

# Counteracting TCP SYN DDoS Attacks using Automated Model

Udaya Kiran Tupakula      Vijay Varadharajan      Ashok Kumar Gajam

Information and Networked Systems Security Research

Division of Information and Communication Sciences

Macquarie University, Sydney, Australia-2109

{udaya, vijay, kumar}@ics.mq.edu.au

**Abstract- We propose modifications to the automated model to counteract TCP SYN Distributed Denial of Service (DDoS) attacks nearest to the attacking source and also discuss the prototype implementation of our technique. It should be noted that we are not solving the TCP SYN problem, but we are enabling the victim to differentiate between the traffic originating from good and bad network domains, trace the router that is nearest to attacking source with a single packet even if the source address of the packet is spoofed and prevent the attack traffic at the router which is nearest to the attacking source. Since our model is invoked only during attack times, it has very less overhead and the main advantage of this technique is that the victim can provide better service for traffic originating from good network domain and completely eliminate or provide limited service for the traffic originating from bad network domain.**

## I. INTRODUCTION

Denial-of-Service (DoS) [1, 2] is an attempt by attackers to prevent access to resources by legitimate users for which they have authorization. In case of distributed denial of service (DDoS), an attacker compromises several hosts on the Internet and uses these compromised computers to launch a coordinated attack on the victim machines.

First we will give a brief overview on how TCP connections are made between two hosts and how TCP SYN flood attacks are performed [8]. A TCP connection should be established between the client and server before they can transfer data between them. Three steps are involved in establishing a connection between the client and server which is called the three way handshake process. First the client sends a SYN request to the server requesting a connection. The server then acknowledges the SYN message by sending a SYN-ACK message to the client. The client sends an acknowledgement to the server and this completes the connection between the client and server. Now the required data can be transferred between the client and server.

In case of TCP SYN flooding attacks, the attacker floods the victim server with the SYN packets by spoofing his 32-bit IP source address. Now the SYN-ACK packets from the server are addressed to the spoofed source, which are unable to respond to the SYN-ACK messages from the victim server. So the server will not get the final ACK from the client. These are called half open connections. The server has a data structure in its memory describing all the pending or half open connections. The attacker can create an overflow of the data structure by creating too many half open connections than the server can handle. If the victim can fill the victim server data structure with the half open connections, then the server cannot accept any new incoming connections. Typically a server maintains the state of half opened connections for upto 75 seconds. After the timeout, the half

open connections will expire. In case of distributed denial of service attacks, since there are many zombies it is very easy for the attacker to exhaust memory or crash the victim server even if it has very high resources such as memory and processing power. Even if we prevent the attack traffic at the victim's network boundary by configuring some security tools like firewall, the malicious SYN packets can consume all the bandwidth of the victim. It is difficult to differentiate between SYN packets originating from bad and good host/networks to the victim since every packet looks legitimate. The victim server can decide a SYN packet to be malicious only if it does not get a final acknowledgement from the client. So identification of attack signature is difficult for these kind of attacks. Attack signature refers to a pattern of traffic that is used to distinguish a malicious packet from normal traffic [3].

Earlier we have proposed [5] an automated model with many advantages to counteract DDoS attacks in single ISP domain and extended [6] the model to counteract DDoS attacks in multiple ISP domains. In this paper we will give a brief overview of our automated model and discuss how our model can be modified to counteract the TCP SYN flood attacks. We will also discuss the prototype implementation of our technique. We have chosen TCP SYN flood attacks as an example because SYN flood attacks have slightly different characteristics when compared to other DDoS flooding attacks. We have already discussed that there is no way to decide if a SYN packet is malicious until the server receives final ACK message from the client. We will show that the victim in our automated model can easily differentiate between the SYN packets that are originating from the malicious and benign network domains. Now the victim can provide better service to the clients originating from good host/network domains and completely eliminate or restrict the SYN packets originating from the malicious network domain to a particular bandwidth.

The paper is organized as follows. Section II discusses the related work. Section III gives an overview of our automated model and section IV discusses how our automated model can be modified to counteract TCP SYN flood attacks. Section V discusses the practical implementation and section VI concludes.

