

Optimizing NIDS Performance

Neil Desai 2002-06-06

Optimizing NIDS Performance

by Neil Desai

last updated June 6, 2002

Introduction

Network intrusion detection systems (NIDSs) face some of the most gruelling challenges of any security product. Not only is the bandwidth these devices monitor increasing, so are the amount of attacks they must guard against. The combination of these two factors could overwhelm a NIDS, causing it to drop packets. To help the NIDS keep up with the demands of today's networks, and the wide variety of threats that besiege them, there are a number of things that the NIDS administrator can do to improve the performance of their NIDS. This article will examine some of those options.

Review The NIDS Deployment Policy

Before doing anything, the NIDS administrator needs to review the current NIDS deployment policy. This policy should be part of a high level overview of how the organization wants to approach intrusion detection in general. Some people may choose to monitor for all attacks, regardless of what systems are prevalent in their organization; for example, looking for RPC exploits in a Microsoft environment. Others may watch only for attacks that would be relevant to their network environment, such as configuring the NIDS to detect all Microsoft exploits in an all Microsoft environment. Some choose the middle road and look at all vulnerabilities for a particular service regardless environment, such as detecting all HTTP exploits in an IIS-only environment. The NIDS policy that the organization lays out will determine which strategy they choose to employ, and in so doing will determine what the NIDS engineers can do to help their NIDS performance.

Filtering Signatures

Once the policy has outlined the general strategy that the security staff will implement, the first thing that the NIDS engineer should do is trim the amount of attacks that the NIDS will look for. The NIDS default configuration may monitor for potential attacks that are not relevant to the environment that. This can lead to wasted resources and, thus, to inefficiencies. For example, if the company is a pure Windows environment there is no real reason to look for RPC exploits.

Signature trimming can remove many unnecessary signatures at a time. For example, if the system is running [Snort](#), the admin can simply edit the snort.conf file and remove the entire rules file (i.e. rpc.rules, x11.rules). To get more granular, the administrator should look at the more common services (i.

e. HTTP, FTP, SMTP) and see if the attack signatures they are looking for match the services that the company runs. In this context, it makes sense to look for signatures that match software vendors that are on the network. For example, if the company uses FTP servers, but none of them are running wu-ftp, the NIDS does not need to be configured to look for wu-ftp exploits.

Most major NIDSs can be classified as either a pattern-matching NIDS (such as [Snort](#), [ISS](#), [Dragon](#)) or protocol-analysis NIDS (such as [BlackICE](#), [SecureNet Pro](#)). Regardless of which type of NIDS is being implemented, it will have to do some form of pattern-matching, or searching in the payload for a particular string, at some point. This is one of the most CPU intensive processes that a NIDS will perform. Different vendors take different approaches to this issue. Snort utilizes special pattern-matching algorithms to gain speed and performance, while BlackICE decodes each packet according to the specific protocol, breaks each packet into the appropriate fields according to the protocol and does an exact match in each particular field. Either way, by limiting the amount of content-based checks that are performed, the NIDS performance will be increased.

To determine if a signature is going to perform some pattern-matching, the NIDS engineer should look at the vendor documentation regarding the signature. (If this does not give any information about the attack being looked for, the information can likely be found on any number of Web sites dedicated to security.) In Snort this information would be found in the reference and content rule option (Figure 1). If the NIDS is looking for any data that resides in the packet data, then it will be using some form of pattern matching to search for the data.

Figure 1. (web-iis.rules)

