Denial of Service Protection
The Nozzle

Elizabeth Strother
North Carolina State University
epriggin@eos.ncsu.edu

August 29, 2000

Abstract

A denial of service attack is a dominating conversation with a network resource designed to preclude other conversations with that resource. This type of attack can cost millions of dollars when the target is a critical resource such as a web server or domain name server. Traditional methods, such as firewalls and intrusion detection systems have failed to provide adequate protection from this type of attack. This paper presents a new protection method called a nozzle. The nozzle is based upon favorable aspects of firewalls and network pumps. It is deployed similar to a firewall such that all conversations from an untrusted user to a critical resource are monitored. The main advantage of the nozzle is the ability to provide a threshold for trusted traffic thus precluding new attacks. A nozzle consists of a series of rings. Each of which has a trusted and untrusted buffer, rules for packet placement, and rules for communication with the next level. Rings are placed in the protocol stack so they can protect particular protocols.

1 Introduction

The process of communicating with a computer systems with the intent of degrading other communication or obtaining unauthorized information is called an attack on the system. An attack aimed directly at degrading or obstructing communication to a computer is called a Denial of Service (DoS) attack. When the target of such an attack is a critical resource such as a company's web server, the attack can cost the company millions of dollars in potential revenue. A report released by the Yankee Group (www.pcworld.com) estimated that during the week of Feb. 6, 2000, victims costs exceeded 1.2 billion dollars due to this type of attack.

What can be done to protect ourselves? The purpose of this paper is to suggest a scheme for providing protection from DoS attacks. Section 2 describes the DoS attacks in more detail, allowing the reader to become familiar with the terminology and definitions. Section 2 also provides some concrete examples of attacks so the user can realize how an average user could launch a DoS attack. Section 3 describes some conventional methods for dealing with DoS attacks. This section points out each methods short comings as well as its benefits. Section 4 introduces our method of preventing DoS attacks and discuss the design, benefits and drawbacks of the method. Section 5 concludes this paper with a discussion of future research.

2 The Attack

DoS attacks can be geared towards exploiting a product defect. Examples of this type of attack are the "bonk" and the "Ping of Death" attacks. The bonk attack exploits a defect in certain IP (Internet Protocol) implementations. The defect is a lack of bounds checking when reassembling an IP packet. IP packets

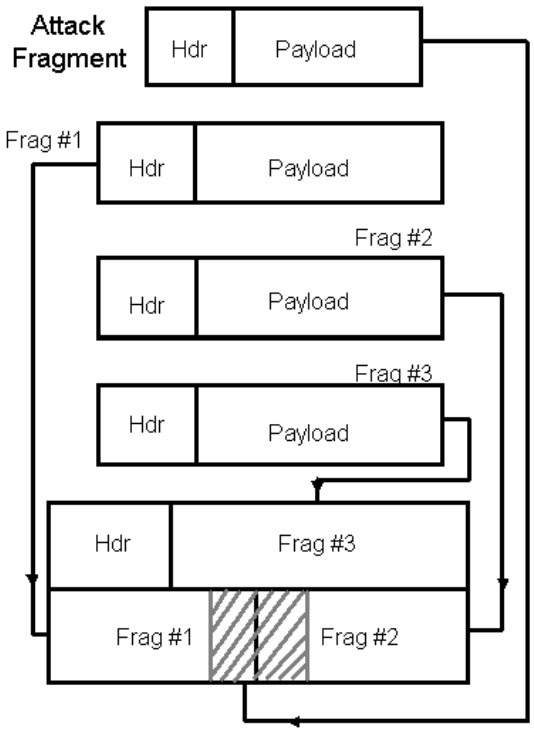

Figure 1: Bonk attack

can be a very large size. When the packet is transmitted, it is encapsulated in the framing structure of its current media. For example, if the workstation is on an Ethernet segment, the IP packet is encapsulated in an Ethernet frame. Each media has a maximum frame size that it can transmit. Packets which would produce a frame larger than the maximum frame size must be divided into 2 or more packets such that the obey the maximum transmission size. This division of the packet is referred to as packet fragmentation.

