Computer Science

Strhuan Blomquist

Network Intrusion Detection Systems Useful or Not?

Bachelor's Project

2004:05

Network Intrusion Detection Systems Useful or Not?

Strhuan Blomquist

This report is submitted in partial fulfillment of the requirements for the Bachelor's degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Strhuan Blomquist

Approved,

Advisor: Hans Hedbom

Examiner: Tim Heyer

Strhuan Blomquist

Abstract

An Intrusion Detection system(IDS) is portrayed as the number one, must have security tool. Vendors claim an IDS can save a company from a lot of plagues like viruses and hackers. A Network intrusion detection system(NIDS) is a network based system that can monitor a network for signs of intruders. This kind of system could also be used to police a network to check employees if they stay within the guidelines set for the network, but is this legal? The questions that need answering are many but the main questions are; are intrusion detection device's really ready for large networks? Have the different IDS vendors adequately addressed the issues brought up in recent years regarding intrusion detection. Maybe the biggest issues with an IDS is; we have an alarm from our IDS, now what do we do?
This paper was written to investigate some of these questions, and to investigate, if any of these vendors actually are creating a complete and competent product that can be an asset to security personnel instead of a burden.

NIDS at the moment seem to create more problems than they solve. The biggest issue at the moment is probably that they give a false sense of security. Other administrators might start to think that they can be more casual about their own security because they got NIDS systems watching their backs. A NIDS also creates an enormous work load for the security administrator to deal with. If they want to use the NIDS to it full potential. The conclusions reached in this paper is that the NIDS vendors still have a long way to go before they will win the hearts and minds of the security community and become an asset.

Acknowledgements

I want to thank IBM Sweden for funding this paper, Leslie Moody for supervising it, and Ellinor Ancker for proof reading it.

Contents

1 Introduction **1**

2 Intrusion Detection **1**

2.1 IDS Detection Techniques . . . 2

2.1.1 Anomaly Detection . . . 3

2.1.2 Signature detection . . . 4

2.2 NIDS . . . 5

2.3 HIDS . . . 6

2.4 NIDS placement . . . 6

2.5 NIDS evasion techniques . . . 8

2.5.1 NIDS attacks . . . 8

2.5.2 Signature tweaking . . . 9

2.5.3 Signature obscuring . . . 9

2.5.4 Timer delayed evasions . . . 10

2.5.5 Encrypted traffic . . . 11

2.6 Legal issues . . . 11

3 The NIDS in the Tests **13**

3.1 Realsecure . . . 13

3.1.1 How does it work . . . 13

3.1.2 Cost . . . 14

3.2 Product Enterasys Dragon Network Sensor . . . 14

3.2.1 How does it work . . . 14

3.2.2 Cost . . . 14

3.3 Snort . . . 14

3.3.1 How does it work . . . 15

3.3.2 Cost . . . 15

4 Tests **15**

4.1 How the tests were done 15

4.2 Testing background 16

4.3 Testing criteria 17

4.4 Test network 18

4.5 Test suite 19

4.6 Deeper description of testing tools 20

4.6.1 Nmap 20

4.6.2 Havoc 20

4.6.3 Stick 21

4.6.4 Datapool 21

4.6.5 Nessus 22

4.6.6 Sara 22

4.6.7 Hydra 22

4.6.8 0-day exploits 23

4.6.9 Fragroute 23

5 Results of testing **24**

5.1 NIDS Results 24

5.1.1 Snort with ACID 24

5.1.2 Realsecure 26

5.1.3 Dragon 29

5.2 Testing software 31

5.2.1 Nmap 31

5.2.2 Havoc 32

5.2.3 Stick 32

5.2.4 Datapool 32

5.2.5 Hydra 33

5.2.6 Nessus 33
5.2.7 Sara 33
5.2.8 0-day exploits 34
5.2.9 Fragroute 34

6 Conclusions from the tests **35**

6.1 Dragon 35
6.2 Snort 35
6.3 Realsecure 36
6.4 Final conclusion 36

7 Future Trends **37**

References **38**

A Snort **40**

B Dragon **41**

C Realsecure **42**

D Dictionary **43**

List of Figures

2.1 Examples of possible Deployments for a NIDS 7
2.2 First three steps for fragroute 10
4.1 Test network Diagram 18

List of Tables

4.1 Testing specifications 19
A.1 Testing specifications for Snort 40
B.1 Testing specifications for Dragon 41
C.1 Testing specifications for Realsecure 42

1 Introduction

There is a constant battle going on every day in corporate networks. The home team is the security personnel trying to create systems that can't be penetrated or immobilized by the away team, viruses and hackers. This war will probably go on forever. The security personnel have several ways of creating a "secure" network. The most common ways are firewalls, anti-virus software and various encryption schemes. The question is how does one know that the system is not being penetrated? This is where an intrusion detection system (IDS) comes in. IDS are divided in to two categories, NIDS which is a network based system that checks a whole network, and HIDS which is a host based system that checks a single machine for breaches.

Intrusion detection has been a topic for debate over the last couple of years. The questions a lot of companies asks are: "do we need it? Is it useful? Are the already existing products any good? Should we wait for a better product?" This paper was constructed primarily to test network intrusion detection systems(NIDS). The questions to be answered are; do they have any business in an enterprise size network, have they fixed all the known vulnerabilities, and can they be used for everyday network administration like seeing trends in a network. This paper was primarily created to test NIDS solutions, some of the IDS package's tested had HIDS modules that could be installed but these were not tested

2 Intrusion Detection

A intrusion detection device tries to identify traffic in a network that has no business being there. Most IDS work by identifying known viruses and intrusion signatures. A IDS can be compared to the cameras used by customs at a border crossing. The firewall is the crossing and if a proper passport is shown a person can get let in, but customs might still pull that person over after the border. The customs department works by profiling how smuggler's behave, to better be able to identify them in a large crowd. There is a lot of different ways to do this and it would be much easier if the smugglers wore a sign saying "I am a criminal". The customs department now have two choices to identify a criminal trying to sneak by them. The first approach is to map out the known characteristics (signatures) of a smuggler and then investigate anybody fitting that description. The second approach is to inverse ones thinking(anomaly detection). Here, the custom's department creates a profile of a non smuggler and anybody not fitting that normal pattern will be pulled over. The first system is easy to use, and one will probably have a low rate of innocent people being pulled over (false positive). The second approach is a bit more cumbersome because one will probably be pulling over a lot more innocent people, but one will hopefully catch smugglers that have changed their appearance to fool the first system. Most IDS systems work by utilizing the first approach (signature checking) because it's easier to implement. The issue for an IDS is that it may have the same job as the customs department, but they normally have a much heavier work load, since there are thousands of different signatures to check for. They also have the same problem that customs do; they are always playing catch up. The more inventive smugglers/hackers are, the harder it is to identify them, and a signature can only be established when one is caught or identified by a security expert. Unfortunately there is no way of knowing how many went through before the signature was identified. Some of the newer IDS are starting to use some parts of anomaly detection to fix these problems, but progress is slow and problematic because throughput is a big issue and anomaly detection usually places a burden on performance.

2.1 IDS Detection Techniques

An IDS has only one task to perform with the traffic that passes it by, that is to decide if it is "legal" or not, raising an alarm if it isn't. This might sound simple, but research has proven that it certainly is not[1][2] [14]. The problems are many but they all stem from the basic definition problem; what is "legal" traffic?

There are a several theories on how to identify ”legal” traffic, but they all can be divided into two main theories. The first theory consists of a database of offending traffic and the IDS tries to compare that traffic to see if there is a match. The other theory is to have a database over normal traffic and to check if this is normal traffic or not (Anomaly detection).

2.1.1 Anomaly Detection

The concept of Anomaly Detection was first introduced by Dorothy Denning [6]. To make an anomaly system a reality, a definition of what is normal is necessary. The rest is classified as abnormal. There are different ways of determining what is normal, but they all have to be built around some sort of statistics or rule guides, for comparison. A complex, and ”administrator heavy” way of creating a IDS system for a network, would be to record and ”data mine” a specific network for a certain amount of time, checking what type of traffic that flows through that network segment. The next step would be to define what application ports, and protocols are being used. One can then build a rule-set on this behaviour, for instance a GET command is only allowed on port 80 and 443, thereby any other traffic using the GET command is marked as abnormal. Creating a rule-set like this for an enterprise sized network would be nearly impossible, especially if false positives at a low degree is preferred. An easier approach to ”anomaly detection” is to perform a statistical survey charting the network over a period of time, thereby constructing a database of how its being utilized. The chances of all occurrences are then computed and stored. The next step would then be to create a baseline for the chances of a certain event occurring, and then verifying each package against that baseline, and if the ”chance factor” of that packet is too low, generate an event. The great advantage of this kind of system is that it is a learning system. The system continuously records statistics, and modifies the statistical database, and its corresponding base line. Then in theory, the system should become smarter over time. This should greatly reduce the amount of false positives.