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS
webdav file lock attempt"; flags:A+; content:"LOCK "; offset:0; depth:5;
reference:bugtraq,2736; classtype:web-application-activity; sid:969; rev:1;)
```

Filtering Unwanted Traffic (Monitoring NIC)

Most NIDSs have some sort of filtering function that allows certain types of traffic to be disregarded. There are a couple instances when this type of filtering may be of value. Firstly, if there is a server or subnet that generates a lot of traffic that does not need to be monitored. One form of this type of traffic would be multicast traffic, which is usually some type of streaming media. Some switch vendors may be able to filter the traffic before it gets to the NIDS. This would be the preferred approach, as the NIDS will not have to unnecessarily process data that has already been determined to be harmless. Secondly, this type of filtering may be useful if there are servers or subnets that generate traffic that is encrypted or that the NIDS otherwise can't decode. Since this type of traffic would need to be thrown away, there is no reason to search the packets in the first place.

Filtering Unwanted Traffic (Management NIC)

Most NIDS are installed with two NICs (Network Interface Cards), one for monitoring a particular network segment and one for management. Depending on how the network is designed there could be a lot of erroneous packets hitting the NIDS management NIC, even in a purely switched environment.

By design, a switch will send all broadcast and multicast packets to all users on the local broadcast domain. Because the NIC will have to at least look at the packet's destination MAC address to determine what to do with the packet, this type of traffic could affect all nodes on the network that receive this type of traffic. Multicast packets are usually larger than broadcast packets because of the content being delivered. There are many applications that take advantage of multicast's delivery that may not be known to the NIDS engineer (i.e Ghost, call tracking systems). If the network design is poor or if the switches are overloaded the switches may be forwarding unicast packets out the wrong ports. If there is enough of this type of traffic it could also affect the network.

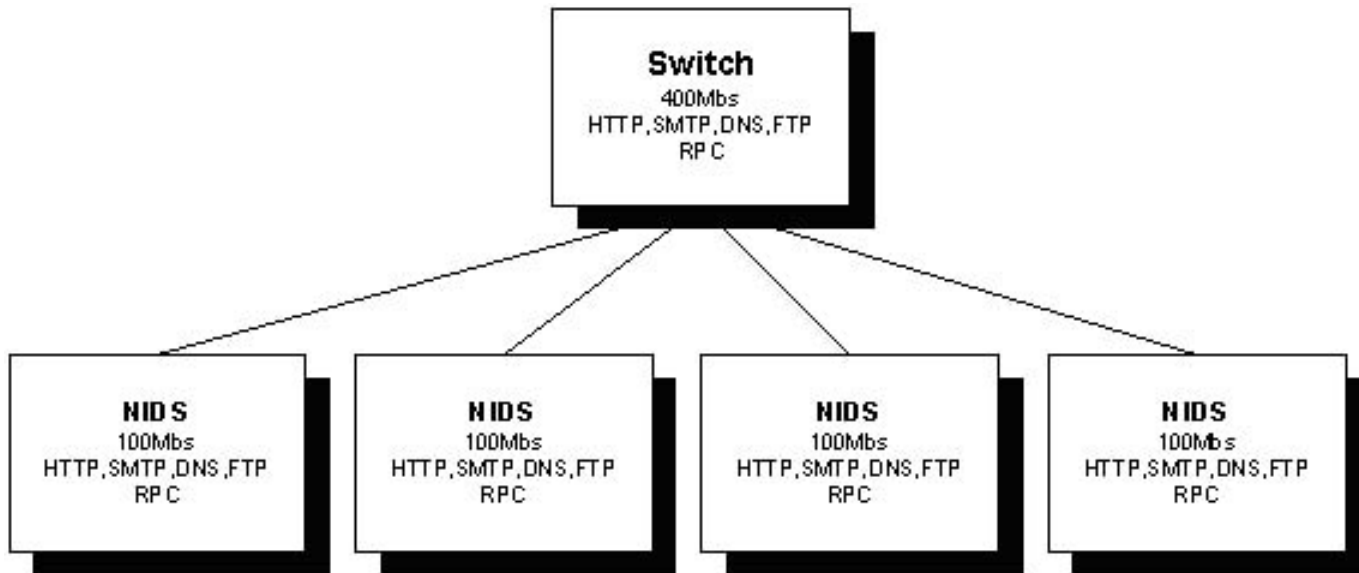
The worst type of traffic (per broadcast domain) would be an excessive amount of ARP request packets. ARP request packets are smaller than most multicast packets, but the problem arises with how the packets are treated. When a multicast packet is received, the node that receives it can look at the destination MAC address and determine what it should do with the packet, either drop it or pass the packet up the TCP/IP stack. All ARP request packets will be accepted by each node that receives it. Each node will decode the ARP packet see if the ARP request is for the IP address of that node. If the ARP request is not for the IP address of that node then that node will discard the packet. If the ARP request is for the IP address of that node then it will respond with an ARP reply. While statically putting ARP mappings in each host would do the job, it would most likely be infeasible. The only way to either stop this traffic altogether, or at least limit the amount of this type of traffic, would be to configure the switch port that the NIDS management NIC is plugged into. Depending on the switch vendor, there may be an option to block all multicast traffic on a per-port basis or limit the amount of traffic (multicast, broadcast and unicast) on a per-port basis.

