DDoS: design, implementation and analysis of automated model

Udaya Kiran Tupakula,* Vijay Varadharajan and Ashok Kumar Gajam

Information and Networked Systems Security Research,
Division of ICS,
Department of Computing,
Macquarie University, NSW-2109, Australia
E-mail: udaya@ics.mq.edu.au
E-mail: vijay@ics.mq.edu.au
E-mail: kumar@ics.mq.edu.au
*Corresponding author

Sunil Kumar Vuppala

Department of Electronics and Computer Engineering,
Indian Institute of Technology Roorkee (IITR),
Roorkee, Uttarakhand 247667, India
E-mail: sunitpec@iitr.ernet.in

Pandalaneni Naga Srinivasa Rao

Hewlett Packard,
450 Alexandra Road,
Singapore 119960
E-mail: srini.rao@hp.com

Abstract: Earlier, we have proposed an automated model to minimise DDoS attacks in single ISP domain and extended the model to multiple ISP domains. Our approach has several advanced features to minimise DDoS attacks in the internet. The focus of this paper is twofold: firstly, to present a detailed description of the design and implementation of the proposed model and second to discuss and analyse the extensive set of results obtained from the implementation and simulations. We describe the prototype implementation of our automated model using NetProwler network intrusion detection system and HP OpenView Network Node Manager. We will also discuss the performance analysis of our model on a large scale using NS2 tool. Both prototype and simulation test results confirm that our approach offers a promising solution against DDoS problem in the internet and the model can be implemented in real time with minor modifications to the existing tools.

Keywords: distributed denial of service; DDoS; automated model; intrusion detection; network management.

Reference to this paper should be made as follows: Tupakula, U.K., Varadharajan, V., Gajam, A.K., Vuppala, S.K. and Rao, P.N.S. (2007) 'DDoS: design, implementation and analysis of automated model', *Int. J. Wireless and Mobile Computing*, Vol. 2, No. 1, pp.72–85.

Biographical notes: Udaya Kiran Tupakula received a BE in Electronics and communication Engineering from the Gulbarga University, India in 1997 and an ME in Information Technology from the University of Western Sydney, Australia in 2001. He received a PhD in Computer Science from the Macquarie University, Australia in 2006 and currently working as a Research Fellow in Information and Networked System Security Research Group. His research interests include security in distributed systems, intrusion detection, mobile adhoc network security and sensor network security.

Vijay Varadharajan is the Microsoft Chair and Professor of Computing at Macquarie University. He is also the Director of Information and Networked System Security Research. He is the Editorial Board Member of several international journals including the *ACM Transactions on Information System Security* and the *Springer-Verlag International Journal of Information Security*. He has published more than 250 papers, has co-authored and edited eight books and holds four patents. His current areas of research interests include trusted computing, security in high speed networks and large distributed systems and mobile adhoc networks security. He is a Fellow of IEE, IEAust, ACS, BCS and IMA.

Ashok Kumar Gajam received a BE in Computer Science from the Karnataka University, India and the Master's degree in Information Technology from the University of Western Sydney, Australia. Currently, he is working as an IT Analyst in Tata Consultancy Services, India. His research interests include wireless technologies and location-aware wireless network security.

Sunil Kumar Vuppala received a BTech in Computer Science and Engineering from the Jawaharlal Nehru Technological University, India and an MTech in Information Technology from the Indian Institute of Technology, Roorkee (IITR), India. He is presently working as Junior Research Associate in Software Engineering and Technology Labs (SETLabs) of Infosys Technologies, Bangalore, India. His research interests lie primarily in mobile ad hoc networks, wireless sensor networks and network security.

Pandalaneni Naga Srinivasa Rao is a Senior Technical consultant in Hewlett Packard, Singapore. He has expertise in designing, developing and implementing technology solutions based on HP OpenView infrastructure solutions for Asia pacific customers.

1 Introduction

Denial-of-Service (DoS) (CERT, 1999, 2000) is an attempt by attackers to prevent access to resources by legitimate users for which they have authorisation. DoS attacks in the internet have become a pressing issue following a series of attacks during recent times. Several papers (Arce, 2004; CERT, 2000) have reported on the prevalence of such attacks on commercial servers and ISPs and the disruption they cause to services in today's internet. In case of Distributed Denial of Service (DDoS), an attacker compromises several hosts on the internet. Often, these are weakly secured machines and they are broken into using well-known defects in standard network service programs and common operating systems. These compromised computers are referred to as zombies. The attacker then uses these compromised computers to launch a coordinated attack on the victim machines. After a series of DDoS attacks that occurred in February 2000, there is an increased gap between the ease of generation of DDoS attacks and the proposed techniques to counteract DDoS attacks.

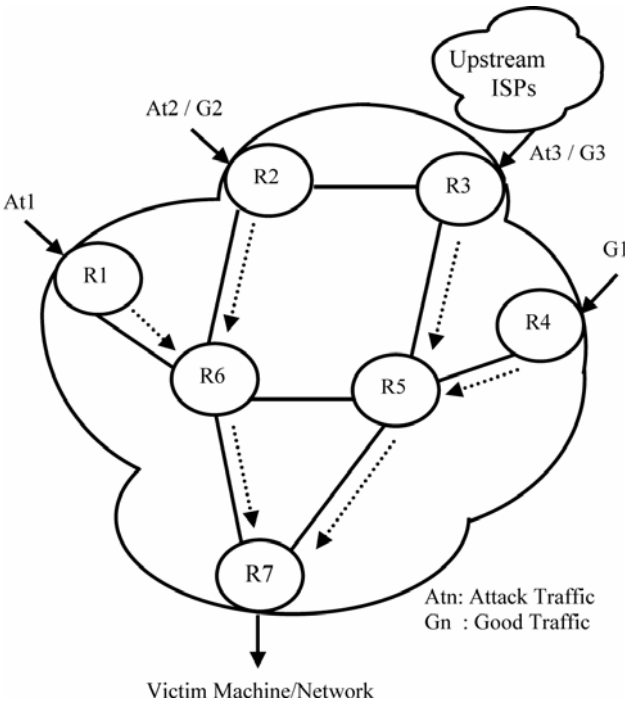
Earlier we have proposed (Tupakula and Varadharajan, 2003a, 2004) an automated model to counteract DDoS within a single ISP domain. We have compared (Tupakula and Varadharajan, 2003a, 2004) our automated model with several proposed techniques (Dean et al., 2001; Mahajan et al., 2002; Stone, 2000; Song and Perrig, 2001; Savage et al., 2000) and argued that our approach has several advantages compared to the other proposed techniques. We have identified different attacks that are possible on our automated model and proposed (Tupakula and Varadharajan, 2003c, 2004) techniques to minimise the identified attacks on our automated model. We have extended (Tupakula and Varadharajan, 2003b) our automated model to counter DDoS attacks in multiple ISP domains. A major advantage of our proposed model is its practical implementation. We have now developed a complete implementation of the proposed system and conducted extensive tests and obtained numerous results which we believe are significant and useful. Hence the focus of this paper is twofold: firstly, to present the detailed design and practical implementation of the previously proposed model and secondly, to discuss and analyse the extensive results obtained from the implementation and simulations to confirm the claims made by the model.

This paper is organised as follows. Section 2 gives an overview of our previous work. Section 3 discusses prototype implementation of our automated model using NetProwler Network Intrusion Detection System (NIDS) and HP OpenView Network Node Manager (NNM) and performance analysis on a large scale using ns2. Section 4 gives a general discussion and Section 5 concludes.

2 Our approach

In our previous work (Tupakula and Varadharajan, 2003a, 2004) we consider an architecture similar to (Stone, 2000) where the network is divided into ISP domains and customers connected to these ISP domains. In general, there are two types of routers in an ISP domain: internal routers and external routers. Internal routers belong to the ISP domain and external routers belong to the customers or another ISP. Internal routers themselves can be of two types, namely edge routers and transit routers. Edge routers are internal routers that are adjacent to one or more external routers. Transit routers are internal routers that are adjacent to other internal routers only. In Figure 1, R1, R2, R3, R4, R7 are edge routers and (R5, R6) are transit routers. While there is a chance for attack traffic to originate from the internal routers within the ISP's network domain, often most attacks (Stone, 2000) originate outside the ISP's domain, pass through the ISP domain and target the victim which is also outside the ISP's domain. Hence all the malicious traffic mostly passes through at least two edge routers of the ISP's domain. The malicious traffic passes through only one edge router of the ISP if both the victim's network and attacking source network are connected to the same edge router of the ISP. The edge router through which the attack traffic enters the ISP's network is called as ingress edge (R1, R2, R3, R4) router and the edge router through which the attack traffic exits from the ISP's network is called as egress edge (R7) router. The edge refers to adjacency between an edge router and an external router. We will focus our discussion on these types of external attacks.

