2008

Botnets and Distributed Denial of Service Attacks

Anilkumar Panicker
San Jose State University

Botnets and Distributed Denial of Service Attacks

A Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

by

Anilkumar Panicker

December 2008

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

_____

Dr. Mark Stamp

_____

Dr. Robert Chun

_____

Dr. Teng Moh

APPROVED FOR THE UNIVERSITY

_____

ABSTRACT

Botnets and Distributed Denial of Service Attacks

With their ever increasing malicious capabilities and potential to infect a vast majority of computers on the Internet, botnets are emerging as the single biggest threat to Internet security. The aim of this project is to perform a detailed analysis of botnets and the vulnerabilities exploited by them to spread themselves and perform various malicious activities such as DDoS attacks. DDoS attacks are without doubt the most potent form of attacks carried out by botnets. In order to better understand this growing phenomenon and develop effective counter measures, it is necessary to be able to simulate DDoS attacks in a controlled environment. Simulating a DDoS attack with control over various simulation and attack parameters will give us insights into how attacks achieve stealth and avoid detection. A detailed analysis of existing DDoS defense strategies and proposals combined with the insights derived from simulation should enable us to come up with innovative and feasible solutions to prevent and mitigate DDoS attacks carried out using Botnets

ACKNOWLEDGEMENTS

I would like to thank Dr. Mark Stamp for his guidance and support throughout the past year. This project could not have been accomplished without encouragement and expert advice from him. His continual guidance in the project from helping to come up with the initial idea and defining the scope to drafting the final report were pivotal in success of the project.

Very special thanks to Dr. Robert Chun for all his valuable suggestions and even offering his high end computer for the purpose of simulation when my computer was proving to be inadequate for the task.

I would also like to thank Dr.Teng Moh for his guidance and support.

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

A botnet is a group of security compromised computers infected with malicious computer applications called bots. Botnets have been extensively used for malicious activities such as distributed denial of service (DDoS) attacks, click fraud, spam, theft of banking information, identity theft, and theft of other sensitive data from the infected computers (Collins, 2007). Botnets are especially threatening to organizations with large internal networks; a single infected computer in the internal network puts the entire network at risk. In contrast to viruses and worms, a botnet has more potential to generate profit for attackers, which has resulted in an increased focus by malware creators on botnets. (Moheeb, 2007) In order to understand this growing phenomenon and develop efficient counter measures, it is necessary to be able to simulate them in a controlled environment.

The aim of this project is to perform a detailed analysis of botnets and the vulnerabilities exploited by them to spread themselves and perform various malicious activities such as DDoS attacks. DDoS attacks are without doubt the most potent form of attacks carried out by botnets. Simulating a DDoS attack with control over various simulation and attack parameters will give us insights into how attacks achieve stealth and avoid detection. A detailed analysis of existing DDoS defense strategies and proposals combined with the insights derived from simulation should enable us to come up with innovative and feasible solutions to prevent and mitigate DDoS attacks carried out using Botnets.

During the course of this project we have been able learn quite a lot about the basic approaches used by the botnets to infect and spread themselves, the communication and control architecture of various types of botnets and the trends in botnet architecture which is making them more resilient to traditional botnet detection and neutralizing tactics.

This study also includes the study of distributed denial of service (DDoS) attacks, which are without doubt the most lethal use of botnets. A host of DDoS attack strategies along with the vulnerabilities and flaws which they exploit are studied.

We describe the simulation environment setup to perform a distributed denial of service attack on a web server over the Internet, This simulation gave us insights into how to design a highly scalable network without compromising too much on the granularity and the level of detail achieved in the simulation. The simulation was especially useful in understanding how the attack traffic can be made to seamlessly blend into ambient traffic and also the specifics of the SYN flood DDoS attack which was simulated.

This is followed by a detailed analysis of various DDoS attack defense proposals, the merits, demerits and feasibility of each of these proposals. Drawing conclusion from all the analysis and simulation finally we make few suggestions on how to improve the security against DDoS attacks.

2. Botnets

A *botnet* is a group of security-compromised computers infected with malicious computer applications called bots. Once infected with a bot and configured according to the attackers' specifications, an attacker can gain complete control of the infected machine. The attacker or the propagator of a botnet is also known as *botnet herder* or *botmaster*. (Moheeb, 2007)

2.1 Lifecycle of a Botnet

The life cycle of a botnet is fairly standard, irrespective of the control and communication architecture used or the initial method of spreading the malicious code to the target machines. (Craig 2007). Figure 3 below shows the typical life cycle of a botnet infection. Different variants of botnets may have a few additional steps to take care of extra functionalities provided by them.

Given below are the steps which are central to all the botnets.

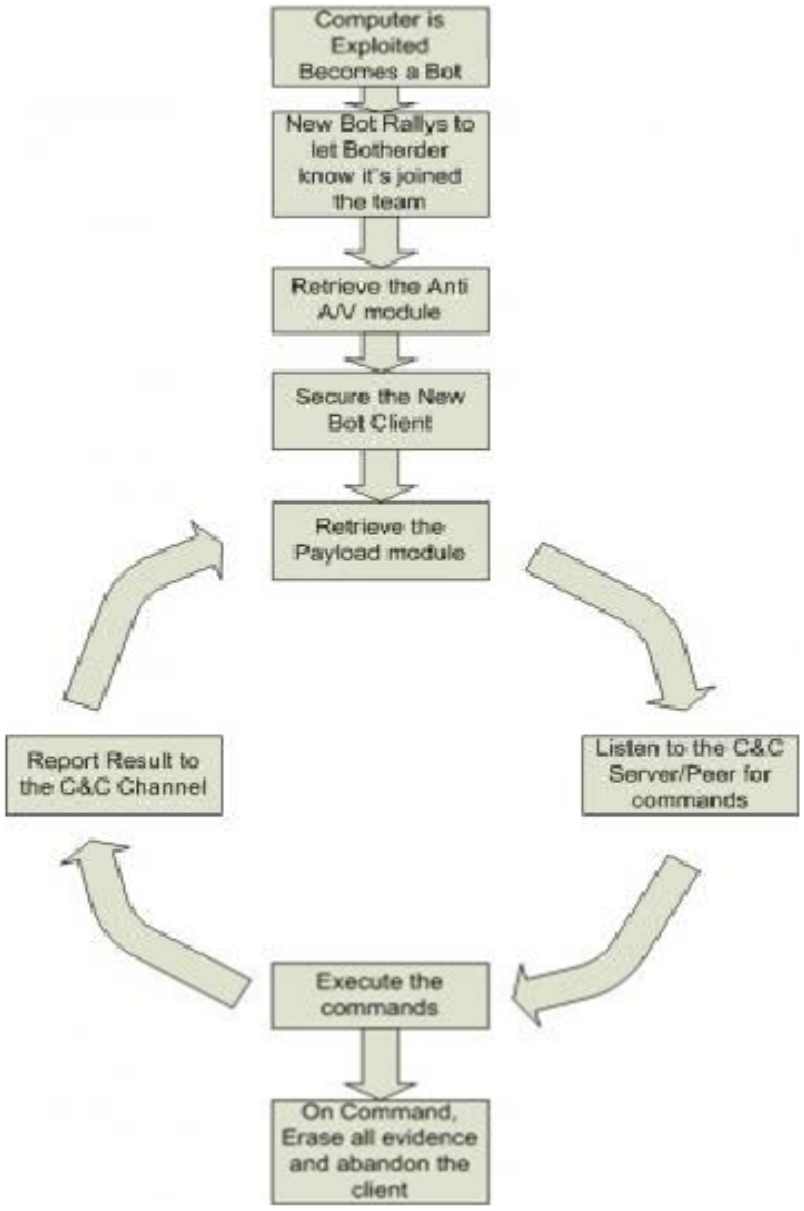

Figure 1: Life-cycle of a typical botnet (Craig, 2007)

The following are the stages in a botnets life (Craig 2007).

1. Exploiting one of the many known vulnerabilities of a target system, the malicious code gains entry into the system and converts it into a bot.

2. The newly infected machine sends a message to the botmaster to let him or her know that it has joined the team. This process is known as rallying.
3. The next step for a bot is to secure itself on this newly infected machine, using the command and control server the bot tries to download and install an anti-antivirus software which will render any antivirus software on the machine ineffective.
4. Amazingly one of the next things that a bot does is to fix vulnerabilities on the host system by applying patches, this will prevent other malicious programs from entering the system and taking over the control of the system.
5. One of the main features of the botnet is that its functionality can change from time to time. The modular design of the botnets allows the botmaster to install payload modules which will implement the functionality which is currently required.
6. The bot then, depending on its architecture listens to either command and control servers or its peers for commands. Most of the botnets use Internet Relay Chat or IRC as the preferred communication medium between the servers and the bots.
7. The bot then executes the command and reports the status back to the command and control servers at appropriate time.
8. At this step the bot again goes back to stage 5, whereby it waits listening on the specified ports for new instructions or new payloads to be delivered.
9. On receiving a predetermined command from the botmaster the bot can completely erase all its traces on the computer and abandon the client. The bot often leaves the system in an unsecured state to ensure that future infections can occur. (Vinoo, 2007)

3. Botnet Control & Communication Architecture

The botmaster choreographs the issuance of commands to the entire botnet using the control and communication infrastructure. The control and communication architecture of botnets is a major area of interest to both botnet creators and security researchers trying to detect and disable botnets. (Wang, 2007) A good understanding of the architectural approach used by a particular botnet can lead to developing an effective detection mechanism for an entire genre of botnets and their mutations.