NIDS Load Balancing

Another type of filtering could be done with some type of hardware device. This would allow traffic to be split up and sent to a NIDS farm (a logical grouping of multiple physical NIDS to handle high bandwidth networks). This is the same idea as DNS or Web server load balancing that is currently used on many networks. [TopLayer](#) makes a product called [IDS Balancer](#), which is aimed at NIDS load balancing. Other vendors, including TopLayer, make layer seven switches that can preform IDS load balancing and much more. With these types of devices, 100Mbps+ of bandwidth can be distributed over multiple sensors in the NIDS farm (Figure 2). In a basic distribution scenario, the traffic would be evenly distributed among each NIDS in the NIDS farm. This means that each NIDS should get an equal amount of traffic. Since

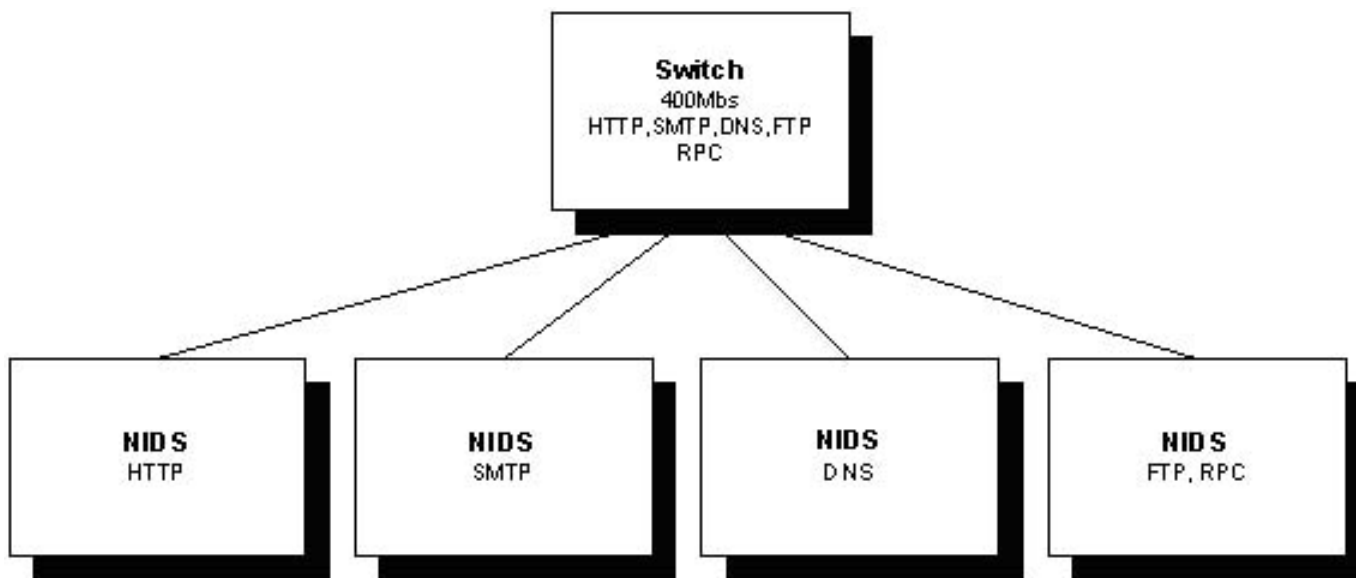
today's NIDSs are more stateful than previous generations, each NIDS would have to get traffic distributed to it based on sessions (conversations), not individual packets. This way the NIDS can watch the entire conversation and be aware of any attacks or anomalies. Each NIDS would have the same policies, so that they would catch the same exploits and anomalies. With devices such as NIDS load balancers and layer-7 switches, the traffic could be filtered before it is sent to the NIDS farm.

Figure 2.



A better alternative to doing a basic even distribution of connections to each NIDS would be to specify that certain NIDS get only certain types of traffic (Figure 3). For example, if the NIDS farm contained four physical NIDS, one of the NIDS would watch only HTTP traffic, one would only watch SMTP traffic, a third would only watch FTP traffic, and the fourth NIDS would only watch DNS and RPC traffic. This would allow each NIDS to be customized so that they only look for certain types of attack signatures and anomalies. Due to the more specific set of traffic being sent to each NIDS, and because each NIDS is customized to look for only certain types of attacks, the overall overhead on the NIDS would be decreased, thus improving the performance of the NIDS.

Figure 3.



The poor man's version of NIDS load balancing involves spreading the NIDS out physically across the corporate network and filter based on subnets. This would be similar to the basic distribution of traffic. The NIDS could be placed on the switch uplink ports, provided that there is no trunking protocol configured for those ports.

Changing Default Timeouts

Depending on the NIDS being deployed, there may be options to configure how many connections are tracked. Altering this configuration will affect the amount of memory used to store this information, as well as the amount of CPU cycles used to search this information. When these timeouts are defined, a certain amount of memory is set aside for these tables, regardless of whether or not they are filled. By doing some traffic analysis on the network, the settings could be changed to make the NIDS more efficient when watching a particular network segment.