Figure 1 ISP backbone network



In Figure 1, Atn represents total attack traffic generated by several attacking sources and Gn represents good traffic to the victim passing through a particular ingress router. Our aim is to identify and eliminate attack traffic (Atn) at the ingress edge routers (At1 at R1, At2 at R2 and At3 at R3) and provide more bandwidth to traffic from good sources (Gn) to the victim. We consider DDoS attacks with spoofed source addresses, broad attack signatures and attacking systems changing the type of attack traffic pattern. Attack signature refers to a pattern of traffic that is used to distinguish a malicious packet from normal traffic. Attack signature in our scheme refers to congestion signature in Mahajan et al. (2002). A detailed discussion on how to identify the attack signature based on some common properties in the attack traffic is considered in Mahajan et al. (2002). In many cases of DDoS attacks, since the source address of attack traffic is spoofed, the identified attack signature only includes the destination address of the victim and some property that is common for all the attack traffic. The main disadvantage of this approach is that the identified attack signature is common for all the attacking sources. If the source address cannot be spoofed, then it becomes easy to identify different attack signatures for different attacking sources. We proposed a packet marking technique that created a common ID *f*or all the traffic passing through a particular ingress router. With such a scheme, even if the source address of the attack traffic is spoofed, the victim is in a position to differentiate traffic from each ingress router and identify different attack signatures for each ingress router. In our model, the victim can identify the attack signatures with the same technique as discussed by Mahajan et al. (2002) or by deploying some security tools like firewalls/Intrusion Detection Systems (IDS) that can perform real time traffic monitoring.

Our architecture involves a controller-agent model. In each ISP domain, we envisage that there exists a controller, which is a trusted entity within the domain and is involved in the management of DoS attacks. In principle, the controller can be implemented on any internal (transit or edge) router or at a dedicated host. The agents are implemented on all the edge routers. The controller maintains the topology of its network and has information about all the agents that are present in the domain. An agent has the information of its domain controller only. Both the controller and agents are designed to handle the attacks on multiple victims simultaneously. To simplify the presentation of our model, we consider a simple scenario and make the following assumptions.

The controller is always available and any host is able to contact the controller at any time. The communication between the controller and its agents is protected and *no internal routers are compromised*. We have proposed secure communication protocols in Tupakula and Varadharajan (2003c, 2004), which we will not be discussing in this paper. The victim has some form of security tools like firewalls/IDS (Axent, 2002) or deploys (Mahajan et al., 2002) technique at the end system to identify the attack signatures. We define attacking source as the actual host or network that is connected to the edge router of the ISP in which our model is deployed. In case of attack traffic originating from upstream ISP customer domains, we consider attack host as the upstream ISP router, which is connected to the edge router of the ISP in which our model is deployed. An attacker may generate any packet. Now let us discuss the operation of our automated model.

The victim that is under attack requests the controller in its domain to prevent the attack upstream. A session is established between the controller and the victim after proper authentication of the victim. Depending on the number of agents present within its domain, the controller generates and issues a unique ID to each of its agents and commands them to mark the victim's traffic with the issued ID. The unique ID consists of controller ID and agent ID. The controller ID is common for all the agents and the agent ID is unique for each agent. The controller updates the victim with the valid IDs (common controller ID and unique agent IDs). The agents filter the traffic that is destined to the victim and mark the packets with the unique ID in the fragment ID field of IP packet (Postel, 1981). All the packets that are fragments and all the packets that are marked by an attacker will be dropped in this stage. Since the controller maintains the topology of its network, the marking command/process slightly varies for ingress and egress agents. If the ingress agent receives a packet that is not marked in the fragment ID field, then it marks the packet with the unique ID. If the ingress agent receives a packet that is already marked in the fragment ID field, then it drops the packet since the packet could be a valid fragment or it could be marked by an attacker. If the egress router receives a packet from an external router other than victims interface (any other customer networks connected to R7 of the ISP), the marking process is similar to the marking at ingress agents. If the egress router receives an unmarked packet (any packets originating in

backbone routers) from other internal router (R6, R5), the packet is marked with the egress agent unique ID and destined to the victim. If the egress router receives marked packets from other internal router (R6, R5), then the packets with valid controller ID are passed to the victim and the rest are dropped. Since agents are deployed on all the edge routers, all the traffic to the victim is marked with the ingress agent unique ID. Since agents are deployed only on the edge routers, any traffic originating in the backbone and any traffic originating from host/network which is connected to egress agent (R7) other than the victims network will be marked with the egress agent unique ID. Though the source address of the attack traffic is spoofed, for example in Figure 1, all the attack traffic At1 will have similar ID (which includes the controller ID and R1 unique agent ID) in the fragment ID field. Since the controller has already updated the victim with valid ID's, this enables the victim to identify different attack signatures for different attacking sources (At1 for R1, At2 for R2 and At3 for R3). The victim updates the controller with different attack signatures based on the unique ID. The controller retrieves the 32-bit IP address of the agent from its database based on unique ID and commands that particular agent to prevent the attack traffic from reaching the victim. Since attack signatures are identified based on the unique ID, only the agents through which the attack traffic is passing (R1, R2 and R3) will receive this command. Now all the agents that receive this command will start preventing the attack traffic from reaching the victim. The traffic that is matching with the attack signature will be dropped and logged at the agent. The traffic that is not matching with the attack signature is marked with the unique ID and destined to the victim. This enables the victim to easily track the changes in the attack traffic pattern. In Figure 1, since the packets are marked even if the traffic is not matching with the attack signature, the victim can easily detect and respond if the attack traffic At1 from R1 changes the attack pattern from *X* to *Y*. Also, since the packet marking is enabled at all the edge routers, the victim can easily detect and quickly respond if the good sources connected to R4 suddenly start sending attack traffic to the victim. The agents update the controller at regular intervals on how much attack traffic the agents are still receiving. Packet marking and prevention process will be done until the agent receives a reset signal from its controller. However the victim can request the controller to continue the packet marking process for an excess amount of time. This is very useful for intermittent type of attacks where attacking systems do not flood the victim continuously but send attack traffic at regular intervals.

We have also proposed an extension of the above model to counteract DDoS attacks in multiple ISP domains. We assume that multiple ISPs cooperate during the time of DDoS attack and proposed two different methods in Tupakula and Varadharajan (2003b). One approach assumes serial connection between different ISPs and other approach considers hierarchical architecture between ISPs and makes use of routing arbiter architecture (MERIT, 2002; USC/ISI, 2002). The prevention process in multiple domains is similar to the prevention process in single ISP domain except the unused bit in the flags field is enabled in this case to indicate that the packets are marked in other ISP domains. However, the model becomes more complex as the number of ISPs increases in case of serial multiple ISP model. We have compared (Tupakula and Varadharajan, 2003a, 2004) our model with several other proposed (Dean et al., 2001; Mahajan et al., 2002; Stone, 2000; Song and Perrig, 2001; Savage et al., 2000) techniques and shown that our model has several advantages compared to other proposed techniques. We have identified in Tupakula and Varadharajan (2003c, 2004) different attacks that are possible on our model and discussed techniques to minimise the identified attacks.

The main advantages of our model are as follows:

1 Agents function as ordinary routers when there is no attack.

2 Victim can easily identify the different attack signatures for each ingress agent.

3 Once victim is authenticated, identifying preventing and tracking any changes in the attack traffic is totally automated process.

4 Approximate source can be traced with a single packet and since all the packets are marked at ingress agent (nearest point to the attacking source), it becomes easy to initiate legal proceedings.

5 Filters are dynamically placed.

6 Once attack signatures are identified, prevention of attack traffic is directly near to the attacking source (ingress agent) instead of hop by hop upstream.

7 Each agent needs to check and prevent only the attack signature passing through it. The identified attack signature is very narrow and hence the delay experienced is very less.

8 There is no need for the agents and controller to identify the attack signatures. Hence implementation costs for controller and agents are less.

9 Fragmented packets destined to a potential victim are eliminated only during the times of attack. (Note a disadvantage is that some of the dropped fragments at the time of attack can be good packets).

10 Once invoked, the response time to identify and prevent the attack is fast.

We demonstrate many of these advantages in this paper including the performance and fast response times, using the implementation and results achieved.

3 Implementation and analysis

In this section, we demonstrate how our model can be implemented with the existing tools. Since the agent mechanism is invoked only during the time of attack and since only victim's traffic is filtered and processed separately, the impact on performance of the agent/router is much less compared to the other schemes. Our model

can be readily implemented with the existing Network Management Systems (NMS) such as HP OpenView (HP, 2003) or advanced IDS such as NetProwler (Axent, 2002) with minor modifications. All these tools easily lend themselves to our model as they are also based on Manager (Controller) and Agent principle. We now describe our prototype implementation using NetProwler NIDS, HP OpenView NNM and discuss performance analysis on a large scale using ns2 tool (NS2, 2003).

3.1 NetProwler NIDS prototype

The NetProwler (Axent, 2002) is an advanced network based NIDS. Its architecture includes console, manager and agents. The console provides graphical user interface to the NetProwler manager. The console lets the administrator to configure and manage all the agents from a single location, monitor attacks detected by the agents and generate/view reports. The manager is the mediator between console and agents. It conveys configuration information from console to the agents and alert messages from agents to the console. The manager maintains the database of all the known attack signature definitions and logs the alerts received from the agents.

The agent monitors the traffic in real time for suspicious behaviour on the network segment where it is installed. Agents are used to:

1 monitor network traffic between servers, clients and other network devices

2 detect network-oriented attacks and

3 respond to network attacks with preconfigured actions, such as terminating a session or hardening a firewall.

The NetProwler agent's graphical user interface lets the authorised users to view the agent's configuration and monitor network conversations in real time. Authorised user can also capture conversations from the agent GUI. However, no configuration operations on the agent can be performed from the agent GUI. The agent configurations can be performed from the console only.