3.1 Centralized architecture / Traditional Botnet architecture

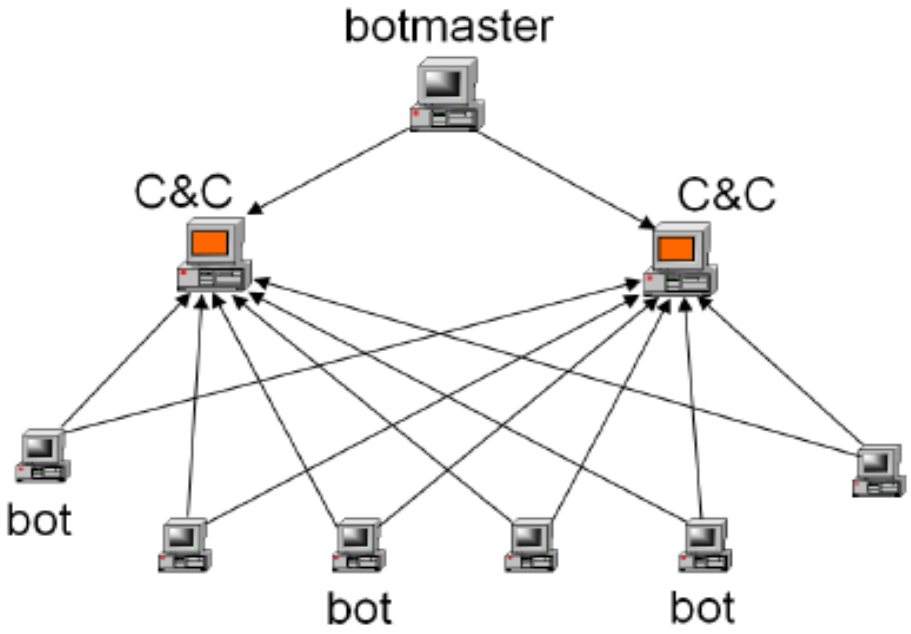

Figure 2: Command and control architecture of Botnet (Usenix, 2007)

Figure 2 shows the centralized command and control architecture used in a majority of existing and traditional botnets. In such kind of a network all the bots are directly connected to a few specialized hosts called command and control server (Usenix, 2007).

The botmaster communicates to the entire botnet exclusively through these few command and control servers. Though this traditional architecture of centralized command and control servers is very efficient in terms of scalability and the ease with which the botmaster can issue commands to his entire network of infected computers, it has a major limitation. Detection or failure of few of these command and control servers can result in botmaster loosing the control of all the computers connected through those servers.(Usenix, 2007) Once a bot is captured, the identities (IP addresses) of these limited command and control servers are revealed. Hence the command and control servers in this architectures act as a single point of failure. P2P architectural approach taken by newer botnet strains tries to mitigate some of the problems with the centralized command and control architecture.

3.2 Peer to Peer Botnets

As the limitations of the centralized command and control architecture became apparent, the attackers began experimenting with a new kind of architecture which would allow them to avoid detection and continue their attacks from distributed locations. These new architectures utilized Peer to Peer communication protocols and are now the most common among the newer kind of botnets.

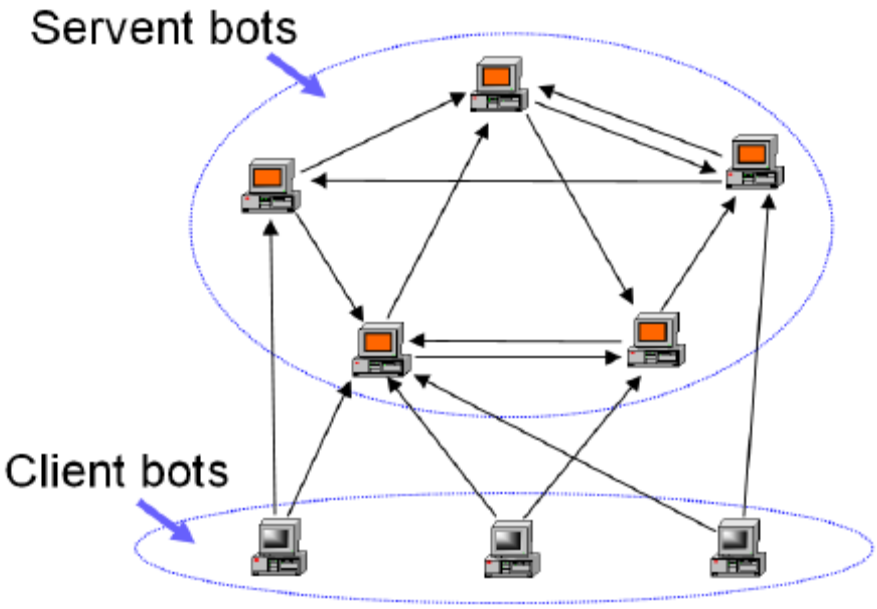

Figure 3: Command and control architecture of the P2P botnet (Usenix, 2007)

Figure 3 shows the command and control architecture of the newer kinds of hybrid peer to peer botnets. The problem of single point of failure is solved by creating redundancy among the command and control servers. These servers can work completely independent of each other, thus even if one of the servers is detected and brought down there is very little effect on the overall functionality of the botnet. In this design not even the servers are aware of all the computers in the botnet, each server is aware of only a subset of the total number of infected computers, this adds to the stealth of the botnet. (Usenix, 2007)

Though the botnets using these newer approaches have succeeded in removing single point of failure to a large extent they are plagued with newer problems like scalability and large amount of network traffic. The peer to peer nature of these newer kinds of

botnets results in a large amount of network traffic, making it susceptible to monitoring via network flow analysis (Usenix, 2007).

Just making a botnet to use P2P protocols is not enough to make it more robust and resistant to detection, hence botnet creators have been experimenting with new techniques such as making use of "Sensor hosts". A *sensor host* is a host which the botmaster utilizes to receive status information from all the bots in the botnet. (Wang, 2007) The IP address of the sensor host changes each time a status report instruction is sent out by the botmaster. This IP address is generally specified in the report instruction itself. The hosts classified as servant bots in this architecture are the ones which have static IP addresses. The sensor bots behave both as servers as well as clients.(Wang, 2007)

In addition to adoption of P2P protocol to increase the resilience of botnets to single point of failure. Botmasters have also trying their hands at making super botnets by combining several smaller botnets and this seems to be the trend towards which the botnet architecture is moving (Ryan, 2006)

4. Botnet Communications

Botnets require a large volume of communication between the control and communication servers and the client bots in order to coordinate the activities of the geographically distributed bots and to distribute payloads which help in configuring the client bots according to the botmaster's requirements.

Traditionally Internet Relay Chat or IRC has been the preferred channel of communication for the botnets. Most of the botnets having the centralized command and control server architecture use the IRC. (Moheeb, 2006) The reason why IRC was chosen as botnet communication channel was due to its obvious advantages such as

1. Good Broadcast Medium: IRC is very good broadcast medium which is suitable for botmaster to broadcast commands to his or her large army of bots. (Puri, 2003)
2. Easy Availability: A large number of IRC servers are freely available to the general public on the internet. (Peng, 2007)
3. Easily Configurable: IRC channels can be easily configured to the specifications required by the botnets. IRC protocol was specifically designed for several different forms of communication such as point to multi point and point to point. (Puri, 2003)
4. Open Source: Large numbers of open source implementations of IRC are freely available on the internet.

The use of IRC in botnet communication became so conspicuous that any software using IRC began to be suspected of being a bot. This led botmasters to come up with new innovative architectures which made use of HTTP and P2P protocols. Many of these new

architectures, which made use of HTTP ports, are much harder to trace because of the large number of such ports on any standard computer. (Ono, 2007). The P2P approach on the other hand provides the benefit of substantially increased robustness in the botnet. (Patrick, 2007)

5. Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks

Distributed denial of service (DDoS) attacks are undoubtedly the most disruptive and potent use of Botnets. With their ability to bring down entire websites and cause widespread disruptions on internet they are emerging to be one of the most prolific uses of the botnets.

DDoS attacks are especially threatening to companies whose business depends on the online availability of their websites. (Peng, 2007).A well planned DDoS attack can simply overwhelm victim and force it to stop completely or run at severely degraded levels. A widespread DDoS attack on the DNS root servers can put the entire Internet at risk. As more and more critical services such as air travel, medical care etc. become increasingly dependent on Internet for their communication needs the damage caused by DDoS attacks can be catastrophic (Peng, 2007)

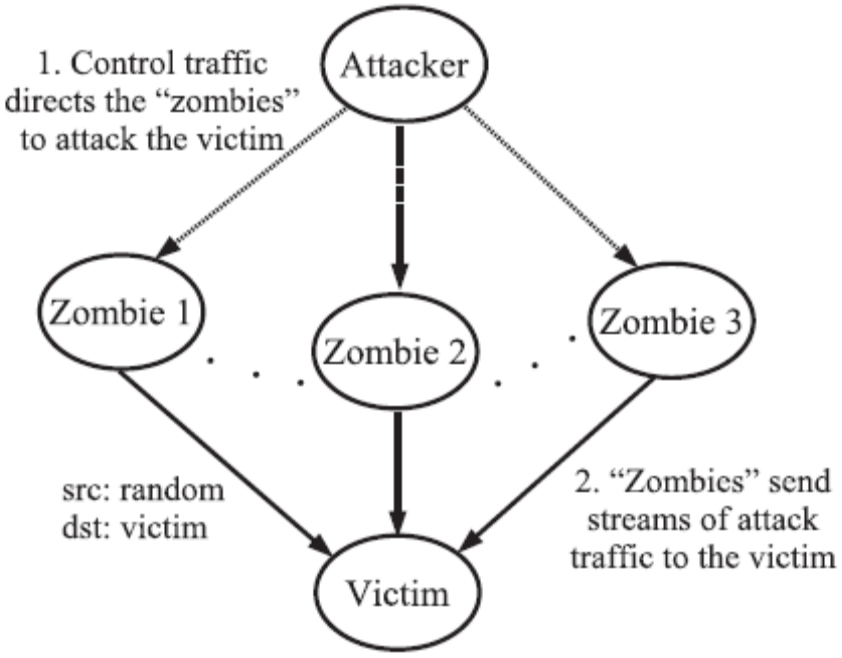

Figure 4: Structure of a typical DDoS attack

(Peng, 2007)

There are many ways in which a Denial of Service attacks can be carried out such as

5.1 ICMP flood: This type of attack is also known as *Ping flood* or *ping of death.* The basic idea is to flood the victim with a large number of ping requests thereby using up all the bandwidth. (Peng, 2007)

5.2 SYN flood: During a *SYN flood* attack the attacker uses the vulnerability of the TCP three way handshake process to exhaust the memory stack allocated by the Operating System. SYN flooding is very similar to ICMP flooding. (Peng, 2007).

