Global Information Assurance Certification Paper

Implementing Network Based Intrusion Detection in a Multi-School Campus

By Walter Brock

SANS GSEC Practical Assignment
Version 1.4c Option 2

6 September 2004

Introduction:

With the ever increasing security threats (Sheriff, Ayers, Aug. 2003) to computer networks of all types, Intrusion Detection Systems have left the exclusive domain of the Chief Security Officer and are now being implemented by network administrators of all varying skill levels. In the world of education and nonprofit organization(or free) IDS's are not simply one alternative, they are often the only alternative.

After presenting the reader with pertinant background information, this paper attempts to present the experiences and lessons learned by the author during the evaluation, testing, and final implementation of a Network Intrusion Detection System (NIDS). This effort took place within a large multi-school campus in central Florida, where the author was emplyed as a network manager, over an approximately 19 month period starting in late 2002 and continuing through June 2004.

Background:

The school district where the author was employed covers an entire county (as do all districts in Florida). All internet access is routed through a single gateway at the district headquarters. A firewall, proxy servers, email scanners, and url content filtering are all performed at this gateway. However, once inside the firewall, no attempt is made to restrict traffic flowing between campuses. Additionally, there is no single individual at the district level tasked with ensuring the security of the network, thus network managers at each campus must contend not only with threats that might make it through the firewall, but also with threats from infected computers and “hackers” eminating from within their own campus and from other campuses throughout the district. Since schools are required by law to protect the privacy and confidentiality of student information stored on their servers, the need for intrusion detection becomes evident.

The environment for this project is unique in that the campus where the author worked actually hosts multiple schools, each fulfilling a different mission, and each requiring different network and computing resources. The campus itself consists of more than a dozen buildings (with more being added) and employs a switched infrastructure based on 100 Mb connections to each desktop. Switches are connected by a fiber backbone. Outlying buildings are connected via an 802.11b wireless employing high gain external antennas.

Though Host IDS (HIDS) is also a necessary part of this protection, HIDS was examined as part of a separate project and will not be covered here.

Methodology:

- ◇ Each Network IDS was installed on a “clean”, “stock” RedHat 9 system running on a Gateway e2000 computer featuring a 2.4.GHz processor and 256 MB of RAM.
- ◇ Each Network IDS was allowed to run for a length of time necessary to confirm the configuration and collect accurate data. This varied from 1 to 7 days. Results were than gathered from system logs and NIDS databases.

Theoretical Background:

Sheriff and Ayers (OCE 2003) define an intrusion threat as “the potential possibility of a deliberate unauthorized attempt to access information, manipulate information, or render a system unreliable or unusable.” With this definition in mind we can then define an intrusion detection system (IDS) as a program or hardware/software combination whose primary purpose is to detect deliberate unauthorized attempts to access information, manipulate information, or render a system unreliable or unusable.

Intrusion detection systems come in two basic forms. There are host based intrusion detection systems (Host Based IDS) and network intrusion detection systems (NIDS). Each of these types has their own advantages and disadvantages and the decision to deploy host based IDS vs. NIDS is not mutually exclusive. The two complement each other and are often deployed together to give more complete coverage of a given system. While the major focus of this paper is on NIDS, host based IDS will also be discussed.

Whether host based or network based, intrusion detection is based on two fundamental paradigms (Sheriff & Ayers OCE 2003, Rozenblum 2001, Deri 2003, Rash 2002).

The first are signatures based systems. These systems work by comparing network traffic or host based files to known signatures. In the case of NIDS, a set of rules will define known network attacks. Any network data which matches these rules will cause the NIDS to issue an alert. Snort is an example of this form of NIDS. In the case of host based IDS this is done by comparing digital signatures or checksums from these files with signatures obtained at install time and stored in a protected database. Tripwire is an example of this type of IDS. Signature based IDS systems are sometimes referred to as misuse detectors, though this phrase has fallen out of use.

The second paradigm is anomaly detection. In anomaly detection, the host or network is monitored for behaviors which would be symptomatic of misuse or an intrusion. SNARE is a host based IDS which works by monitoring all activity occurring on the system, such as system calls and accesses to memory looking for signs of malicious or unauthorized behavior. Most NIDS implement anomaly detection through some form of statistical analysis. At this time the author knows

of no open source NIDS which implement this model. However, NTOP is presented as an example of an open source solution for monitoring network utilization and does provide some insight into traffic behavior on the network.

Capturing network traffic:

In order to implement an effective IDS solution it is important to understand certain fundamental principles involved in reading data (capturing traffic) from the network. (Taylor 2003, Deri 2002, Baur 2002).

Under normal circumstances, the packets of data on a network segment are only read by the intended receiver. All other nodes on the network will normally ignore data packets not addressed to them. In order to capture this traffic, the NIDS will place its network interface card (NIC) in "promiscuous" mode. In this mode, the NIDS will read all traffic which flows on the network link to which it is attached regardless of the intended destination. Note that this does not affect the actual flow of traffic and in no way prevents or delays the data from reaching the addressee.

