases:

Phase 1. Allows two gateways (one of which may be a client acting as its own gateway) to authenticate each other and establish communications parameters for phase 2 communications.

Phase 2. Allows two gateways to agree on IPsec communications parameters on behalf of sets of hosts on either side of the gateway.

Phase 1

During phase 1 negotiation, two endpoints authenticate each other. Based on policies enforced at each end, they decide that the other party is to be trusted. How two parties trust each other can be based on one or more data items, including the following:

- Pre-shared key
- RSA-encrypted nonces
- Digital certificates using X.509

The endpoints also agree on the particular integrity and encryption algorithms to use when exchanging their later stage phase 1 and all phase 2 messages.

Two endpoints use a single bidirectional SA at the end of their phase 1 negotiation. There are two phase 1 negotiation modes:

Aggressive Mode. Three messages are exchanged to arrive at the bidirectional SA.

Main Mode. Six messages are exchanged to negotiate the SA. Main mode differs from aggressive mode in that the transmitted identities used for authentication are encrypted as part of the protocol. This keeps the identities of the two endpoints secret.

A single phase 1 SA may be used to establish any number of phase 2 SAs. During the negotiation process, the two endpoints generate a shared secret that is used to encrypt their further communications. This shared secret is generated using public-private key cryptography in which two parties can generate a common data string without explicitly transmitting that data.

The set of parameters negotiated during phase1 are described in the following table.

Phase 1 negotiated parameters

PARAMETER	USAGE
Mode	The basic mode of phase 1 communications: Aggressive mode: Three messages exchanged without identity protection. Main mode: Six messages exchanged with identity protection.
DH Group	The public-private cryptography used to create the shared secret uses an algorithm called Diffie-Hellman. DH Groups are different bit length selections used in this calculation.

Test Methodologies for IPsec VPN

PARAMETER	USAGE
Encryption Algorithm	<p>The encryption algorithm used to protect communications during phase 1 and phase 2 message exchange. The two most common algorithms in use today are as follows:</p> <p>3DES: A 168-bit algorithm using the Digital Encryption Standard (DES) three times.</p> <p>AES: The Advanced Encryption Standard, which may be used in any bit length. Common bit lengths are 128, 192 and 256 bit.</p>
Integrity Algorithm	<p>The cryptographic checksum used over the packet to ensure data integrity. Some of the most common algorithms are as follows:</p> <p>MD5</p> <p>SHA-1</p> <p>AES-XCBS</p> <p>SHA256</p> <p>SHA384</p> <p>SHA512</p>
SA Lifetime	<p>The negotiated SA must be renegotiated after a period of time. This lifetime is itself negotiated.</p>

Phase 2

In phase 2, each of the crypto endpoints attempts to negotiate the following SAs:

An outbound IPsec SA: A unidirectional SA used to protect IPsec traffic sent to the remote tunnel endpoint.

An inbound IPsec SA: A unidirectional SA used to process IPsec traffic received from a remote crypto endpoint.

Phase 2 messages operate under the protection of a phase 1 SA by using the negotiated shared secret between the gateways. In addition to negotiating integrity and encryption parameters, they also contribute random data to be used in generating the keys for the encryption algorithms that encrypt the payload data.

The following table describes the parameters for phase 2 negotiation.

Phase 2 negotiated parameters

PARAMETER	USAGE
Mode	<p>The basic mode of phase 2 communications:</p> <p>Transport mode: The original packet header is preserved. Used only for Client to LAN IPsec networks.</p> <p>Tunnel mode: Security gateways encapsulate and encrypt the original packet.</p>
DH Group	<p>The public-private cryptography used to create the shared secret using an algorithm called Diffie-Hellman. DH Groups are different bit length selections used in this calculation.</p>
Encryption Algorithm	<p>The encryption algorithm used to encrypt the data stream between the gateways during IPsec communications. The two most common algorithms in use today are as follows:</p> <p>3DES: A 168-bit algorithm using the Digital Encryption Standard (DES) three times.</p> <p>AES: The Advanced Encryption Standard, which may be used in any bit length. Common bit lengths are 128, 192 and 256 bit. Aside from the classic AES flavor like CBC (Cypher Block Chaining), other SuiteB encryption algorithms like GMAC (Galois Message Authentication Code) or GCM (Galois Counter Mode) are gaining momentum.</p>
Integrity Algorithm	<p>The cryptographic checksum used over the packet to ensure data integrity. Some of the most common algorithms are as follows:</p> <p>MD5</p> <p>SHA-1</p> <p>SHA256</p> <p>SHA384</p> <p>SHA512</p>
SA Lifetime	<p>The negotiated SA must be renegotiated after a period of time. This lifetime is itself negotiated.</p>

Xauth and Modecfg

IKE Extended Authentication (Xauth) is an enhancement to the existing IKE protocol. Xauth is not a replacement for IKE; it is an extension of it. While IKE performs device authentication, Xauth performs user authentication. Xauth user authentication occurs after IKE authentication phase 1, but before IKE IPsec SA negotiation phase 2. With Xauth, after a device has been

authenticated during normal IKE authentication, IKE can then authenticate a user using that device.

Modecfg (mode-configuration) is an IPsec feature that functions like DHCP for IPsec clients. Modecfg enables a responder to send (push) addresses (such as a private IP address, a DNS server's IP address) to an initiator.

Modecfg can also work in the opposite direction, with the client retrieving (pull) address information from the server. Modecfg is typically used in remote-access scenarios, where addresses may be part of a pool, with different privileges given to different addresses, or groups of addresses.

Modecfg occurs right after IKE phase 1, but before IKE IPsec SA negotiation phase 2.

IPComp

IP compression (IPComp) is a protocol that improves the performance of communications between hosts by reducing the size of the IP datagrams sent between them.

IPsec peers can negotiate to use IPComp as part of the setup of a Child SA. A peer requesting a Child SA can advertise that it supports one or more IPComp compression algorithms. The other peer indicates its agreement to use IPComp by selecting one of the offered compression algorithms.

NAT-T

NAT-T (network address translation traversal) was developed to address the problem of using IPsec over NAT devices. NAT devices work by modifying the addresses in the IP header of a packet. Under IPsec, this causes the packets to fail the checksum validation provided by IPsec. To IPsec, the packets appear to have been modified in transit, something IPsec is intended to prevent.

NAT-T detects the presence of NAT devices between two hosts, switches the IPsec function to a non-IPsec port, and encapsulates the IPsec traffic within UDP packets. To preserve the original source and destination port numbers, NAT-T inserts an additional header containing the port numbers between the IP header and the ESP header.

For example, after IKE peers initiate negotiation on port 500, detect support for NAT-T, and detect a NAT device along the path, they can negotiate to switch the IKE and UDP-encapsulated traffic to another port, such as port 4500.

IKEv2

Version 2 of IKE, initially defined in RFC 4306, simplifies the IKE protocol. The main differences between IKEv1 and v2 are as follows:

- **Simplified initial exchange:** In IKEv2, the initial contact between peers is accomplished using a single exchange of four messages. IKEv1 provides a choice of eight separate exchange mechanisms.

- **Reduced setup latency:** The initial exchange of two round-trips (four messages), coupled with the ability to simultaneously set up a child Security Association (SA) on the back of that exchange, and reduces setup latency for most common setup scenarios.
- **Fewer header fields and bits:** The domain of interpretation (DOI), situation (SIT), and labeled domain Identifier fields have been removed in IKEv2, as have the commit and authentication only bits.
- **Fewer cryptographic mechanisms:** IKEv2 protects its own packets with an ESP-based mechanism very similar to the one it uses to protect IP payloads, simplifying implementation and security analysis.
- **Increased reliability:** In IKEv2, all messages must be acknowledged and sequenced (in IKEv1, message IDs are random), which reduces the number of possible error states.
- **Resistance to attacks:** To better resist attacks, an IKEv2 host does not do much processing until it has satisfied itself that a potential peer is authentic. IKEv1 is vulnerable to DoS attacks (attack by causing excessive processing) and spoofing (access using a forged address).

In addition to the original IKEv2 specification defined in RFC 4306, a subsequent set of RFCs were issued like:

- RFC 4718 IKEv2 Clarifications and Implementation Guidelines, provided further details on implementing IKEv2.
- RFC 5996 which replaces and updates RFC 4306, and includes all the clarifications from RFC 4718.
- RFC 7296 obsoletes RFC 5996, and includes all the errata for it

Initial Exchanges

Communication between IKEv2 peers begins with exchanges of IKE_SA_INIT and IKE_AUTH messages (in IKEv1, this is known as Phase 1). These initial exchanges normally consist of four messages, although there may be more for some scenarios. All IKEv2 message exchanges consist of request and response pairs.

The first pair of messages (IKE_SA_INIT) negotiate the cryptographic algorithms to be used, exchange nonces, and exchange Diffie-Hellman values.

The second pair of messages (IKE_AUTH) authenticates the previous messages, exchanges identities and certificates, and establishes the first Child SA. Parts of these messages are encrypted and have their integrity protected using keys established through the IKE_SA_INIT exchange, to hide the peers' identities from eavesdroppers. Furthermore, all fields in all messages are authenticated.

Initiator to Responder

The initial exchange begins with the initiator sending the following to the responder:

- An IKE header that contains the security parameter indexes (SPIs), version numbers, and has various flags set or unset.
- A payload listing the cryptographic algorithms that the initiator supports for the IKE SA.
- A payload containing the initiator's Diffie-Hellman value.
- A payload containing the initiator's nonce (a random or pseudo-random number that is used only once in a session).

Responder to Initiator

The responder replies to the initiator with the following:

- A payload naming the cryptographic suite selected by the responder from those offered by the initiator.
- A payload containing the responder's Diffie-Hellman value.
- A payload containing the responder's nonce value.
- Optionally, the responder may send a certificate request as well.

At this point in the negotiation, each peer uses the nonces and Diffie-Hellman values to generate the seed values to be used in turn to generate all the keys derived for the IKE SA. Keys are generated for encryption and integrity protection (authentication); separate keys are generated for each function in each direction.

An additional value is derived from the Diffie-Hellman values, to be used to generate keys for child SAs.

Beyond this point, all parts of the messages exchanged between the peers are encrypted and authenticated, except for the headers.

Initiator to Responder

In the next series of exchanges, the initiator asserts its identity, proves that it knows the secret corresponding to identity and integrity, and protects the contents of the first message using the AUTH payload. If a certificate was requested, it may return the certificate and a list of its trust anchors. If it does send a certificate, the first certificate provided contains the public key used to verify the AUTH field. At this stage, if the responder hosts multiple identities at the same IP address, the initiator can specify with which of the identities it wants to communicate. The initiator next begins negotiating a child SA.

Responder to Initiator

The responder replies by asserting its own identity, optionally sending one or more certificates (again with the certificate containing the public key used to verify AUTH listed first), authenticates its identity and protects the integrity of the second message with the AUTH payload, and completes negotiation of a Child SA.

Child SAs

What is referred to in IKEv1 as a phase 2 exchange is known in IKEv2 as a Child SA. A Child SA consists of a single request and response pair of messages, and can be initiated by either peer after the initial exchanges are completed.

All messages following the initial exchange are cryptographically protected using the cryptographic algorithms and keys negotiated in the first two messages of the IKE exchange. Because either endpoint can initiate a Child SA, the term initiator in the context of a Child SA exchange refers to the endpoint that initiates the Child SA. The first Child SA is established using messages 3 and 4 of the initial IKE SA exchange, and establishes the parameters for using ESP, AH, and IPComp. Subsequent Child SAs can be initiated to create a new IPsec SA or to perform rekeying of the IKE SA in 2 messages.

Deleting an IKE SA automatically deletes all Child SAs based on it; deleting a Child SA deletes only that Child SA. Unless the Child SA is being used for rekeying, the Child SA exchange includes a Traffic Selector payload. A traffic selector is an address or range of addresses that an IPsec gateway uses to decide what to do with an inbound packet. Traffic Selector payloads specify the selection criteria for packets to be forwarded over SAs.

If an IPsec gateway receives an IP packet that matches a 'protect' selector in its Security Policy Database (SPD), it must protect that packet with IPsec. If there is no SA established, it must create one.

The portion of the Child SA message after the header is encrypted, and the entire message (including the header) is integrity protected (authenticated) using the cryptographic algorithms negotiated for the IKE SA.

Requesting Internal Addresses on Remote Networks

IKEv2 includes a mechanism for external hosts to obtain a temporary IP address for a host on a network protected by a security gateway. This mechanism, described in section 2.19 of RFC 4306, involves adding a Configuration Payload (CP) request to Child SA request.

When a security gateway receives a CP request for an address, it can either obtain an address from an internal pool or it may query external servers (such as DHCP or BOOTP servers) to obtain the address. To return the address, the gateway returns a CP reply.

This mechanism provides IKEv2 with functionality similar to XAUTH and MODE-CFG in IKEv1.

Informational Exchanges

At various points during the life of an IKE SA, the peers may need to send messages to each other regarding control parameters, errors, or notice of certain events. To accomplish this, IKEv2 defines an Informational exchange. Informational exchanges occur only after the initial exchanges and are cryptographically protected with the IKE SA's negotiated keys.

Messages in an information exchange contain zero or more notification, delete, and configuration payloads. An Informational request may contain no payload. In this event, the exchange functions as a Keep Alive message and response.

The recipient of an Informational request always sends a response to it; otherwise, the sender would assume that the message was lost and retransmit it.

Cookies

IKEv1 supported cookies, and IKEv2 continues that support. Internet security association and key management protocol (ISAKMP) fixed message header includes two eight-octet fields titled 'cookies,' and that syntax is used by both IKEv1 and IKEv2, though in IKEv2, they are referred to as the IKE SPI and there is a new separate field in a Notify payload holding the cookie.

Rekeying

Rekeying refers to the re-establishment of SAs to replace SAs that have expired or are about to expire. If attempts to rekey an SA fail, the SA and its Child SAs are terminated. The peers can then negotiate new SAs.

To improve the performance and reduce the potential number of lost packets, most IKE v2 implementations allow SAs to be rekeyed before they expire (in-place rekeying). To rekey a Child SA within an existing SA, a new, equivalent Child SA is created and the old one is deleted. To rekey an SA, a new equivalent SA is created with the peer. The new SA inherits all the original SA's Child SAs, and the old SA is deleted by sending a message containing a 'Delete' payload over it. The Delete payload is always the last request sent over an SA that terminates normally.

In IKEv1, peers negotiated SA lifetimes with each other. In IKEv2, each peer selects its own lifetime for an SA, and is responsible for rekeying the SA, when necessary. If the two peers select different lifetimes, the peer that selects the shorter lifetime initiates rekeying.

If an SA and its child SAs have carried no traffic for a long time and if its endpoint would not have initiated the SA without any traffic for it, the endpoint may close the SA when its lifetime expires, instead of rekeying it.

Some IKE peers may impose a random delay before initiating rekeying. This is done to prevent a collision-like situation in which both peers select identical lifetimes for an SA, and then simultaneously attempt to rekey it, potentially resulting in duplicate SAs.

IKEv2 does not prohibit duplicate SAs. RFC 4306 states that endpoints can establish multiple SAs between them that have the same traffic selectors to apply different traffic quality of service (QoS) attributes to the SAs.

EAP

In addition to authentication using public key signatures and shared secrets, IKEv2 continues to support the Extensible Authentication Protocol (EAP), which is defined in RFC 3748.

EAP is typically used in scenarios requiring asymmetric authentication, such as users authenticating themselves to a server. For this reason, EAP is typically used to authenticate the initiator to the responder, and in return, the responder authenticates itself to the initiator using a public key signature.

EAP is implemented in IKEv2 as an additional series of AUTH exchanges that must be completed to initialize the IKE SA.

If an initiator wants to use EAP to authenticate itself, it indicates that by omitting the AUTH payload from message 3 of the initial IKE message exchange. Because it has sent an ID payload, but not an AUTH payload, the initiator has declared an identity, but has not proven it. If the responder is willing to allow authentication by EAP, it places an EAP payload in message 4 and defers sending further IKE messages until it has authenticated the initiator in a subsequent AUTH exchange.

Using BreakingPoint for IPsec testing

BreakingPoint tests a DUT for control plane-related metrics like scalability (the number of tunnels that it can create) and performance (the rate at which it creates them), but mainly from the data-plane perspective for performance (the real application traffic load it can sustain) and security resilience (running malicious traffic) over IPsec tunnels.

This section describes some of the common VPN scenarios and how BreakingPoint terminology compares with an actual VPN.

Similarly, a number of test cases will follow to emphasize most of the common aspects that need to be considered and tested in any IPsec-enabled environment.

BreakingPoint tests an IPsec security gateway DUT by emulating other IPsec gateways connected to one side of the DUT and the servers on the other side of the gateway, as shown in the following figure.

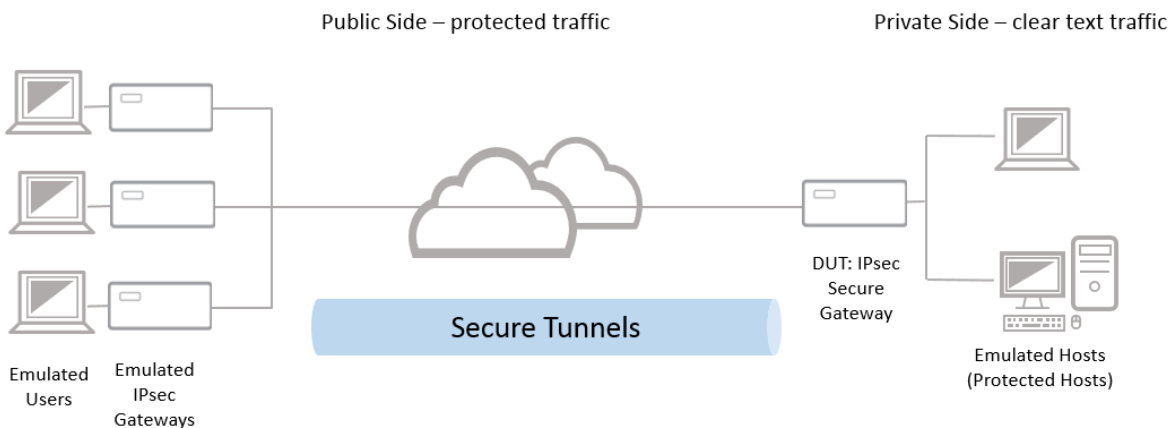


Figure 31. BreakingPoint terminology and actual VPN

Above figure shows that BreakingPoint emulates IPsec secure gateways, and the attached users or clients. IPsec tunnels are set up for the links between the emulated IPsec gateways and the DUT.

The emulated remote IPsec gateways are termed IPsec Routers (configured in the Network Neighborhood) in BreakingPoint. The emulated user behind the remote IPsec gateway is configured as a Static Host encapsulated in the IPsec Router container (all configuration done in the Network Neighborhood). The BreakingPoint test ports that emulate IPsec Routers and their attached endpoints are referred to as Public Side ports.

Test Methodologies for IPsec VPN

BreakingPoint also emulates the hosts on the private network, which are the targets of the phase 2/child SAs established by the IPsec Routers. The emulated hosts in the private network that are the data destination endpoints are configured as Static Hosts elements without any other encapsulation and directly connected to the physical interface. The BreakingPoint ports that emulate the private hosts are called Protected Hosts ports or Private Side ports.

The DUT is assumed to be an IPsec gateway. Two of the DUT's ports are used during testing:

- The Public Port is connected to the public, insecure network (emulated by the BreakingPoint IPsec Router port) and carries IKE communications and IPsec traffic. The address of the DUT interface that is used for establishing IPsec tunnels with the emulated IPsec gateways is referred to as the IKE Peer Address.
- The Private Port is connected to the private, secure network (emulated by the Protected Hosts port) and carries cleartext traffic. The address of the interface that is used to forward clear text traffic to the Protected Hosts is referred to as the Private Port IP Address.

The below figure shows how BreakingPoint compares with a real VPN.

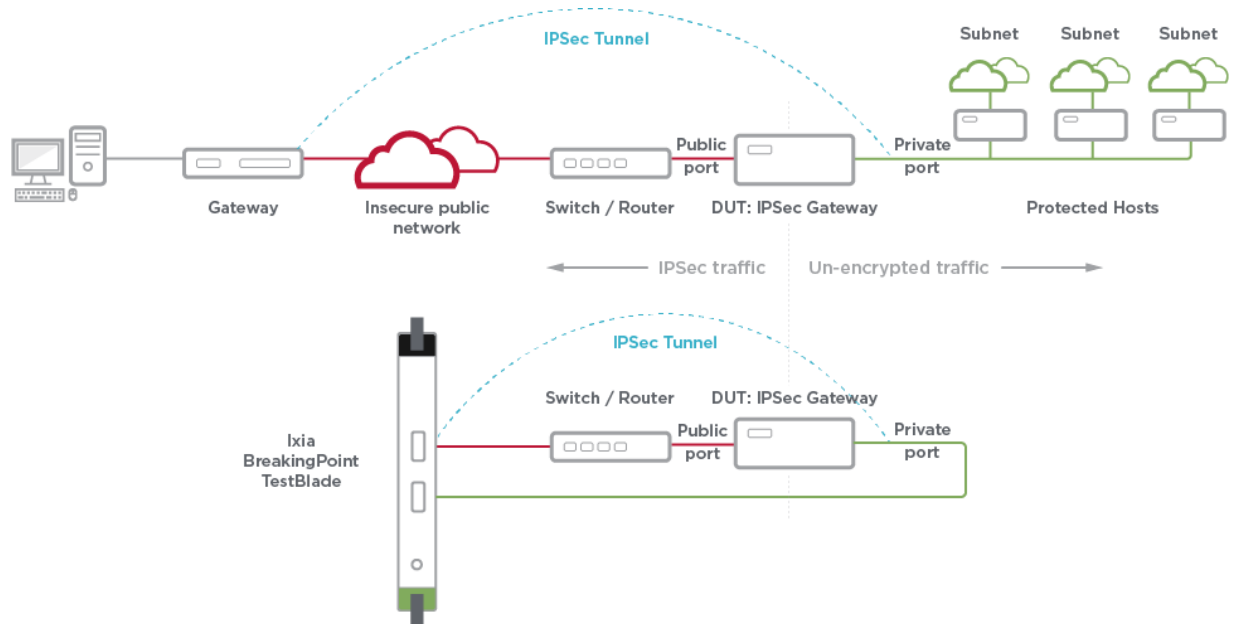


Figure 32. BreakingPoint versus actual topology

Test Case: IPsec—Real Application Mix Forwarding Performance

Overview

With the advent of converged networks, security is becoming a prime concern. Protecting data through encryption is facing an unprecedented growth in demand, but at the same time, high transfer rates must be achieved using small or large packets, or a mix of frame sizes. It is needless to mention that even if stronger Encryption or Encryption at all is going to be deployed, users won't accept service performance degradation. Therefore, proper capacity and resource planning is a must.

IPsec is one of the most widely used VPN technology. Because it provides protection at the IP level (Layer 3), it can be deployed to secure communication between a pair of gateways, a pair of endpoints, or even between a gateway and an endpoint. It offers the security features that are required in the enterprise and service provider infrastructures.

Before information can be transferred, an IPsec tunnel is established between two security gateways (SGs) using a two-phase process. Phase 1 establishes communications between the SGs, while Phase 2 establishes the communication for the network behind the SGs. Only after the completion of Phase 2, the tunnel is considered established, and the data sessions between the source and destination hosts can be validated.

Depending on the deployment model, data is securely transferred over a small number of IPsec tunnels or across a high number of concurrent tunnels.

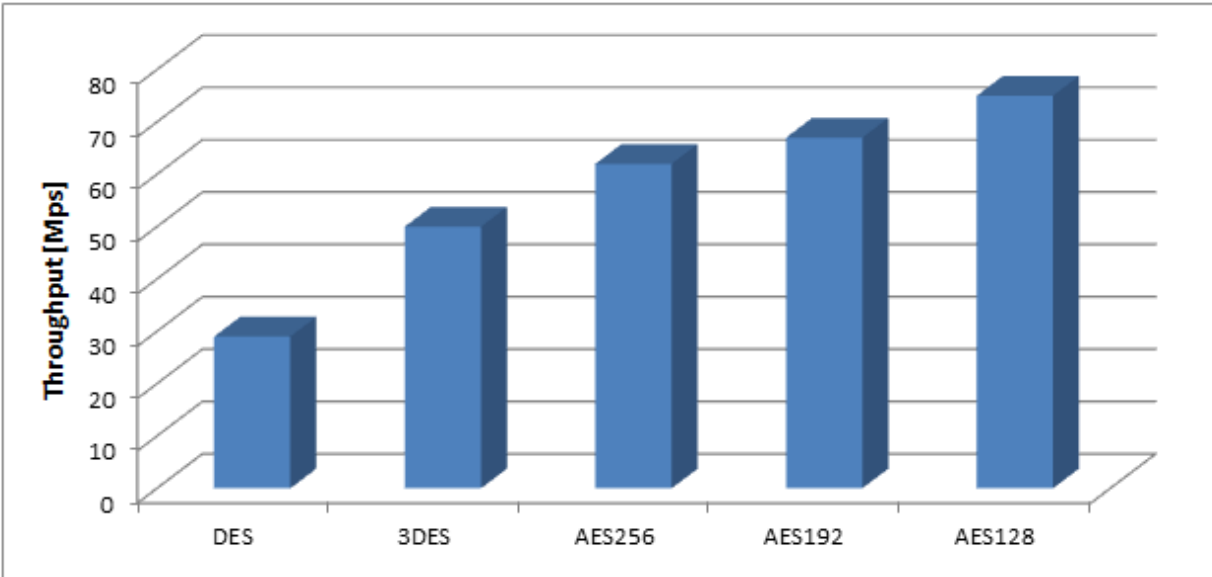
Because of the protocol complexity, IPsec performance can have degradations due to a large diversity of factors. Depending on the architecture of every DUT, its capacity to encrypt and decrypt traffic may be more or less impacted.

Data rates performance is primarily affected by the following factors:

- Encryption algorithm type (DES/3DES/AES) and its key length (128, 192, 256 bit)
 - DES (56 bit) and 3DES (168 bit)
 - AES128 (128 bit), AES192 (192 bit) and AES256 (256 bit)
 - Null

A sample result highlighting the throughput performance of a security gateway while using different encryption algorithms is shown in the following figure:

Test Methodologies for IPsec VPN



- Key size of the encryption algorithm
 - For example: 128 bit, 192 bit or 256 bit for AES encryption (see sample results in the preceding figure)
- Hash Algorithm type (HMAC-MD5 vs. HMAC-SHA1 or HMAC-SHA1)
 - HMAC-MD5 is expected to have a better performance compared with HMAC-SHA1 or HMAC-SHA2 because of the size of the secret key, which is 128 bytes compared with 160 bytes for SHA1
- The traffic type
 - small packets versus large packets versus IMIX
 - UDP versus TCP
 - stateless versus stateful
 - data traffic versus voice traffic
- Number of concurrent tunnels that are concurrently used to exchange traffic
- The overall IPsec rekey rate
 - Rekeying may degrade performance by increasing frame loss because of tunnel renegotiation

The definitive confirmation that the IPsec solution is performing as expected occurs only if it is being validated under the same conditions as in the production deployments. This implies a high-fidelity reproduction of upper layer application data type, load, and distribution.

It is not surprising nor uncommon for certain issues to occur only in very specific environments, caused by hard-to-identify triggers in the application behavior.

Objective

This test methodology provides step-by-step instructions that demonstrate how to configure BreakingPoint to determine the maximum traffic load that can be performed over 100 IPsec tunnels by using a real-life application traffic blend.

The test topology consists of a remote access deployment where one BreakingPoint test port emulates 100 IPsec clients connected to the public interface of the DUT (IPsec VPN Gateway). The second test port emulates the protected IP endpoints located behind the private interface of the DUT.

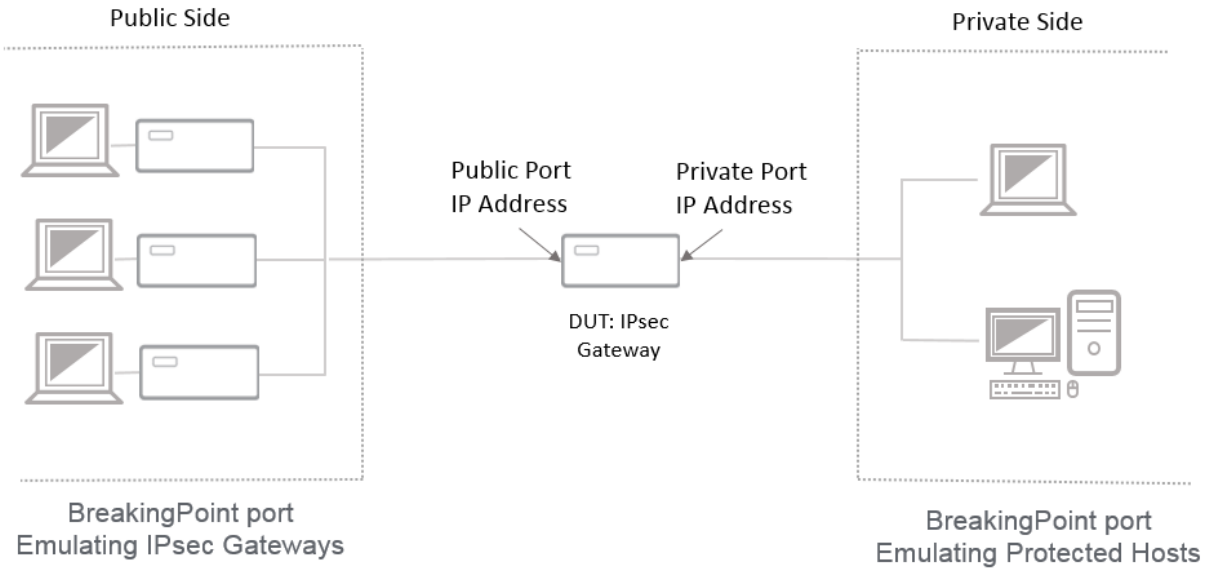
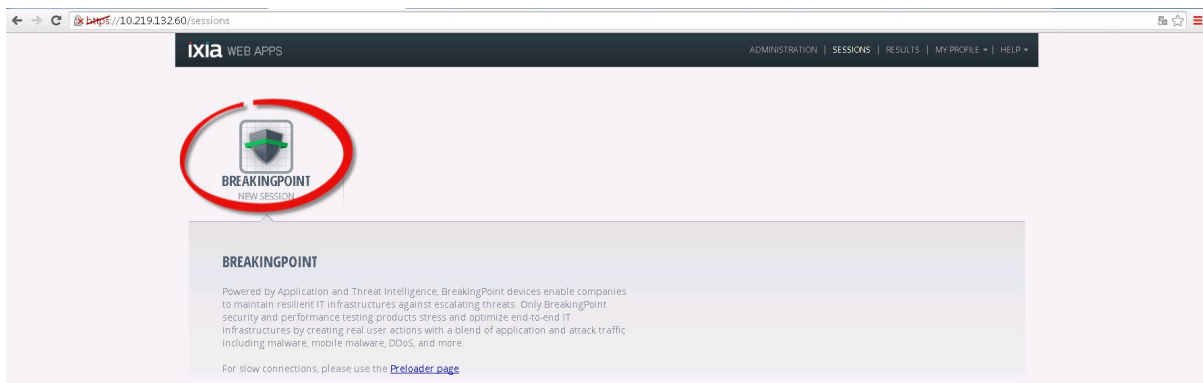


Figure 33. Test Topology

Step-by-Step Instructions

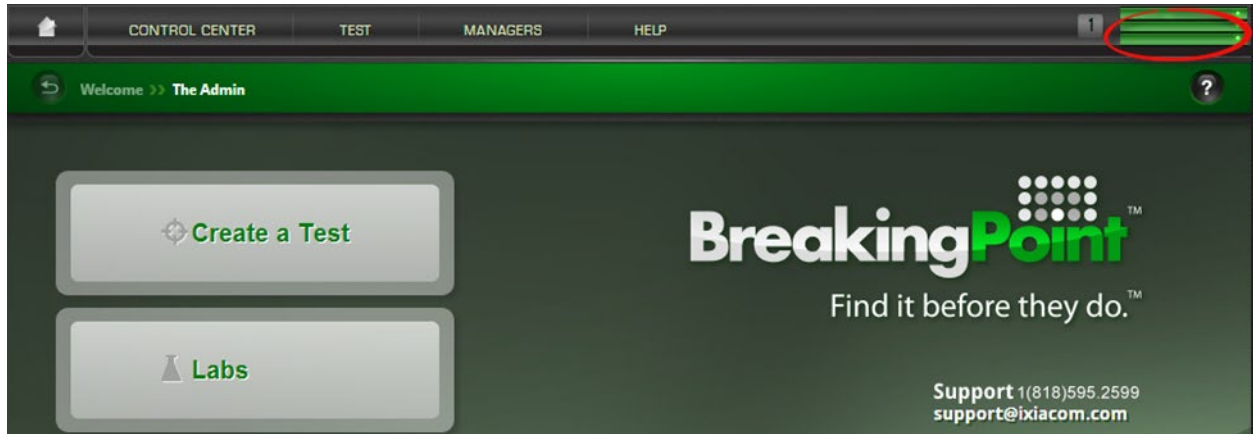
Defining the Network and Traffic Flows

1. Using a web browser, connect to the Ixia Web Apps GUI and start a new BreakingPoint Web Session:

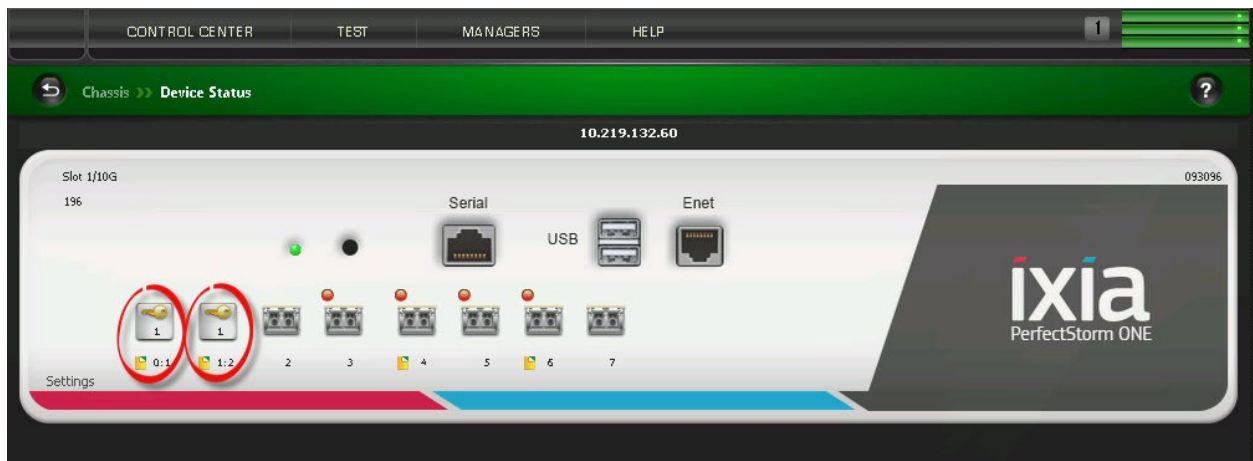


Test Methodologies for IPsec VPN

2. Once the BreakingPoint web home page loads, reserve the test ports to be used in the physical test setup to generate/receive traffic:
 - a. Click on the Device Status button located on the upper right corner:



- b. In the new screen select the physical ports that are to be used in the test:



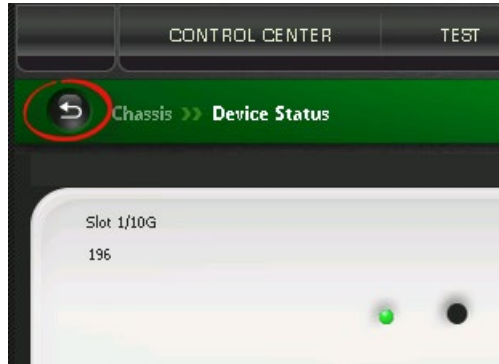
In this example, we will use physical ports 0 and 1 from the PerfectStorm One appliance.

Note: An important thing to remember when reserving your ports is the order in which you reserve them. Whenever you reserve a port, the system will automatically map that port to an interface on the chassis.

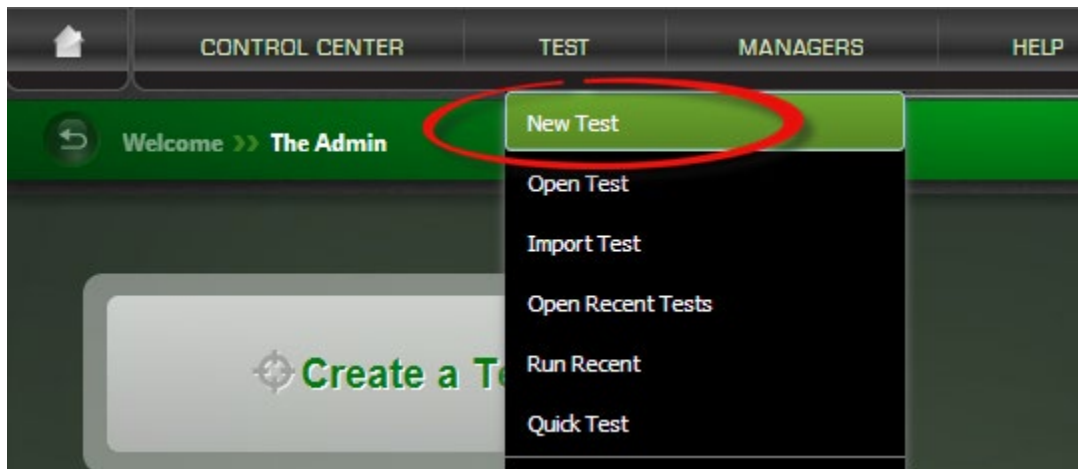
Note: The resources of each blade are allocated in proportion to the number of ports reserved on the blade. In some cases, it may be necessary to reserve additional ports to secure enough resources for the test being performed. The sessions and bandwidth available to a test may be insufficient to adequately perform the test if too few ports have been reserved.

Test Methodologies for IPsec VPN

- c. Once the proper test ports have been selected click on the back arrow to return to the initial screen:



3. Next, select **Test** -> **New Test** option from the upper menu bar to start configuring the test:



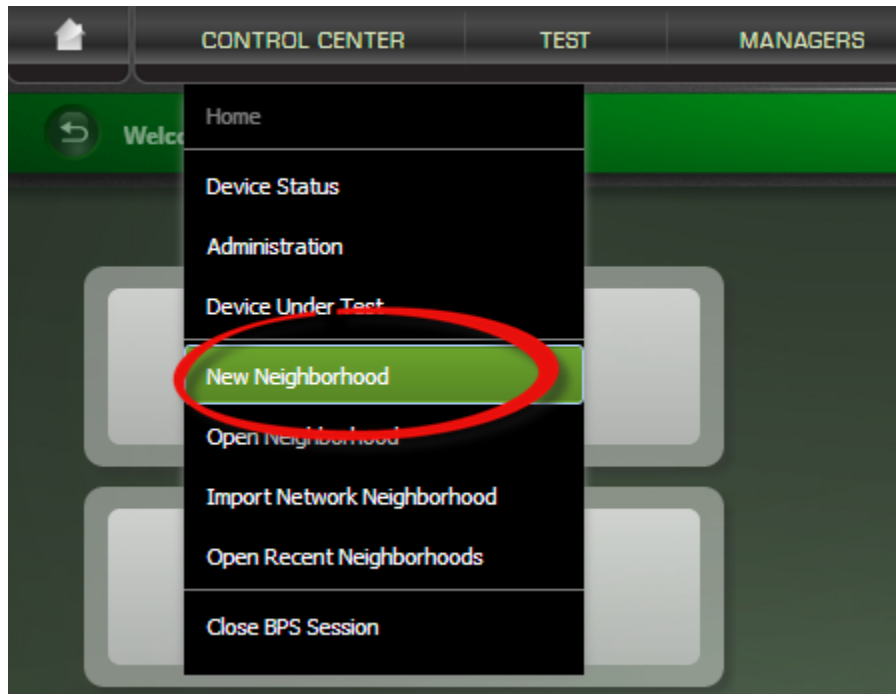
4. The new **Test Workspace** page has multiple sections that allow us to virtually control all aspects of the test. One of the most important such sections is the **Network Neighborhood** pane, which allows us to configure the MAC and IP layer stack parameters like MAC address VLANs, IP addressing scheme, MTU, etc.

Since IPsec is an OSI Layer 3 technology, it is being configured as part of the Network Neighborhood.

5. To create a new Network Neighborhood that will emulate our IPsec scenarios follow the steps below:

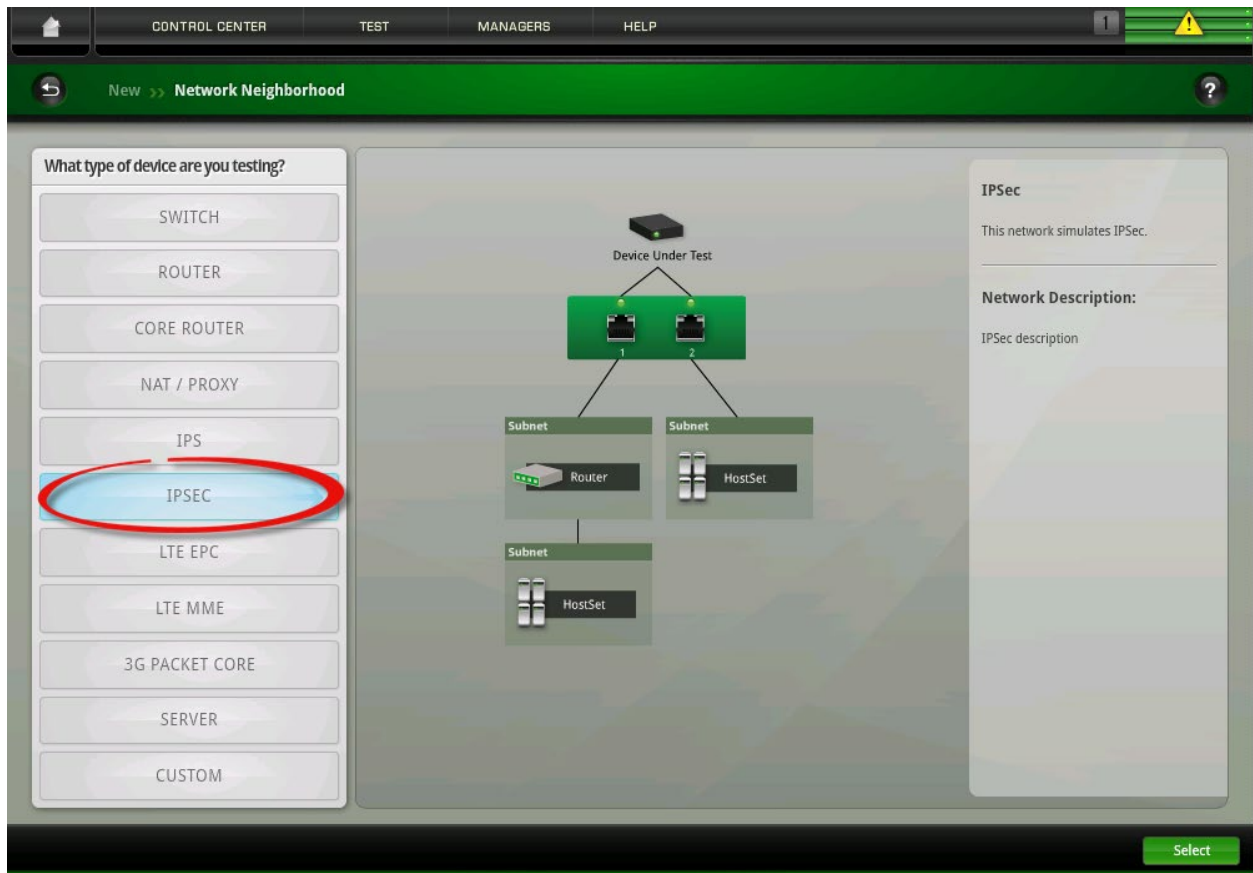
Test Methodologies for IPsec VPN

- a. From the upper menu bar select **Control Center** -> **New Neighborhood**:



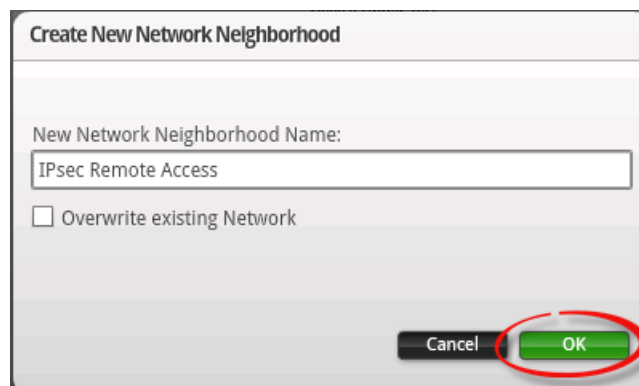
Test Methodologies for IPsec VPN

- b. Select the DUT type used for the test, and a corresponding default Network Neighborhood will be created:



Since the purpose of this test case is to emulate an IPsec environment, we will use *IPSEC* as DUT type.

- c. Enter an easy-to-recall name for the new Network Neighborhood and click OK:



Test Methodologies for IPsec VPN

- i. According to the DUT type we previously selected, a default Network Neighborhood will be created. Since we chose *IPSEC* DUT type, the following elements are automatically created:
- Two *INTERFACES*: Provides access to interface level parameters like MTU, MAC, Impairments, and Packet Filters.
 - One *IPsec IPv4 Router*: configures the external IP address of the Emulated IPsec clients

PARAMETER	CONFIGURED VALUE	COMMENTS
Container	Interface 1	The Interface number that the emulated IPsec clients resides on.
IP Address	1.0.0.2	The external IP address of the first emulated IPsec client
Gateway IP Address	1.0.0.1	The IP address of the local network gateway. Used in case the DUT (VPN Concentrator) is on a different IP subnet than the emulated IPsec clients
Netmask	8	The prefix length
IKE Peer Address	1.0.0.1	The IP address of the DUT (VPN Concentrator). This IP address will be used as the destination IP address of the initiated IPsec tunnel. In our case the DUT is on the same subnet as the emulated IPsec clients
Config	ipsec_config_1	This parameter points to the IPSEC Configuration element (which defines the IPsec local policy) used by the IPsec clients emulated within this range (see below point)

- One *IPsec Configuration*: defines the IPsec local policy (needs to match the policy defined on the DUT for the IPsec tunnels to be successful). The following

Test Methodologies for IPsec VPN

table summarizes the IPsec settings available in this element and the values configured for our test:

PARAMETER	CONFIGURED VALUE	COMMENTS
IKE Version	IKEv2	Defines the IKE version to be used
IKE Mode	none	When using IKEv1 there are two modes supported: Main mode or Aggressive mode.
1-to-1 IKE peers	Unchecked (default)	<p>This option should be used only when running back-to-back tests.</p> <p>When checked, the outer IP address of the IKE Peer Address will be incremented with 0.0.0.1 for each emulated tunnel.</p> <p>When unchecked, a single IP address will be used as a destination for the IKE messages (the IKE Peer Address).</p>
PSK	ipsectest	The value of the PreSharedKey
Left ID	none	<p>The identity value for the emulated IPsec gateways. Identity will be sent as xike-ng.</p> <p>Use prefix @ to highlight "Identity Type = FQDN"; the value following the prefix represents the FQDN (in this case xike-ng)</p> <p>To define "Identity Type = User FQDN" place the @ symbol between username and FQDN (e.g.: admin@ixiacom.com)</p> <p>To use "Identity Type = IP ADDRESS" leave the field empty. The IP address of each gateway will be used for identification.</p>
Right ID	none	The identity of the IPsec responder (StrongSwan Gateway). Use same guidelines as in Left ID to define other Identity Types
IKE DH	modp-1536 (5)	IKE Diffie - Hellman Group
IKE Encryption	aes128-cbc	Phase1 encryption algorithm
IKE Integrity	hmac-sha1	Phase1 authentication algorithm

Test Methodologies for IPsec VPN

PARAMETER	CONFIGURED VALUE	COMMENTS
IKE PRF	hmac-sha1	Hashing algorithm for the Pseudo Random Function
IKE Lifetime	86400	IKE Lifetime in seconds
ESP Encryption	aes128-cbc	Phase2 encryption algorithm
ESP Integrity	hmac-sha1	Phase2 integrity algorithm
ESP Lifetime	3600	Phase2 Lifetime
PFS	modp-1536 (5)	Perfect Forwarding Secrecy
NAT Traversal	disabled	NAT-T detects the presence of NAT devices between two hosts, switches the IPsec function to a non-IPsec port, and encapsulates the IPsec traffic within UDP packets. To preserve the original source and destination port numbers, NAT-T inserts an additional header containing the port numbers between the IP header and the ESP header. For example, after IKE peers initiate negotiation on port 500, detect support for NAT-T, and detect a NAT device along the path, they can negotiate to switch the IKE and UDP-encapsulated traffic to another port, such as port 4500 (the BreakingPoint IPsec plug-in listens on port 4500 to establish a connection for IKEv2.).
Debug Logging	disabled	Used only for debug purposes. When enabled, verbose debug info is logged which can impact the performance.
Initiation Rate	10	IPsec tunnel initiation rate
Max Outstanding	0	Upon tunnel initiation, the tunnels will move to a pending state waiting to be established. This parameter defines the maximum number of tunnels that can be in pending state (outstanding). Once this maximum threshold is hit, tunnel initiation will wait until some tunnels will connect. To configure an unlimited number of IPsec tunnels set this parameter to 0.

Test Methodologies for IPsec VPN

PARAMETER	CONFIGURED VALUE	COMMENTS
Tunnel Setup Time	600	The time in seconds an IPsec tunnels waits for responses until moving to failed state
Retransmission Interval	3	The time interval in seconds at which IKE retransmissions will happen
Rekey margin	10	The amount of time before SA expiration when rekeying should start.
Initial Contact	enabled	Enables transmission of the INITIAL CONTACT payload which notifies the peer IPsec gateway that the current IKE SA is the only one currently active
Enable Xauth	Off	XAUTH performs user authentication. XAUTH user authentication occurs after IKE authentication phase 1, but before IPsec SA negotiation phase 2. With XAUTH, once a device has been authenticated during normal IKE authentication, IKE can then also authenticate the user of that device.
UserName	None	The username in case Xauth is enabled
Password	none	The password in case Xauth is enabled
Send Wildcard TSr with ModeCFG	disabled	It enables the transmission of wildcard for the TSr (responder TrafficSelectors) when using Remote Access with ModeCFG

- **Two IPv4 STATIC HOSTS:** These will represent the actual endpoints that will be performing traffic through the IPsec tunnel. In this example, we will use:
 - a. One host simulating the endpoints that will sit on the emulated IPsec client:

PARAMETER	CONFIGURED VALUE	COMMENTS
Container	IPsec Router IF1	Traffic generated by this host will be encapsulated in the corresponding IPsec tunnel configured as the container.

Test Methodologies for IPsec VPN

PARAMETER	CONFIGURED VALUE	COMMENTS
Base IP Address	<i>0.0.0.0</i>	<p>For Remote Access scenarios, the inner IP address (protected by the IPsec tunnel) will get acquired from the DUT (VPN Concentrator) using ModeCFG.</p> <p>To run site-to-site tests, the Base IP address needs to be configured to the base address of the hosts that will be protected (hence located “behind”) by the emulated IPsec Gateways.</p>
Count	100	The number of emulated IPsec clients. Each emulated IPsec tunnel has a single endpoint “behind” it. When emulating multiple IPsec clients, the IP Address of the IPsec IPv4 Router element will increment with 0.0.0.1 for each emulated tunnel.
Gateway IP Address	<i><empty></i>	Since the traffic will be tunneled through the IPsec tunnel there is no need to configure it.
Netmask	<i><empty></i>	The prefix length. Also used as traffic selectors. When the container of this entry is set to an IPsec router, the value of the Netmask will be automatically overridden to /32.
PSN Address	2.0.0.0	The IP address of the endpoints located on the cleartext side. This parameter will be used as remote traffic selectors and usually in a two arm test, it should match the IP address configured in the IPv4 Static Hosts element on the cleartext side,
PSN Netmask	8	The prefix length of the endpoints located on the cleartext side. This parameter will be used as remote traffic selectors and usually in a two-arm test, it should match the prefix length configured in the IPv4 Static Hosts element on the cleartext side,

- b. The second host as the endpoint that will be located behind the DUT (VPN Concentrator), on the protected network hence running cleartext traffic.

Test Methodologies for IPsec VPN

PARAMETER	CONFIGURED VALUE	COMMENTS
Container	Interface 2	The hosts defined in this entry are located on the cleartext side therefore they will receive and generated unencrypted traffic. Such hosts are directly configured on the physical interface without the IPsec container.
Base IP Address	2.0.0.2	The IP address of the clertext hosts.
Count	2	The number of clertext hosts.
Gateway IP Address	2.0.0.1	The Gateway IP address used by the clertext hosts.
Netmask	8	The prefix length of the clertext hosts.
PSN Address	<empty>	Since the cleartext host have no IPsec tunneling the PSN Address and Netmask are not applicable,
PSN Netmask	<empty>	Since the cleartext host have no IPsec tunneling the PSN Address and Netmask are not applicable,

The screenshot shows the NetworkMiner software interface with the following configuration details:

- INTERFACE (2):** Untagged Virtual Interface

DEL	ID	Number	MTU	MAC Address	Duplicate MAC Address	VLAN Key	Ignore Pause Frames	Description
x	Interface 1	1	1500	02:1A:C5:01:00:00	<input type="checkbox"/>	Outer VLAN	<input type="checkbox"/>	
x	Interface 2	2	1500	02:1A:C5:02:00:00	<input type="checkbox"/>	Outer VLAN	<input type="checkbox"/>	
- IPSEC IPV4 ROUTER (1):** Simulated IPsec IPv4 router

DEL	ID	Container	IP Address	Gateway IP Address	Netmask	IKE Peer Address	Config
x	IPsec Router IF1	Interface 1	1.0.0.2	1.0.0.1	8	1.0.0.1	ipsec_config_1
- IPSEC CONFIGURATION (1):** IPsec Configuration
- IPV4 STATIC HOSTS (2):** Simulated IPv4 endpoints

DEL	ID	Container	Tags	Base IP Address	Count	Gateway IP Address	Netm
x	Static Hosts i1_default	IPsec Router IF1	Lab Client,i1_default	0.0.0.0	100		
x	Static Hosts i2_default	Interface 2	Lab Server,i2_default	2.0.0.2	2	2.0.0.1	8

To better understand how these Network Neighborhood elements are mapping to our test scenario please refer to below diagram:

Test Methodologies for IPsec VPN

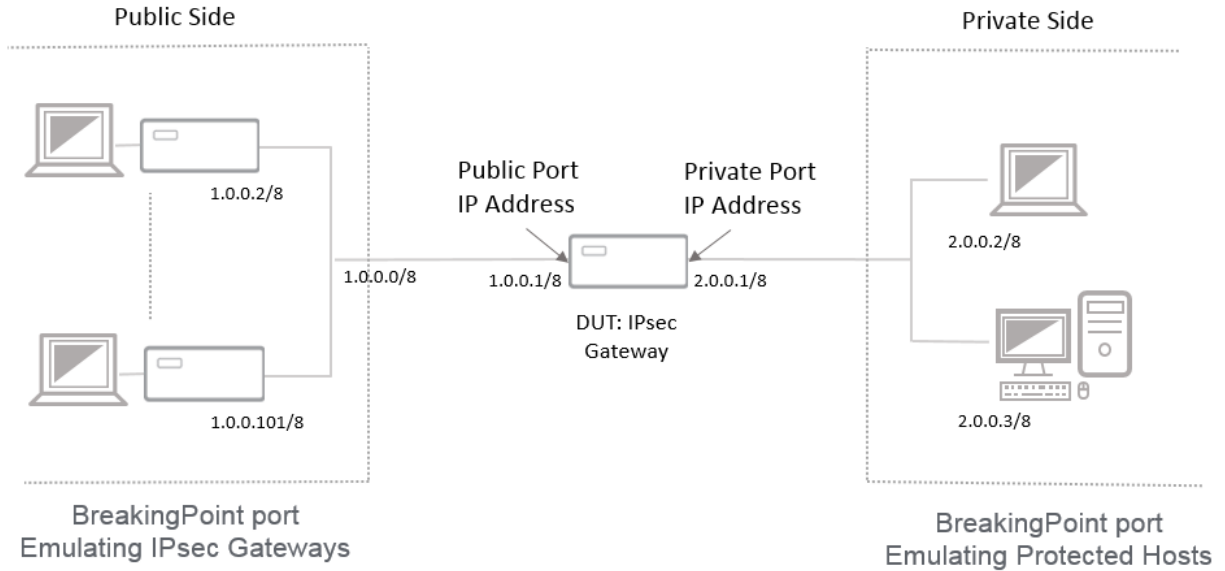
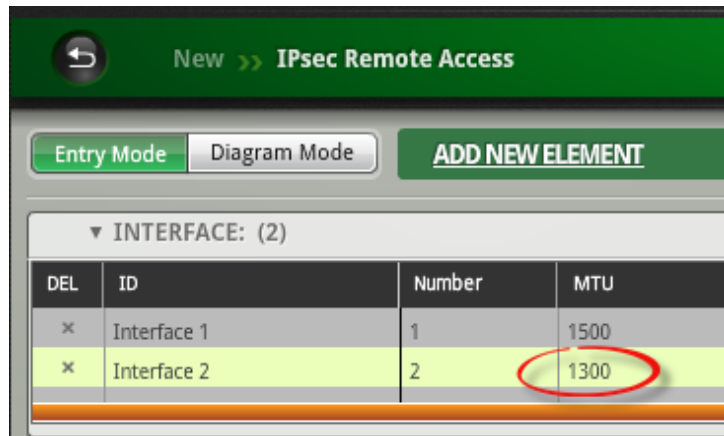


Figure 34. Test Topology

Note: Special consideration needs to be given to the effects caused by the additional IPsec overhead on the public or encrypted interface. To avoid fragmentation at the DUT level, make sure that the frames generated by the protected hosts (i.e. interface 2 in our example) plus the additional IPsec overhead to be added by the DUT is not greater than the MTU on the public or encrypted link. For that either lower the MTU on the protected or cleartext link (e.g. to an appropriate value – for example 1300 Bytes) or use a traffic profile that generate packets up to that value (e.g. 1300 Bytes).