5.3 HTTP flood: This kind of attack floods the web server with HTTP requests. Port 80 being the default HTTP port is left unfiltered on most of the servers to allow the HTTP web traffic to pass through. HTTP floods have become one of the most common forms of attacks carried out by the Botnets (Peng, 2007). HTTP flood is a form of application based bandwidth attack.

5.4 SIP flood: *SIP flood* is again an application based bandwidth attack like the HTTP flood attack. SIP (Session Initiation Protocol) flood targets the SIP proxy servers used for making VoIP calls. (Peng, 2007)

5.5 Teardrop attack: This attack utilized vulnerability in the earlier Microsoft Operating Systems and some older versions of Linux whereby improperly framed or overlapping IP fragments were sent to the victim thereby crashing it. (DOE-CIRC, 1998)

5.6 Peer to peer attacks: These kinds of attacks exploit flaws and bugs in peer to peer servers to launch DDoS attacks. The attacker can exploit the flaw to force innocent peers in the network to connect to the target. This results in the target

being swamped with thousands of connection requests within very short time. (Peng, 2007)

5.7 Permanent denial of service: Permanent denial of service attack is also known as *Plashing*. Plashing causes permanent damage to the hardware of the target by targeting any network configurable device software. (Peng, 2007)

5.8 Infrastructure attacks: Infrastructure attacks target the Internets infrastructure such as the DNS root servers. They can be extremely dangerous as they have the potential to bring down the entire Internet. (Peng, 2007)

A particular DDoS attack can use strategies from any of the above listed attacks.. Most of the bandwidth attacks tend to be of the DDoS nature.

6. Indications of DoS and DDoS attacks

According to the United States Computer Emergency Readiness Team or US-CERT some of the common signs of a Denial of Service attacks are as follows (McDowell, 2008)

1. Degraded network performance in opening files or accessing websites
2. Inability to access particular website
3. Inability to access any website
4. Substantial increase in the number of spam mails received. (McDowell,2008)

A Denial of service attack against a server does not affect just the victim but also affects the performance of several other computers in the vicinity of the victim. All the devices

which share the same network resources as the victim are affected by the attack. (Peng, 2007)

Though a sudden increase in traffic can be an indication of a DDoS attack not all sudden increases in traffic to a website can be attributed to a denial of service attacks. The sudden surge in traffic could be the result of a phenomenon called “Slashdot effect”.

7. Slashdot Effect

A *Slashdot effect* is observed whenever an ill prepared or a small website not having sufficient bandwidth or serving capability gets listed on a popular website such as Slashdot.org as a reference link to an article. The resulting sudden surge in traffic as a result of thousands of page requests to the website from the members of Slashdot often results in the website not being able to handle the traffic and as a result going down and becoming unavailable. (Wikipedia, 2008). This phenomenon is also known as *flash crowd*.

8. Why do we need a Botnet Simulator?

Figure 4 shows the number of serious software vulnerabilities reported each year worldwide.

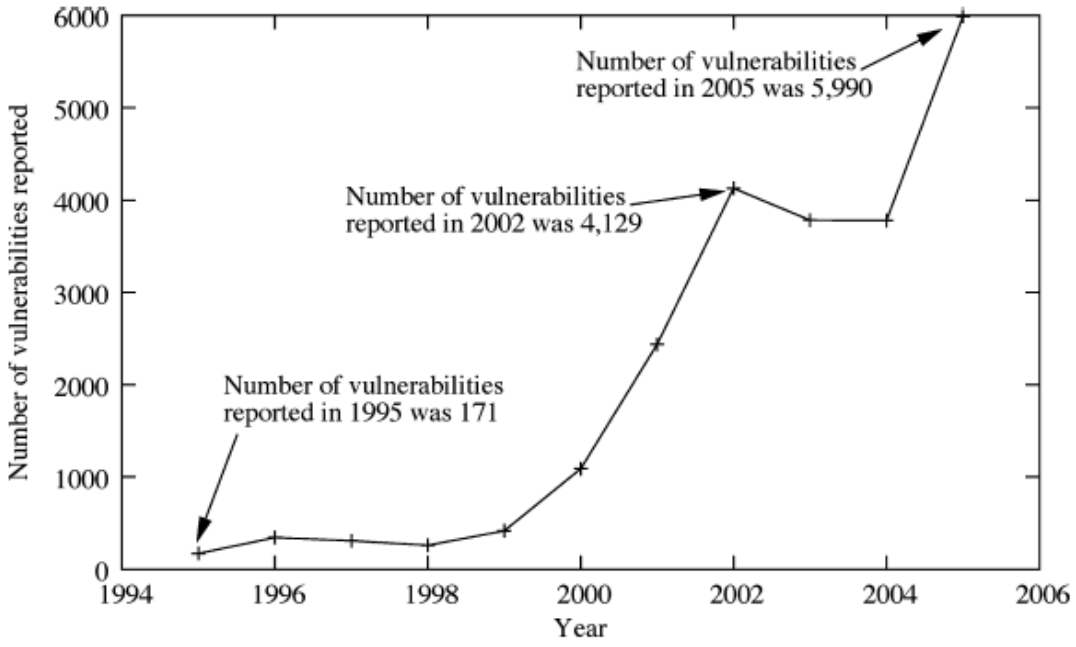

Figure 5: The number of vulnerabilities reported each year according to CERT (Peng, 2007)

As the computer systems and software become more complex there will be a corresponding increase in the number of new vulnerabilities created which can be exploited by botnets. Botnets have much more potential for generating profit for their owners as compared to other malicious programs such as Viruses, Trojans, Worms and Spy-ware. Botnets are also much more configurable as they can be updated at any stage in their life cycle to make use of any new vulnerability detected in computer systems. All these combined with a host of freely available online botnet creation tools have resulted in a virtual explosion of botnets and botnet variants on the Internet (Puri, 2003).

This simulator would allow us to mimic various kinds of botnet DDoS attacks in a controlled environment. It will be especially useful in simulating newly discovered botnets and their possible mutations. This could prove to be an effective first step towards botnet detection and prevention.

9. The Simulation:

Any efficient simulator trying to simulate network events should be capable of scaling up to the required proportions while still maintaining desired level of granularity in various simulation parameters. Failure to do so will result in loss of the fidelity of the simulated results (Wei, 2006). In order to achieve high quality and reliable results from the simulator, the tools which were chosen were highly scalable and from open source domain with proven track record in realistic network simulation.

9.1 Key Components of Simulation:

Following are the key components used in creating the simulation environment. The entire simulation environment runs on gOS or good-OS which is based on Ubuntu but with a lightweight window manager which consumes fewer resources and keeps maximum system resources free for applications such as network simulations. (Good, 2008)

9.1.1 Omnet++

Omnet++ is a discrete event simulation environment which is widely used for simulation of communication and queuing networks.(OmnetPP, 2008). Omnet++ along with INET

framework is very well suited for the simulation of large scale networks with complex topologies. The main feature of Omnet++ is that it offers component architecture for models (OmnetPP, 2008). The components themselves are programmed in C++. This component based model of Omnet++ allows the designer of the network to reuse the components and also create complex components using basic components.

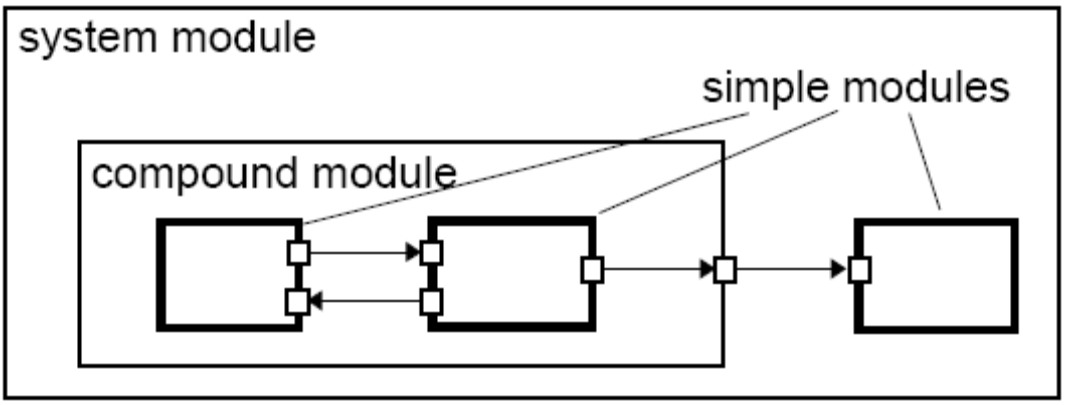

Figure 6: Model Structure in Omnet++: Compound and simple modules, gates and connections (Andras, 1997)

Creation of complex components and models is done using a high level topology description language called NED. There is no restriction on the depth of the modules or the level of nesting which can be done with NED. This allows the user to create complex systems. (Andras, 2005).