## II. RELATED WORK

Operating System (OS) vendors have provided [8] patches for their own OS to deal with TCP SYN flood attacks. Firewall and Intrusion detection system vendors have developed their own techniques to deal with these attacks. Some of the techniques that deal with the SYN flood attack include SYN Cookies [9], SYN Defender [10], SYN Cache [11], SYN Proxying [12] and SYN Kill [13]. All these

mechanisms can be implemented at the victim server or within the security tools (firewalls) at the victim's boundary network. Even if any of these mechanisms were deployed, it is shown in [15] that many of the commercial firewalls which are designed to counter SYN flood attacks are easily vulnerable to the SYN flood attacks if the attacker can generate 14,000 packets per second. Since there are many compromised systems in case of DDoS [1,2] attack, generating 14000 packets is a very simple task for an attacker. In order to counter these attacks effectively, the proposed techniques should counter the attack nearest to attacking source.

### III. OUR AUTOMATED MODEL

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 denial of service attacks. The controller can be implemented on a dedicated host and agents are implemented on all edge routers. The controller and agents are identified with their ID's. The controller assigns an ID for itself and a unique ID for each agent.

During the time of an attack, the victim requests the controller in its domain to prevent the attack. A session is established between the victim and the controller after proper authentication of the victim. Depending on the number of agents present within its domain, the controller will generate and issue the controller ID and unique agent ID to each agent and commands its agents to mark the traffic to the victim. Now the controller updates the victim with the controller ID and the unique agent ID. The agents filter the traffic that are destined to the victim and mark the traffic with controller ID and agent ID in the fragment ID field. Packets will be marked in such a way that only the first agent that sees the traffic will mark the packet. If an agent receives a packet that is already marked then it checks the packet for a valid controller ID. Packets with valid controller ID are passed and the rest are dropped. Since agents are deployed on all the edge routers, all the traffic to the victim is marked with the controller ID and the ingress/first agent ID in the fragment ID field. Since the victim knows the controller ID and valid agent ID, it can identify different attack signatures that are to be prevented at different agents. Now the victim updates the controller with different attack signatures based on the agent ID. The controller retrieves the 32-bit IP address of the agent based on agent ID and commands that particular agent to prevent the attack traffic from reaching the victim. As attack signatures are identified based on the agent ID, only the agent through which the attack traffic is passing will receive this command. Now all the agents that receive this command will start preventing and log the traffic that is matching the attack signature from reaching the victim. The traffic that does not match the attack signature will be marked with the controller ID and agent ID and destined to the victim. This is to enable the victim to easily track the changes in attack traffic. The agents will update the controller on how much attack traffic they are receiving. Prevention will be done until the agent receives a reset signal from its controller.

### IV. COUNTERACTING TCP SYN FLOOD ATTACKS USING AUTOMATED MODEL

Let us consider a simple SYN Flood attack as shown in Fig: 1. Here R1, R2 are transit routers and R0, R3, R4, R5, and R6 are edge routers. Let us consider that the attack is originating from the upstream ISPs or from other customers network domain of the ISP and targeting the victim. The dotted lines in Fig: 1 shows the direction of flow of the traffic to the victim V. Ati is the attack traffic from malicious sources and Gi is the good traffic from good sources to the victim. For simplicity, let us assume that the upstream ISPs do not co-operate with the ISP to prevent the attack on the victim. So in this scenario we can only prevent the attack at the ingress edge routers of the ISP.