NetProwler is capable of performing stateful dynamic signature inspection. Stateful means that NetProwler can remember the contents of the active sessions of the monitored network. Dynamic implies that the administrator can create new attack signatures and have them activated in real time without having to take the system offline. Signature inspection works by comparing an attack signature (a set of rules that describe an attack) to a communication packet.

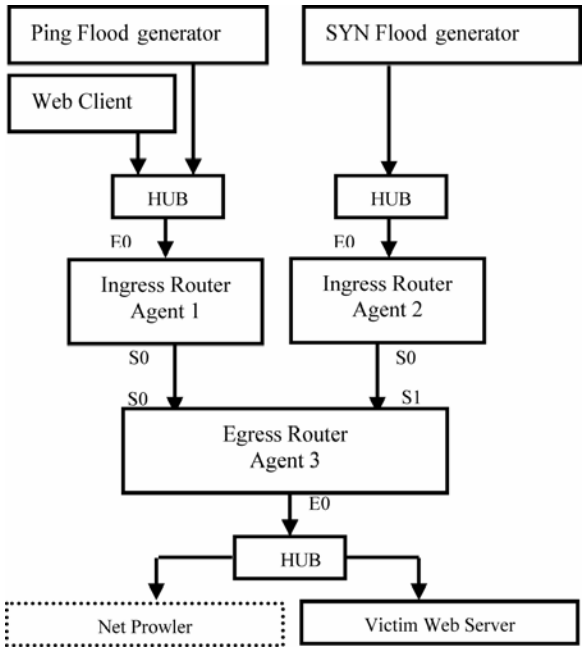
The following features are very favourable to suit our model: dynamic attack signature creation (or downloads from the vendor) and application of new attack signatures without taking the system offline. Each agent can harden a different device (even more than one). All agents can be configured only from the console. This is important because even if the attacker has access to the agents he cannot change any configurations. The communication between console/manager and the agents is encrypted. For the same attack signature different priority level can be assigned to different agents of the manager. For the same attack signature, different priority levels can be assigned to different monitored systems within the agent.

The main limitation of the NetProwler (and many of the NIDS) is that it can identify the attack signatures based on the source address or the destination address. Since the source address of the attack traffic can be easily spoofed, it is not efficient against DDoS attack. We modified the NetProwler to support implementation of our model.

3.1.1 Single ISP model

We have implemented the scheme with network servers, Cisco routers and NetProwler NIDS. Since we do not have access to modify the fragment ID field at the routers during the fly, we have used the Type of Service (ToS) field in the IP packet (Postel, 1981) for marking the packets with our packet marking technique. A NetProwler agent was used at the victim to identify the attack signatures and update the controller. Both the NetProwler agent and NetProwler manager (controller) were installed on a single machine. The NetProwler machine is shown in dotted line in Figure 2 since it operates in a promiscuous mode. The NetProwler agent maintains virtual database of attack signatures and monitors all the traffic passing through the network to which it is connected. In the case of attack on monitored machines, the NetProwler agent can be configured automatically to tighten a firewall and/or send an e-mail to the administrator and/or log all the traffic that is matching the attack signature and/or or reset a connection. In our case the NetProwler agent is configured to send an alert to the controller and log all the packets that are matching with the attack signature.

Figure 2 Single ISP prototype



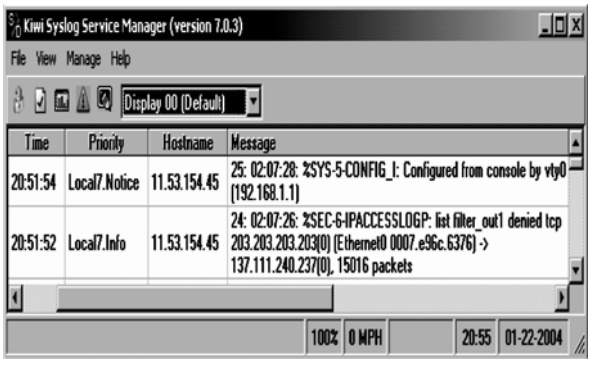
In this scenario, the NetProwler agent was configured to identify more than 10 ping requests or more than five concurrent TCP SYN requests per second as a high priority attack signatures. At time T1, we generated a

simple ICMP flood and TCP SYN flood (Phrack magazine, 1996) on the victim web server. As soon as the attack traffic was generated, the NetProwler agent detected the attack signature and sent an alert to its manager (controller) and logged all the traffic that is matching the attack signature. At this point, it should be noted that if we configure to block all the TCP SYN packets at the victim web server, then all the good TCP SYN packets from web client to the victim web server would also be dropped. Now our idea is to differentiate the traffic originating from different networks and identify different attack signatures for each customer network and provide access to the good traffic from web client to the victim web server, assuming that the source address of the attack traffic is spoofed.

At time T2 we configured the ingress routers/agents to mark all the web servers' traffic in the ToS field of IP packet (Note that this configuration is done automatically with our HP OpenView NNM implementation). The precedence field in the ToS field was used to mark the controller ID and the group field was used to mark the agent ID. The controller ID was set to 4. The ingress router agent 1 was configured to mark with the agent ID 1 and agent 2 was configured to mark the traffic with agent ID 2. So all the traffic originating from agent 1 was marked with the controller ID 4 and agent ID 1.

Similarly all the traffic originating from agent 2 was marked with controller ID 4 and agent ID 2. The NetProwler agent logged all the traffic. The logged data was analysed to identify the attack signatures. Most of the logged ICMP traffic was having agent ID 1 in the ToS field. So we have identified ICMP flood as an attack signature for agent 1. Similarly we could determine TCP SYN flood as attack signature for agent 2 because most of the logged TCP SYN packets were having agent ID 2 in the ToS field. At time T3, using Access Controls Lists (ACLs) in router, we have configured agent 1 to deny access and log all the ICMP traffic to the victim web server and agent 2 was configured to deny access and log all the TCP SYN traffic destined to the victim web server. The packets logged at the NetProwler agent decreased and the packets logged at ingress agents increased. The ingress agents were configured to log the dropped packets and send feedback to the syslog server (Kiwi, 2003). In this scenario, the syslog server was installed on the victim machine. Figure 3 shows the feedback message received from the ingress agents.

Figure 3 Feedback messages from agents



3.1.2 Multiple ISP model

Similarly we have implemented the multiple ISP model as shown in Figure 4. In Figure 4, a single router was used in ISP2 as both ingress and egress router but are displayed as separate routers for clarity. The model was tested over a period of time with the prototype model by varying the controller/agent IDs and attack signature/patterns. Controller ID was varied over a period of time to generate traffic from multiple ISP domains. For each controller ID/ISP, agent IDs were varied over a period of time to generate traffic from multiple agents within the ISP. We have developed a GUI to analyse the captured packets and identify attack signatures based on our packet marking technique.

The GUI was developed as an add-on tool, which can be integrated with most of the NIDS or NMS with minor modifications. The GUI gives the options to enter the IP address of the victim machine/network and also specify the log file for which attack signatures are to be identified. Figure 5 shows the identification of attack signatures based on our packet marking technique using our GUI. From Figure 5(a), it is clearly evident that the victim is experiencing TCP SYN flood and ICMP flood DDoS attack at its end. Figure 5(a) is similar to the output of NetProwler (and most of the NIDS). But the main disadvantage of this interface is that the victim can only identify a broad attack signature for all the attacking sources. As already discussed, if the firewall is tightened or filters are applied at the egress router of the ISP to drop all the TCP SYN traffic and ICMP traffic to the victim, all the good TCP SYN packets and good ICMP (if any) are equally penalised with the attack traffic.

Figure 4 Multiple ISP prototype

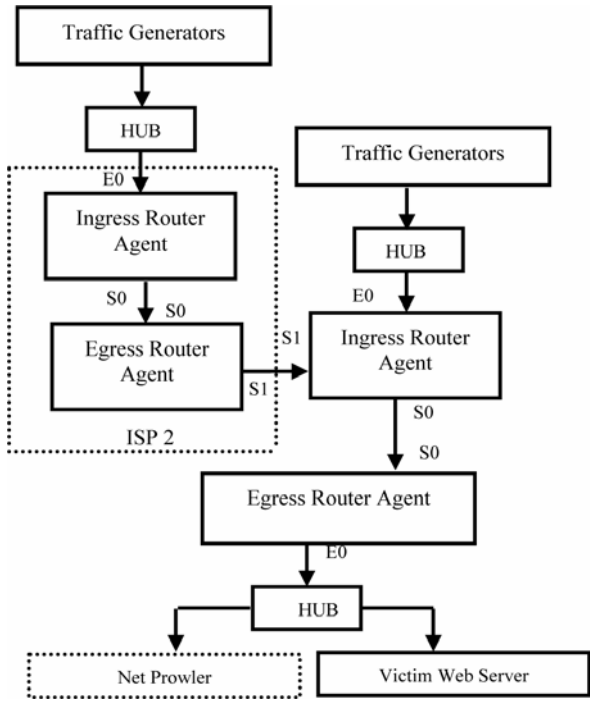


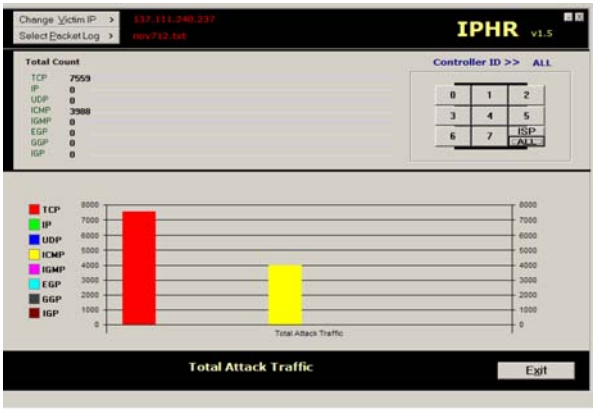
Figure 5(b) enables the victim to differentiate attack traffic originating from each ISP domain and identify attack signatures for each ISP domain. The victim is in a position to determine the total attack traffic originating from four

ISP domains. ISP1 and ISP4 are contributing to both TCP SYN flood and ICMP flood while ISP5 is only contributing to TCP SYN flood and ISP6 is only contributing to ICMP flood traffic. Now if the ISPs are willing to cooperate to prevent attack on the victim, ISP1 and ISP4 can block all the TCP SYN and ICMP packets that are destined to the victim. ISP5 can block all the TCP SYN packets and ISP6 can block all the ICMP packets that are destined to the victim. This enables access to the good ICMP traffic (if any) from ISP5 and good TCP traffic from ISP6 to the victim.