9.1.2 NED, Topology Description language:

NED is a high level topology descriptor language used for describing the topology of a model. NED allows modular description of a network topology. (Andras, 2005). NED files are represented with a .ned extension. Omnet++ simulation environment allow us to

dynamically load the NED file when the simulation is running or convert the NED file into C++ file using NED compiler. (Andras, 2005)

9.1.3 INET Framework:

INET framework is used over Omnet++ simulation environment to create simulations of wired, wireless and ad-hoc networks. It provides support for IP, PPP, TCP/UDP, OSPF, RIP and several other protocols (Andras, 2008). The use of INET framework spares the designer of the network the implementation of common protocols used over the Internet. Link Layer models such as Ethernet, PPP and 802.11 are already built in INET. The standard behavior of routers, switches and other network devices used in creating a network are all defined by the INET.

In addition to all the protocol implementations and defining the behavior of standard networking devices, INET framework also provides a scenario manager which allows us to change parameters in the model in the middle of a simulation run. INET also performs the task of logging the events happening during the simulation with the help of inbuilt trace program called NAMTraceWriter. (Andras, 2008). All these features of INET are quite instrumental in creating realistic simulations of the Internet.

9.1.4 Realistic Simulation Environment for Omnet++ (ReaSE):

ReaSE is an extension to the INET framework which allows us to create realistic simulation environments considering multiple aspects of a real world network such as traffic patterns and attack traffic. ReaSE also takes care of topology generation both on AS level and Router level (Gamer, 2008).

9.2 A Sample Network:

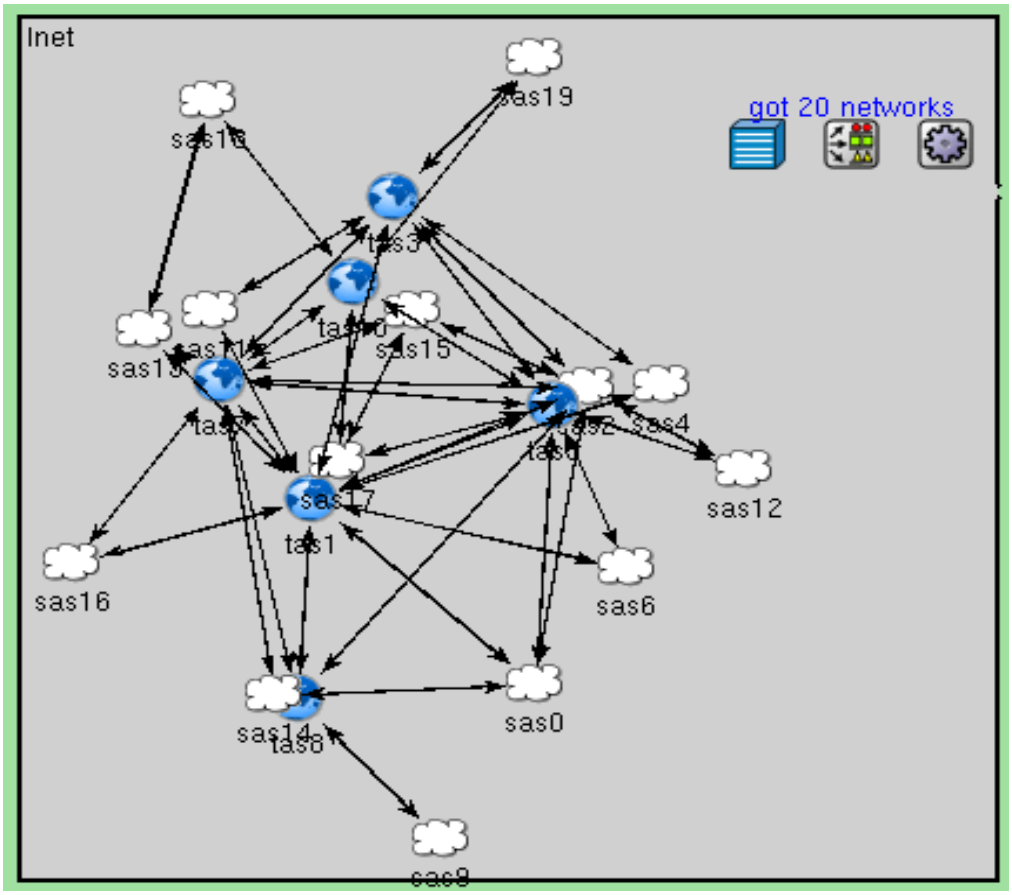

Figure 7: A Network created using the ReaSE extension for INET for simulation of Internet.

The figure above shows a sample network topology for simulation of Internet, generated using the ReaSE extension for INET. This network is formed using the example given in ReaSE.(ReaSE, 2008) This network consists of 20 clouds, each of which is represented using a cloud or a globe in the diagram. In total the sample network has a total of 20 clouds with 3630 devices which include hosts, web servers, Interactive servers, mail servers, streaming servers, routers and gateways. All the links shown in the above diagram are part of the internet backbone and hence have very high bandwidth.

The networks represented with globe are the ones with high bandwidth core routers. The link bandwidth, ambient background traffic and the composition of each of the cloud is defined using a NED file. All these parameters can be changed in a running simulation with the help of TGM network configurator, Connection manager and Traffic profiles manager. All three of them can be accessed by clicking on the three boxes on the right top of the topology showing the entire INET.

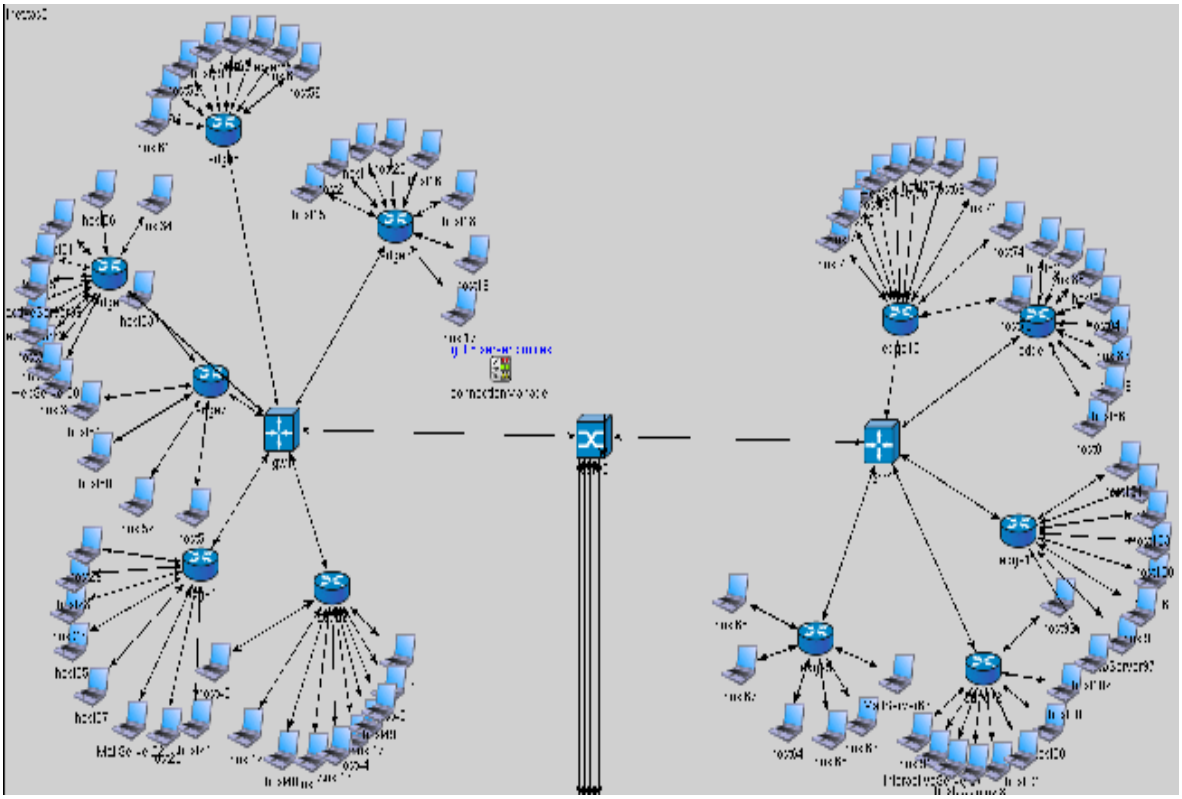

Figure 8: Expanded view of one of the clouds from the above sample network.

The figure above shows the expanded view of one of the clouds for the network topology shown in the figure 7. Please note the four tier architecture of the cloud.

The network is built on a four layer approach. The first layer consists of all the hosts and servers. Hosts are the end user computers; it is on these computers with minimum protection that the botnets rely on to establish their network and launch attacks from. All

the DDoS zombies are present at this level. There are four kinds of servers in this network they are mail server, streaming server, web server and Interactive server. The second layer consists of edge routers, all of the subscriber hosts are directly connected to only one of the edge routers. The edge routers belong to end user organizations. Edge routers provide connectivity between end user hosts and gateways.(Searchnetworking, 2007).

The third layer of the network is composed of gateways. A single *gateway* can have multiple edge routers connecting to it, in effect several LANs can converge at a gateway. The primary difference between a router and a gateway is that gateways can operate on all seven OSI layers and can convert protocols among communication networks (Wikipedia, 2008). The gateways play a crucial role in allowing interoperability among different networks. In most of the cases a full fledged computer server performs the task of a gateway.