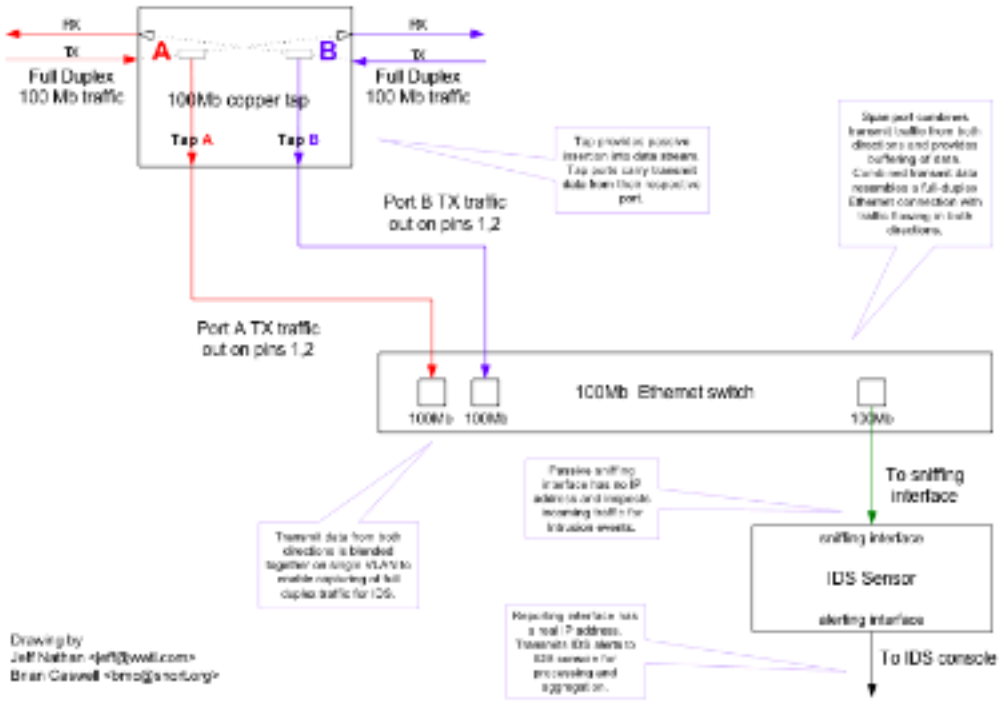
If our network uses a passive hub, all we need to do is simply plug the NIDS network cable into the hub and we can read all the data flowing on that hub. Hubs use a bus technology which shares the network bandwidth with all nodes attached to the hub. This is similar in concept to an old fashioned telephone party line. However, if ethernet switches are employed this method will not work. In an ethernet switch, each data packet is sent only to the output port where the intended receiving node is located. Ethernet switches use built in intelligence in the form a microprocessor and detected software to detect and store the media access control addresses (the "hardware addresses) of each node attached to the switch. Data is routed and internal dedicated connections connect only to the appropriate node. This is similar to modern private exchange phone systems. A NIDS on another port will never see this data and so cannot monitor it and detect an intrusion.

One solution to the problem of switched networks is the span port (sometimes referred to as a spanning port or mirror port). If the switch supports this option, than one port can be configured to span all of the the other ports. That is to say, the data from all of the other ports will be mirrored to the span port. In this way a NIDS listening on the span port can receive all of the data for the network in similar fashion to a hub and thus detect intrusions. Span ports work well for small networks employing only a few switches, but become cumbersome to implement on larger networks multiple switched segments.

Another solution to the problem of switches is the ethernet tap. An ethernet tap is a device that sits in line with the data connection between the ethernet switch and the network router (or another switch). The tap allows the NIDS to capture

the data flowing between the switch and the other device. The one disadvantage to the ethernet tap is that any data flowing between internal nodes on the switch will not be seen, thus any attempts at intrusion emanating from within the network will be missed using this configuration. It is not uncommon on larger networks to deploy multiple NIDS in order to catch both internal and external intrusion attempts. Often the data from span ports on multiple switches will be combined together using a hub or a multi port tap. Figure 1 illustrates the use of an ethernet top to route network data from one network switch segment to another switch. The switch is then configured with a span port to allow a single NIDS to monitor data from multiple switches and segments.

Figure 1: Ethernet tap and switch with span port.