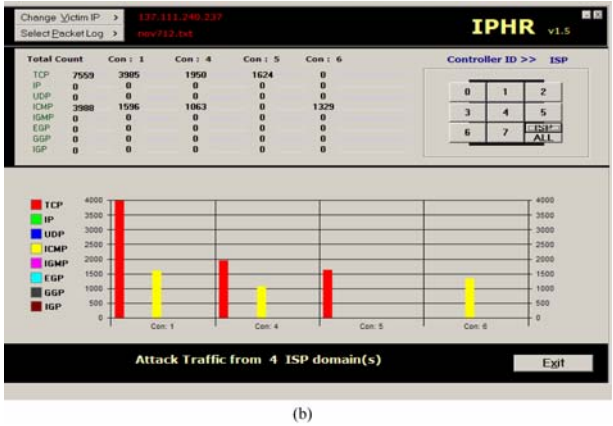
Figure 5(c) enables the victim to differentiate the attack traffic originating from each agent within the ISP domain and identify unique attack signature for each agent. Now the victim can determine easily that agent 2 is contributing to both types of attack traffic and agents 4 and 8 are contributing to only single type of attack traffic pattern and apply a filter to prevent only the attack traffic from reaching the victim. Figure 5(c) is efficient than Figure 5(b) since it enables the victim to determine the ingress agent of each attack packet and identify unique attack signature for each agent.

The GUI in Figure 5(b) and (c) can be easily integrated with the existing NIDS or NMS. With our technique, the victim is in a comfortable position to identify the total attack traffic at its end and easily differentiate between the attack traffic originating from each ISP domain and also from each agent within the ISP domain. Further the victim can identify the nearest router/agent with a single packet even in case of multiple ISP domains.

Figure 5 (a) Total attack traffic at the victim; (b) attack traffic from each ISP and (c) attack traffic from each agent within the ISP

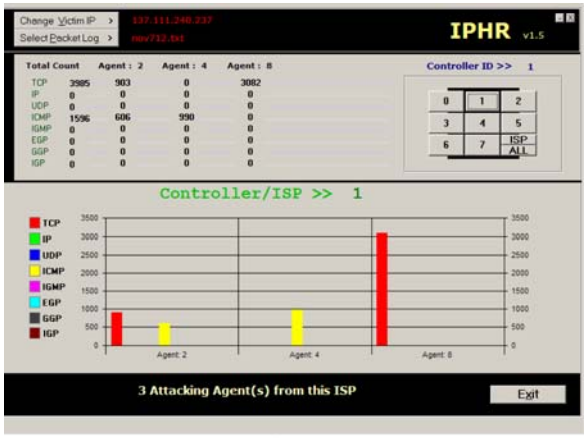


(a)



(b)

Figure 5 (a) Total attack traffic at the victim; (b) attack traffic from each ISP and (c) attack traffic from each agent within the ISP (continued)



(c)

3.2 HP OpenView NNM prototype

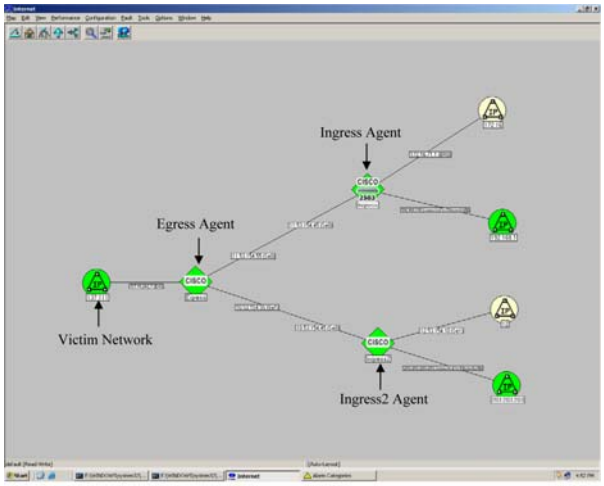
NNM is based on SNMP management model. The NNM architecture consists of network management station (manager/controller), agents (managed nodes) and SNMP as its standard communication protocol. The manager and agent can be installed on the same machine. NNM can be used to manage any objects such as severs, hubs, switches, routers, gateways, databases, etc., that are capable of supporting SNMP.

NNM can automatically detect the network topology by polling the nodes to detect the type and status of the objects. It has several other features such as keep a track of the network topology by discovering any new added objects and detect configuration changes. The administrator can apply thresholds and monitor several parameters of the managed object (e.g. disk space, CPU load, network traffic, collected MIB data, interface errors, data rate, number of packets per interface). NNM can be configured to perform automatic action if some events are detected on the monitored system. The events can be sent as alarms and this is also visible as a change in colour of the object where the event has occurred. This enables the administrator to easily locate the problem and quickly respond to the problem. The main purpose of the NNM is to ensure efficient operation of the network by automatically detecting the turbulences in the network. Though NNM can detect some abnormal conditions in the network, it should be noted that NNM is not designed to detect the DoS/DDoS attacks on the victim machines. We now discuss how NNM has been modified to suit our model.

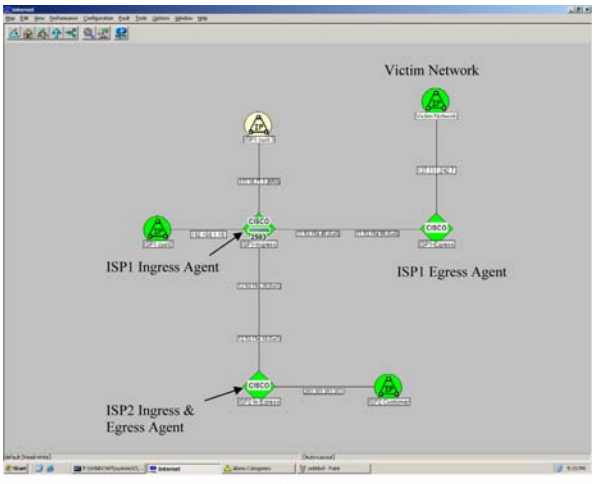
Figure 6(a) represents single ISP prototype and Figure 6(b) represents multiple ISP prototype. NNM was installed on a 2.2 GHz machine with 512 MB RAM running windows 2003 server. The NNM node is deployed within the victim's network. NNM was configured to automatically detect the network topology. The green colour shows that the device is enabled and working properly. The faded colour indicates that the interface is down. NNM was configured to monitor ICMP and TCP SYN flood attacks on the victim machine 'Manu' which is running a web server.

The victim machine (object of the NNM) was configured to send an alert to the NNM manager if the number of ICMP or number of half opened TCP connections reaches a particular threshold. If the NNM manager receives this alert from the victim node, the model is invoked and NNM manager sends the command to the ingress agents to mark the victim's traffic in the ToS field with the controller ID and unique agent ID. The automatic configuration code is written using Perl script. After marking is invoked, a new threshold is applied at the victim to monitor the attack traffic originating from each ingress routers. If the attack traffic originating from any of the ingress routers reaches a threshold, the victim sends a high priority alert to the NNM manager and requests to prevent the attack traffic at the identified ingress router. The NNM manager commands that particular agent to apply an ACL to prevent the attack traffic from reaching the victim. The ingress agent starts dropping the attack packets and sends feedback messages to the syslog server at regular intervals on how much attack traffic is being dropped at its end. This is similar to the messages shown in Figure 3. When attack traffic is filtered at the ingress agent, the victim notices a decrease in the attack traffic received at its end and sends a confirmation signal (a normal priority alert) to the NNM manager. The process is repeated recursively.