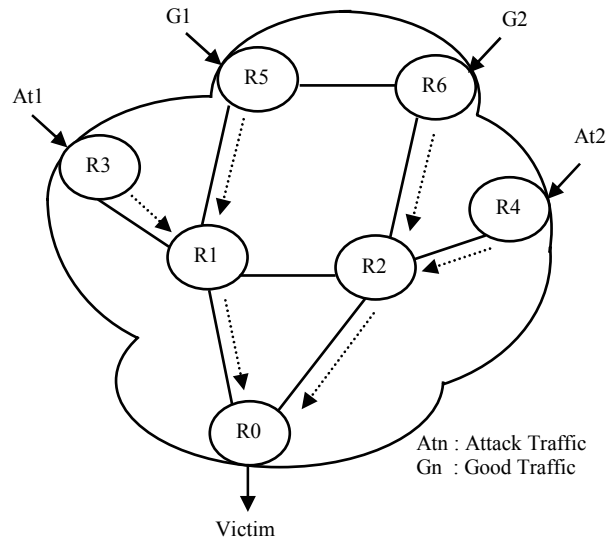


Fig 1: ISP backbone network

Our aim is to identify the attack at the victim, uniquely identify TCP SYN Flood as attack signatures for agents R3 and R4 and configure them to prevent the attack at the agents R3 and R4. At the victim, we should also be able to identify that the traffic from agents R5 and R6 is good traffic in order to provide better service to the hosts/networks G1 and G2. Let us assume that the controller is implemented on a dedicated host (not shown). During the time of attack the victim sends a request to the controller to prevent the attack. The controller issues a unique id (common controller ID and unique agent ID) for each of its agents and commands them to mark the traffic that is destined to the victim. The agents filter the victim's traffic and mark the traffic with the unique ID issued by the controller. Even if source addresses are spoofed, for example, all the attack traffic At1 will have a common ID of R3 (which is R3's unique agent ID) in the fragment ID field. Over a time  $\Delta T$  the victim can observe that it is not receiving the final acknowledgements for the SYN packets which have the unique IDS of R3 and R4. This simplifies the attack signature identification process at the victim. The victim can also notice that all the traffic from agents R5 and R6 is good traffic since it receives the final acknowledgements from the clients.

### Identifying Attack Signatures at the victim

For each agent ID for time  $\Delta T$

1. Count the number of SYN packets received with destination address equal to victim.
2. Count the number of ACK packets received with destination address equal to victim.
3. If total number of SYN minus ACK greater than specified number (for example 10 or 20) identify TCP SYN flood as attack signature for the particular agent ID.

This is very simple process of attack signature identification. However the victim can now efficiently use the techniques discussed by Mahajan et al. [4] to identify a narrow attack signature for each agent. This is because, without our packet marking technique there is no way for Mahajan et al. technique to determine if the packet is originating from good and bad hosts/network and every SYN packet appears to be legitimate. Now with our packet marking technique even if every SYN packet appears to be legitimate there is atleast one similarity (controller Id and unique agent ID) for all the SYN packets originating from malicious host/network. Mahajan et al [4] technique involves identification of congestion signatures based on some common properties of all the attack traffic and finding a very narrow attack signature so that it will not disadvantage other legitimate traffic which is matching the attack signature. So our packet marking technique greatly enhances the attack identification process of Mahajan et al. technique because, without our packet marking technique every SYN packet looks legitimate. With our packet marking technique, even if every packet appears to be legitimate all the packets originating from attacking host/network will have a common agent ID. Now we can apply Mahan et al. [4] technique and identify a very narrow attack signature for agents R3 and R4. Alternatively victim can deploy real time traffic monitoring tools like Symantec Manhunt [18] which can even detect zero day attack signatures. Now it is to the decision of the victim if it wants to completely eliminate the TCP SYN traffic or if it wants to provide limited service to the traffic originating from agents R3 and R4. Once the victim identifies the attack signature for each agent, it updates the controller with the attack signatures and requests the controller to completely eliminate or restrict the traffic originating from the identified malicious host/network. The controller retrieves the 32-bit IP address of the agent based on the agent ID and commands that particular agent to prevent or limit the TCP SYN from reaching the victim. The agents will start preventing/limiting the TCP SYN packets from reaching the victim. Packets that are not matching with the attack signature are again marked with the ID's and destined to the victim. This enables the victim to easily track the changes in attack traffic pattern and achieve quick response. The agents update the controller on how much attack traffic they are receiving. Prevention will be done until the agents receive a reset signal from the controller.