Core routers are present at the fourth level of the network. *Core routers* connect multiple gateways to the internet backbone. They operate at the core link speed and are generally the latest and greatest in the current router technology. (Wikipedia, 2008). A core router can be connected to multiple other core routers through the internet backbone. As of today most of the core links are above the speed of 10Gbit/s with a few links reaching speeds of 40Gbits/sec.

All the servers in the network were set to have a maximum queue length of 1000 requests, After the number of TCP requests made crosses 1000 requests per second, the subsequent new requests coming in get dropped

9.3 Simulating the DDoS attack:

Some of the server and network configuration for this attack are as follows

1. The target for the DDoS attack was Inet.sas0.WebServer33.
2. Attack Destination port was port 80. (The reason for choosing this port was that most of the servers leave port 80 unfiltered for the purpose of allowing HTTP traffic.)
3. The DDoS attack used here is SYN flooding.
4. Attack was carried out using the 79 DDoS zombies spread across all the clouds in the network.
5. Attack Probability was set at 0.5. (Which means that once the attack starts the Zombies will be sending out attack packets to the victim for half the duration of the attack)
6. Attack began in the 170^{th} second of simulation and went all the way till the end of simulation at 3600 seconds.

The SYN flood attack exploits the vulnerability of TCP three-way handshake protocol, Each SYN request coming from each of the DDoS zombies will result in opening up a new TCP connection at the target. The OS running on a server would allocate a limited amount of memory stack to keep track of all the TCP connections being established, once this number crosses a certain threshold the quality of service being offered by the server begins to deteriorate until eventually the requests start getting timed out or the server totally goes offline.

In order to simulate the effect of DDoS attack generated by botnets the attack was carried out using zombies spread across the entire network. Random hosts from across various

clouds in the network were converted to DDoS zombies just as a regular botnet would infect host computers on the internet with particular set of vulnerabilities.

```
      display: "i=device/laptop";
    host42: DDoSZombie;
      display: "i=device/laptop";
    host43: InetUserHost;
      display: "i=device/laptop";
    WebServer44: WebServer;
      display: "i=device/laptop";
    host45: InetUserHost;
      display: "i=device/laptop";
    host46: InetUserHost;
      display: "i=device/laptop";
    host47: InetUserHost;
      display: "i=device/laptop";
    host48: InetUserHost;
      display: "i=device/laptop";
    host49: InetUserHost;
      display: "i=device/laptop";
    host50: DDoSZombie;
      display: "i=device/laptop";
    host51: InetUserHost;
      display: "i=device/laptop";
    host52: InetUserHost;
      display: "i=device/laptop";
    host53: InetUserHost;
      display: "i=device/laptop";
    host54: InetUserHost;
      display: "i=device/laptop";
    host55: InetUserHost;
      display: "i=device/laptop";
    host56: InetUserHost;
      display: "i=device/laptop";
    host57: InetUserHost;
      display: "i=device/laptop";
    host58: DDoSZombie;
      display: "i=device/laptop";
```

Figure 9: Snippet of code from the .ned file of the network where random nodes in the network are converted into DDoS Zombies

The screen shot in figure 9 shows a snippet of code from the .ned file for the network, please observe that random hosts from the network were converted into DDoS zombies. In all a total of 79 hosts across the entire network were infected by the botnet and converted into DDoS zombies.

```
AttackPath (187.507) :<Inet.sas14.host94>-->Inet.sas14.edge12-->Inet.sas14.gw2-->Inet.sas14.core0-->Inet.sas0.core0-->Inet.sas0.gw
1-->Inet.sas0.edge6-->Inet.sas0.WebServer33

AttackPath (187.841) :<Inet.tas3.host55>-->Inet.tas3.edge9-->Inet.tas3.gw2-->Inet.tas3.core0-->Inet.tas1.core0-->Inet.sas0.core0--
>Inet.sas0.gw1-->Inet.sas0.edge6-->Inet.sas0.WebServer33

AttackPath (189.324) :<Inet.sas16.host89>-->Inet.sas16.edge13-->Inet.sas16.gw2-->Inet.sas16.core0-->Inet.tas1.core0-->Inet.sas0.co
re0-->Inet.sas0.gw1-->Inet.sas0.edge6-->Inet.sas0.WebServer33

AttackPath (189.54) :<Inet.sas11.host60>-->Inet.sas11.edge11-->Inet.sas11.gw2-->Inet.sas11.core0-->Inet.tas1.core0-->Inet.sas0.cor
e0-->Inet.sas0.gw1-->Inet.sas0.edge6-->Inet.sas0.WebServer33

AttackPath (189.672) :<Inet.sas17.host81>-->Inet.sas17.edge12-->Inet.sas17.gw2-->Inet.sas17.core0-->Inet.tas5.core0-->Inet.sas0.co
re0-->Inet.sas0.gw1-->Inet.sas0.edge6-->Inet.sas0.WebServer33
```

Figure 10: screenshot of the terminal showing the complete path from attackers to victim during the 187th and 189th second of attack simulation

Figure 10 shows the complete attack path followed by the attack packets to reach from the DDoS zombie to the target victim. One visible impact of all the incoming attack packets is that there is a lot of traffic going through the core router core0 of cloud sas0, gateway router gw1 and the edge router i.e. edge6 to which the victim web server 33 is connected. This causes the performance of all the servers in cloud sas0 which are connected to core router core0 to suffer.

The attack was carried out in such a manner that the attack traffic would seem as close as possible to the regular ambient traffic. This was achieved by keeping the attack packets going out from each DDoS zombie within a certain limit per unit time.

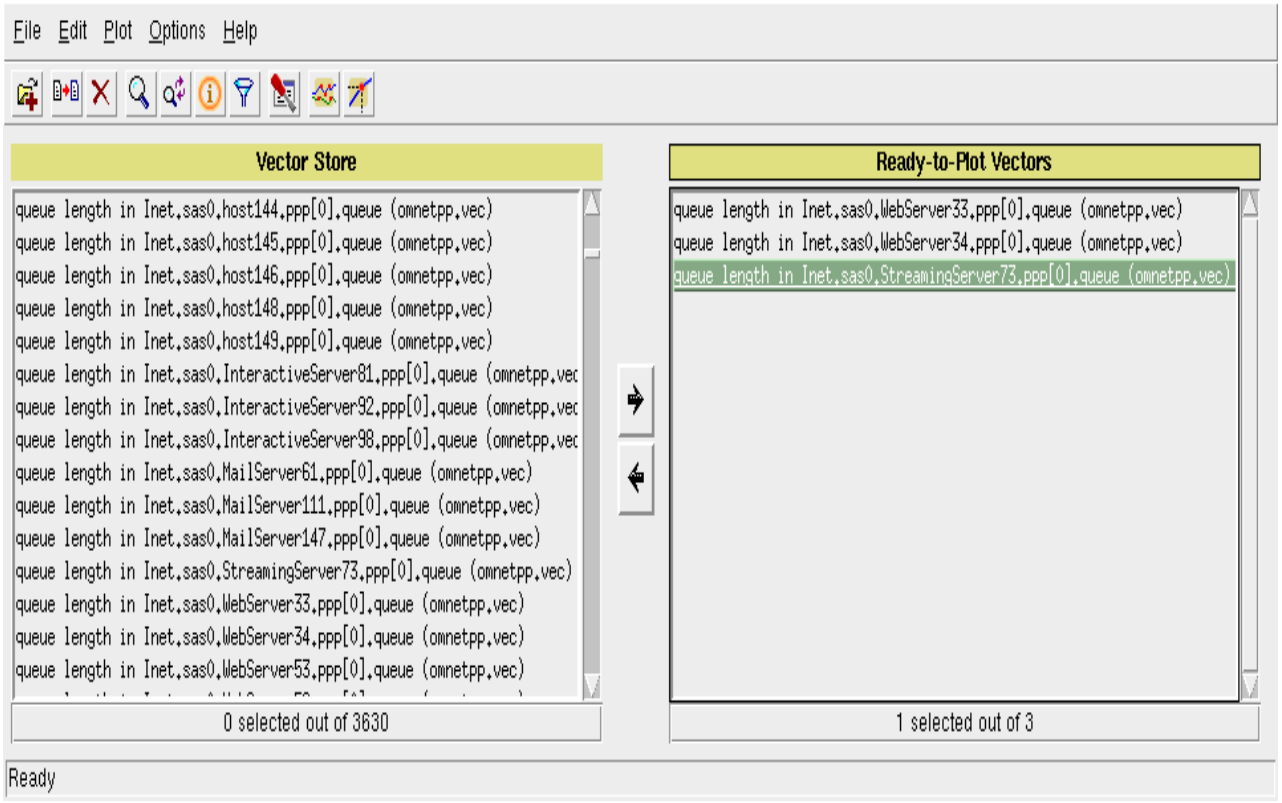

Figure 11: Plove Vector plotter being used to select the Vectors for three servers from the same network cloud being selected to plot the results. (OmnetPP, 2008)

The data from the simulation of DDoS attack is recorded in the form of a vector file by the INET framework. The vector file is stored with a .vec extension. This vector file contains a vector for each of the 3630 devices in the network. Since the SYN attack focuses on exhausting the queue capacity of the target server the vector records the length of the queue for each of the devices in the network. The size of the vector file for an hour long simulation comes out to be near 73 Mb.

The data recorded in this vector file was plotted onto a graph with the help of a vector data plotter called plove which is built into Omnet++ simulation environment. Figure11 shows the selection of three vectors corresponding to WebServer33, Mailserver61 and

InteractiveServer92 from the vector store. The vector store has vector recordings for all the 3630 device vectors in the network.