Each fragment contains information necessary to reassemble the fragment into the original data. This information includes its offset the original packet. For example, figure (1) shows four packet fragments. The first three fragments are 600 bytes each and the last fragment is 200 bytes. The first fragment's payload should be placed in bytes 0-599; the second in bytes 600-1199; the third in bytes 1200-1799; and the last in bytes 1800-1999. The "bonk" attack exploits the lack of bounds checking by sending fragments with offsets that do not align. In figure 1, an attack packet is shown in the reassembled packet. The attack packet would solicit a starting position of byte 1549 in the original packet. This would make reassembly of all the packets impossible since the positions overlap. Certain operating systems will not handle this properly and will stop further communication until the system is restarted.

The "Ping of Death" attack exploits another software defect. This attack is aimed at the ICMP echo request (ping) software. Echo is a function designed to send a packet for the purpose of determining connectivity. If data is sent in the payload of the ICMP request, the response echos the data back to the requester. This protocol is often used with a small payload to provide a fast, low bandwidth test of connectivity. Because of this typical usage, some software does not handle large payloads. If it receives an ICMP request packet with a payload greater than 4000 bytes the software generates an exception and halts communication on the network. These attacks will not work on all applications and are considered defects in particular software implementation. There has been much research geared towards designing defect free software [6][7][8] and this paper is not designed to cover this area of research. Therefore, the class of DoS attacks which exploit software defects will not be examined in this paper but the nozzle does allow for configuration to block such attacks.

A more serious class of attacks is one which exploits vagueness or omissions in a protocol specification because it affects all implementations. A well-used example of this class of attack is the "smurf" attack. IP has several required fields which are specified in the packet header. The designers specified that one of the required fields specify the senders IP address so that replies or acknowledgments could be returned to the sender. The senders address does not have to be accurate though. The routing used with IP does not guarantee that the senders address is correct. The Internet Protocol is designed to route on a hop by hop basis to provide a manageable configuration and also allow for fluctuations in the routing structure. To accomplish this diversity, only the next

hop routing information is stored. That is, a router only knows the next machine to pass the packet to and has no knowledge of where the packet has come from. To protect against false address, egress routing has been suggested [3]. In egress routing, the router keeps track of what source addresses it expects to leave on its outbound network interface. If it sees an address other than what it expects, it discards it. This solution, however, contradicts the flexible dynamic routing designed in IP networks. The range of addresses that might traverse the router are subject to change and the routers are designed to provide this flexible configuration. Therefore, with all software written error free and in accordance with the specification, a user can claim a bogus address and the true identity will be untraceable. A DoS attack called "Land" can be launched where the sending address is equal to the destination address and thereby establish communication in a loop on the victim machine. If this communication is designed to pass a lot of traffic the victim's resources can be consumed talking to itself and thereby block outside conversations. This type of attack is possible because the specification did not design any authentication methods to verify the senders address.

The most common type of DoS attack is designed to attack a well designed protocol and a defect-free application by generating a workload which can not be sustained by the system. A common example of this type of attack is the "Syn flood" attack. The TCP protocol is designed to provide reliable communication for an application. This means that if a packet gets dropped on the network, TCP will detect this lose and request retransmission. TCP sends the sender an acknowledgment of receipt when it receives the data. If the sender does not receive this acknowledgment within a specific time period, the packet is retransmitted. To establish the connection, the sender sends a packet called a SYN packet (this is a request for synchronization). Upon receipt, the destination or receiver sends an acknowledgment that it received the SYN request and also sends its own request for synchronization (ACK-SYN). The original sender then acknowledges the receipt of the receivers SYN. This is called a three way handshake. Once this is complete, reliable communication can be conducted. However, before the handshake is complete, the receiver has no idea how long the propagation delay is between itself and the other machine. Therefore, once sending the ACK-SYN packet, it will wait longer than usual (often up to 2 minutes) to receive a response. An attacker can take advantage of this wait period by sending SYN request packets and not responding to the ACK-SYN packets. See figure (2). The receiver will have to store information necessary to correlate the expected ACK response with the packet sent. There is a limited amount of space available for this storage so if enough packets are received this space will be exhausted and the machine will cease to receive any additional SYN request. Therefore, no valid connections can be made during this period. This type of attack basically exhausts a limited resource by producing more requests than the machines expects to receive.