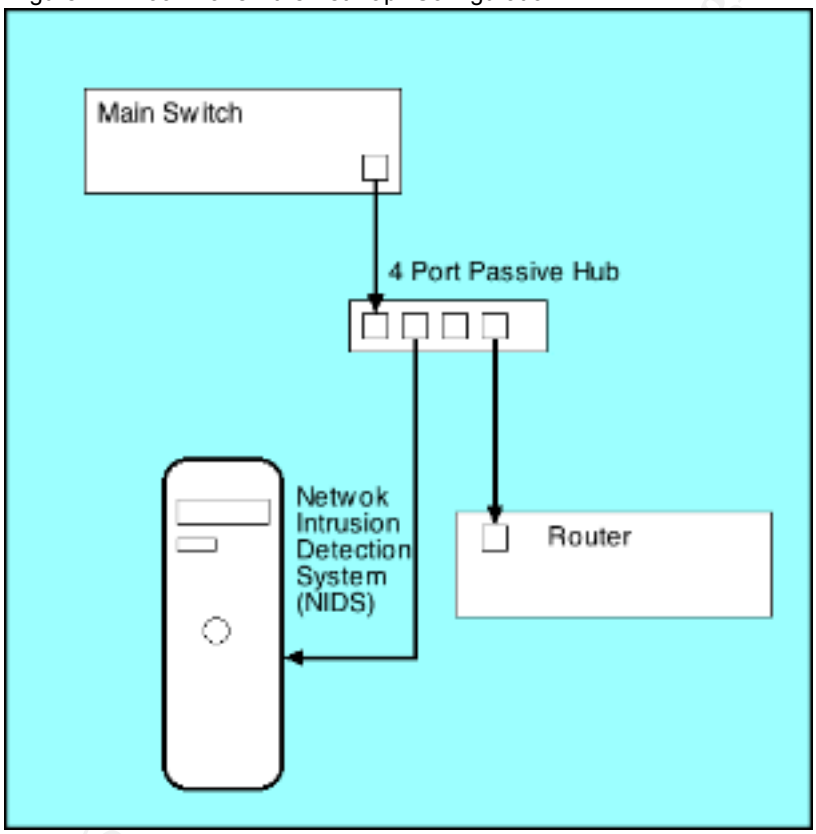
Source: Jeff Nathan (jeff@wwti.com) and Brian Caswell (bmc@snort.org)

Ethernet taps can range in cost anywhere from a few hundred to several thousand dollars making them prohibitively expensive for budget challenged organizations. An alternative is the “poor man’s ethernet tap”. This consists of placing a passive hub in line between the switch and router. Since all data between the switch and router must now flow through the hub, the data can be easily monitored by the NIDS. However, this is not an optimal solution. First, almost all switches and routers support full duplex connections where as passive hubs support only half duplex connections. This will force the switch and router to renegotiate a half duplex connection. This can slow communications and lead to increases in latency. Second, an ethernet tap

provides a straight through connection between the hub and router. Should the ethernet tap loose power or should its internal electronic components fail, the physical connection between the switch and router will not be broken.

In the case of this test, the router supports a connection to a 1.54 Mbps T1 line. This being the case, it was felt a 10 Mbps hub between the switch and router would not noticeably affect network traffic and, indeed, this proved to be the case. Figure 2 illustrates the configuration used for this test.

Figure 2: "Poor Mans Ethernet Tap" Configuration.



Source: The Author

One last method of intercepting network traffic is called ARP spoofing. ARP, or Address Resolution Protocol is the method by which network devices identify the MAC address of other devices on the network. It is ARP that match internet IP addresses with MAC addresses. In this monitoring method, the NIDS sends an ARP message to each network node to be monitored. This ARP message falsely informs the monitored nodes that the NIDS is actually the network router. The effected nodes will now forward any outgoing traffic to the NIDS rather than to the router. The NIDS then reads the data and forwards it to the real router.

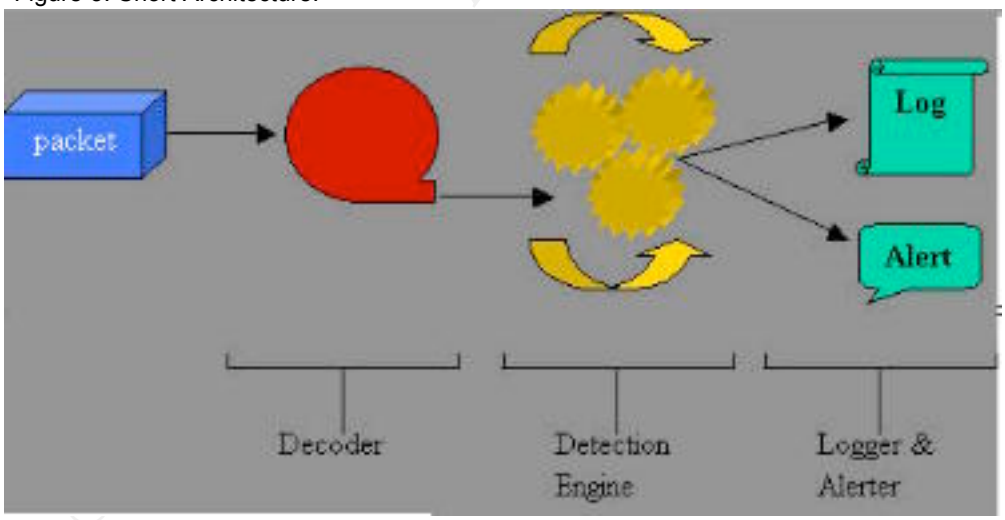
ARP spoofing has multiple disadvantages. First, it dramatically increases the load on the NIDS and can increase network latency. Second, should the NIDS