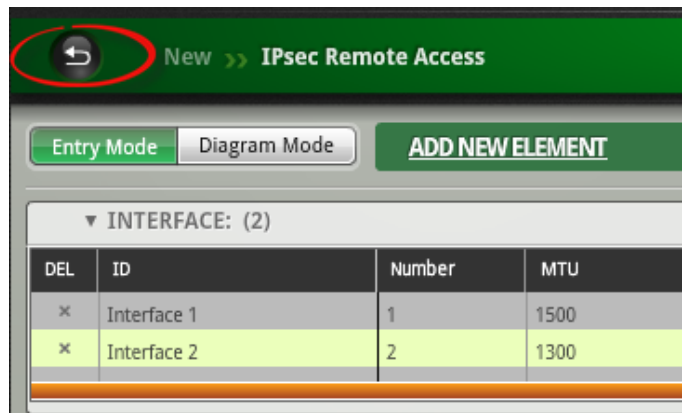


Test Methodologies for IPsec VPN

- ii. Save the new Network Neighborhood configuration by clicking the **Save** button located on the lower right corner:

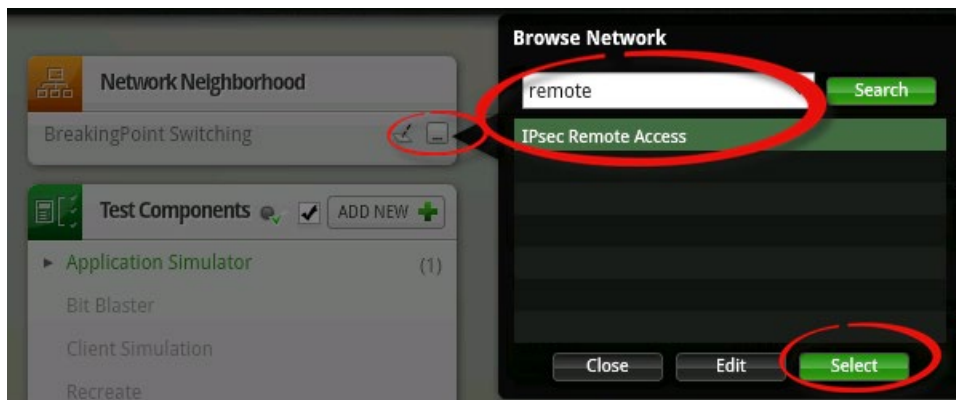


- Click on the back arrow to return to the main test screen:

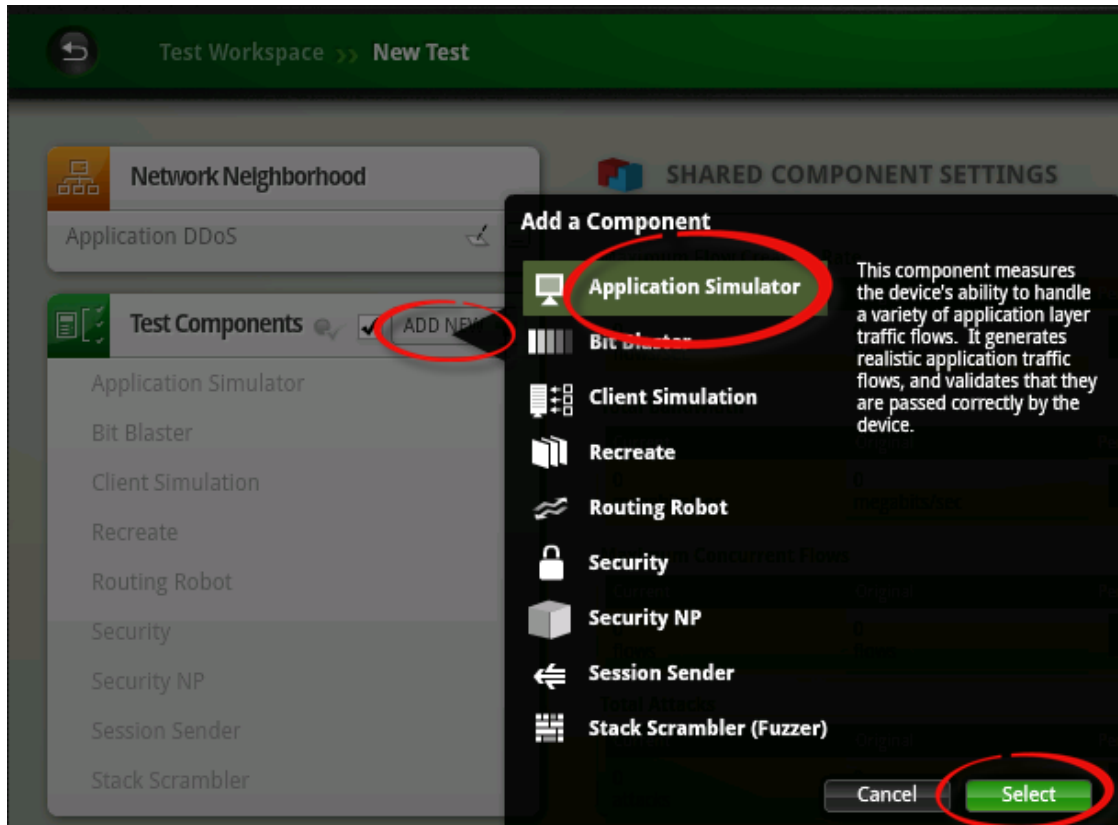


Now that the required IPsec-enabled Network Neighborhood has been configured, the rest of the test, containing the Application traffic can be configured as well

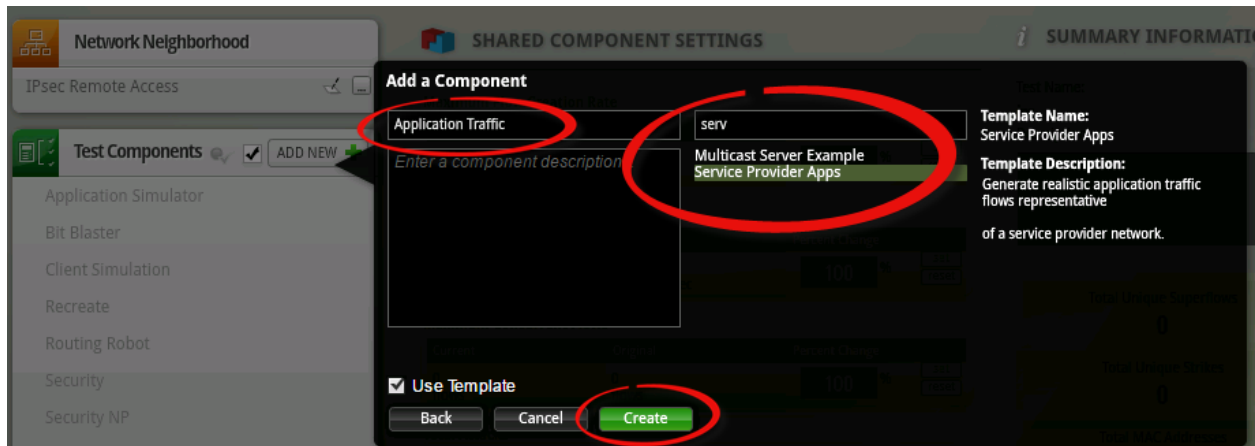
6. From the main test screen click on the browse button from the Network Neighborhood section to search and select the above created Network Neighborhood:



- Click on the **ADD NEW** button from the **Test Components** section. Choose *Application Simulator* from the selection list and click on **Select** button:



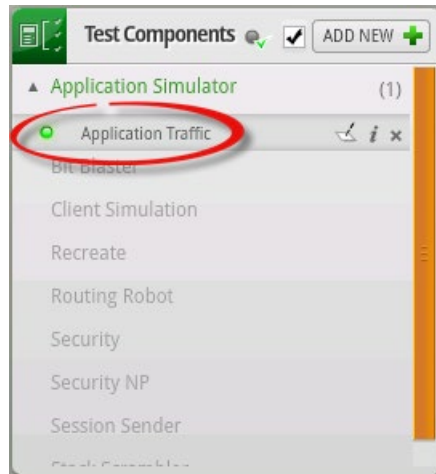
- Tick the **Use Template** checkbox then select *Service Provider Apps* as a template. Optionally, rename the component to something more meaningful like *Application Traffic* and click **Create** button:



This canned template generates realistic application traffic representative of a service provider network. However, to increase the relevance of this test case the actual application profile should be configured (using the corresponding Superflows) to match as close as possible the traffic mix and distribution seen in the particular production deployment.

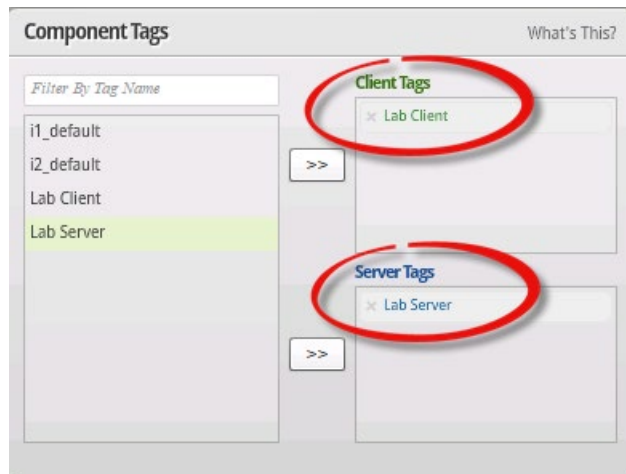
Test Methodologies for IPsec VPN

- A new entry (i.e. *Application Traffic*) will be created under **Application Simulator** Test Component. Click on the newly created component to edit its parameters:



- In the next steps, we will configure the AppSim test component:

- In the Component Tags section, make sure to assign the proper interface tags:
 - For the **Client Tags** assign the tag corresponding to the *IPv4 Static Host* Network Neighborhood element emulating the IPsec clients.
 - For the **Server Tags** assign the tag corresponding to the *IPv4 Static Host* Network Neighborhood element emulating the cleartext hosts:

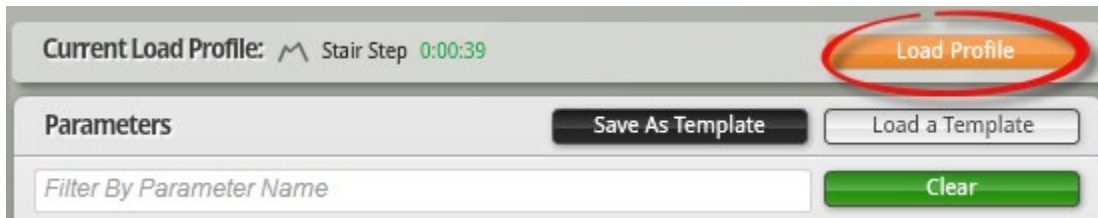


- The **Parameters** section contains multiple test component attributes that are grouped into category folders based on their functional purpose. For the scope of this test since

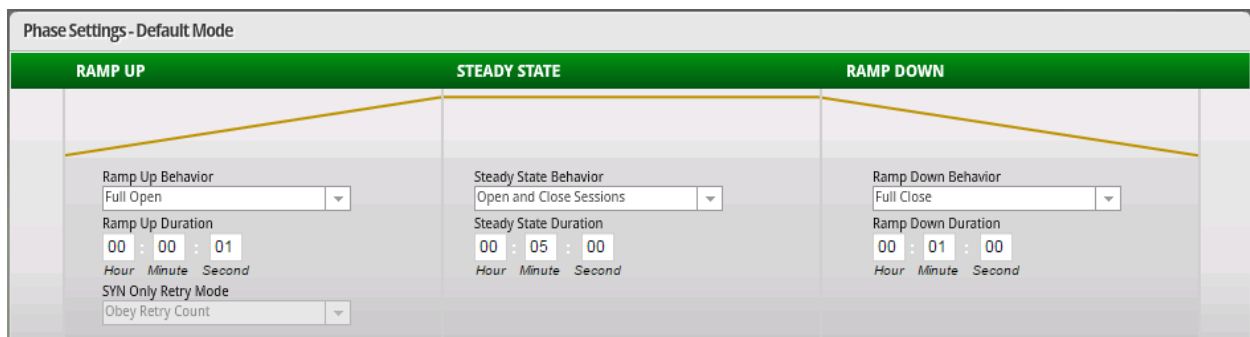
Test Methodologies for IPsec VPN

we have used a template, most of the parameters can be left with their existing value except the following parameters:

- c. **Data Rate:** check the **Unlimited Data Rate** option so that the test will not be limited by the amount of generated throughput. Instead the traffic load will be controlled by the number of Superflows per second as configured below.
- d. **Maximum Simultaneous Superflows:** Configure this value to the total number of simultaneous Superflows targeted to be achieved by the legitimate users. For this example, we will use 100,000.
- e. **Maximum Super Flows Per Second:** set the value to the maximum desired value (for this test we will set it to 1,000).
- f. Configure the legitimate traffic pattern using the Load Profile section:
 - i. Click on the **Load Profile** button:



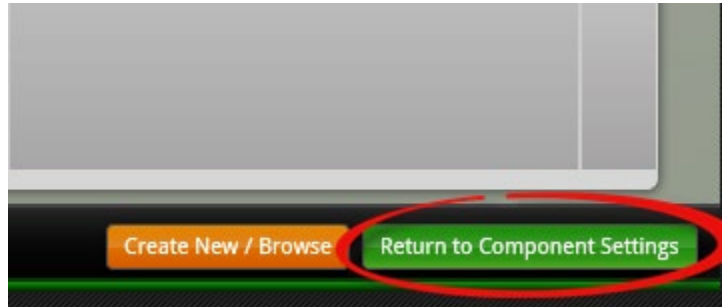
- ii. Change the Ramp Up Duration to 1 seconds.
- iii. Configure Steady State Duration to 5 minutes.
- iv. Configure Ramp Down Duration to 1 minute.



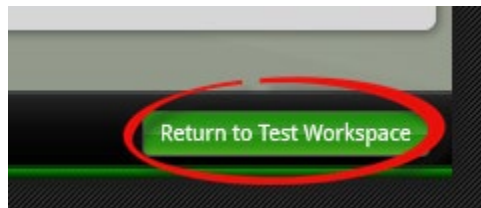
In this example, the AppSim component is configured to generate legitimate traffic for a duration of 6 minutes and 1 second.

Test Methodologies for IPsec VPN

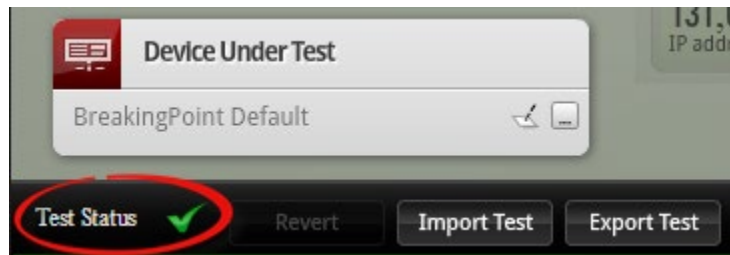
- v. Click on the **Return to Component Setting** button to go back to the Test Component configuration screen:



- vi. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:



11. Make sure the Test Status indicated (on the lower left corner) has a green checkmark:

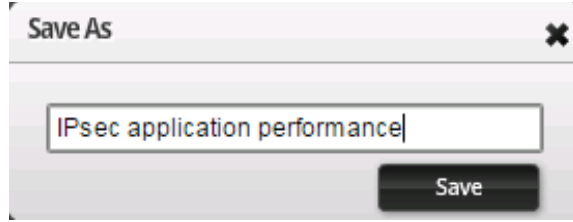


If there is not, determine what is wrong by selecting **Test Status** and viewing the errors.

12. Select **Save and Run** from the lower right corner:



13. If the test has not previously been saved, enter a name for the test and click **Save**:



Run the test, and while the test is running, continue to monitor the DUT with respect to the target rate and any failure/error counters. See the **Results Analysis** section for important statistics and diagnostics information.

In most cases, interpretation of the statistics is non-trivial, including what they mean under different circumstances. The **Results Analysis** section that follows provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

In case the DUT is able to successfully sustain the traffic load up to the maximum configured value without any failures, the test target load needs to be further increased and the test re-run again. To determine when the DUT has reached its MAX_Capability, see the Results Analysis section on interpreting results before making a decision.

Results Analysis

The maximum load performance test requires an iterative method in which the test is gradually increasing the traffic rate, allowing enough time to correctly evaluate in a relevant manner the device behavior for any discreet value.

Test Variables

The main test variables impacting the throughput are as follows:

PARAMETER NAME	CURRENT VALUE	COMMENTS
MSS Value	1460 bytes	<p>Available options</p> <p>MSS value is user configurable (see AppSim component setting).</p> <p>Data Rate Performance</p> <p>Higher degradation for smaller MSS values.</p> <p>Recommended Trials</p> <p>Repeat the test for the following MSS values: 64, 128, 256, 512, and 1024 bytes.</p>

Test Methodologies for IPsec VPN

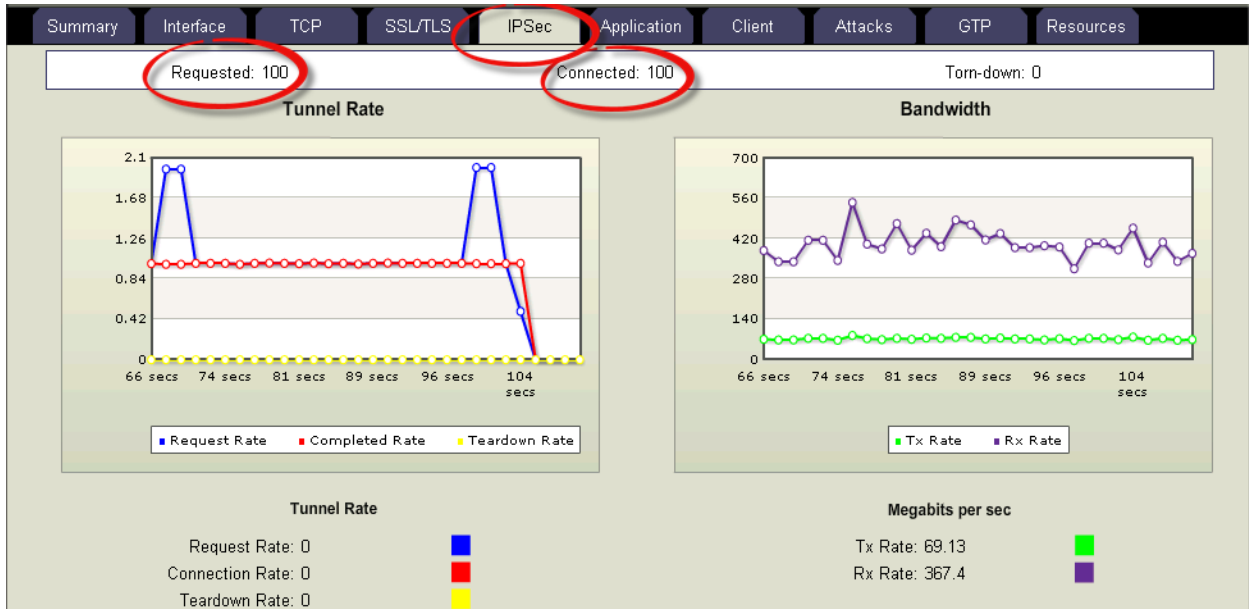
PARAMETER NAME	CURRENT VALUE	COMMENTS
Phase 1 & 2 Encryption Algorithm	AES128	<p>Available options: Null, DES, 3DES, AES 128/192/256</p> <p>Data Rate Performance Repeat the test for DES, 3DES, and AES256.</p> <p>Recommended Trials Repeat the test for Null, DES, 3DES, and AES256,</p>
Phase 1 & Phase 2 Hash Algorithm	HMAC-SHA1	<p>Available options: MD5, HMAC-SHA1, HMAC-SHA256, HMAC-SHA384, HMAC-SHA512</p> <p>Data Rate Performance HMAC-MD5 is expected to have a better performance than HMAC-SHA1 because of the size of the secret key. Similarly, HMAC-SHA256 and HMAC-SHA512 will exhibit a certain degradation.</p> <p>Recommended Trials Repeat the test for HMAC-MD5, HMAC-SHA256 and HMAC-SHA512.</p>
Traffic Type	Pre-canned Service Provider Mix	<p>Available options: UDP, TCP Stateful application traffic Security Strikes</p> <p>Data Rate Performance Higher degradation when small frames and stateful traffic is used.</p> <p>Recommended Trials Trials using custom application mix that matches the traffic profile from the end customer deployment production networks</p>

The following key performance statistics (specified in below section) must be monitored. The importance of these statistics is that it helps identify if the device has reached its saturation point, and identify issues. Also, interpreting the results in the correct manner will ensure that transient network, device or test tool behavior do not create a false negative condition.

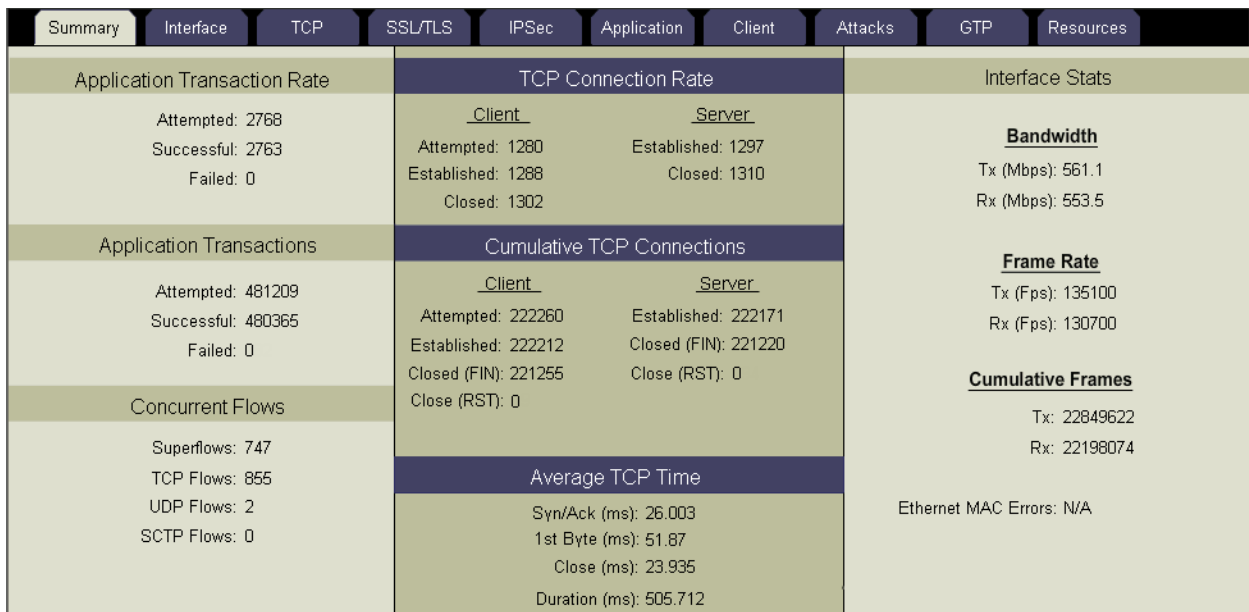
Real-time Statistics

After the test is started and initialized, the screen will automatically switch to the Real Time Statistics window.

First, the **IPsec** tab should be verified to ensure that all the initiated IPsec tunnels have been established (in the same page the Tunnel Setup rate is displayed as well):



The **Summary** tab presents basic metrics that provide a good understanding of the overall test progress, the most relevant for our test case being TCP Connection Rate, Cumulative Connections, as well as Interface Bandwidth and Frame Rate.



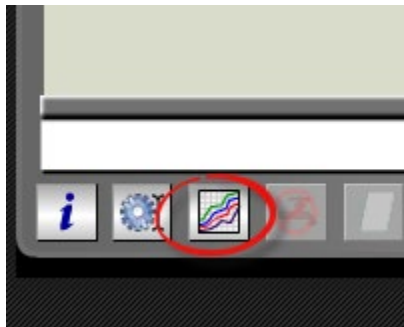
The **Interface** tab provides instant access to key statistics that enable visibility into the encryption and decryption forwarding performance of the DUT:

Test Methodologies for IPsec VPN

		Frame Rate (Frames/Sec)		Data Rate (MBit/Sec)	
		Rx	Tx	Rx	Tx
<input checked="" type="checkbox"/>	Interface 1	72980	73170	552	118.3
<i>Ethernet MAC errors: 0</i>					
<input checked="" type="checkbox"/>	Interface 2	69190	72900	76.2	514.5
<i>Ethernet MAC errors: 0</i>					

Special consideration should be given for these counters since the IPsec side interface includes the IPsec overhead as well.

Beside the real-time statistics, detailed post-test analysis can be performed. In the lower left corner of the Real Time Statistics window, select the graph button to view detailed results. This will open the results in a new browser window.



On the left side of the detailed report window is the navigation panel, where you can navigate and browse the results. The results and test information will be displayed on the right side of the browser:

Navigate

- IPsec application performance
- Revision History
- Synopsis
- Table of Contents
- Test Environment
- Detailed Summarized Statistics
- Test Results for Application Traffic
 - Component Description
 - Test Component Criteria
 - Settings
 - App Profile Summary

IPsec application performance

5 Test Environment

5.1. Settings

Seed override
Data rate reporting type

5.2. Interfaces

Number	Auto
1	false
2	false

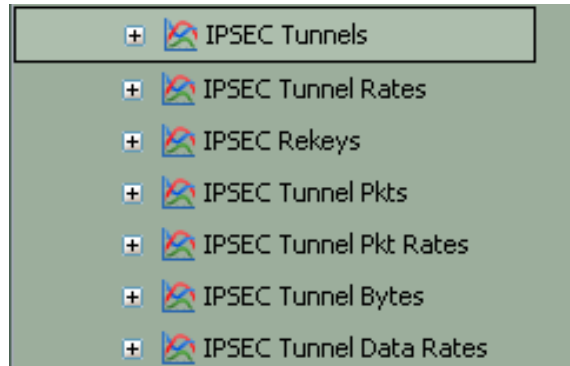
5.3. Impairments

Interface	Rate
-----------	------

5.4. Packet Filtering

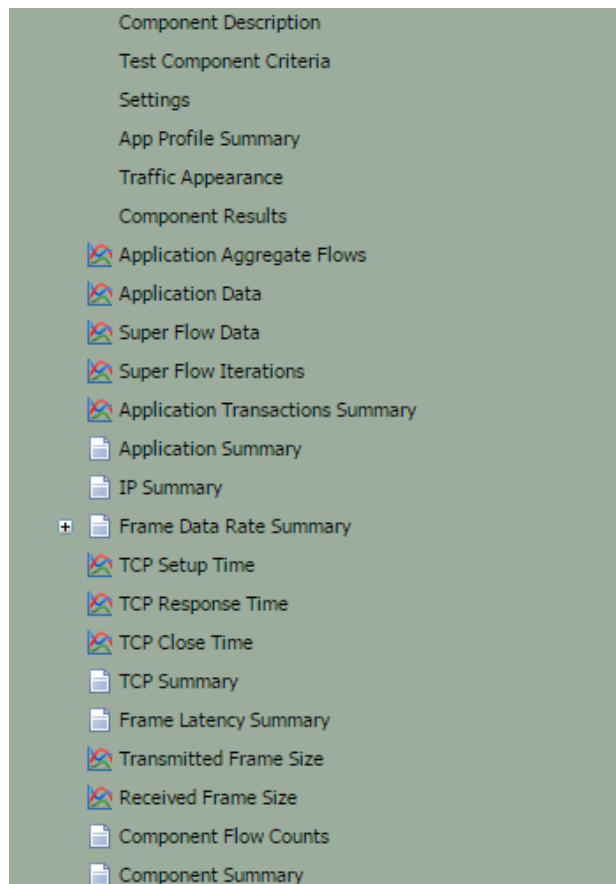
Test Methodologies for IPsec VPN

Further IPsec related metrics are located under **Aggregated Stats -> Detail**:



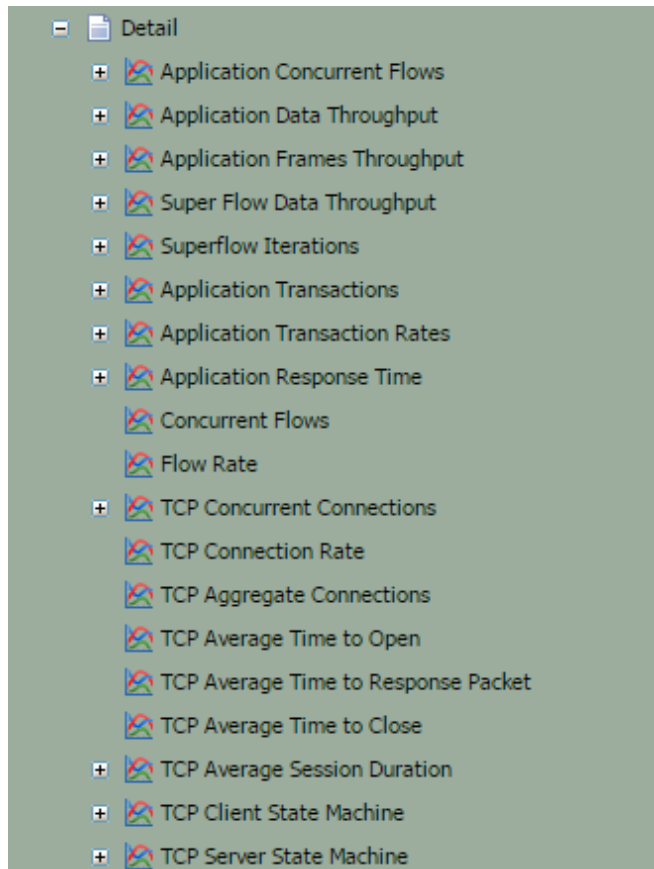
Aside from the IPsec related statistics various upper layer protocol and application metrics needs to be investigated. In the test report, under **Test Results for Application Traffic** various such related KPIs can be located such as:

- **Application Transaction Summary** for aggregated application transaction successes of failures
- **Application Summary** for overall potential application failures
- **TCP Summary** to inspect various retries or resets counters
- TCP latency metrics like **TCP Setup Time, TCP Response Time, TCP Close Time**



Test Methodologies for IPsec VPN

Additionally, for deeper investigation and traffic analysis the **Detail** section offers even more granular data points for every individual protocol and Superflow, Application Response Time and many others:



Conclusions

This test methodology demonstrates how to configure BreakingPoint to determine the maximum application data rates that can be securely transmitted over 100 IPsec tunnels and reviews some of the main parameters that affect the data rate performance.

Test Methodologies for Traffic Fuzzing and Negative Testing

Test Case: Fuzzing

Overview

Fuzzing is an effective way to measure the DUTs' reliability when handling invalid input. This type of testing is a cost-efficient method of negative testing to find issues that are difficult and time-consuming to find manually.

In the Stack Scrambler component, BreakingPoint provides capabilities to generate fuzzed network traffic.

Objective

The objective of this test case is to validate the DUT's capabilities to properly handle various levels of invalid traffic. In this test BreakingPoint will generate TCP traffic, 10% of the packets having invalid TCP flags. The expectation is that the DUT will properly handle the valid traffic and do not crash or degrade the performance due the invalid traffic.

Setup

The setup requires at least one server and one client port. The TCP client traffic will pass through the DUT to reach the TCP server. The TCP client and server are connected to the DUT as per below diagram:

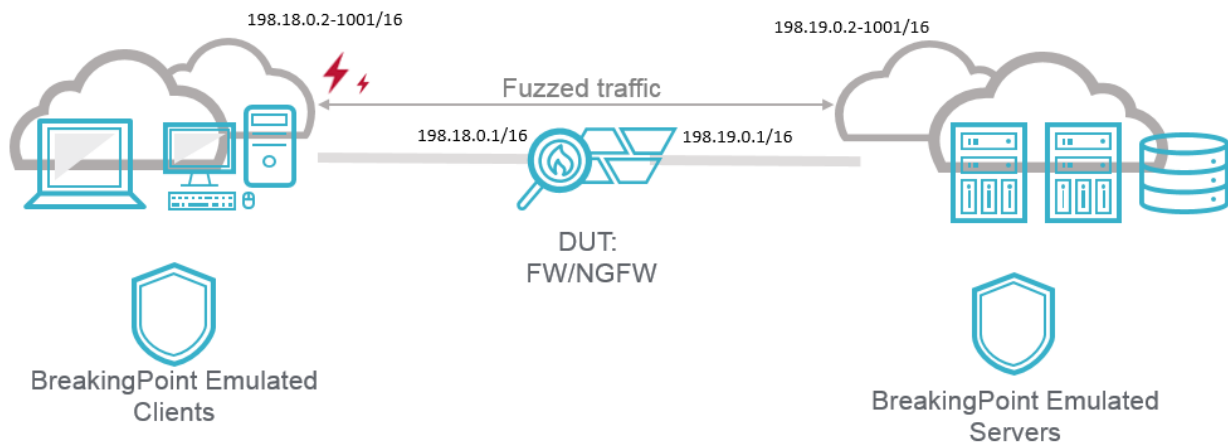


Figure 35. Test setup

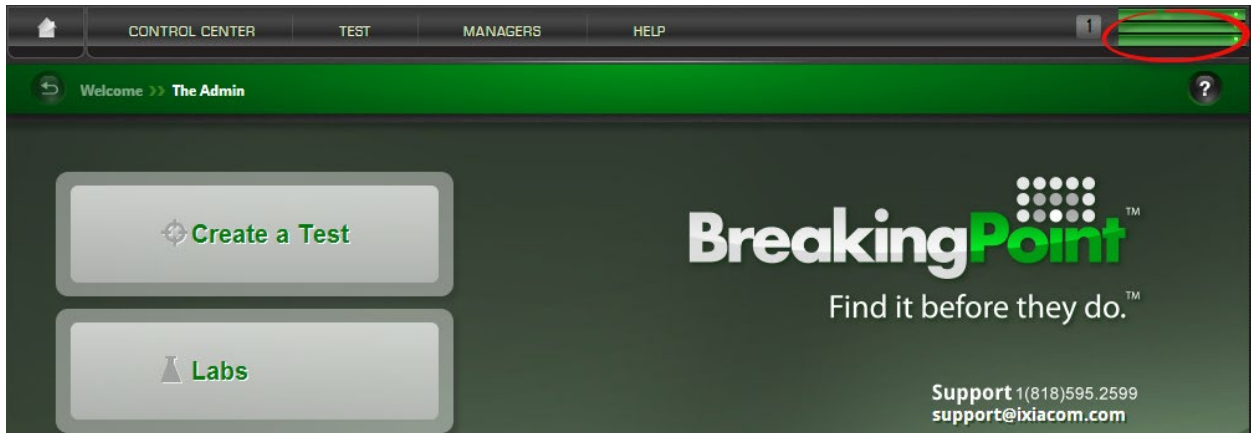
Step-by-Step Instructions

General recommendation: configure the test tool to run a baseline test, which is a BreakingPoint port-to-port test, to verify the test tool's performance. Note that the network configurations must change between running the port-to-port and DUT test. Physical cabling will change to connect

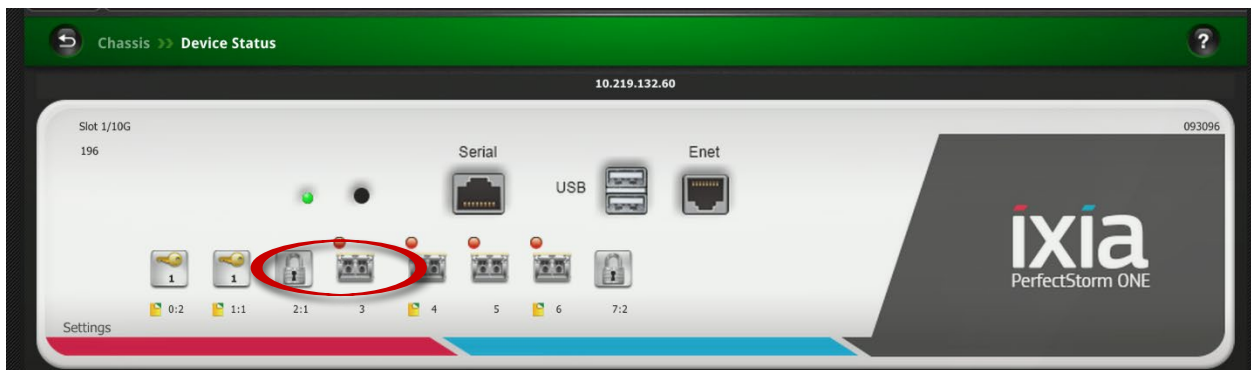
Test Methodologies for Traffic Fuzzing and Negative Testing

the test ports to the DUT. A layer 2 switch that has a high-performance backplane is recommended in a switched environment. Once you have obtained the baseline performance, setup the DUT and the test tool as per the above test diagram and below step by step instructions.

1. Open a Web browser and connect to the BreakingPoint Web GUI.
2. Reserve the test ports to be used in the physical test setup to generate/receive traffic:
 - a. Click on the Device Status button located on the upper right corner:



- b. In the new screen select the physical ports that are to be used in the test. Please note that in this example, a Perfect Storm One appliance is used.



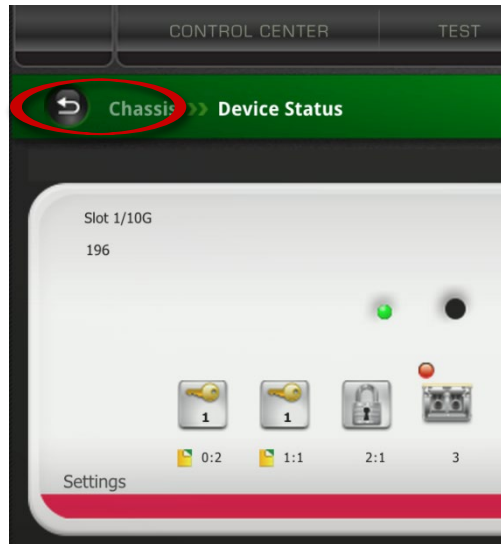
In this example, we will use ports 0 and 1 of the Perfect Storm ONE 8x10GE appliance.

Note: An important thing to remember when reserving your ports is the order in which you reserve them. Whenever you reserve a port, the system will automatically map that port to an interface on the chassis.

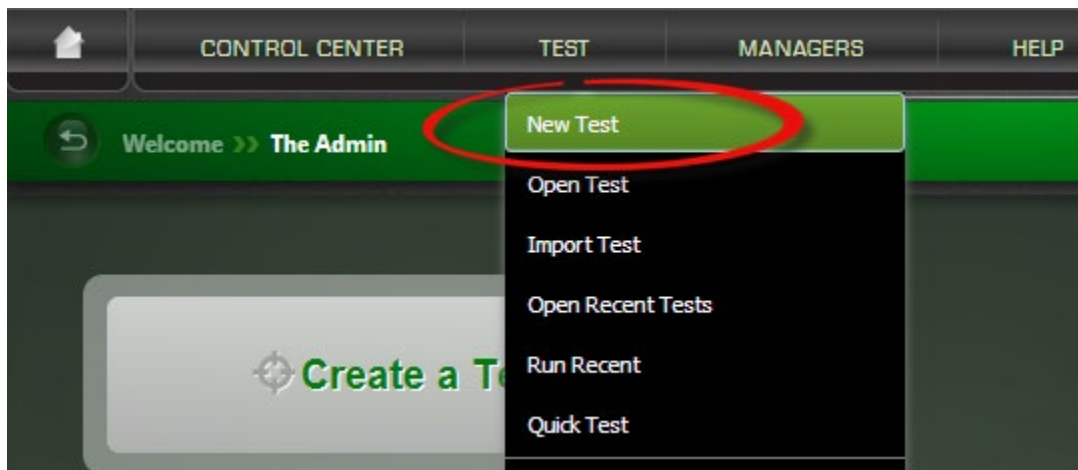
Note: The resources of each blade are allocated in proportion to the number of ports reserved on the blade. In some cases, it may be necessary to reserve additional ports to secure enough resources for the test being performed. The sessions and bandwidth available to a test may be insufficient to adequately perform the test if too few ports have been reserved.

Test Methodologies for Traffic Fuzzing and Negative Testing

- c. Once the proper test ports have been selected click on the back arrow to return to the initial screen:



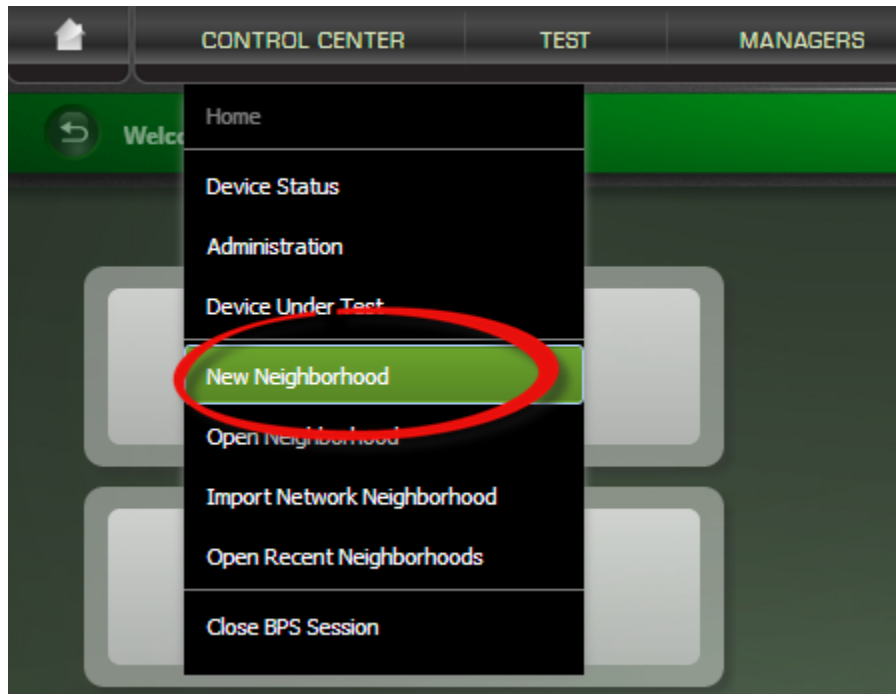
3. Next, select **Test** -> **New Test** option from the upper menu bar to start with configuring the test:



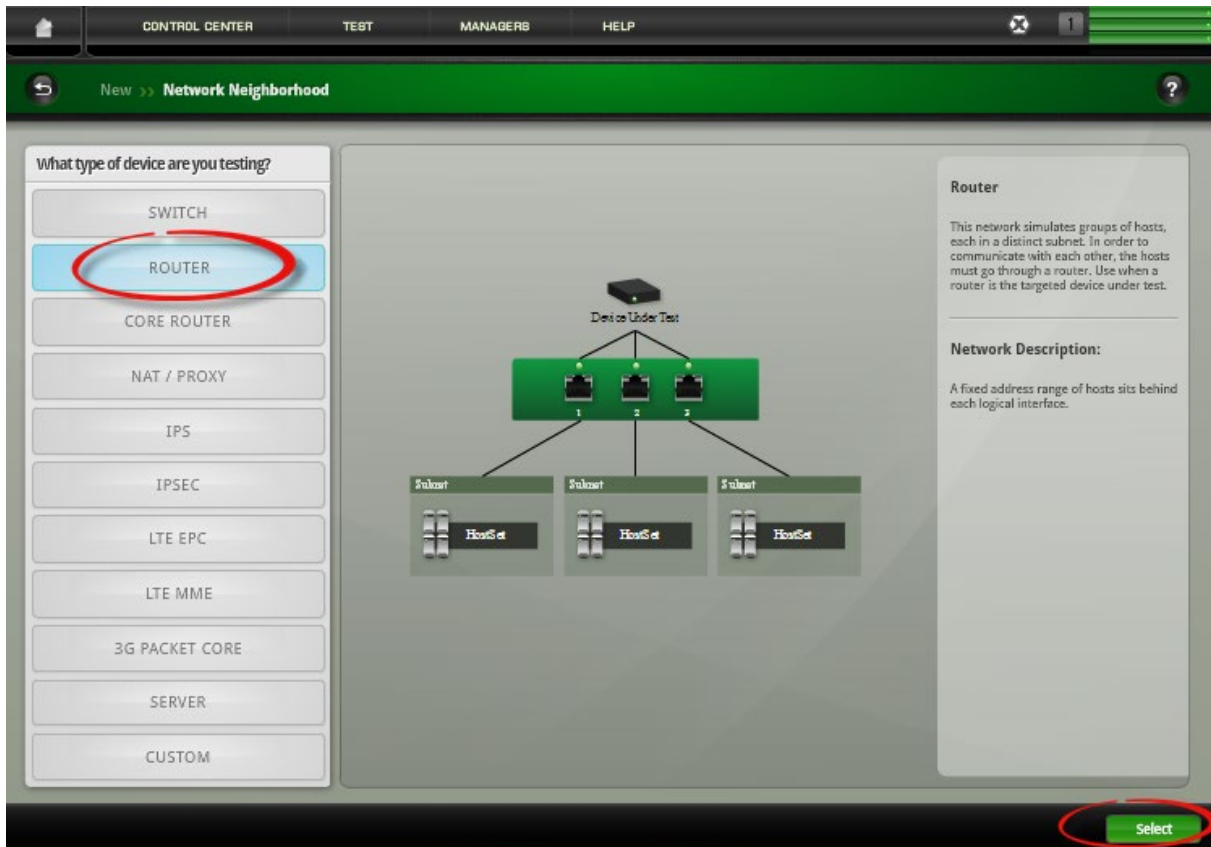
4. Since we already reserved the physical test ports (and, if applicable extra ports for more resource reservation) now we need to configure the MAC and IP layer parameters like MAC address VLANs, IP addressing scheme, MTU, etc. All these settings, among others are controlled/defined from the **Network Neighborhood**. The steps below show how to create a new Network Neighborhood. You can also use an existing Network Neighborhood or modify an existing one.

Test Methodologies for Traffic Fuzzing and Negative Testing

- a. From the upper menu bar select **Control Center** -> **New Neighborhood**:

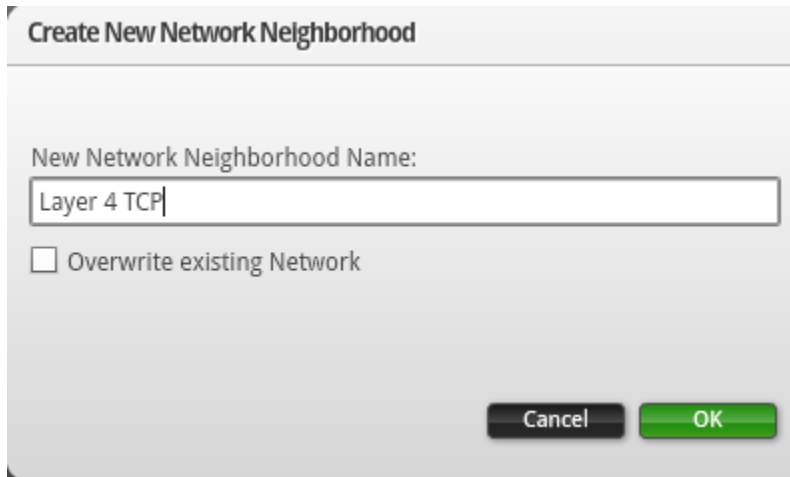


- b. Select the DUT type used for the test and a corresponding default Network Neighborhood will be created:



For this test, we will use *ROUTER* as DUT type.

- c. Enter an easy-to-remember name for the new Network Neighborhood and click **OK**:



- d. According to the DUT type we previously selected, a default Network Neighborhood will be created. Since we chose *ROUTER* DUT type, the following elements are automatically created:
 - i. Two *INTERFACES*: Provides access to interface level parameters like MTU, MAC, Impairments, and Packet Filters.
 - ii. One *IPv4 EXTERNAL HOSTS*: configures the IP address of the external end hosts/servers. This element is not required for devices that are Pass-Through. It is only needed for devices that are terminating the TCP Connection (e.g. Server Load Balancers).
 - iii. Two *IPv4 STATIC HOSTS*: Provides access to IP related parameters of the BreakingPoint emulated hosts. The Static Hosts represents the BreakingPoint TCP emulated clients and servers. In this example, we will use one element for the TCP emulated clients and another element for the TCP emulated servers.

Entry Mode		Diagram Mode		ADD NEW ELEMENT		EXPAND ALL COLLAPSE ALL		KEYBOARD SHORTCUTS	
▼ INTERFACE: (2)					Untagged Virtual Interface				
DEL	ID	Number	MTU	MAC Address	Duplicate MAC Address	VLAN Key			
×	Interface 1	1	1500	02:1A:C5:01:00:00	<input type="checkbox"/>	Outer VLAN ▼			
×	Interface 2	2	1500	02:1A:C5:02:00:00	<input type="checkbox"/>	Outer VLAN ▼			
× ▶ IPv4 EXTERNAL HOSTS: (1)					External hosts used as a test target				
× ▶ IPv4 STATIC HOSTS: (2)					Simulated IPv4 endpoints				

Test Methodologies for Traffic Fuzzing and Negative Testing

Note: Make sure to configure the IP and MAC addresses according to physical test setup address assignment scheme.

The screenshot shows the configuration interface for Layer 4 TCP. The 'IPv4 STATIC HOSTS' section is highlighted with a red box. It contains a table with the following data:

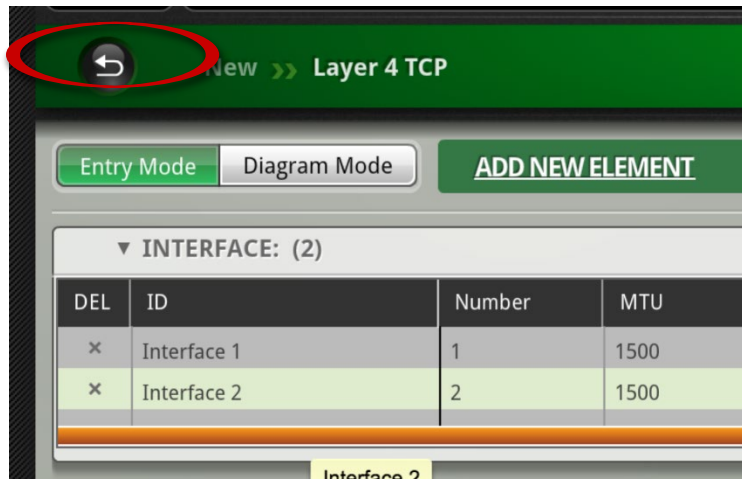
DEL	ID	Container	Tags	Base IP Address	Count	Gateway IP Address	Netr
x	Static Hosts i1_default	Interface 1	Lab Client,i1_default	198.18.0.2	1000	198.18.0.1	16
x	Static Hosts i2_default	Interface 2	Lab Server,i2_default	198.19.0.2	1000	198.19.0.1	16

- e. Save the new Network Neighborhood configuration by clicking the **Save** button located on the lower right corner:

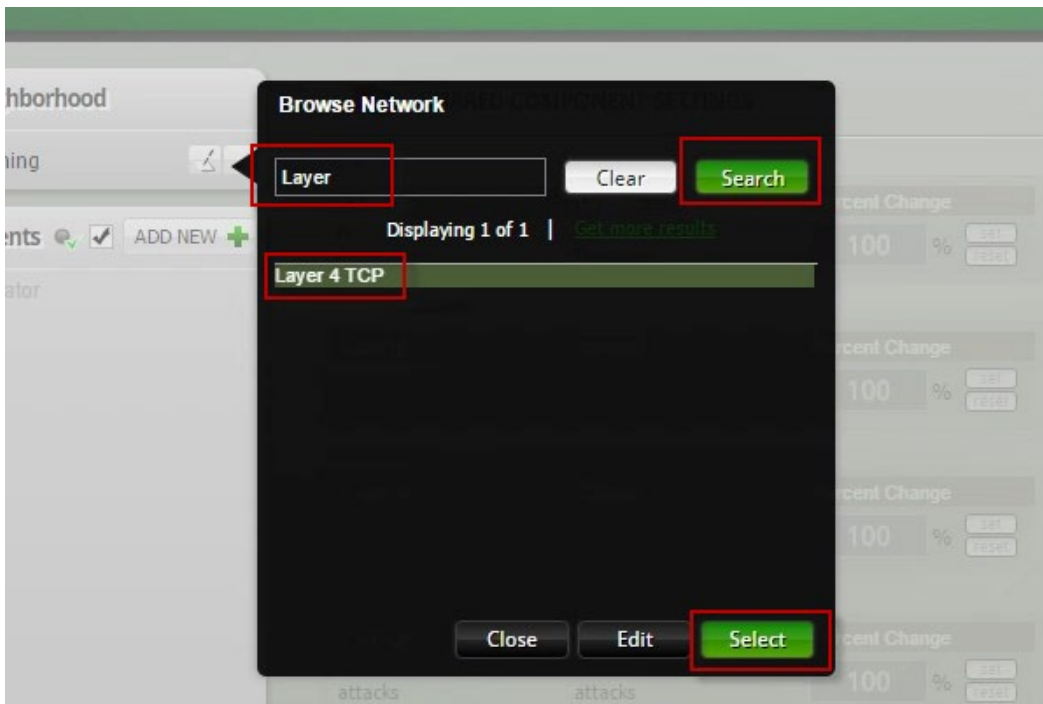
The screenshot shows the same configuration interface as above, but with the 'Save' button in the bottom right corner highlighted with a red circle. The 'Save' button is green and located next to 'Save As' and 'Close' buttons.

Test Methodologies for Traffic Fuzzing and Negative Testing

- f. Click on the back arrow to return to the main test screen:



- g. From the main test screen click on the browse button from the **Network Neighborhood** section to search and select the above created Network Neighborhood:



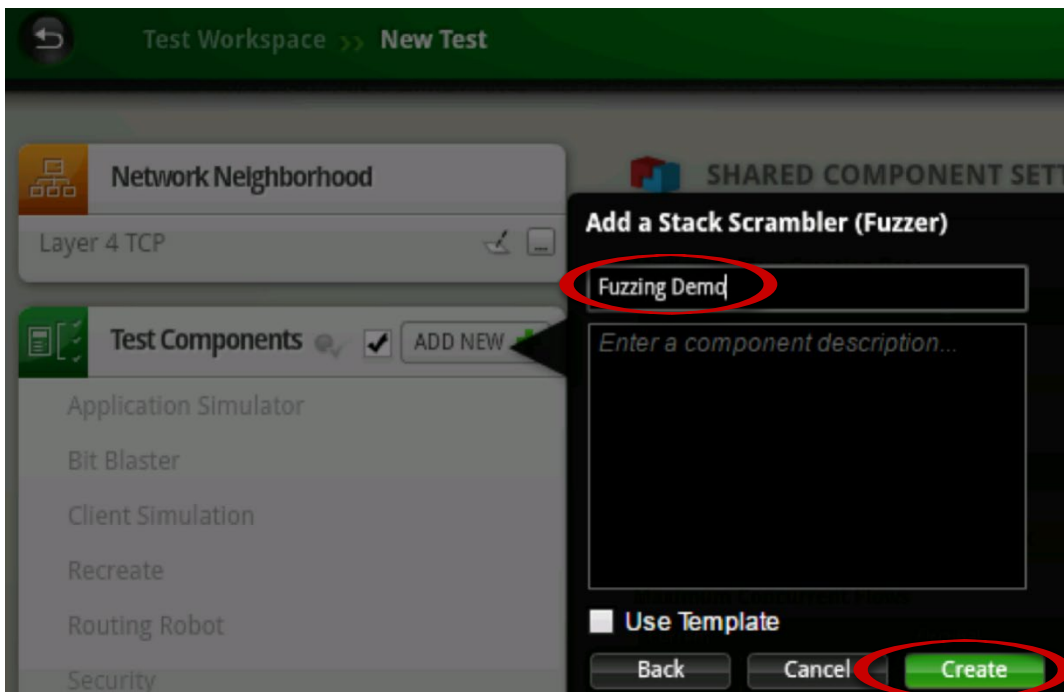
After configuring the MAC and IP layer parameters the actual traffic component needs to be created as well.

Test Methodologies for Traffic Fuzzing and Negative Testing

5. Click on the **ADD NEW** button from the **Test Components** section. Choose *Stack Scrambler* from the selection list and click on **Select** button:

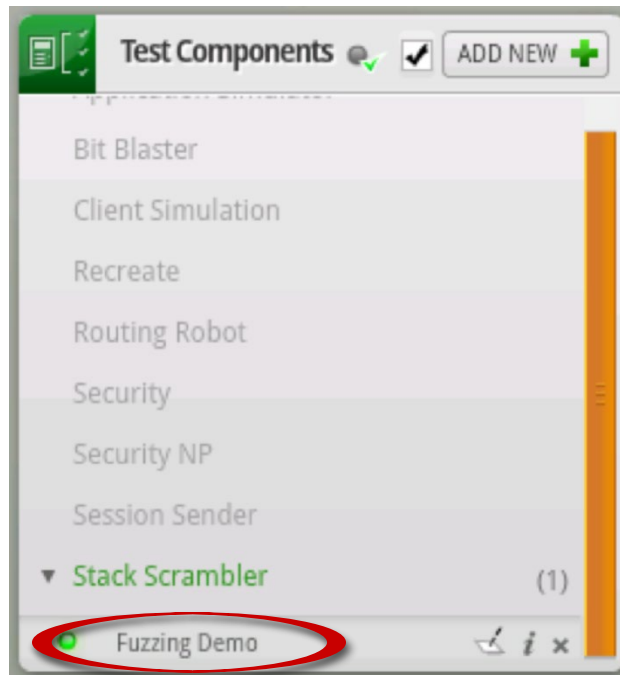


6. Rename the component from *StackScrambler_1* to *Fuzzing Demo* or anything else meaningful for your test; then, click **Create** button:

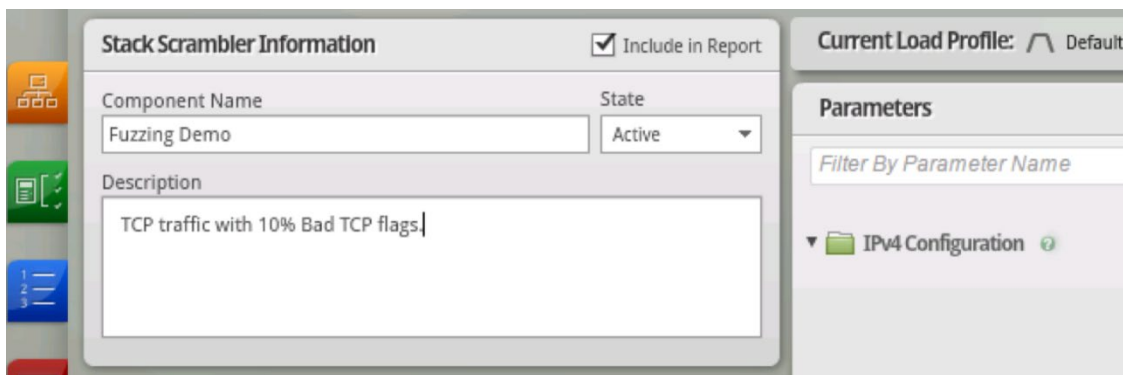


A new entry (i.e. *Fuzzing Demo*) will be created under **Stack Scrambler (Fuzzer)** Test Component.