Figure 6 (a) Single ISP prototype and (b) multiple ISP prototype (for colours in Figures see online version)



(a)

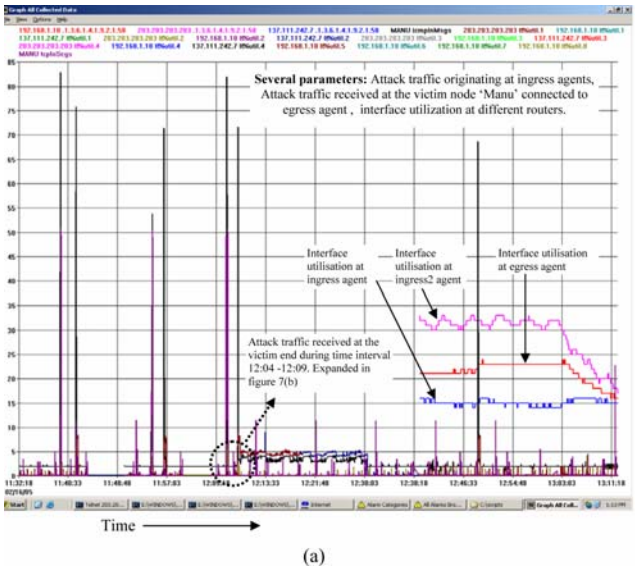


(b)

We have tested extensively our automated model using the NNM prototype and conducted performance analysis for the single ISP situation. Figure 7(a) represents different parameters, which are tested over a period of time. Figure 7(a) represents the CPU load, interface utilisation, bandwidth consumed by the attack traffic. Figure 7(b) represents the attack traffic at the victim and Figure 7(c) shows the alarms that are received at the NNM manager.

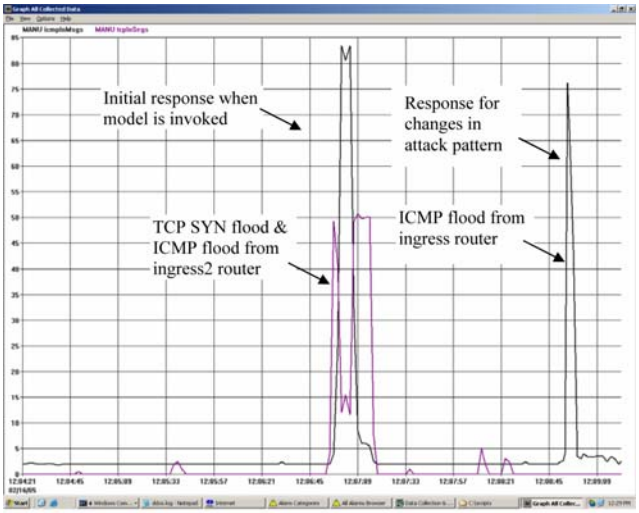
Let us discuss a single case of our analysis in detail. In the following scenario, the total process is completely automated except attack generation. We have generated TCP SYN flood (Phrack magazine, 1996) attack and ICMP flood attack at approximate time interval 12:06 from the network (203.203.203.X) connected to ingress2 router (see Figure 6(a)). The attacks were targeting the victim machine 'Manu' (network address 137.111.X.X) which is connected to the egress router. The victim notices a sudden rise (see Figure 7(b)) in the number of half opened TCP connections and ICMP messages received at its end and sends high priority alert signals to the NNM manager. The NNM manager received the high priority attack signals at time intervals 12:06:57 and 12:06:59 (see Figure 7(c)). As the NNM manager receives the alert from the victim (monitored object Manu), perl script is automatically invoked and configures both the ingress agents to filter the victim's traffic and mark the packets with the controller ID and unique agent ID in the ToS filed of IP packet. After marking is invoked, threshold is automatically applied to monitor the attack traffic originating from each ingress router. When the attack traffic from ingress 2 agent reaches the threshold, a filter is automatically applied to block the attack traffic (TCP SYN and ICMP) at the ingress 2 when the NNM manager receives an alert at time interval 12:07:05. The NNM manager receives normal priority alerts at time intervals 12:07:17 and 12:07:19 when the victim notices decrease in the attack traffic received at its end. The feedback messages from ingress 2 agent to the syslog server are similar to the Figure 3.

Figure 7 (a) All data; (b) attack traffic at the victim; (c) alerts at NNM manager; (d) different parameters; (e) throughput; (f) CPU load versus attack traffic and (g) interface utilisation at ingress router

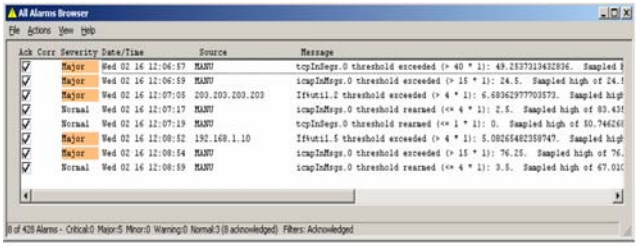


(a)

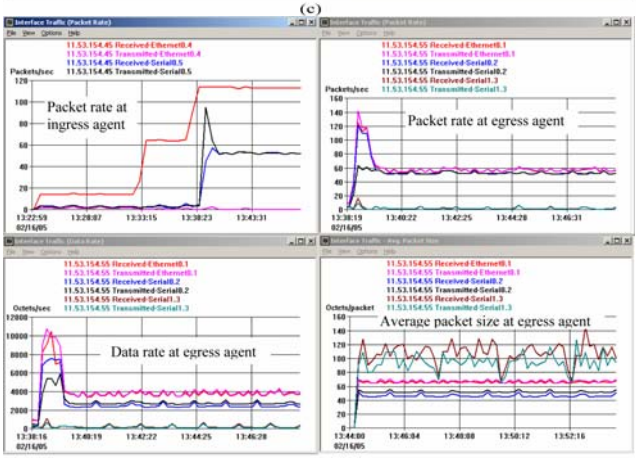
Figure 7 (a) All data; (b) attack traffic at the victim; (c) alerts at NNM manager; (d) different parameters; (e) throughput; (f) CPU load versus attack traffic and (g) interface utilisation at ingress router (continued)



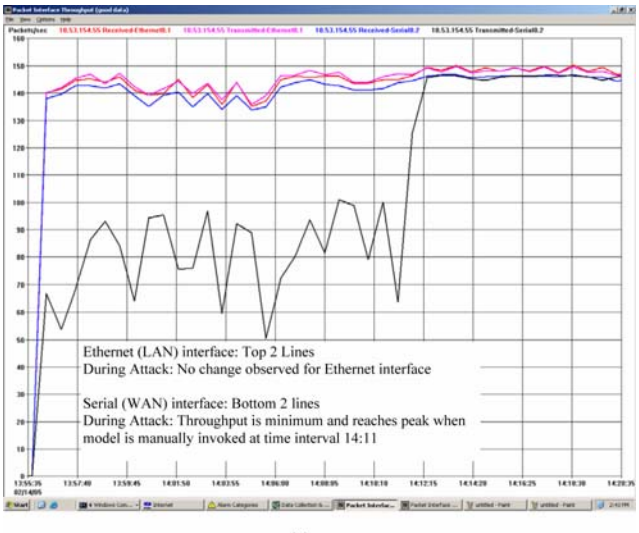
(b)



(c)

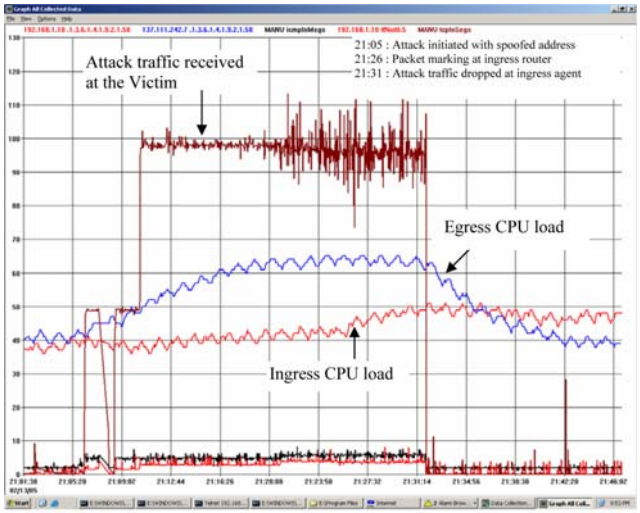


(d)

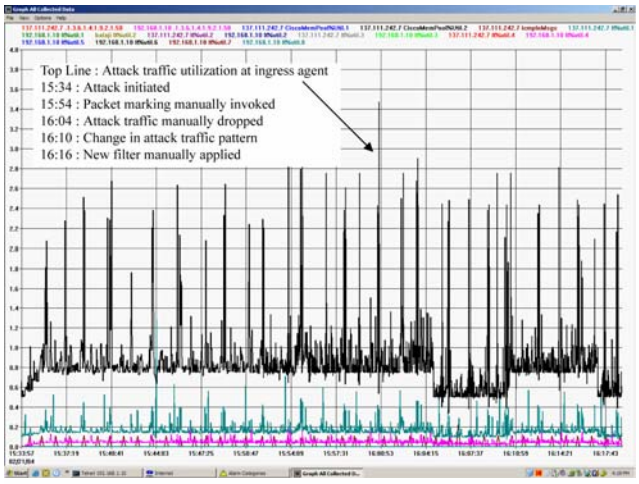


(e)

Figure 7 (a) All data; (b) attack traffic at the victim; (c) alerts at NNM manager; (d) different parameters; (e) throughput; (f) CPU load versus attack traffic and (g) interface utilisation at ingress router (continued)



(f)



(g)

Now we have manually generated ICMP flood attack from network (192.168.1.X), which is connected to ingress router. After packet marking is invoked, since attack traffic thresholds were individually monitored for each of the ingress agents, the NNM received high priority alert stating the ingress router and the attack signature (ICMP flood) at time intervals 12:08:52 and 12:08:54. The NNM manager commanded the ingress agent (see Figure 6(a)) to prevent ICMP traffic from reaching the victim. A normal priority alert was received at time interval 12:08:59 at the NNM manager when the victim machine 'Manu' noticed decrease in attack traffic received at its end.