In order to clearly demonstrate the effects of the DDoS attack and contrast the amount of traffic the target is getting as compared to other comparable servers in the same network, the following servers were selected to be plotted

1. Inet.sas0.WebServer33. (This is the target of the DDoS attack)
2. Inet.sas0.Mailserver61 (A mail server in the same cloud as the target victim)
3. Inet.sas0. InteractiveServer92 (A server in the same cloud as the target victim)

The ambient traffic is kept pretty low to clearly show even small spikes in attack traffic on the target.

Figure 12 shows the graph plotted with the recorded vector file of the DDoS attack. X-axis shows the time in seconds while the Y-axis shows the length of the queue. The victim or the target server is represented by blue color.

The entire simulation lasted for 3600 seconds, of these 3600 seconds the attack lasted for 3430 seconds. Within these 3430 seconds the number of TCP requests pending or the queue length of webserver33 i.e. the victim crossed 500 twice, At one instance the queue length even crossed 1000, which resulted in server dropping all new requests coming into the queue which could very well have been legitimate traffic. Even though the queue length of other servers plotted in the graph remains unchanged to a large extent the performance of these servers suffer due to the fact that the links and routers in the immediate vicinity of the target often get saturated with heavy data flow through those devices.

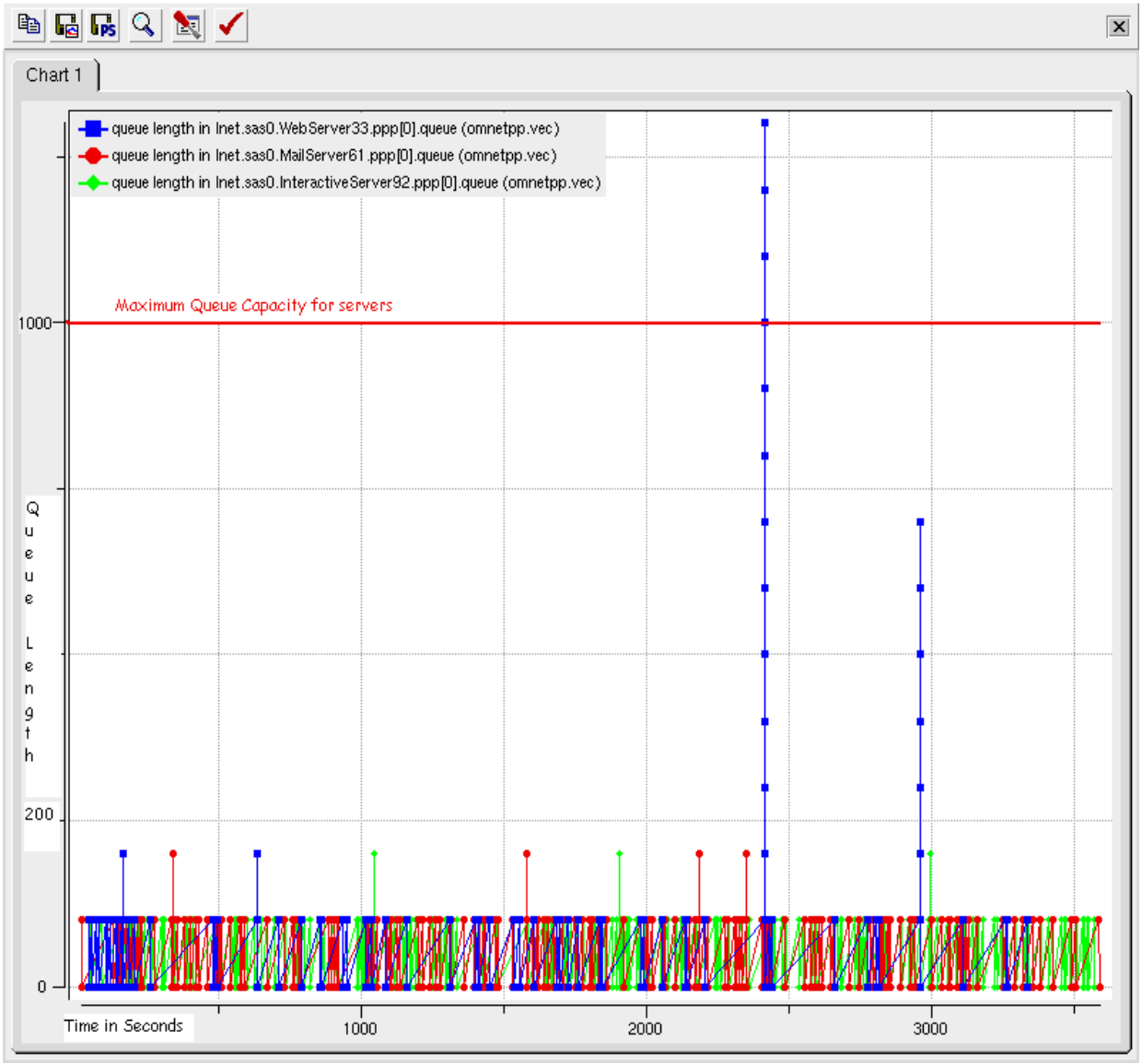

Figure 12: Graph showing DDoS attack happening on WebServer33 with egress filtering(OmnetPP, 2008)

This graph shows the effects of DDoS attack happening on the webserver33. The defense mechanism used here is egress filtering applied at the edge routers.

Simulating DDoS attack with Automatic usage capping

The Egress filtering scheme which was used as the main method of defense against DDoS attacks in the previous experiment has a major limitation that it works on the assumption that the attack traffic uses spoofed addresses. A Bot master can effectively nullify any effect of this filtering scheme by simply configuring the botnet not to generate any traffic with spoofed address.

In the first experiment the address spoofing flag of the DDoS zombies was reset thereby instructing the bots to generate non spoofed attack traffic. The Egress filtering schemes implemented on the edge routers of individual LANs were effectively incapable of detecting any malicious activity and thereby stopping the attack traffic from leaving the system. The (Access Control List) ACL defined for these routers using egress filtering cannot detect any anomaly when the attack is carried out using non spoofed attack traffic.

In order to overcome these limitations and effectively stop even the non spoofed attack traffic we use Automatic usage capping. The basic idea behind usage capping is to block traffic coming in from a particular IP address if the number of requests generated from that IP address crosses a particular threshold within a specified period of time.

Usage Capping Implementation

The threshold for number of requests that can be allowed from a particular IP address can only be effectively set at the server or very close to the server depending on the type of services being offered by the server and the current traffic scenario at the server. Hence this DDoS attack prevention strategy was implemented on the attack victim itself.

The way this prevention strategy is implemented is by recording the IP address of all the HTTP requests coming in at port 80 of the victim server. One of the apparent short comings of this proposed DDoS prevention scheme which immediately became evident during implementation was that the log size for the list of all the IPs requesting access to the server soon ballooned to a large size and a lot of time and computation was wasted on maintaining and updating the list.

A quick solution to the problem of over-sized log, which worked out really well was flushing the list at the end of each threshold time period and starting a new log, The flushing of the log is done only after the list of defaulting IPs is extracted from the it. These IP addresses are added to the ACL of the firewall running on the victim server to simply drop all traffic coming in from these IP addresses.

Results of Usage Capping Experiment

The automatic usage capping scheme was able to block non spoofed attack traffic which we were not even detected by egress filtering scheme. Though this defense mechanism is capable of detecting the source and hence defending against DDoS attacks in a much better way than Egress filtering. there is however a subtle limitation which is that this scheme can not instantaneously categorize an IP address as a malicious IP and block it. It has to wait and observe till the number of requests from a particular IP address crosses a threshold number in a threshold time interval before the IP can be categorized as an attack source. During this window of indecision there was a substantial surge in attack traffic to the victim. Though this defense scheme was not able to limit the attack during

its initial stages, it effectively dropped a large number of attack packets on the basis of rouge IPs which it had identified during the initial phases.