### V. IMPLEMENTATION

In this section, we demonstrate how our model can be easily implemented in practice with the existing routers. 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 many of the existing Network Management Systems (NMS) such as CiscoWorks[16], HP Openview[17] or advanced intrusion detection systems (IDS) such as NetProwler[18] or ISS Real Secure with minor modifications. All these tools easily lend themselves to our model as they are also based on Manager (Controller) and Agent principle.

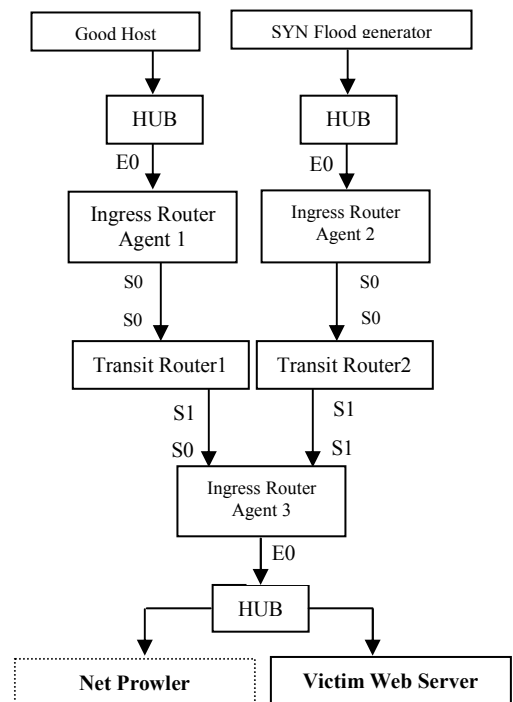


Fig 2: Implementation of our model

We have implemented the model with Cisco 2500 series routers and Netproowler advanced network based intrusion detection system. Packets were marked in ToS field instead of the fragment ID field. A Netproowler agent was used at the victim to identify the attack at the victim and update the controller with attack signatures. Both the NetProwler Agent and Netproowler Manager were installed on a single machine. The Netproowler machine is shown in dotted line in Figure 2 since it operates in a promiscuous mode and is not visible to the outside world. We have also installed kiwi sys log demon [19] on the Net Prowler machine to receive feedback from the ingress agents.

The Net Prowler agent maintains a database of attack signatures and monitors all the traffic passing through the network to which it is connected. If any traffic is found to match with the attack signature, it sends an alert to its Manager and performs the configured action. In the case of an attack on monitored machines, the Net Prowler agent can be configured to automatically tighten a firewall and/or send an email 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 Net Prowler agent is configured to send an alert to the controller and log all the packets that are matching with the attack signature. There exists proper authentication between the Netprowler Agent and Netprowler Manager and all the communication between them is secured.

In this scenario, the Netprowler Agent is assumed to be the victim (security tool at the victim end to identify attack signatures), Netprowler Manager is the controller of ISP domain and routers act as the agents. At time T1, we generated a simple SYN flood using [14] on the victim Web Server. To make it more interesting we have generated the attack by spoofing the source address with that of good host. This is a very simple attack scenario and it should be noted that in this case we could perform Ingress filtering in order to eliminate all the spoofed traffic from SYN flood generator. In a complex network (Internet) the attacker can still forge his address with a valid IP address, which is within the domain. To demonstrate how attack is identified and prevented in our model we have not performed ingress filtering in this attack scenario. The Netprowler agent was configured to identify more than 20 half open connections as a high priority attack signatures. As soon as the attack traffic was generated, the Netprowler detected the attack signature and sent an alert to its Manager (controller) and logged all the traffic that is matching with the attack signature. Figure 3 shows the alerts sent by the victim (Net prowler Agent) to the ISP Controller (Net Prowler Manager).