The response time of our system is in the order of seconds and mainly dependent on how fast the victim can identify and respond to the changes in attack traffic. Since packet marking is already invoked, it helps the victim to easily identify the changes in attack traffic pattern for each agent and dynamically tune to the changes in the attack traffic pattern. In our analysis, after packet marking is invoked, we have assigned a fixed threshold to monitor the attack traffic originating from each ingress agent. In general, the thresholds for each ingress agent should be dynamically decided by considering several factors

such as the number of ingress routers, the resources available at the victim, severity and amount of the attack traffic, etc.

Since the response time is of the order of seconds, we could not study *automatically* the overhead on the routers due to marking and dropping of the packets. Further we could not notice any change in the throughput. So we have tested these parameters by *manually* configuring the NNM under different scenarios. The following results were generated between a single ingress agent and egress agent.

The top two graphs in Figure 7(d) shows the total traffic (good/attack) received at the ingress agents and total traffic received at the egress agents (just before the attack and after the invocation of our model). The two graphs below represent the data rate and packet rate received at the egress edge router. Figure 7(e) shows the throughput at the egress router. There was no change in the throughput of the ethernet (LAN) interface. The throughput of the ethernet (top two lines) interface was high even during the attack. The throughput of the serial (WAN) interface (bottom two lines) dropped to a minimum during the attack and reaches maximum when the model (attack traffic filtering) is manually invoked at time interval 14:11.

Figure 7(f) shows the CPU load on the ingress (top line) and egress (bottom line) routers before the attack, when the attack starts at time interval 21:05 and after our model is invoked. In Figure 7(f) packet marking is invoked at ingress agent during the time interval 21:26 and filter is applied at the ingress router during the time interval 21:31. Figure 7(g) shows the interface utilisation (top line) at the ingress router during the attack and after our model is manually invoked. In Figure 7(g) attack starts at 15:34, packet marking is manually invoked at 15:54, filter is manually applied at 16:04, attack traffic pattern is manually changed at 16:10 and a new filter is applied at 16:17.

Let us now consider some additional interesting observations from the results in Figure 7(f) and (g).

If the attack traffic is generated with spoofed source address, the additional overhead on the CPU load of the egress router is almost double compared to the additional overhead on the ingress router due to attack traffic. This is because the reply packets from the victim will be destined to random spoofed source addressed in the internet (instead of the attacking source). This may vary for different DDoS attacks.

There is additional overhead on the ingress routers due to packet marking technique and packet dropping. The overhead on the ingress router due to our packet marking technique and dropping of the attack packets is less compared to the gain in CPU processing power of the egress agent.

Though there is overhead on the CPU load due to packet marking and dropping, from Figure 7(g), it is clearly evident that there is a drop in the interface utilisation of the ingress router when attack traffic is filtered. This bandwidth can be used to route good traffic to the victim or other sources in the internet.

3.3 NS2 simulations

We also analysed our model in a large scale situation using ns2 tool (NS2, 2003). Consider again the Figure 1. The victim can be a single machine or network. There can be number of customers connected to the ISP. The attack traffic to the victim is originating from other customer networks connected to the ISP domain and also from the upstream ISPs/internet. As shown in the Figure 1, there can be some good networks within the single ISP which send only good traffic to the victim and bad networks which send only attack to the victim. Some may consist of both good and bad networks sending good/attack traffic to the victim machine/network. In the simulation, each edge router is connected to 10 traffic sources, which can contribute to good/attack traffic to the victim's network. We define few parameters before discussing the simulation results.

Response time (Rt): total time taken for marking and start of the dropping of attack packets at the ingress edge router of the ISP after our model is invoked. The response time for attack identification and prevention is mainly dependent on how quickly the victim can identify that it is under attack and how quickly the victim can identify the changes in attack traffic patterns.

Goodput: $Y/(Y \quad X)$ where Y is the total number of bad packets dropped at the ingress edge router after the invocation of the model. X is the bad packets that reached to the victim.

Percentage of overhead packets: the percentage (%) ratio of control packets transmitted in the simulation to the total bad traffic in the simulation. *Control packets* are of different types:

Victim to controller: attack alert to the controller, update controller with attack signatures.

Controller to edge routers: the mark and drop messages as requested by victim.

Edge router to controller: feedback messages at regular intervals for the attack traffic dropped at the agent.

Controller to victim: authentication of the victim, update victim with valid ID's, decision based on feedback for reset of the model.

3.3.1 Single ISP simulations

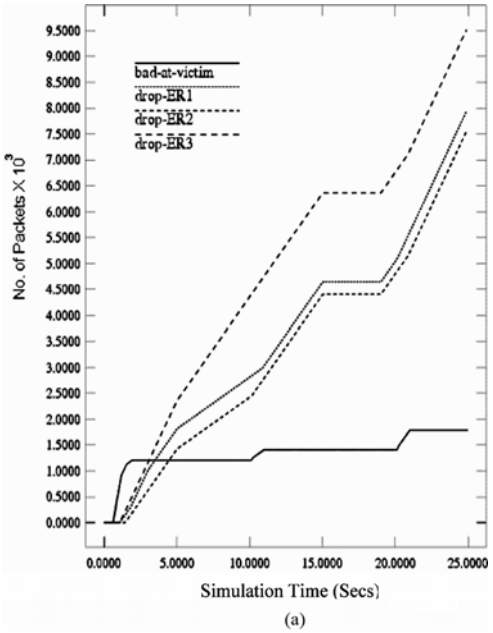
The simulation results for a single ISP model are shown in Figure 8. Figure 8(a) shows the simulation results for a single ISP for the architecture that is shown in Figure 1 and with path length equal to 27. As shown in Figure 8(a), the model is invoked when the threshold of the attack traffic at the victim reaches 1000 packets. The solid line in Figure 8(a) represents the attack traffic received at the victim. Once the attack packet reaches 1000 the victim sends a signal to the controller and the controller will issue the mark command to its agents. As soon as the packet marking starts the victim is in a position to identify different attack signature for each agent and prevention of attack is done only at the agents/routers through which

the attack traffic is passing that is, at edge routers R1, R2, R3. The agents start dropping the attack packets when it receives a command from its controller to prevent the attack packets. Since the victim identifies that the traffic G1 is good, agent R4 will not receive any command to drop the traffic. Hence the good traffic is completely protected. The dotted lines shows the attack packets dropped at the edge routers. Once dropping starts at the edge routers, the attack traffic received at the victim remains constant and the packets dropped at the ingress edge routers (ER1, ER2, ER3) increases.

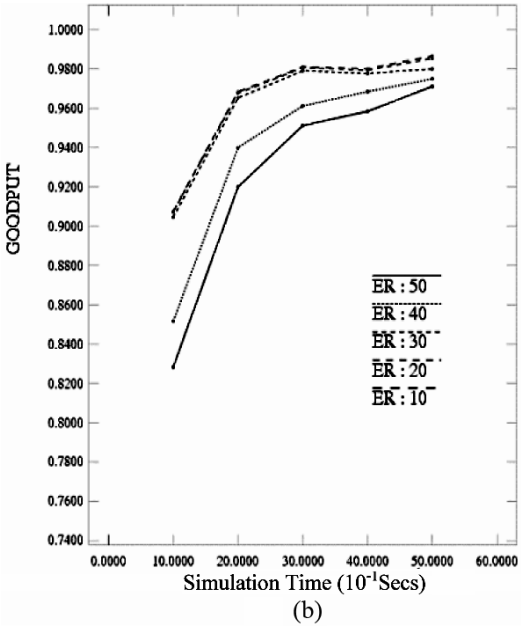
At approximately 10 sec the attacking sources change the attack traffic pattern, so there is increase in the attack packets received at the victim. The victim updates the controller with different attack signatures and the controller updates the agents with new attack signatures. This is response time of our model for change in attack traffic pattern. This rise indicates the dropping time only as marking is already active. Between, 15.0 and 19.0 sec all the curves rest in a flat zone representing the absence of attack traffic. After this stage again change of pattern results in the rise of the bad traffic at victim (Solid line). This shows response time for *intermittent attacks*. For every 3 sec feedback messages are sent from ingress edge routers to the controller indicating number of dropped packets in that interval. Figure 8(b) shows the graph of goodput versus simulation time when our model is invoked. As simulation time passes we can observe the increase in goodput. The parameter ER indicates number of ingress edge routers. As number of ingress edge routers increase, initial goodput is low due to the fact that invocation of model takes more time to stop all attack traffic. But as time passes we are able to get considerable goodput as high as 98.5%.

Figure 8(c) shows the graph of the percentage of overhead calculated in terms of packets. As the number of edge routers increase the overhead packets also increases and it depends to some extent on number of attack signatures. As number of attack signatures increase the model involves sending of more number of packets for the communication between the victim/controller/agents. Figure 8(d) shows the response time for varying path lengths. Path length is the number of hops between the victim and ingress edge routers. For each ingress edge router, there are 10 attached sources which may be sending good/bad traffic to the victim network. As the path length increases, response time also increases. There is slight change in response time due to change in number of ingress edge routers. This shows linear dependency. This is very advantageous when compared to other proposed techniques (Dean et al., 2001; Savage et al., 2000) where response time to detect the approximate source of attack itself consumes considerable amount of time and is exponentially dependent on number of attacking sources and the path length.