There may be options available to define timeouts for particular protocols. These may be as generic as TCP and UDP timeouts, or they may be as granular as HTTP, DNS and SMTP timeouts. Depending on the type and amount of a particular protocol on the network, the NIDS administrator may need to specify a different timeout than the default. For example, with HTTP, the timeouts would be less because the amount of traffic is usually high while telnet could have a longer timeout because of the low amount of that protocol on the network.

IP fragmentation and TCP retransmissions are normal functions that may occur on the network being monitored. By monitoring the network with a sniffer, the admin can determine the amount of fragmented IP packets and TCP retransmissions that normally cross the network. Many NIDS reassemble IP fragments before attempting to match the packet for an attack. Because of this type of reassembly, the packets will be stored in memory until the IP packet is fully assembled and then the packet will be checked for vulnerabilities. This may be another area in which changing the default

settings may better suit the needs of the environment. Again, the settings that control the timeouts to wait for IP fragments before flushing them from memory, if any, will determine the amount of memory that is set aside for this information. By seeing how many TCP retransmissions occur on the network normally, the NIDS engineer could specify different timeouts for TCP traffic that may alleviate a lot of false positives. If the amount of memory is increased, the number of connections that must be monitored is increased, or the timeouts are increased, the NIDS performance may suffer; however, these changes may need to be made in order to keep the false positives down.

To help prevent IDS evasion, or facilitate traffic normalization, the NIDS may offer some way to normalize the data before it is monitored as a potential attack. This could apply to issues like UNICODE evasion within HTTP traffic, fragmentation at the application layer, or waiting for the TCP three-way handshake to complete before checking for any attacks. There may be a method to change the way that the packets are handled prior to the attack recognition phase. The less the packets or sessions (conversations) that will have to be normalized, the more CPU cycles and memory can be dedicated to other functions, which will decrease the overhead on the NIDS.

Conclusion

We have looked at some ways that a NIDS can be deployed in a high bandwidth environment. However, this information could be used in any NIDS implementation. Each vendor's product will have different options available to it to customize the way that the NIDS handles packets. Some of the things that were discussed in this paper may or may not be possible depending on the vendor. If there is any issues or uncertainties in modifying the existing NIDS implementation to gain performance, make a copy of what is currently done and make the changes only to a few NIDS to see what, if any, performance is gained.

[Privacy Statement](#)

Copyright 2006, SecurityFocus