The problem with this kind of system is that if a hacker starts out very slow in their hacking attempt, and over a period of time slowly expands the limits of their attack, the guarding NIDS will not react to the intruder. The hacker has taught the system a new flow, and changed the base line for normal behaviour so the threshold on how much a statistical anomaly the system can expect will never be overstepped. Now the hacker can unhindered do whatever he/she wants to a totally blind system!

Another issue is creating the statistical database in the first place. The Utopian idea would be to create an anomaly free environment to create statistics from. This environment needs to be very similar to the live system, and then by putting the NIDS into the simulated environment create a properly initialized database. This database is later used in the live system. The problem is, how does one create a safe environment for the NIDS to gather as much statistics as is needed. It is very hard to simulate the normal traffic of an enterprise system in a lab environment. Creating the database in the "wild" is not to be recommended because it's difficult to know if the system is clean or not.

Another down side to an anomaly based system, is that when an anomaly alarm is generated neither the nature nor the cause of the alarm is presented. The only information available is that something doesn't fit the baseline criteria. It will then be the administrator's task to try to figure out if this could be a potential problem, and this will increase the administrators work load.

2.1.2 Signature detection

Signature detection is based around a signature database and comparison algorithms. The signatures can be based on a single package, or a whole session of packages, but the longer the session the more the NIDS has to work, so usually a NIDS will use as short a signature as possible. Examples of criteria to check for are:

- "Protocol stack verification": The hacker tries to brake the rules set for the protocol. "Ping of Death" is one of these kinds of exploits. The hacker sends a packet that is larger than the limit set by the ICMP protocol, causing the computers to crash.

- "Application protocol verification" Most attacks utilize strange application protocol behaviour like changing flags or using them in strange combinations looking for combinations the application doesn't cover. An attack like winnuke is a good example. Winnuke sends "junk" to port 139 this is marked as "out of band" by the NetBIOS protocol, this caused widows 95 machines to crash with a blue screen. This is typical hacker behaviour, trying to exploit an application with a poorly implemented protocol handling.

The great advantage of a signature database is its ease of deployment. A IDS signature based system should be ready to be deployed "straight out of the box", it does not require a period of learning. The other advantage is that one will get a detailed report on what "exploit" was used to hack the system. The disadvantage is that the IDS can only be as effective, if its database is updated on a regular basis and that adds to administrators work load. The other problem is that it's easy to circumvent a "signature based system", because sometimes all it takes is to change the "attack signature" slightly and the IDS will not notice the exploit being utilized.

2.2 NIDS

A NIDS has basically two components; a network card set in promiscuous mode, recording all traffic on a certain network segment, and a piece of software analysing the recorded traffic. Promiscuous mode means that the network card will process all traffic that passes it by, in normal mode the network card will only process packets that concerns it. The analysing software can utilize any of the previous described techniques. The most commonly used practice is to use signature detection with some anomaly capabilities in an attempt to make the system smarter.

2.3 HIDS

A HIDS is a host based intrusion detection system and can utilize either techniques described earlier. A HIDS has much more of an advantage of catching a hacker than a NIDS. The reason for its advantage is its ability to observe an attack on all network layers. The HIDS can do this because it's placed on the actual host, and can therefore use "full packet assembly". Detection on what the actual program is receiving is therefore much easier. It also has access to the logging capabilities of the system, and can make more educated assumptions about what a hacker is doing. The HIDS is tailor made for the host it sits on where as a NIDS needs to be more versatile because it might be guarding several different systems and this makes its job harder. A HIDS is actually a much better solution than a NIDS, if one has unlimited resources, but having a HIDS on each machine in a network is a less than attractive solution. One issue is that if forty machines have a HIDS running on them. One will have forty machines generating logs and probably traffic to a centralized logging and management point, this instead of one NIDS sending reports. HIDS will not be covered in this paper.

2.4 NIDS placement

To secure a large enterprise network a company needs to place NIDS on all vital parts of the network. The security personnel need to make a risk assessment, to determine which points on the network are worth securing. The choices an administrator has in were to place a NIDS are many, these are some of the common places where a NIDS usually is placed.

- **Behind the firewall**
  To detect people that have circumvented the firewall, or Trojans that are trying to get out of the network (or misbehaving employees). An internal NIDS will generally generate less alarms than a NIDS outside the firewall and can therefore have a harder

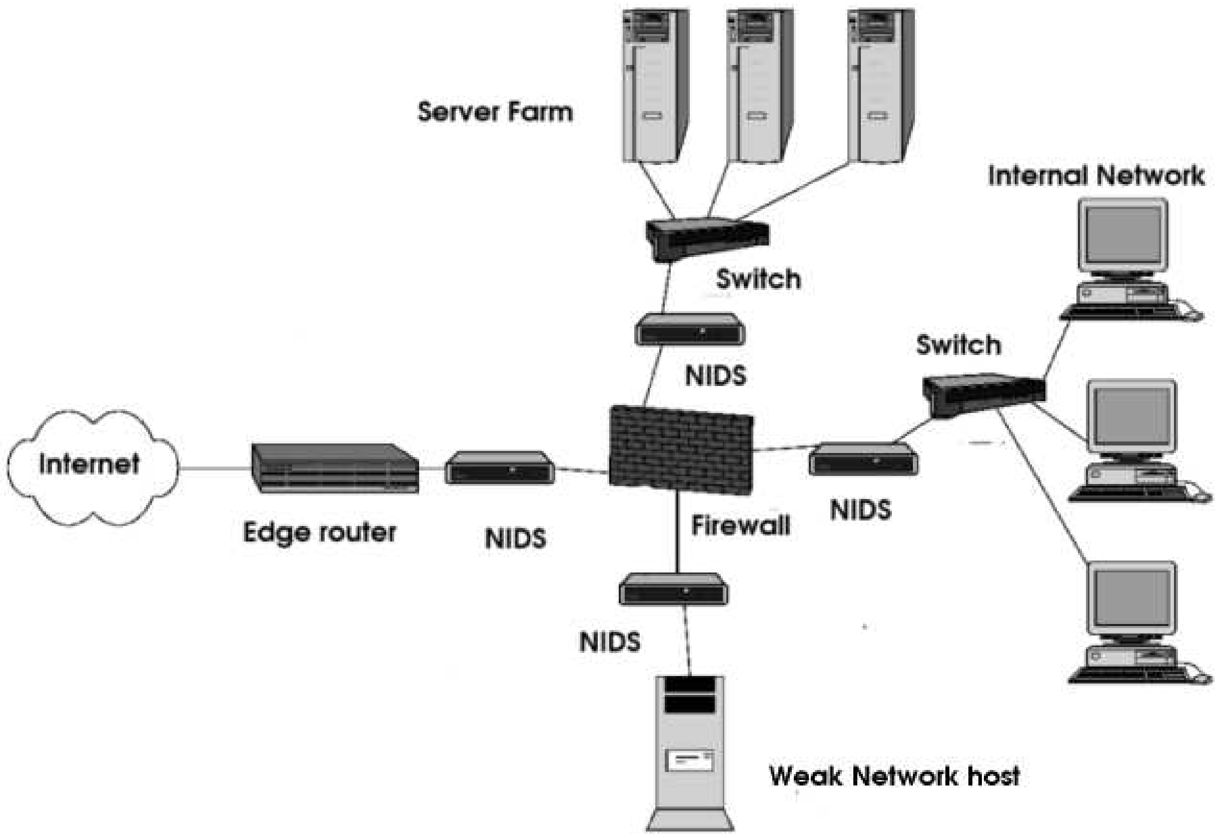

Figure 2.1: Examples of possible Deployments for a NIDS

rule-set and a lower threshold.

- **Outside firewall**

  To detect impending attacks against the network, the NIDS is placed on the outside of the firewall. The outside NIDS will generate a lot of alerts if it's not given a very narrow set of rules. A good example would be if an alarm for every port scan attempted against an Enterprise network was created, the number of logs would be huge. The sensible thing here, would be a threshold on the number of scans, before an alarm. Then maybe the chance of catching scans like the ones "Msblaster [16]" generated will improve, and hopefully the network will be saved from the next big "worm attack".

- **Network hosts**

  Hosts that cannot support a HIDS system or have poor logging skills, and therefore are vulnerable, can have NIDS in close proximity, as a sort of caretaker. This would not be the optimal solution, but it can be an easy way of verifying that the weaker system is not corrupted, and that no one is using it as a launching platform for other attacks. Another issue that a NIDS can help with, is the logging of networks where there is a fear that a hacker can get root access, and possibly cover their tracks by removing all logs on the host, even with a HIDS working on the system.