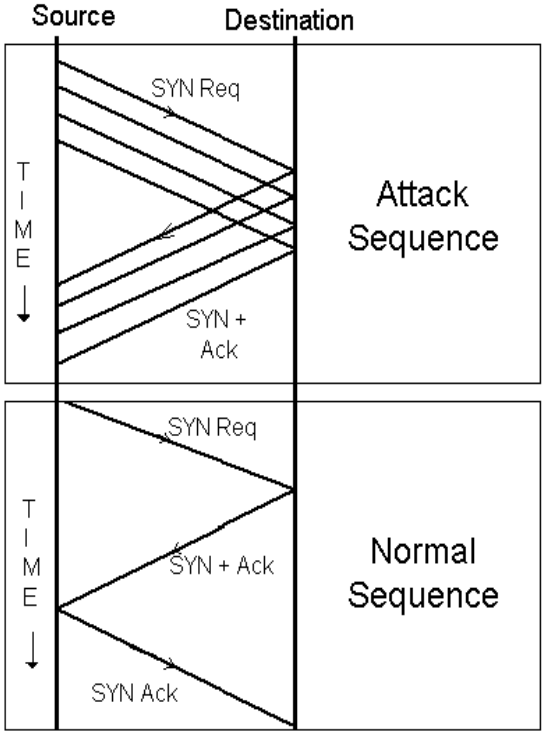

Figure 2: Syn Flood attack

We now have a good understanding of the types of attacks that we are attempting to guard against. We are concerned with attacks which exploit vagueness and omissions in communication specifications and attacks which exhausts resources. Software defects need to be guarded if they involve defects in the communications system. Before presenting our solution, we will examine other solutions and investigate their strengths and weaknesses.

3 Conventional Protection

In today's networks, there are several methods or products which are designed to preclude DoS attacks. These methods include deploying firewalls to restrict incoming traffic, Intrusion detection systems to detect known attacks, and network pumps to minimize DoS attacks in multi-level secure systems. We will examine the benefits and drawbacks of each of these solutions.

A popular yet often ineffective means of providing protection is a firewall. A firewall is a workstation deployed between the company's network and the Internet. A firewall can examine any packet moving between the internal network and the Internet or a firewall can be configured to serve as a proxy for communications. A firewall is designed to limit the incoming traffic thereby reducing the risk of an attack. However, as mentioned earlier in this paper, there is no guarantee that the source address in a packet is valid so any filtering based on source address is subject to vulnerabilities. A firewall also provides no method of limiting valid traffic such that a machine is not simply overwhelmed with requests.

Firewalls, however, are based upon a valid premise. If there is no need for traffic to access the network, there is no reason to allow the traffic on the network. If only HTTP (HyperText Transfer Protocol) is allowed from the Internet, there is no need to pass ICMP or UDP traffic. This reduces the type of attacks which can be targeted to the network. This in no way provides a complete solution. Configuring a firewall to block traffic based upon source address precludes only the most novice attacks while configuring the firewall to limit the type of traffic allowed on the network reduces the attack set. Therefore, limiting the type of traffic is a beneficial solution.

A newer method for detecting network attacks is to monitor the network for known attack signatures using an Intrusion Detection System (IDS). An attack signature is a sequence of events which is known to occur prior to an attack. For example, the "Ping-O-Death" attack mentioned earlier in this paper has a signature of an ICMP echo request with a payload exceeding 4000 bytes. An ID system could be configured to monitor the network for such a packet. If one is detected, the IDS could remove the packet from the network and generate an alarm indicating that an attack has been detected. An ID system which drops packets matching known attacks reduces the set of attacks but the attack signature must be known in order the for the IDS to be effective. Attackers are devising new attacks everyday and there is no way of knowing the attack signature until the attack has attacked at least one site.