Fig 3: Alerts at Net Prowler IDS

At this point, it should be noted that if we configure to block all the SYN packets at the victim web server, then all the good SYN packets from good host/network to the victim web server would also be dropped. Alternatively even if block only the SYN packets which have the source address as shown in the alerts this will also penalize the traffic from

good host since the attack was generated with the source address of good host. Now our idea is to differentiate the traffic originating from different ingress agents and identify attack signature only for the ingress agent through which the attack traffic is passing. At time T2 we have manually (as discussed this can be done automatically with NMS or IDS for example CiscoWorks with minor modifications) configured the Ingress routers/agents to mark all the web servers traffic in the ToS field of IP packet. The precedence field in the ToS field was used to mark the controller ID and the ToS field was used to mark the agent ID. The controller ID was set to 5 (represented in decimal format). 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. Listed below are the commands to filter and mark the victims traffic with controller ID and unique agent ID at the ingress agents.

For Ingress Router Agent 1.

Configure terminal ↵

Enter configuration commands, one per line. End with CNTL/Z.

```

IngAgent1(config)# interface S0
IngAgent1(config-if)# ip policy route-map X.X.X.X1
IngAgent1 (config-if)# route-map X.X.X.X
IngAgent1(config-if)# match length 0 - max
IngAgent1(config-if)# set ip precedence 5
IngAgent1(config-if)# set ip tos 1
  
```

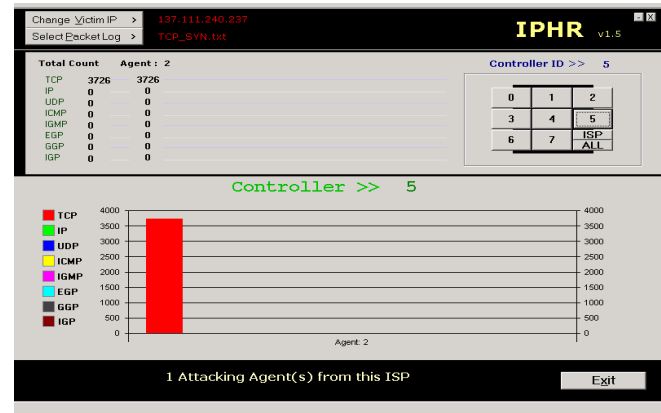


Fig 4: Identification of Attack Signatures at the Victim

Similarly we have configured Ingress agent 2 to mark the traffic with ip precedence =5 and ip tos=2. So all the traffic originating from agent 1 was marked with the controller ID = 5 and agent ID = 1. Similarly all the traffic originating from agent 2 was marked with controller ID =5 and agent ID =2. The NetProwler agent logged all the traffic. We have developed a GUI tool using Visual Basic and MS Access to analyze the logged data. The GUI was developed to identify

<sup>1</sup> X.X.X.X represents 32-bit IP address of the victim

attack signatures for multiple ISP domains but here we are only considering a single ISP model. The logged data was analyzed using our GUI tool to identify the attack signatures using the algorithm as discussed in section IV. The output of our analysis is shown in fig 4. From the fig 4 we can easily determine that the attack traffic is originating from Ingress agent 2.

At time T3, using access controls in router, we have configured only Ingress Agent 2 to deny access and log all the TCP SYN packets that are destined to the victim web server. Listed below are the commands for ingress agent2 to block the traffic that is matching with the attack signature.

```
IngAgent2#config t
```

Enter configuration commands, one per line. End with CNTL/Z.

```
IngAgent2(config)#logging Y.Y.Y.Y2
```

```
IngAgent2 (config)#ip access-list extended filter_out
```

```
IngAgent2(config-ext-nacl)#deny tcp any X.X.X.X mask syn log-input
```

```
IngAgent2(config-ext-nacl)#permit ip any any
```

```
IngAgent2(config-ext-nacl)#interface S0
```

```
IngAgent2(config-if)#ip access-group filter_out out
```

The ingress agent 2 was also configured to send the logged data as a feedback to the controller (Kiwi sys log daemon). The packets logged at the NetProwler agent decreased and the packets logged at Ingress agent 2 increased. Figure 5 shows the feedback messages from ingress agent2 to the controller.