- **Server farms**

  On the backbone net of large server farms, monitoring their traffic for attacks towards the farm, or on individual servers that have a high level of traffic travelling to them, and cannot have a HIDS system.

2.5 NIDS evasion techniques

There are several ways of eluding a NIDS[17]. Most hackers are expecting a NIDS to guard a corporate network and have therefore developed a couple of ways to elude a NIDS. There are even specially designed tools to fool a NIDS like Stick (paragraph 4.6.3 page 21) and Fragroute (paragraph 4.6.9 page 23).

2.5.1 NIDS attacks

The object of this technique is to stress the NIDS into going down, or not be able to log the attacks. There are several ways of doing this, and there are some tools that have been created to make this possible. Stressing the NIDS to go down entails creating a "distributed denial of service"(Ddos) attack that will overwhelm the NIDS and make it crash or not log the actual attack. To launch a Ddos attack one could use automated tools like "Snot" and "Stick" (paragraph 4.6.3 page 21). The other way is more devious, and involves fingerprinting the NIDS. This is done to discover what NIDS are running and where they are located. This can be difficult because a NIDS is a passive device, only listening to the network and does not interact with it. Since a NIDS usually wants to alert a centralized management point, a hacker can catch these messages. With these messages a hacker may be able to discover what kind of NIDS is placed on the network and where. Then, by utilizing some exploit on the actual NIDS machine, bring it down, thus making the network blind.

2.5.2 Signature tweaking

By slightly changing an exploit, and thereby not fitting the NIDS signature for that exploit, evading the NIDS may be successful. The chances of such an attack would have a very high success rate, if the hacker can figure out what kind of NIDS it's trying to trick. If the NIDS is an open source system, like Snort, the hacker can then download the actual signature it's trying to avoid, and then check to see how he/she can alter the attack pattern as to evade the NIDS.

2.5.3 Signature obscuring

This involves making the signature harder to understand but not altering it. The easiest way to create this effect is to use a tool like Fragroute. Fragroute takes a packet and breaks it up into very small pieces and then, by sending them in a jumbled order it tries to trick the NIDS. The normal way a command like "cat /etc./shadow" would travel to a server would be in one packet. Fragroute on the other hand would divide the commands letters into byte sized packets. Fragroute would for instance put the "c" in one packet the "a" in another packet and so on. This would mean that fifteen individual packets are created. Fragroute then rearranges the order of the packets. The "c" is now packet six instead of the packet one, and "a" might be in packet fifteen and so on. Now the order is all mixed around, but the reassembly process on the other side will put the packets together in the

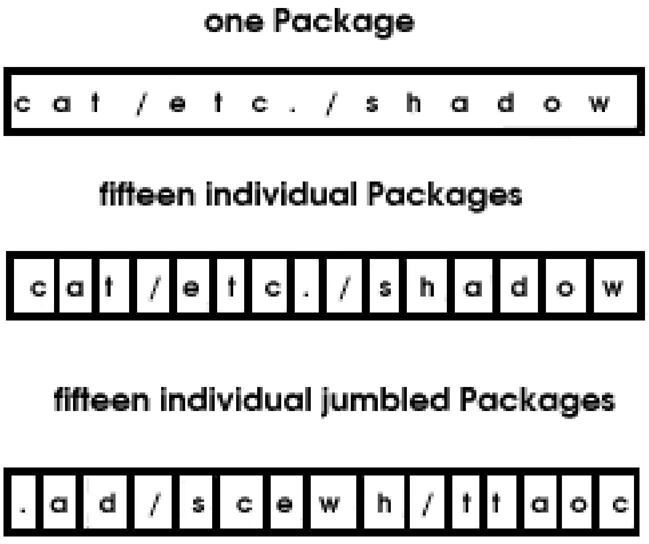
one Package
c a t / e t c . / s h a d o w
fifteen individual Packages
c a t / e t c . / s h a d o w
fifteen individual jumbled Packages
. a d / s c e w h / t t a o c

Figure 2.2: First three steps for fragroute

correct order. Then Fragroute creates some garbage packets that it sends in between. In the example above there were fifteen packets from the beginning, but now their might be forty packets, but only fifteen are important. To really make the NIDS confused Fragroute can pause a bit among some packets. This kind of attack will make it very hard for a NIDS to match the signature with its database. The NIDS needs to do a full "stream assembly" to be able to view the entire command, and that will create problems because different systems assemble packets in different ways, plus it gives the NIDS a lot of extra overhead.

2.5.4 Timer delayed evasions

Here a hacker will do a very slow attack, or probe, where each part of the attempt will have a time delay between each attempt. An example would be a dictionary attack against a password protected server, where there will be a one minute delay between each tried login. Port scans are also an easy timer delayed NIDS evasion technique, where there will be a delay between each port probed.

2.5.5 Encrypted traffic

If the server that a hacker wants to exploit has an encrypted channel, then the easiest way to evade the NIDS is simply to use this channel. This is because the NIDS has a more or less impossible task to detect attacks through an encrypted channel. This is a big problem for a NIDS as networks are utilizing more and more encrypted traffic. If a host is exposed to hackers and uses encrypted channels it would need a HIDS to protect it. Because a HIDS sits on the actual server it can catch the traffic after it has been decoded by a lower network layer.

2.6 Legal issues

There is a EU committee working on the subject of Intrusion Detection and they have released a working paper on the subject. Whatever conclusions they will reach, will be concluded in guidelines for the EU [20] which should prove interesting for future IDS.

At the moment though, IDS systems seem to be in a bit of a legal gap. A commonly used metaphor for NIDS is to compare them to a video camera watching and recording everything that is happening. One can continue with this analogy and look at the law in Sweden concerning the subject of surveillance cameras. Swedish law states that you need permission from the county council (1998:150 About public camera surveillance) to set up a surveillance camera. In addition, one needs to put up clear signs that inform the public that they are under camera surveillance. This metaphor can be applied to a company wanting to set up a NIDS on a public network. This means they will probably need to put up some sort of warning, or sign on a general information site, saying that they might be recording the visitors session on the net.

The law is a bit different when applied to a common workplace. The law states [4, 22] that an employer can install a video camera on a non public place without a permit if all employees are informed. The law seems to have been constructed around the same view in the regard to company networks, where a network can be under company surveillance for security reasons, without any previous applied permission. Usually a company covers its back by letting all employees sign a contract saying that the company can check their traffic at the company's own discretion. By doing this the company might be legally covered, but to a global network there is usually a lot of traffic coming from the outside and those visitors might not have signed any contracts. This is where things get a bit controversial. If one has a situation where one person "Bob" has signed a contract in which he states that his traffic can be monitored by his employers. Then we have a second person "Alice", who wants to chat with Bob, but Alice doesn't work at the same company as Bob. The company, that Bob works for can still monitor their chat because one person, involved in the communication has signed a contract giving his permission. The issue that arises here is that if, for instance a customer is using a company's sales site, then it's technically not another "person" sitting on the other end, but the company can still monitor the customers' traffic to their site for security reasons.

The other legal issue, is that if an employer has set up a NIDS, and it's set up for security reasons then the company cannot use information from that "surveillance" to evaluate an employees performance [4, 21]. This is where problems can occur. If a company has a policy that informs the employees that at any time they can monitor their internal network to verify internal guidelines are being kept. If then, the company is checking to verify that an employee is not surfing to pornographic homepages, then this employer can only gather this information for the specific purpose of checking for "that" offence. They cannot use data previous gathered by a NIDS, because they set up the NIDS for security purposes. [4, 20] [5].

A more serious problem can occur if a company is not vigilant. What would happen if two company's are in a conflict with each other? Let's call the conflicting companies A and B, lets presume a third company C handles the computer network for both companies. Company C has an intrusion detection system which checks traffic for both sides. Now, what if company A wants to sue company B and A wants the NIDS logs for communication between company A and B? This puts company C in a "conflict of interest". There is no clear cut way out of these kinds of situations if company C has not taken situations like this into account when they drew up the contracts between companies A and B.

3 The NIDS in the Tests

3.1 Realsecure

Realsecure version 7.0 with signatures collected 040407.
Realsecure will run on Windows, Sun solaris, HP-UX, AIX, IPSO and Linux but the management console Siteprotector will only run on Windows. ISS supplies a full package with management system called Siteprotector, a vulnerability scanner, NIDS and a HIDS. Realsecure also has an analysis tool called "Security fusion". Security fusion claims to be able to detect if the attack was successful or not, and can gives an impact assessment and a broader view of attacks, because the module can correlate data from more sources than simply the alert. Realsecure can generate SNMP traps for events and has modules to port it to many different systems and OS's. Realsecure needs a database to collect its information to, for instance MSDE2000 which is included with the installation package. Realsecure has Active response capability's to block intruders from misusing a network.