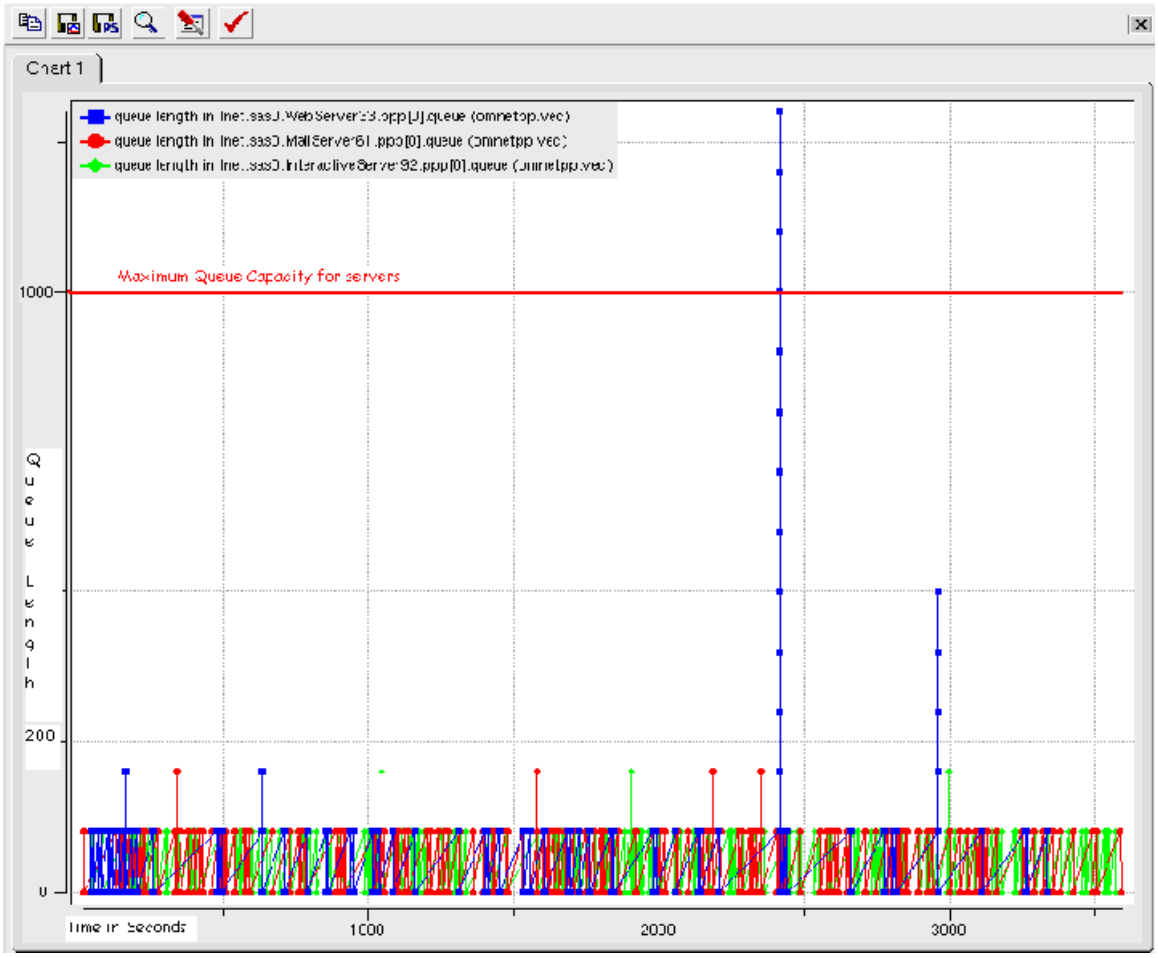

Figure 13: Graph showing DDoS attack happening on WebServer33 with Automatic usage capping (OmnetPP, 2008)

The figure above clearly shows the effects of Automatic Usage capping applied to webserver33. The Automatic usage capping though seems to be ineffective at first, it was able to limit the attack traffic once it marked all the rouge IP addresses.

10. Analysis of existing DDoS attack defense proposals:

Defense against DDoS attacks can be based on one of the following four approaches (Peng, 2007)

1. Attack prevention
2. Attack detection
3. Attack source identification and
4. Attack reaction

Attack prevention focuses on stopping an attack before it reaches its target, whereas *attack detection* aims at detecting a DDoS attack once it has began. (Peng, 2007)

Attack source identification is aimed at identifying the original attacker or source of attacks. Attack source identification strategies are generally used as deterrent to attackers rather than a prevention or mitigation strategy. (Peng, 2007)

Attack reaction aims to nullify the effects of DDoS attacks and it is generally the last line of defense against a full blown DDoS attack taking place. (Peng, 2007)

10.1 Attack Prevention:

Attack prevention as the name suggests tries to stop an attack before it reaches its target. This main assumption under which this approach works is that the attacks source address is spoofed, Based on this assumption a verity of packet filtering schemes are applied at router level to allow only packets with authentic addresses to pass through. (Peng, 2007)

This approach is effective in preventing DDoS attacks to a large extent since a lot of DDoS attacks depend on spoofed address to hide the real identity of the attacker.

Some of the common packets filtering schemes used are

10.1.1 Ingress/Egress filtering:

The *Ingress/Egress* filtering scheme filters packets entering or leaving the local network based on the authenticity of the packets address. (Peng, 2007) This filtering scheme is enforced at edge router either in the local network or at the Internet Service Providers place. (Peng, 2007) If the scheme is implemented in the local networks edge router, it effectively blocks all the outgoing packets with spoofed addresses similarly the edge router in ISPs network will block all the incoming packets from the given network which does not fall in the legitimate address range of that network. (Peng, 2007)

It is best to apply this filtering scheme at the edge router of the local network, as it is easier to determine what IP address can be expected from within this network, also edge routers at the leaf network are much less burdened as compared to routers at the ISPs end. (Peng, 2007). The Ingress/Egress filtering scheme can be further extended to be applied to protocol types, port numbers etc. (Peng, 2007).

The main limitation of this filtering scheme is that if the attack does not use spoofed addresses then this filtering scheme will fail. Botnets with huge armies of thousands of zombies do not even care to spoof the source address; in such cases the strategy adopted by the Ingress/Egress filtering will fail. (Antonatos, 2007)

10.1.2 Router based Packet Filtering:

Router based packet filtering scheme extends the idea proposed in the Ingress/Egress filtering scheme to the core of the Internet (Peng, 2007). The basic idea is that as in case of an edge router for a leaf network, the packets on a particular link on the core could

have originated only from a limited set of IP addresses. Any packet from outside this known set of IP addresses can be easily identified and singled out. (Peng, 2007)
Though this filtering scheme sounds quite efficient it has quite a few drawbacks. If there is a route change due to traffic congestion or a link failure it can result in a lot of legitimate packets get classified as being spoofed and hence getting blocked.
Furthermore for this filtering scheme to work efficiently it must be implemented on a substantial number of links on the internet (Peng, 2007), which is difficult to achieve.

10.1.3 Source address validity enforcement protocol (SAVE):

SAVE protocol tries to overcome the main limitation of router based filtering scheme i.e. inability to take care of dynamic routing changes (Peng, 2007). SAVE achieves this by constantly sending messages from source to all the destinations and constantly updating the routers in a manner similar to which the routing tables are updated.
The main limitations of SAVE is that it is ineffective without widespread implementation across the Internet and also inability to take care of non spoofed attack packets.

10.2 Attack Detection

After attack prevention the next step in protection against DDoS attacks is attack detection. *Attack detection* aims at detecting an attack once it has began. The efficiency of any attack detection scheme is determined by the percentage of attacks it can successfully identify. (Peng, 2007)
Degradation of performance of system is a strong indication of a DDoS attack (Peng, 2007), however a DDoS attack need not be based on software bugs, vulnerabilities in

system or protocol flaws and hence it is very difficult to distinguish it from legitimate traffic. A major challenge in attack detection is that of false positives because of this inability to easily distinguish between attack and legitimate traffic. (Peng, 2007) Flash crowds and traffic created due to Slashdot effect are all legitimate traffic but often times trigger false alarms.

There are a number of assumptions based on which most of the attack detection takes place some of these assumptions are as follows.

1. Most of the attack traffic does not follow traffic control protocols. (Peng, 2007)
2. There will be a traffic flow imbalance between attacker and the victim, if the victim is not able to respond to all the attack packets (Peng, 2007)
3. Attack traffic is created in a random pattern to avoid detection (Peng, 2007) and
4. For each kind of attack, we can observe a strong correlation between the attack traffic at target and abnormal behavior at source. (Peng, 2007)

But nailing down a DDoS attack based on these assumptions is really difficult. The availability of a large number of zombies in a botnet allows each zombie to mimic an authentic user as close as possible thereby avoiding detection based on these assumptions. (Peng, 2007)

Anomaly based attack detection schemes show considerable promise in their ability to detect new DDoS attacks, but the main problem with this approach is that of creating a normal traffic profile. (Peng, 2007)

10.3 Attack Source identification

Attack source identification is aimed at identifying the original attacker or source of attacks. Attack source identification strategies are generally used as deterrent to attackers rather than a prevention or mitigation strategy. (Peng, 2007)

Some of the common approaches used for attack source identification are

10.3.1 Backscatter trace back

The *backscatter trace back* approach to locate the source address of the attacker works on the assumption that the attackers use randomly spoofed IP addresses to conceal their original identity. (Peng, 2007) This approach makes use of a *sinkhole router*, which is a router configured in such a way that all the ICMP unreachable messages from the edge router of the ISP are forwarded to it. (Peng, 2007) An ICMP unreachable message is created whenever a packet is dropped by the edge router of the ISP because the target may be already down due to the attack or extremely slow in responding due to the excessive traffic.

The ICMP unreachable message contains the source address of the edge router that generated the ICMP unreachable message which effectively reveals the ingress point of the attacking traffic (Peng, 2007).

The main drawback of this scheme is that the attacker can simply stop using spoofed addresses and thus evade detection. Furthermore it is not a trivial task to set up a sinkhole router and expect that spoofed packet will use an IP address in the range which the sinkhole server is looking for. (Peng, 2007) Attack source identification is of little use in

case of DDoS as most of the Botnets do not use spoofed addresses to carry out the attack (Peng, 2007).

10.4 Attack Reaction

Attack reaction aims to nullify the effects of DDoS attacks and it is generally the last line of defense against a full blown DDoS attack currently underway.

The main focus of attack reaction is to protect bottleneck resources. A *bottleneck resource* could be a low bandwidth link or a network device which is not able to process huge amount of attack traffic fast enough. (Peng, 2007) A denial of service attack takes effect only when it overwhelms the target networks resources and the first one to run out of capacity in an attack are the bottleneck resources. So the main focus of the attack reaction schemes is to take care of the bottleneck resources so that at least partial service can be restored. (Peng, 2007)

Bottleneck resource management can be broadly classified into host resource management schemes and network resource management schemes. (Peng, 2007) Bottleneck resource management schemes tend to be applied close to the target host and the target network hence they are one of the simplest schemes against DDoS attack and most of the commercially available DDoS prevention products use bottleneck resource management. (Peng, 2007)

11. Suggestions to improve security against DDoS attacks

After studying in detail the various DoS and DDoS attack strategies used by the attackers, actual attack instances on websites and a host of attack prevention and mitigation proposals, here are some suggestions which could make a website and its network infrastructure much more resilient to DDoS attacks. In general implementation of these proposals on a wider scale could make the Internet as a whole much more resistant to DoS and DDoS attacks.