ID systems which police traffic for attacks signatures are known as knowledge based systems. If the system has the ability to learn attack signatures, it is called a behavior based ID system. The concept of learning the attack signatures such that the system remains up to date with little human interaction is a wise idea. The problem is that this learning must take place in real time as the packets are traversing the network. Any delay in processing can greatly impact the quality of service or allow for security exposures. Another problem with behavior based ID systems is false positives reports. Whenever a system attempts to learn a behavior at the same time as guarding against the behavior it runs the risk of falsely identifying an occurrence. In most networks, the number of false positives must be sufficiently low so as to not disrupt normal communications.

A Network Pump has been used to protect MLS systems from DoS attacks and provide some level quality of service for its users [5]. A multi-level secure system (MLS) is a system in which devices are categorized as high or low. High devices are regarded as the sensitive machines which should not expose information to machines of a lessor sensitivity (low machines). High devices are only allowed to send acknowledgment information to low devices but they

can communicate sensitive data with other high machines. Low machines can send any level of data to any machine. The network must ensure that high machines do not send sensitive data to low machines using a covert channel. A covert channel is a connection which hides data within valid messages. For example, a confidential file could be sent byte by byte in the sequence number field of each acknowledgment message until the entire file is transferred. The network pump is designed to protect against covert channels and DoS attacks. Covert channels are eliminated restricting high machines from communicating directly with low machines. Acknowledgments are sent to the low machines directly from the pump which eliminates the possibilities of a covert channel from the high machine. Use of the pump also enables performance metrics such as throughput, fairness and reliability. These measures are needed in order to guard against denial of service attacks. If a system can not guarantee a user a certain portion of resources, the user always runs the risk of being starved from the system.

The network pump provides connectivity between low machines and high machines. See figure 3. Each low machine connects to the network pump. The pump has a buffer for each link connection. These buffers can differ in size from one link to another. Each slot in the buffer is capable of storing one message. A low machine transmits a message directed to a high machine. The network pump receives the message in the corresponding input buffer. Messages are removed from the buffer according to a max-min scheduling algorithm. The max-min scheduling algorithm divides the rates equally if the requests are greater than a proportional division of the output. If a queues rate is less than an equal division, it will be fully satisfied and any remaining bandwidth will be divided equally amongst the remaining queues. Messages are removed from the buffer by a trusted low process (TLP). This process directs messages to the corresponding output queue based upon the destination of the message. When a message is successfully placed on the correct output queue, the TLP sends an acknowledgment message back to the low machine. If the input buffer is full, a low machine can not send another message until a message is removed from the

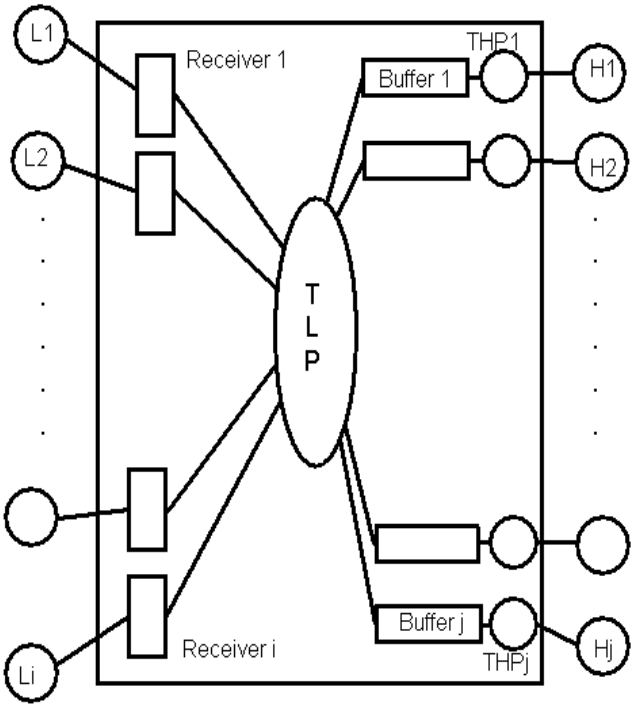

Figure 3: Network Pump

buffer. This allows the pump to act as a throttle for traffic. If a high device is in heavy use, the TLP will not be able to place the message on the output buffer until the high device has worked thru its queue, thus the machine can not be overwhelmed by excessive use.