3.1.1 How does it work

Realsecure is a signature based system. Realsecure is different from the two other NIDS tested, in that it that they use what Realsecure calls "state full operations" to help lower the rate of false negatives and improve performance. Realsecure also claim to do full packet reassembly and therefore claims to be immune to fragmented attacks and Ddos attack like

Stick and Snot.

3.1.2 Cost

Starting at `$9000` per server. For further info `www.iss.net`

3.2 Product Enterasys Dragon Network Sensor

Dragon with a signature file collected 040325.
Dragon is a Unix based intrusion detection system and will run on most Unix systems. Dragon has a web based management tool which needs a web server installed to work. Dragon comes with a NIDS and like Realsecure, it also has a HIDS in the package, and an attack generating tool to test the system. Dragon, like Realsecure, has an active response capability called "Event snipering" were an intruder will be blocked from any future abuse of a network. With these capabilities it can alter switches, routers and firewalls to block out a malicious person.

3.2.1 How does it work

Dragon is signature based, but claims to employ some anomaly detection qualities. Dragon's anomaly detection has a learning mechanism to make it more effective to the environment in which it has been placed.

3.2.2 Cost

Starting at `$3000` plus. For further info `www.enterasys.com`.

3.3 Snort

Signatures collected 040326.
Snort is a light open source NIDS developed by Martin Roesch. Because its open source,

signatures are developed and updated at a very high rate. Snort has several different ways to be deployed. In this testing case Snort was deployed using ACID (Analysis Console for Intrusion Databases). Acid is the most widely utilized log analysis tool for Snort. Snort can run on most OS and hardware and has plug-ins for a lot of different features like SNMP, SSL, IPS and Tivoli. If an administrator wants commercial support for their Snort there are company's that provide this like `www.silicondefense.com` and `http://www.sourcefire.com`

3.3.1 How does it work

Snort, like the other NIDS in this test suite, work with the help of a signature database. Snort like Realsecure claim to be able to handle different states of a session but not at the scale of Realsecure. At the moment Snort has problems with gigabyte networks.

3.3.2 Cost

Free open source product. For further info `www.snort.org`

4 Tests

To verify the claims of the NIDS vendors and the testing how well NIDS work different tests were performed.

4.1 How the tests were done

The tests were done by setting up a test network and running a test suite at each NIDS individually. The tests were run three times at different intervals to check that the same results were reached. The background load was set at about thirty Mbit to give the NIDS something to work with. The background traffic consisted of previously recorded packets from a live network. The previously recorded traffic was captured with the tool

"Tcpdump" and then replayed with "Tcpreplay", both open source tools for Linux. The captured traffic has been run through Snort previously to investigate if the recorded traffic had any alarms that would disturb the testing process. Recorded traffic that triggered alarms was discarded. The NIDS's ran on a Pentium 800 Mhz INTEL box with two hard drives, one with Linux 8.0 and the other one with Windows 2000 server. Dragon and Snort ran on the Linux harddrive and Realsecure on the Windows harddrive. The victim was a Windows 2000 server running IIS and SMTP pop3 server. The IIS was running a webserver and FTP server with an optional SSL connection. The server had no patches installed or firewall. The victim had Ethereal running to confirm that all traffic and attacks reached the victim. The tests where designed to test how well the NIDS's worked "out of the box" so to speak. All NIDS where tested running their standard default settings.

The attack tests main purpose was to determine if an alarm was generated when the attack was staged. Then by examining the log, investigate if the alarm corresponded to what attack was staged, and how many alarms where generated for the attack. One big issue in the test was to see how well the different NIDS dealt with NIDS evasion tools like Fragroute, Stick and slow probing. The tests also involved how well they dealt with their immediate environments like their own LAN segment. If a NIDS doesn't control its own LAN segment a hacker might want to set up capturing device of their own, in order to fingerprint the NIDS to better circumvent it.

4.2 Testing background

By studying previous benchmark results [17] [23] [18] [26] [22] It was concluded that Snort, Realsecure, Dragon and Cisco were the leading systems. Since Cisco did not have a software version and were in the middle of developing a new version of their IDS. They seem to have put the old one on the "back burner" so they were excluded from the test. This was unfortunate, because they seem to be the leading company concluded from previous tests[23] [18] [26] [22]. The test should have included Checkpoints new solution. It is not really an IDS, but an internal security gateway with NIDS analytic capabilities to both alarm and take preventive actions. Checkpoint claims that their solution is not "signature based" but "anomaly based" and it would have been very interesting to test it. But this solution was a hardware one and they were hesitant to let it be tested.

The test suite was created by studying some benchmark test evaluation papers [13][27].This in order not to make the same mistakes that previous testers have done.

4.3 Testing criteria

Different elements to test the different systems towards?

- **Diagnostic:**
  How easy is it to identify, and find the computer that is misbehaving.

- **Reports:**
  How easy is it to make different views, and reports, with the tool so different departments can specify what they want to study from the NIDS?

- **Ease of deployment:**
  How long from receiving the product, until it's up and running, with or without Internet connection?

- **Ease of use:**
  Does it require training to use or is it self explanatory?

- **Customization:**
  How much and to what degree.

- **Vulnerability**
  How vulnerable are they to different NIDS evasion and attacking tools?

- **Effectiveness**

  How effective are they at catching hackers and viruses trying different exploits and attacks?

4.4 Test network

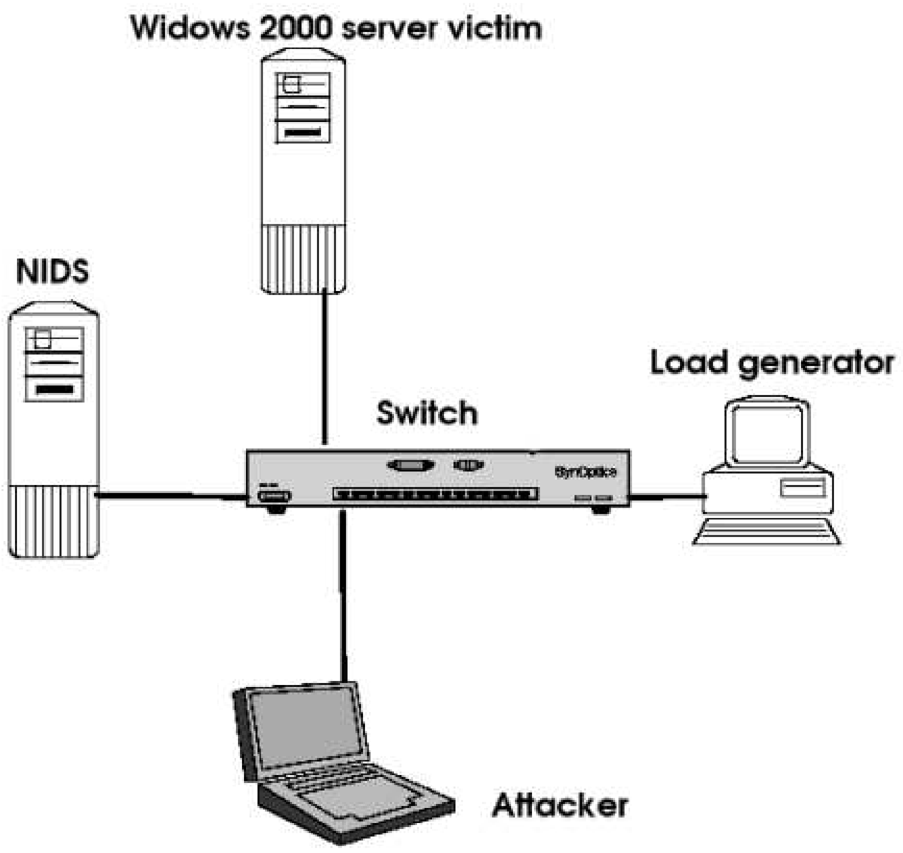

Figure 4.1: Test network Diagram

NIDS: Intrusion Detection system to test

Load Generator Traffic recorded from a real network. The traffic adds to the work load the NIDS needs to traverse. Send rate at about 30 Mbit/s

Windows 2000 server: A Windows 2000 server running server applications, This is the victim of the attacks. Ethereal is running to check that the packets arrive.

Attacker: A laptop that is launches different kinds of attacks towards the server. The attacker is running Linux Redhat 9.0 on IBM thinkpad.