malfunction in any way, network traffic will be temporarily interrupted. Finally, many modern switches are configured to check the MAC addresses of all incoming packets against their internal routing table and reject those that don't match its own ARP address table. This would effectively block data flow to the NIDS since the switch would recognize that the MAC address of the NIDS does not match up with the IP address of the actual router. While this method can be employed successfully, under certain limited circumstances it is not recommended.

NIDS:

The first three NIDS to be examined here are all based on a program called Snort (Westphal 2000, McClure 2003, Gaur 2001) Snort, which is available at www.snort.org, refers to itself as a "light weight NIDS". This moniker is misleading. In fact Snort is an extremely powerful program. Snort consists of three basic components. First is the decoder, which as its name implies, decodes the captured packets. Second is the detection engine which compares the decoded packets to the defined rules. Last is the alert & logging component which records packets flagged by the detection engine to the defined logging facilities and generates whatever alerts may have been configured by the administrator.

Figure 3: Snort Architecture.



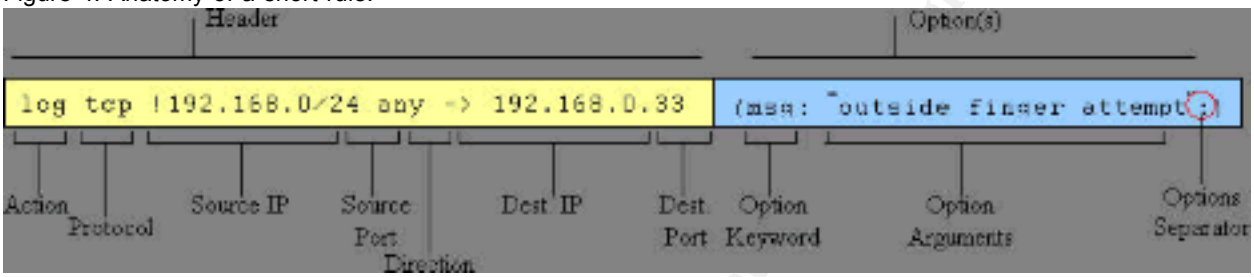
Source: http://www.linuxjournal.com/article.php?sid=4668

Snort Rules:

As mentioned above, Snort works by comparing captured packets to a predefined set of rules. Snort comes with a set of several thousand default rules in just over two dozen categories. Each category can be included or excluded as part of the basic configuration process. For example, a network running a

Microsoft SQL Server installation would not need to include the Oracle rules. However, even after excluding unnecessary categories, the Snort administrator will still need to write new rules. This might be to tell Snort to ignore certain types of traffic on the local network, such as “ignore port scans from the managers workstation”. Another possibility would be to include new attack signatures not included in the default set , though updates to the default rules are made on a regular basis. Figure 4 shows the syntax for a basic Snort rule.

Figure 4: Anatomy of a snort rule.



Source: http://www.linuxjournal.com/article.php?sid=4668

EVALUATION PHASE

PureSecure™

Introduction:

The first of the Snort based systems we will examine is PureSecure. PureSecure actually has its roots as an open source project. In fact, older versions are still available from at least one Linux distribution. PureSecure is a commercial product of Demarc and is available on the Internet at www.demarc.com. Though no longer entirely open source, it is based on a host of open source components including the Snort intrusion detection engine, Apache web server and MySQL database. Demarc adds its own proprietary front end to these components as well as a customized shell based installer. (Slonaker 2003).

Capabilities:

PureSecure™ actually provides three main functions. It’s strength lies in its Snort based network IDS (NIDS). It also can be configured to do basic file integrity checks, similar to, but not as sophisticated as Tripwire. Lastly, it provides basic network service monitoring. The sensor can be configured to measure latency between itself and other hosts through ping checks, and basic service monitoring through TCP connections to predefined ports on those hosts. I can also track certain fundamental metrics such as CPU utilization and disk space on machines hosting a PureSecure™ sensor.

Installation:

The first time the author installed PureSecure, just over a year ago, was a terrible experience. The machine on which it was installed was a production server which had been in use for over a year. The PureSecure install script had a terrible time with conflicts between the software it was trying to install and software already on the server. In the end, the automated install was abandoned and completed manually. Configuration data had to be manually copied and pasted to multiple configuration files. User permission had to be defined on dozens of files and directories. Finally, the MySQL database was manually created. Once it was finally running, the browser based interface was impressive, but performance was laboriously slow. The server on which it was installed was simply underpowered for the application.