7. Click on the newly created component to edit its parameters:

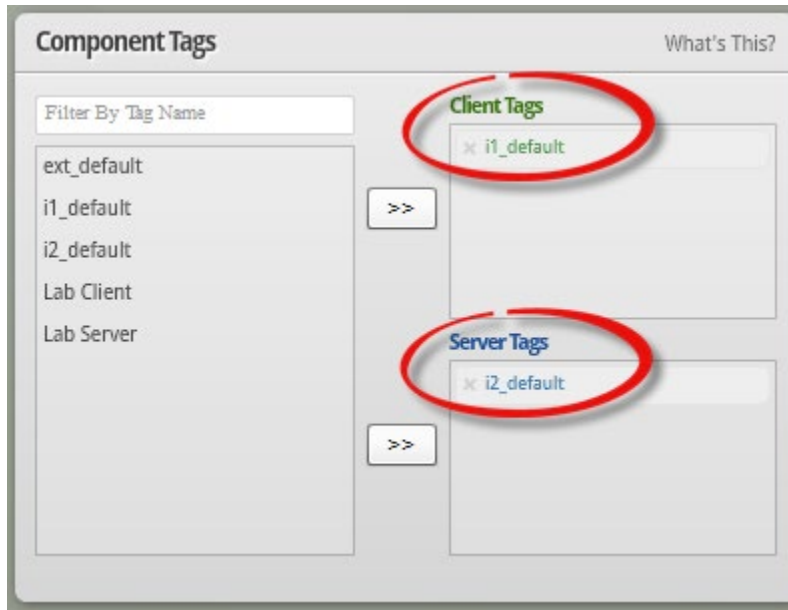


8. In the next steps, we will configure the *Fuzzing Demo* test component:
 - a. A meaningful description can be added in the **Description** box for easy reference of what this particular component is configured for:



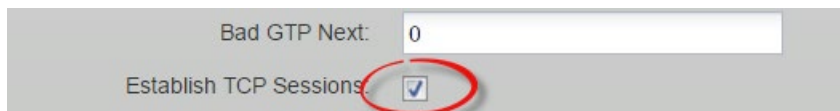
- b. In the **Component Tags** section make sure to assign the proper interface tags. For the **Client Tags** assign the tag corresponding to the Static Host element emulating the TCP client as configured in the Network Neighborhood. For the **Server Tags** assign the tag

corresponding to the Static Host element emulating the TCP servers as configured in the Network Neighborhood:



- c. The **Parameters** section contains multiple test component attributes that are grouped into category folders based on their functional purpose. For the scope of this test the following parameters will be modified:
 - i. Group: **Payload**, Parameter: **Transport**: Select *TCP* from the drop-down list (only TCP traffic will be generated and impaired)
 - ii. Group: **Data Rate**, Parameter: **Minimum Data Rate**: configure the minimum data rate *1000*. This value is in Megabits/second.
 - iii. Group: **Stack Scrambler**, Parameter: **Bad TCP Flags**: change the default value of 0 to *10* (the percentage of TCP packets with bad TCP Flags).

Note: If fuzzing traffic is passing through a stateful device such as a firewall, it is important to enable the **Establish TCP Sessions** parameter. Otherwise, malformed TCP packets will be dropped.



For this test, the remaining parameters can be left to their default values.

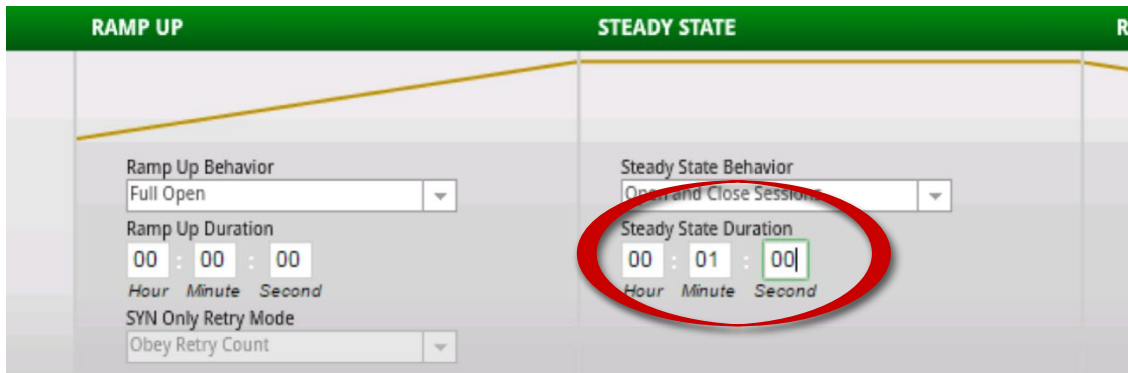
d. Configure the test traffic pattern using the Load Profile section:

i. Click on the **Edit Load Profile** button:

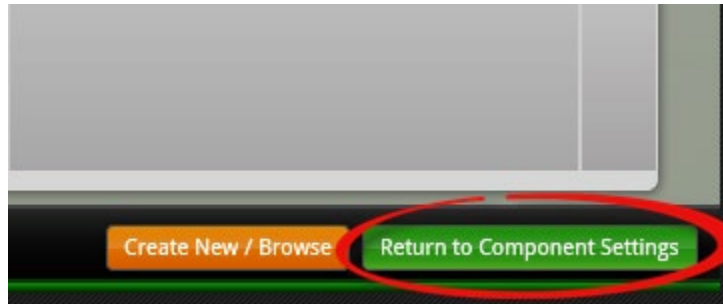


For this test, we will use the default traffic pattern; only the duration of the sustain state will be change to 1 minute:

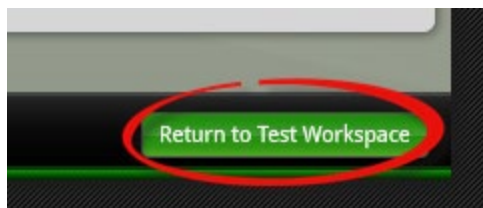
ii. Set the Steady State Duration from 30 seconds to 1 minute:



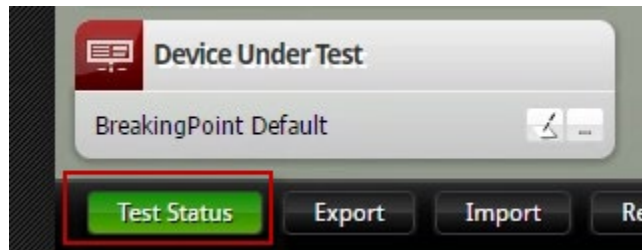
iii. Click on the **Return to Component Setting** button to go back to the Test Component configuration screen:



e. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:

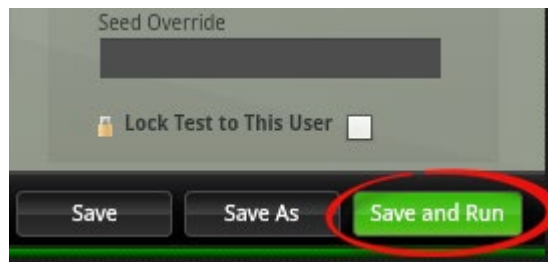


9. Make sure the **Test Status** indicated (on the lower left corner) is now green:



If there is not, determine what is wrong by selecting **Test Status** and viewing the errors.

10. Select **Save and Run** from the lower right corner:



11. If the test has not previously been saved, enter a name for the test and click **Save**:



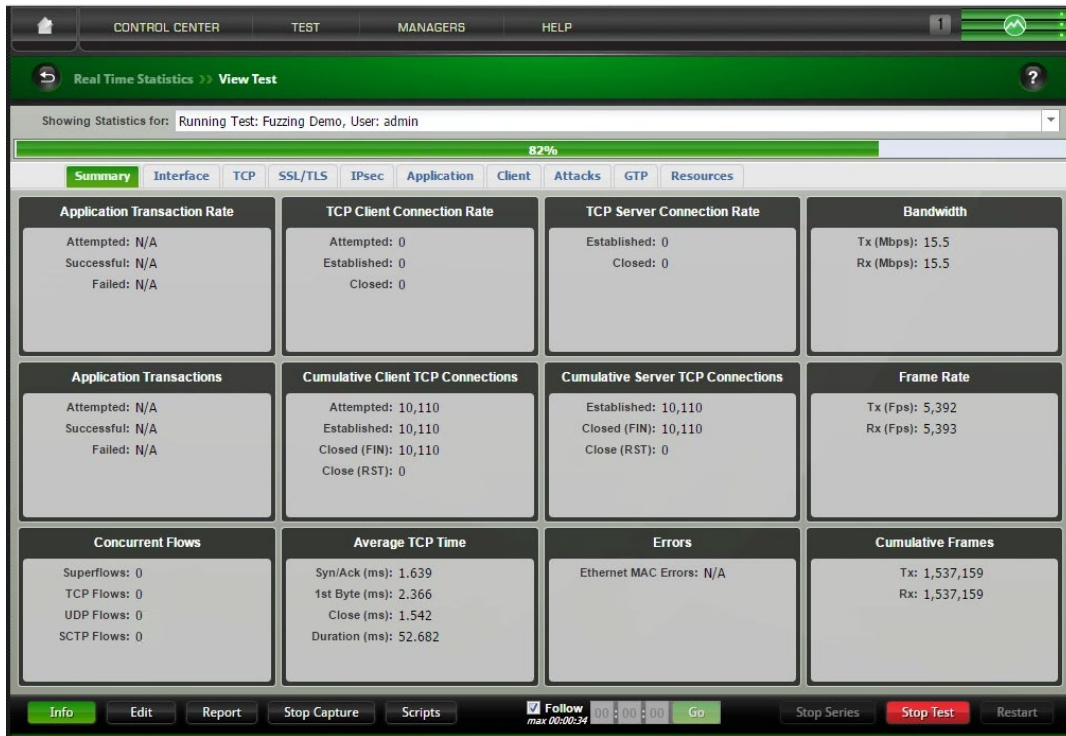
Run the test and while the test is running continue to monitor the DUT with respect to the target rate and any failure/error counters. See the **Results Analysis** section for important statistics and diagnostics information.

Real-time Statistics

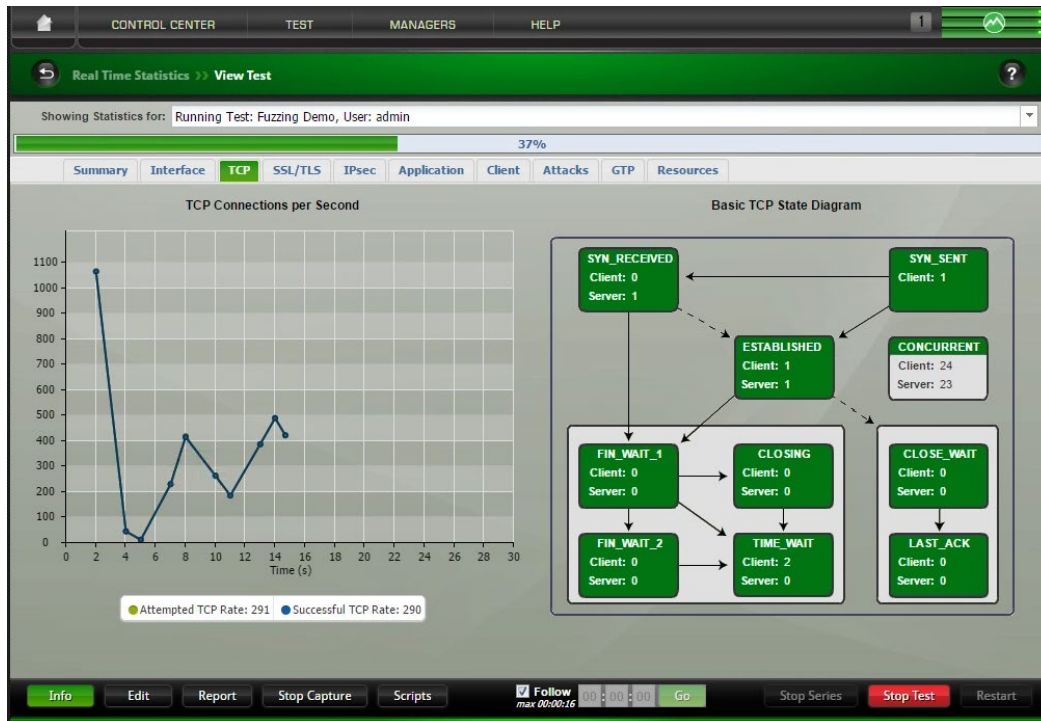
After the test is started and initialized, the screen will automatically switch to **Real Time Statistics** window.

Test Methodologies for Traffic Fuzzing and Negative Testing

The **Summary** tab presents basic metrics that provide a good understanding of the overall test progress, the most relevant for our test case being TCP Connection Rate, Cumulative Connections as well as Interface Bandwidth and Frame Rate.

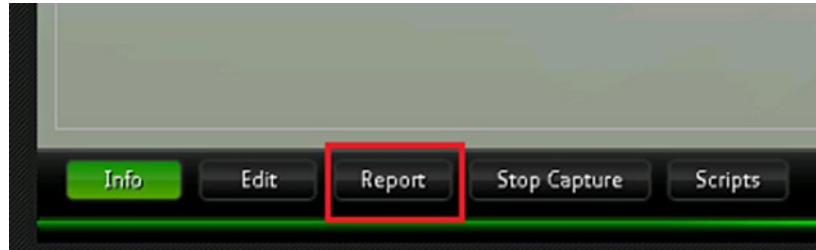


The **TCP** tab provides instant access to key statistics that should be examined for failures at the TCP level.

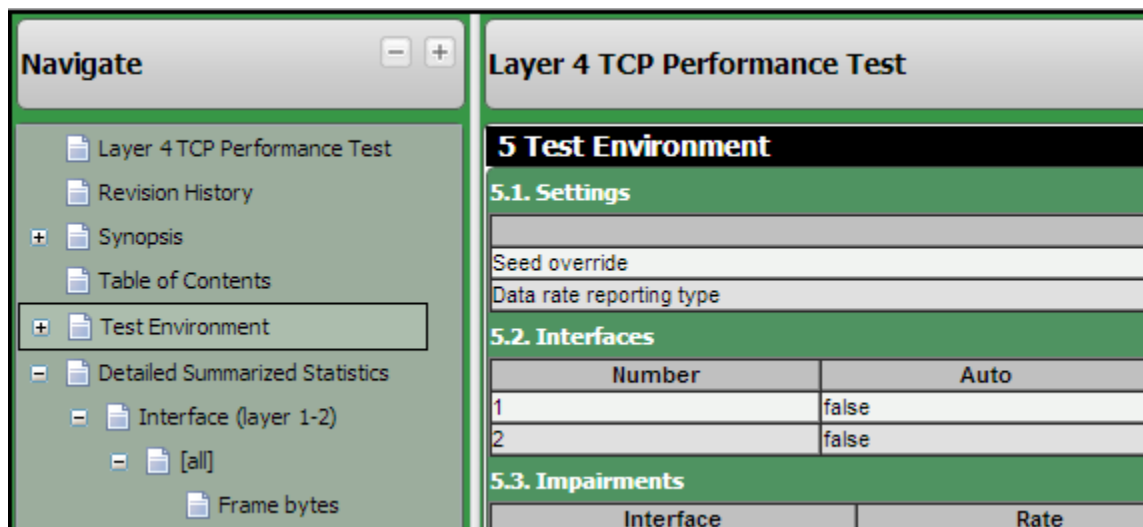


The TCP view can quickly indicate if the device is unable to keep up with the targeted test objective (i.e. TCP Connection Rate: Attempted vs. Successful).

Beside the real-time statistics, detail post-test analysis can be performed. In the lower left corner of the Real Time Statistics window, click the **Report** button to view detailed results. This will open the results in a new browser window.



On the left side of the detailed report window is the navigation panel, where you can navigate and browse the results. The results and test information will be displayed on the right side of the browser:

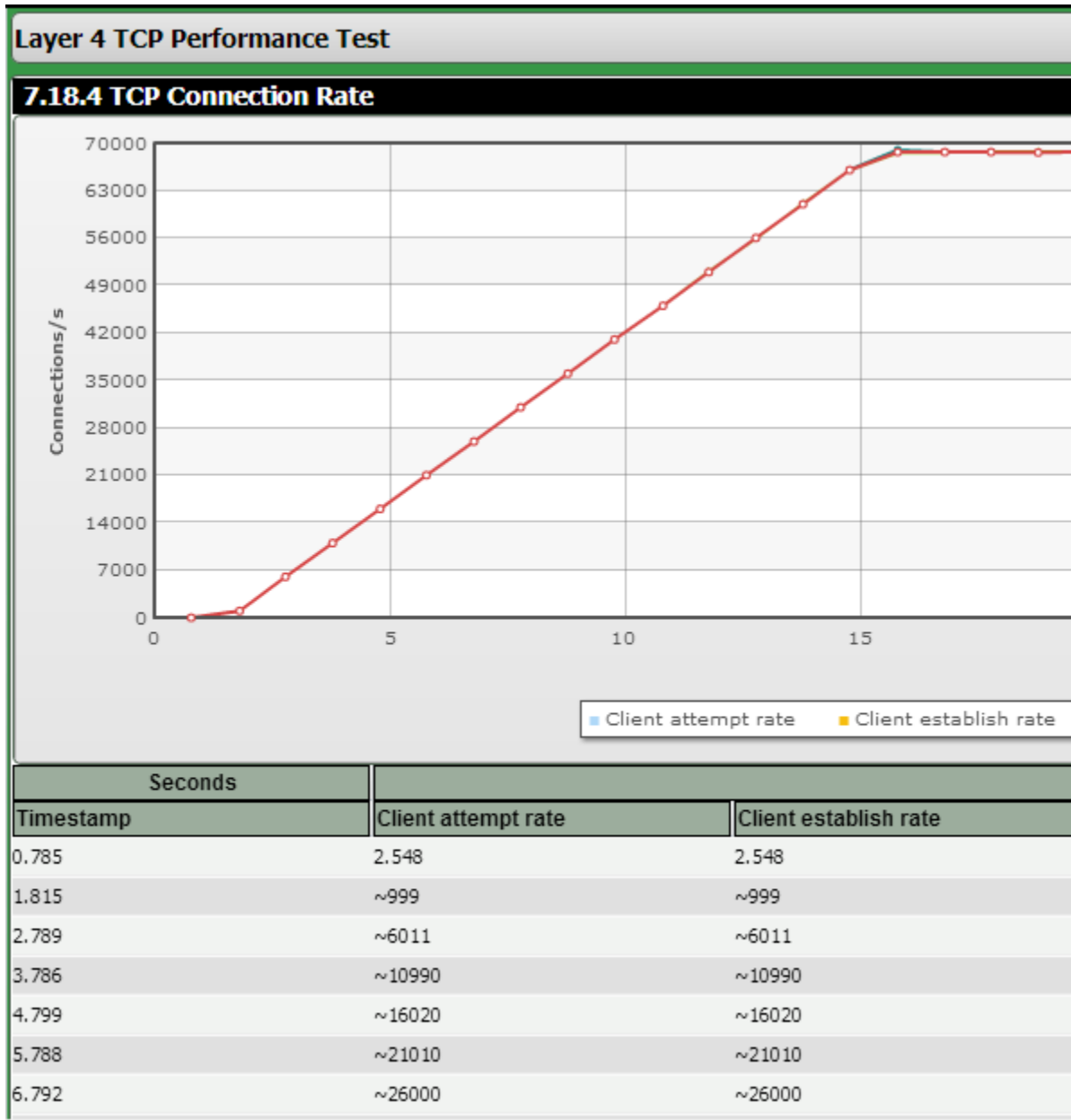
A screenshot of a web browser window displaying a detailed report. On the left is a 'Navigate' panel with a tree view containing folders like 'Layer 4 TCP Performance Test', 'Revision History', 'Synopsis', 'Table of Contents', 'Test Environment', 'Detailed Summarized Statistics', 'Interface (layer 1-2)', '[all]', and 'Frame bytes'. The 'Test Environment' folder is selected. The main content area is titled 'Layer 4 TCP Performance Test' and contains sections: '5 Test Environment', '5.1. Settings' (with fields for 'Seed override' and 'Data rate reporting type'), '5.2. Interfaces' (with a table), and '5.3. Impairments' (with a table).

Number	Auto
1	false
2	false

Interface	Rate
-----------	------

From the navigation panel go to: **Test Results for TCP Sessions folder -> Detail -> TCP Connection Rate**. Once TCP Connection Rate has been selected, a graph will display the

Client and Server attempt rate, establish rate and close rate. Also, a table is displayed showing the values used to create the graph.



Test Variables

BreakingPoint can generate in the same time valid and invalid (fuzzed) traffic; to validate the capability of the DUT to still handle valid traffic while fuzzed traffic is present, you can add other test components, e.g. Application Simulator.

Multiple parameters can be fuzzed; the test should be repeated for various parameters and various percentages of affected packets – in this test, only the TCP flags of 10% of the packets are invalid.

Test Methodologies for Data Leakage or Data Loss Prevention

Enterprises need to protect its confidential information or proprietary intellectual property from leaking outside of the company. Whether by the hand of hackers or, more likely careless or malicious employees, the cost of exfiltration of that valuable data is high.

The confidential data varies from industry to industry and can be anything that matches a specified pattern. As an example, certain keywords embedded in office documents may trigger the DLP (Data Leakage/Loss Prevention) mechanism. In other examples, the vendors must look to detect leakage of one or more selected documents. In many cases, such documents consist in confidential data that is not accessible to the IT administrator, so vendors provide tools that generate a signature for such documents and later use it as a “signature” for the particular version of the document.

For the medical sector, leakage of patient information is considered confidential and must be withheld within the premises of the hospital’s virtual network. As an example, in USA hospitals must comply with HIPA policies.

Payment Card Industry (PCI) compliance is a complex and ever evolving subject affecting millions of businesses – acquiring banks, Independent Sales Organizations (ISOs), processors, hosts, shopping carts, e-commerce and retail merchants and other merchant services providers. The Payment Card Industry Data Security Standard (PCI DSS) is a set of requirements designed to ensure that ALL companies that process, store or transmit credit card information maintain a secure environment. Essentially any merchant that has a Merchant ID (MID). A DLP device protecting such devices will need to recognize different credit card & debit card patterns from various financial institutions.

Typical enterprises and organizations use DLP devices to protect their most valuable information – their intellectual property (IP). This information can reside in Microsoft Office documents such as business presentations, product requirements documents, design documents (CAD), customer or sales databases and/or source code (C, C++, Java, ASP)

The DLP testing follows the same concept with the anti-malware test cases described above, except the malware content consists in legitimate files without malicious content, but represented by those files embedding confidential information. Another key difference is that the DLP inspection is focused on outbound communication, rather than both inbound and outbound.

The concept of DLP can be further extended to cover Lawful Intercept which refers to monitoring of application traffic for certain key words like ‘bomb’ or ‘financial results’ or certain triggers like credit card or social security numbers. Organizations tasked with monitoring criminal communications or ethical behavior rely heavily on the performance and accuracy of lawful intercept (LI) systems to analyze the flow of digital messages across networks. Missing a single stream of data could prevent prosecuting a crime, stopping a terrorist attack, or complying with government regulations.

Test Case: Validating Basic Effectiveness of DLP Engines

Overview

Data loss prevention (DLP) engines provide content-aware, centralized policy control to identify, monitor, and protect sensitive data. But simply implementing a DLP system doesn't ensure proper protection. This test enables users to measure how well DLP policies actually detect and protect, and to optimize DLP accuracy for optimal security. BreakingPoint System provides more than 300 applications including social media, file transfer, P2P and a portfolio of major web mail protocols to simulate a traffic mix that is representative of your own networks. It also provides the flexibility to customize application flows such as email with real implanted content to emulate the data exfiltration behavior.

DLP system performance varies with data load, only real-world workload testing at scale takes the guess work out of deploying reliable DLP systems. The very purpose of DLP solutions, their reliance upon DPI, intolerance for latency, and the dynamic and complex nature of the Internet traffic they are designed to inspect make validating accuracy, performance, and scalability a challenge.

Objective

Measures the effectiveness of the DLP system to prevent data exfiltration under realistic workload. In this test case, we will use a transactional data protocol as the application to extract an archive of customer contact details, in the midst of a realistic load of enterprise application traffic mix to assess the accuracy and performance when the inspection rules enabled of the DLP engine.

Setup

This setup requires one initiator/client and one responder/server test port to emulate the entire enterprise application mix and the exfiltration traffic flow. However, more ports can be used to scale the performance of the test tool.

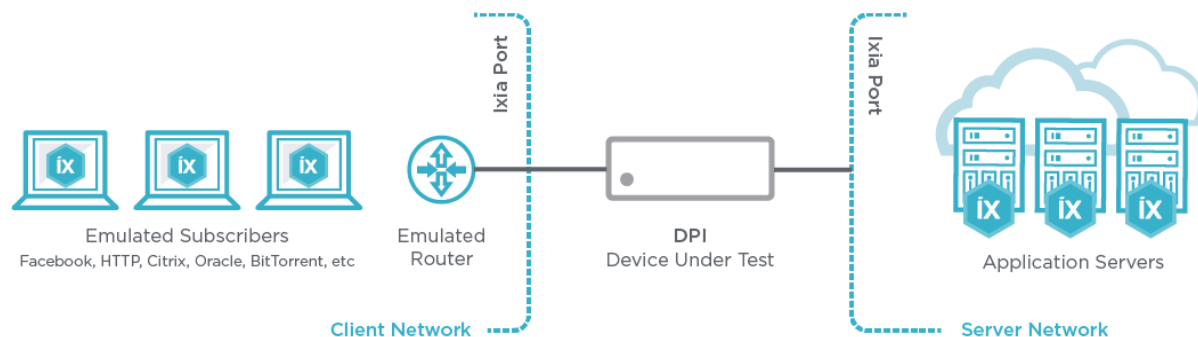
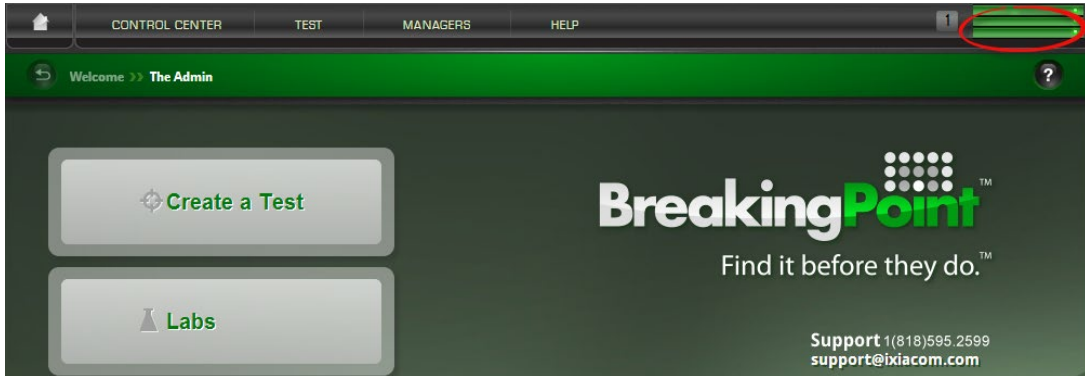


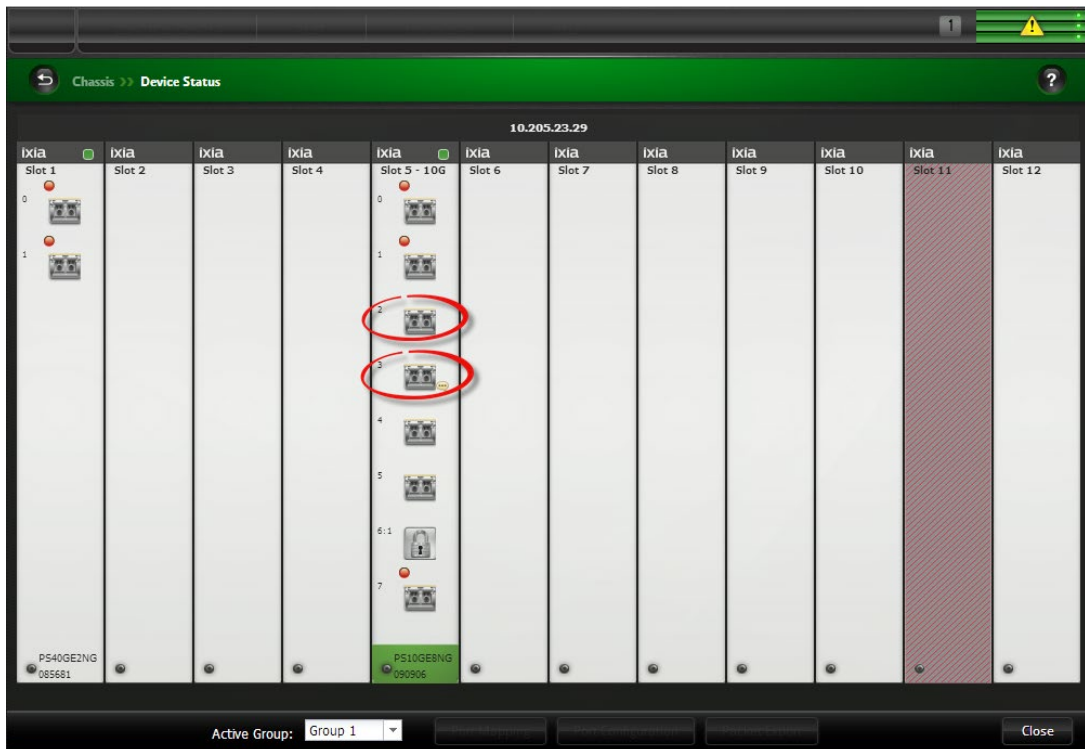
Figure 36. DPI Accuracy Test Topology

Step-by-Step Instructions

1. Open your favorite Web browser and connect to the BreakingPoint Web GUI.
2. Reserve the test ports to be used in the physical test setup to generate/receive traffic:
 - a. Click on the Device Status button located on the upper right corner:



- b. In the new screen select the physical ports that are to be used in the test:



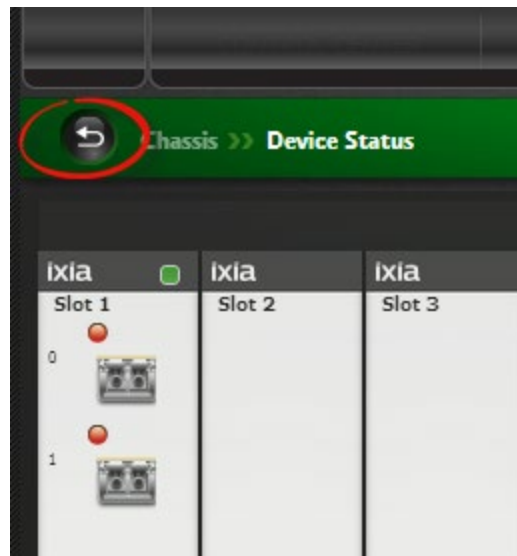
In this example, we will use ports 2 and 3 from the blade located in slot number 5.

Test Methodologies for Data Leakage or Data Loss Prevention

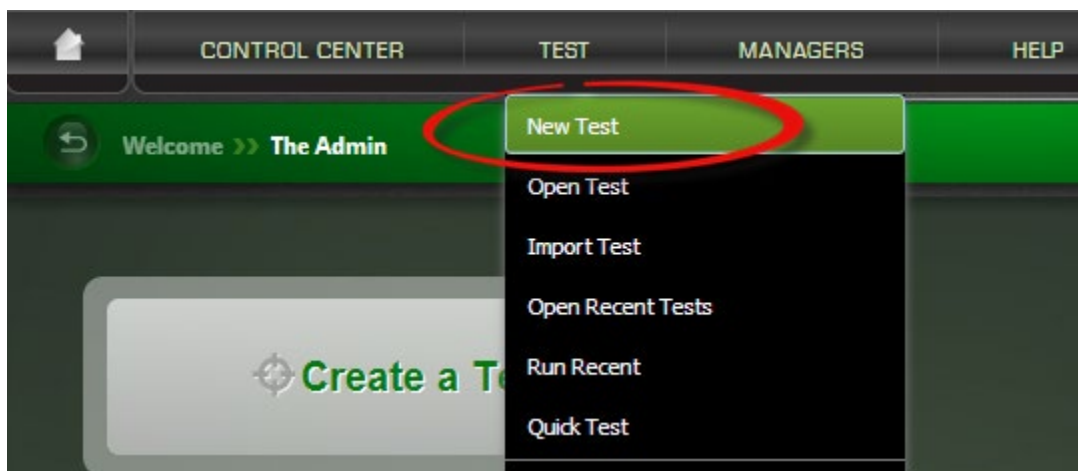
Note: An important thing to remember when reserving your ports is the order in which you reserve them. Whenever you reserve a port, the system will automatically map that port to an interface on the chassis.

Note: The resources of each blade are allocated in proportion to the number of ports reserved on the blade. In some cases, it may be necessary to reserve additional ports to secure enough resources for the test being performed. The sessions and bandwidth available to a test may be insufficient to adequately perform the test if too few ports have been reserved.

- c. Once the proper test ports have been selected click on the back arrow to return to the initial screen:



3. Next, select **Test -> New Test** option from the upper menu bar to start with configuring the test:



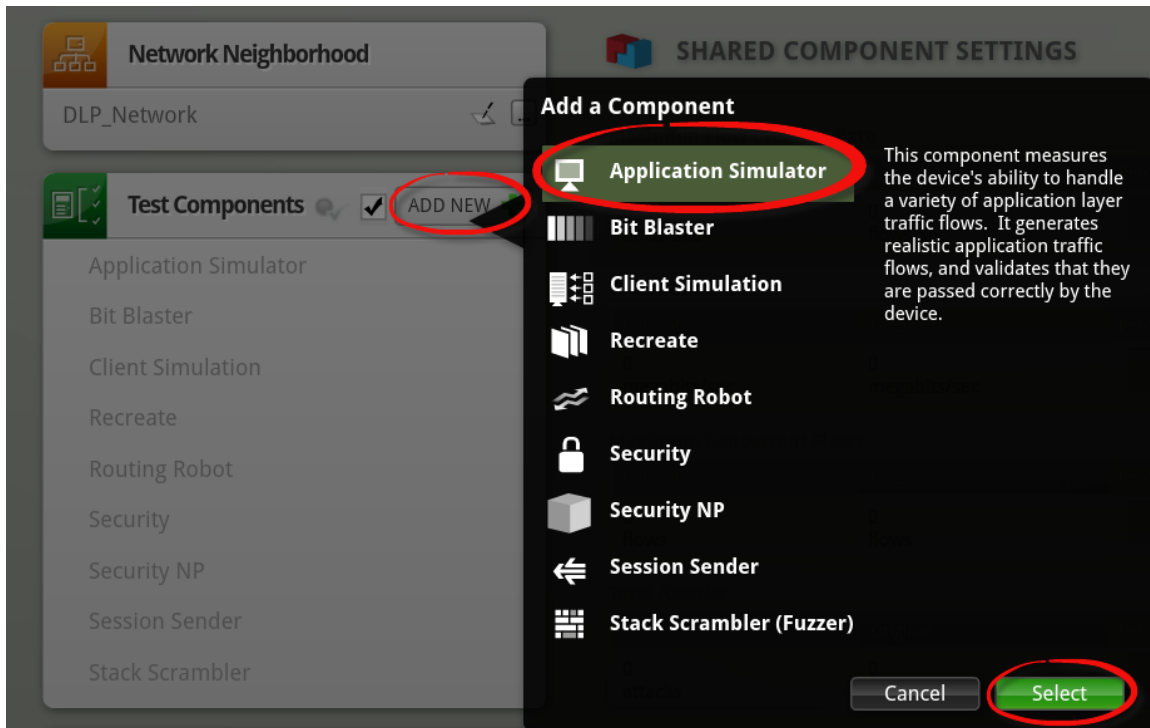
4. Since we already reserved the physical test ports (and, if applicable extra ports for more resource reservation) now we need to configure the MAC and IP layer parameters like MAC

Test Methodologies for Data Leakage or Data Loss Prevention

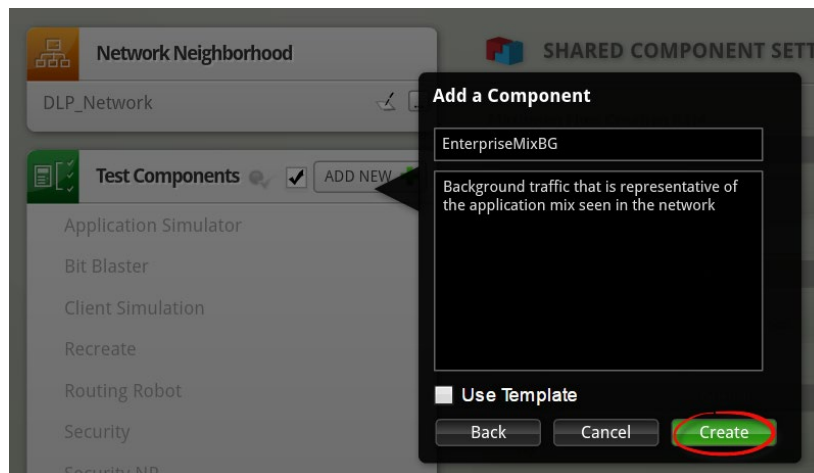
address VLANs, IP addressing scheme, MTU, etc. All these settings, among others are controlled/defined from the **Network Neighborhood**:

a. From the upper menu bar select **Control Center** -> **New Neighborhood**:

5. Click on the **ADD NEW** button from the **Test Components** section. Choose *Application Simulator* from the selection list and click on **Select** button:



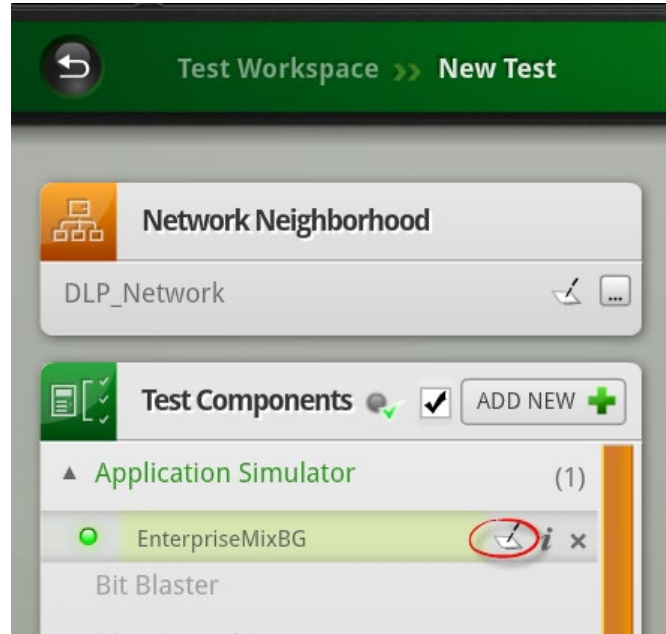
6. Rename the component from AppSim_1 to EnterpriseMixBG and click Create button. This AppSim component will simulate the background application traffic and provide the realistic workload that the DLP engine will be need to inspect and police.



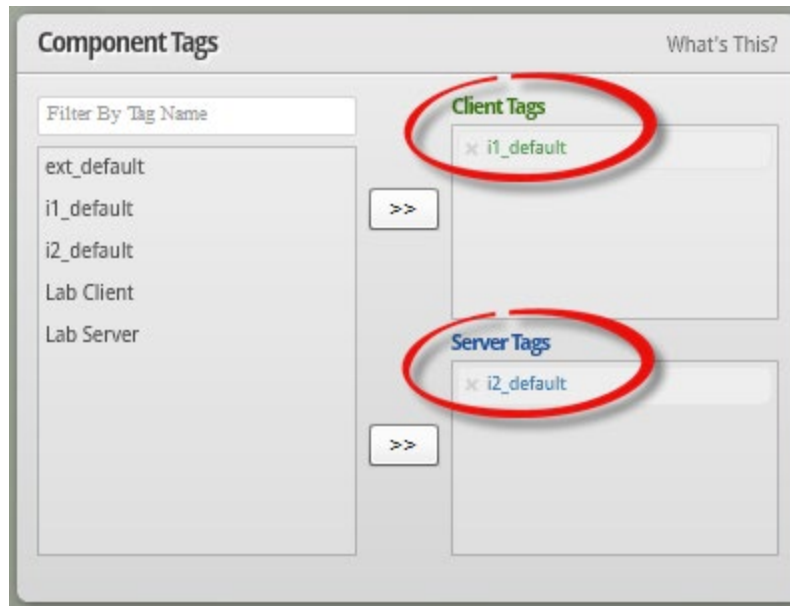
Test Methodologies for Data Leakage or Data Loss Prevention

A meaningful description can be added in the Description box for easy reference of what this particular component is configured for. A new entry will be created under Application Simulator Test Component.

7. Click on the newly created component to edit its parameters:

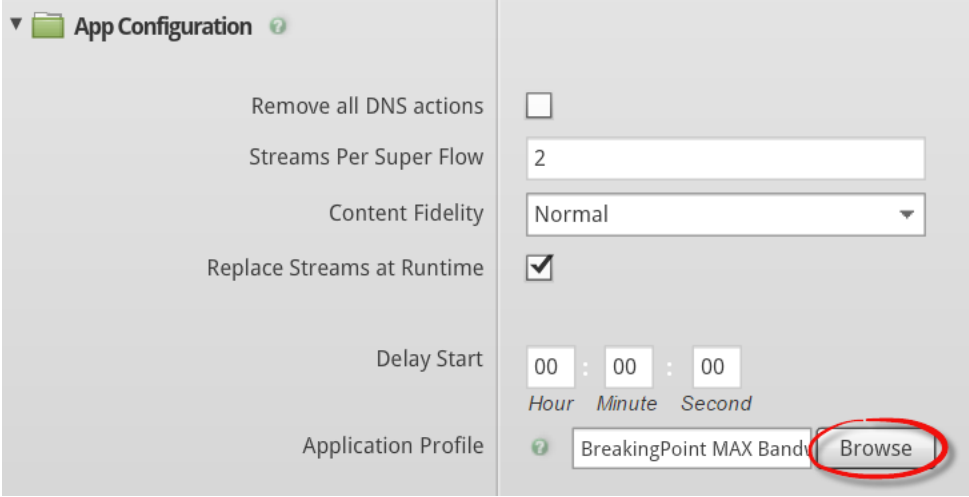


- a. In the **Component Tags** section, make sure to assign the proper interface tags. For the Client Tags assign the tag corresponding to the Static Host element emulating the TCP client as configured in the Network Neighborhood. For the Server Tags assign the tag corresponding to the Static Host element emulating the TCP servers as configured in the Network Neighborhood:



Test Methodologies for Data Leakage or Data Loss Prevention

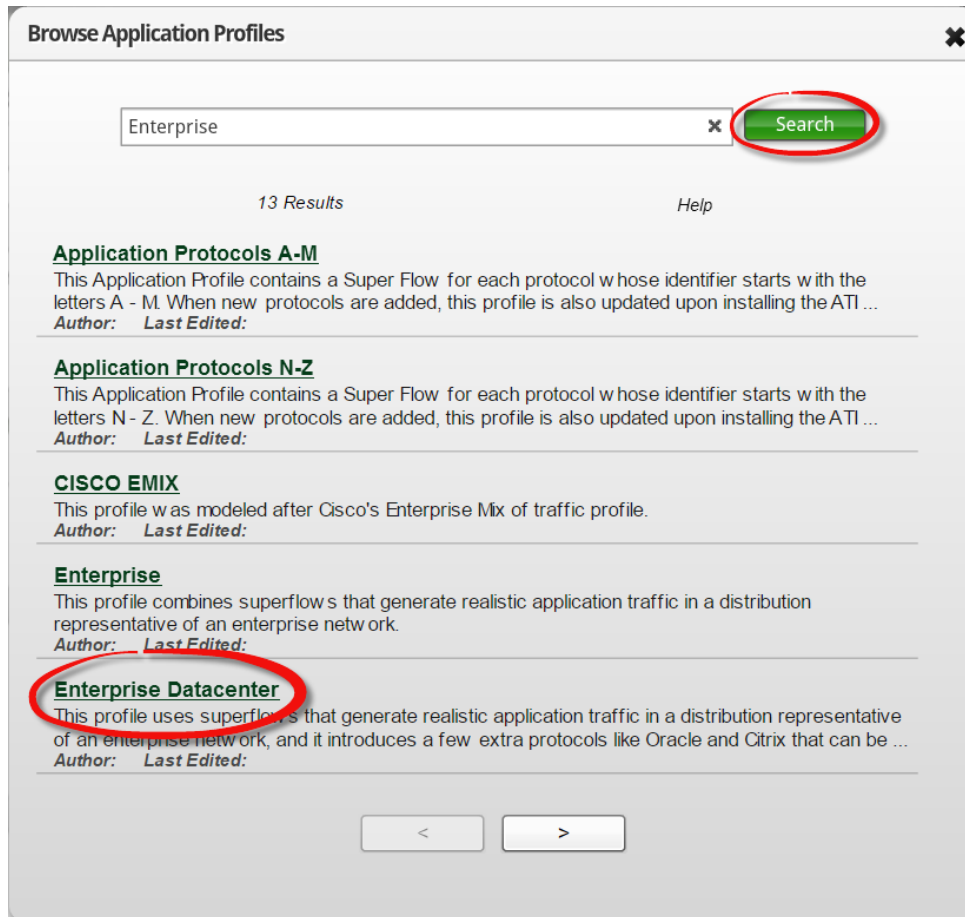
- b. The **Parameters** section contains multiple test component attributes that are grouped into category folders based on their functional purpose. For the scope of this test the following parameters will be modified:
- Minimum Data Rate:** configure the minimum data rate *10000*. This value is in Megabits/second.
 - Maximum Simultaneous Super Flows:** set the value to the maximum desired value (for this test we will set it to *1,000,000*).
 - Maximum Super Flows Per Second:** set the value to the maximum desired value (for this test we will set it to *1,000,000*).
 - Application Profile:** Browse to select an appropriate pre-defined application mix that is representative of your network traffic. For this test select *Enterprise Datacenter*



The screenshot displays the 'App Configuration' interface. On the left, a sidebar lists configuration options: 'Remove all DNS actions', 'Streams Per Super Flow', 'Content Fidelity', 'Replace Streams at Runtime', 'Delay Start', and 'Application Profile'. The right pane shows the corresponding settings: 'Remove all DNS actions' is unchecked; 'Streams Per Super Flow' is set to '2'; 'Content Fidelity' is set to 'Normal'; 'Replace Streams at Runtime' is checked; 'Delay Start' is set to '00 : 00 : 00' (Hour, Minute, Second); and 'Application Profile' is set to 'BreakingPoint MAX Bandwidth' with a 'Browse' button circled in red.

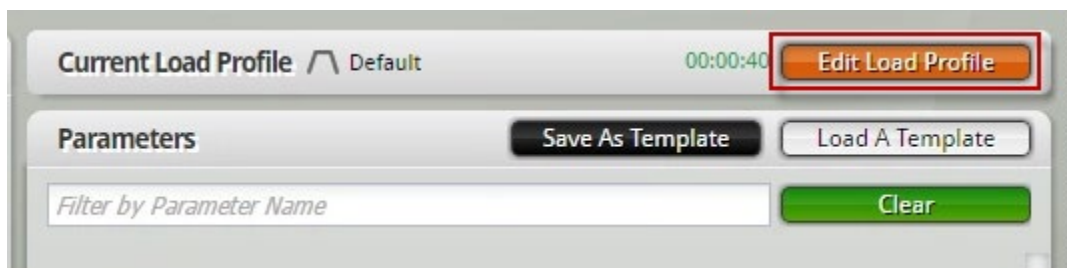
Test Methodologies for Data Leakage or Data Loss Prevention

- Search for Enterprise Datacenter pre-defined mix



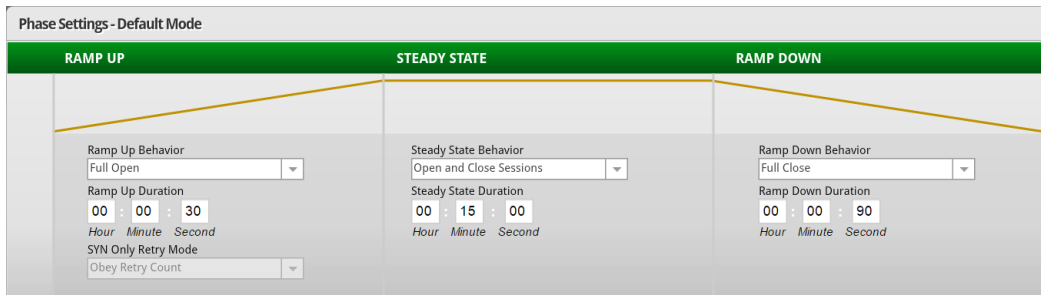
For this test, the remaining parameters can be left to their default values.

- c. Configure the test traffic pattern using the Load Profile section:
 - i. Click on the **Edit Load Profile** button located at the upper right:

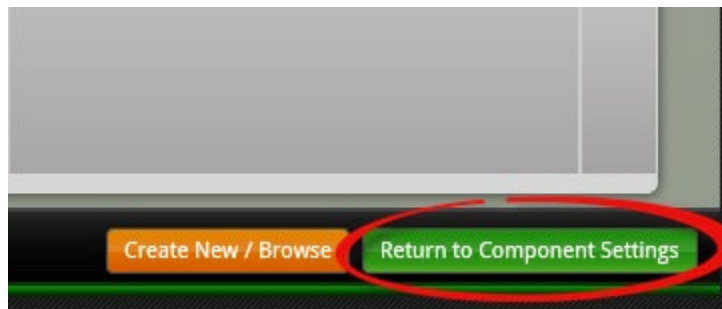


Test Methodologies for Data Leakage or Data Loss Prevention

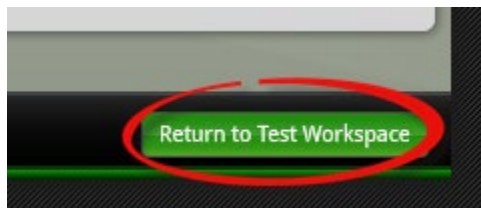
- ii. Modify the Steady State, Ramp-up and Ramp-down durations



- iii. Click on the **Return to Component Setting** button to go back to the Test Component configuration screen:



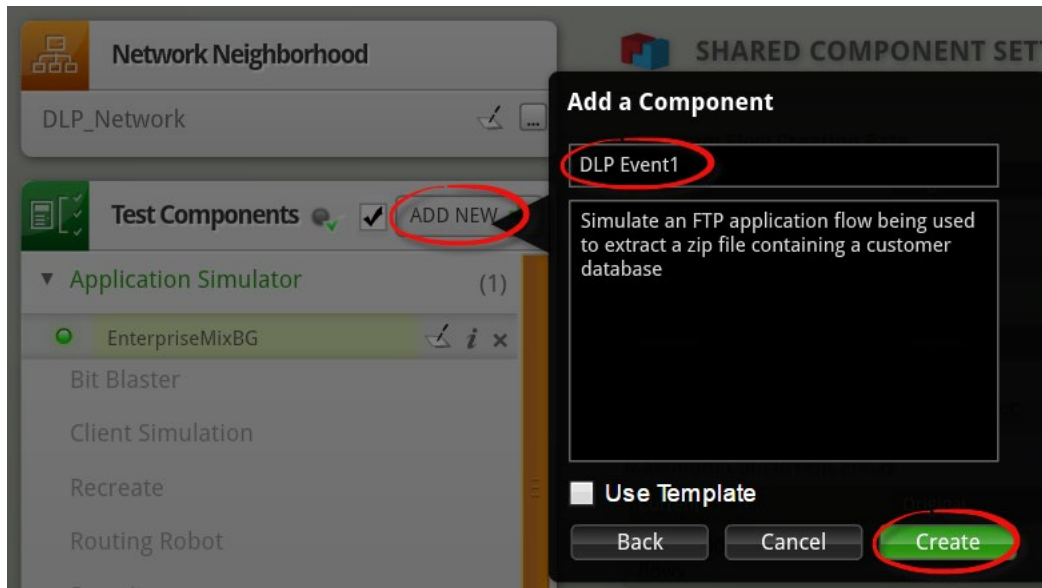
- d. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:



8. Now to configure the application traffic that will emulate the data exfiltration behavior. Click on the **ADD NEW** button from the **Test Components** section. Choose *Application Simulator* from the selection list and click on **Select** button.

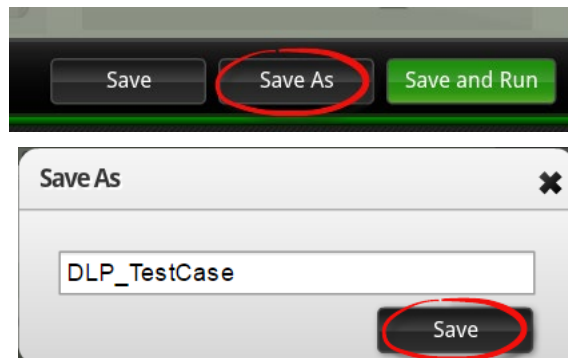
Test Methodologies for Data Leakage or Data Loss Prevention

- a. Rename the component from *AppSim_1* to *DLP Event1* and click Create button.

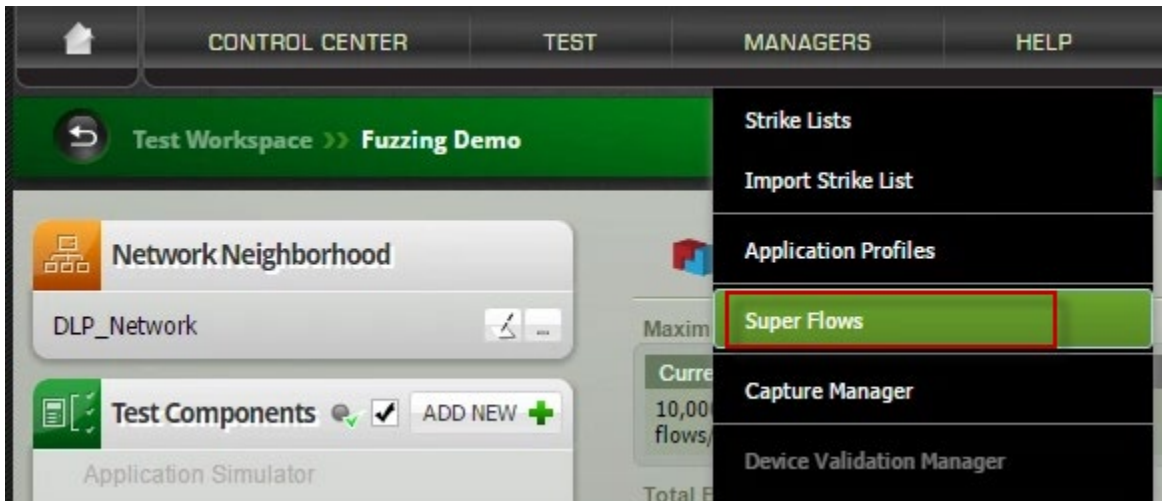


A meaningful description can be added in the Description box for easy reference of what this particular component is configured for. A new entry will be created under Application Simulator Test Component.

9. Select **Save As** from the lower right corner and enter a name for the test and click **Save**

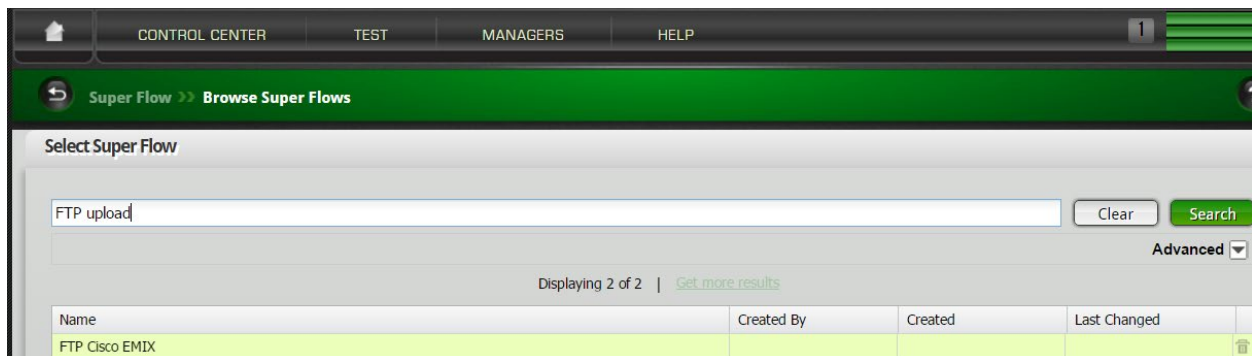


10. Before configuring the *DLP Event1* component, an application profile with the customized data protocol flow will need to be created. Select **Managers** -> **Super Flows** option from the upper menu bar to start customizing the flows.



11. Create an application flow to emulate the data exfiltration scenario. Since in a typical DLP scenario a client will “upload” a file or folder that may contain sensitive/tracked information. The Superflow search provides you option to go for such specific searches like “Upload” **Search** “Upload” to find some apps that have the upload command in it. Select the pre-defined ones from the search. Open one of the selected.

[Note: For this particular example, we have chosen FTP as the application however any other applications protocol can be used for DLP test as BreakingPoint allows the flexibility to edit the application flows and the data patterns]



12. Customize the application. By example, the super flow might consist of DNS and data transaction flows. In this, scenario the traffic is destined to a known server IP therefore the **DNS** flow can be deleted. Similarly, if the application needs to upload a file to outside server,

Test Methodologies for Data Leakage or Data Loss Prevention

therefore **Download** and other non-relevant actions can also be deleted by clicking on the trash can icon but keeping them won't impact the test.

The screenshot displays the 'Super Flow Details' configuration window. On the left, the 'Name' is 'FTP Cisco EMIX' and the 'Description' states: 'This Super Flow simulates an FTP session in which the client resolves the FTP server, logs in, downloads a file with a min size of 4kb, and then uploads the same file before quitting the session.[RFC 959][RFC 1035]'. The 'Category' is 'Testing and Measurement'. The 'Tags' section also contains 'Testing and Measurement'. At the bottom left, there is a 'Manage Hosts' button and a checkbox for 'Lock Super Flow to this user'. The main area is divided into two tables: 'Flows' and 'Actions'. The 'Flows' table has columns for '#', 'Protocol', 'Value', 'Client', and 'Server'. It lists two flows: '1' with protocol 'DNS' and 'Client' as 'Client', and '2' with protocol 'FTP' and 'Client' as 'Client'. The 'DNS' flow is circled in red. The 'Actions' table has columns for '#', 'Action', 'Value', 'F...', 'Flow', and 'Source'. It lists 11 actions, including 'Resolve', 'Welcome Banner', 'Login', 'Directory Listing', 'CWD', '230 CWD Response', 'Download', 'Upload', 'QUIT', '221 Goodbye', and 'Close'. Each action has a trash can icon next to it, indicating it can be deleted. At the bottom of the window, there are 'Export', 'Save As', and 'Save' buttons.