4.5 Test suite

Test	**Flags**	**Short Description**
Nmap	-sS	Syn scan for port
Nmap	-sF	Stealth scan
Nmap	-sX	Stealth scan
Nmap	-sN	Stealth scan
Nmap	-sU	UDP scan
Nmap	-f -sS	fragged scan
Nmap	-D -sf	spoofs several IP to hide the scanning
hydra	-C logpass ftpserver.se ftp	dictionary attack ftp 57 attempts
hydra	-C logpass ftpserver.se ftp -t 20	57attempts 2 tries with 20 second delay
hydra	-l admin -P pass ftpserver.se ftp	900 attempts with buffer overflow
havoc	-i eth0	random ARP traffic generator
datapool.sh	-d victim	Ddos attack
winnuke	host port	ping of death windows exploit
stick		stick IDS flooder
stick + winnuke		drowning the attack
fragroute + hydra		fragmented hydra
tesoiis	host port url	iis 4.0 exploit by eeye security old
fragroute + teoiis		fragmented attack
fragroute + winnuke		fragmented attack
stick + hydra		trying to hide the attack
sara		security audit tool
nessus		security audit tool

Table 4.1: Testing specifications

4.6 Deeper description of testing tools

The different tools, used in the tests were chosen with respects to their different abilities and that their common techniques are used by hackers and viruses. Most of the tools used are widely known and can easily be found on the Internet.

4.6.1 Nmap

Nmap[8] is a free, widely used scanner that administrators and hackers use to explore networks. It can rapidly scan a large network for different services, and has a lot of different ways of scanning, some stealthier than others. Nmap also has OS identifying capabilities, that identify an OS by mapping their response times to certain events, and reactions to certain packets. Nmap was included as a good measure port scanner to test how well the NIDS reported, and found the different stealthier types of scans. Nmap also can fragment its traffic, but not at the degree of a tool like Fragroute. Nmap can use very slow probing, which could make it harder to catch, or spoof several different addresses to make it harder to recognize who is scanning. Nmap is such a common tool that all NIDS systems should have all its capabilities mapped out.

4.6.2 Havoc

Havoc[21] creates random ARP traffic and puts a network segment at a standstill while it is running. Because it's so local, and low level, it's hard to detect and pinpoint. Only a certain net segment will be affected, so only if the NIDS is listening to that segment would it notice this kind of attack. The attack was created to see if NIDS keeps track of its local environment on a lower level, considering ARP-poisoning could be an indication that somebody is trying to fingerprint the actual NIDS itself. This tool is much less subtle than ARP-poisoning but for testing purposes this tool should be easier for them to catch.

4.6.3 Stick

Stick[9] is a NIDS flooder otherwise known as a Ddos attacker. Stick utilizes Snort rules to create random packets designed like the attack signatures, thus creating a NIDS alarm storm. This creates huge NIDS logs which could cause a lot of stress on a poorly built NIDS system. It can be used to either stress test a NIDS, or to evade it, by creating so much noise, a real attack gets lost in the huge log of attacks it generates. One other aspect of the test was to investigate if an administrator could get an alarm saying ”this is a Stick attack”, instead of trying to conclude it by the shear mass of alarms, and therefore missing any real live one’s.

4.6.4 Datapool

Datapool[25] is an old collection of typical ”script-kiddie scripts” to attack a network with. They are not that dangerous any more, because of their age, most of them are five years old. Some of the Ddos flooding scripts could still be dangerous if utilized by a large number of computers. The tool was chosen because of its age. Every NIDS should recognize the scripts run by it. The test also checked if a tool like this could overwhelm a NIDS. Datapool has an automated tool that can run all the scripts in its arsenal at varying speeds, and amounts, thereby overwhelming a victim. Datapool has 106 different scripts that can run ranging from IIS exploits, ”flooder ”,”nuker” and ”dos attacks”. This should create as much traffic as Stick, but the difference is that these are actual attacks, and not just signatures of attack, like Stick creates. Because Stick only simulates attacks a NIDS can probably disregard them depending on the implementation of the NIDS, thereby limiting its workload. Datapool also has a source and binary directory, if one wants to run a script individually. Some of these scripts were used in other tests, involving Stick and Fragroute. For those tests, Winnuke (which is classic NetBios attack for Windows) and TEOISS (an ”IIS GET request” that crashes web servers) where used.

4.6.5 Nessus

Nessus[7] is a vulnerability-scanner that is developed for system administrators to check if a system is secure, or not, by checking how open and vulnerable a system is against different attacks. Nessus performs its tests by utilizing a database of common attacks and exploits, testing them against the system and observing if the system responds. Because Nessus simulates attacks against a system a listening NIDS lights up like a Christmas tree and it is an interesting test of the system. The other thing to test was if the NIDS actually could identify a vulnerability assessment tool. The reason for this is to see if the NIDS can differentiate between an attack and a probe. An administrator would be more appreciative of an alarm that informs them that this IP is running Nessus, instead of a NIDS that spurts out 200 alarms making an administrator very worried, thinking the system is under a full frontal attack when it just is somebody probing it (administrator should be concerned with the probe though!). The other reason why Nessus was chosen is that it's starting to become a security standard, sufficient to say the NIDS developers should have no problems identifying it.

4.6.6 Sara

Sara[3] is the same kind of tool as Nessus, but based on Satan and older. It should therefore be even easier for the system to identify it. This test was also designed to observe how similar these scanners are, and if the different NIDS's could differentiate between Nessus and Sara. This gives an indication of how much time a NIDS vendor has spent customizing their product.

4.6.7 Hydra

Hydra[11] is a dictionary attack tool to attack ftp, websites, telnet,pop and IMAP. It is very fast and can divide the attacks in small tasks, to be less obvious to a NIDS. This makes a NIDS having to check time intervals, and amounts to be able to identify it. Hydra also gives an indication on how high threshold a system has, if it warns for three failed attempts, or fifty, this will be an indicator on the "sensitivity level" of the system .

4.6.8 0-day exploits

Three reasonably new exploits where downloaded to see if the signature files that where being used from the vendors sites, kept up. The first two where older than the downloaded signature files, the last one is younger than the signature files. The newer exploit was tested to see if the "anomaly detection" the different vendors claim that they have implemented,is able to catch a new exploit? All exploits were downloaded from `http://www.k-otik.com/` a French security homepage. The exploit was download in source form and were then compiled and tested.

- 04.14.2004 Microsoft IIS SSL Remote Denial of Service Exploit (MS04-011): Causes IIS severer SSL function to freeze up and become unresponsive.

- 03.28.2004 RealSecure / Blackice ISS_pam1.dll : Remote Overflow Exploit attack on the NIDS itself causes disturbance in communications.

- 02.14.2004 Microsoft Windows ASN.1 LSASS.EXE Remote Exploit (MS04-007): Kills the lsass.exe process by using a flaw in the WINS net-bios port 139. Causes a message to appear on the victim informing the server that it will reboot in one minute.

4.6.9 Fragroute

Fragroute[24] works by utilizing the evasion attempt described in paragraph 2.5.3 page 9. To utilize Fragroute one tells the program, which target one wants to send the obscured packets to, and to which degree it should be altered. Fragroute will take all communications with that victim and try to obscure them. Fragroute has some problems, and sometimes it will create such a mangled communication that the victim cannot assemble the message. This means that the message needs to be resent and after a while the message will be deciphered .

5 Results of testing

Each test result is presented according to the test specifications outlined in paragraph 4.3, and all results concluded from each individual testing software represented in paragraph 4.6. The first part describes how the actual individual NIDS performed as devices, and second part, the testing software is presented according to each tool tested, to give an easier overview of how each piece of software affected the NIDS. The last two parts(Vulnerability and effectiveness paragraph 4.3) of the test specification was tested with these tools

5.1 NIDS Results

5.1.1 Snort with ACID

- **Diagnostic:**
  ACID gives a pretty clear view of who is attacking and it has capabilities to utilize a "whois database" depending on if you have the NIDS connected to the Internet or not. The issue with Snort is that if you have the time, one can find great diagnostic tools to do "data mining" and reporting with, but the standard ACID installation that was installed had very little in the way of diagnostic tools. ACID is mainly a way to give a web presentation of what alarms are being flagged by Snort. Acid has a search function to search for alarms in the SQL database, where the alarms are stored. This in the hope of creating statistics, and diagnose problems over longer time period.

- **Reports:** As stated in this test, ACID was used. ACID is a pretty straightforward system and the layout is easy and comprehensible, but it lacks a bit in the way of automating reports. In the quest to improve Snort, a better tool for making reports was tried, but it would not work correctly, and created database errors. But as stated before this is an open source product and there are several other hopefully "well written report tools" for Snort. Snort has a module to generate SNMP traps, but it's not standard and is a challenge to install. It took about three days of experimenting to get it to work properly!

- **Ease of deployment:**
  Snort can be deployed in nearly any OS systems, but if the system is going to cost as little as possible then Linux running Apache and PHP with ACID is the preferred option. The time to set up depends on the administrators Linux skills. The time it took to install the test NIDS was about a day starting with a clean formatted hard-drive, downloading all the different dependencies and deploying them. Dependencies can easily be the biggest problem, while implementing Snort. There is a lot of packages that need to be downloaded(up to 10 depending on what OS). Implementation would be easier if the Snort developers would put together some software packages, including all necessary parts for some of the leading OS's to help with installation.