The install of PureSecure for this evaluation was an altogether different experience. The install was performed on a pristine RedHat Linux install that had never been in production or even on the network and had never had any of the conflicting software on it.

The process began by registering for and downloading the installer script. While PureSecure™ is a commercial product it is offered free for personal and noncommercial use[1] . The installer script is a 300 KB shell script. It asks a few questions, than downloads the source code for the Snort IDS, Apache web server, MySQL database server, as well as several library files and perl modules. Compiling from source takes longer but also insures optimum performance for a given installation and helps to dramatically reduce problems with library conflicts and dependencies.

After a little over two hours, the installer had finished compiling and installing all of the open source components and was ready to download and install the PureSecure daemon itself. This daemon and its associated web files is the heart of PureSecure. The daemon controls the snort process, database logging, and handles the user interface via the web server.

The entire install process was nearly flawless and the IDS went online without a hitch.

User Interface:

As mentioned previously, PureSecure™ is accessed through a browser based

[1] The download page for PureSecure Personal Edition states that the product is only free for “home” use. However, the license file included with the software states clearly, that PureSecure is free for noncommercial use and specifically sites non profit organizations as an example of such use.

interface. The web server is configured to use SSL encryption, but oddly enough, there is an option at install time to allow anonymous log ins to the web console. I declined that option and set up a password protected log in. Administrative functions are password protected in either case, but I cannot imagine any network administrator allowing anonymous log ins to his or her IDS!

The PureSecure™ interface is well thought out. New users will want to spend some time with the equally well written documentation before configuring some of the more advanced functions.

Figure 5: PureSecure Summary Page.

Source: http://www.demarc.com

Figure 6: PureSecure Configuration Page.

Source: http://www.demarc.com

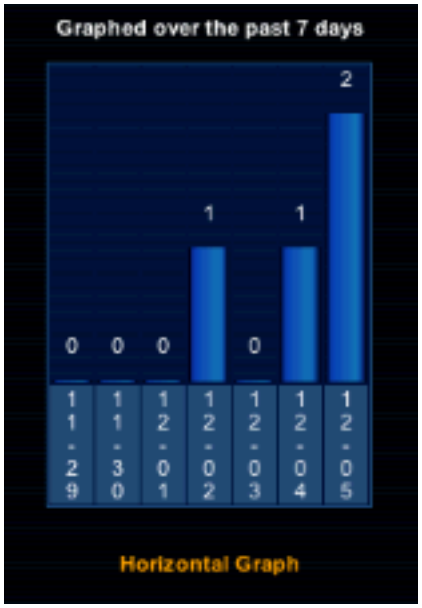
Results:

The results from PureSecure were quite impressive. Over a seven day period, PureSecure logged over 150,000 events. Of these, 345 were considered to be unique. The overwhelming majority of these events were actually normal network traffic. Fortunately, the ability to show unique event types (see Figure 3) made the job of sorting normal traffic from genuine security concerns dramatically easier. Once the normal traffic types are identified, new rules can be inserted into the Snort configuration to instruct the NIDS to ignore those types. For example, in an earlier test, PureSecure logged over 345,000 events in a one week period. The creation of a "local" rule set instructing Snort to ignore certain type of normal traffic resulted in the results seen here. Further "tweaking" of the rules could easily reduce the number of false positives by another order of magnitude.

Figure 7: PureSecure Unique Events Page.

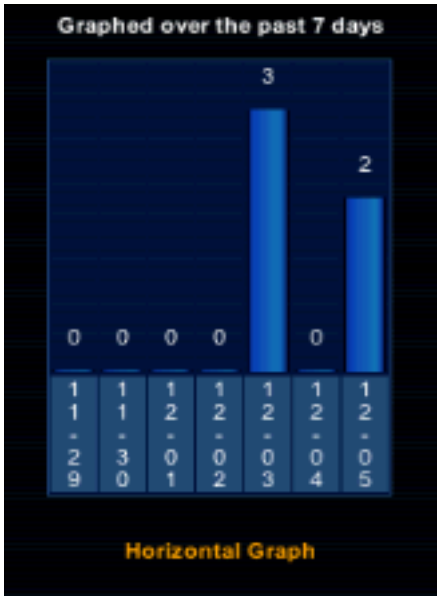
Source: The Author

Figure 8: PureSecure Graph:
Port Scans Detected Over 7 Day Period, Host 1.



Source: The Author

Figure 9: PureSecure Graph
Port Scans Detected Over 7 day Period, Host 2.



Source: The Author

MIDAS v2.0

Introduction:

MIDAS is an entirely open source project and is intended to be an entirely free alternative to PureSecure. It is built on exactly the same open source foundation as PureSecure (Snort, Apache, MySQL & assorted components) and is available at http://midas-nms.sourceforge.net. Version 2.0i was tested. (Metcalf 2002).