- Parameterize the **Upload** action. Have a file ready that contains certain keywords that you may be putting in the DLP at a later stage like the below doc that has some keywords like "Confidential", "iMotoAndroNex," etc.: Save it with a name like "Secret.docx"

QUARTERLY REPORT-HIGHLY CONFIDENTIAL

Q1-2016

To all our acquaintance, this is the quarterly report of our highly secret iMotoAndroNex phones. Please ensure you do not distribute this beyond this group. Also it's a violation to put this document on any file sharing, sites, FTP servers etc. Below are some of the pictures and the high-level architecture of our

Test Methodologies for Data Leakage or Data Loss Prevention

b. Upload the word doc in the client action

The screenshot displays the 'Super Flow Details' and 'Flows' sections. The 'Flows' table lists a single flow:

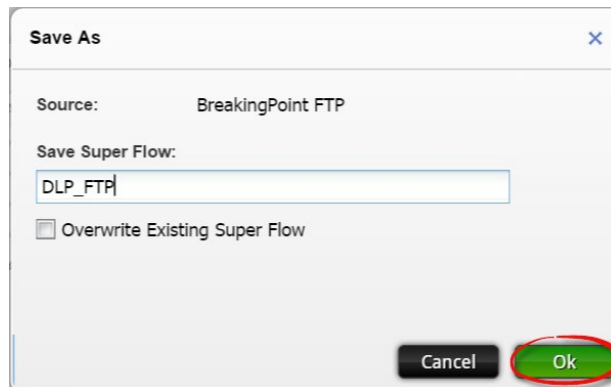
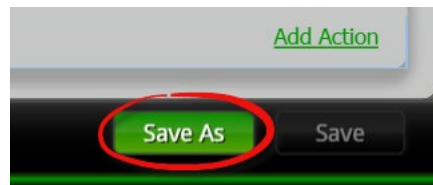
#	Protocol	Value	Client	Server
1	FTP		Client	FTP Server

The 'Actions' section lists several actions, with the 'Upload' action (row 7) highlighted. Its configuration is as follows:

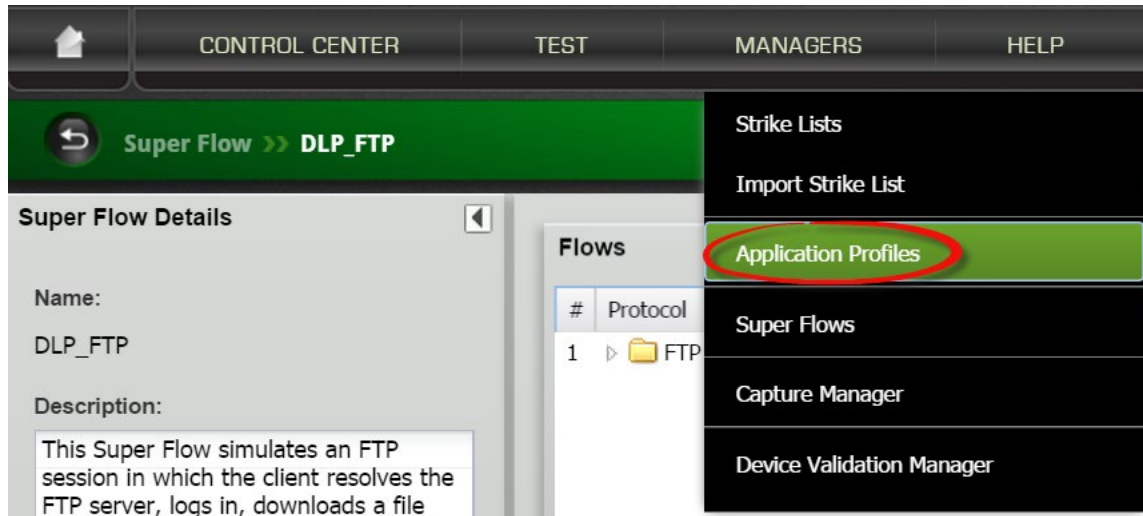
#	Action	Value	F...	Flow	Source
5	230 CWD Response		1	FTP	server
6	Download		1	FTP	client
7	Upload		1	FTP	client
	Transaction Flag	Continue			
	File min size	4096			
	File max size	4096			
	Name of the upload...				
	Request Data	secret.docx			
8	QUIT		1	FTP	client
9	221 Goodbye		1	FTP	server
10	Close		1	FTP	server

The 'Request Data' field for the 'Upload' action is circled in red, containing the text 'secret.docx'. The 'Upload' action icon is also circled in red.

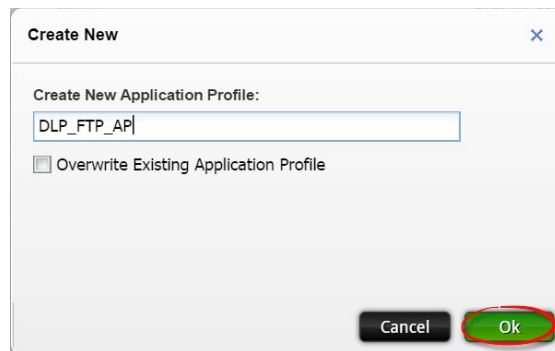
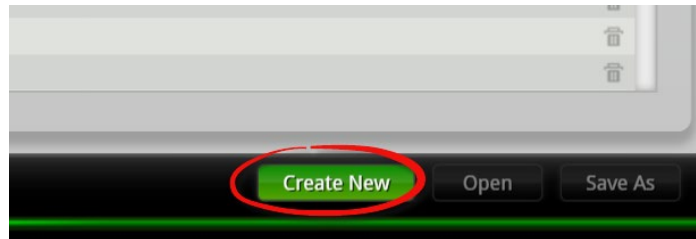
c. Click on **Save As** from the lower right corner and enter a name for the customized super flow and click **Ok**



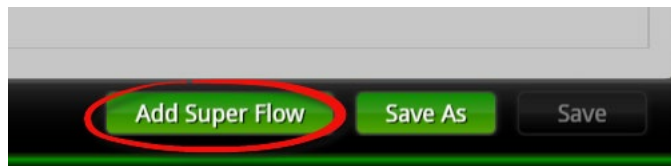
13. Create an Application Profile containing the DLP_<Protocol Name> super flow. Select **Managers** -> **Application Profiles** option from the upper menu bar.



- a. Click on **Create New** from the lower right corner and enter a name for the application profile and click **Ok**

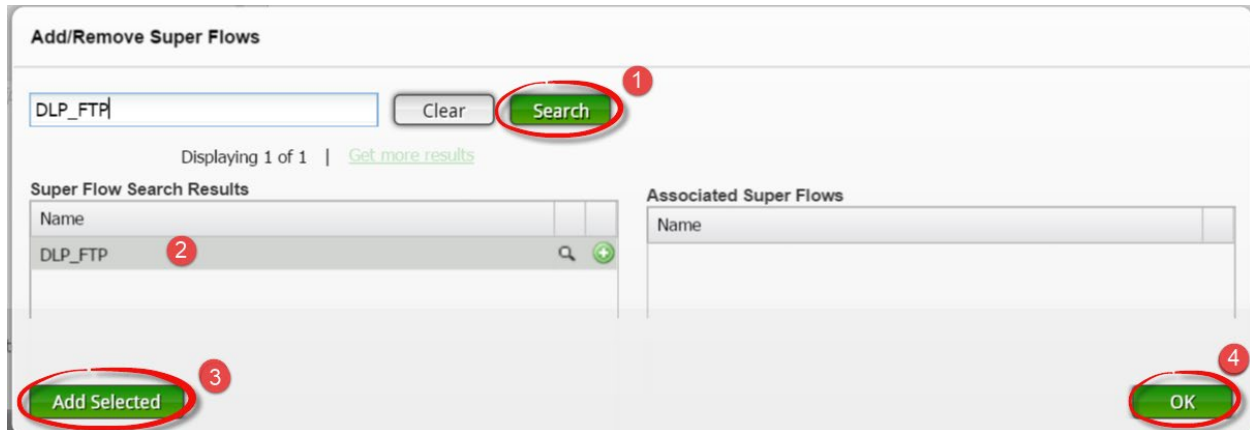


- b. Click on Add Super Flow



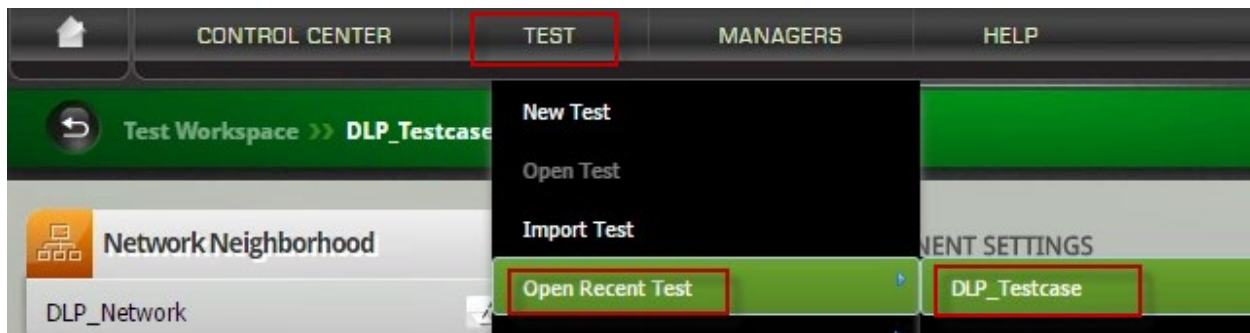
Test Methodologies for Data Leakage or Data Loss Prevention

- c. Enter DLP_<Protocol Name> in the search field, once narrowed down click on the DLP_<Protocol Name> flow in the search results. Then click on **Add Selected** and **OK**.



- d. Click on Save from the lower right corner

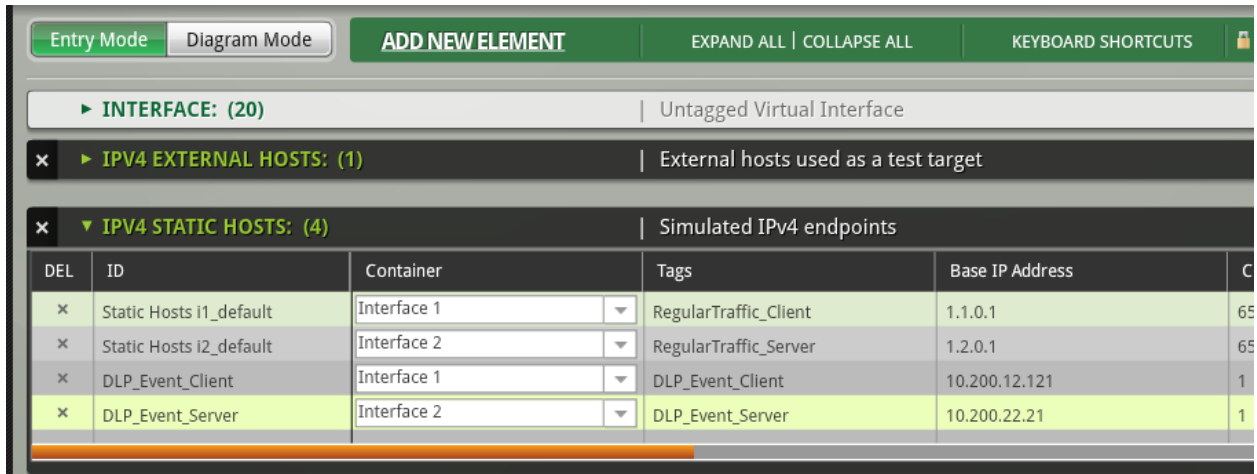
14. Select **Test** -> **Open Recent Test** option from the upper menu bar to resume configuring the test. Select *DLP_TestCase*, which is the last saved configuration file.



15. Edit the existing "Network Neighborhood" from Archive, save it as DLP_Network. Add the two new components for the DLP events, assign one to the client interface and the other to the server. The network neighborhood ID client and server should carry the background



Test Methodologies for Data Leakage or Data Loss Prevention

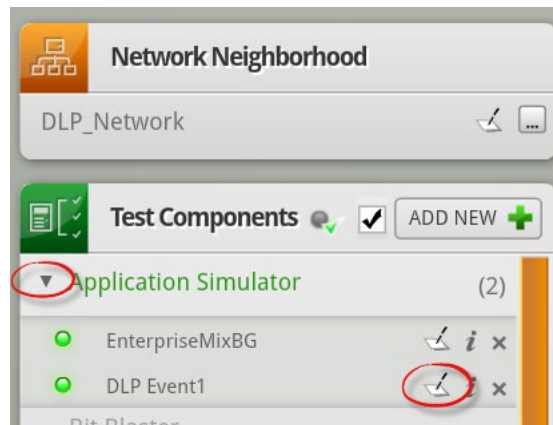
traffic and the “DLP_Event_Client” and the “DLP_Event_Server” ID should carry the DLP event traffic.



The screenshot shows a network configuration interface with a table of IPv4 static hosts. The table has columns for DEL, ID, Container, Tags, Base IP Address, and a count. The rows are:

DEL	ID	Container	Tags	Base IP Address	C
×	Static Hosts i1_default	Interface 1	RegularTraffic_Client	1.1.0.1	65
×	Static Hosts i2_default	Interface 2	RegularTraffic_Server	1.2.0.1	65
×	DLP_Event_Client	Interface 1	DLP_Event_Client	10.200.12.121	1
×	DLP_Event_Server	Interface 2	DLP_Event_Server	10.200.22.21	1

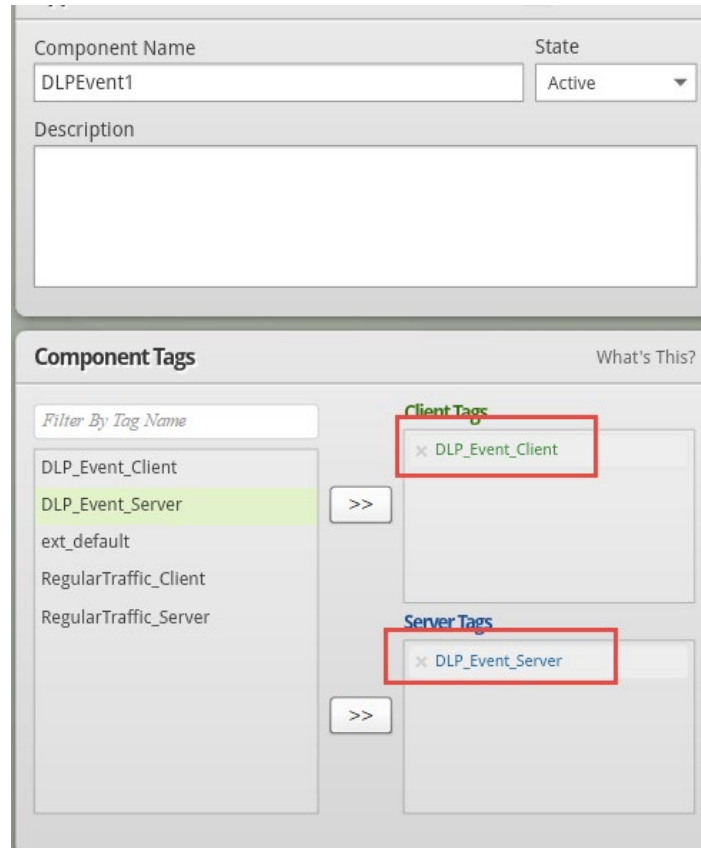
16. Resume configuration of the DLP Event1 Application Simulator component. Expand the Application Simulator components by clicking on  and start editing by clicking on .



- a. In the **Component Tags** section, make sure to assign the proper interface tags. For the Client Tags assign the tag corresponding to the Static Host element emulating the TCP client as configured in the Network Neighborhood. For the Server Tags assign the tag

Test Methodologies for Data Leakage or Data Loss Prevention

corresponding to the Static Host element emulating the TCP servers as configured in the Network Neighborhood:

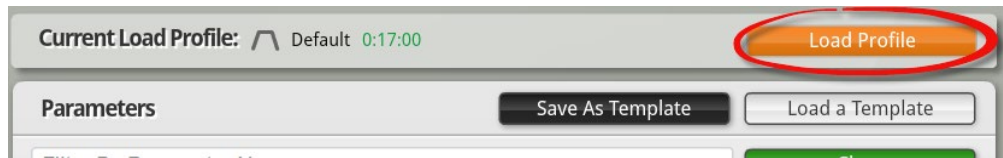


- b. The **Parameters** section contains multiple test component attributes that are grouped into category folders based on their functional purpose. For the scope of this test the following parameters will be modified:
 - i. **Minimum Data Rate:** configure the minimum data rate *10000*. This value is in Megabits/second.
 - ii. **Maximum Simultaneous Super Flows:** set the value to the maximum desired value (for this test we will set it to *1,000,000*).
 - iii. **Maximum Super Flows Per Second:** set the value to the maximum desired value (for this test we will set it to *1,000,000*).
 - iv. **Application Profile:** Browse to select the DLP_<Protocol Name>_AP profiles that was create previously to emulate the data exfiltration.

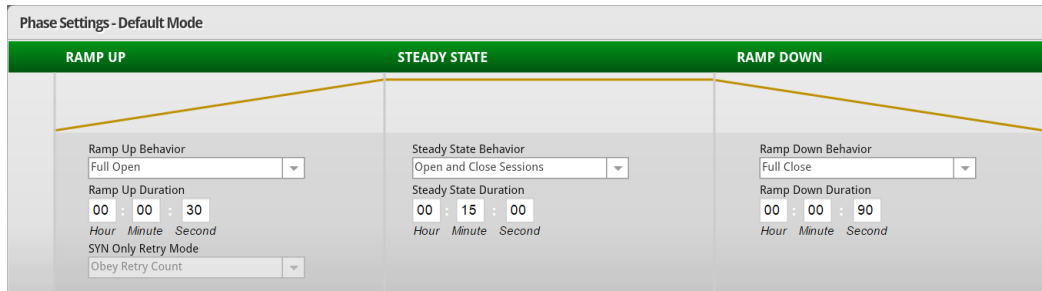
For this test, we suggest to set the “Minimum Data Rate”, Maximum Simultaneous Super Flows” and “Maximum Superflows per second” to 1.

Test Methodologies for Data Leakage or Data Loss Prevention

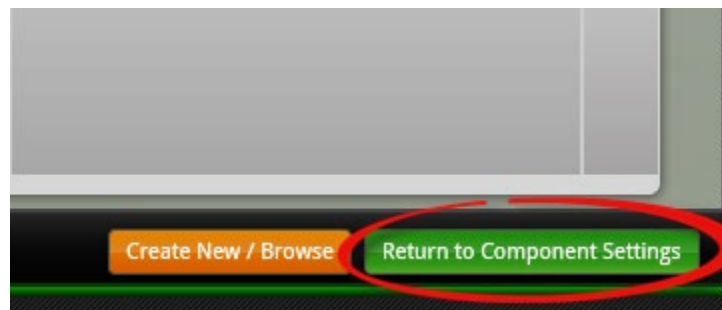
- c. Configure the test traffic pattern using the Load Profile section:
 - i. Click on the **Load Profile** button located at the upper right:



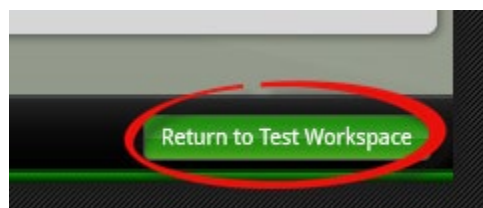
Modify the Steady State, Ramp-up and Ramp-down durations



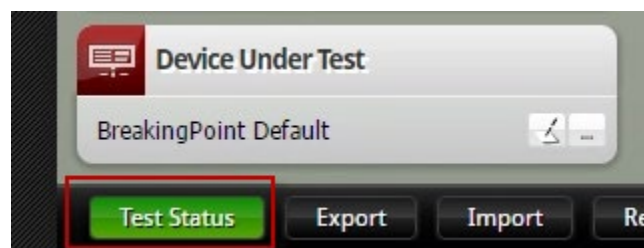
- ii. Click on the **Return to Component Setting** button to go back to the Test Component configuration screen:



- d. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:



17. Make sure the **Test Status** indicated (on the lower left corner) is now green:

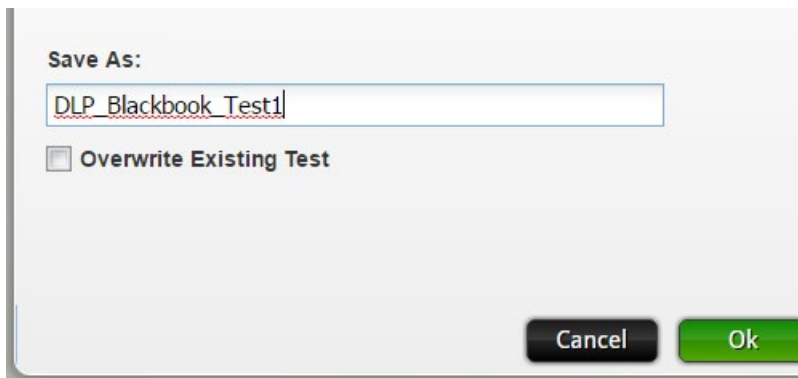


If there is not, determine what is wrong by selecting Test Status and viewing the errors.

18. Select **Save and Run** from the lower right corner:



19. If the test has not previously been saved, enter a name for the test and click **Save**:



DUT Configuration

The DUT configurations should be such that its able to detect the data loss patterns that we are sending and be able to block the same. Different devices have different types of data loss profile configurations and you should be checking the one suiting the present DUT.

Result Analysis

The main objective of this test is to validate that the DLP is happening through the DUT. To confirm the policies are effective the test should first be run without any policies configured in the DUT>

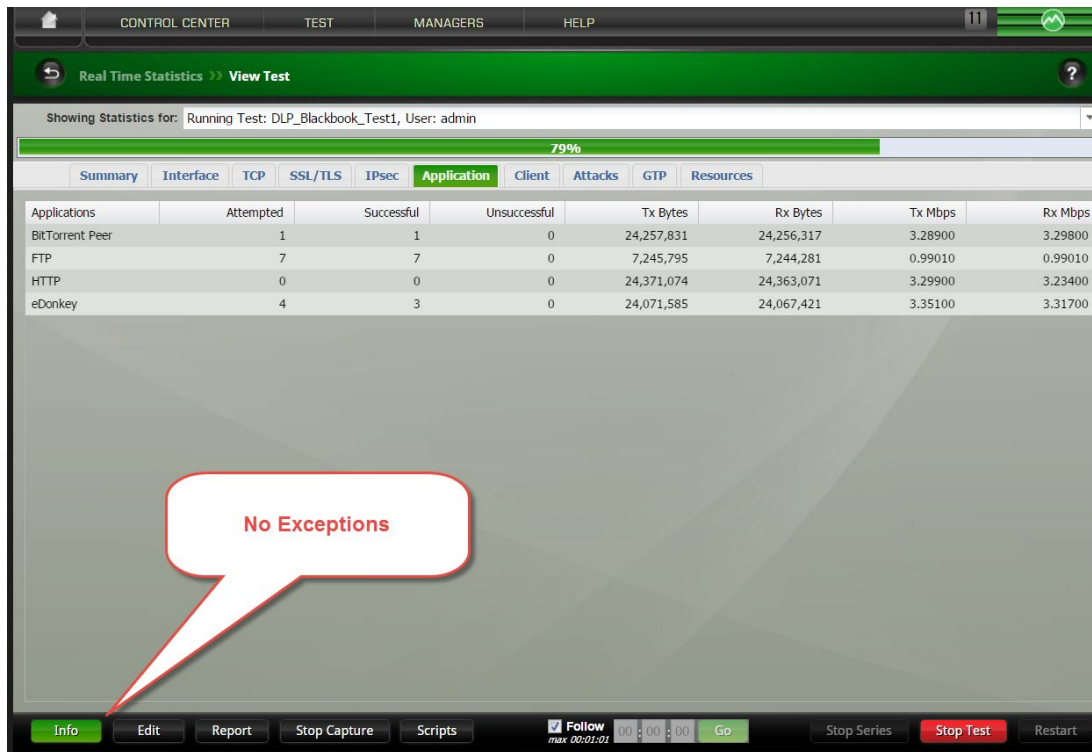
Real-time Statistics

After the test is started and initialized, the screen will automatically switch to Real Time Statistics window.

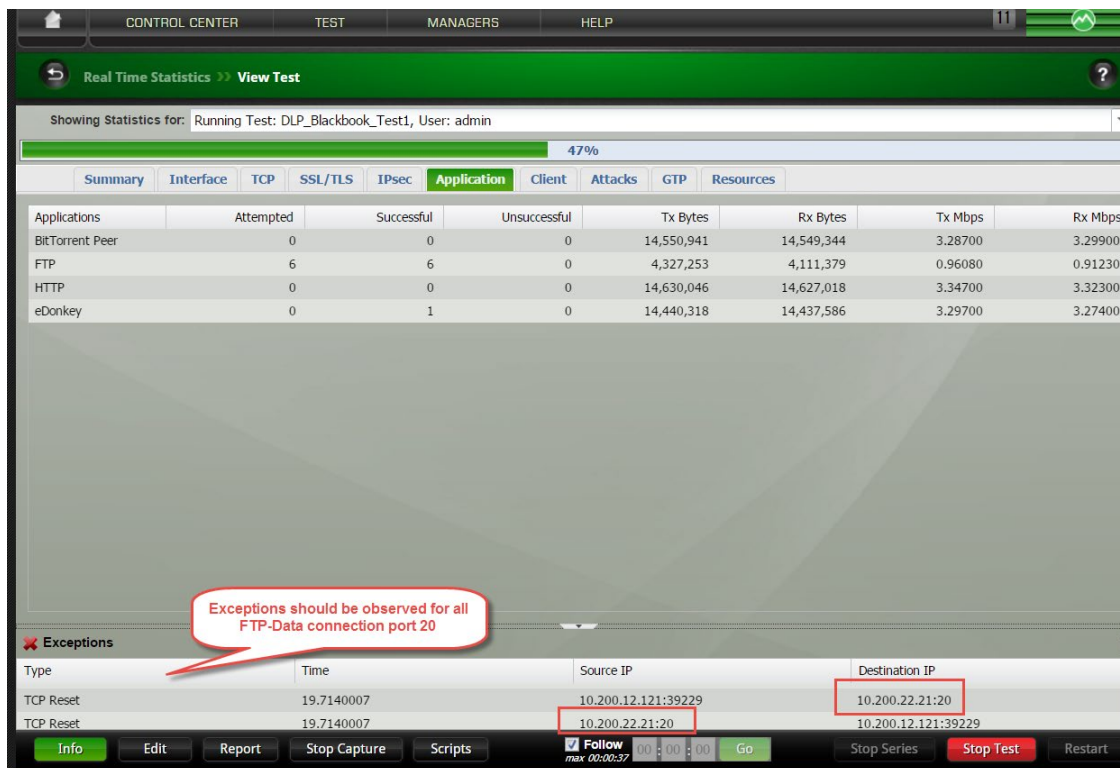
The **Summary** tab presents basic metrics that provide a good understanding of the overall test progress. Without the filtering rules in place the test should run without any issues and we should not see any exceptions. If the statistics shows errors and exceptions you can check

Test Methodologies for Data Leakage or Data Loss Prevention

connectivity issues, policy issues, etc. Exceptions at this point when the DUT is not applying filtering policies may lead to erroneous conclusions.



Now apply the data filtering rule on the DLP device that should block any documents having keywords like "Confidential", "iMotoAndroNex," etc. Rerun the test by clicking restart.



Test Methodologies for Data Leakage or Data Loss Prevention

No other connections should be blocked or we shouldn't see exception or changes in any other apps as the data loss was happening only through the data protocol.

To double confirm you can also download the traffic capture from the test run

CONTROL CENTER TEST MANAGERS HELP

Real Time Statistics >> View Test

Showing Statistics for: Running Test: DLP_Blackbook_Test1, User: admin

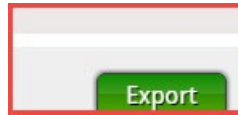
5%

Summary Interface TCP SSL/TLS IPsec Application Client Attacks GTP Resources

Applications	Attempted	Successful	Unsuccessful	Tx Bytes	Rx Bytes	Tx Mbps	Rx Mbps
BitTorrent Peer	0	0	0	1,609,638	1,608,124	3.28700	3.29900

PS40GE2NG 092690 PS10GE8NG 531278 PS10GE8NG 092630 PS40GE2NG 530000 PS40GE2NG 531293 PS10GE1NG 533571

Preferences Licenses Active Group: Group 11 Port Mapping Port Configuration Packet Export Close



Slot1Port0Hostnpl-0.1463166763455.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.flags.reset eq 1

No.	Time	Source	Destination	Protocol	Length	Info
2887	1.927588	10.200.22.21	10.200.12.121	TCP	64	20 → 17854 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
4470	2.894625	10.200.22.21	10.200.12.121	TCP	64	20 → 55914 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
6010	3.862692	10.200.22.21	10.200.12.121	TCP	64	20 → 65531 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
7543	4.833631	10.200.22.21	10.200.12.121	TCP	64	20 → 7316 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
9079	5.803623	10.200.22.21	10.200.12.121	TCP	64	20 → 55410 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
106...	6.775724	10.200.22.21	10.200.12.121	TCP	64	20 → 10259 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
121...	7.746694	10.200.22.21	10.200.12.121	TCP	64	20 → 28896 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
136...	8.717634	10.200.22.21	10.200.12.121	TCP	64	20 → 13026 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
152...	9.691654	10.200.22.21	10.200.12.121	TCP	64	20 → 31259 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
167...	10.664635	10.200.22.21	10.200.12.121	TCP	64	20 → 36964 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
183...	11.638651	10.200.22.21	10.200.12.121	TCP	64	20 → 1169 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
198...	12.612630	10.200.22.21	10.200.12.121	TCP	64	20 → 20030 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
214...	13.585622	10.200.22.21	10.200.12.121	TCP	64	20 → 50104 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
230...	14.558606	10.200.22.21	10.200.12.121	TCP	64	20 → 21959 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
245...	15.532612	10.200.22.21	10.200.12.121	TCP	64	20 → 48581 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
262...	16.510624	10.200.22.21	10.200.12.121	TCP	64	20 → 55114 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0
278...	17.487553	10.200.22.21	10.200.12.121	TCP	64	20 → 30793 [RST, ACK] Seq=1 Ack=1449 Win=5792 Len=0

Frame 15251: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)

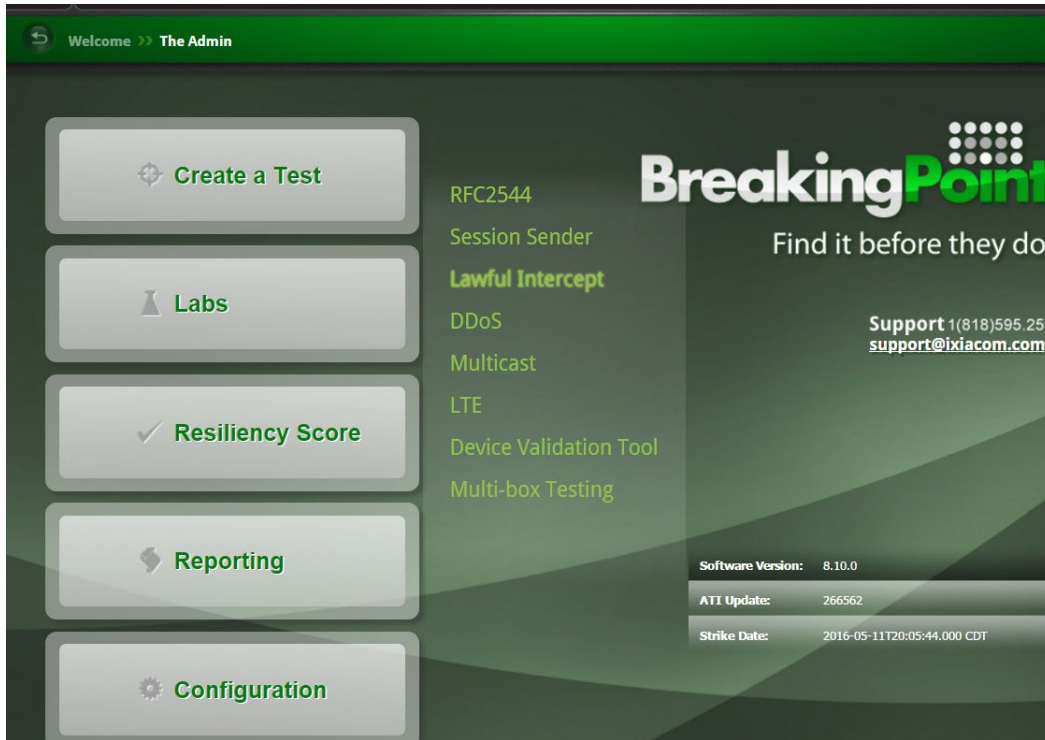
Ethernet II, Src: MS-NLB-PhysServer-26_c5:02:ff:fe (02:1a:c5:02:ff:fe), Dst: MS-NLB-PhysServer-26_c5:01:00:00 (02:1a:c5:01:00:00)

Internet Protocol Version 4, Src: 10.200.22.21, Dst: 10.200.12.121

Transmission Control Protocol, Src Port: 20 (20), Dst Port: 31259 (31259), Seq: 1, Ack: 1449, Len: 0

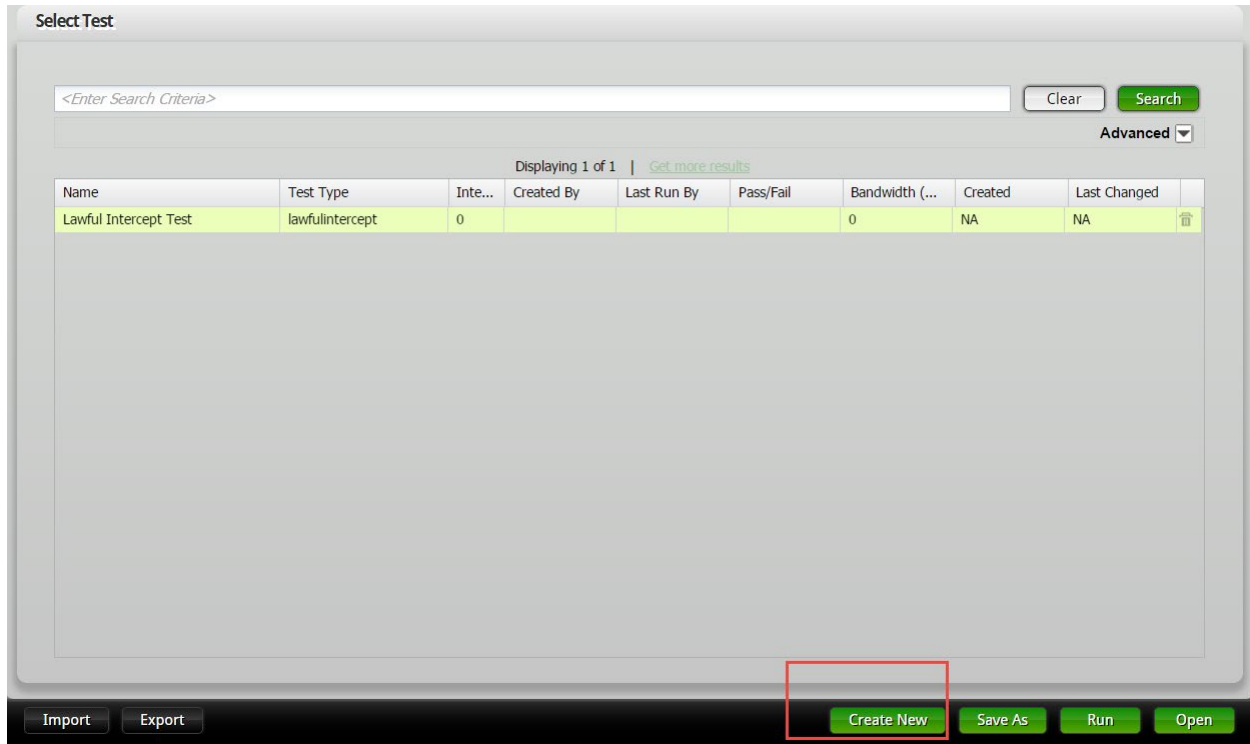
Data Leakage Test: Lawful Intercept Labs

1. The lawful intercept lab is a readymade lab in BreakingPoint that can help in running quick data leakage test. To open the lab, you need to go the main screen and click on Labs -> Lawful Intercept



Test Methodologies for Data Leakage or Data Loss Prevention

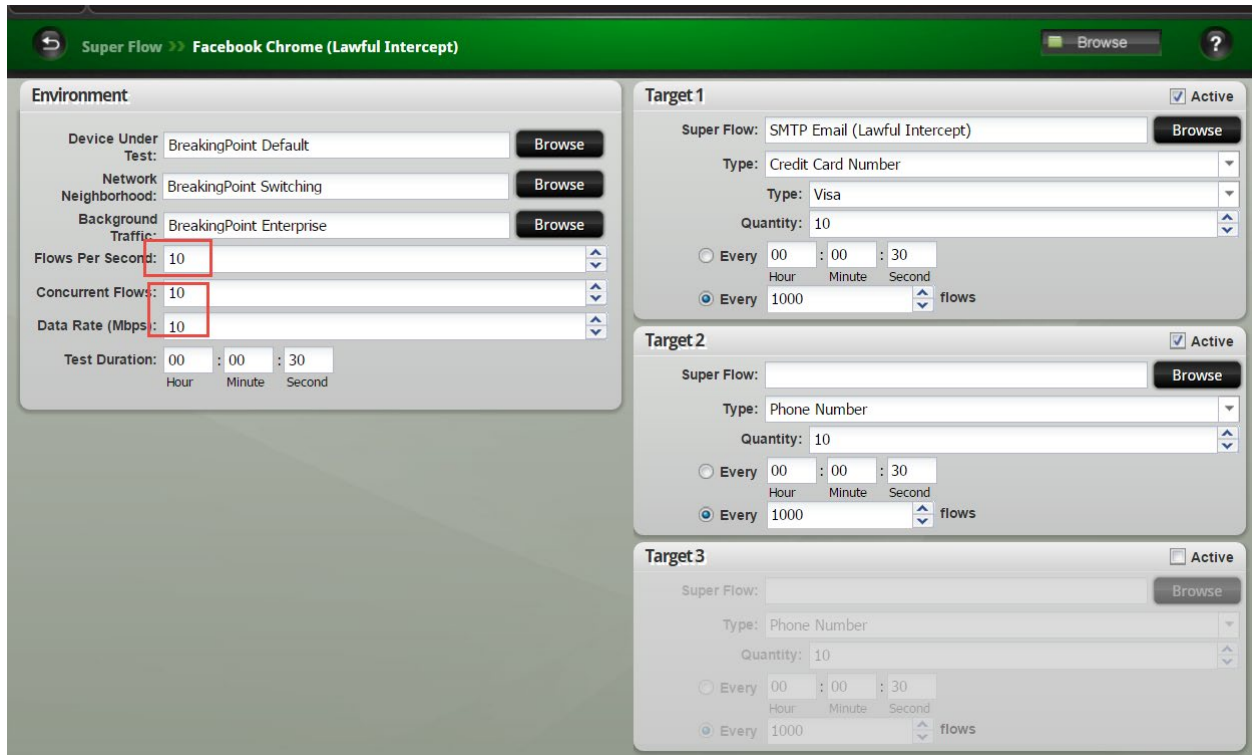
- This will provide the option to create a new lab or pre-select an existing one. For this case, we will go for a new lab to create something from the scratch.



- A new lab would have most of the default settings to run it as it is. However, for this particular lab we would try to change some of the parameters. At this point it should also be noted that if we plan to change the "Network Neighborhood," it needs to have "Client_Lab" and "Server_lab" components that would let the Lab understand the DLP direction and the IP address source and destinations. In this example, we are using the default switching workflow as our DLP is set in bridged mode.

Test Methodologies for Data Leakage or Data Loss Prevention

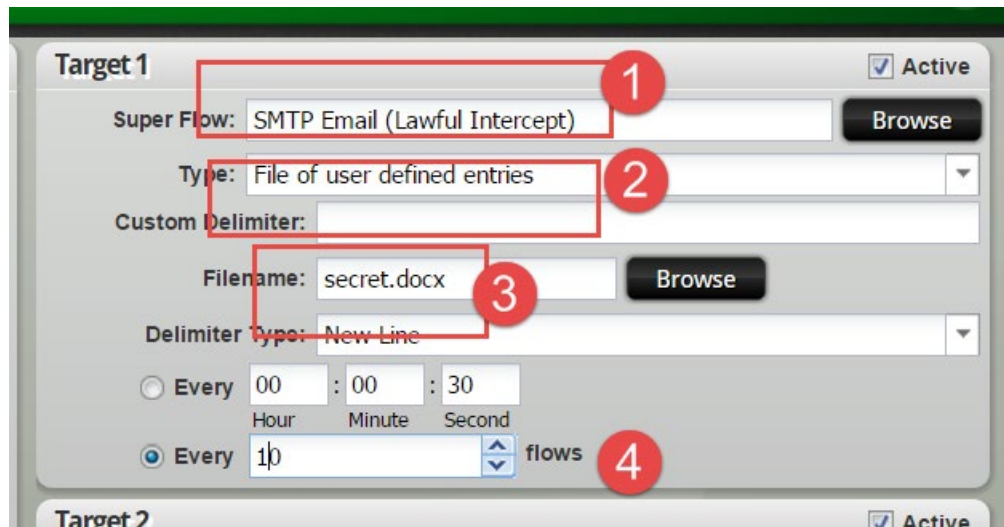
- Since this is not a performance test, we are reducing the flows per second, the concurrent flow and the Data Rate to 10.



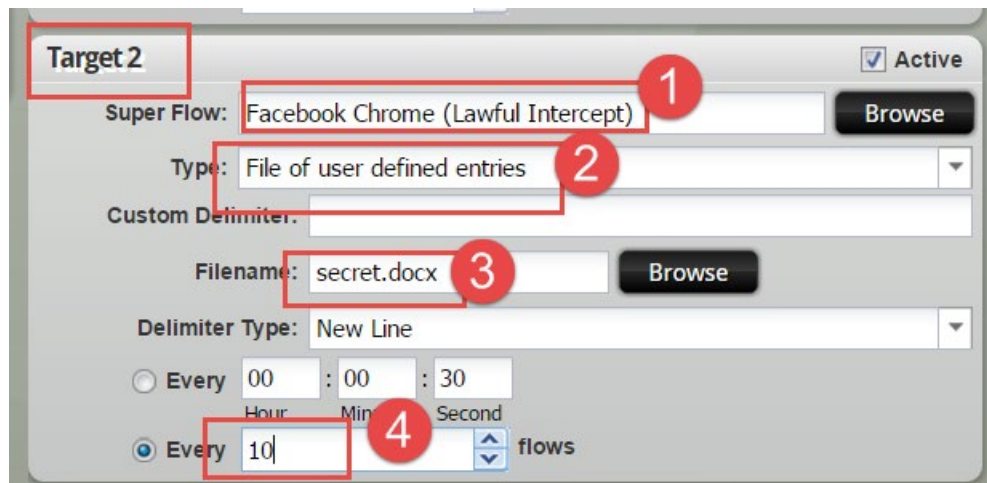
- In the lab, we will be using two different apps to send the interesting data out of the network. One of them is SMTP and the other would be through Facebook. To keep consistency, we will be using the same secret documents content that we had used in the previous DLP test.
 - Select a Superflow with a needle (you can click on the browse) to get the options of the available Superflows.
 - For the filename select the same file "Secret.docx" (If this file is not present in BreakingPoint which can happen if you have not run the previous test) than upload the file.

Test Methodologies for Data Leakage or Data Loss Prevention

- c. Then we set the frequency of the transfer of the keywords and it will be in every 10 flows.



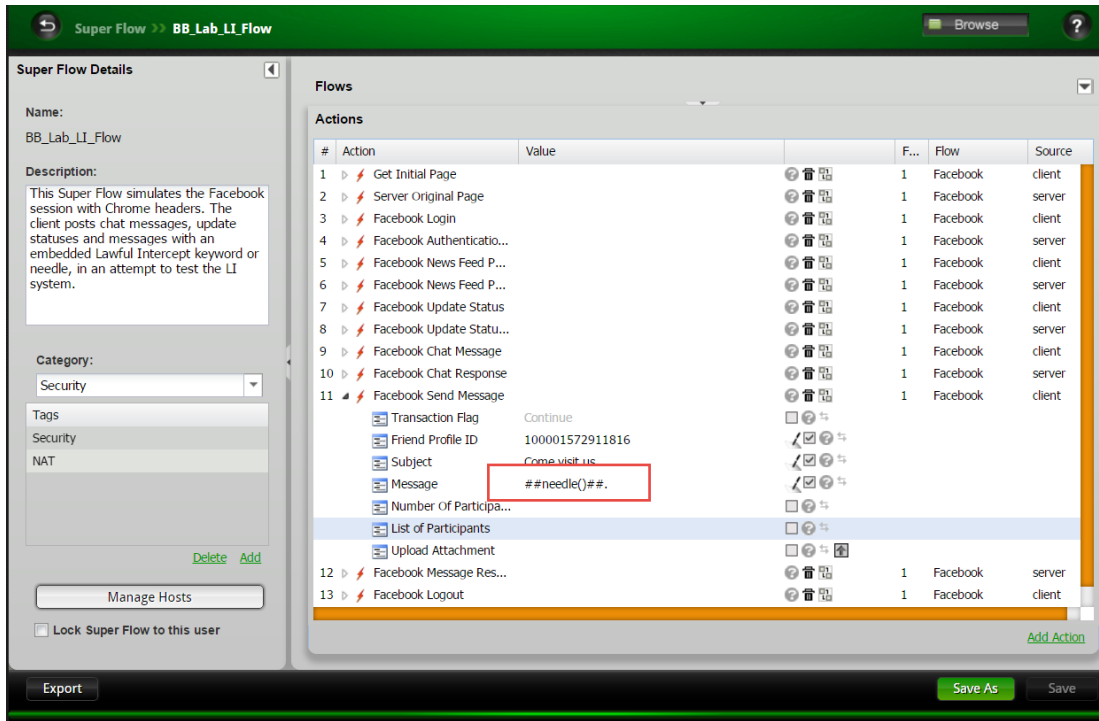
6. For the next flow, you can select the Facebook flow and have the same secret document in it. This will ensure the Facebook flow will also send the same keywords/secrets.



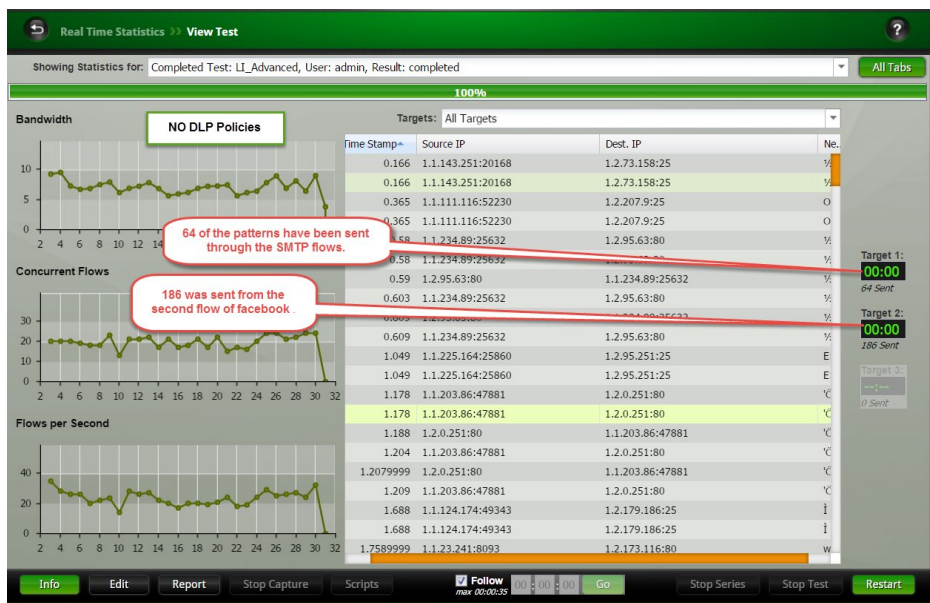
[Note: You can create your own flows as well. The Browse for super flow can show any flows that has “##needles()##” in it. Below is the example of a Superflow having needle in it. For example, you can open any of the Superflows and add the “##needle()##” in the appropriate field

Test Methodologies for Data Leakage or Data Loss Prevention

and the LI lab will ensure that the needle is replaced with the appropriate keywords during test run].



7. Now once the Facebook flow has been setup now you have two flows SMTP and Facebook, which are both trying to send the sensitive data out of the network. Clicking the save and run will now start sending the traffic.
8. At first there should not be no DLP set up as its important to see that the traffics are not blocked for some other reasons (wrong network setup, firewall policies, etc.). A test run should show both target 1 and Target2 being able to send some of the sensitive data.

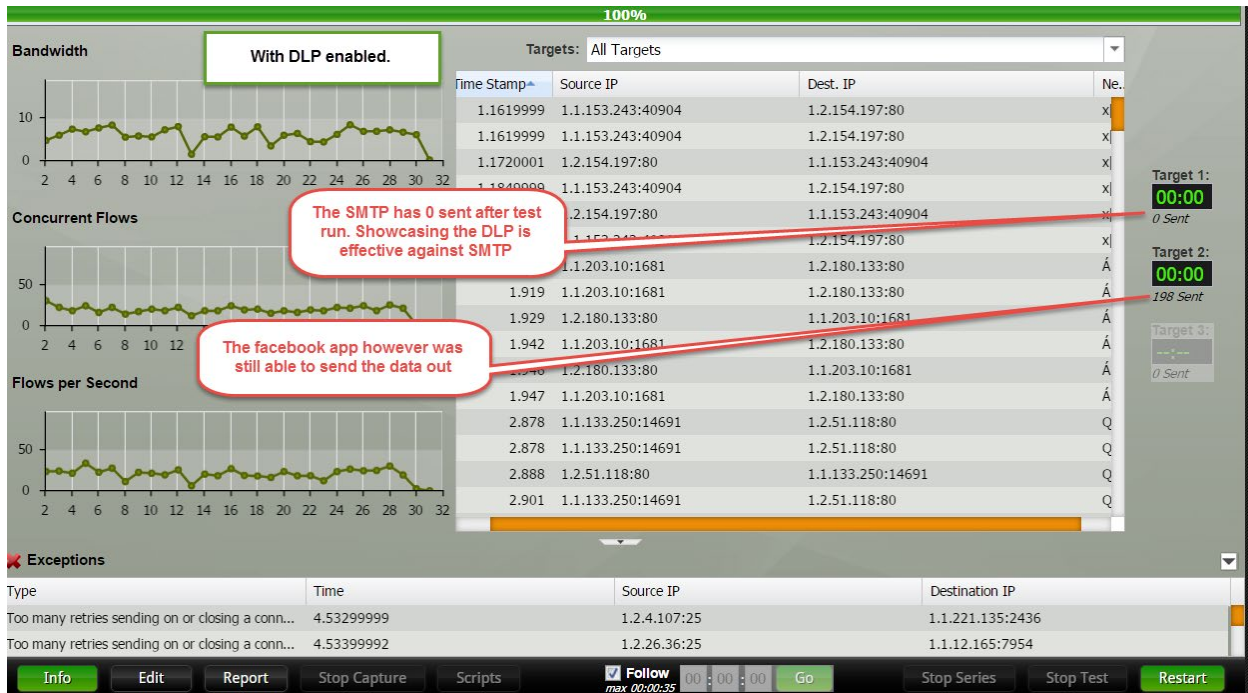


Test Methodologies for Data Leakage or Data Loss Prevention

9. Now we should re-run the test with the DLP policies enabled. In the case of my DUT we find something interesting. The data leakage happening through SMTP has been mitigated effectively however the same data leakage that was happening through Facebook wasn't mitigated, thereby exposing the loophole in the DLP device.

This brings us to the interesting task of understanding the true efficiency of the DLPs and there are different ways/variants of tests that can be tried like:

- Different Applications
- Encrypting the application.
- With VPN
- Changing the type of attachment, pdf, exe, encrypted doc, etc.
- Converting the doc into picture.



Test Variables

BreakingPoint offers the following test configuration parameters, which provide the flexibility to simulate a high number of different tests with various DLP traffic and below are few of the examples.

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
IP Version	IPv4	IPv6, IPsec, DSLite, 6rd, selected Mobility stacks, etc.
Benign Traffic	Pre-Canned Mix	Custom application mix that matches the traffic profile from the end customer deployment production network that needs to be validated.
Application Type	Facebook, SMTP, HTTP(S), Gmail, etc.	BreakingPoint has more than 300 apps where <code>##needle()</code> can be used and checked if the DLP device is capable to prevent data leakage through that particular app.
Encryption	None	SSL, IPsec
File Types	Docx	Pdf, exe, swf, and any other attachments the application can take.

Conclusions

This test methodology demonstrates how to configure BreakingPoint to determine the Data Loss Prevention (DLP) efficiency of a device. Two tests have been run as part of this exercise, the first one uses the manual configuration of putting a file in a Superflow and then understanding the efficiency of the DLP to understand and restrict data leakage. The second one uses the Lawful Intercept lab and the “needle” in a haystack format to insert interesting data into multiple applications and test the ability of a DLP engine in blocking advanced data leakage scenarios.

Test Methodologies for Virtual Environment Security

Service providers are rethinking how to monetize new services and deliver satisfying end-customer experiences by leveraging the economy of scale and elasticity of network virtualization infrastructures. Three key roles in IT that are setting the priorities. The network architect goal focuses on keeping performance up and costs down. The QA/Test Engineer needs to make sure the network can withstand heavy workloads as well as remain resilient against cyber threats. The Operations Director focuses on the overall system integrity to deliver uncompromising performance with tight security while maintaining agility in the development cycle. Virtualization powers these initiatives. However, everything old is new again, everything proven is unproven, so the need for virtual environment application and security test is more important than ever.

Test Case: Lateral Threat Propagation within Software Defined Data Center

Overview

Traditional and next-generation workloads are transitioning to the cloud at an increasing pace. In addition to the economy of scale that is afforded by the virtual infrastructure, there are traditional security concerns associated with identified vulnerabilities in these workloads and the challenges required to secure cloud environments. Software defined networks (SDNs) take important steps towards isolating tenant workloads from other tenants through network segmentation and applying access policies between virtual network functions (VNF) and their associated virtual machines (VM).

Lateral threat propagation is a targeted attack for achieving lateral movement within software defined data centers (SDDC) that takes advantage of an exploit within one application layer to derive access to another application layer. An example would be the case where a web server is compromised and used to inject malicious code to gain root access to an application server. Next, the root access is used to access customer data from a connected database server.

Objective

Ixia's BreakingPoint VE (Virtual Edition) enables the emulation of advanced lateral threat propagation.

Setup

This setup requires a minimum of two virtual test ports to emulate server-to-server (S2S) communication between emulated virtual machines hosted in one hypervisor server and a second set emulated virtual machines hosted in another hypervisor server elsewhere in the software defined datacenter.