Since a max-min queuing algorithm is used to fairly distribute messages, a DoS attack can only be successful if all low sides cooperate to launch the attack. In this case, all the low side machines are attackers and thus no meaningful message is precluded from the high side so the attackers have not been successful in denying any service. There are however practical limitations to this solution which make it impractical outside of this specific application. First, there must be a dedicated link and buffer combination for each low side machine and each high side machine. This is required to ensure fairness amongst

the machines. If we tried to apply this solution to an Internet connected device, the possible number of connections could exceed a million. Clearly, the number of connections is a limiting factor of the system. In addition to this limitation, the network pump does not analysis any of the packets for possible attacks. It assumes that a DoS attack will only occur from overwhelming the machine and does not consider attacks based upon faulty software or protocol omissions.

4 The Nozzle Solution

The solution we present for protection from denial of service attacks incorporates many of the positive features of the earlier mentioned systems without the drawbacks. It is primarily a combination of a firewall and a network pump. Like a network pump, our device will send acknowledgments and throttle the data to the servers. However, instead of a dedicated link for each incoming device, our solution, the Nozzle, has only one incoming link for all messages and can drop packets if they do not meet specified criteria, similar to a firewall.

All traffic from the untrusted network must pass through the nozzle to reach the protected resources. A nozzle is comprised of a series of "rings" applied in a layering fashion. Any ring in the nozzle can be configured to block traffic based upon certain predefined criteria such as source address or traffic type just like a firewall. It is this layering concept which is a unique design point which differs from existing firewalls. Layering is a familiar concept among computer specialists. This allows them to build upon previous work. Administrators develop security policy for different layers of the protocol stack and then configure the rings accordingly. This layering will ease the management of the nozzle and allow different permissions to be associated with different rings. For example, rings which are at the application layer may be configurable by a number of administrators but lower level rings may require extra security. An administrator could configure the lower level rings with special permission which prohibits changes from other administrators. Current firewall solution only provide a single level of administrative security. An administrator either has access to the firewall or not.

Each ring has a trusted buffer and an untrusted buffer. Data can only be presented to a ring by placing the message on the untrusted buffer. Each ring has a configurable policy to move data from the untrusted buffer to the trusted buffer. A sweep of the buffer will occur periodically to move items to the trusted buffer or perform whatever operation is necessary to determine if the packet can be moved to the trusted buffer. A ring can be configured to request additional information from the source thereby gathering additional information about the source to determine trustworthiness. Figure 4 depicts a nozzle.

Communication with the final destination can only occur when a packet is on the trusted buffer. In this case, the ring must be configured to communicate with the final destination rather than passing the packet to a higher level ring. If the ring policy does not specify to communicate with the final destination, the packet is passed to the untrusted buffer of the next higher ring. Multiple rings can be configured to communicate with the final destination. For example, if a ring has a policy to pass ICMP traffic the final destination, it can also have policies to pass UDP, FTP, or any other criteria. An administrator should avoid ambiguous policies though. A rings policy to pass to the untrusted buffer of a higher ring is evaluated before passing a packet to the destination so if an ambiguous policy is created, the path requiring a higher level of security will be taken. For example, if a ring at the TCP level specifies to pass TCP sessions to the destination and pass TCP traffic to the next higher ring, all TCP traffic will be passed to the higher level rather than to the final destination.