11.1 Protection against common DDoS attack strategies:

Attackers sometimes use networks with vulnerabilities to launch attacks on target victim, the network vulnerabilities get exploited and the resources of an innocent network are used by the attacker to his advantage. One of the classic examples of this is the ICMP flood attack. In ICMP flooding the attacker sends an ICMP echo packet with spoofed source address to vulnerable networks IP broadcast address, the result is that the echo packet is sent to all the hosts in the network. Now all the hosts in the network respond to the ICMP echo request by sending an echo reply to the source address in the spoofed ICMP echo request packet which was the IP address of the victim. You can prevent your network being used to carry out an ICMP flood attack by preventing an ICMP echo request being sent to the broadcast address of the network from a host outside the network or better still by default keeping ping requests to broadcast addresses completely off. Some simple measures like this can go a long way in preventing a DDoS attack and being fooled into becoming part of a DDoS attack.

11.2 Improvements to operating systems and other vulnerable software

Attackers often employ flawed implementation of protocols in OS and various other system and communication software to carry out a denial of service attacks. One classic example of this was teardrop attack which exploited the vulnerability in the earlier Microsoft Operating Systems and some older versions of Linux whereby improperly framed or overlapping IP fragments were sent to the victim thereby crashing it. (McDowell, 2008) and (Peng, 2007)

Attackers often exploit the vulnerabilities in peer to peer software to mislead peer hosts and wrongfully instructing them to connect to a victim computer, the victim will be flooded with thousands of connection requests within seconds resulting in a DDoS attack. (Peng, 2007)

All these kinds of attacks can be avoided if the operating systems and software which are intended for wide scale deployment are properly vetted and suitable tested for specific DDoS attack vulnerabilities.

11.3 Vigilance and initiative on the part of the ISPs

Internet service providers can play a very big role in making the Internet as a whole more secure and immune to DDoS attacks. ISPs can effectively implement ingress filtering at their edge routers to effectively stop all attack traffic with spoofed addresses. This will help in isolating and stopping the attack traffic as close as possible to the source.

11.4 Prevention of spread of Botnets

Attacks carried out by botnets are usually immune to a lot of DDoS attack prevention techniques because of the fact that most of the botnet based DDoS attacks do not use spoofed addresses.

Packet filtering based attack prevention and detection schemes fails to stop an attack which does not use spoofed addresses. Even anomaly based attack detection schemes are helpless against botnet based DDoS attacks as they closely resemble real traffic patterns and hence do not stand out as an anomaly.

Prevention is better than cure, rather than having to cope with the attacks carried out by botnets it is best to prevent the spread of botnets in the very first place.

11.5 Stopping the attack traffic as close as possible to the source.

Most of the DDoS prevention strategies in use right now focus on stopping the attack traffic at the target or in the immediate vicinity of the target. These may prevent the victim host from going offline but there is still a considerable impact on the performance of the target host and its network. The huge amount of attack traffic saturated the routers and low bandwidth links leading to degraded performance from all the hosts in the network.

11.6 Over provisioning.

Over provisioning is one of the simplest methods to alleviate the effects of DDoS attacks. Although it may sound too simple to be effective, over provisioning of

hosts and increased Internet connection bandwidth are some of the most effective DDoS attack mitigation strategies employed by major websites. (Peng, 2007) The main disadvantage of this approach is that it involves huge capital investments which not everyone can afford.

11.7 Challenge response system and automatic usage capping

Challenge response system is a good approach to effectively stop automated zombies from generating requests to systems which are meant to be used by humans. (Peng, 2007) The idea is to present a challenge which can only be solved by a human whenever a request comes in. Challenge response policy can be enforced when ever there is a sudden surge in traffic to a website, this can effectively neutralize any DDoS attack that may be taking place by preventing automated zombies from creating any more attack traffic.

Automated usage capping is a policy which is already being used by many free email service providers. It is also a reasonable policy to limit the number of requests which are generated per unit of time from a particular IP address.

Though any one of the above listed policies alone may be inadequate in dealing with the problem of DDoS attacks, a combination of policies from the above listed portfolio of approaches should be able to deal with most of the DDoS attacks.

12. Conclusions

There is considerable amount of research being done to find effective and fool proof methods for detection and prevention of DDoS attacks, Yet DDoS remains one of the single biggest threats to most websites. Lack of credible DDoS attack mitigation tactics is evident from the fact that most of the high profile websites still depend on over provisioning as their primary defense against DDoS attacks.

The suggestions made in this paper can go a long way in improving security against DDoS attacks but distributed nature of Internet and the lack of a central authority causes inability to enforce protocols and regulations which could have made DDoS attack very difficult. Making the Internet immune to DDoS attacks will also require considerable effort on the part of operating system and communication software developers to thoroughly test their products for any vulnerability which can be exploited by botnets and taking remedial actions if necessary.

ISPs because of their role on the Internet are in a unique position to prevent and limit the scope of any DDoS attack taking place, they can enforce ingress/egress filtering schemes on their edge routers which can easily block any spoofed attack traffic originating out of their network but there is no law enforcing ISPs to do so and with the lack of financial initiative very few of them actually do so. Even filtering out spoofed traffic is no longer enough to completely stop DDoS attack traffic generated using botnets as many of the newer botnets no longer use spoofed address to hide the identities of their zombies.

A complete eradication of DDoS attacks thus seems impossible unless wide ranging initiative is taken by the ISPs and software providers on this front.

.

References

Andras. V. (1997). PARAMETRIZED TOPOLOGIES FOR SIMULATION PROGRAMS. Paper presented at the *European Simulation Symposium,* Passau, Germany. *, 9th* Retrieved from http://www.omnetpp.org/downloads/docs/papers/cnds98-paramtop.pdf

Andras. V. (2005). *Omnet++, discrete event simulation system, user manual* Retrieved from http://www.omnetpp.org/external/whatis.php

Andras. V. (2008). *INET framework documentation and tutorial.* Retrieved 11/17, 2008, from http://www.omnetpp.org/staticpages/index.php?page=20041019113420757

Antonatos S., & Markatos. (2007). Honey@home: A new approach to LargeScale threat monitoring. *ACM, Worm'07*(November 02)

Collins, P., Shimeall, T., Faber, S., & janies, J. (2007). Using uncleanliness to predict future botnet addresses. Paper presented at the *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement,* San Diego, California. 93-104. doi:10.1145/1298306.1298319

Craig, S. (2007). *What is a botnet.* Retrieved 04/01, 2008, from http://www.naspa.com/botnets

DOE-CIRC. (1998). *I-031a: Malformed UDP packets in denial of service attacks.* Retrieved 11/17, 2008, from http://www.ciac.org/ciac/bulletins/i-031a.shtml

Gamer, T., & Scharf, M. (2008). Realistic simulation environment for IP-based networks. *ACM,* (OMNeT++ 2008 March 3)

Good OS. (2008). *Good O.S.* Retrieved 11/17, 2008, from http://www.thinkgos.com

McDowell, M. (2007). *National cyber alert system, cyber security tip ST04-015.* Retrieved 11/07, 2008, from http://www.us-cert.gov/cas/tips/ST04-015.html

Moheeb A., Jay Z., & Fabian M. (2006). A multifaceted approach to understanding the botnet phenomenon. *IMC'06,* Rio de Janerio, Brazil.

OmnetPP. (2008). *What is omnet++.* Retrieved 11/17, 2008, from http://www.omnetpp.org/external/whatis.php

Ono, K., Kawaishi, I., & Kamon, T. (2007). Trend of botnet activities. Paper presented at the *41st Annual IEEE International Carnahan Conference on Security Technology,* ottawa, Ont. 243-249. doi:10.1109/CCST.2007.4373496

Patrick Collins M., & Joseph B. (2007). Using uncleanliness to predict future botnet addresses. *ACM, IMC' 07*(October 24-26)

Peng, T., Leckie, C., & Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys, 39*(1) doi:10.1145/1216370.1216373

Puri, R. (2003). *Bots & botnet: An overview.* Retrieved 01/21, 2008, from http://www.sans.org/reading_room/whitepapers/malicious/1299.php

ReaSE. (2008). *ReaSE- realistic simulation environments for IP based networks.* Retrieved 11/17, 2008, from http://www.omnetpp.org/article.php?story=20080902024855846

Ryan V, . (2006). In Jhon A (Ed.), *Attack of the 50 foot botnet*. Alberta, Canada:

searchnetworking. (2007). *Edge router.* Retrieved 11/17, 2008, from http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci212031,00.html

Vinoo T., & Nitin J. (2007). Bot countermeasures. *Journal in Computer Virology,*

Wang, P., Sparks, S. & Zou, C. C. (2007). *An advanced hybrid peer-to-peer botnet.* Retrieved 01/28, 2008, from http://www.usenix.org/event/hotbots07/tech/full_papers/wang/wang.pdf

Wei, S., & Mirkovic, J. (2006). A realistic simulation of internet-scale events. Paper presented at the *Proceedings of the 1st International Conference on Performance Evaluation Methodologies and Tools,* Pisa, Italy. *, 180* doi:10.1145/1190095.1190131

Wikipedia. (2008). *Core router.* Retrieved 11/17, 2008, from http://en.wikipedia.org/wiki/Core_router

Wikipedia. (2008). *Gateways.* Retrieved 11/17, 2008, from http://en.wikipedia.org/wiki/Gateway_(computer_networking)

Wikipedia. (2008). *Slashdot effect.* Retrieved 11/17, 2008, from http://en.wikipedia.org/wiki/SlashDot_Effect