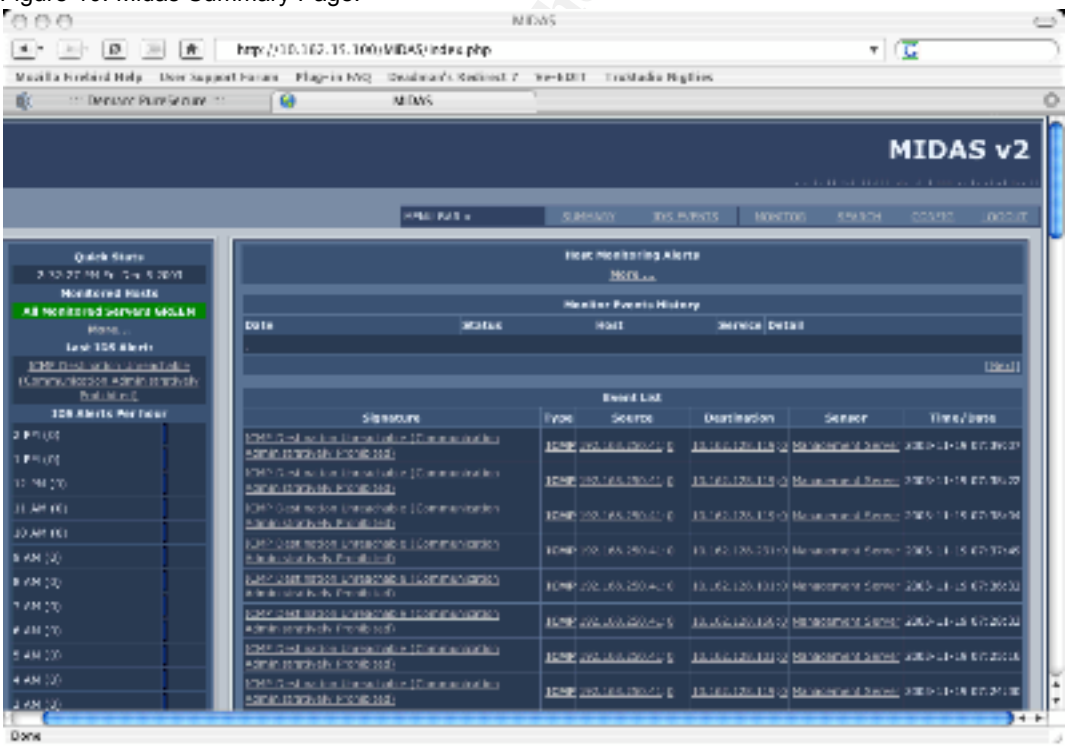
Capabilities:

MIDAS attempts to equal the functionality of PureSecure by adding its own integrity and service checking capabilities to the Snort NIDS. However, MIDAS lacks many of PureSecure's more advanced capabilities, such as managing numerous sensors from a single console.

User interface:

MIDAS presents the user with a web based interface strikingly similar to PureSecure (see Figure 10). Interestingly, a check of the MIDAS web site shows that an entirely new interface is under development though no time frame is given for its release.

Figure 10: Midas Summary Page.



Source: The Author

Installation:

Despite the similarities with PureSecure, the MIDAS installation was anything but easy. After downloading and compiling the source code the next step was to configure the daemons. MIDAS uses not one, but six separate daemon processes. Each has its own XML formatted configuration file that must be edited manually. Unfortunately, the daemons give very little feed back by way of error codes (even with the debug level set high) which makes it very difficult to figure out what is wrong when they refuse to launch. This is further compounded by the fact that a minimum install requires no less than three of the daemons to function correctly together. A great deal of trial and error is necessary.

Once the three daemons (one to control Snort, one to handle database logging, and one to handle control functions) are working, the Snort sensor must be configured via the web interface.

This process involves creating at least one "sensor" in the database and then copying the default Snort rule set and config file to this sensor. Once that is done, the rules can be customized to fit the local network usage. Unfortunately, MIDAS falls down badly here. Where PureSecure did this process automatically at install, MIDAS requires the user to follow multiple steps just to get to this point. Worse, PureSecure automatically verifies Snort config syntax. MIDAS has no such facility. The default Snort rules caused Snort to fail with no error warnings at all in the interface.

With no way to verify the Snort rules, the author was forced to write a shell script with which to verify them manually. The process consisted of making a change in the web interface, letting MIDAS generate the Snort config file, running the shell script, finding the error, editing the offending rule in the web interface, generate, find error, etc. until a working configuration was finally generated. This process took the better part of a day. Another problem was that in many cases the cause of the error simply could not be identified. In these cases the syntax in the web based interface appeared correct, but the generated rule would cause Snort to crash. The only "fix" was to delete or comment out the offending rules leaving potential holes in the security coverage. Not an optimum solution. Clearly, there is a problem in the daemon process that translates the rule files from binary large objects (BLOBS) in the database to ascii text files for Snort.