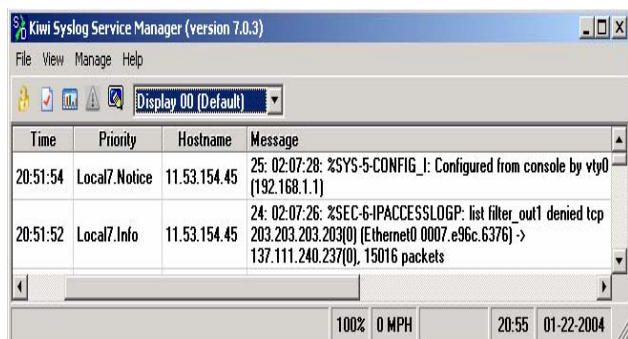


Fig 5: Feedback from Ingress Agent 2 to controller

## VI. CONCLUSION

In this paper, we have discussed how to counteract the TCP SYN flood attacks using our automated model against DDoS attacks. We have discussed the prototype implementation of our model. We have shown that our model

<sup>2</sup> Y.Y.Y.Y represents 32-bit IP address of the kiwi syslog server machine.

is very efficient in tracing the approximate source of attack and prevents the attack traffic at the nearest router to the attacking system.

## REFERENCES

- [1] Computer emergency response team. CERT advisory CA-2000-01 denial-of-service developments. <http://www.cert.org/advisories/CA-2000-01.html>, Jan.2000.
- [2] Computer emergency response team. CERT advisory CA-1999-17 denial-of-service tools. <http://www.cert.org/advisories/CA-1999-17.html>.
- [3] Robert Stone: "CenterTrack: an IP overlay network for tracking DoS floods," In proceedings of 9<sup>th</sup> Usenix Security Symposium, August 2000.
- [4] Ratul Mahajan, Steven M.Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker, "Controlling high bandwidth aggregates in the network," Draft, February 2001.
- [5] Udaya Kiran Tupakula, Vijay Varadharajan, "A practical method to counteract denial of service attacks," In proceedings of the twenty -fifth Australasian computer science conference ACSC2003, Australia. Feb 2003.
- [6] U.K.Tupakula, V. Varadharajan, " Counteracting DDoS Attacks in Multiple ISP Domains Using Routing Arbiter Architecture", Proceedings of 11<sup>th</sup> IEEE International Conference on Networks ICON2003, Australia. Sept 2003.
- [7] J.Postel : Internet protocol. RFC 791,Sept.1981.\
- [8] Computer emergency response team. CERT advisory CA-1996-21 TCP SYN flooding and IP Spoofing Attacks. <http://www.cert.org/advisories/CA-1996-21.html>.
- [9] D.J.Bernstein and Eric Schenk, " Linux Kernel SYN Cookies Firewall Project", <http://www.bronzesoft.org/projects/scfw>.
- [10] Check Point Software Technologies Ltd. SynDefender: <http://www.checkpoint.com/products/firewall-1>.
- [11] J.Lemon, "Resisting SYN Flooding DoS Attacks with a SYN Cache", Proceedings of USENIX BSDCon'2002, February 2002.
- [12] Netscreen 100 Firewall Appliance, <http://www.netscreen.com/>.
- [13] C.L.Schuba, I.V.Krsul, M.G.Kuhn, E.H.Spafford, A.Sundaram and D.Zamboni, " Analysis of a Denial of Service Attack on TCP", Proceedings of IEEE Symposium on Security and Privacy, May 1997.
- [14] Phrack magazine, issue 48-File 13of 18, "Project Neptune", Author: daemon9/route/infinity for Phrack Magazine. Date: July 1996 Guild Productions, Kid. <http://www.fc.net/phrack/files/p48/p48-13.html>.
- [15] T.Darmohray and R.Oliver, " Hot Spares for DoS attacks", <http://www.usenix.org/publications/login/2000-7/apropos.html>.
- [16] [www.cisco.com](http://www.cisco.com)
- [17] [www.hp.com](http://www.hp.com)
- [18] [www.symantec.com](http://www.symantec.com)
- [19] [www.kiwisyslog.com/software\\_downloads.htm](http://www.kiwisyslog.com/software_downloads.htm)