The main advantage to the nozzle is its ability to limit incoming traffic. Each ring has a configurable threshold and maintains the average response time for items held on the untrusted buffer. When the filled portion of the trusted buffer reaches the threshold, the timeout value for items on the untrusted buffer will be decreased until either the filled portion drops below the threshold or the timeout values reaches the average response time for the buffer. If the timeout value reaches the average response time for the buffer, a cleaning sweep of the buffer is performed. When a cleaning sweep is performed, the

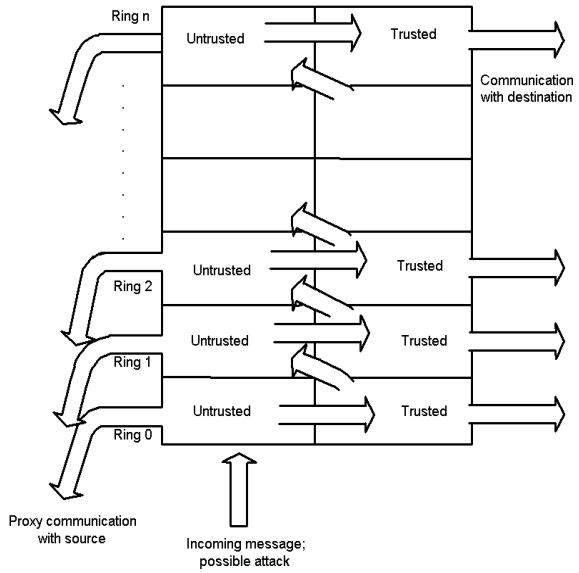

Figure 4: Nozzle Diagram

nozzle can be configured to send a management agent an indication that a sweep is occurring. This can alert an administrator of an attack or a performance problem. In the cleaning sweep, each item is either be moved to the trusted buffer or dropped according to the rings trust policy. If an item is waiting for some additional information to determine its trustworthiness, it will be deemed as untrusted and dropped. Only trusted traffic will remain in any buffer on the ring after a cleaning sweep. If there is more trusted traffic than is allowed on the trusted buffer, excess traffic will be dropped. This allows trusted traffic to continue when the system is under attack. Also, when under attack, packets similar to the attack signature are not automatically precluded from communicating with the final destination. If the packet is received anytime other than when a cleaning sweep is performed, the packet will have an opportunity to prove it is trustworthy. Since communication with closer resources will have a lower response time, these communications are more likely to be accepted as trusted than devices with higher propagation delays.

A standard input queue consists of a first-in, first-out design and has a get and put function. Get removes the oldest item from the queue and put pushes a new item into the queue. Our queues still use the get and put functions but they act differently. Our data structure is actually a queue and a buffer. The queue consists of the trusted traffic, ordered by a timestamp of when it was placed on the trusted queue. The buffer contains currently untrusted traffic. It also has a timestamp as to when it was placed in the buffer. There is a policy that will specify what criteria must exist of the traffic to be considered trusted. A get call will remove the oldest item from the trusted queue. After the item is removed, it will create a thread that will move as much data as possible from the untrusted buffer to the trusted queue. The move will occur if the data has met the configured trust policy. If the trusted buffer exceeds its configured threshold, a cleaning sweep occurs. If any untrusted data has been on the buffer for more than the timeout period, it is removed. During a put function, an item is inserted into the untrusted buffer. After the item is inserted, if the item was untrusted traffic (according to the policy) a thread is created and the buffer is searched for other traffic which might belong to this communication. If additional traffic is found, all the traffic in this communication is evaluated to see if it meets the trust criteria. During the put function, if the item can not be inserted onto the untrusted buffer because the buffer is full, a cleaning sweep is performed.

To demonstrate this solution, we present an example of an IP nozzle which is designed to prevent all the attacks presented in this paper and control the amount of traffic presented to a machine to avoid DoS based upon overuse. The first ring in the nozzle is placed between the data link layer and network layer. All incoming IP traffic will be placed on the untrusted buffer of this ring. Since this example is an IP nozzle, all non-IP traffic will either automatically be blocked or passed to the final destination based on the user's configuration. For this example, we will assume non-IP traffic is blocked. Our policy for the trusted buffer at ring 0 will be that all packet fragments are present and only complete reassembled messages are passed to the next level. Therefore, if a packet fragment is received, it will be held in the untrusted buffer until all packets are received and re-

assembled or until a timeout period has expired.