Results:

It was observed that almost immediately after launching MIDAS there was a significant reduction in the number of events recorded as compared to PureSecure. This was puzzling since both programs used the same version of Snort with essentially the same rule set. To confirm this, port scans were launched against the MIDAS server from two different work stations. The MIDAS server failed to register either port scan. This was very disturbing. A check of the Snort config generated by MIDAS verified that it was configured to load its port

scan detector.

After 24 hours of monitoring, MIDAS had only captured only two types of unique events (see Figure 7) out of a total of 149 events total. Oddly, one of those events was a scan on port 135 from a machine outside the local network. Why only this one scan was detected out of a total of 5 scans registered that day on another server remains a mystery.

Figure 11: Midas Unique Events Page.

Top 10 Unique Events

Freq	Signature	Sensor	First Event	Last Event
148	ICMP Destination Unreachable (Communication Administratively Prohibited)	Management Server	2003-11-19 07:39:07	2003-11-18 11:09:29
1	[spp_stream4] STEALTH ACTIVITY (NULL scan) detection	Management Server	2003-11-18 15:08:18	2003-11-18 15:08:18

Source: The Author

As a final confirmation that the problems encountered lie in the MIDAS implementation alone. Snort was launched manually using the configuration generated by PureSecure, but configured to log to the MIDAS database. A quick check of the MIDAS front end showed that Snort was now logging the expected number and type of events.

Given the extreme difficulties in installing and configuring MIDAS, the glaring holes in its IDS capabilities, and the very poor results, the test was ended after only one day of monitoring.

MIDAS v2.2

Midas v2.2 was examined several months after the initial evaluations as part of a reimplementation of the original project. The results of this new evaluation are presented later in this paper.

ACID

Introduction:

ACID (Analysis Console for Intrusion Detection) is the most mature of the three Snort based solutions examined here and is available at http://acidlab.sourceforge.net. To use ACID, Snort must be configured to log its output to a MySQL database. It's browser based interface allows the user to view Snorts output in several different ways such as individual events, unique events, or graphically. (Metcalf 2002).

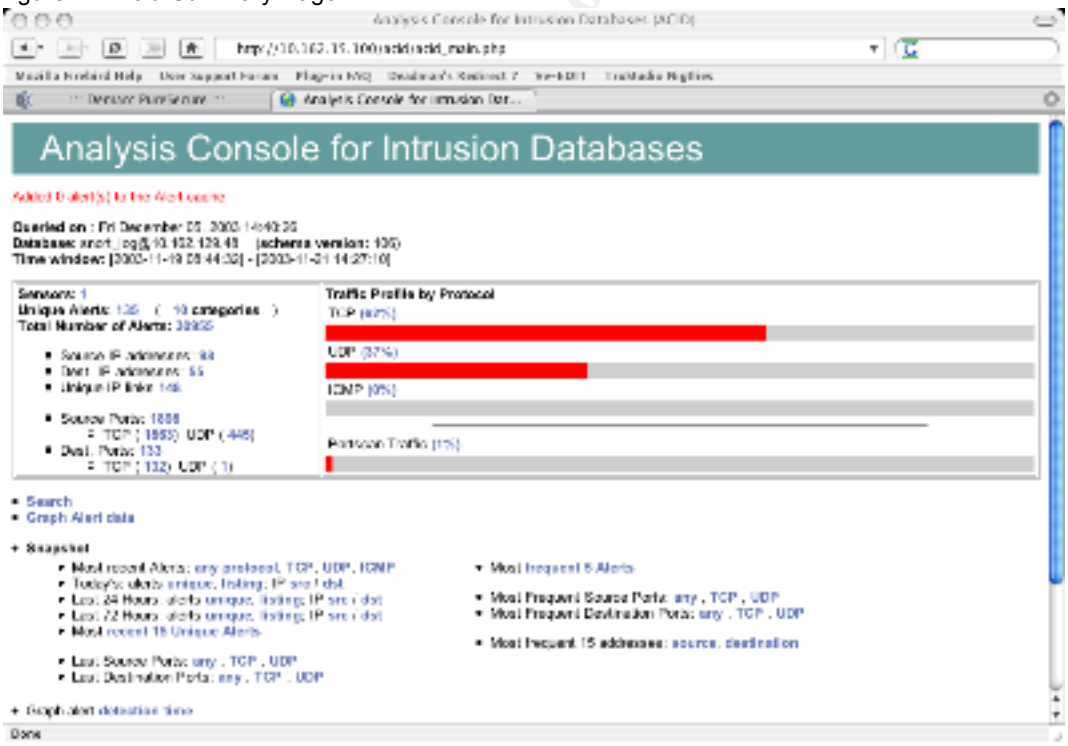
Capabilities:

ACID takes a very different approach from either PureSecure or MIDAS. First, ACID does not attempt to add any other functions to Snort such as integrity checking or service monitoring. Second, ACID does not attempt to automate the administration of Snort in anyway. The user must manually configure Snorts config file and rule sets and launch Snort either from the command line or from a startup script. ACID acts simply as a front end to Snort allowing the user to view Snorts output through a web browser.