- **Ease of use:**
  There are several different ways to utilize Snort. Snort can be used in different modes, packet sniffer,packet logger and intrusion detector. The intrusion detection mode is easiest when started utilizing the file "snort.conf" which gives a nice overview of all different "flags", and commands that can be set including a standard default setting. Each line of the commands have text descriptions of options and that with the help of the snort manual is easy to use.

- **Customization:**
  This is a bit of a cumbersome procedure in Snort. There are some customizations

tools, but they are still in a "beta stage", and can be a bit unreliable. The big problem with snort is that it doesn't have any "built in support" for updates, and rule-changes this can put a strain on administrative routines. Every time you get an update one has to do a lot of checks before one implements it, not to lose all your old customizations. The problem is that the new rules can easily overwrite the old ones and in doing so lose any comments,customisations and own rules in the process. Once again there are some "coca cola" crowd "hacky tools" to help with this but they can be "iffy".A good policy is always take backups, and do not depend on these tools. In testing one of these "automated tools " progress was delayed because the rules needed to be reconfigured.

5.1.2 Realsecure

- **Diagnostic:**

  Realsecure has a nice interface, it is well laid out and it's easy to diagnose what the problem is and where it originated from. One problem occurred though that made testing of this product incomplete. The "Security fusion module" kept on complaining that the distributions where not compatible. Trying to upgrade everything to the latest version, resulted in the same result. An older release was tried and but the problem was still present. If the message was disregarded, then the Realsecure application server went down and never recovered. The message that some parts of Realsecure might not work if the installation was forced were not wrong.

  This was very unfortunate, as the report on this part cannot be completed. The "Security fusion model" claims to be able to analyse and see if an attack succeeded or not. They claim that it has a smoother sorting, where the attack patterns can be analysed and the potential threat be reported, to better give an administrator a finer view of what has occurred. ISS seems to have listened to previous benchmarks [23] in the design of this module. All diagnostic power of Realsecure is linked with the "Security fusion module" so all testing only results in a message reporting that one needs this module to run the diagnostics.

- **Reports:**
  Here Realsecure has taken reporting capabilities seriously. Realsecure has automated reporting tools that can generate reports and send them wherever they need to go. Realsecure has built in support to send SNMP traps, so alarm can be generated and picked up by SNMP administrative tools. Realsecure also can be implemented in Tivoli, IBM's monitor management tool for NIDS.

- **Ease of deployment:**
  Installing this software was not without problems. The part that involved testing how easy Realsecure was to install and update without an Internet connection to the actual machine, presented some software bugs. If the machine was connected to the Internet it took approximately 6 hours to install and update all the parts of Realsecure and the machine was up and running without any problems what's so ever. When the test was conducted on a machine not connected to the Internet, it took three days and about six installations before it started to run smoothly. The Realsecure installation process did not even want to start if a network connection wasn't present. While trying the install and uninstall procedures a peculiarity arose. The uninstall program only worked half of the time! This created some interesting hours trying to remove any trace of it from the Windows registry. The program did not want to install until all signs of the old program were removed. In desperation the test was tried by installing all the software from the Internet, then removing the box and inserting it into the test network. After this the Realsecure application database would not start. It gave a "code 260 can't start the application", and there was no documentation for this error message. After some assistance from Realsecures help desk it was concluded that this code could originate in the "ini" files which had the

old IP address and didn't understand the change of IP address associated with the new environment. An attempt to change all files that pointed to the old IP address and DNS name in the hope of fixing the problem was in vain. The only solution was to reinstall the software.

To get updates remotely, Realsecure has a tool that can be used to download all updates on an alternate machine and then inserted into the customers Realsecure. This is done by transferring all the files that have been downloaded with the alternate computer to the Realsecure computer, and issuing the update command on each individual module. After this, the new updates should be found and installed according to the manual. The components in Realsecure could not find any updates and claimed there where no available updates. After a bit of searching a fix was found by accessing the Realsecure core module properties and when closing it again all the updates are found. This fix is a bit unconventional and it seems that the coding for this software needs some work. Another problem occurred when an attempt was made to add a second network sensor from a second machine, the activation key would not work. A customer support call fixed this problem. The fix involved removing the new sensor as an asset and then let Realsecure go out and search for it, add it again, remove all keys and then add them again. This fixed all key problems.

- **Ease of use:**
  Realsecure seems to have spent some time on their management module, which is very easy to use and it's easy to get an overview of what is going on within the system. Realsecure creates a lot of different views, statistics and generates specialized reports. Realsecure has automated update capabilities if it is connected to the Internet. The tool informs the user that there are new updates and they are ready to be installed at the press of a button. Realsecure should be easy to anyone used to the Windows environment.

- **Customization:**

  Customization is pretty easy, it's just a case of switching on and off different rule sets. The customers own signature can be written with the help of an API. Most signatures these days are produced by users of Snort. The Snort signature system has become something of a security standard. Because Realsecure is "closed software" a signature like Snorts can't be used. There is a way around this though, which is called Trons.[10]Realsecure utilizing Snort signatures with the help of the Trons modules can't utilise the full identification algorithm that Realsecure employs, but it is under development. Realsecure can send a response to modify a firewall using OPSEC, this to lock out the offending IP address from the network. The offending IP address can fully automate Realsecure and make it compatible with other security vendors.

5.1.3 Dragon

- **Diagnostic:**

  Dragon has a bit of a cumbersome interface where understanding the layout is not straightforward. There is a lot of scrolling windows and some of them are set always to go back to a standard value and some of them do not, this creates a lot of ambiguity. But it has a lot of different features and uses. The management tool gives a lot of information about whom and what is attacking. One tool that stands out in the Dragon console is "mksession" which recreates an attack scenario and lets the administrator analyses it to observe what happened. This tool is a perfect diagnostic tool.

- **Reports:**

  Dragon has a tool for generating reports and statistics which gives a pretty nice overview. Dragon can be integrated into Tivoli for management. Dragon has a well made data mining tool which lets you go back into the database and retrieve old statistics. This to get a better overview of what is occurring over a period of time. Dragon also has an extra packet called a "trending" console that can be separately installed to generate reports. The installation is a bit cumbersome and takes a while to get going. The "trending console" automates the process of seeing patterns over a fixed time frame.

- **Ease of deployment:**
  Dragon was built for Linux and works well in that environment. One has to meet some requirements concerning Apache, Tomcat and so on to get it going. The thing that creates a bit of problems was their browser support. They only supported an old outdated version of Netscape on Linux and an just as old Explorer. This could be a bit of a problem having to use a specific browser just for Dragon. Utilizing a newer version browser results in a lot of failed requests. Installing Dragon is pretty easy. Dragon comes in a rpm build and with an install script that gives Dragon an easy setup. It took a couple of hours and then it was up and running. Dragon had no issues being installed connected to a network or not as Realsecure had.

- **Ease of use:**
  Dragon is not the easiest to use as it has a couple of idiosyncrasies that are not straightforward. It takes a day or two to figure out how to use the Dragon web interface correctly. The push configuration can sometimes be a source for some problems. The problem is that some days it will take a push for a new "config" right away without any hangups,and then the next day it gives a lot of errors and cannot push the update even though one is doing the same thing as yesterday. Another weird thing happened while trying to change the "network range" Dragon was supposed to listen to. After the change the network sensor would not start. This problem disappeared after the change was reversed and dragon was restarted. The range could then be changed and restarted a second time.

- **Customization:**

  If one come to terms with the Gui, customization is integrated into the system. The web Gui can change all configuration files, and push changes to all network sensors and Dragon has an automated upgrade tool for signatures. This tool worked without a hitch. Dragon has several different modules to be able to customize Dragon for the needs of its environment.

5.2 Testing software

5.2.1 Nmap

Realsecure identified all scans by Nmap, two of them Realsecure even identified as Nmap scans. Slow scans between 15 seconds, and 5 minute, Realsecure flagged after the first scan attempt. Fragmented scans Realsecure found, and correctly identified the scans, and flagged the fragmented packets as a possible IDS evasion attempt.
Snort caught all scans, but generated a lot more individual alarms than Realsecure and Dragon. The slow scans were caught but it took a bit longer to catch than with Realsecure. Snort like Realsecure identified some of the scans as being generated by Nmap. Fragmented scans seem to create a bit of a problem for Snort. Snort creates a lot of alarms about fragmented packages, and generated alarms that somebody was scanning the victim. As stated before Snort generates a lot of individual alarms when scanned by Nmap but the actual alarms concerning the scans where less when fragmented. The conclusion drawn, was that some of the alarms seem to have been missed, and the order in which alarms got lost seemed to change between each test, which could not be explained.
Dragon identified all normal Nmap scan and stealth attempts. Previous benchmark stated that UDPscans [22] weren't identified by Dragon, these they seem to have fixed with this version because Dragon had no problems identifying a UDP scan. Dragon does not identify any of the scans as actually coming from Nmap, only as TCPscans. Dragon does not create

a lot of different alarms, like Snort and Realsecure, for the different scan attempts Dragon describes all the scans as the same! Dragon has the one flaw, when slow scans were being tested Dragon missed all the scan attempts. Fragmented packets did not seem to present a problem for Dragon, the evasion attempt was identified and the scans where presented as correct alarms. The only issue with a fragmented attempt,is that it does generate a lot more alarms.