The next ring level will have 2 rings. One will be between the IP level (network layer) and the TCP level (Transport layer). The other ring at this level is between the IP level and the ICMP level. If a message is not a TCP message or an ICMP message, we will configure our nozzle to drop the message. You could place another ring in between the UDP and IP levels if desired. In our example, messages from the trusted IP level (ring 0) can only be passed to the untrusted buffer leading the TCP stack of ring 1 or the untrusted buffer leading to the ICMP stack of ring 1.

On the TCP ring 1, SYN requests will be kept on the untrusted buffer. A SYN ACK will be sent back to the source. If the message is not a SYN ACK, the message will be passed to the trusted buffer of ring 1. If the nozzle receives an ACK for the SYN ACK, a connection is established with the final destination using the true source address in the connection establishment. The nozzle will act as a proxy, masquerading as the source address to the destination and as the destination to the source, until the connection is established. This allows the nozzle to determine trustworthiness without exposing the source to potential attacks. Once the connection is established, the TCP traffic is considered trusted and the nozzle no longer acts as a proxy. If the source address fails to fully establish a connection without a specified time period, the nozzle will reset the connection is has established to the source.

An initial timeout period of 2 minutes will exist for the SYN-ACK packets. If the occupied portion of the untrusted buffer exceeds the threshold value, the timeout value will be decreased until either the occupied space of buffer is lower than the threshold or the minimum timeout value equals the average propagation delay for this ring. If the timeout reaches the average propagation, a cleaning sweep will be performed on the untrusted buffer. If a message is not a SYN request, it will be moved to the trusted buffer else it will be discarded.

If the message is an ICMP request, it will be placed on the untrusted buffer of the ring leading to the ICMP layer. This layer is defined to only trust ICMP echo messages which conform to the average packet

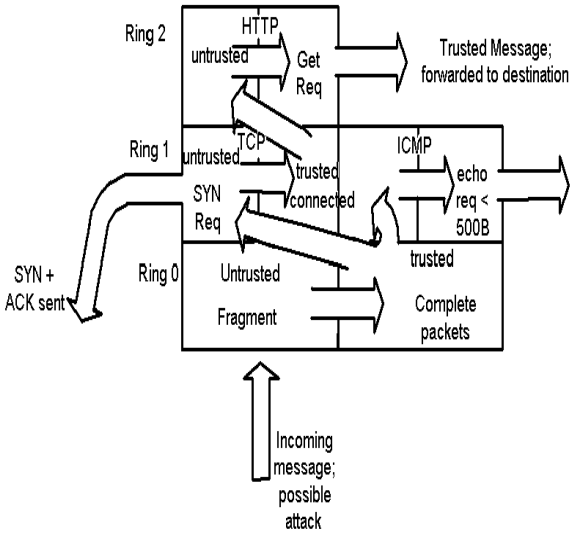

Figure 5: Nozzle Example

length. Therefore, the trusted buffer will maintain an average length field. If the request is an ICMP echo (request or reply), it will be moved to the trusted buffer if it does not exceed the average length plus some defined variance, else it will be dropped.

Ring 3 of our example will be placed between the TCP level and the HTTP level. In our example, we will only pass traffic to our interior web site if it is a HTTP get request or an ICMP echo request. Therefore, if it is a HTTP get request, the message is passed to the source, otherwise, the message is discarded.

From our example, it is easy to see that packet fragmentation attacks can not be launched against the protected machines, because all packets will be reassembled and verified before they are passed to their final destination. Also, SYN flood attacks can not be successfully launched since connections are established from the nozzle before they are established with the final destination. The nozzle, itself, is not susceptible to a syn flood attack since it has a limited number of buffers for incoming packets. When the buffer crossed a predefined threshold, the timeout value will be decreased so that connection request packets are given a lower service rate than es-

tablished connections. This will allow trusted traffic to continue to be passed when under attack and new requests can be established if the requesting machine has a good response time. The timeout value will have a minimum value of the average propagation delay. If the timeout is reduced down to this minimum value and the buffer still continues to fill, a cleaning sweep with move all connected traffic to the trusted buffer and discard all connection requests. A "Ping-of-Death" attack is not possible since the ICMP ring will maintain an average packet size and discard exceptionally large packets.