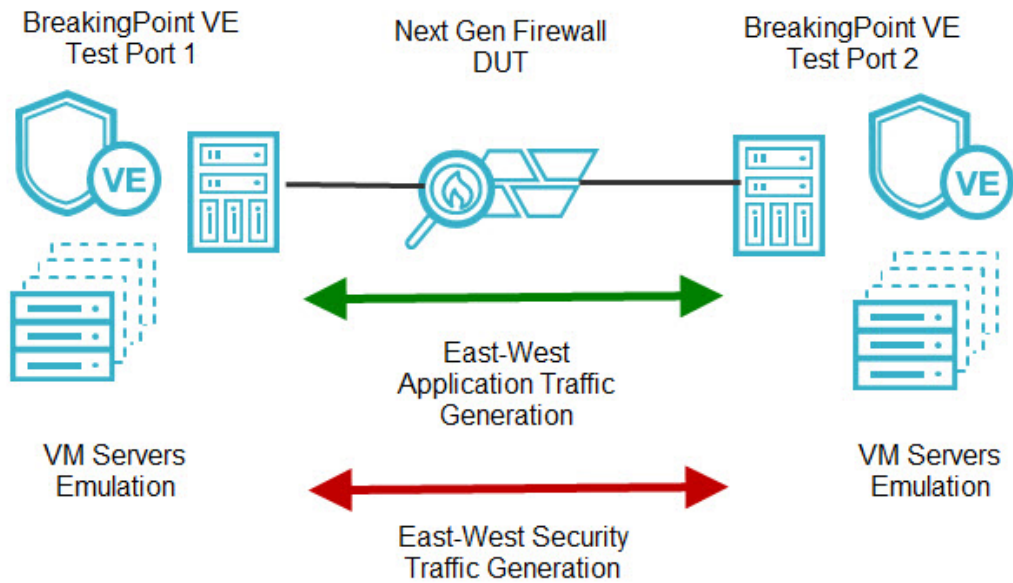
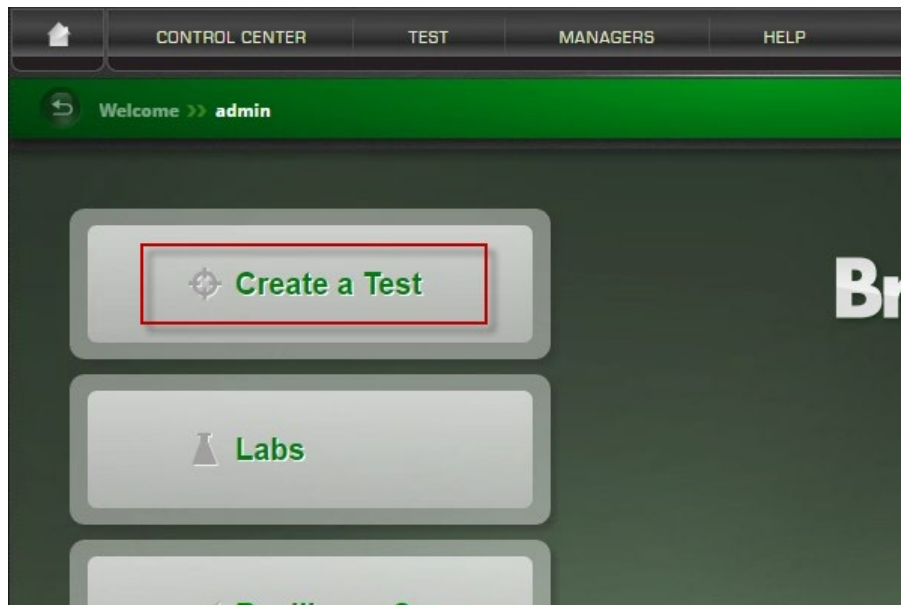


Figure 37. Lateral Threat Propagation Test Topology

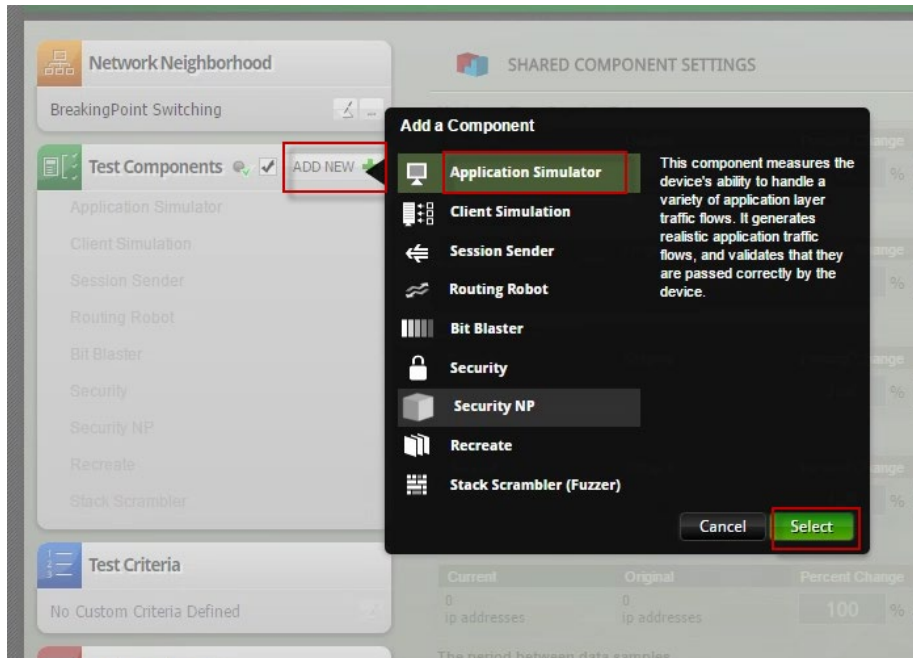
Step-by-Step Instructions

1. Click on the **Create a Test** button on the main BreakingPoint session page.

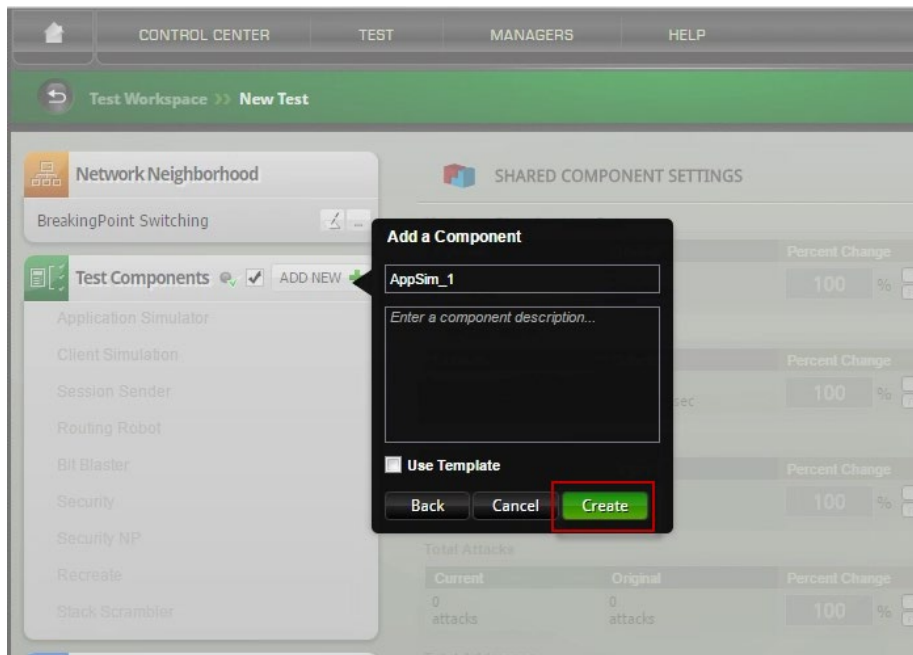


Test Methodologies for Virtual Environment Security

2. Click Add New button next to Test Components. Select Application Simulator and then click the Select button.

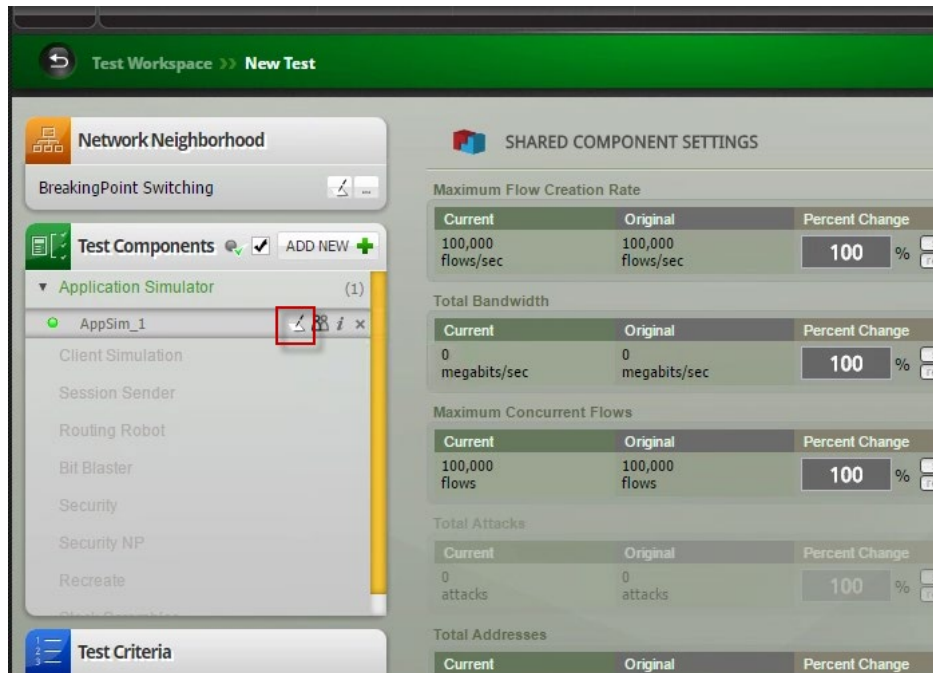


3. Name the component and then click Create button.

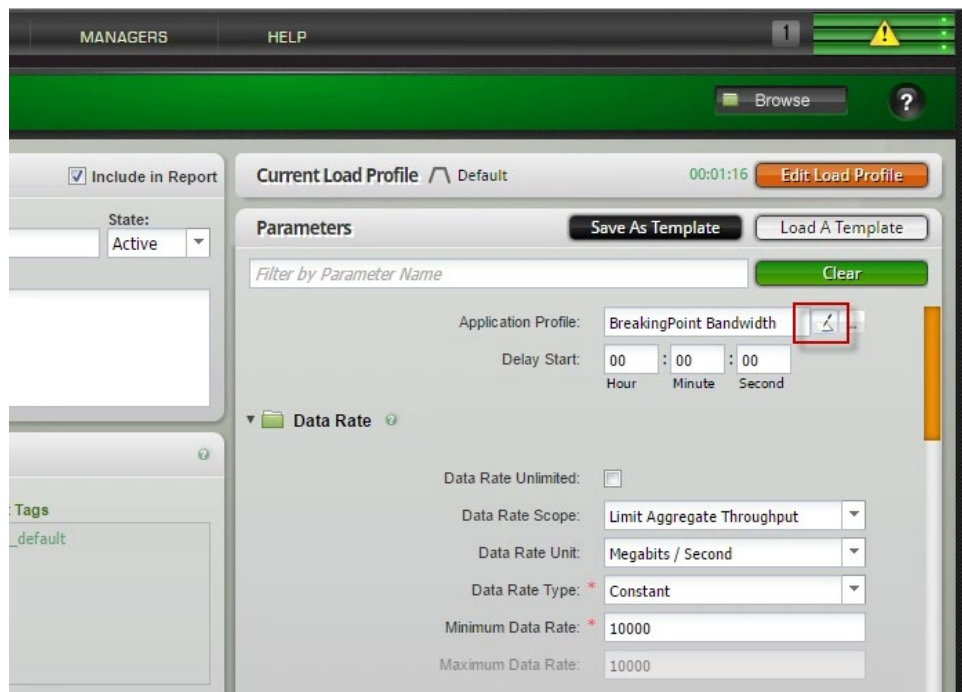


Test Methodologies for Virtual Environment Security

4. Click the pencil icon to Edit the AppSim component.

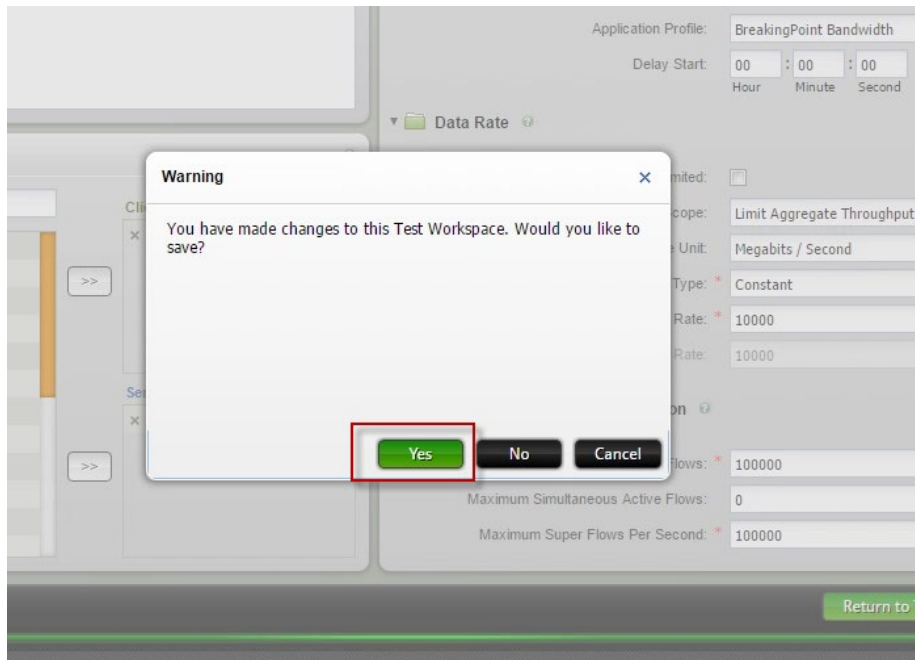


5. Click the pencil icon to **Edit** the Application Profile.

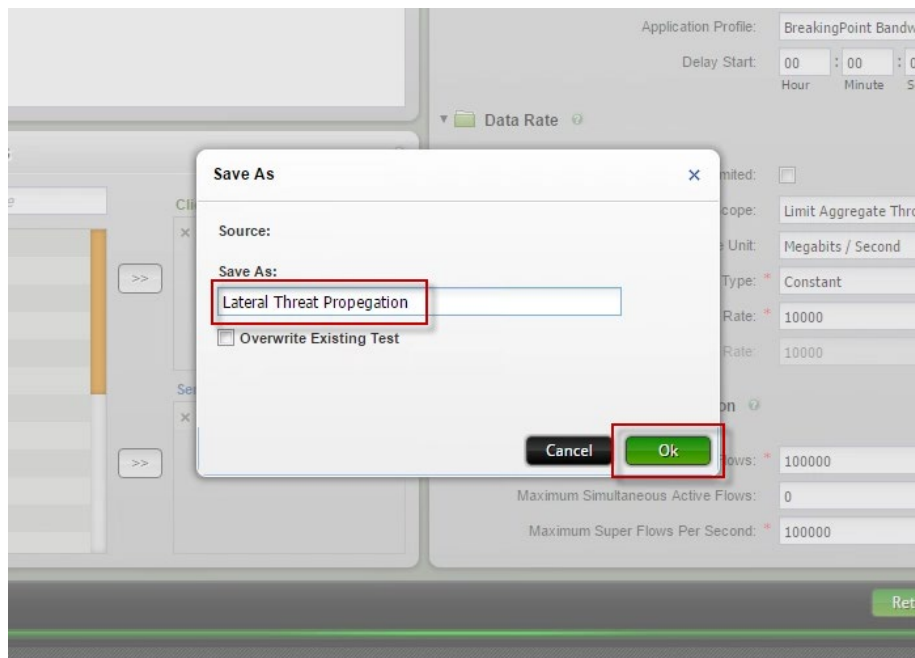


Test Methodologies for Virtual Environment Security

6. Click the **Yes** button to save changes made to the Test Workspace.

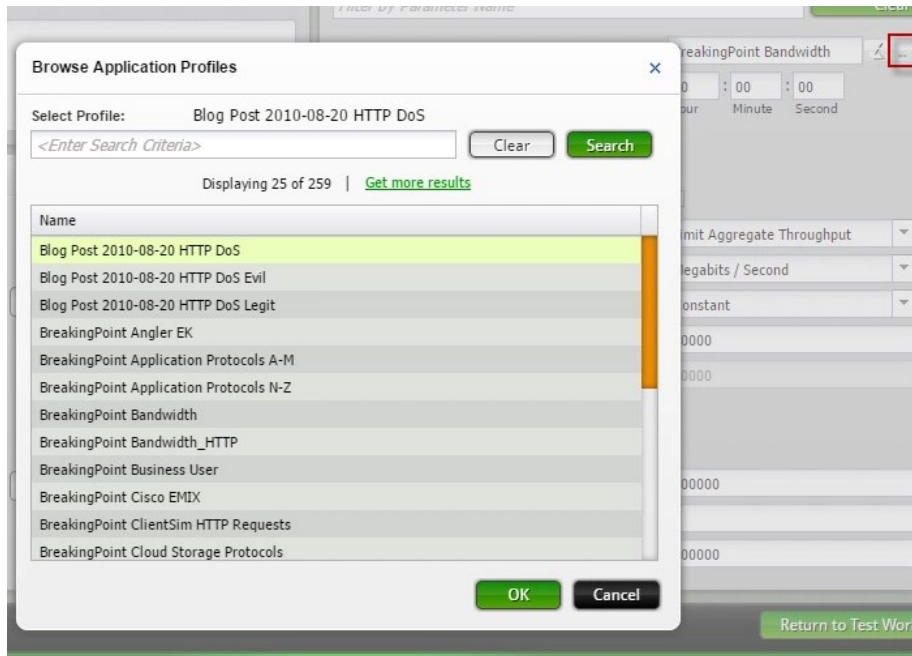


7. Name the test and click the **OK** button to save.

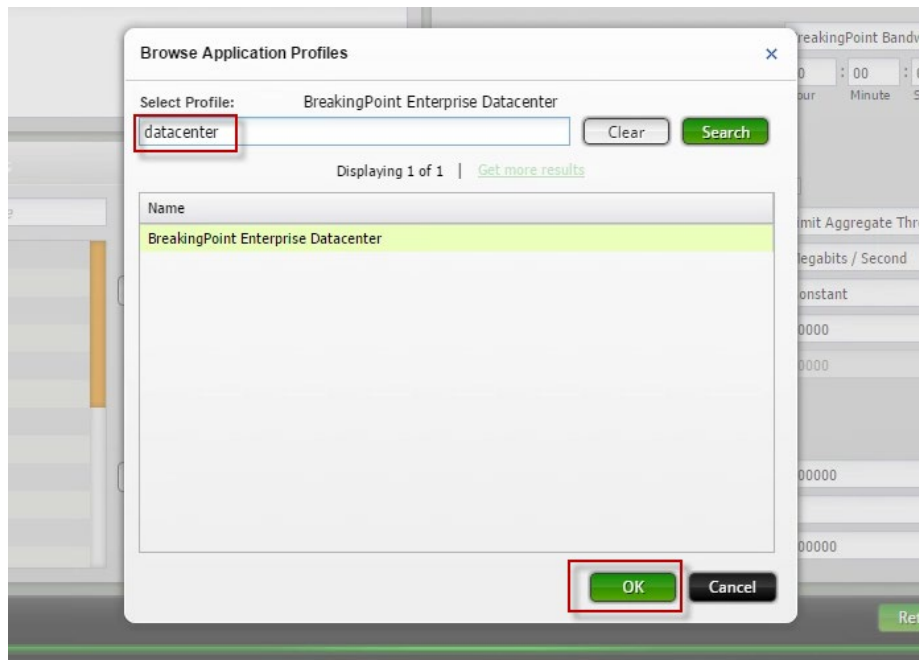


Test Methodologies for Virtual Environment Security

- Alternatively select the “...” button to select a different Application Profile.

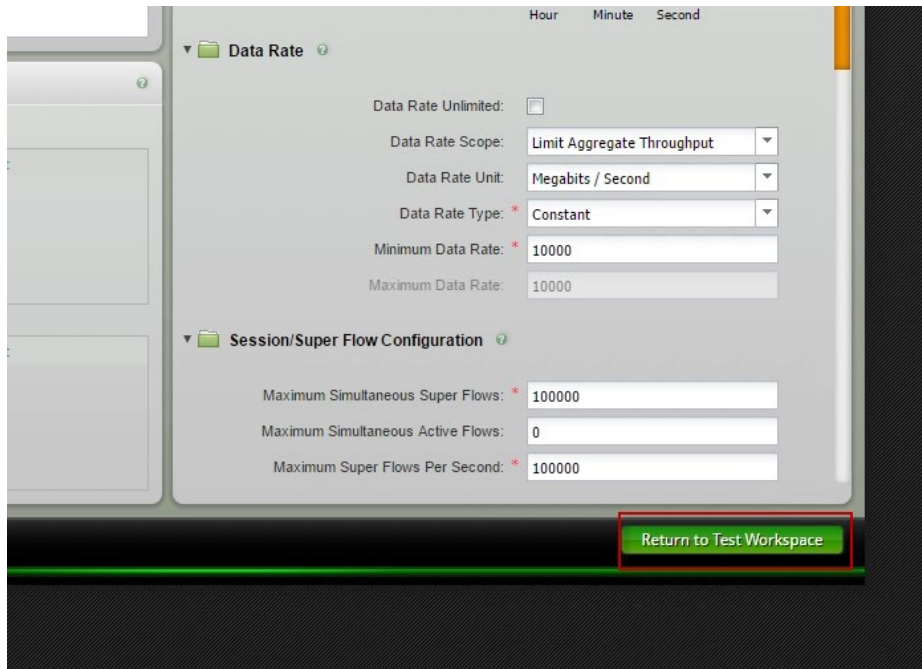


- Type in the name of the Application Profile you want to use and then click the OK button to select.

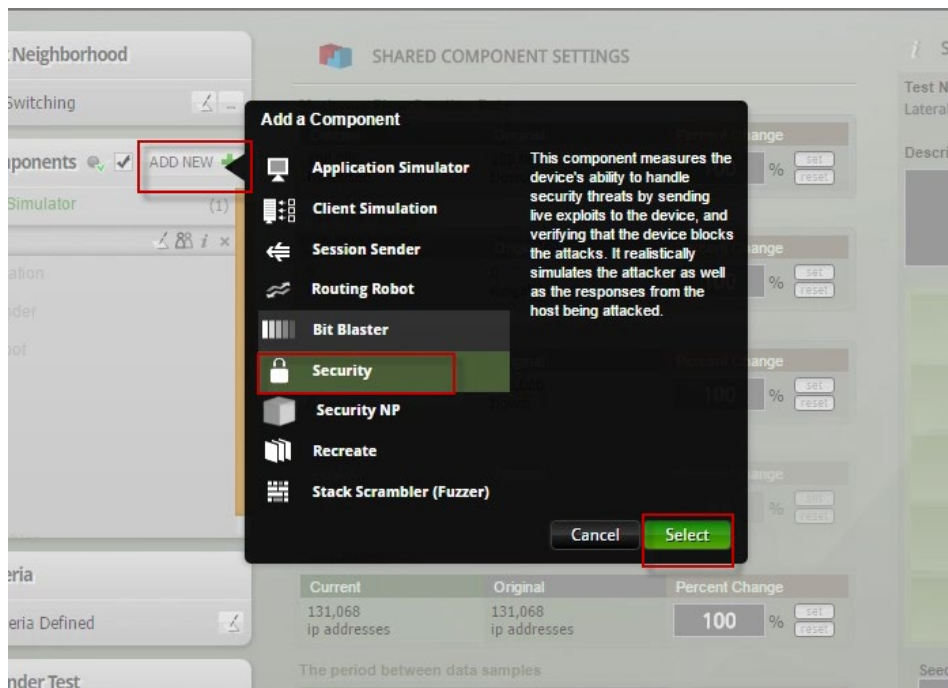


Test Methodologies for Virtual Environment Security

10. Click the **Return to Test Workspace** button when finished with the Application Profile.

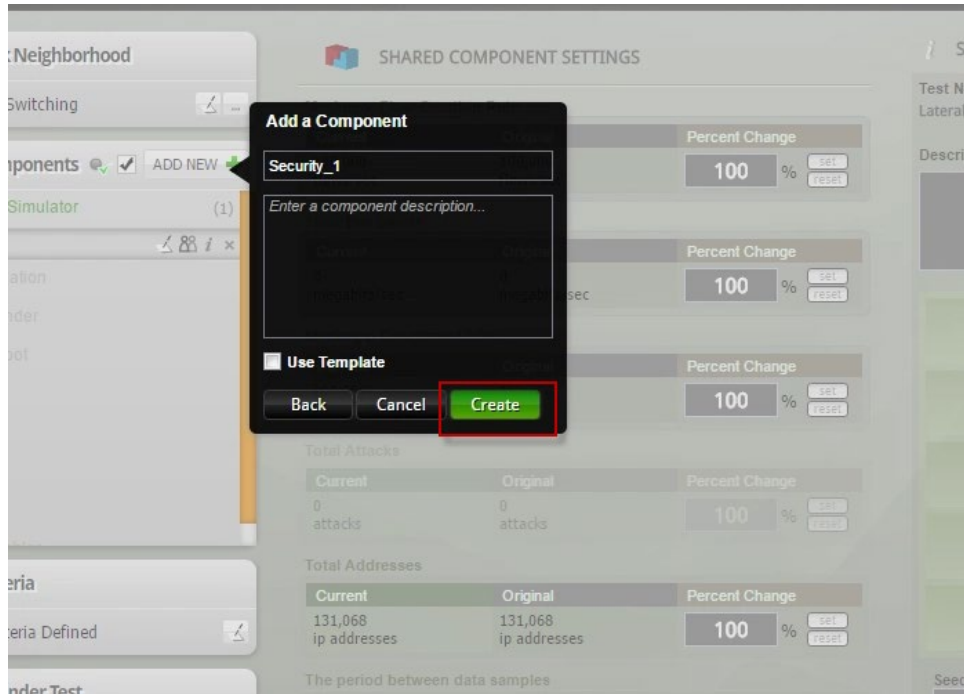


11. Click the **Add New** button next to the **Test Components** and select **Security** option.

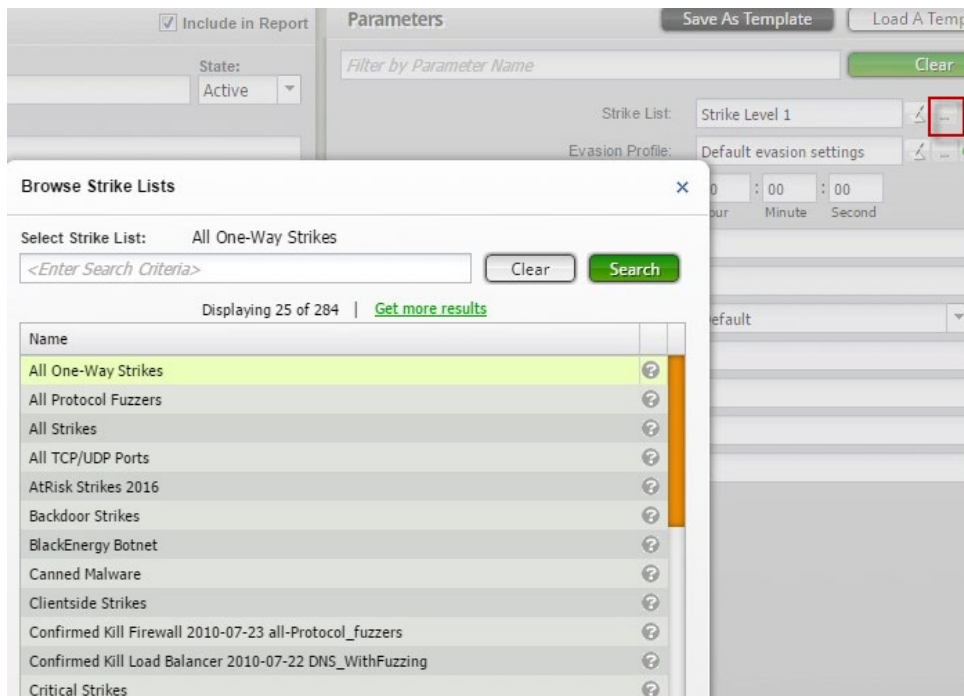


Test Methodologies for Virtual Environment Security

12. Name the security component and click the **Create** button.

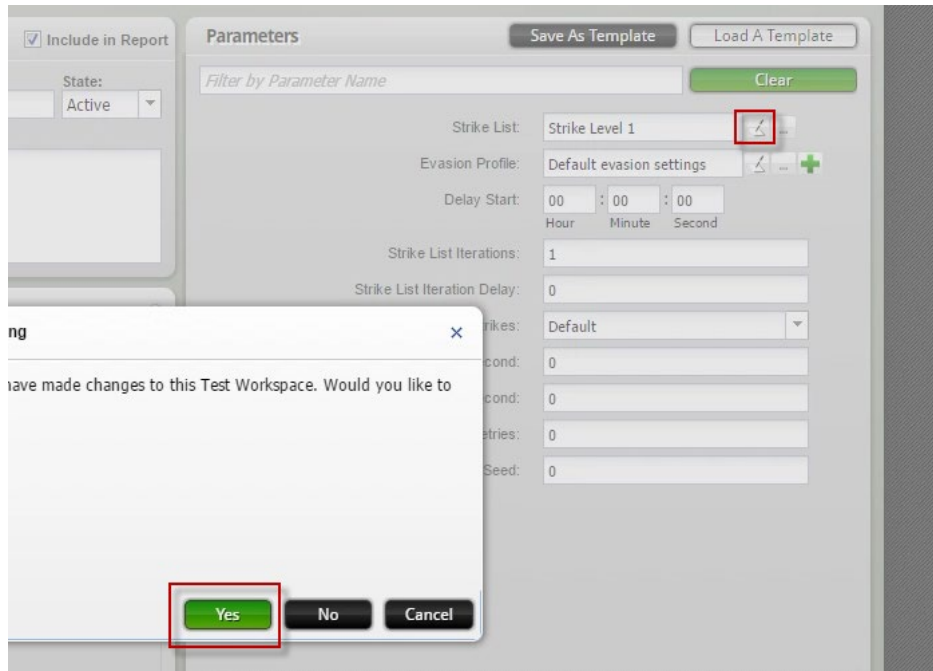


13. Click the “...” button next to the **Strike List** parameter to browse for a Strike List template.



Test Methodologies for Virtual Environment Security

14. Click the pencil icon to **Edit** the Strike List parameter. Click the **Yes** button to save changes to the Test Workspace.

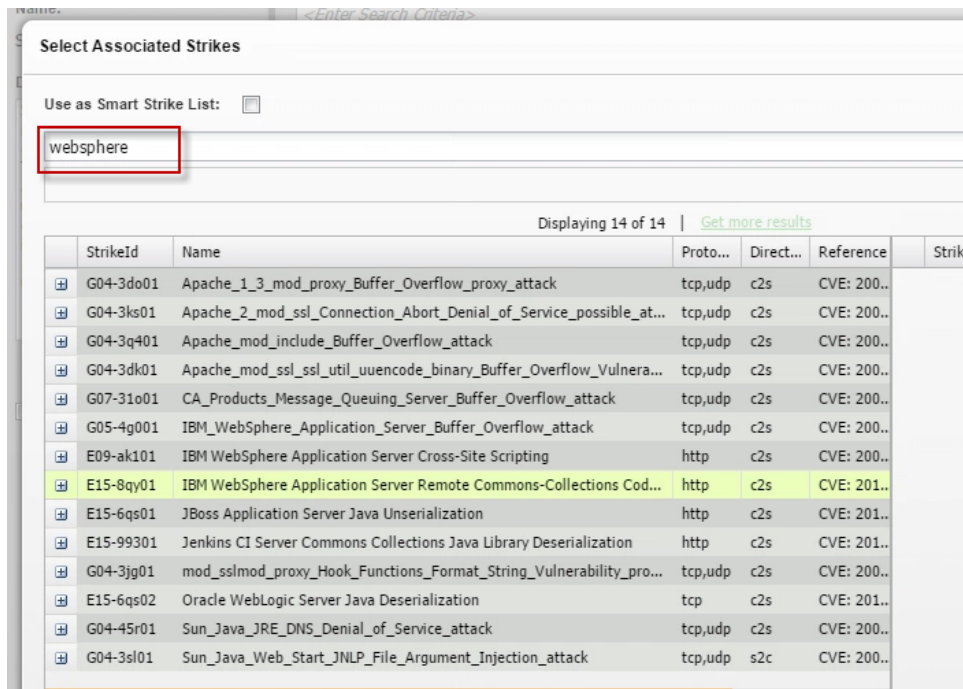


15. Click the Add **Strike** button.

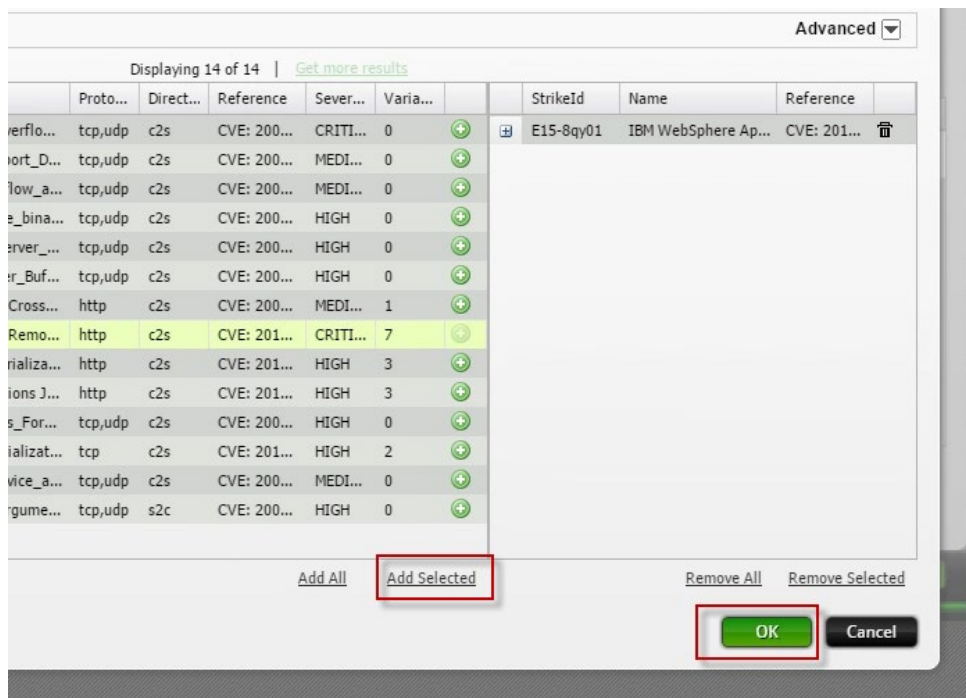


Test Methodologies for Virtual Environment Security

16. Enter the search text of the Strike pattern desired and click the **Search** button.

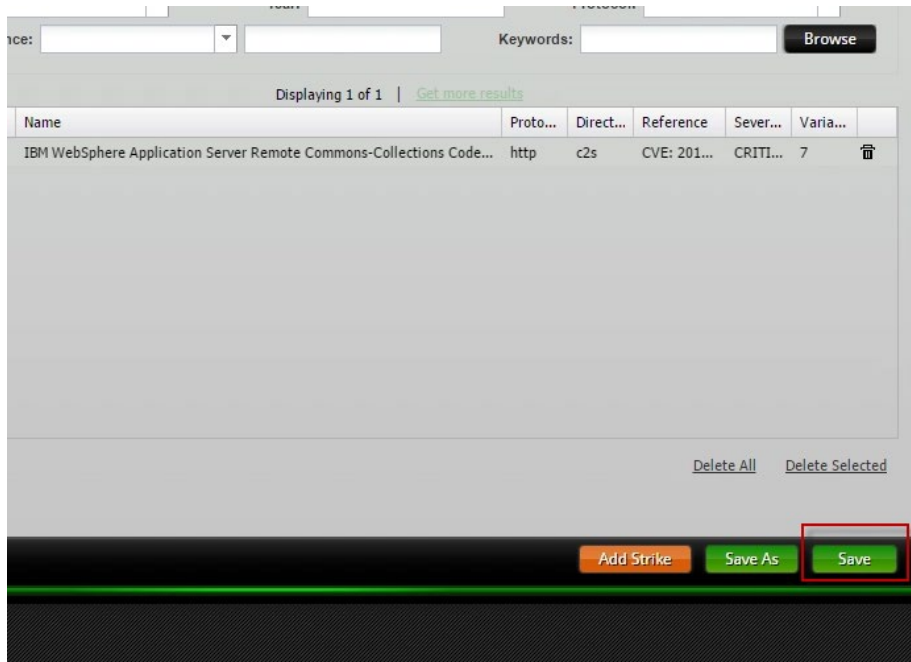


17. Select the desired strike pattern from the search results and click the **Add Selected** link to add the strike to append. Click the **OK** button to finish.

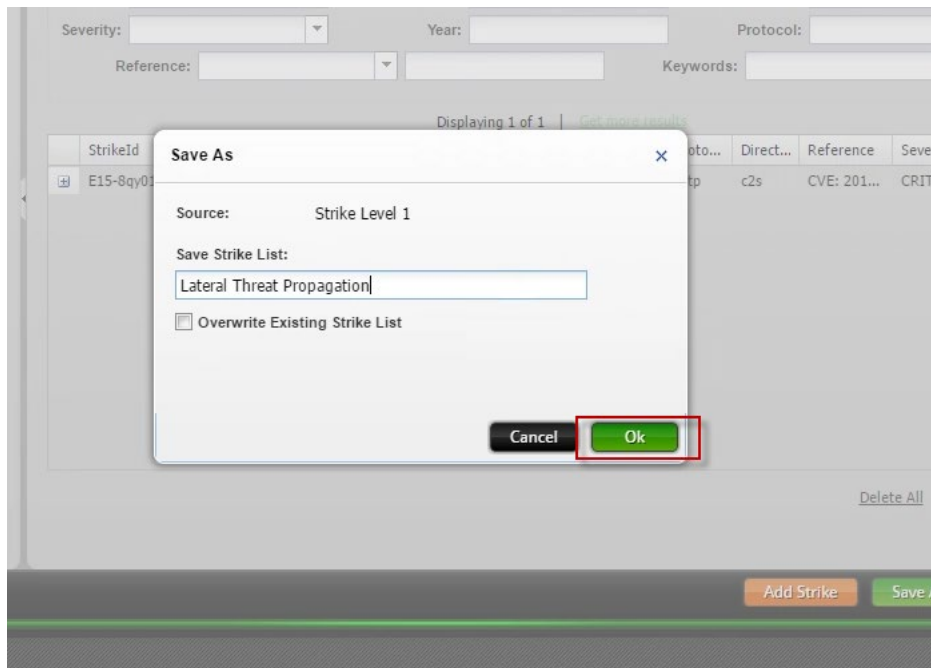


Test Methodologies for Virtual Environment Security

18. Click the **Save** button when finished adding Strikes to the Strike List.

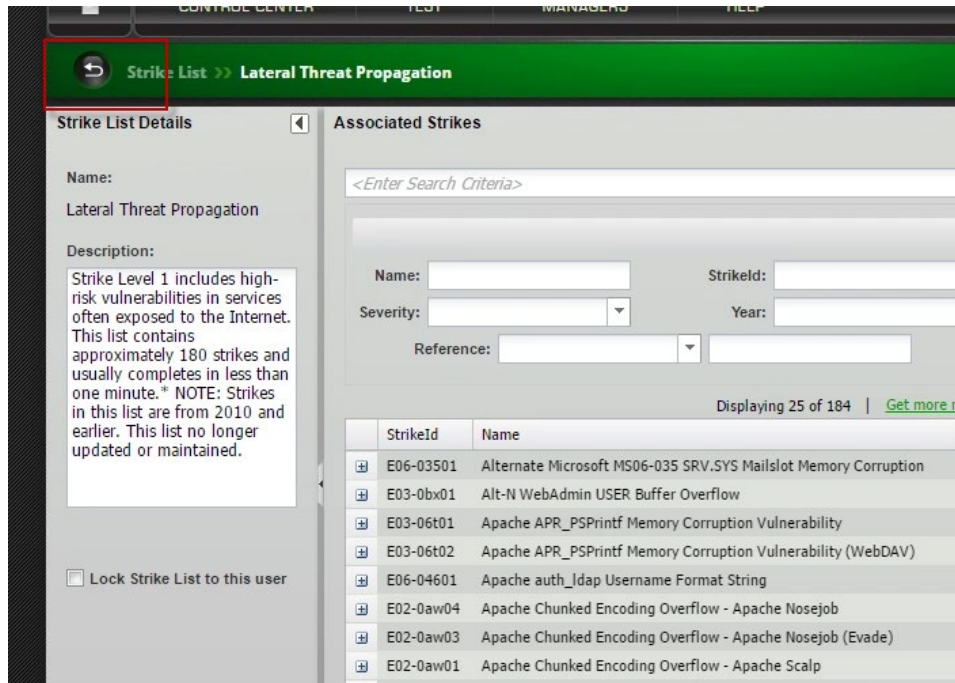


19. Name the modified Strike List and click **OK** to save.



Test Methodologies for Virtual Environment Security

20. Click the **Back** button to return to the Test Workspace.

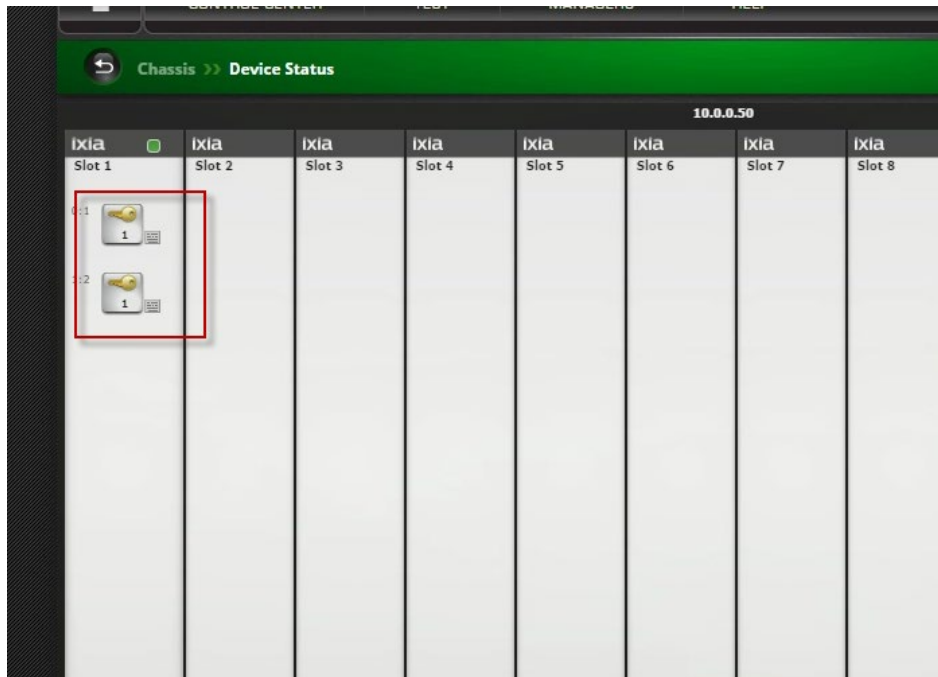


21. Click the **Device Status** button to view the virtual load modules.



Test Methodologies for Virtual Environment Security

22. Click on each virtual port to assign ownership to the current user for use in the test.



23. Click on the **Close** button to finish working with the virtual chassis.



24. Click **Save and Run** button to execute the Lateral Threat Propagation test case.



Result Analysis

This section covers the key statistics and events that BreakingPoint VE provides for this type of test.

Real-time Statistics

After the test is started and initialized, the screen will automatically switch to Real Time Statistics window.

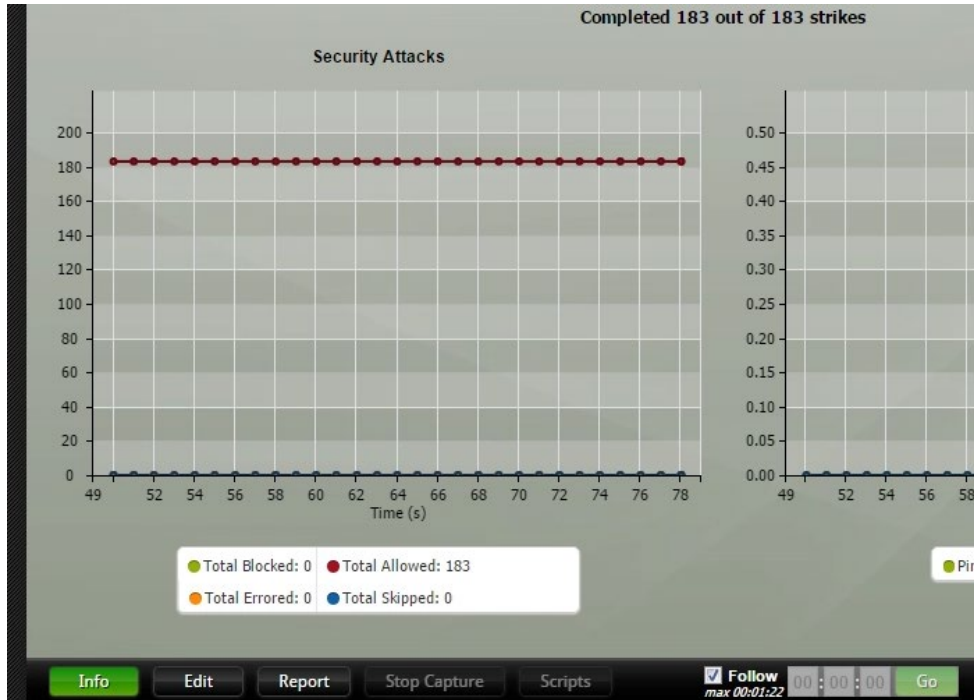
The “Attacks” tab shows how many attacks will be run and how many of those have been completed thus far. At the bottom of this page are more detailed cumulative statistics, the most relevant being:

- Blocked: Of those attacks completed, how many have been blocked by the device
- Allowed: Of those attacks completed, how many have been allowed by the device

Also, some of the strikes can be *Skipped* which means that some of the attacks configured in the strike list are “deprecated” and therefore they will be skipped. Users that prefer to run them can create an evasion profile that will have “Allow Deprecated” option set. Skipped also will occur when there is an IPv6 network neighborhood configured to run a Strike that requires IPv4, and vice versa.

Test Methodologies for Virtual Environment Security

The test will stop once all the strikes have been completed.



Test Variables

Make sure to tailor the East-West traffic to best match your server-to-server communication and enhance the strike list by adding additional strikes in each direction for different protocols. Some examples of critical strikes used in a threat propagation test are included in the table below.

STRIKE ID	NAME	PROTOCOL	DIRECTION	REFERENCE	SEVERITY
E15-8qy01	IBM WebSphere Application Server Remote Collections Code Execution Vulnerability	http	c2s	CVE:2015-7450	CRITICAL
E12-6x001	Java Sandbox Breach via Glassfish	http	s2c	CVE: 2012-5076	CRITICAL
G04-3hy01	Macromedia_JRun_4_mod_jrun_Buffer_Overflow_Vulnerability_attack	tcp, udp	c2s	CVE: 2004-0646	CRITICAL

Test Methodologies for Virtual Environment Security

STRIKE ID	NAME	PROTOCOL	DIRECTION	REFERENCE	SEVERITY
E13-hbf01	McAfee Web Reporter JBoss EJBInvokerServlet Marshalled Object Code Execution	http	c2s	url: http://secunia.com/advisories/54788/	CRITICAL
G08-63c01	Oracle_BEA_WebLogic_Server_Apache_Connector_Buffer_Overflow_attack-Atk-1	tcp, udp	c2s	CVE: 2008-4008	CRITICAL

Conclusions

Ixia's BreakingPoint VE (Virtual Edition) enables the emulation of advanced lateral threat propagation. Insertion of BreakingPoint VE into a continuous integration / continuous deployment (CICD) chain including virtual firewalls and security appliances proves the effectiveness of advanced security controls in preventing a variety of server-to-server threat vectors that can be propagated within the virtualized data center.

Test Case: Performance Acceleration with Intel DPDK Support

Overview

The industry has been working on getting as much performance as possible out of VNFs, which has led to several optimization technologies and strategies. One key technology that is making its way into a wide variety of VNFs is the data plane development kit (DPDK) from Intel. According to Intel, packet processing and throughput is significantly enhanced to the tune of up to 10X improvement compared with systems that have not used the DPDK. Software engineers can compile their VNFs with the DPDK code base to take advantage of the kit's libraries for executing code in the Linux user space that provides several optimizations. There are software abstractions for large-scale multi-core systems, manager routines for memory via use of a ring to store free objects in the memory pool, and poll mode drivers that work asynchronously to reduce processing overhead—just to name a few.

Objective

The BreakingPoint VE open virtual architecture appliance (OVA) for the virtual blade can be downloaded from the Ixia Support website and deploys with VMXNET3 interfaces on VMware ESXi to take advantage of the Intel DPDK enhancements supported by newer versions of VMware Hypervisor.

Setup

This new Performance Acceleration feature has some important prerequisites:

1. The server processor that the virtual blade will be using needs to have single instruction multiple data (SIMD) extensions of SSSE3 or above enabled
2. At least 8GB of RAM needs to be allocated for the virtual blade
3. The hypervisor needs to be a version of VMware ESXi 6.0 with a build number of 3029758 or above

Ixia engineering recommends using default settings for VMware hypervisor configuration options under the **Hypervisor->Configuration->Software->Advance Settings->Net path**.

Test Methodologies for Regenerating Production Network Traffic

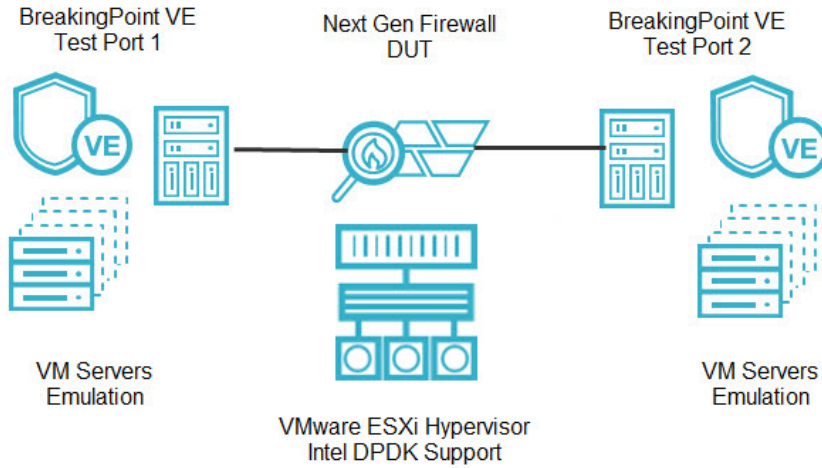


Figure 38. BPS-VE ports deployed on Intel DPDK via VMXNET3

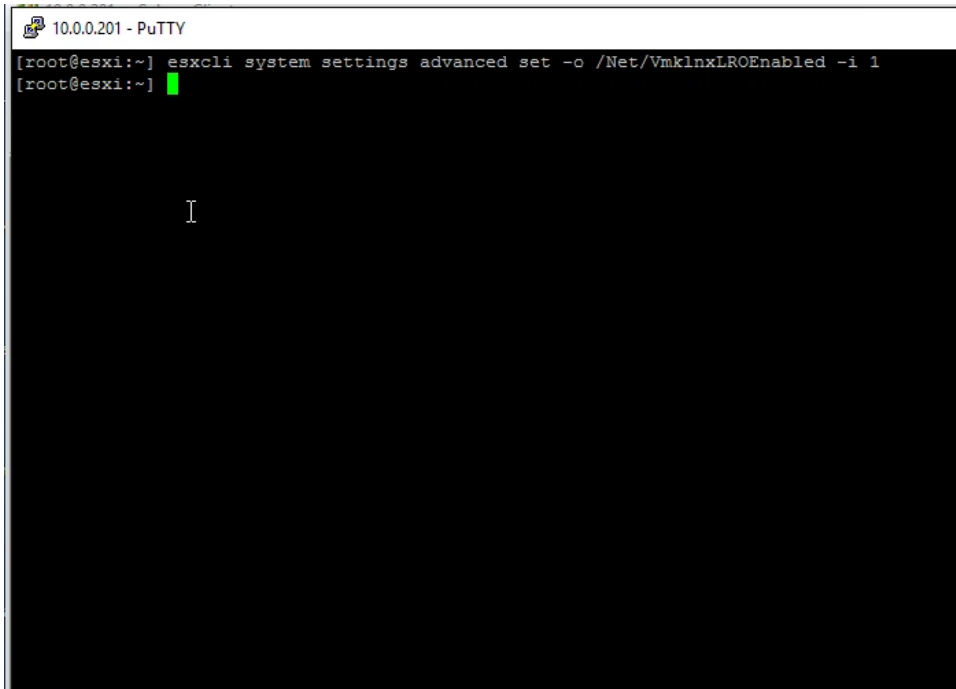
Step-by-Step Instructions

1. Configure (TCP segmentation offload) TSO and (large receive offload) LRO in the VMware ESXi 6.0 hypervisor via SSH. Refer to the Test Parameters table for hypervisor configuration option values. Example: **esxcli system settings advanced list -o /Net/VmkernelLROEnabled**

```
10.0.0.201 - PuTTY
[root@esxi:~] esxcli system settings advanced list -o /Net/VmkernelLROEnabled
Path: /Net/VmkernelLROEnabled
Type: integer
Int Value: 1
Default Int Value: 0
Min Value: 0
Max Value: 1
String Value:
Default String Value:
Valid Characters:
Description: LRO enabled in vmkernel
[root@esxi:~]
```

Test Methodologies for Regenerating Production Network Traffic

2. Refer to the VMware knowledge bases for additional details on how to verify or enable TSO and LRO if they are disabled via SSH. Example: `esxcli system settings advanced set -o /Net/VmknxLROEnabled -i 1`



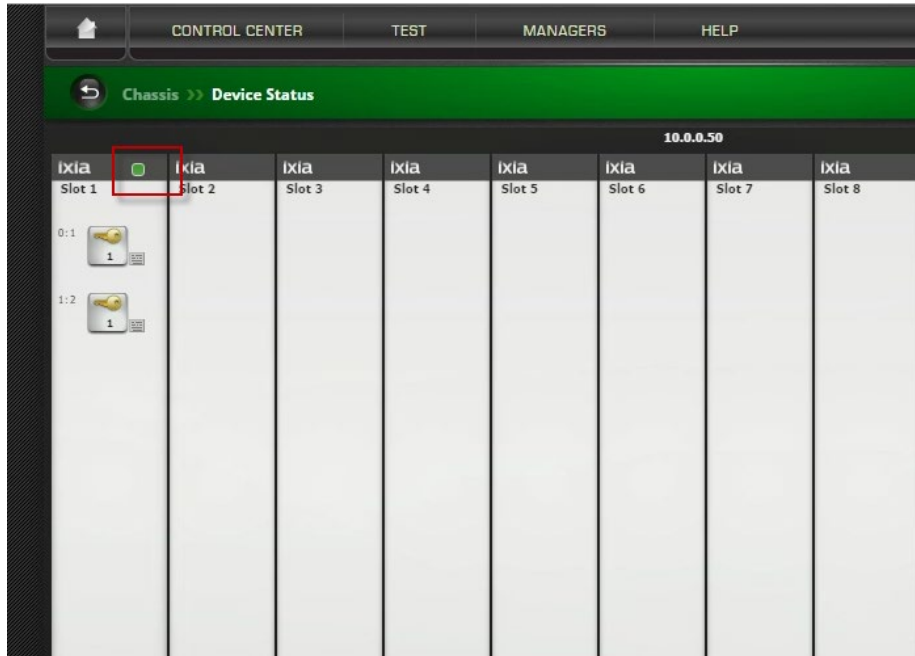
```
10.0.0.201 - PuTTY
[root@esxi:~] esxcli system settings advanced set -o /Net/VmknxLROEnabled -i 1
[root@esxi:~]
```

3. Navigate to the Device Status page in the BreakingPoint web interface.

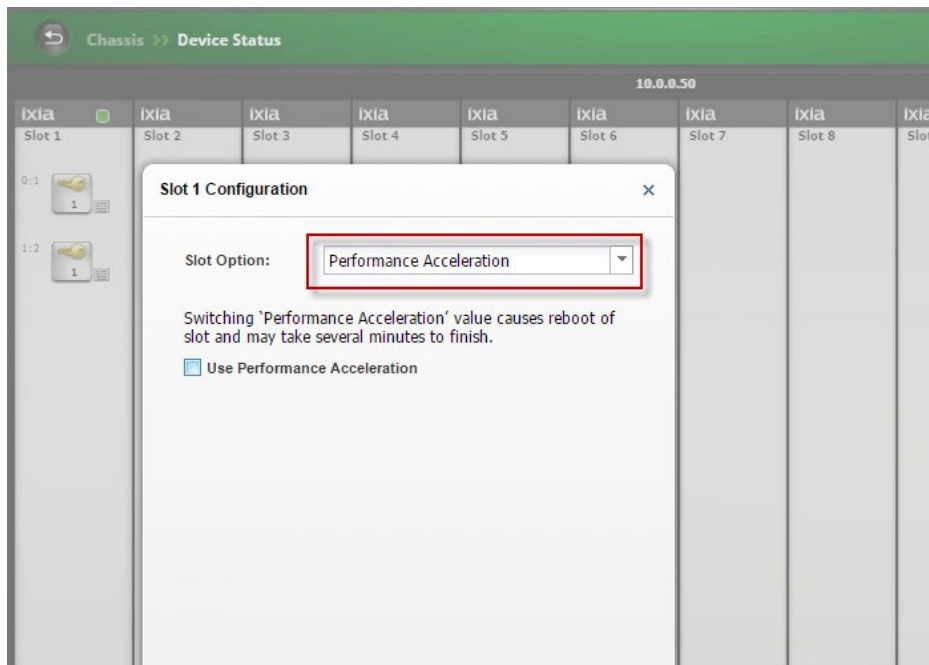


Test Methodologies for Regenerating Production Network Traffic

- Each vBlade on the Device Status page of the GUI displays a green slot configuration button at the top-right corner.

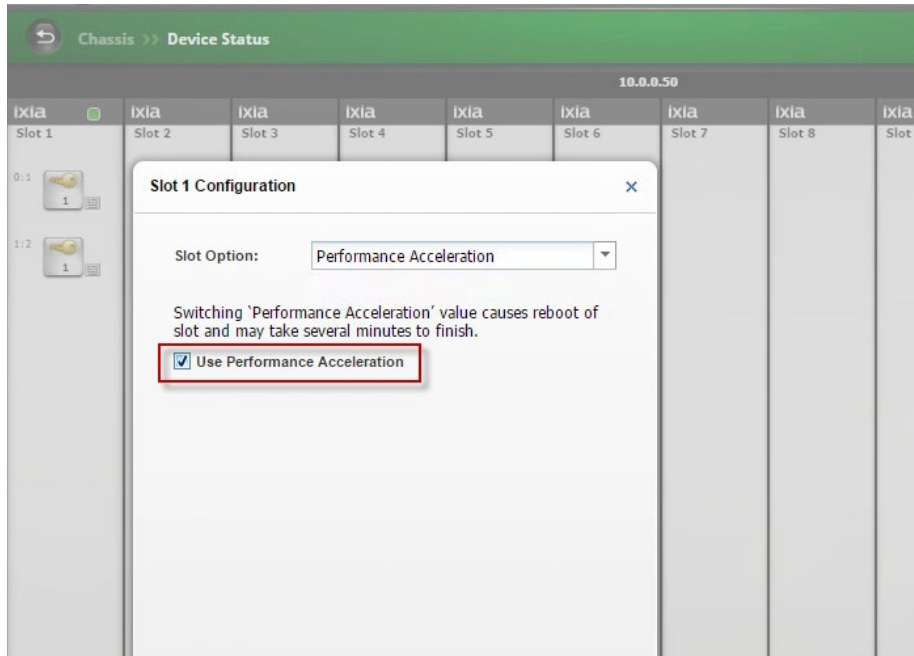


- Click the slot configuration button to see a pop-up window offering an option to Enable Performance Acceleration.

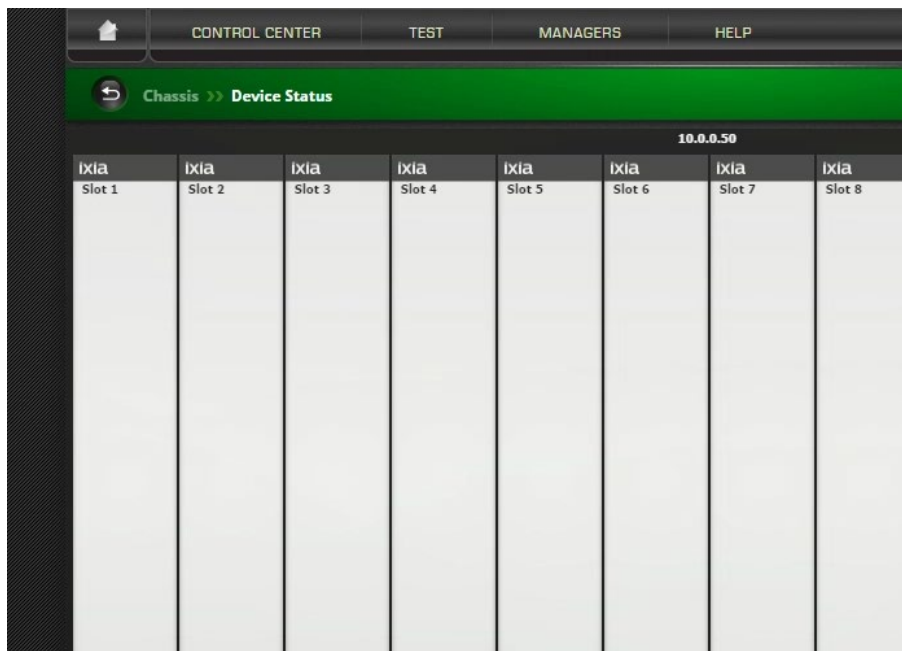


Test Methodologies for Regenerating Production Network Traffic

6. Select the Enable Performance Acceleration option and click the Apply button. Note that there is a warning regarding the fact that the virtual blade will need to automatically reboot itself to switch modes into the performance optimized code path, which is not the default.

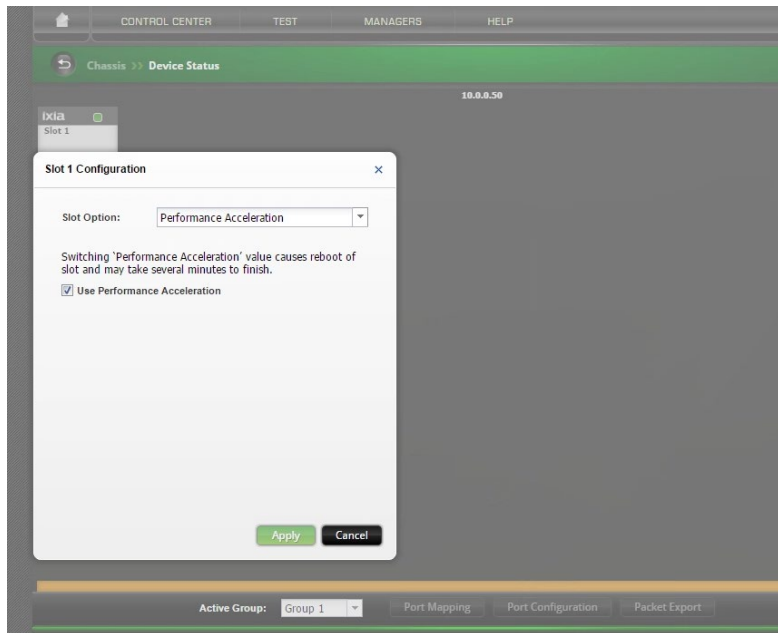


7. The blade will disappear from the Device Status page after a brief period of time and will reappear upon completion of the reboot.