5.2.2 Havoc

Snort and Dragon did not notice when Havoc ran, they do not seem to have had this kind of local vulnerability view when they designed their NIDS systems.
Realsecure generated an enormous number of alarms for IP_DUPLICATE. The interesting side-effect was that Realsecure came to a screeching halt! Other attacks during one minute "Havoc" session resulted in delays between a minute, and 10 minutes before an alarm could be logged. Havoc also caused big memory surges to Realsecures monitoring device. On one occasion the machine stopped and needed to be rebooted.

5.2.3 Stick

All NIDS systems seem to stand up well to Stick attacks, but stick does generate a lot of noise. The systems generate a lot of alarms when hit by Stick. This by itself might be a source for concern, because real attacks can disappear in the shear volume of alarms generated by stick. No NIDS systems identified stick as the tool attacking them, or any other automated attack pattern tool being used.

5.2.4 Datapool

Datapool generates huge logs like Stick and all the different NIDS seemed to stand up well. Most of the individual scripts were correctly identified and all where caught. Datapools database of old "script kiddie" hacks seem to be covered by all NIDS.

5.2.5 Hydra

Dragon and Realsecure reacted to Hydra in the same way, they created alarms both for slow attacks buffer overflow attempts, and low number of attempts only 20 tries or less. Dictionary attacks seem not to present any problems for these NIDS's. Both of these systems seem to have a low threshold for failed admin attempts though, which in the long run might produce a lot of false positives.
Snort seems not to have taken dictionary attacks into account. The test ran up 200 hundred attempts and Snort did not flag the traffic as illegal. Snort only reacted to massive buffer overflow attack against a ftp site with long passwords. It seems that Snort doesn't think 200 hundred failed attempts with admin as a login name is worth mentioning!

5.2.6 Nessus

Nessus generated large alarm logs on all three NIDS's. All three systems identified some of the alarms as being generated by Nessus. Realsecure and Dragon only wrote up one of the alarms generated by Nessus to actually be made by Nessus. Snort identified several of the alarms as being of "Nessus origin".

5.2.7 Sara

Dragon was the only one that correctly identified Sara, as the scanning tool used. As in the case with Nessus it was only one of the alarms that were identified as Sara.
Realsecure identified Sara as being Satan, because Sara is a further development of Satan it seems likely that Realsecure just needs to update their identification database.
Snort was the NIDS that did worst in this experiment, it identified several of the scans as being generated by Nessus and none of the alarms were ever identified as SARA or Satan. Which means Snort cant differentiate between scanners. This could create problems for an administrator in the long run.

5.2.8 0-day exploits

Dragon caught the lssas.exe exploit, but the other exploit's went by unnoticed by all three NIDS systems. All the exploits caused problems on the victim which is unfortunate considering the disappointing results. These tests show that 0-day exploits are still a big issue for NIDS, and the "anomaly detection" that has been added seems not to be able to handle new exploits.
The Realsecure exploit was only tested against the Realsecure NIDS because the Realsecure machine was the intended victim to test against. The Realsecure server did not notice the exploit and after the exploit was executed Realsecure Network sensor was non responsive. So it seems that the exploit was successful!

5.2.9 Fragroute

This was a real test for the different NIDS systems considering this tool has been on the market for some while, and their has been a lot of interesting articles about the tool [12]. The evasion technique that fragroute utilizes seems to still be a big problem though.
Dragon and Snort apparently have not had much luck fixing their NIDS against fragroute. Both Dragon and Snort have the same results, they both create alarms that fragmented packets are being found, but they cant identify the specific exploit. This has the affect that the actual attack is totally hidden from a security administrator and any attempt to find out if the attack was successfully cannot be made using the NIDS.
Realsecure being "statefull" and utilizing full packet reassembly, is outstanding in this test. Any test run with fragroute,Realsecure caught it,identified that somebody was fragmenting packets, and what hack attempt that was being utilized. In this respect Realsecure is way ahead of their competitors.

6 Conclusions from the tests

6.1 Dragon

Dragon has done well in previous tests [23] and should perform well in an enterprise sized network. The big issue is why buy this product when one can get much the same product from Snort for no cost? Dragon is in the same league as Snort and when buying a product it needs to be better than the open source solution, and Dragon is not better then Snort. Dragon also had some problems with diffrent NIDS evasion tools and Dragon has some problems when it comes to how "user friendly" it is.

6.2 Snort

Snort is really the winner in a lot of aspects, it's free,it runs well, and it can be install on nearly any platform or OS. The problem with Snort is that it lacks completeness, an administrator needs so many different components and fixes to make it do what they want it to do. Because of this, it will take a long time until one finds the solution that fits the needs required of it. This unfortunately is the curse of all open-source products. An administrator could probably with a lot of time and effort "concoct" a super product that will do nearly everything the administrator wants it to do, but nobody has the time and the patience to achieve this. What is needed is an "of the self" solution, and Snort is not that kind of product at the moment. Snort will probably have problems keeping up in the future, because all major vendors are creating big packet solutions with more than just NIDS capabilities. Snort doesn't have the backing to make this a reality and Snort still seems to have some issues with NIDS evasion techniques and Gigabit networks.

6.3 Realsecure

After all the testing a winner needs to be declared and the one that stands out is Realsecure. They have a complete product and seem to be one generation ahead of their competition. The issues with Realsecure is stability, they seem to have some software glitches and the administrative security console is very heavy and cant be run easily in the network. Over a longer period of time Realsecure might be a big burden on an administrator, but that can only be concluded at a later date. The other issue is they need to port the management console to other platform, only running under Windows seems to be a bit narrow. But Realsecure is very smart product and has all the tools an administrator hopefully is looking for in a NIDS. In testing it has been concluded that it's much smarter than the other products and supposedly should to be able to keep track on a large enterprise network. One issue is the point brought up in [17]. This paper raises the issues in that with all Realsecure's new intelligence comes new ways of evading detection. This could in the long run create a whole new set of tools to evade a Realsecure intrusion detection solution.

6.4 Final conclusion

The test has a winner- Realsecure- but on the question "Do any of these products have any place in an enterprise sized network?" the conclusion is probably not! What it depends on is how much time and effort a security conscious company is willing to invest. A NIDS is more a smart logging device then a intrusion detector. The administrator needs to determine, if what they want, is yet another log to keep track of. If what administrator craves is a device that makes educated guesses of trends in the network, and is able to warn an administrator that a virus is spreading, or a hacker is at work, then a NIDS product will not fulfil their desires . NIDS systems at the moment only create a lot of guess work for an administrator to deal with. A NIDS can also create problems because they might create a false sense of security. Administrators start to ignore other logs that they checked before, or fall behind on patching because they think the NIDS will catch any problems that can occur.

As stated a NIDS is a very high maintenance tool and a company could probably dedicate several man-hours watching IDS logs. One problem is that usually, between the time an alarm is generated to the time it takes a security expert actually to respond to the alarm, the attack is already over, or the virus has already spread. Then the IDS is reduced to a statistics tool for managers to justify their budgets to show how many attacks a week they receive, but- there is hope!

Companies like Realsecure and Cisco are creating smarter and smarter products. The more an IDS can communicate with other products like firewalls vulnerability scanners and other network traffic loggers the better they will become (in the case of vulnerability scanners Realsecure and Cisco are already there). Clean, ordinary NIDS systems will probably be totally gone in the near future. Only integrated solutions with IPS, NIDS, with self creating signature files will be left. With these kind system one can hopefully guard an enterprise sized network and stop any hacker or virus in their tracks before they can do any damage. Systems like this should hopefully be able to make intelligent decisions changing router and firewall configurations without the help of a monitoring administrator. But before this utopia becomes a reality, the IDS vendors need to win the trust of the security community, as it is now an administrator is very apprehensive to let a IDS make security related decision for them.