5 Future work

Configuring a set of rings to act as a secure mechanism could be a difficult and tedious task for a network administrator. To reduce the complexity and risk of misconfiguration, applications should be allowed to configure application level rings. A standard set of parameters such as average delay and buffer sizes will be configured by the administrator. Each ring will be provide a method for communicate with source devices from the untrusted buffer and to destination devices from the trusted buffer. The application will only need to set the policies for the ring. A secure protocol must be established to authenticate applications and grant modification rights to particular ring levels. This would allow dynamic configuration of the nozzle to allow for transient trusted communications. Since permissions are associated with different rings, dynamic configuration can be achieved without exposing the network to security vulnerabilities.

A ring must be able to perform a cleaning sweep faster than the time necessary to fill 10% of the buffer. Ring buffer sizes are configurable and cleaning sweep times will depend on the policy in place to move data from the untrusted to the trusted buffer. Due to these dependencies, much work needs to be performed to provide configuration restriction that keep timing problems from occurring.

The nozzle itself must be able to pass traffic from the incoming interface to the outgoing interface fast enough that the buffers do not fill up due to its own performance bottlenecks. Because of this potential problem, the number of rings should be kept to a minimal amount. There must be a way of communicating the available throughput to the administrator. Companies will be reluctant to deploy a device which inhibits communication without knowing of its performance. Since the nozzle's performance is dependent on how it is configured, the maximum and minimum throughput must be given to the administrator. When the nozzle is dropping traffic due to a cleaning sweep, this means that either the network is under attack or the nozzle is unable to keep up with the incoming traffic. The nozzle will generate an SNMP trap to be sent to a management device so that the administrator is made aware of the possible attack or performance problem.

The last problem which needs to be addressed it the order in which rings are evaluated when thresholds are exceeded. If lower rings are evaluated first, an attack could be successful. Rings must be evaluated from the higher levels down. This will allow the true attack to be eliminated without dropping unnecessary traffic. When a new ring is added into the system, a check needs to be performed to ensure consistency with the existing policies.

References

[1] Jonathan K. Millen. A Resource Allocation Model for Denial of Service. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 137-147. Proceedings., 1992.

[2] Durst, Champion, Witten, Miller, Spagnuolo. Testing and Evaluating Computer Intrusion Detection Systems. Communication of the ACM. July, 1999. Vol. 42 No 7. pp 53-61.

[3] Schuba, Krsul, Kuhn, Spafford, et. al.. Analysis of a Denial of Service Attack on TCP. In *IEEE Symposium on Security and Privacy*, pages 208-223. Proceedings., 1997.

[4] Che-Fn Yu, Virgil Gligor. A Formal Specification and Verification Method for the Prevention of Denial of Service. In *1988 IEEE Symposium on Security and Privacy*, pages 187-202. Proceedings., 1988.

[5] Kang, Moskowitz, and Lee. A Network Pump. IEEE Transactions on Software Engineering, Vol. 22, No. 5, May 1996.

[6] Kolkhorst, B.G., Macina, A.J.. Developing error-free software. IEEE Aerospace and Electronics Systems Magazine, Vol.3, Issue 11, Nov. 1988, pp 25-31.

[7] Carnot, M., DaSilva, C., Dehbonei, B., Mejia, F.. Error-free software development for critical systems using the B-Methodology. Third International Symposium on Software Reliability Engineering, Proceedings., 7-10 Oct. 1992, pp. 274 - 281.

[8] Chmura, L.J., Norcio, A.F., Wicinski, T.J.. Evaluating software design processes by analyzing change data over time. IEEE Transactions on Software Engineering, Vol. 16, Issue 7, July 1990 pp. 729-740.