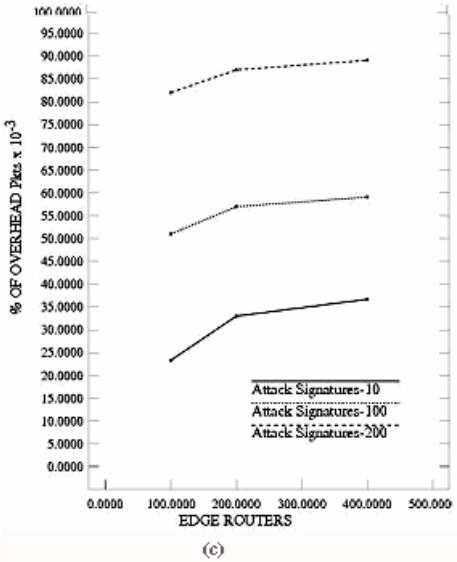
Figure 8 (a) Single ISP operation; (b) goodput; (c) percentage of overhead packets; (d) response time for varying path length and (e) response time for multiple paths



(a)



(b)



(c)

Figure 8 (a) Single ISP operation; (b) goodput; (c) percentage of overhead packets; (d) response time for varying path length and (e) response time for multiple paths (continued)

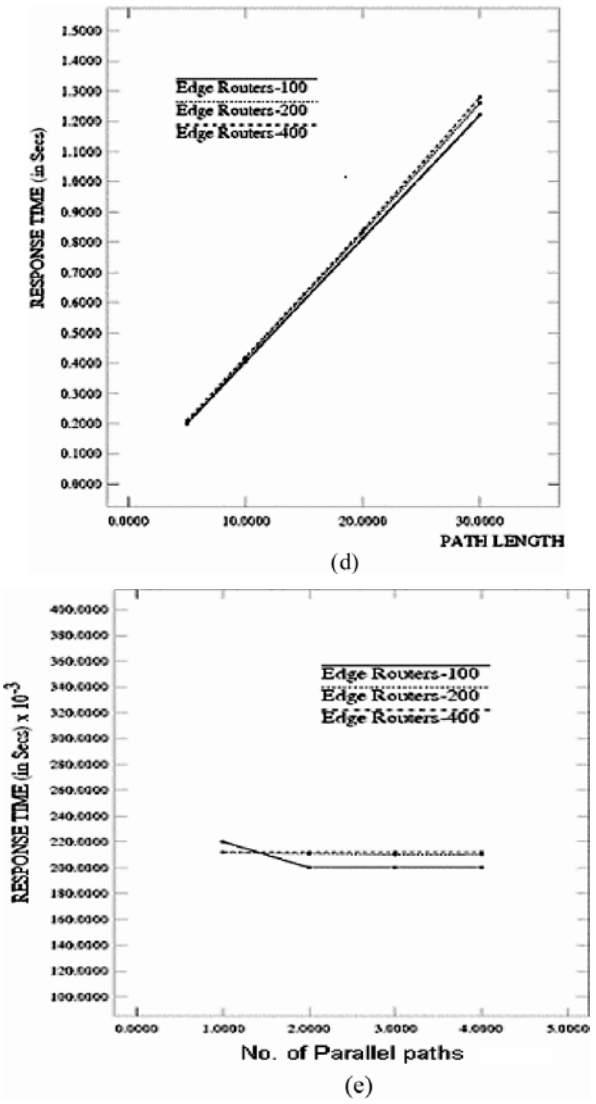


Figure 8(e) shows the response time for multiple paths. Parallel paths are between the victim and ingress edge routers. As number of parallel paths increases, response time slightly decreases or remains constant. This is due to availability of better path. There is no significant change in response time due to change in number of ingress edge routers.

3.3.2 Multiple ISP simulations

We have performed simulations for multiple ISP model for two different scenarios. One scenario considers an architecture where the ISPs are serially connected and the second scenario considers a hierarchical approach. Figure 9(a) and (b) show the results for ISPs connected serially. Figure 9(a) shows working model for multiple ISP with 4 ISPs with 10 ingress edge routers for each ISP and 10 traffic sources for each ingress edge router. The solid line indicates the bad traffic received at victim. It starts with a sharp growth in the number of bad packets received due to the threshold limit and the time taken to invoke the model. Other dashed lines represent total packets dropped at the ingress edge routers at different ISPs. After threshold value (1000 packets) the model is invoked and it consumes more time to start dropping of attack packets at all the ISPs. The rise in solid line indicates the total time taken for packet marking and dropping of the attack packets at the ingress edge routers of different ISPs.

At 5, 10 and 20 sec we can observe the rise in bad packets at victim (solid line) due to *change in attack traffic pattern*. The attack pattern continuously changes between the time intervals at 5 and 10 sec. The attack packets dropped at the ingress edge routers increases steeply. This is the response time for change in attack traffic pattern. After this stage, there is no much significant rise in the number of bad packets captured at victim and it tends to remain constant. For every 10 sec feedback messages are sent from ingress edge routers to the controller indicating number of dropped packets in that interval.

Figure 9(b) shows the graph of % of overhead calculated in terms of packets. As the number of ISPs increase the overhead packets also increases and it is linearly dependent on the number of attack signatures. Figure 9(c) shows the graph of goodput. 'S' represents serial model of multiple ISP and 'H' is for hierarchical model. As simulation time passes we can observe the increase in goodput. The parameter ER indicates ingress edge routers. It is evident that hierarchical model offers better goodput than the serial multiple ISP model.

Figure 9(d) and (e) represent the response time for hierarchical and serial model of multiple ISP. As number of ISPs increases path length also increases. As number of ISPs increase, response time also increases. This shows linear dependency. There is slight change in response time due to change in number of ingress edge routers. From Figure 9(d) and (e), we can see that hierarchical model offers better response than the serial multiple ISP model.

Figure 9 (a) Multiple ISP operation; (b) percentage of overhead packets; (c) goodput; (d) response time for hierarchical model and (e) response time for serial model

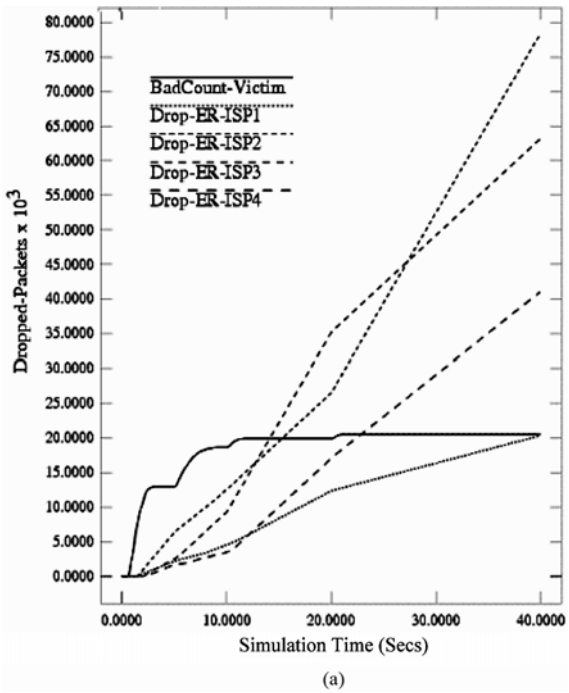
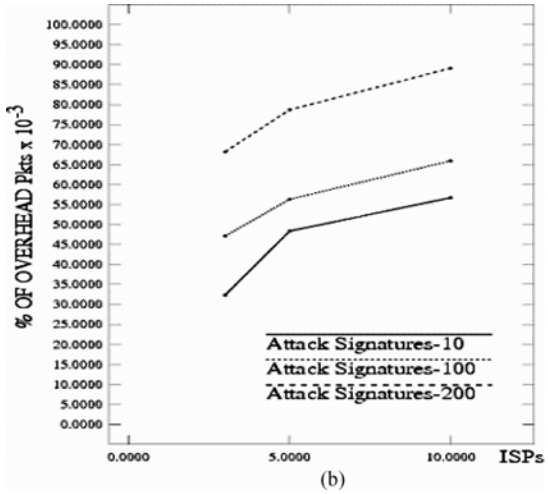
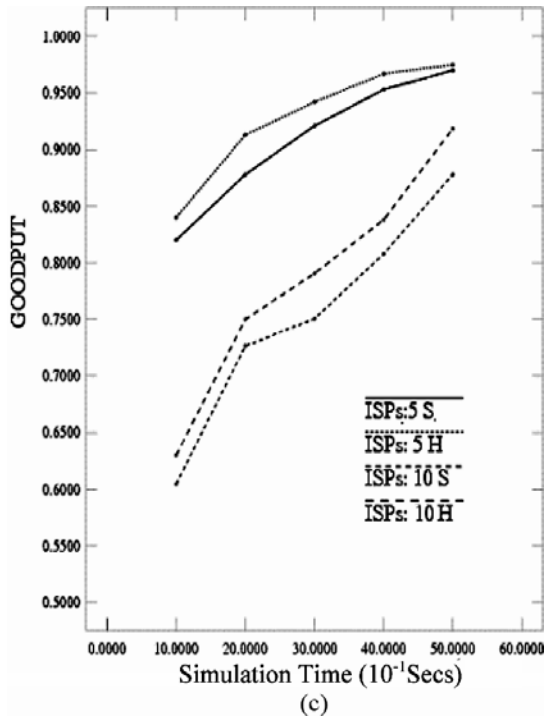