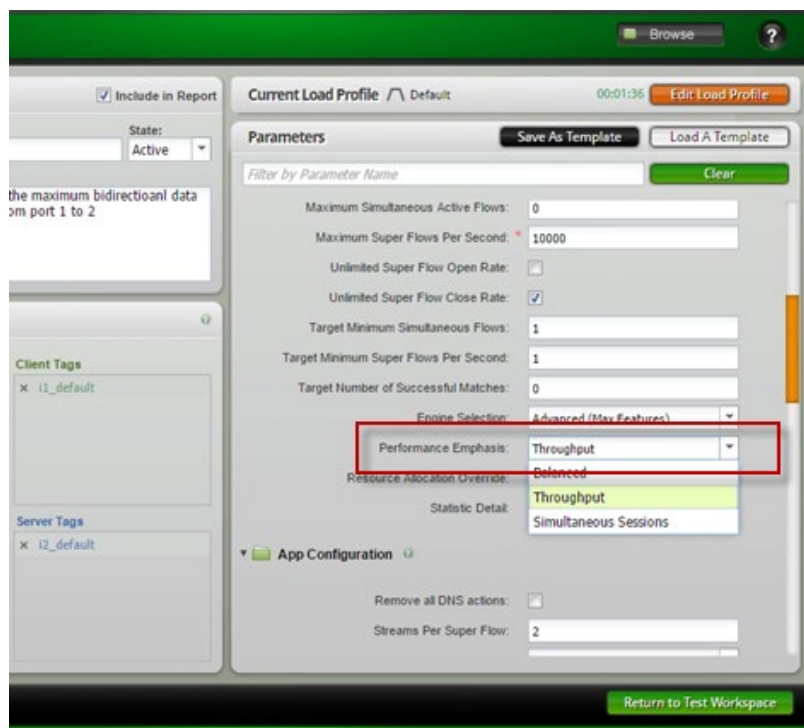


Test Methodologies for Regenerating Production Network Traffic

- To verify that the change is in effect on a virtual blade, you can again click the green slot configuration button and see the checkbox next to Enable Performance Acceleration and the Apply button will be grayed out.



- Select a Performance Emphasis of Throughput from the dropdown in each Application Sim component you are going to use in a test that makes use of the virtual blades you previously enabled for Performance Acceleration.



Result Analysis

Below are some representative performance numbers showing the significant impact Intel DPDK in a VMXNET3 environment can have on throughput performance for different traffic mixes. Be sure to benchmark before DPDK and after DPDK for your specific environment as processor speeds, hypervisor settings, resource oversubscription and other factors can impact performance.

	Before DPDK	After DPDK
HTTP Throughput	21.1Gbps	50.1Gbps
Application Mix Throughput	16.8Gbps	46.4Gbps

Test Variables

The default Performance Emphasis is balanced and while you will benefit from performance optimizations applied by the Ixia engineering team over the last several releases, you need to use Throughput Emphasis to signal to the BPS engine that you want the additional benefits of Intel DPDK in a VMXNET3 environment.

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
Throughput Emphasis	Performance	Balanced (Non-DPDK code path)
/Net/UseHwTSO6	1	0
/Net/UseHwTSO	1	0
/Net/Vmxnet3SwLRO	1	0
/Net/Vmxnet3HwLRO	1	0

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
/Net/VmklInxLROEnabled	1	0
/Net/TcpipLRONoDelayAck	1	0
/Net/TcpipHWLRONoDelayAck	1	0
/Net/TcpipDefLROEnabled	1	0
/Net/TcpipDefLROMaxLength	32768	1 - 65535
/Net/VmklInxLROMaxAggr	6	0 - 24
/Net/LRODefThreshold	4000	0 - 65535
/Net/LRODefBackoffPeriod	8	0 - 65535
/Net/LRODefUseRatioNumer	1	0 - 255
/Net/LRODefUseRatioDenom	3	0 - 255
/Net/LRODefMaxLength	65535	1 - 65535

Conclusions

Application test teams continue to push the performance limits of what common off-the-shelf hardware and virtual network function infrastructure (VNFI) can achieve. To stay ahead of the explosion in growth of performance testing of virtual network infrastructures, our engineering team is leveraging every tool in their toolbox.

Long gone are the days of simply needing to verify just basic connectivity between groups of unrelated virtual machines. Today's virtual networks are many-layered and have complex interdependencies, yet still need to fulfill the service level agreements (SLAs) traditionally serviced by dedicated hardware. Users can now accelerate the performance of BreakingPoint VE in generating application test traffic and security test traffic by taking advantage of Intel DPDK functionality

Test Methodologies for Regenerating Production Network Traffic

Service providers, enterprises and network equipment manufacturers have always struggled to accurately mimic production network traffic in a controlled test environment for fault analysis or to validate architectures and devices before deployment. With today's increasingly complex networks, dynamic applications, and user behavior, this has always been a time-consuming, complicated, and oftentimes frustrating process. The reality is that there has been no major innovation in this field over a long period of time and current existing solutions are not often suitable for today's network environments characterized by a sheer volume of traffic, high quality standards and an ever-increasing demand in fast turnaround times to fix issues.

Traditional approaches like replaying packet captures have many disadvantages such as:

- Slow and time consuming: capturing real traffic, filtering, exporting, loading capture to a replay tool, configuring test
- Extremely limited: often a small window of time can be replayed due to the size of capture
- Potential policies in production network may not allow distributing packet captures that include the actual payloads

Fortunately, there are technologies that can be leveraged to provide an alternative solution, such as traffic metadata. Traffic metadata consists of key data that summarizes and characterizes various network traffic aspects - it is basically data about data. One of the most common and widely used formats to transfer such summarized network information is NetFlow, a protocol originally developed by Cisco Systems which was ratified into RFC 7011 – IPFIX (or otherwise known as NetFlow v10). NetFlow exported data contains a basic subset of information like: transport protocol (TCP/UDP), source and destination IP addresses and ports, start time, end time, and bytes exchanged for that flow.

Using NetFlow records from production traffic to generate a **test configuration** file yields unprecedented benefits, such as:

- Capturing the temporal network behavior by following the dynamic traffic composition changes over time
- Recording traffic characteristics over extended periods of time
- Using only traffic metadata enhances internal and external collaboration by eliminating potential policies against distributing captured network traffic that include real payloads

In this test methodology, we will introduce an innovative approach for translating production network insights into real traffic configurations for lab test environments.

Test Case: Pre-Deployment Validation with Production Application Mixes

Overview

Network operators, ranging from large service providers to private, enterprise networks have a constant pressure to keep up with the ever changing and complex set of requirements which eventually translates in implementing new or improved architectures, update security policies or constantly apply software patches. While most of the energy and focus is targeted at addressing these items, a very important aspect of this process is testing and validation before moving to production. Implementing a new network architecture or improving the existing infrastructure without proper testing and validation can lead to major performance degradation or even service disruptions.

When performing such pre-deployment validation tests, it is critical to use the traffic profile that most closely represents the production environment where the solution will be placed.

Using Ixia's TrafficREWIND virtual appliance, users can easily translate production network insight into test traffic configurations with high fidelity. TrafficREWIND is fed with network traffic information using Ixia's ATIP (Application and Threat Intelligence Processor) generated NetFlow metadata. ATIP is a visibility solution that inspects and reports on production network traffic. TrafficREWIND records and synthesizes these NetFlow characteristics and generates a test traffic profile. The resulting test traffic profile can be used in Ixia's BreakingPoint application and security test solution.



Figure 39. Innovative solution that translates production network insights into traffic stimulus for realistic pre-deployment testing

Objective

The objective of this test is to validate a new or improved network architecture implementation (which can range from a complete replacement of various network elements to just software patches or new traffic policies) with simulated production-like application traffic as sampled from a live network.

Setup

The current setup consists of two distinct environments:

1. The production environment: to monitor the live network and translate production network insight into test traffic profiles
 - Vision ONE (with ATIP): to inspect production traffic and generate NetFlow records
 - TrafficREWIND virtual appliance: records and synthesizes NetFlow records and generates a test traffic profile



2. The lab test bed environment: used to validate the new or improved network architecture
 - Ixia BreakingPoint: application and security test solution used to run the TrafficREWIND exported test traffic profile, hence emulating the production network traffic characteristics.
 - DUT/SUT (Device Under Test/System Under Test): the device or the collection of devices and elements comprising the new or improved network architecture to be implemented in the production environment.

Test Methodologies for Regenerating Production Network Traffic

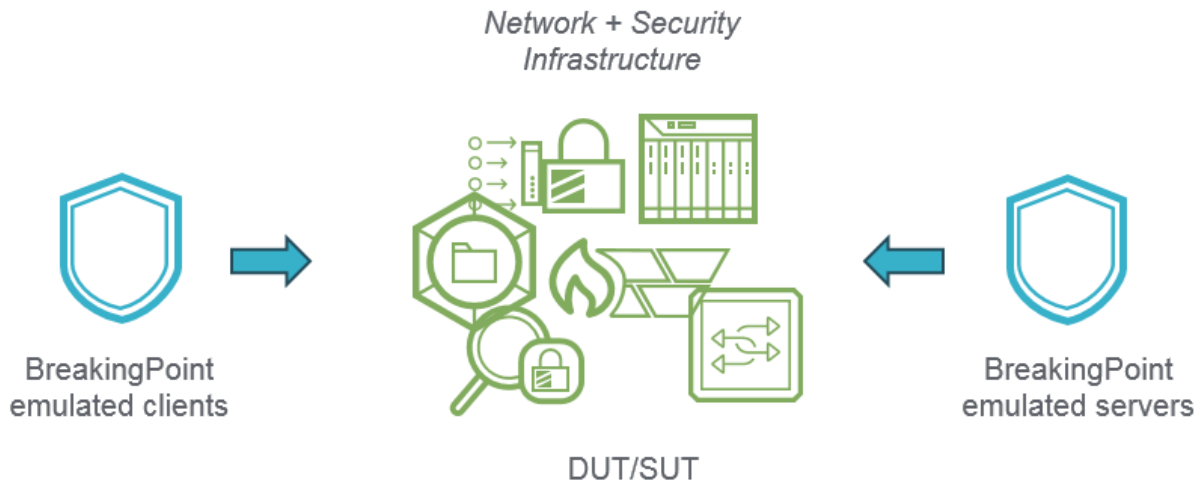
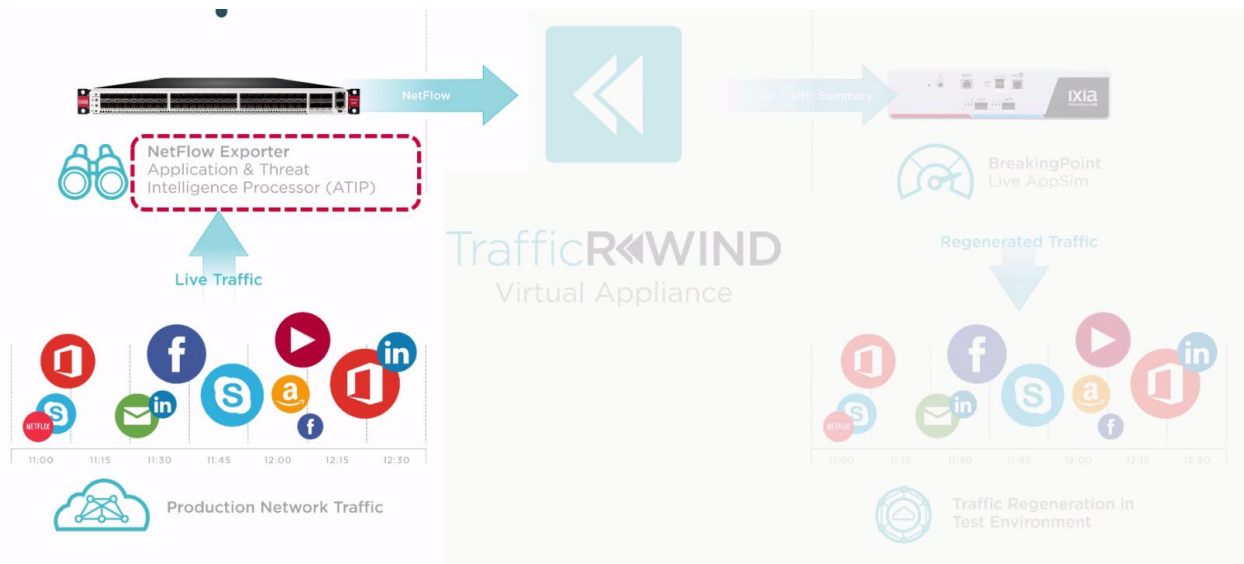


Figure 40. Ixia BreakingPoint test ports connected directly to the system under test.

Step-by-Step Instructions

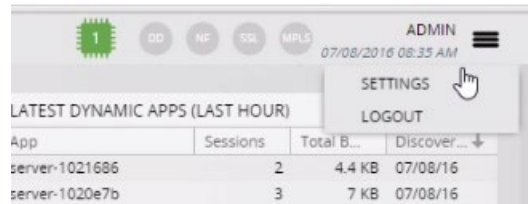
This section provides step-by-step instructions to execute this test scenario using the tools mentioned above.

First, we will configure the elements of the production environment. Since the scope of this test methodology is not to go exhaustively through the configuration options of Vision ONE we will start by configuring the ATIP module to export NetFlow records:

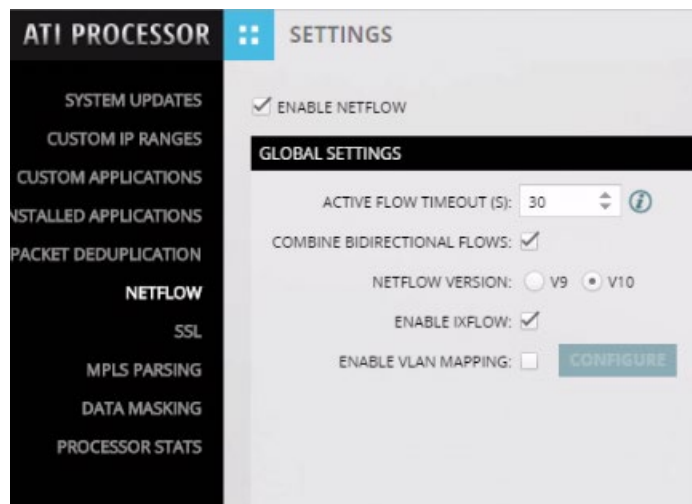


Test Methodologies for Regenerating Production Network Traffic

1. Using a web browser, connect to the ATIP UI and from the upper right corner navigate to the **Settings** menu:



2. In the **NetFlow** menu select the following:
 - a. Check the **Enable NetFlow** checkbox
 - b. Check the Combine Bidirectional Flows checkbox
 - c. Select NetFlow Version 10
 - d. Check the **Enable IxFlow** checkbox



- e. In the bottom section of the page, **NETFLOW COLLECTORS** pane, configure the IP address of the TrafficREWIND virtual appliance: in this case, it is *10.216.102.130* and the destination port is *UDP 20001*

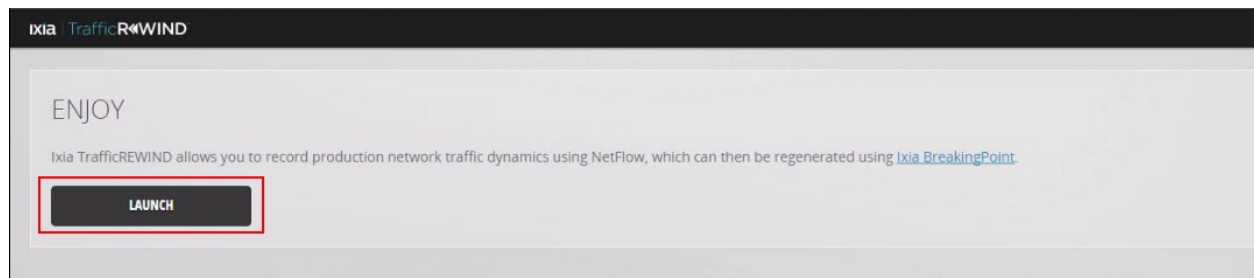
NETFLOW COLLECTORS						
ENABLED	IP ADDRESS	PORT	PROTOCOL	SAMPLE RATE	SAMPLE UNIT	FILTERS
YES	10.216.102.39	20001	UDP	100	%	0
NO	10.200.117.92	20001	UDP	100	%	0
NO	10.200.117.91	20001	UDP	100	%	0
NO	10.216.102.92	20001	UDP	100	%	0
NO	10.216.111.1	20001	UDP	100	%	0
NO	10.216.102.125	20001	UDP	100	%	0
YES	10.216.102.130	20001	UDP	100	%	0
NO		0		100	%	0

Test Methodologies for Regenerating Production Network Traffic

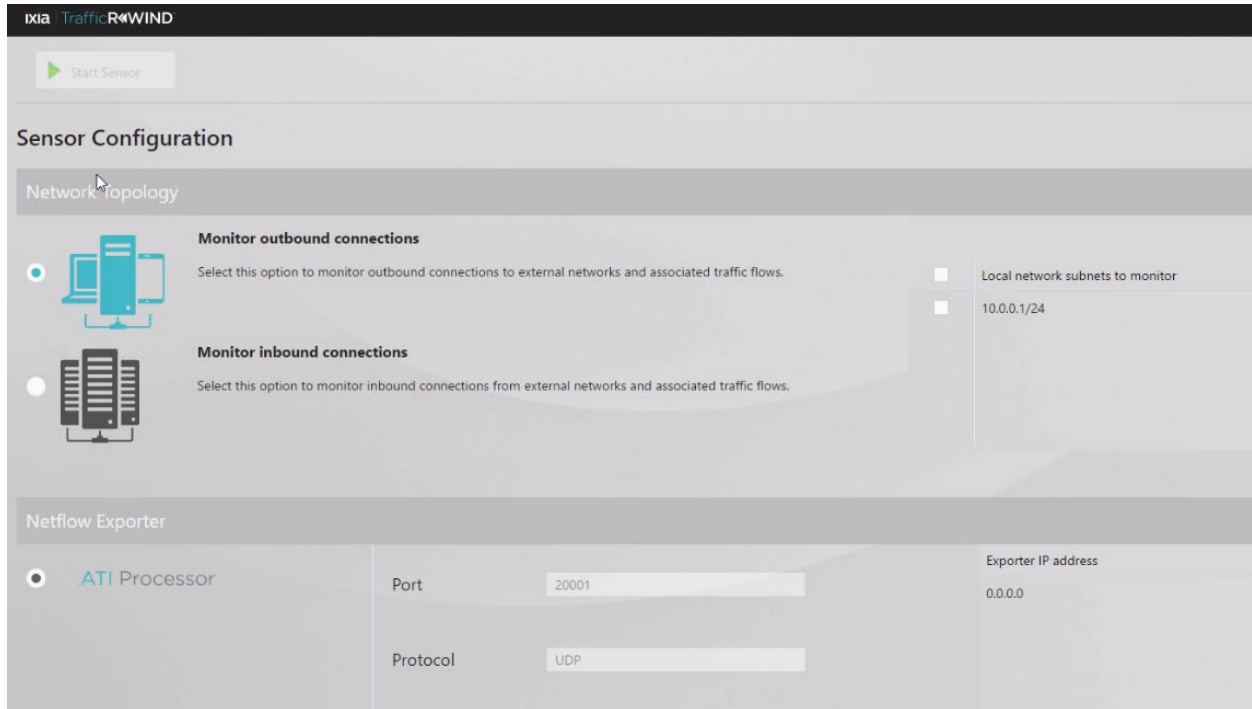
Now, that the ATIP module has been configured to export NetFlow records, we can proceed and configure the TrafficREWIND virtual appliance, the central piece of this end-to-end solution.



3. Using a web browser, connect to the TrafficREWIND IP address and use your Ixia account credentials to log in. Once logged in, click on the Launch button to start a new TrafficREWIND UI session:



4. The new displayed page is the TrafficREWIND Sensor configuration:



The screenshot shows the 'Sensor Configuration' page in the IXIA TrafficREWIND interface. At the top left, there is a 'Start Sensor' button. The main section is titled 'Sensor Configuration' and is divided into two parts: 'Network Topology' and 'Netflow Exporter'. In the 'Network Topology' section, there are two radio button options: 'Monitor outbound connections' (which is selected) and 'Monitor inbound connections'. To the right of these options, there are checkboxes for 'Local network subnets to monitor' and a text input field containing '10.0.0.1/24'. In the 'Netflow Exporter' section, there is a radio button for 'ATI Processor'. Below this, there are input fields for 'Port' (20001), 'Protocol' (UDP), and 'Exporter IP address' (0.0.0.0).

This is the page where the entire TrafficREWIND configuration is being done.

First thing we need to do is to configure the network context we want to monitor. There are basically two options:

- **Monitor Outbound Connections** – mainly used in enterprise-like scenarios where most of the traffic is being initiated from the inside-out
- **Monitor Inbound Connections** – typical for datacenter environments where most of the sessions are being initiated from the outside-in

In our case, since the new architecture to be implemented is relevant for an enterprise network environment we will choose **Monitor Outbound Connections** option.

5. Once the network topology has been selected, we also need to define the corresponding **Local network subnets to monitor**: in the case of **Monitor Outbound Connections** we need to specify the source subnet(s) only, while for **Monitor Inbound Connections**, the destination subnet(s) only.

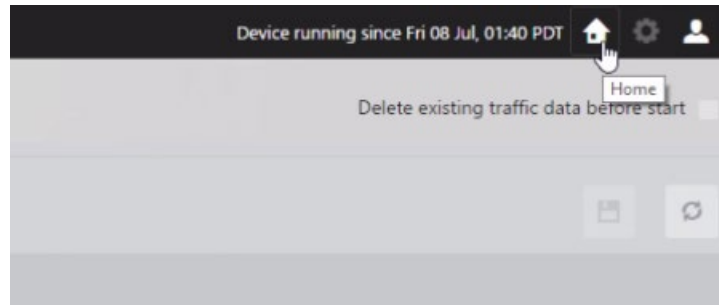
In our example, we will choose *10.0.0.1/16* as the **Local network subnets to monitor**.

Note: configuring the correct network subnets is very important for TrafficREWIND as it is being used during the interpretation and synthetization of NetFlow records.

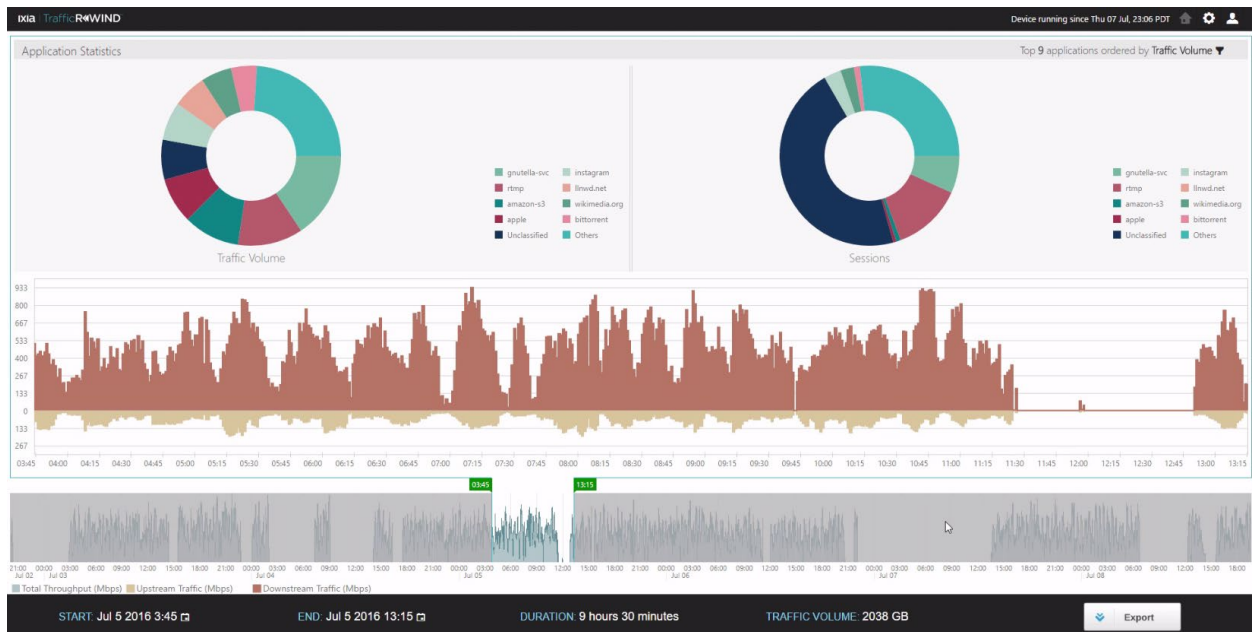
6. Last step is to configure the NetFlow **Exporter IP address**, from which traffic metadata will be received which obviously in our case is the previously configured ATIP. Now we can save the configuration (from the upper right corner) and once the sensor is ready to parse NetFlow information, the Start Sensor button becomes active.

Test Methodologies for Regenerating Production Network Traffic

7. Press the **Start Sensor** button to trigger the process of recording and synthesizing traffic characteristics.



8. From the upper right corner, press the home button to switch to the main dashboard which offers **Application Statistics** distribution charts analyzing the traffic volume and sessions of the imported NetFlow.

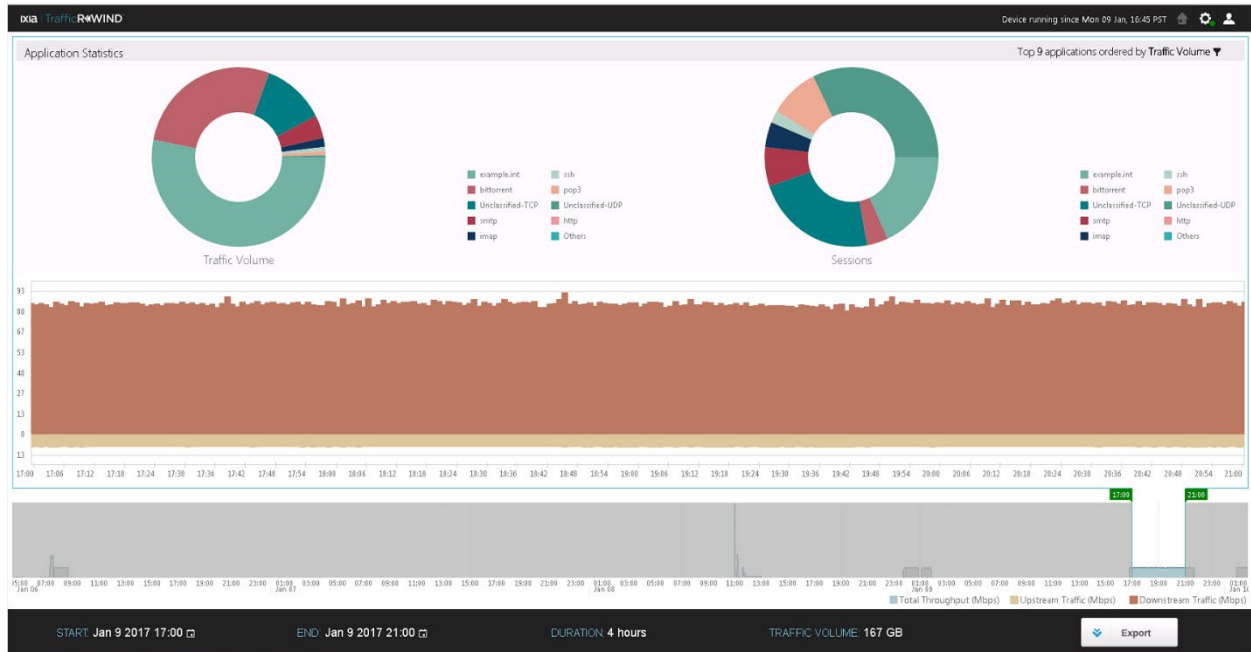


Note: Typically, it will take about 6 to 8 minutes from the time the sensor has been started to get main dashboard populated with traffic data information.

Having access to the production network traffic characteristics, users can select a particular time segment (up to 1 day) of the recorded network traffic using the sliding window selectors at the bottom of the screen and export it as test traffic profile for use with BreakingPoint. Traffic composition and volume has a granularity of 15 minutes (i.e. each 15 minutes, application distribution and traffic volume is changing according to the averaged recorded conditions for that time period).

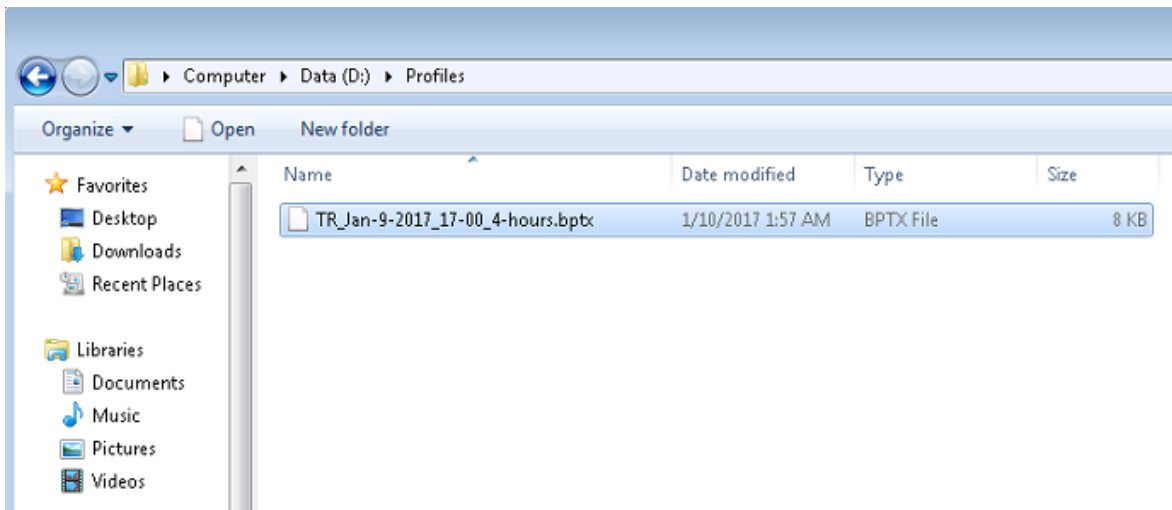
Test Methodologies for Regenerating Production Network Traffic

9. Select the required time segment of the recorded network traffic using the sliding window selectors at the bottom of the screen and click on the **Export** button export.



For our test case, we will be selecting a 4 hours' representative time segment which will encompass the most relevant and critical traffic activities. The goal is to validate the new or updated architecture with a traffic profile that is as close as possible to production traffic.

After pressing the Export button, a .bptx traffic profile file will be generated which will be used with the Ixia BreakingPoint test tool to recreate the corresponding traffic conditions. Save the exported traffic profile in a location where it can be used later and imported in BreakingPoint:



Test Methodologies for Regenerating Production Network Traffic

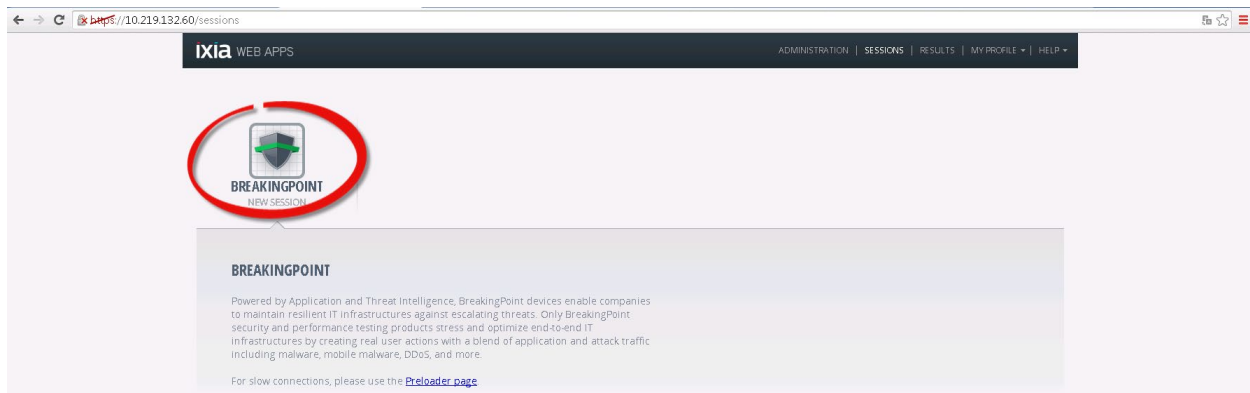
Next, we will proceed with the last part of this test: lab testing validation.



The objective of this test is to reproduce the production network traffic characteristics and to validate the new architecture which is first designed in a lab environment in the form of a SUT (System Under Test). Ixia BreakingPoint will be used as a wrap-around traffic generation tool connecting the internal emulated clients on one side and the external servers on the other side.

Below section provides step-by-step instructions to execute this test.

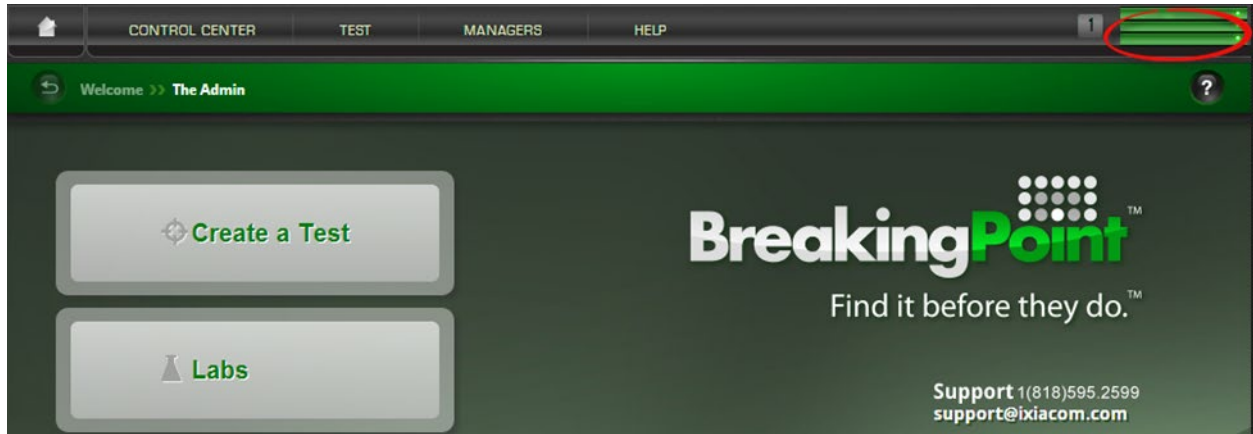
10. Use a web browser connected to the Ixia Web Apps GUI and start a new BreakingPoint Web Session:



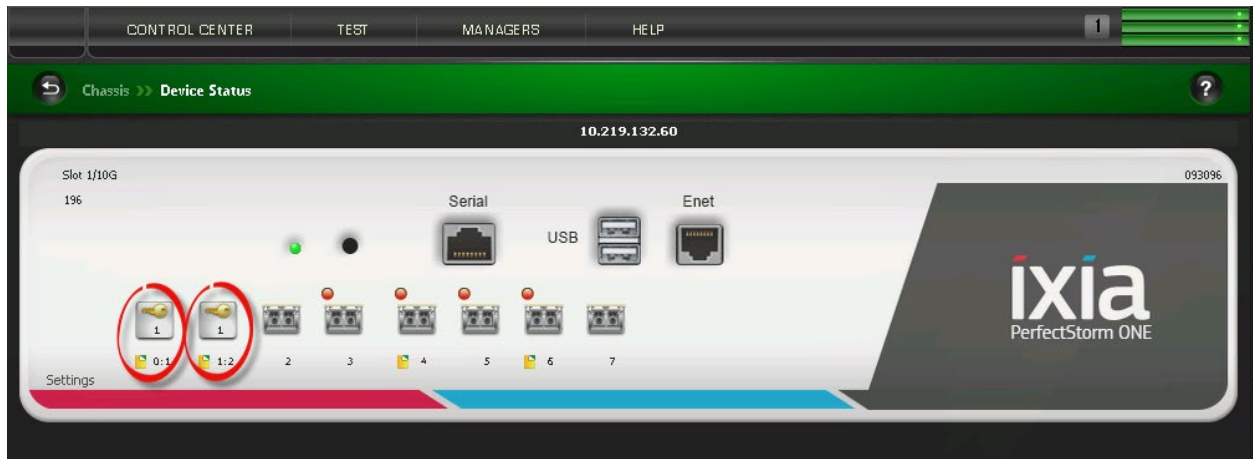
Test Methodologies for Regenerating Production Network Traffic

11. Once the BreakingPoint web home page loads, reserve the test ports to be used in the physical test setup to generate/receive traffic:

a. Click on the Device Status button located on the upper right corner:



b. In the new screen select the physical ports that are to be used in the test:



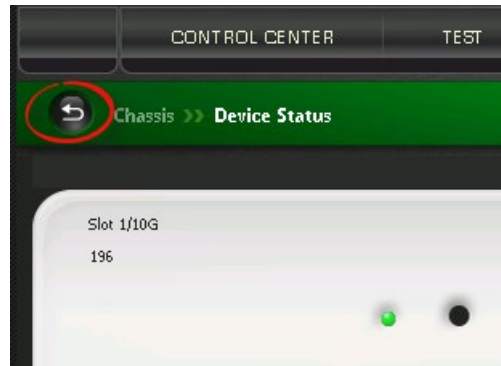
In this example, we will use physical ports 0 and 1 from the PerfectStorm One appliance.

Note: An important thing to remember when reserving your ports is the order in which you reserve them. Whenever you reserve a port, the system will automatically map that port to a logical interface: first reserved port will be Interface1, the second reserved port will be Interface2, and so on. The logical interface parameters (MAC and IP address along with other parameters) will be configured as part of the Network Neighborhood, as we will see below.

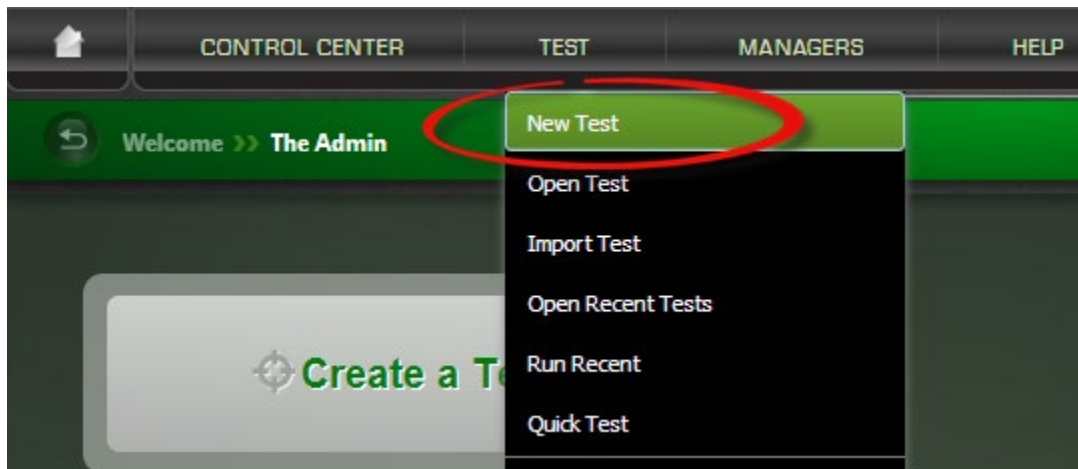
Note: The resources of each blade are allocated in proportion to the number of ports reserved on the blade. In some cases, it may be necessary to reserve additional ports to secure enough resources for the test being performed. The sessions and bandwidth available to a test may be insufficient to adequately perform the test if too few ports have been reserved.

Test Methodologies for Regenerating Production Network Traffic

- i. Once the proper test ports have been selected, click on the back arrow to return to the initial screen:



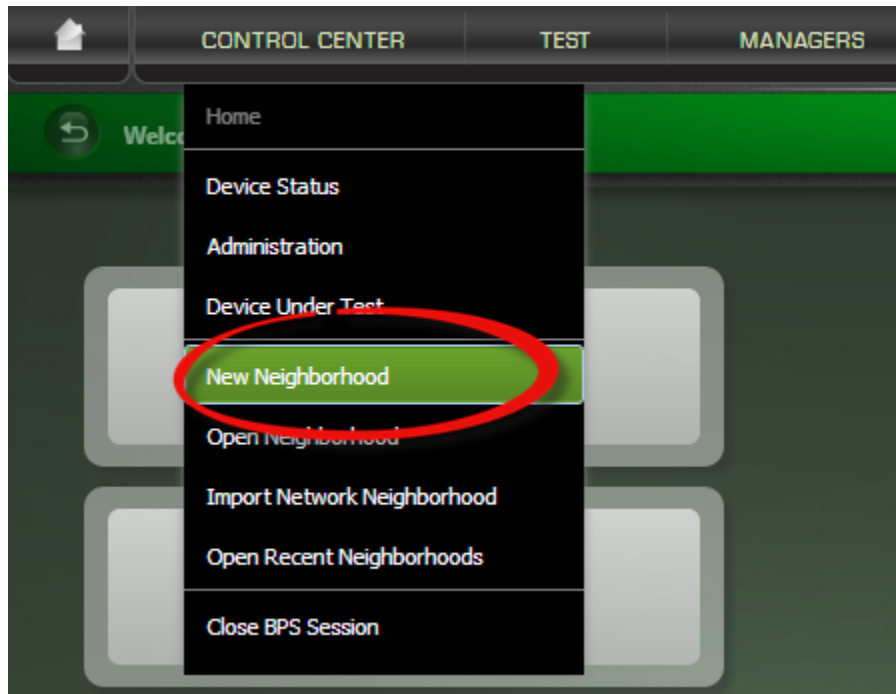
12. Next, select **Test** -> **New Test** option from the upper menu bar to start configuring the test:



13. Since we already reserved the physical test ports (and, if applicable extra ports for more resource reservation) now we need to configure the MAC and IP layer parameters like MAC address, VLANs, IP addressing scheme, MTU, etc. All these settings, among others are controlled/defined from the **Network Neighborhood**:

Test Methodologies for Regenerating Production Network Traffic

- a. From the upper menu bar select **Control Center** -> **New Neighborhood**:



- b. Select the DUT type used for the test and a corresponding default Network Neighborhood will be created:



For this test, we will use *ROUTER* as DUT type.

Test Methodologies for Regenerating Production Network Traffic

- c. Enter an easy-to-remember name for the new Network Neighborhood and click **OK**:

- d. According to the DUT type we previously selected, a default Network Neighborhood will be created. Since we chose *ROUTER* DUT type, the following elements are automatically created:

- i. Two *INTERFACES*: Provides access to interface level parameters like MTU, MAC, Impairments, and Packet Filters.
- ii. One *IPv4 EXTERNAL HOSTS*: configures the IP address of the external end hosts/servers. We will not use this element for our specific test scenario.
- iii. Two *IPv4 STATIC HOSTS*: Provides access to IP related parameters of the BreakingPoint emulated endpoints. The Static Hosts represents the BreakingPoint emulated internal clients and external target servers. Recall that TrafficREWIND was configured to monitor a network that was typical for enterprise environments where internal clients are initiating connections to external servers. It is a good practice to configure the same set of IP address (both for clients as well as for servers) according to the actual production network. In this example, we will use the following:
 - One element for the internal clients with 1000 IP addresses
 - One element with a 10 IP addresses for the external emulated servers.

Entry Mode		Diagram Mode		ADD NEW ELEMENT	EXPAND ALL COLLAPSE ALL	KEYBOARD SHORTCUTS	Lock Network Neighborhood to This User		
▼ INTERFACE: (2)				Untagged Virtual Interface				ADD ROW +	
DEL	ID	Number	MTU	Use vNIC MAC Address	MAC Address	Duplicate MAC Address	VLAN Key	Ignore Pause Frames	Descr
x	Interface 1	1	1500	<input checked="" type="checkbox"/>	02:1A:C5:01:00:00	<input type="checkbox"/>	Outer VLAN	<input type="checkbox"/>	
x	Interface 2	2	1500	<input checked="" type="checkbox"/>	02:1A:C5:02:00:00	<input type="checkbox"/>	Outer VLAN	<input type="checkbox"/>	
x ▶ IPv4 EXTERNAL HOSTS: (1)				External hosts used as a test target					
x ▼ IPv4 STATIC HOSTS: (2)				Simulated IPv4 endpoints				ADD ROW +	
DEL	ID	Container	Tags	Base IP Address	Count	Gateway IP Address	Netn		
x	Static Hosts i1_default	Interface 1	Internal_Clients	10.0.0.2	1000	10.0.0.1	16		
x	Static Hosts i2_default	Interface 2	External_Server	100.0.0.2	101	100.0.0.1	16		

Test Methodologies for Regenerating Production Network Traffic

Note: Make sure to configure the IP addresses according to the physical test setup address assignment scheme. In our example, we will use:

- Internal Clients will be emulated starting with IP address 10.0.0.2, 10.0.0.3, ... 10.0.3.233 (1000 clients). The gateway used to reach the external network will be 10.0.0.1 (typically the internal IP address of the exit point on the System Under Test). For easy reference configure the Tags field to “*Internal_Clients*”
- The external servers will be emulated starting with IP address 100.0.0.2, 100.0.0.3, ... 100.0.0.11 (10 Servers). The gateway used to reach the private network will be 100.0.0.1 (typically the IP address of the public interface of the System Under Test). For easy reference configure the Tags field to “*External_Servers*”

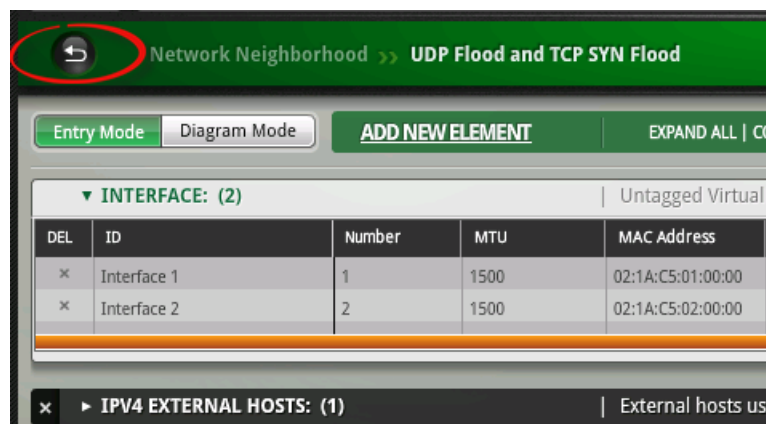
Note: Users can also add a Virtual Router Network Neighborhood element to emulate a router in front of all these Client or Server endpoints, which will provide the benefit of avoiding an ARP storm on the interface connected to the DUT/SUT (which can be caused in case many emulated IP addresses are directly connected to the SUT without a Virtual Router)

For more details on how to add and configure a Virtual Router Network Neighborhood element, please refer to Appendix A.

- e. Save the new Network Neighborhood configuration by clicking the **Save** button located on the lower right corner:

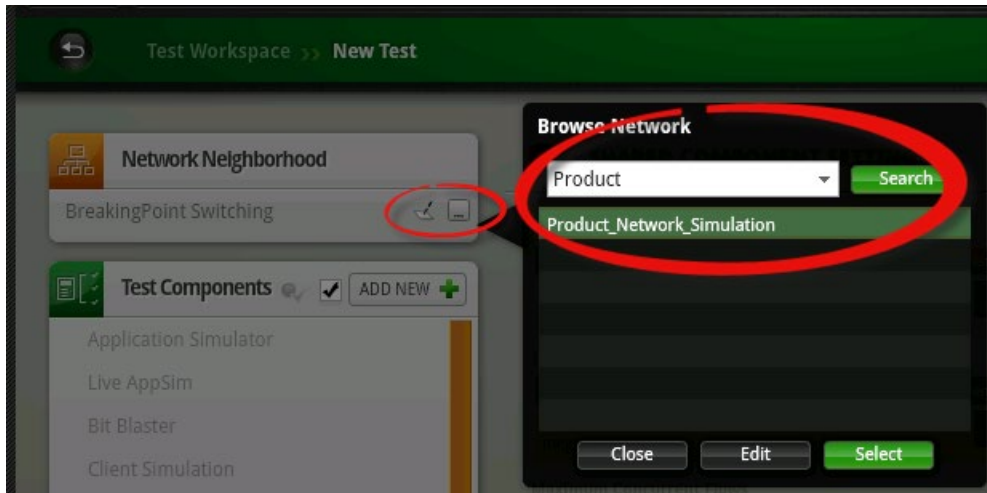


- f. Click on the back arrow to return to the main test screen:



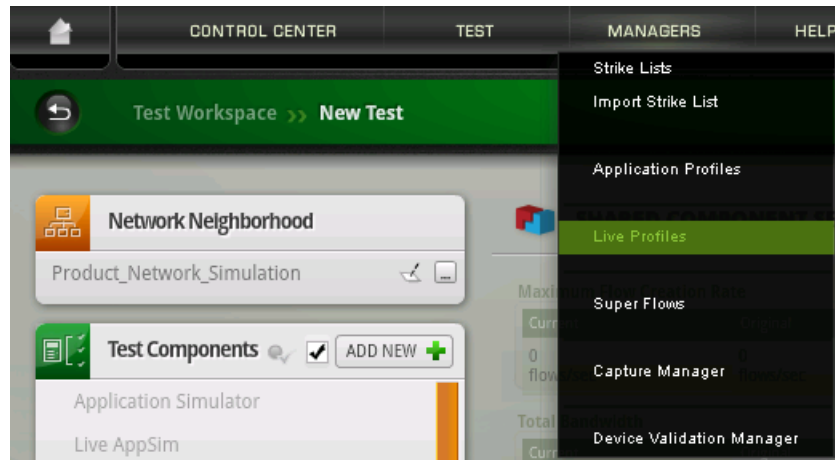
Test Methodologies for Regenerating Production Network Traffic

14. From the main test screen click on the browse button from the **Network Neighborhood** section to search and select the above created Network Neighborhood:

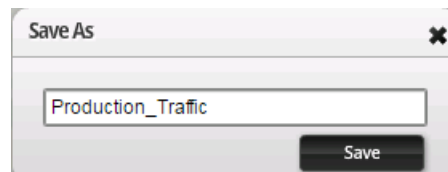


After configuring the MAC and IP layer parameters the actual test traffic needs to be created. First, we will import the TrafficREWIND exported file in the form of a **Live Profile**, then we will use the **LiveAppSim** Test Component to run the previously imported **Live Profile**.

15. Click on **Mangers** tab, and then select **Live Profiles**:

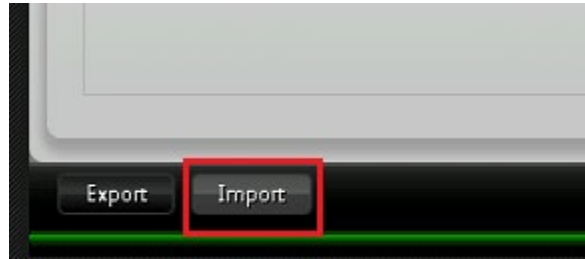


16. When prompted to save the test, select yes and provide a relevant and easy to remember name:



Test Methodologies for Regenerating Production Network Traffic

17. In the new Live Profiles screen, click on **Import** button from the lower left corner then browse and select the previously exported TrafficREWIND profile (at above step 11).



18. Once the live profile has been imported, the UI will show the details of the corresponding production traffic:



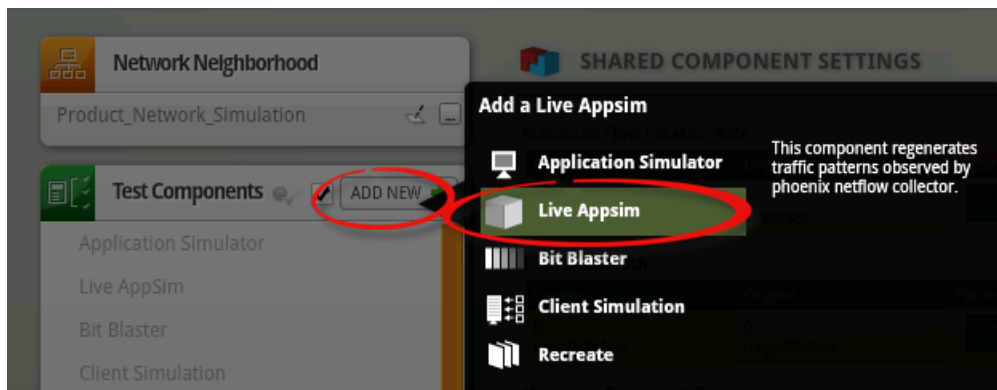
Very important to note is that aside from the application type and distribution, the temporal nature of production network traffic is represented as well, hence the traffic composition is not the same for the entire test duration. As we can see in this profile, traffic composition changes every 15 min. This evolution takes application mix testing to the next level of realism.

Once the Live Profile has been imported it is ran using the new Live AppSim test component

19. Select the previously saved test from **Tests** -> **Open Recent Tests**



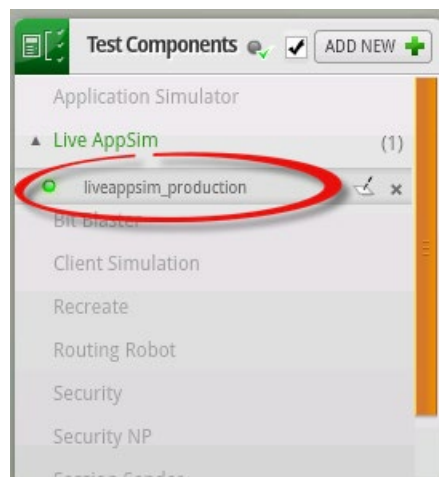
20. Click on the **ADD NEW** button from the **Test Components** section. Choose **Live AppSim** from the selection list and click on **Select** button:



21. Rename the component from *LiveAppSim_1* to *liveappsim_production* and click **Create** button

A new entry (i.e. *liveappsim_production*) will be created under **Live AppSim** Test Component.

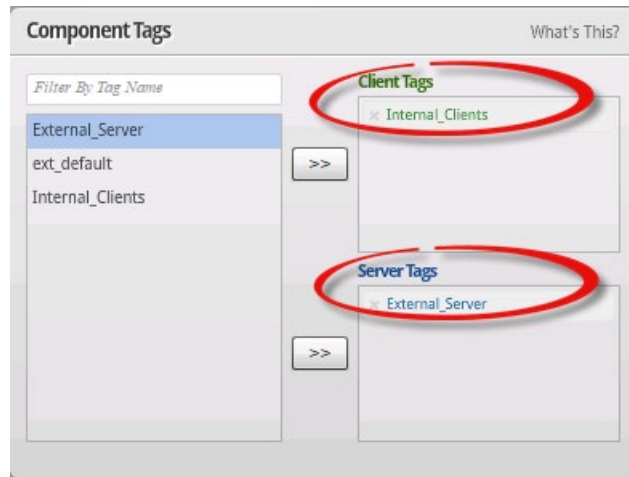
22. Click on the newly created component to edit its parameters:



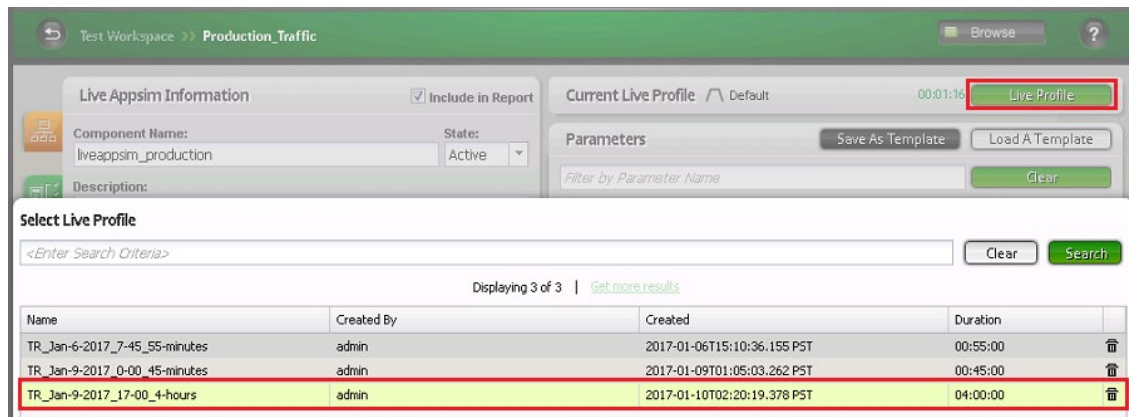
23. In the next steps, we will configure the **Live AppSim** test component:

- a. In the **Component Tags** section make sure to assign the proper interface tags.

For the **Client Tags** remove the existing tags and assign the tag corresponding to the internal clients as configured in the Network Neighborhood. For the **Server Tags** remove the existing tags and assign the tag corresponding to the Target, emulating the external servers as configured in the Network Neighborhood:



- b. In the **Current Live Profile** section click on **Live Profile** button and select the previously imported Live Profile

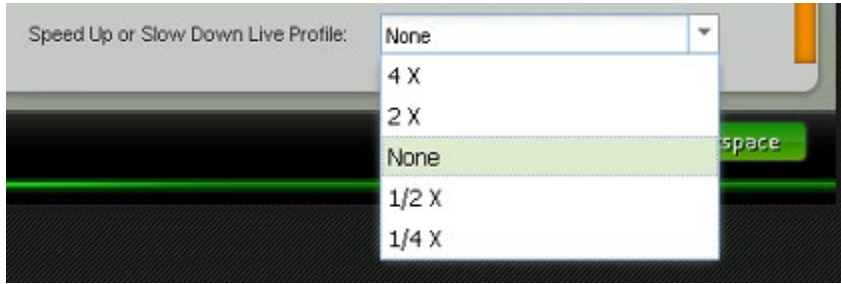


For this test, the remaining parameters can be left to their default values.

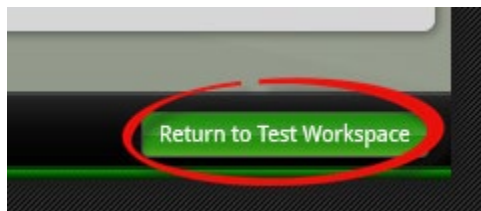
Note: A useful parameter is the **Speed Up or Slow Down Live Profile** option which offers the ability to time scale the test. For example, for traffic profiles with long time periods the test can

Test Methodologies for Regenerating Production Network Traffic

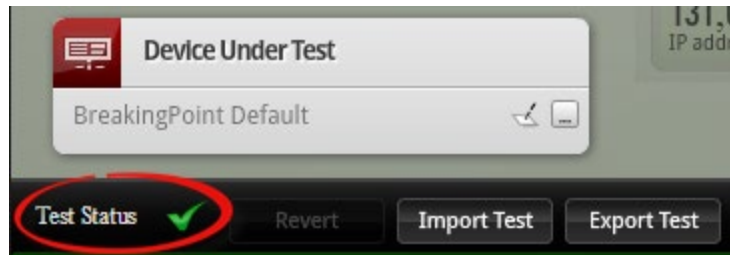
be configured with a **4 x** speed up factor which reduces the total test duration by approximately one fourth of the original profile duration.



- c. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:

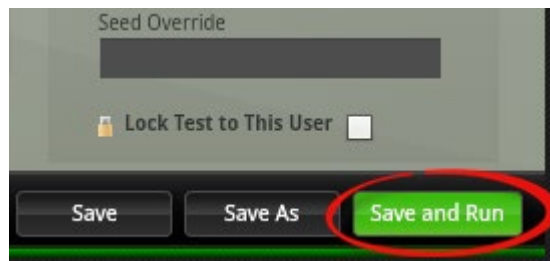


24. Make sure the **Test Status** indicated (on the lower left corner) has a green checkmark:



If there is not, determine what is wrong by selecting **Test Status** and viewing the errors.