7 Future Trends

The future of NIDS looks bright, they will most likely stay on, but in new forms. A lot of company's are implementing NIDS into their network and there is a lot of research on the subject. The future of intrusion detection, seems to be systems that can create their own signatures and can take more active decisions how to handle intruders. The IDS community's biggest challenge is to get the false positives down, and to correctly identify what is going on. There is a lot of research in making NIDS that have larger amounts of session based control[19]. With these kinds of systems a NIDS may monitor a clients complete session with a server, and give a deeper analyses of what the user has done. These systems will be more effective and should help in catching 0-day exploits. Systems that creates their own signatures with the help of "honey pots" are already starting to pop-up on a research basis [15], and this could be another solution to catch 0-day exploit's.

References

[1] Stefan Axelsson. Aspects of the modeling and performance of intrusion detection. Technical report, 1999.

[2] Stefan Axelsson. The base rate falacy and its implications for the difficulty of intrusion detection. Technical report, 1999.

[3] Advanced Research Corporation. Security auditor's research assistant. *Sara.* http://www-arc.com/sara/.

[4] Datainspektionen. Personuppgifter i arbetslivet, 2001.

[5] Datainspektionen. E-post p jobbet. 2002.

[6] DOROTHY E. DENNING. An intrusion-detection model. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-13, NO. 2, FEB 222-232.*, 1987.

[7] Renaud Deraison. The "nessus" project. *Nessus.* http://www.nessus.org/.

[8] Fyodor. The art of port scanning. *Nmap*, 1997. http://www.insecure.org/nmap/.

[9] Coretez Giovanni. Fun with packets:. *Designing a Stick.* http://packetstormsecurity.nl/distributed/stick.htm.

[10] Robert Graham. Trons in realsecure 7. 2003.

[11] Van Hauser. The hacker's choice. *Hydra.* http://www.thc.org.

[12] Michael Holestein. How does fragroute evade nids detection. *Intrusion detection FAQ*, 2002.

[13] Marcus j Ranum. Experiences benchmarking intrusion detection systems. Technical report, NFR security, 2001.

[14] Wanderley J. Abreu Jr. Network intrusion detection system. *On Mass Parallel Processing Architecture*, 1999.

[15] Christian Krebich and Jon Crowcroft. Honeycomb creating intrusion detection. 1990.

[16] Robert Lemos. Msblast epidemic far larger than believed. *CNET News*, 2004. http://news.com.com/2100-7349-5184439.html.

[17] QinetiQ Ltd. Intruding without detecting. Technical report, QinetiQ Ltd, 2003.

[18] David Ludlow. Grouptest: Ids the top five ids systems reviewed and rated. 2002. http://www.vnunet.com/Products/Software/1130200.

[19] Masaki Minami Jun Murai Masayoshi Mizutani, Shin Shirahata. The design and implementation of session based nids, 2004.

[20] Internal Working Group on Data Protection in Telecommunication. working paper on intrusion detection systems, 2003. Adopted at the 34th meeting 2-3 September 2003 Berlin ,www.datenschutz-berlin.de.

[21] Raptor. Havoc. *Random ARP traffic generator*. http://www.0xdeadbeef.info/.

[22] Greg Saoutine. Barabarian at the gate. *Feature*, 2002.

[23] Greg Shipley and Patrick Mueller. Dragon claws its way to the top. 2001. http://www.networkcomputing.com/1217/1217f2.html.

[24] Dug song. Fragroute. http://monkey.org/ dugsong/fragroute/.

[25] Spender. Datapool v3.3. http://www.packetstormsecurity.org/DoS/indexsize.html.

[26] Betsy Yocom and Kevin Brown. Review: Intrusion-detection products grow up. 2001. http://www.nwfusion.com/reviews/2001/1008rev.html.

[27] NIST: Peter Mell Vincent Hu MIT Richard Lippmann Josh Haines Marc Zissman. An overview of issues in testing intrusion detection systems. Technical report, NIST and MIT, 2002.

A Snort

Test	**Flags**	**ALARM**	**Identifyied**
Nmap	-sS	YES	YES
Nmap	-sF	YES	YES
Nmap	-sX	YES	YES
Nmap	-sN	YES	YES
Nmap	-sU	YES	YES
Nmap	-f -sS	YES a lot more	YES lost two
Nmap	-D -sf	YES	YES
hydra	-C logpass ftpserver.se ftp	NO	–
hydra	-C logpass ftpserver.se ftp -t 20	NO	–
hydra	-l admin -P pass ftpserver.se ftp -t 20	YES	YES
havoc	-i eth0	NO	–
datapool.sh	-d www.webserver.se	YES	YES
winnuke	host port	YES	YES
stick		YES	NO
stick + winnuke		YES	10 min delay
fragroute + hydra		YES	NO
tesoiis	host port url	YES	YES
fragroute + teoiis		YES	NO
fragroute + winnuke		YES	NO
stick + hydra		NO	NO
sara		YES	NO
nessus		YES	YES

Table A.1: Testing specifications for Snort

04.14.2004 Microsoft IIS SSL Remote Denial of Service Exploit (MS04-011): NO ALARM

02.14.2004 Microsoft Windows ASN.1 LSASS.EXE Remote Exploit (MS04-007) :NO ALARM

B Dragon

Test	**Flags**	**ALARM**	**Identifyied**
Nmap	-sS	YES	YES
Nmap	-sF	YES	YES
Nmap	-sX	YES	YES
Nmap	-sN	YES	YES
Nmap	-sU	YES	YES
Nmap	-f -sS	YES frag alarms	YES lost one
Nmap	-D -sf	YES	YES
hydra	-C logpass ftpserver.se ftp	YES	YES
hydra	-C logpass ftpserver.se ftp -t 20	YES	YES
hydra	-l admin -P pass ftpserver.se ftp -t 20	YES	YES
havoc	-i eth0	NO	–
datapool.sh	-d www.webserver.se	YES	YES
winnuke	host port	YES	YES
stick		YES	NO
stick + winnuke		YES	YES
fragroute + hydra		YES	NO
tesoiis	host port url	YES	YES
fragroute + teoiis		YES	NO
fragroute + winnuke		YES	NO
stick + hydra		YES	YES
sara		YES	YES
nessus		YES	YES

Table B.1: Testing specifications for Dragon

04.14.2004 Microsoft IIS SSL Remote Denial of Service Exploit (MS04-011) NO ALARM
02.14.2004 Microsoft Windows ASN.1 LSASS.EXE Remote Exploit (MS04-007) :ALARM AND IDENTIFYED

C Realsecure

Test	**Flags**	**ALARM**	**Identifyied**
Nmap	-sS	YES	YES
Nmap	-sF	YES	YES
Nmap	-sX	YES	YES
Nmap	-sN	YES	YES
Nmap	-sU	YES	YES
Nmap	-f -sS	YES	YES
Nmap	-D -sf	YES	YES
hydra	-C logpass ftpserver.se ftp	YES	YES
hydra	-C logpass ftpserver.se ftp -t 20	YES	YES
hydra	-l admin -P pass ftpserver.se ftp -t 20	YES	YES
havoc	-i eth0	YES	NO froze 25 min
datapool.sh	-d www.webserver.se	YES	YES
winnuke	host port	YES	YES
stick		YES	NO
stick + winnuke		YES	YES
fragroute + hydra		YES	YES
tesoiis	host port url	YES	YES
fragroute + teoiis		YES	YES
fragroute + winnuke		YES	YES
stick + hydra		YES	YES
sara		YES	YES as SATAN
nessus		YES	YES

Table C.1: Testing specifications for Realsecure

04.14.2004 Microsoft IIS SSL Remote Denial of Service Exploit (MS04-011) NO ALARM 02.14.2004 Microsoft Windows ASN.1 LSASS.EXE Remote Exploit (MS04-007) :NO ALARM 03.2004 RealSecure / Blackice iss_pam1.dll Remote Overflow Exploit:NO ALARM and problems with connections with sensor

D Dictionary

NIDS: Network Intrusion Detection System

HIDS: Host Intrusion Detection System

Vulnerability-scanner: A tool that assesses a server or workstations level of intrusion defence depending on its response to different probes.

Tivoli: is a network administration tool to survey large networks state of "well being".

Ddos: Distributed denial of service attack.

ACID: Analysis Console for Intrusion Databases.

ARP-poisoning: A process where a hacker tries to listen to another computers traffic on switched LAN segment. This is done by manipulating the ARP communications with the switch, tricking it to send traffic to the hackers computer.

Honey Pot: is a system that is designed to simulate several systems. The "honey pot" creates a tempting target for a hacker to attack. The "honey pot" gives the administrator a chance of easily deflecting and identifying a malicious user from critical systems.

Rpm: is a command line driven package management system for Linux. This is used for easy installations of programs in Linux.

IPS: Intrusion Prevention System; is an IDS system that takes action against found intruders for example by shutting down their connections.

0-day exploit is a unknown and undocumented vulnerability in a system. An Administrator has zero days warning of the vulnerability.