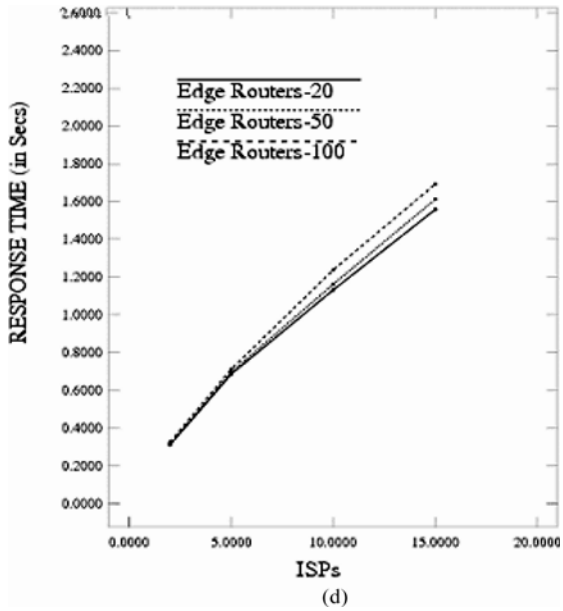
Figure 9 (a) Multiple ISP operation; (b) percentage of overhead packets; (c) goodput; (d) response time for hierarchical model and (e) response time for serial model (continued)



(b)

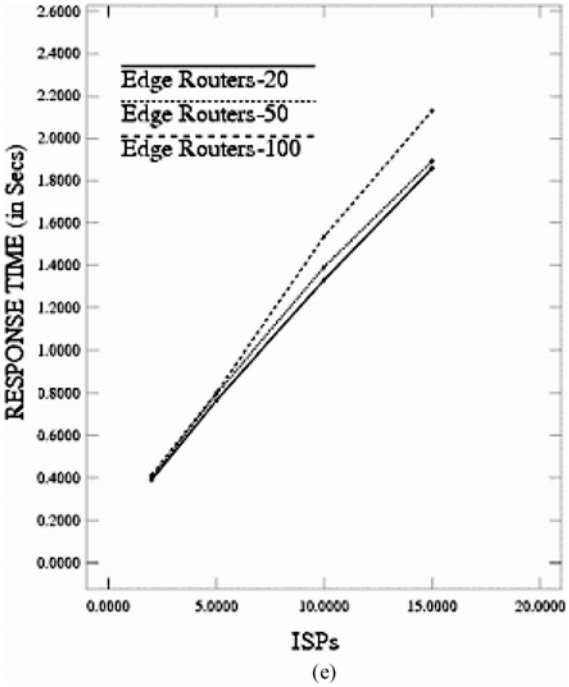


(c)



(d)

Figure 9 (a) Multiple ISP operation; (b) percentage of overhead packets; (c) goodput; (d) response time for hierarchical model and (e) response time for serial model (continued)



(e)

4 Discussion

We have shown that our proposed model can be easily integrated with NetProwler NIDS and HP OpenView NNM and practical implementations can be achieved. Though there is additional overhead due to marking and dropping of the packets, it should be noted that the overhead is minimal and only during the times of attack. Furthermore, the overhead on the ingress edge router is a gain to transit routers in addition to the gain in egress edge router. It should be noted that we have tested our model on older Cisco 2500 series routers. So our approach will be even more efficient if implemented on the current routers since the current routers have advanced features.

The current routers are capable of performing the ACL operations at line speed. For instance, Cisco claims that the Parallel Express Forwarding (PXF) technology (PXF, 2002) which is supported by its routers has several advanced features to support maximum forwarding performance with flexible options. ACLs can be processed at rates above two million packets per second. In addition, some routers have features to improve the ACL performance through hardware using an improved ASIC design or through microcode enhancement in packet switch. Hence the ISP need not worry about the overhead on their routers due to our model.

5 Conclusion and future work

In this paper, first we have given a brief overview of our model described previously and highlighted the advantages of our model over currently existing schemes. The focus of this paper has been twofold:

1 to demonstrate the practical implementation of the model and get extensive results from the implementation and simulations and
2 to analyse the results to show that they match with the claims made by the model.

We have discussed the prototype implementation of our automated model using NetProwler NIDS and HP OpenView NNM. We have developed a GUI to identify attack signatures based on our packet marking technique. The main advantage of the GUI is that it readily integrates with the existing NIDS or NMS.

Results from the prototype and NS2 simulations confirm that our approach is very promising solution against DDoS attacks. To the best of our knowledge we believe that our system offers several unique advantages over other existing schemes. In particular, the response time of our model is fast and adapts to changing attack traffic patterns. The complete design of the automated model and the analysis of its characteristics are described in the thesis (Tupakula, 2005).

In our future work, we will be investigating to extend this model to counter DoS/DDoS in some of the wireless network architectures. Wireless nodes have extremely limited resources and are easily susceptible to DoS/DDoS attacks. To counter DoS/DDoS attacks in wireless networks, the technique should have minimal overhead on the wireless infrastructure, have very fast response to identify and prevent the attack nearest to approximate source. We believe that our approach could provide some useful possibilities to counteract DoS/DDoS attacks at least in some of the wireless network architectures.

Acknowledgement

Vijay Varadarajan and Udaya Kiran Tupakula would like to thank Australian Research Council and Shearwater Solutions (previously Global Business Solutions) for their financial support for conducting this research project.

References

Arce, I. (2004) 'More bang for the bug: an account of 2003's attack trends', *IEEE Security and Privacy*, Vol. 2, No. 1, pp.66–68.

Axent (2002) *NetProwler*, Available at: http://www.axent.com/Axent/Products/NetProwler.

CERT (1999) 'CERT advisory CA-1999-17 denial-of-service tools', Available at: http://www.cert.org/advisories/CA-1999-17.html.

CERT (2000) 'CERT advisory CA-2000-01 denial-of-service developments', Available at: http://www.cert.org/advisories/CA-2000-01.html.

Dean, D., Franklin, M. and Stubblefield, A. (2001) 'An algebraic approach to IP traceback', *Proceedings of the NDSS*.

HP (2003) 'HP OpenView network node manager: managing your network with HP OpenView network node manager', Available at: http://ovweb.external.hp.com/ovnsmdps/pdf/t2490-90004.pdf.

Kiwi (2003) 'Kiwi Syslog Daemon', Available at: http://www.kiwisyslog.com/products.php.

Mahajan, R., Bellovin, S.M., Floyd, S., Ioannidis, J., Paxson, V. and Shenker, S. (2002) 'Controlling high bandwidth aggregates in the network', *ACM CCR Journal*, Vol. 32, No. 3, pp.62–73.

MERIT Network Inc. (2002) *Route Server Next Generation Project*, Available at: http://www.merit.edu/networkresearch/projecthistory/routeserver/index.php.

NS2 (2003) *Network Simulator*, Available at: http://www.isi.edu/nsnam.

Phrack magazine (1996) Issue 48-File 13of 18, 'Project Neptune', Author: daemon9/route/infinity for Phrack Magazine, July 1996, Guild Productions, Kid, Available at: http://www.fc.net/phrack/files/p48/p48-13.html.

Postel, J. (1981) *Internet Protocol*, RFC 791.

PXF (2002) 'Cisco Systems, Parallel express forwarding on the cisco 1000 series', Available at: http://www.cisco.com/warp/public/cc/pd/rt/10000/prodlit/pxfw_wp.pdf.

Savage, S., Wetherall, D., Karlin, A. and Anderson, T. (2000) 'Practical network support for IP traceback', *Proceedings of the ACM SIGCOMM Conference*, pp.295–306.

Song, D. and Perrig, A. (2001) 'Advanced and authenticated marking schemes for IP traceback', *Proceeidngs of IEEE INFOCOM*.

Stone, R. (2000) 'CenterTrack: an IP overlay network for tracking DoS floods', *Proceedings of the Ninth Usenix Security Symposium*.

Tupakula, U.K. and Varadharajan, V. (2003a) 'A practical method to counteract denial of service attacks', *Proceedings of the 25th Australasian Computer Science Conference ACSC2003*, Australia, pp.275–284.

Tupakula, U.K. and Varadharajan, V. (2003b) 'Counteracting DDoS attacks in multiple ISP domains using routing arbiter architecture', *Proceedings of the 11th IEEE ICON Conference*, Sydney, Australia.

Tupakula, U.K. and Varadharajan, V. (2003c) 'Analysis of automated model against DDoS attacks', *Proceedings of Australian Telecommunications Networks and Applications Conference, ATNAC2003*, Melbourne, Australia.

Tupakula, U.K. and Varadharajan, V. (2004) 'Tracing DDoS floods: an automated approach', *Journal of Network and Systems Management*, Vol. 12, No. 1.

Tupakula, U.K. (2005) 'Modeling, design and implementation techniques for counteracting denial of service attacks in networks', *PhD Thesis*, Macquarie University, Australia.

USC/ISI (2002) 'Brief summary of ISI's contributions to the Routing Arbiter effort', Available at: http://www.isi.edu/div7/ra/Publications.