25. Select **Save and Run** from the lower right corner:



At this stage, BreakingPoint will start regenerating the production network traffic characteristics based on the imported live profile.

BreakingPoint offers a broad suite of statistics to monitor various KPIs relevant for an extended set of different tests. Refer to the Result Analysis section to analyze the results for this test.

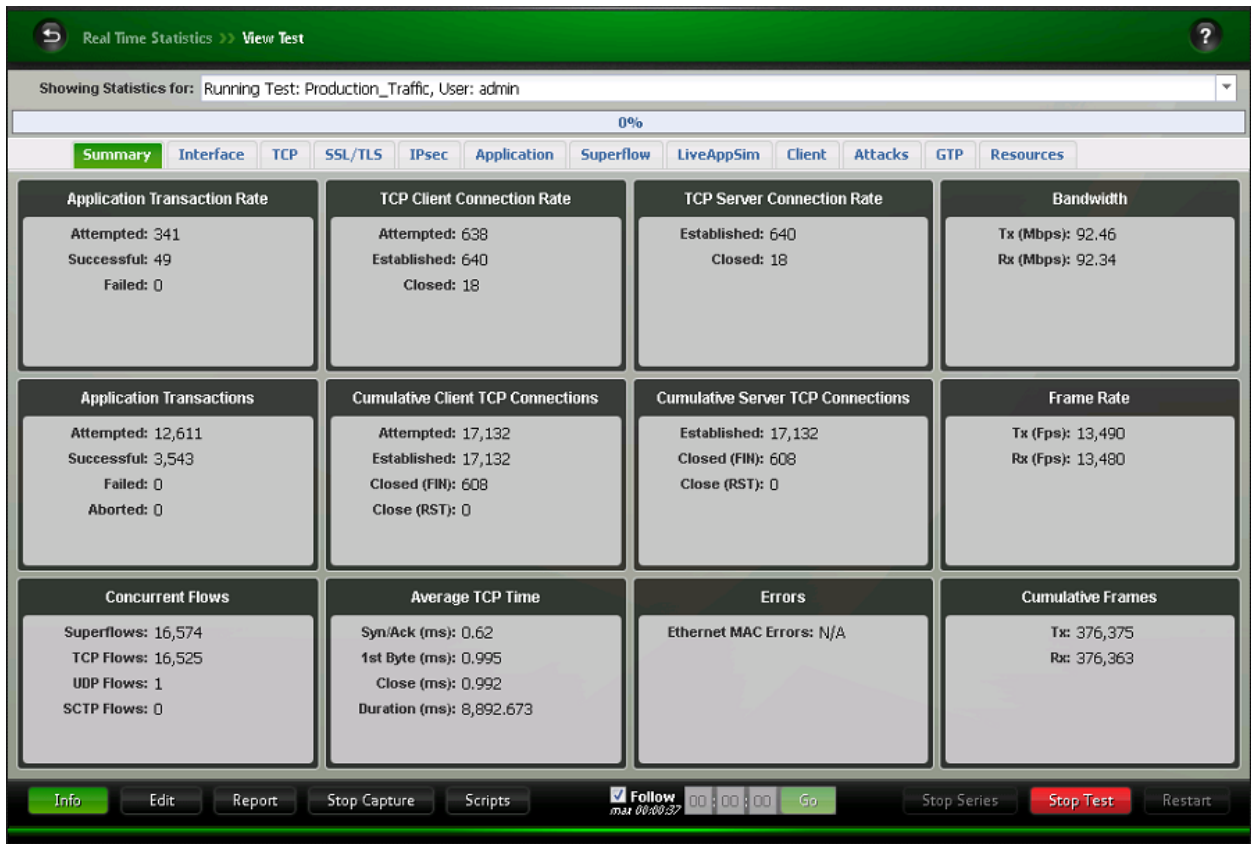
Results Analysis

The main objective of this test is to validate the new network architecture implementation using similar traffic characteristic as the production network. It is very important to read and interpret below statistics in the context of the entire system configuration, hence in accordance to the SUT type, new implementation details, configured policies, etc.

Real-time Statistics

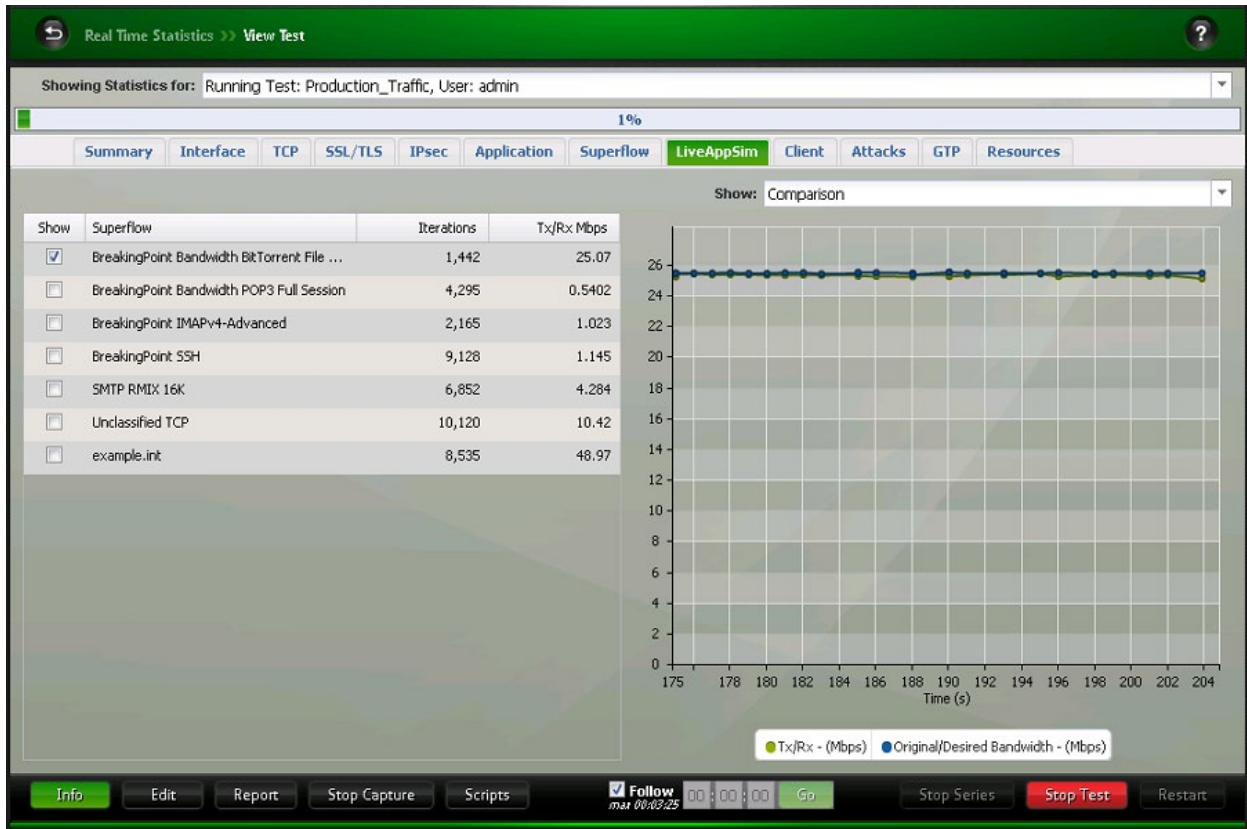
After the test is started and initialized, the screen will automatically switch to Real Time Statistics window.

The **Summary** tab presents basic metrics that provide a good understanding of the overall test progress.



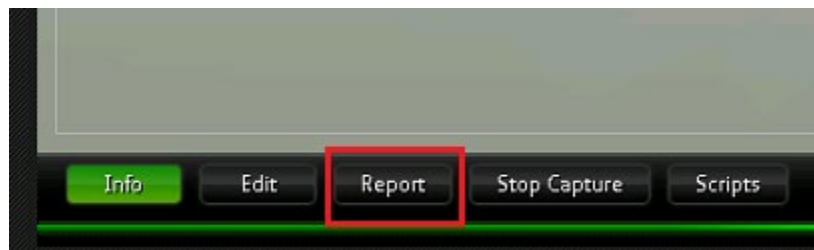
Test Methodologies for Regenerating Production Network Traffic

Another important tab is the dedicated **LiveAppSim** tab: here we can see for each of the applications a comparison graph with the test achieved throughput vs the original observed one.



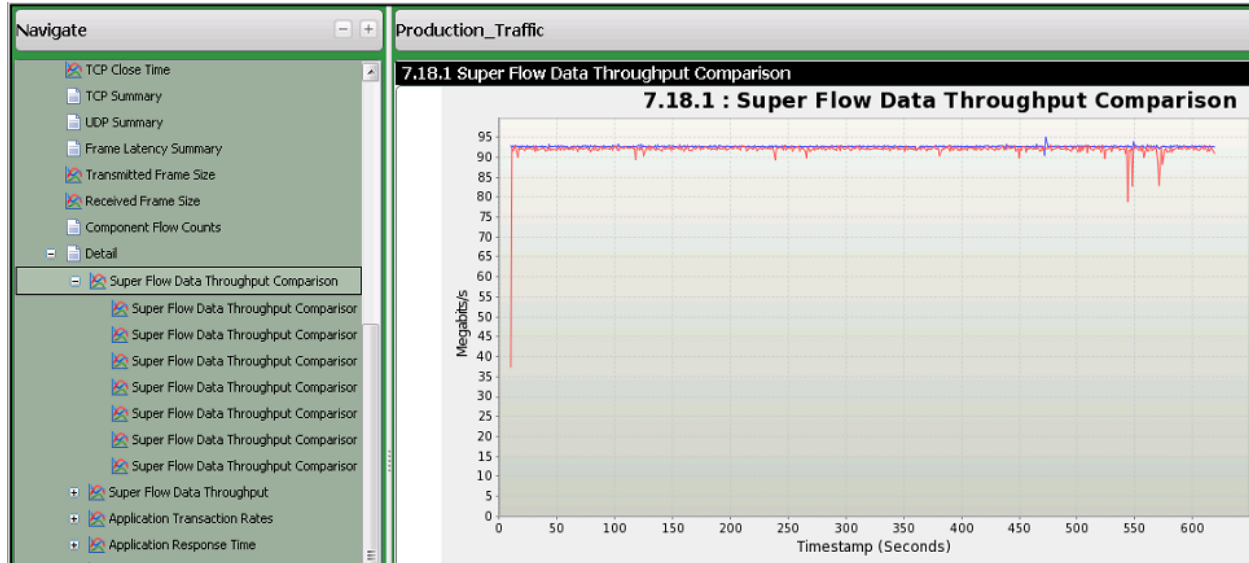
Monitoring the original (similar to the production network) vs current test achieved throughput on a per application basis offers users the ability to promptly identify potential issues with the new network architecture implementation before the deployment it in the real production environment.

Beside the real-time statistics, detailed post-test analysis can be performed. In the lower left corner of the Real Time Statistics window, click on the **Report** button to view detailed results.



Test Methodologies for Regenerating Production Network Traffic

This will open the results in a new browser window. On the left side of the detailed report window is the navigation pane, where you can navigate and browse the results. The results and test information will be displayed on the right side of the browser:



Detailed statistics can be investigated based on each test component, TCP/UDP traffic type, transmit vs. receive traffic, latency values, etc.

Test Variables

BreakingPoint offers the following test configuration parameters that provides the flexibility to simulate a high number of different tests with various degrees of complexity and stress levels to assess the new network architecture implementation capabilities and the impact over the legitimate traffic profile that the solution would experience in a production network.

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
IP Version	IPv4	IPv6, DHCP, etc. to test various access technologies
Network Security Attacks	None	Aside from the production-like network traffic, additional security strikes (malware, exploits, DDoS) can be layered on to assess the new network implementation's ability to protect against such attacks and their impact on the legitimate traffic.

Test Methodologies for Regenerating Production Network Traffic

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
Incremental Traffic	None	Other custom application mixes that potentially matches the estimated future traffic growth from the deployment production for proper future sizing

Conclusions

In this test case, we have been through the configuration steps of the TrafficREWIND solution and its supporting elements: ATIP and BreakingPoint.

This test methodology demonstrates how the innovative TrafficREWIND solution opens unprecedented capabilities in terms of pre-deployment validation with production-like application mixes improving productivity and increasing confidence for upcoming live production changes

Test Methodologies for Encrypted Traffic Inspection

Encryption is an incredibly powerful tool in the cyber-security arsenal, providing a strong additional layer of security for sensitive data. Sandvine¹⁴ predicts 70% of the global internet traffic will be encrypted by the end of 2016 and the trend will continue to grow.

Increased encryption adoption has also been driven by the growing availability of free Certificate Authorities, and by large hosting providers such as CloudFlare and Amazon moving to SSL.

This is an encouraging development, as encryption protects data in transit from interception and snooping by prying eyes. It makes it difficult for would-be cybercriminals to harvest information from, say, the contents of corporate emails. However, there is a darker side to SSL encryption: it can also mask malicious content, such as malware. A 2016 Ponemon Study¹⁵ found that nearly half of organizations in a range of industry sectors that had suffered a cyberattack, subsequently found that their attackers used SSL encrypted data traffic to conceal malware and smuggle it onto corporate networks.

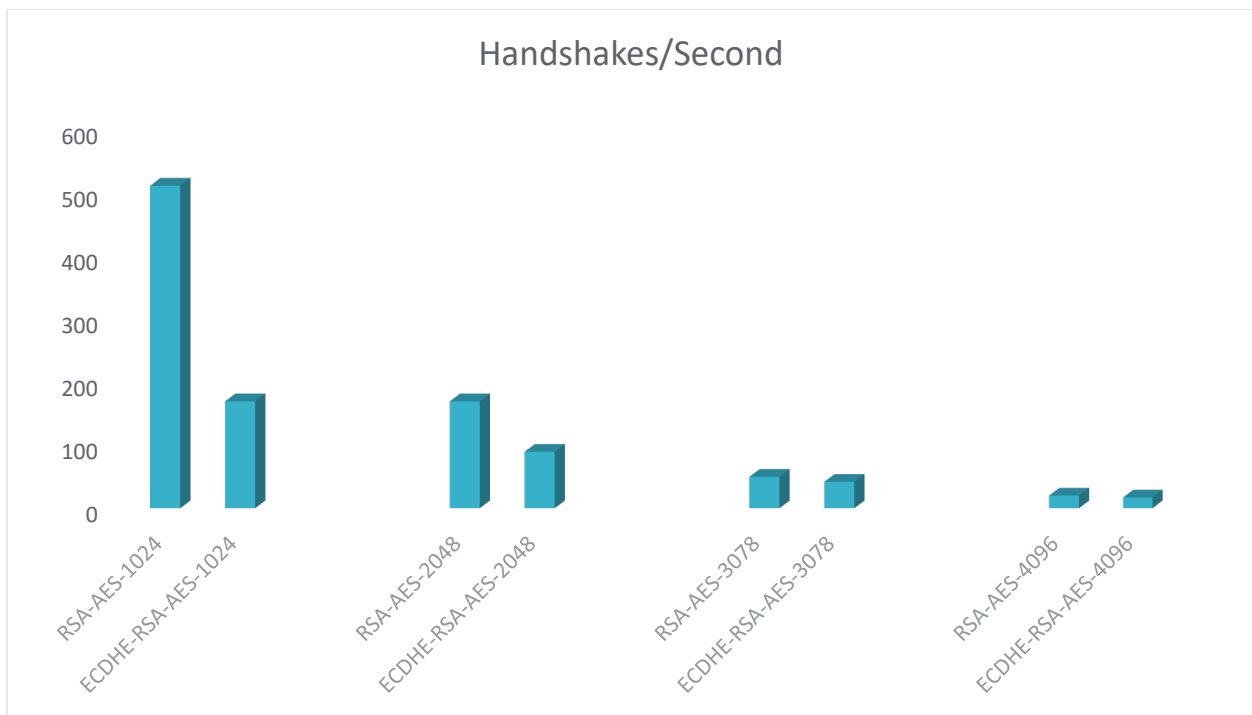
¹⁴ <https://www.sandvine.com/pr/2016/2/11/sandvine-70-of-global-internet-traffic-will-be-encrypted-in-2016.html>

¹⁵ https://info.a10networks.com/SSLi_WW_Content_ExecutiveSummary_6CriticalDiscoveries_TYPage.html?_ga=1.51542185.1233164609.1472655516

Test Case: Measuring Encrypted Traffic Inspection Capabilities of SSL Proxy Devices

Overview

The internet is slowly moving towards 100% encryption. Most websites have implemented encryption and others are following suite. People are being consistently educated about the benefits of SSL which has also contributed to them staying away from plain text sites. Similarly, streaming services have also realized the value of protecting people’s privacy and are also rapidly moving towards encryption. With all the good things that have been brought through encryption, this has also become an effective tool for the hackers and criminals to perpetrate their malicious intent in plain sight. This poses additional challenges for the security tools, as they now need to dig deeper to extract the data hidden within the secured ciphers. SSL inspection for a device in the middle generally comes at a cost, most encryption are very resource intensive and when inspection is enabled can cause significant increase in latencies and packet drops to the extend where people may consider turning them off. Similarly, several devices publicize their SSL performance numbers with weaker or out-of-date encryption algorithms and weaker key sizes. However as shown in the picture below there is almost an exponential drop in performance as we move to larger key sizes and stronger encryptions.



Objective

This test measures the security effectiveness of network-based SSL proxy devices against encrypted traffic. As part of the test we will be using one configuration and run CPS test to measure max SSL handshakes with RSA_AES256_SHA384 with 2K key size. Look at the variable section to get more details on other relevant ciphers.

Setup

The setup requires at least two test ports – one acting as a client side of the proxy and the other as the server end of the proxy with the device under test in the middle.

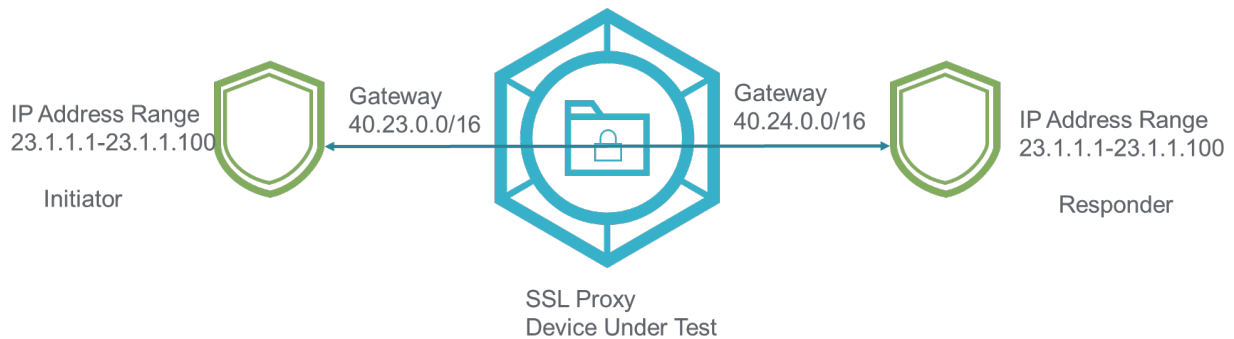
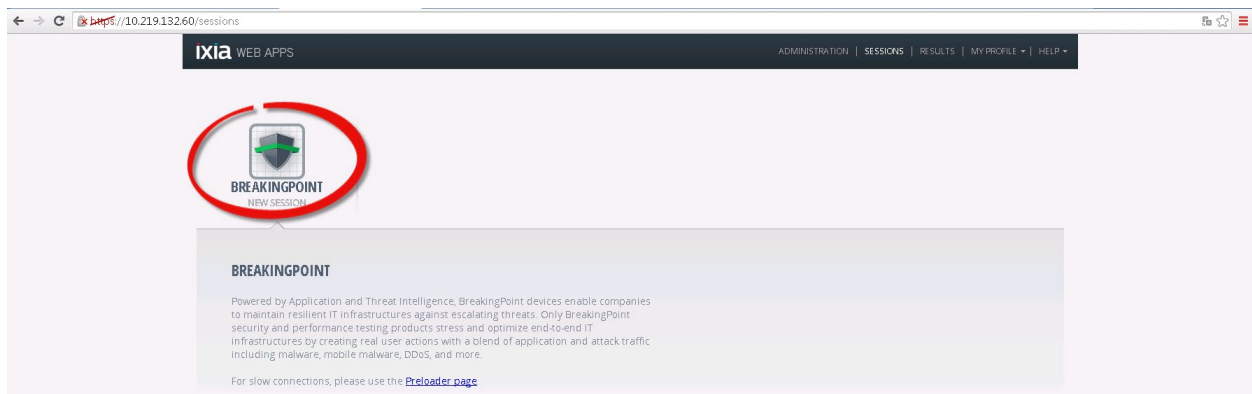


Figure 41. Test Setup

Configure the policy of the device to allow both inbound and outbound communication for any traffic/protocol on any port (allow ANY to ANY). Configure the SSL Proxy in the device to ensure it has allowed the traffic to pass through. Configure the SSL proxy within the device to have the similar keys and certs to be able to terminate, inspect and reinitiate the SSL traffic.

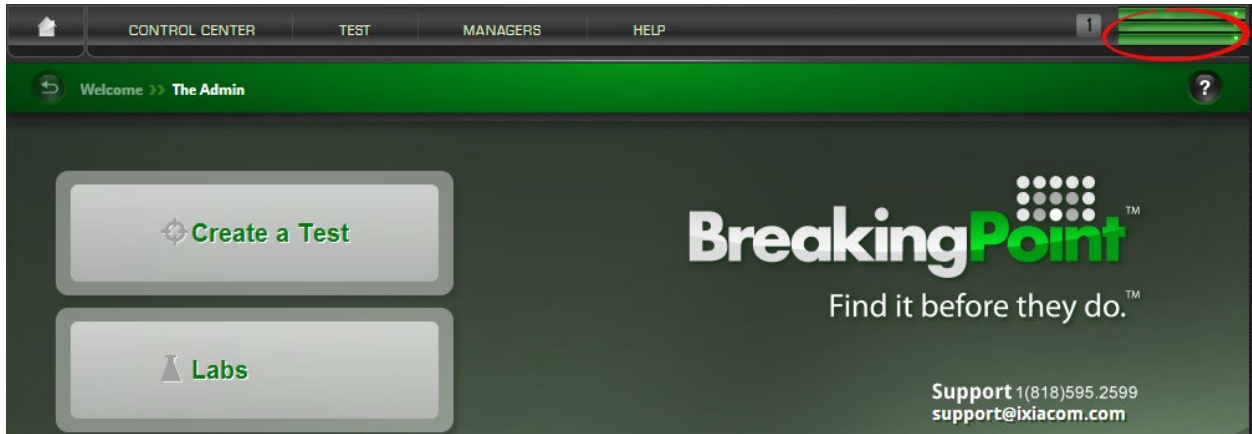
Step-by-Step Instructions

1. Use a web browser connected to the Ixia Web Apps GUI and start a new BreakingPoint Web Session:

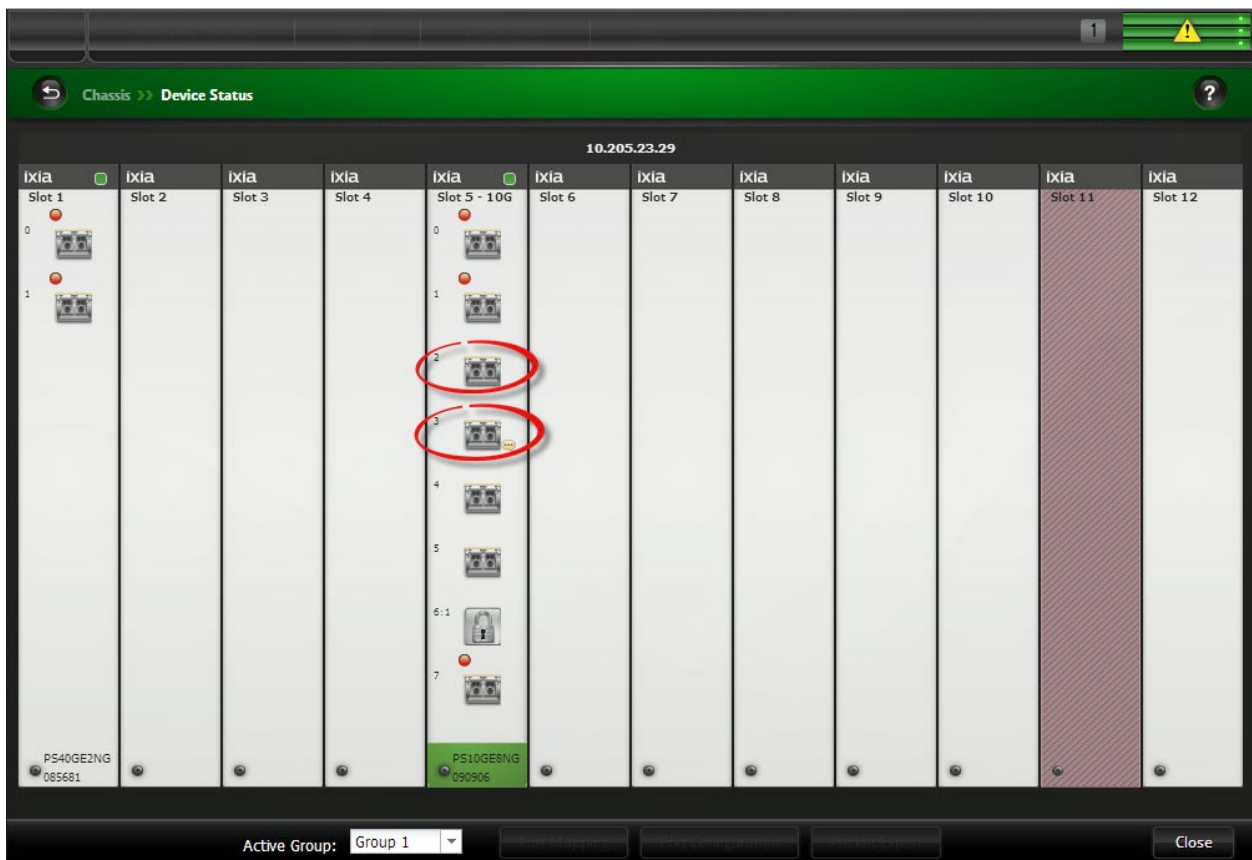


Test Methodologies for Encrypted Traffic Inspection

2. Once the BreakingPoint web home page loads, reserve the test ports to be used in the physical test setup to generate/receive traffic:
 - a. Click on the Device Status button located on the upper right corner:



- b. In the new screen select the physical ports that are to be used in the test:



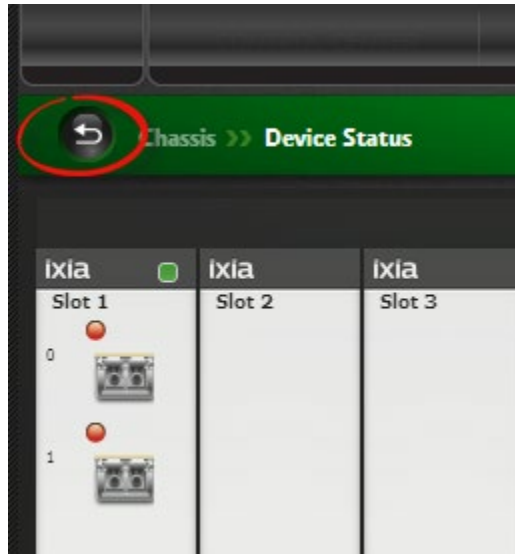
In this example, we will use ports 2 and 3 from the blade located in slot number 5.

Note: An important thing to remember when reserving your ports is the order in which you reserve them. Whenever you reserve a port, the system will automatically map that port to an interface on the chassis.

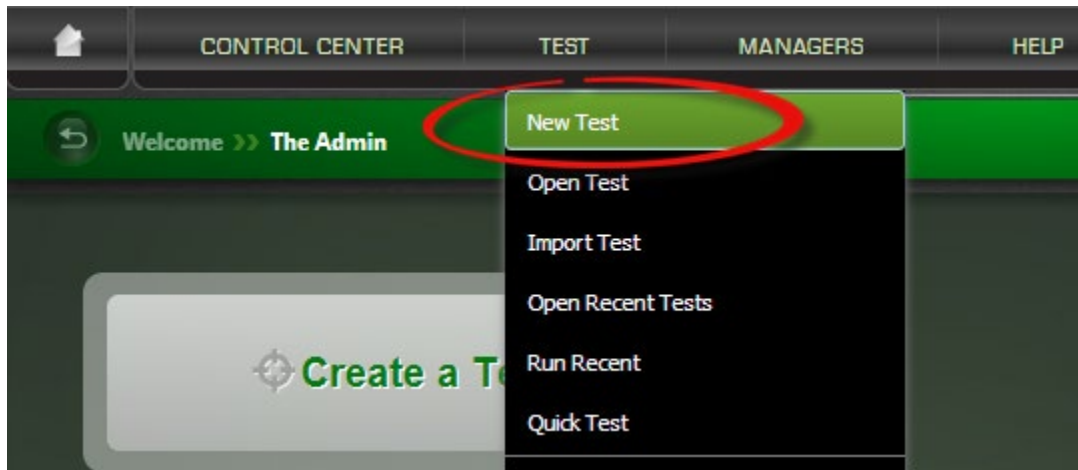
Test Methodologies for Encrypted Traffic Inspection

Note: The resources of each blade are allocated in proportion to the number of ports reserved on the blade. In some cases, it may be necessary to reserve additional ports to secure enough resources for the test being performed. The sessions and bandwidth available to a test may be insufficient to adequately perform the test if too few ports have been reserved.

- c. Once the proper test ports have been selected click on the back arrow to return to the initial screen:



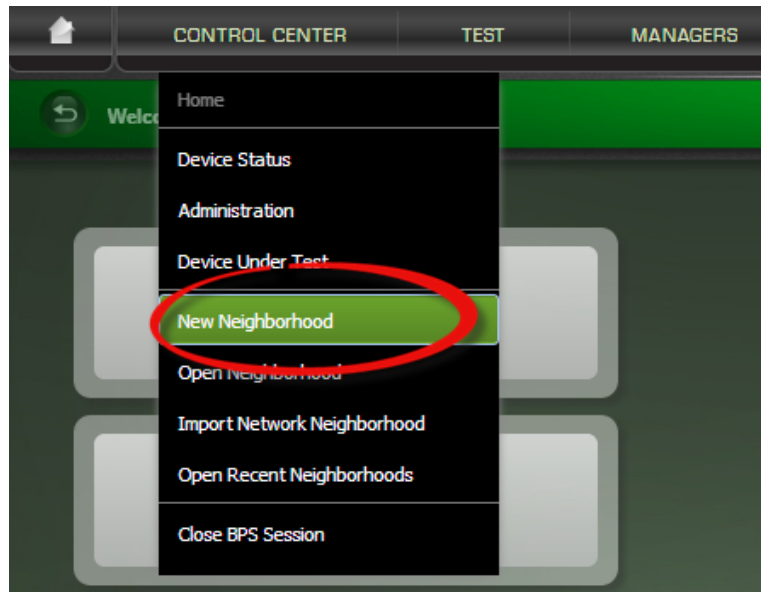
3. Next, select **Test** -> **New Test** option from the upper menu bar to start with configuring the test:



Since we already reserved the physical test ports (and, if applicable extra ports for more resource reservation) now we need to configure the MAC and IP layer parameters like MAC address VLANs, IP addressing scheme, MTU, etc. All these settings, among others are controlled/defined from the **Network Neighborhood**:

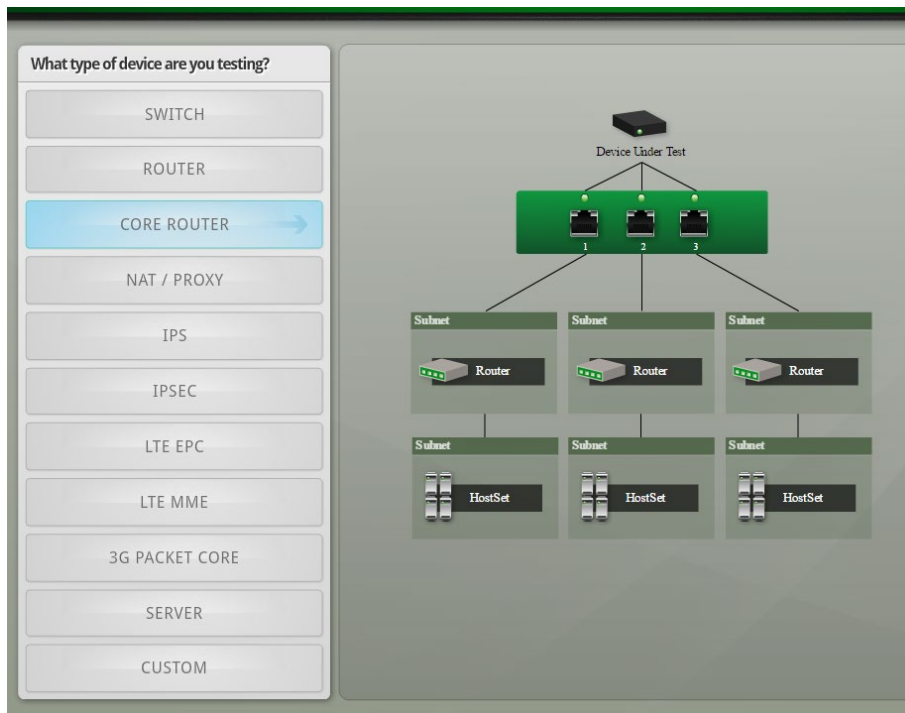
Test Methodologies for Encrypted Traffic Inspection

- a. From the upper menu bar select **Control Center** -> **New Neighborhood**:



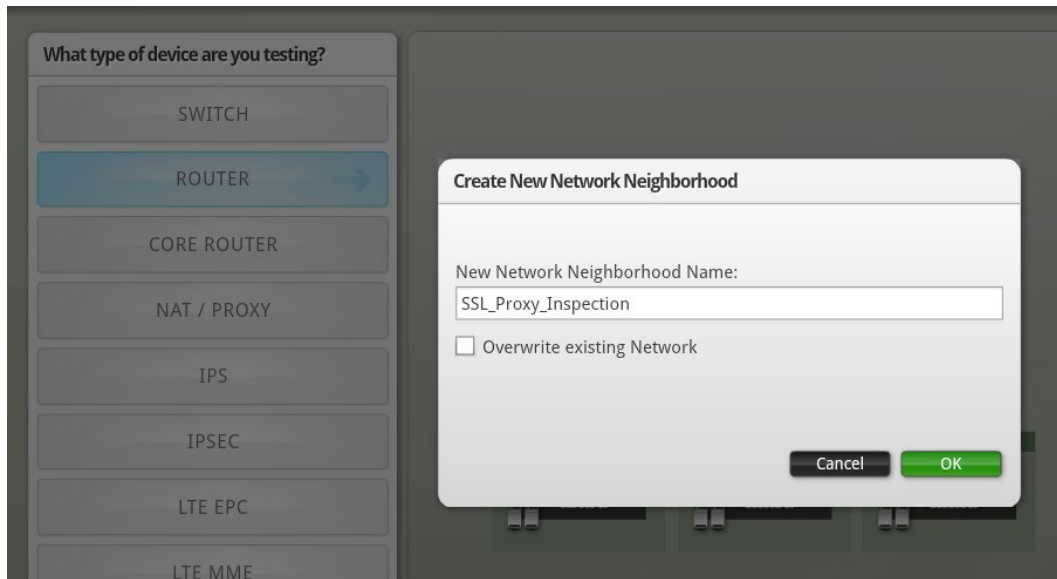
- b. Select the DUT type used for the test and a corresponding default Network Neighborhood will be created:

For this test, we will use *CORE ROUTER* as DUT type.



Test Methodologies for Encrypted Traffic Inspection

- c. Enter an easy-to-remember name for the new Network Neighborhood and click **OK**:



- d. According to the DUT type we previously selected, a default Network Neighborhood will be created. Since we chose *ROUTER* DUT type, the following elements are automatically created:
- i. Two INTERFACES: Provides access to interface level parameters like MTU, MAC, Impairments, and Packet Filters.
 - ii. One IPv4 EXTERNAL HOSTS: configures the IP address of the external end hosts/servers. This element is not required for IPS devices that are Pass-Through. It is only need for devices that are terminating the TCP Connection (e.g. Server Load Balancers).
 - iii. Two IPv4 STATIC HOSTS: Provides access to IP related parameters of the BreakingPoint emulated hosts. The Static Hosts represents the BreakingPoint emulated attackers and target servers.
 - iv. Two IPv4 ROUTERS provides flexibility to route multiple subnets behind it. Set the ID as ClientSideGateway, container as Interface1 and IP address as 40.23.0.2 and gateway as 40.23.0.254 and Netmask as 16.

Test Methodologies for Encrypted Traffic Inspection

- v. Two IPv4 ROUTERS provides flexibility to route multiple subnets behind it. Set the ID as ServerSideGateway, container as Interface2 and IP address as 40.24.0.2 and gateway as 40.24.0.254 and Netmask as 16.

The screenshot shows a network configuration interface with three main sections:

- INTERFACE: (2)** Untagged Virtual Interface

DEL	ID	Number	MTU	Use vNIC MAC Address	MAC Address	Duplicate MAC Address	VLAN Key	Ignore Pause Frames	Descr
x	Interface 1	1	1500	<input checked="" type="checkbox"/>	02:1A:C5:01:00:00	<input checked="" type="checkbox"/>	Outer VLAN	<input type="checkbox"/>	
x	Interface 2	2	1500	<input checked="" type="checkbox"/>	02:1A:C5:02:00:00	<input checked="" type="checkbox"/>	Outer VLAN	<input type="checkbox"/>	
- IPV4 ROUTER: (2)** Simulated IPv4 router

DEL	ID	Container	IP Address	Gateway IP Address	Netmask
x	ClientSideGateway	Interface 1	40.23.0.2	40.23.0.254	16
x	ServerSideGateway	Interface 2	40.24.0.2	40.24.0.254	16
- IPV4 STATIC HOSTS: (2)** Simulated IPv4 endpoints

- vi. For this test, under IPv4 static hosts we will use the following:
- Change the first entry (with ID as Static Hosts i1_default) to have a tag of “Trusted”, Base IP Address as 23.1.1.1, Count of 100 and Container being ClientSideGateway.
 - Change the first entry (with ID as Static Hosts i1_default) to have a tag of “Trusted”, Base IP Address as 24.1.200.1, Count of 100 and Container being ServerSideGateway.

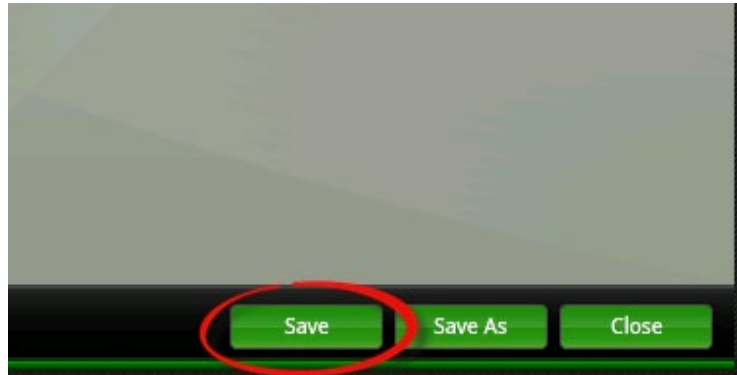
The screenshot shows the updated network configuration interface, specifically the IPV4 STATIC HOSTS section:

- IPV4 STATIC HOSTS: (2)** Simulated IPv4 endpoints

DEL	ID	Container	Tags	Base IP Address	Count	Gateway IP Address	Netr
x	Static Hosts i1_default	ClientSideGateway	Client_Side_Network	23.1.1.1	100		16
x	Static Hosts i2_default	ServerSideGateway	Server_Side_Network	24.1.200.1	100		16

Test Methodologies for Encrypted Traffic Inspection

- e. Save the new Network Neighborhood configuration by clicking the **Save** button located on the lower right corner:



- f. Click on the back arrow to return to the main test screen:

A screenshot of the 'Network Neighborhood' configuration interface. The title bar shows 'Network Neighborhood >> SSL_Proxy_Content_Inspection'. A back arrow icon is circled in red. Below the title bar, there are tabs for 'Entry Mode' and 'Diagram Mode', and a green 'ADD NEW ELEMENT' button. The main area contains three expandable sections: 'INTERFACE: (2)', 'IPV4 ROUTER: (2)', and 'IPV4 STATIC HOSTS: (2)'. Each section contains a table of configuration parameters.

INTERFACE: (2) Untagged Virtual Interface							
DEL	ID	Number	MTU	Use vNIC MAC Address	MAC Address	Duplicate MAC Address	VLAN
x	Interface 1	1	1500	<input checked="" type="checkbox"/>	02:1A:C5:01:00:00	<input checked="" type="checkbox"/>	Oute
x	Interface 2	2	1500	<input checked="" type="checkbox"/>	02:1A:C5:02:00:00	<input checked="" type="checkbox"/>	Oute

IPV4 ROUTER: (2) Simulated IPv4 router					
DEL	ID	Container	IP Address	Gateway IP Address	Netmask
x	ClientSideGateway	Interface 1	40.23.0.2	40.23.0.254	16
x	ServerSideGateway	Interface 2	40.24.0.2	40.24.0.254	16

IPV4 STATIC HOSTS: (2) Simulated IPv4 endpoints					
DEL	ID	Container	Tags	Base IP Address	Count
x	Static Hosts i1_default	ClientSideGateway	Client_Side_Network	23.1.1.1	100
x	Static Hosts i2_default	ServerSideGateway	Server_Side_Network	24.1.200.1	100

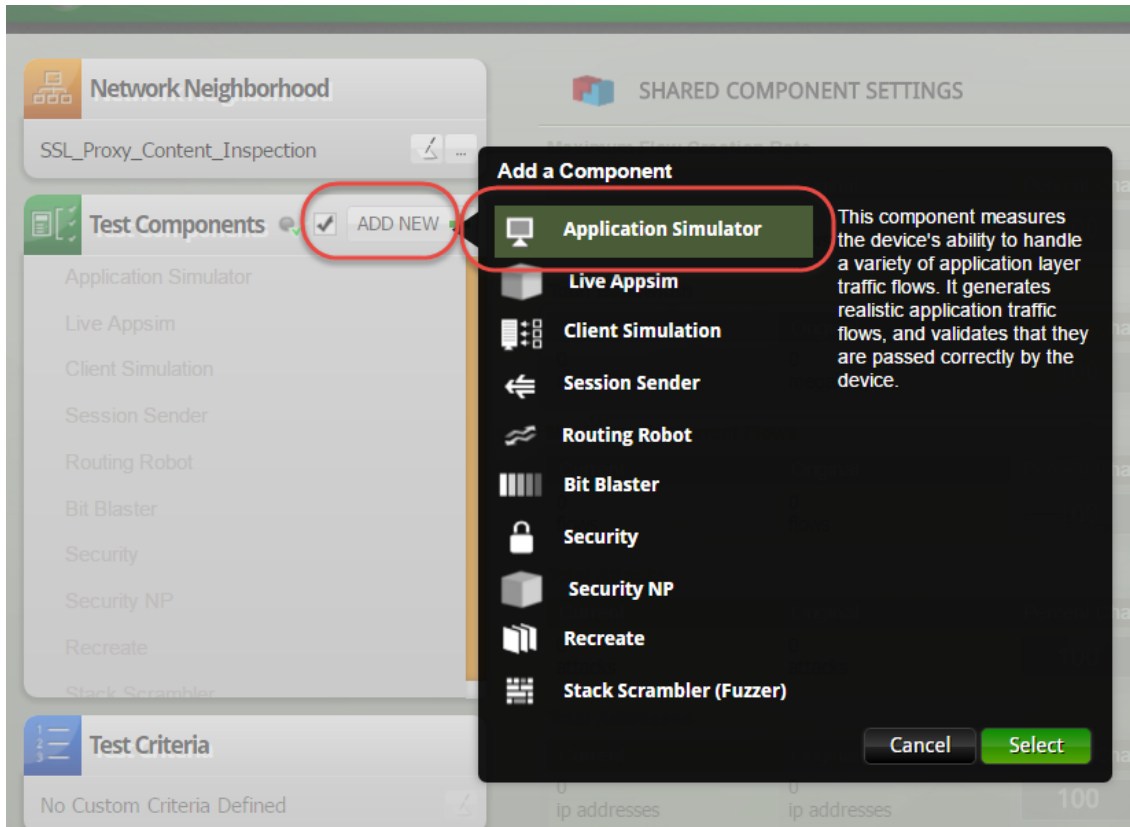
- g. From the main test screen click on the browse button from the **Network Neighborhood** section to search and select the above created Network Neighborhood:

After configuring the MAC and IP layer parameters the actual traffic component needs to be created.

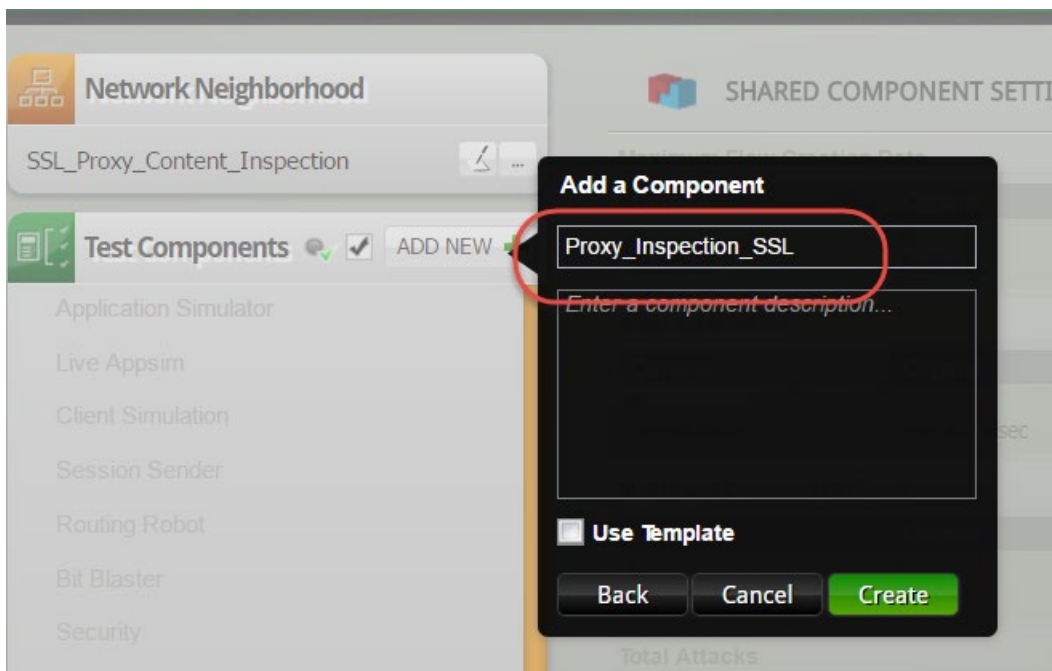
The Test Component is the entity that controls the traffic patterns. For this example, we will use the Security test component to emulate the actual attacks.

Test Methodologies for Encrypted Traffic Inspection

4. Click on the **ADD NEW** button from the **Test Components** section. Choose *Application Security* from the selection list and click on **Select** button:

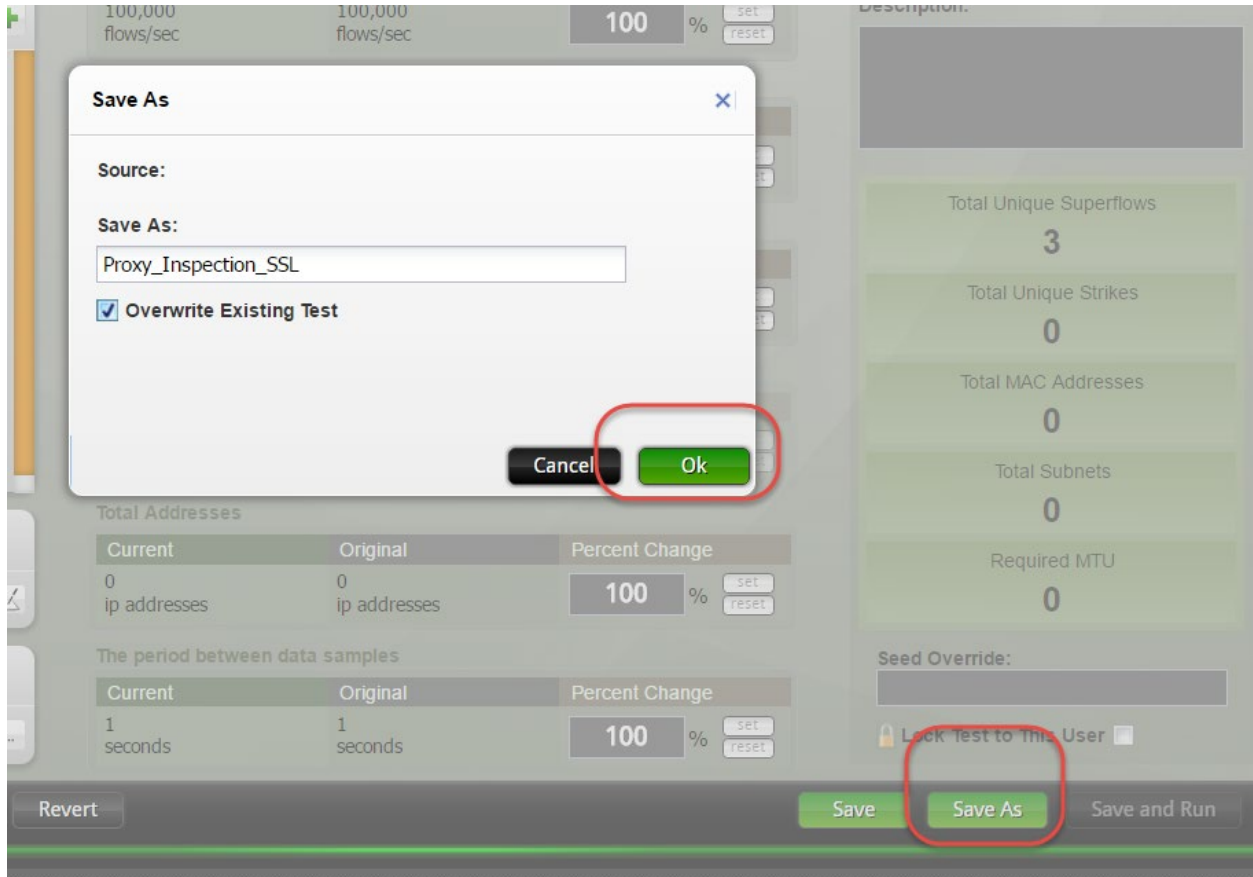


5. Rename the component from *Application Simulator* to something more meaningful if required and click Create button:

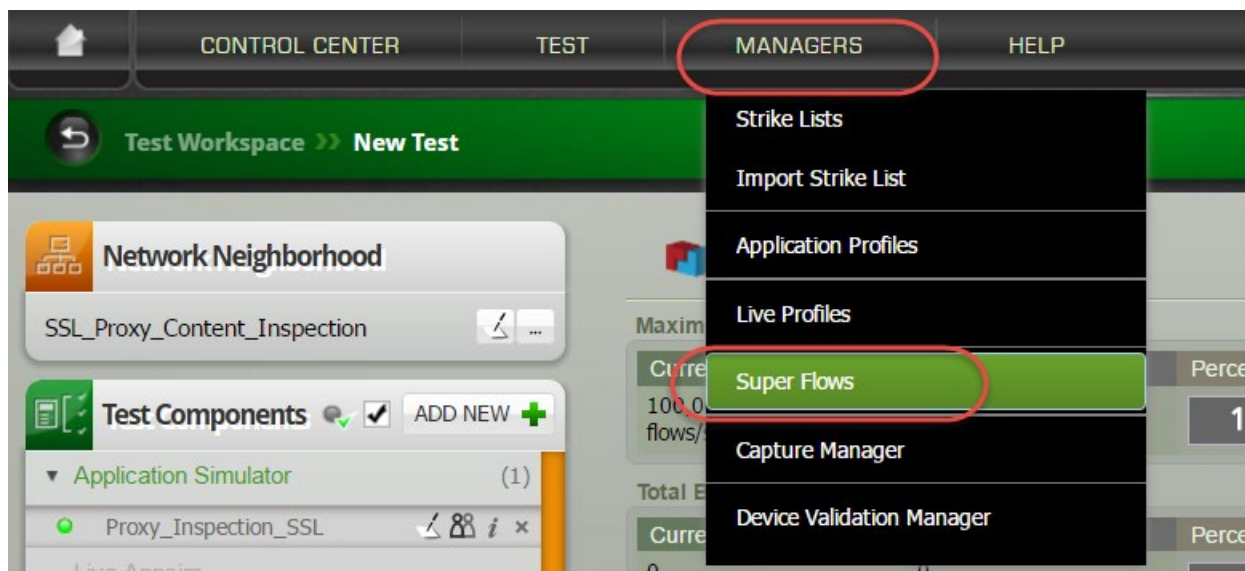


Test Methodologies for Encrypted Traffic Inspection

A new entry (i.e. *Critical Strikes*) will be created under **Application Simulator** Test Component. It's a good idea to save the test at this point with a proper name using the "Save As" button.

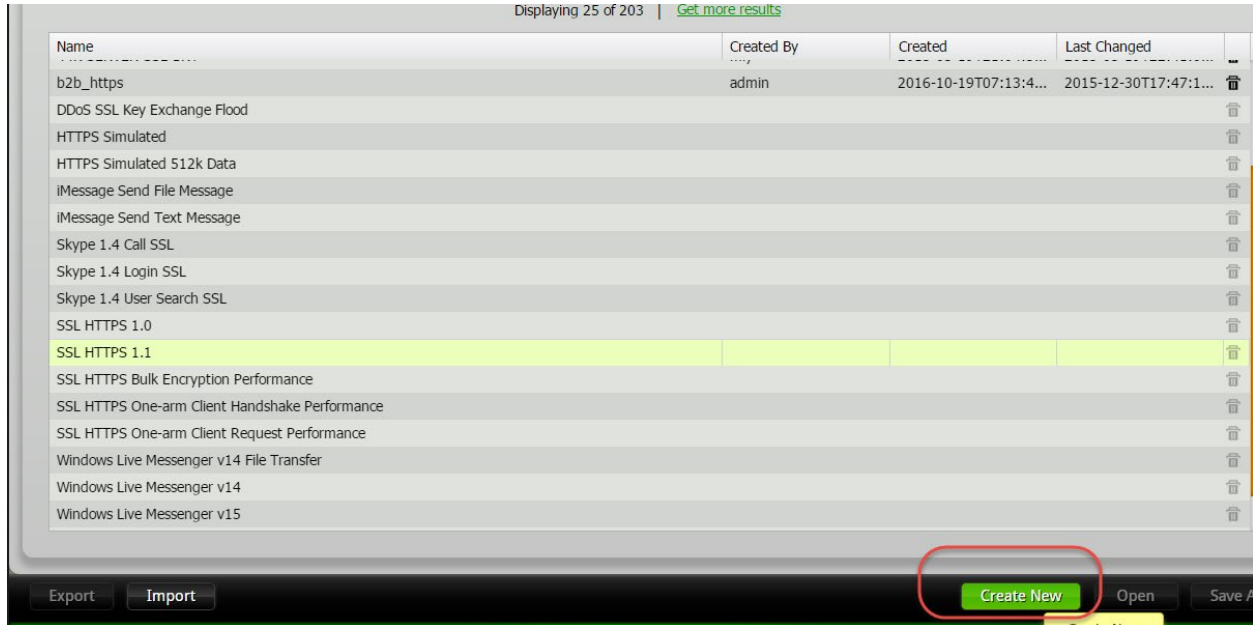


6. Once done, let's move to the Superflows to create our SSL Superflow from Managers -> Superflows. If the test need

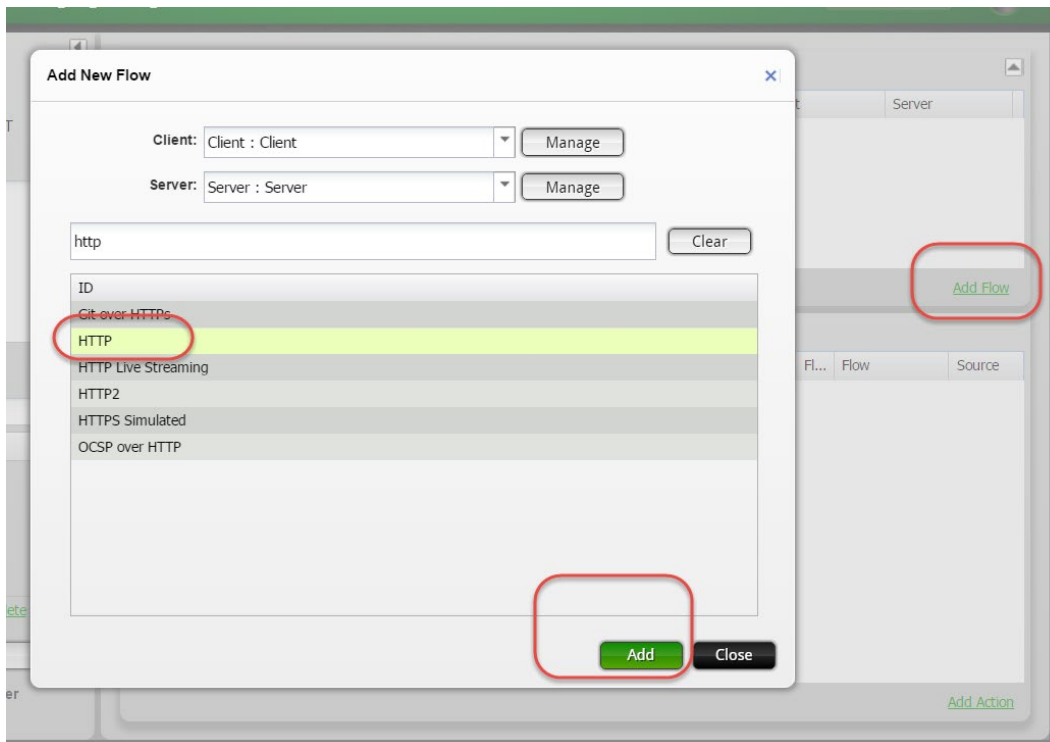


Test Methodologies for Encrypted Traffic Inspection

- It's always advisable to start from an existing sample Superflow from the examples we already have as there would be some tests that will have the ciphers. For this particular test, we will create everything ground up. Click on Create New to create a new super flow with a proper name.

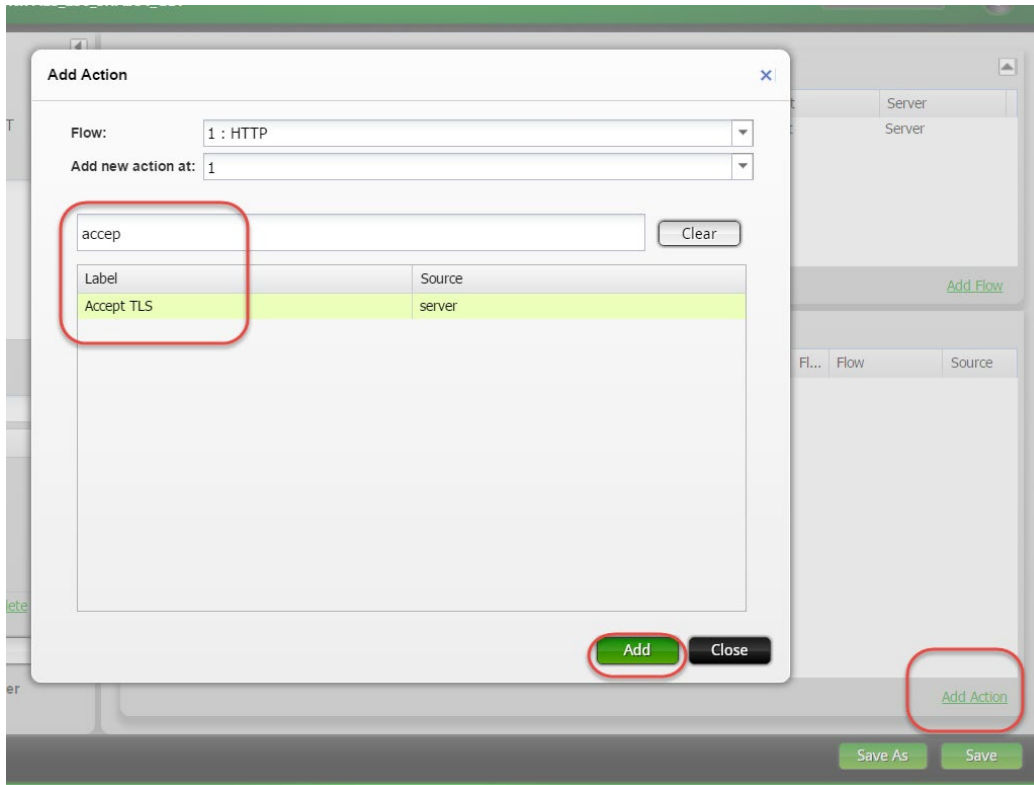


- In the super flow select the “Add Flow”, search for HTTP and select the HTTP flow.

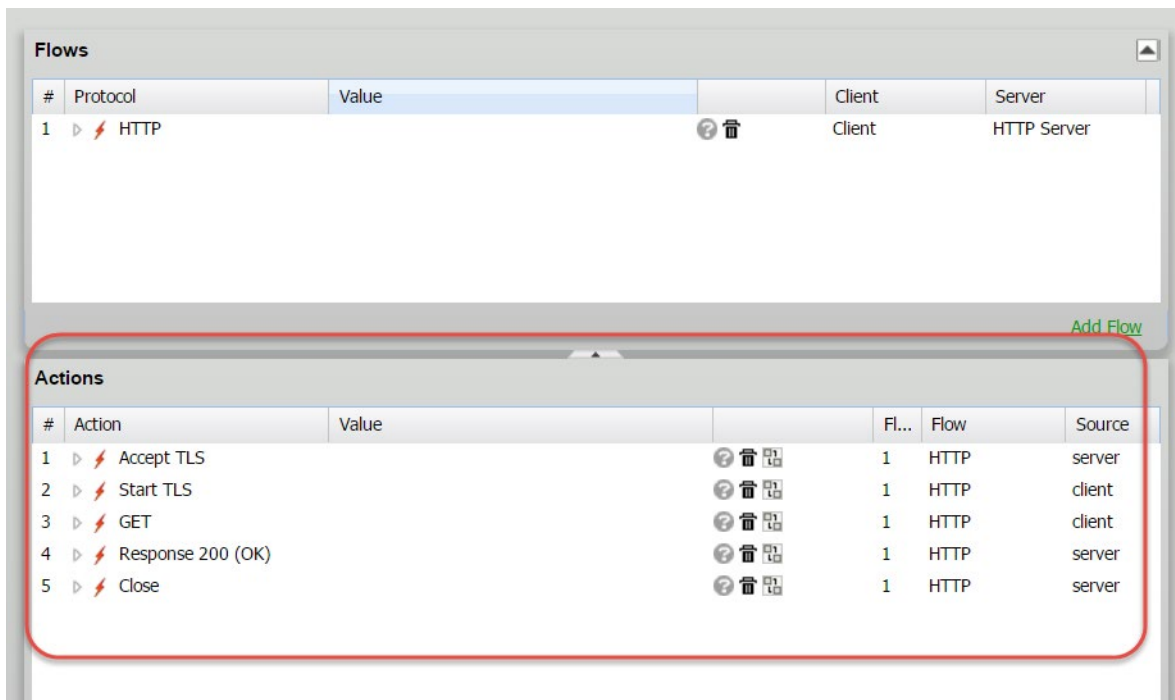


Test Methodologies for Encrypted Traffic Inspection

9. Add the following actions to “Accept TLS”, “Send TLS”, “GET”, “Response 200 OK”, “Close”. Search for each of the commands in the search bar and add them one by one.

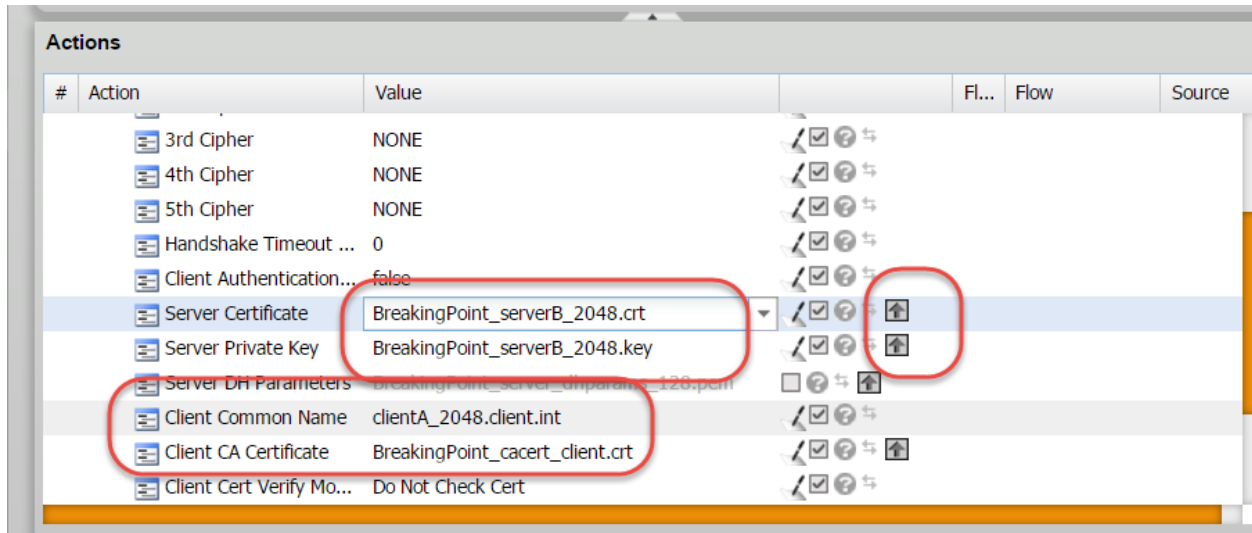


10. The flow should now look something like this

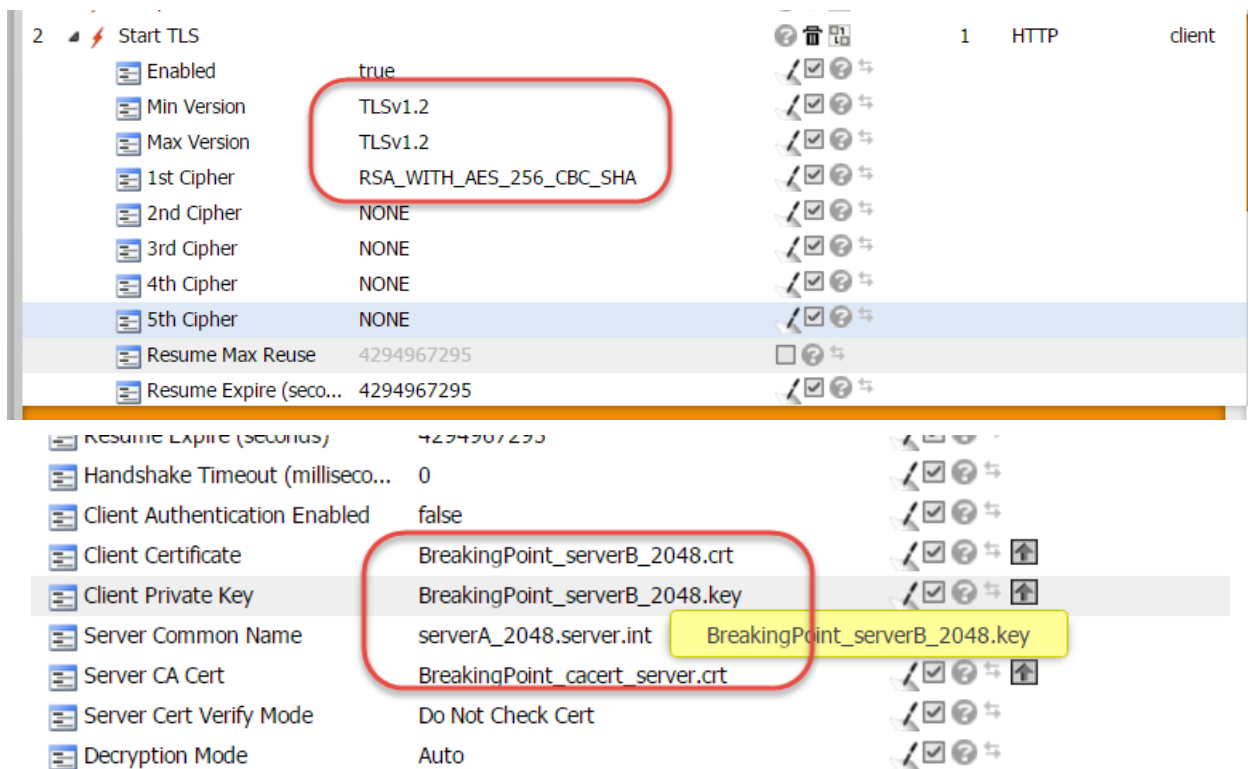


Test Methodologies for Encrypted Traffic Inspection

upload your own server and client certificate using the upload arrow at the right as long as the same cert and key this should not be a problem.

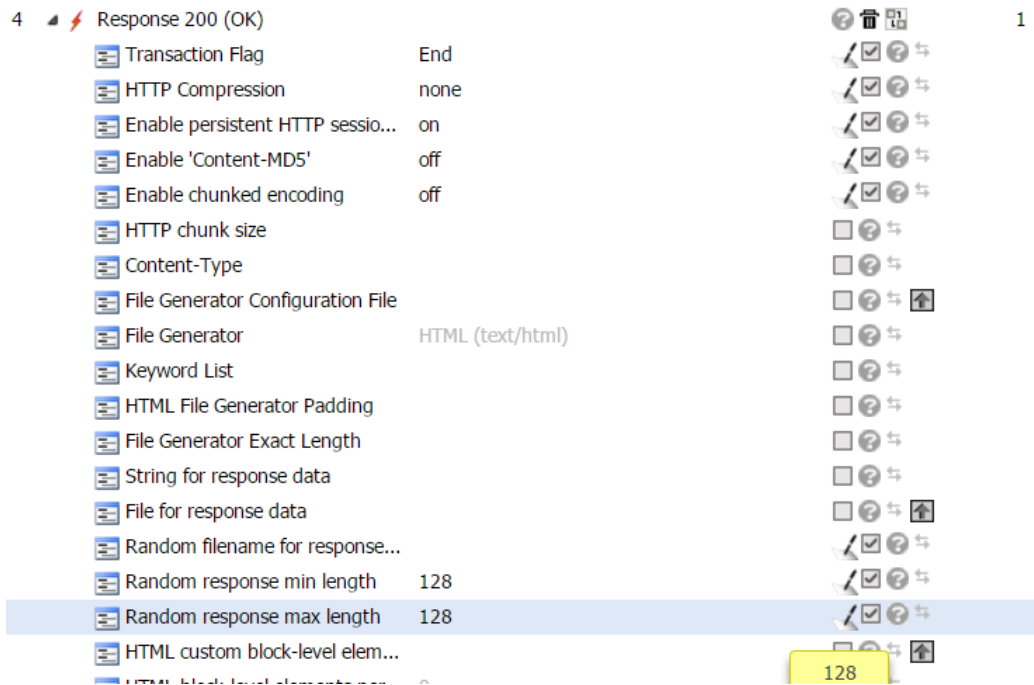


- Repeat the same for the **Start TLS** at the client side. Ensure that they are using similar TLS versions and similar key size and cert. If there's a server CA being used ensure the server CA cert matches that of the DUT. As mentioned earlier, as long as the DUT's **private key and cert** are same and have a similar CA cert than the inspection will happen successfully.

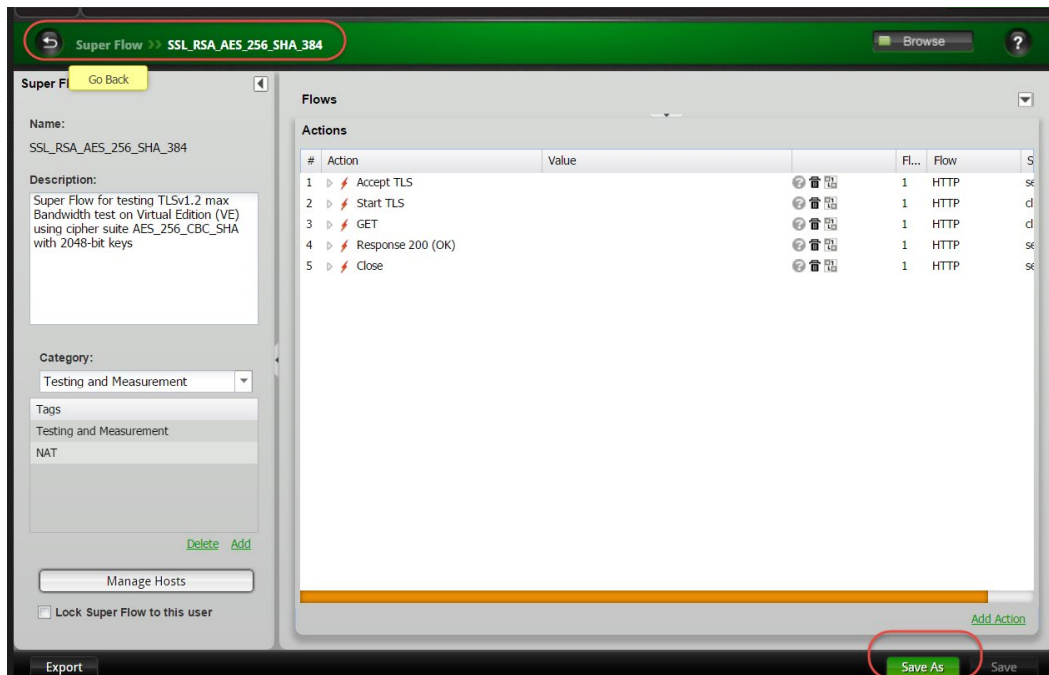


Test Methodologies for Encrypted Traffic Inspection

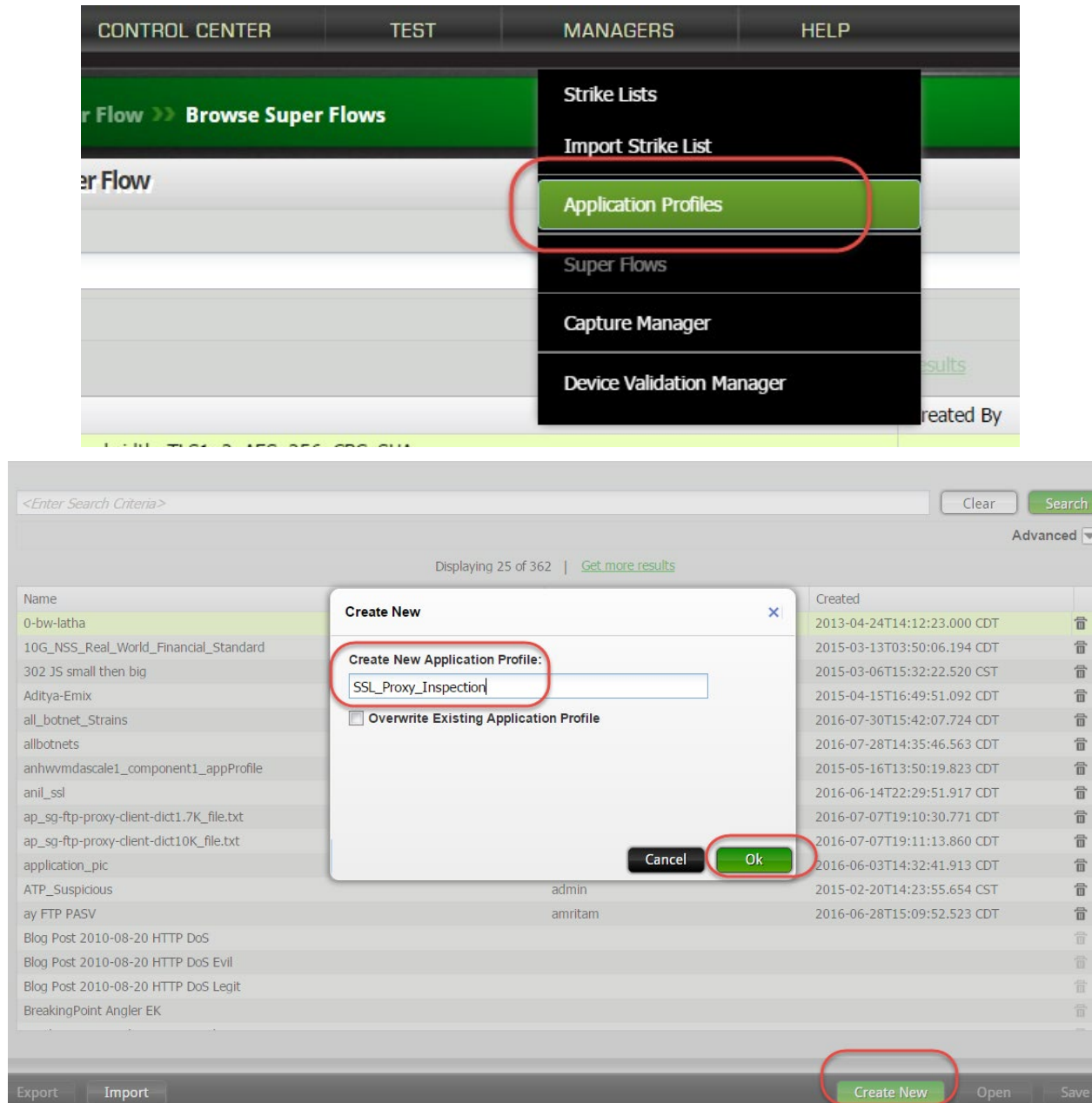
14. The first test would be a small packet, high session per second test so set the “Response” page should be smaller. To achieve that set the minimum and maximum page size as 128 bytes.



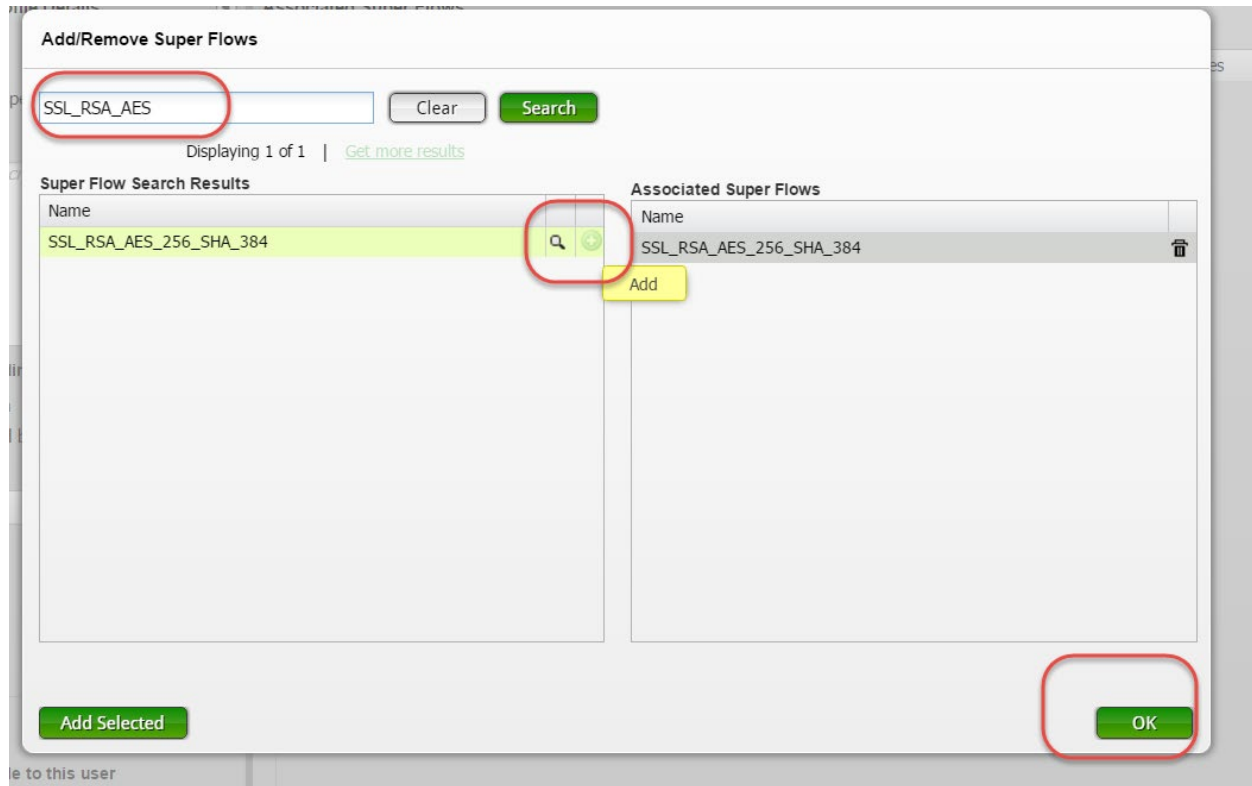
15. Save the Superflow with a name that can be remembered easily and go back to the manger.



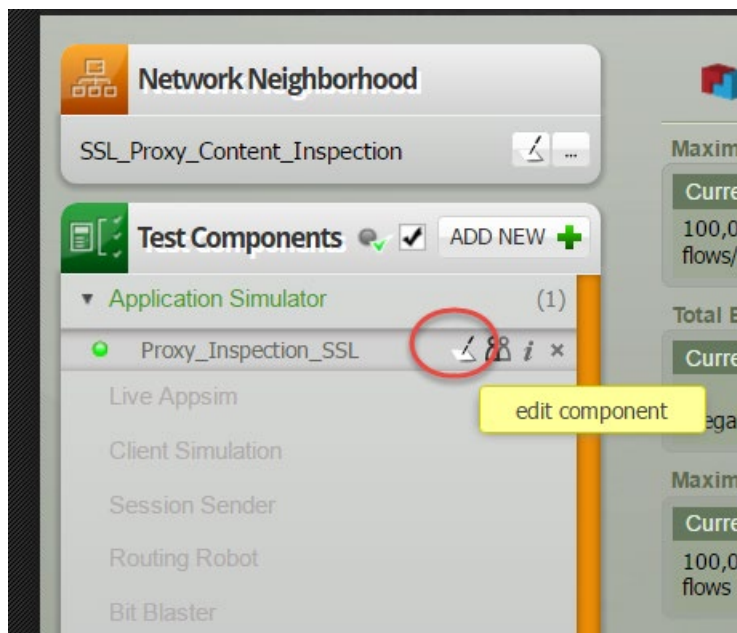
16. Go to the “**Application Profiles**”, and create new application



17. Search for the newly created Superflow and add the same.



18. Save the Application Simulator and go back to the saved test and open the Application Simulator component that we had previously created. Click on the edit button to edit the same.

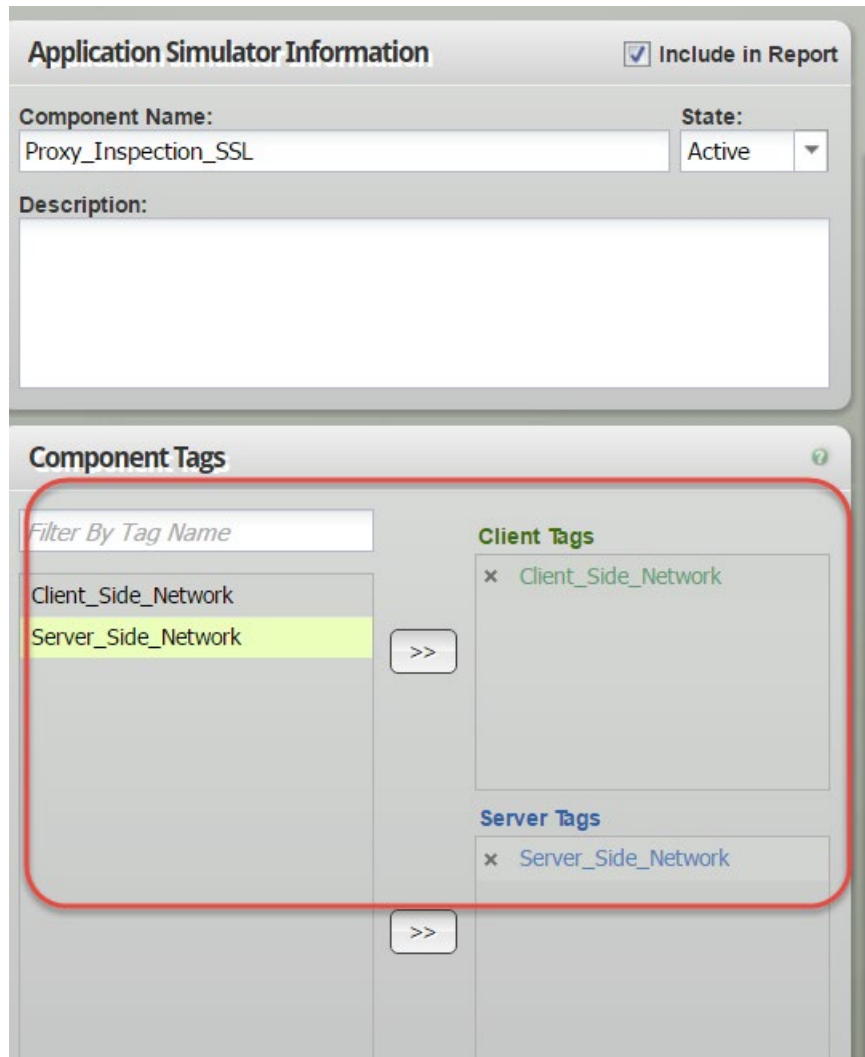


Test Methodologies for Encrypted Traffic Inspection

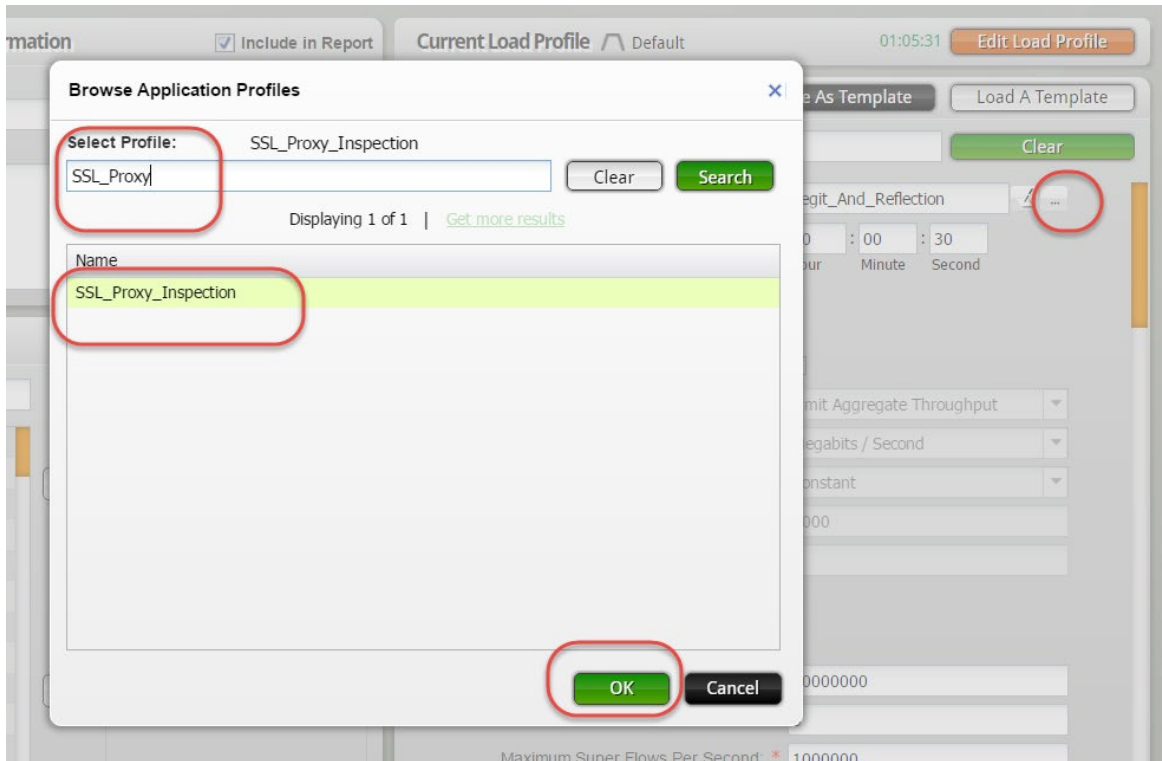
19. In the next steps, we will configure the Application Simulator test component:

- a. In the **Component Tags** section make sure to assign the proper interface tags. For the Client Tags, assign the tag corresponding to the IPv4 Static Host Network Neighborhood element emulating the client side of the proxy

For the Server Tags, assign the tag corresponding to the IPv4 Static Host Network Neighborhood element emulating the server side:

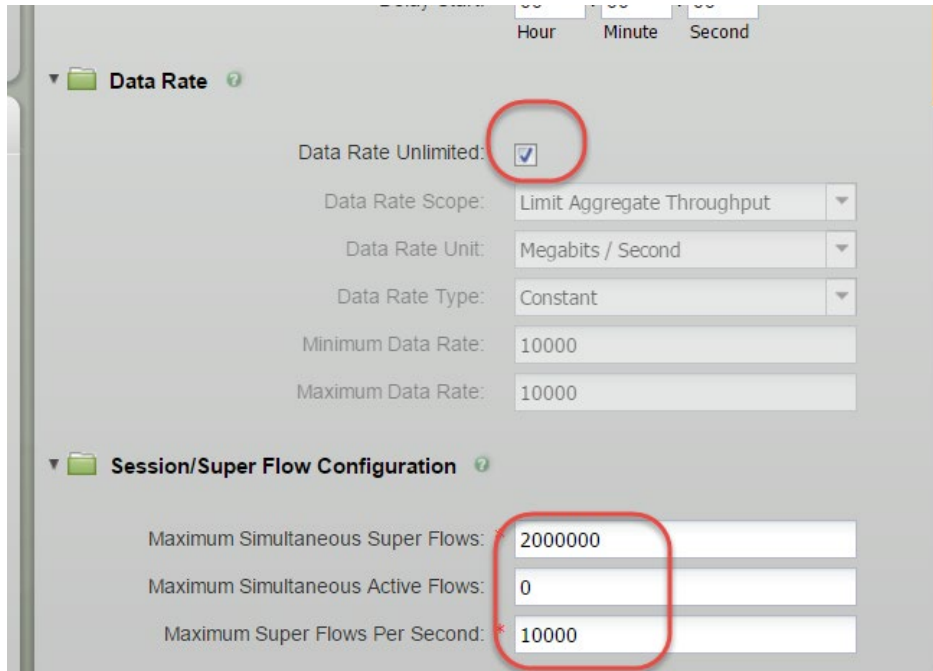


- b. From the Browse Application Profile tab search for the newly created “**Application Profile**” and select ok.



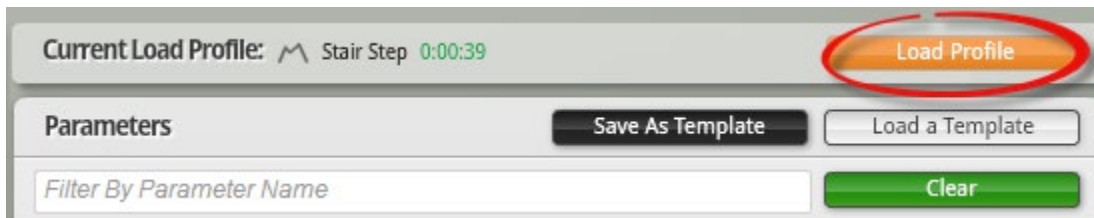
- c. The **Parameters** section contains multiple test component attributes that are grouped into category folders based on their functional purpose. For the scope of this test, since we are trying to find the max sessions per second you can set the Maximum Super Flows per Second value to the DUT’s max capability.
- Data Rate:** check the **Unlimited Data Rate** option so that the test will not be limited by the amount of generated throughput. Instead the traffic load will be controlled by the Maximum Super Flows per second as configured below.
 - Maximum Simultaneous Superflows:** Configure this value to the total number of simultaneous Superflows targeted to be achieved by the legitimate users or the max the DUT can support. For this example, we will use 200,000.

- iii. **Maximum Super Flows Per Second:** set the value to the maximum desired value or DUTs capacity (for this test we will set it to *10,000*).



- d. Configure the legitimate traffic pattern using the Load Profile section:

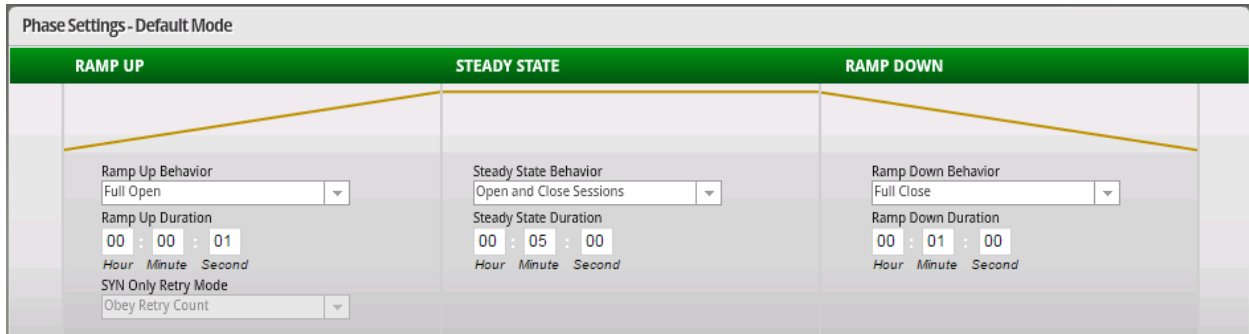
- i. Click on the **Load Profile** button:



- ii. Change the Ramp Up Duration to 1 seconds.
- iii. Configure Steady State Duration to 5 minutes.

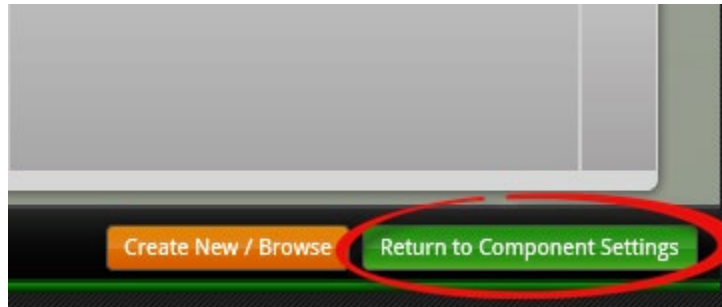
Test Methodologies for Encrypted Traffic Inspection

- iv. Configure Ramp Down Duration to 1 minute.

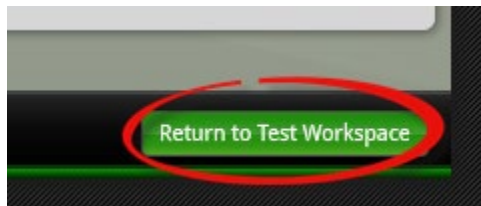


In this example, this AppSim component is configured to generate legitimate traffic for a duration of 6 minutes and 1 second.

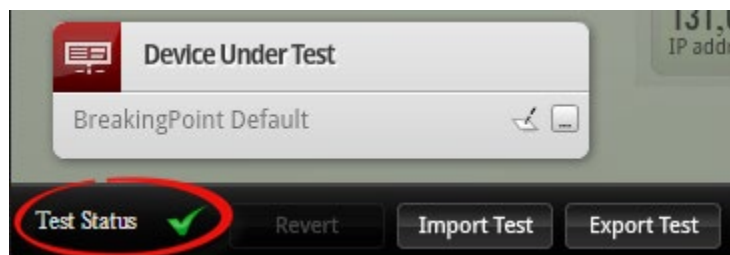
- v. Click on the **Return to Component Setting** button to go back to the Test Component configuration screen:



- vi. Once the Test Component has been configured, click on the **Return to Test Workspace** button to return to the main test screen:

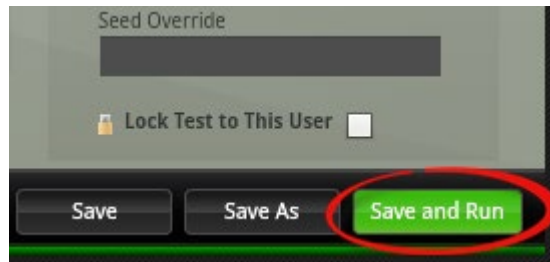


Make sure the **Test Status** indicated (on the lower left corner) has a green checkmark:



If there is not, determine what is wrong by selecting Test Status and viewing the errors.

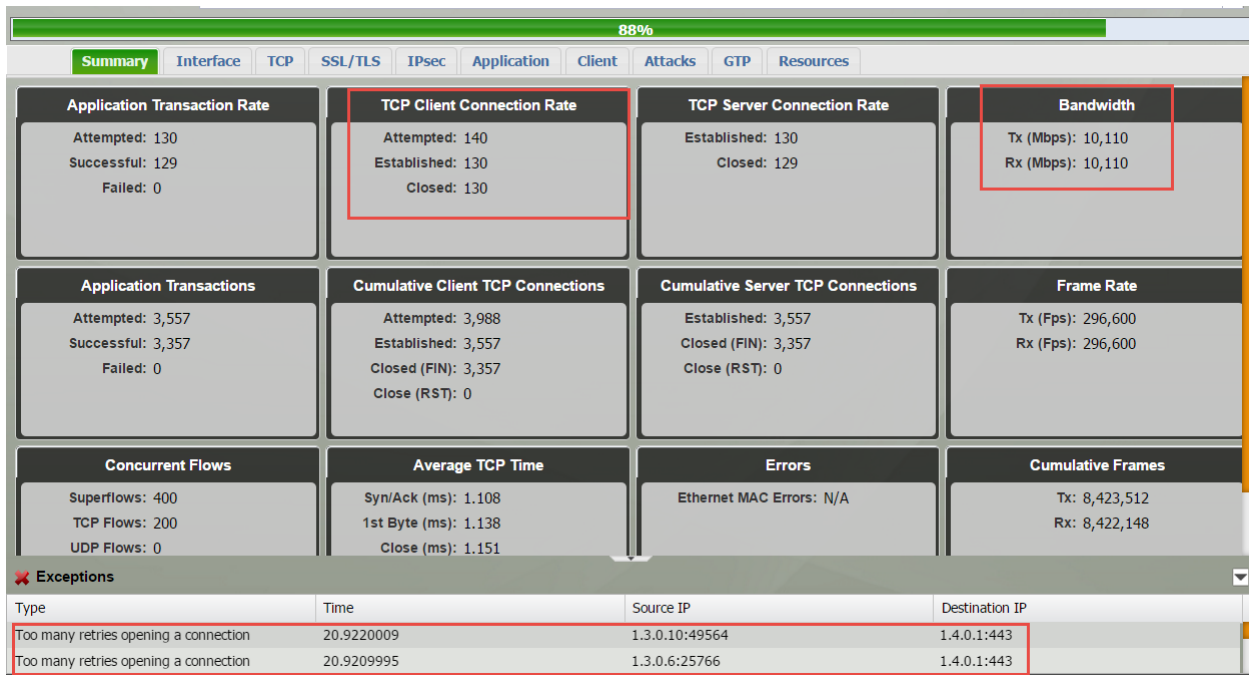
20. Select **Save and Run** from the lower right corner:



Results Analysis

This section covers the key statistics and events that BreakingPoint provides for this type of test.

Real-Time Stats



Test Methodologies for Encrypted Traffic Inspection

The Summary shows the total throughput, connections per seconds, any exceptions, etc.



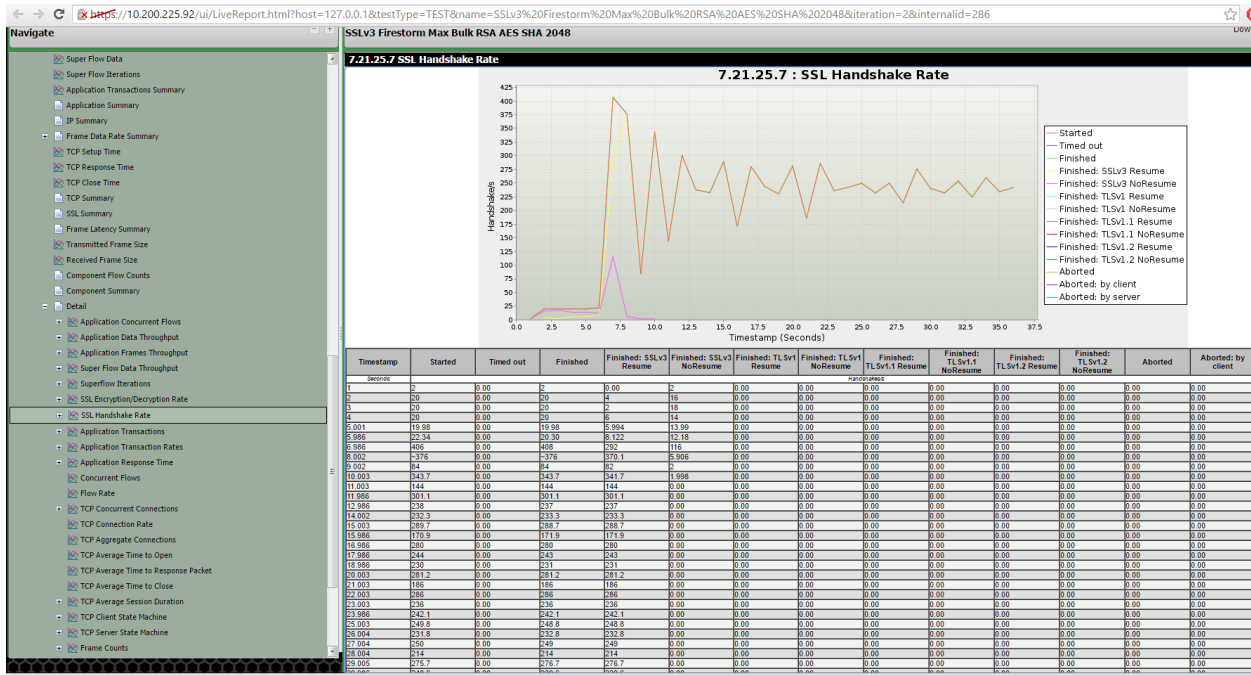
The SSL Handshake and throughput can be seen in SSL/TLS traffic.

Exceptions

Type	Time	Source IP	Destination IP
Too many retries opening a connection	33.0019989	1.3.0.1:44192	1.4.0.9:443
exceptions dropped due to high rate	32.9020004	NA:	NA:

Info | Edit | Report | Stop Capture | Scripts | Follow max 00:00:43 | Go | Stop Series | Stop Test | Restart

[View the report](#)



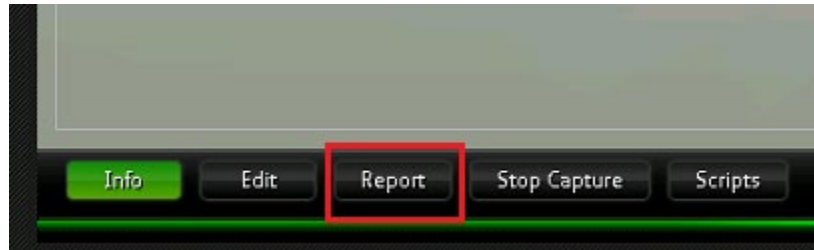
The Reports can be referenced for further granularities.

Test Methodologies for Encrypted Traffic Inspection

The test will stop once all the strikes have been completed.

Report

A detailed report can be generated selecting the **Report** button from the lower left corner of the Real Time Statistics window. This will open the results in a new browser window.



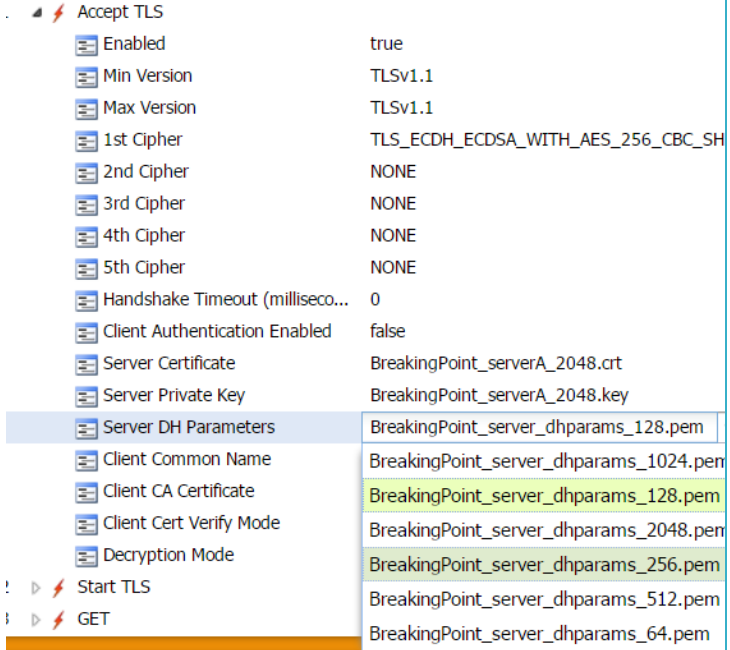
Once a test report is generated, click on the table of contents from the left pane

Test Variables

Use the following test configuration parameters to repeat the test.

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
IP Version	IPv4	IPv6, IPsec, DSLite, 6rd, selected Mobility stacks, etc.
Cipher Type	RSA_AES256_SHA384	<p>Select from the long list of different ciphers available.</p> <p>The dropdown menu lists the following cipher suites: RSA_WITH_AES_256_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_DH_anon_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_GCM_SHA384, TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDH_RSA_WITH_AES_128_CBC_SHA, TLS_ECDH_RSA_WITH_AES_256_CBC_SHA, TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA, and TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA.</p>
Key Sizes	2048	From 512 to 1024, 2048, 3072 and 4096
Curve Sizes	NA	For EC ciphers, you can use curves of 256, 384 or 521

Test Methodologies for Encrypted Traffic Inspection

PARAMETER NAME	CURRENT VALUE	ADDITIONAL OPTIONS
Application Type	HTTP	300 other applications that supports encryption like Facebook, YouTube, smtp, ftp or even send malwares as attachments etc.
Payload Value	128	Can be of 1Meg or more to check max throughput be achieved instead of max sessions.
DH Params	NA	<p>For ECDH select different types of DH params to be sent by selecting the DH.pem files.</p>  <ul style="list-style-type: none"> Accept TLS <ul style="list-style-type: none"> Enabled true Min Version TLSv1.1 Max Version TLSv1.1 1st Cipher TLS_ECDH_ECDSA_WITH_AES_256_CBC_SH 2nd Cipher NONE 3rd Cipher NONE 4th Cipher NONE 5th Cipher NONE Handshake Timeout (milliseco... 0 Client Authentication Enabled false Server Certificate BreakingPoint_serverA_2048.crt Server Private Key BreakingPoint_serverA_2048.key Server DH Parameters BreakingPoint_server_dhparams_128.pem Client Common Name BreakingPoint_server_dhparams_1024.pem Client CA Certificate BreakingPoint_server_dhparams_128.pem Client Cert Verify Mode BreakingPoint_server_dhparams_2048.pem Decryption Mode BreakingPoint_server_dhparams_256.pem Start TLS GET BreakingPoint_server_dhparams_512.pem GET BreakingPoint_server_dhparams_64.pem
Session Reuse	0	Enable session re-use to achieve higher performance with BreakingPoint.

Conclusions

As SSL traffic grows, we see more security devices needing to decrypt SSL traffic. The example shows the proper way to baseline SSL performance under different conditions that are relevant with the ciphers commonly used by browsers like Chrome, Mozilla, and Safari. Users of this technology also need to know the performance penalty exacted to enable the CPU-intensive SSL operations.

SSL traffic can also be optimized without compromising on the security. For example, a user can test some of the elliptical curve ciphers and compare it with the non-elliptical variant to check the performance difference and justify implementing them in your network.

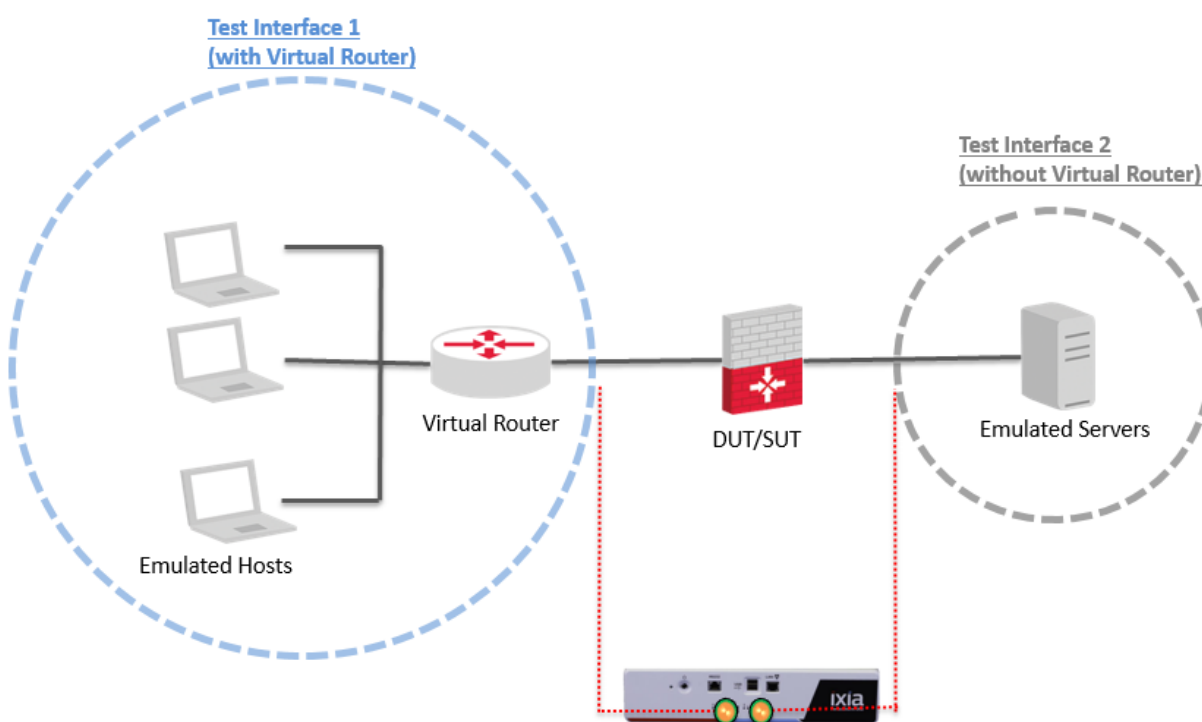
For equipment manufacturers or software developers, this also provides the opportunity to establish handshake and encryption performance of different ciphers, key or curve sizes, and perform optimization in hardware and software to reduce latencies and increase efficiencies.

Appendix A: Configuring a Virtual Router

In many situations, it is important to have an additional intermediary Layer-3 network node emulated in front of the emulated traffic endpoints. In BreakingPoint terminology, such an element is called Virtual Router and it is acting as a router sitting in front of the various hosts and endpoints, which are associated with it.

Below diagram is a logical representation of the network elements emulated by two test interfaces:

- Test Interface 1 – having a Virtual Router in front of the emulated hosts
- Test Interface 2 – no Virtual Router, emulated servers are directly connected



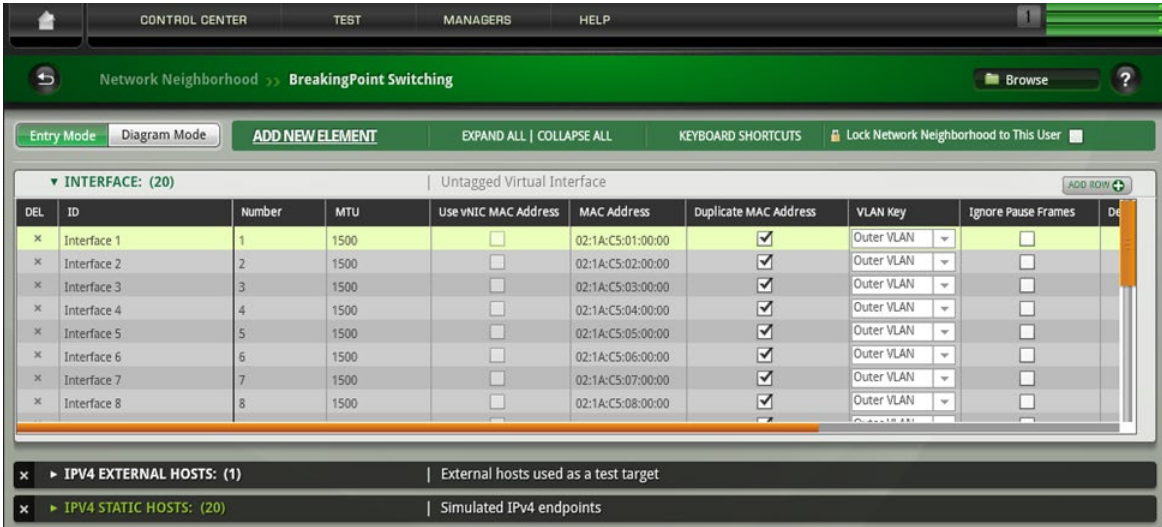
The main benefits that a Virtual Router element offers are:

- Avoid an ARP storm on the interface connected to the DUT/SUT (which can be caused in case many emulated IP address are directly connected to the DUT/SUT without a Virtual Router in between)
- Replicate the real-world scenarios where the hosts are not directly connected to the DUT or SUT, sharing the same IP subnet ranges.

To configure or add a Virtual Router to a Network Neighborhood follow bellow step-by-step instructions:

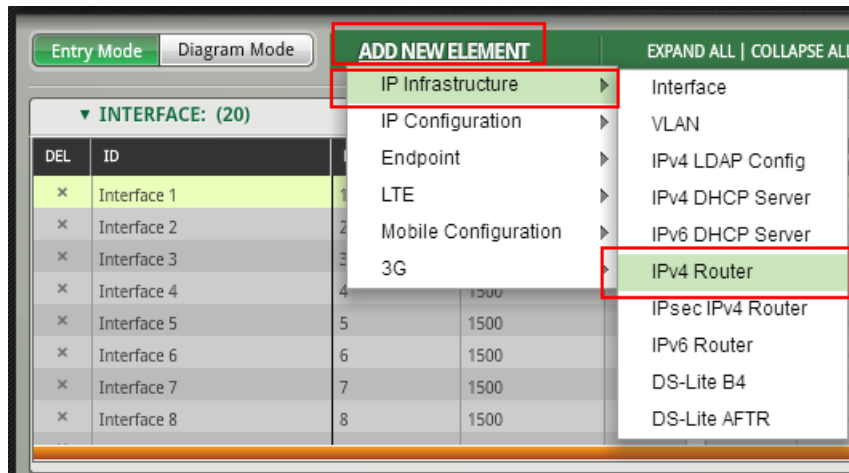
Appendix A: Configuring a Virtual Route

1. Open a new Network Neighborhood or edit your previously configured Network Neighborhood:



In this example, we will use as a template the default BreakingPoint Switching Network Neighborhood

2. From ADD NEW ELEMENT-> IP Infrastructure choose IPv4 Router:



Note: in case of an IPv6 Network Neighborhood, an IPv6 Router can be selected

Appendix A: Configuring a Virtual Route

3. For the newly added IPv4 Router element configure the following settings:

PARAMETER	CONFIGURED VALUE	COMMENTS
ID	VR1	The Identifier to be used by this virtual router element
Container	Interface 1	The physical interface number that the emulated virtual router resides on.
IP Address	10.0.0.2	The external IP address of the virtual router (DUT/SUT facing)
Gateway IP Address	10.0.0.1	The Gateway IP address used by the virtual router (typically the DUT IP address)
Netmask	8	The prefix length of the virtual router.

4. Next, we need to configure the emulated hosts that will be associated with this virtual router. For this example, we will configure the first **IPv4 Static Host** range to sit behind the above created virtual router by selecting **VR1** from the **Container** drop down menu:

The screenshot shows a configuration interface with three main sections:

- IPv4 ROUTER: (1)**: A table with columns DEL, ID, Container, IP Address, Gateway IP Address, and Netmask. The entry for **VR1** is shown with Container set to **Interface 1**, IP Address **10.0.0.2**, Gateway IP Address **10.0.0.1**, and Netmask **8**.
- IPv4 EXTERNAL HOSTS: (1)**: A section for external hosts used as a test target.
- IPv4 STATIC HOSTS: (20)**: A table with columns DEL, ID, Container, Tags, Base IP Address, Count, Gateway IP Address, and Netmask. A dropdown menu for the **Container** column is open, showing options from **Interface 1** to **VR1**. The **VR1** option is highlighted with a red box. The first row in this table is **Static Hosts i1_default**, which is associated with **Interface 1**, has a **Base IP Address** of **1.1.0.1**, a **Count** of **65534**, and a **Gateway IP Address** of **1.0.0.1**.

Note: If multiple host subnets are needed “behind” the same virtual router, additional IPv4 Static Hosts ranges can be configured with the same virtual router as a container.

Appendix A: Configuring a Virtual Route

5. According to the initial diagram, we will leave the 2nd test interface having a static host range directly emulated, without a virtual router:

DEL	ID	Container	Tags	Base IP Address	Count	Gateway IP Address	N
*	Static Hosts i1_default	VR1	Lab Client,i1_default	1.1.0.1	65534	1.0.0.1	8
*	Static Hosts i2_default	Interface 2	Lab Server,i2_default	1.2.0.1	65534	1.0.0.1	8

Note: When configuring virtual routers, it is important to make sure that the DUT is also properly configured to route traffic to the end hosts. Typically, this can be easily achieved by configuring a static route to the end hosts IP subnet with the virtual router IP address as next hop.

Learn more at: www.keysight.com

For more information on Keysight Technologies' products, applications, or services, please contact your local Keysight office. The complete list is available at: www.keysight.com/find/contactus