User Interface:

ACID presents the user with a straight forward summary web page (see Figure 12). Details and graphs are available through links on this page. The web interface does not provide any sort of configuration screens either for ACID itself or for Snort. All configurations are done manually.

Figure 12: Acid Summary Page.

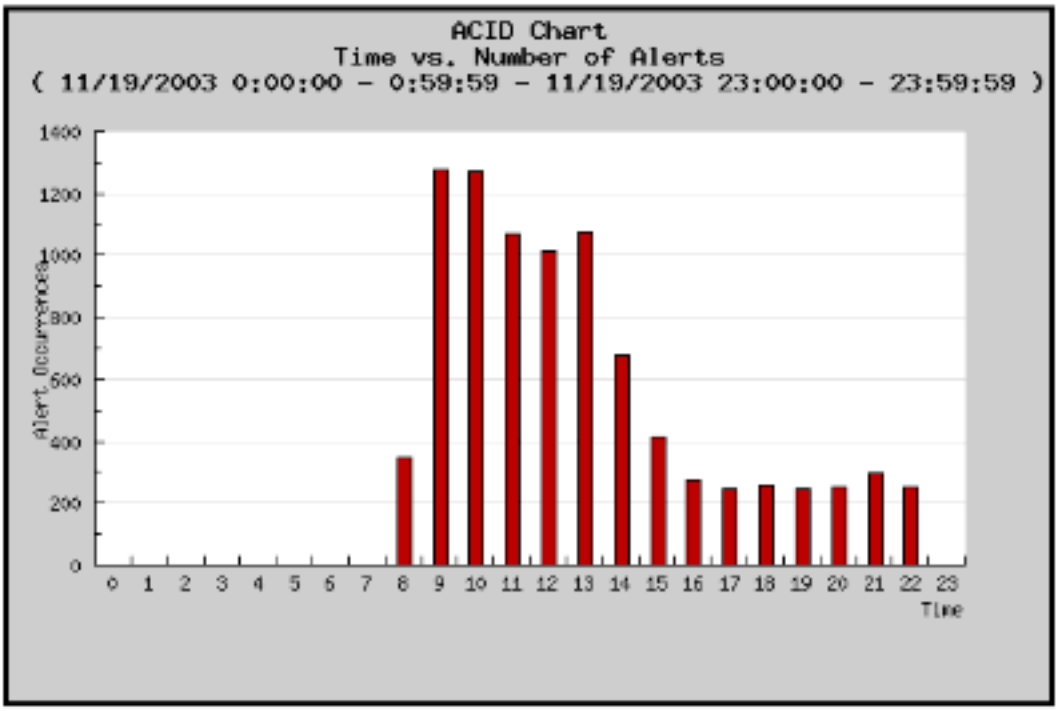


Source: The Author

ACID also provides excellent graphic reporting abilities (see Figure 13), including the ability to graph types of events and trends over any specified period

of time. Unfortunately, these reports must be configured from a series of drop down menus each and every time they are run. For a smaller network this is merely tedious. For a larger network the amount of time required to generate reports could become an issue.

Figure 13: ACID Graph Showing Number of Alerts in a 24 Hour Period.



Source: The Author

Installation:

Installation consists of downloading a single compressed tar file from the ACID web site. This tar file is then installed on to the web server and the configuration files editing to point to the database server and base URL. A MySQL database must then be created using the included SQL script.

In addition, ACID requires several external components including the adodb database abstraction library and the jgraph graphing libraries. Upon launching the console for the first time a php script checks for the presence of these libraries and directs the user to the appropriate download sites if they are not found.

In this evaluation it took less than 30 minutes to install and configure ACID. This included the time needed to download and install the other required components.

As noted above, ACID makes no attempt what so ever to provide any sort of automated configuration for Snort. It is up to the user to install and correctly configure Snort. Thus a greater degree of knowledge is require on the part of the network administrator. For this test the Snort configuration generated by PureSecure was used. The configuration file was merely edited to point to the ACID database on the MySQL server rather than the Snort database.

Results:

ACID logged a total of 30955 alerts in a 48 hour period. Of those, 135 were considered to be unique events. Again, as in PureSecure, the ability to view the unique events on a separate report made eliminating false positives very easy to do. The ease of installation, combined with the intuitive web based interface make ACID an excellent choice for smaller networks, providing the network administrator has the knowledge and expertise required to install and configure Snort manually.

NTOP[2]

Introduction:

NTOP (available at http://www.ntop.org) is a very mature program for monitoring network using. NTOP was originally written by Luca Deri with the first version appearing in 1993. It is written in "C" and available for multiple host operating systems including most flavors of Unix/Linux and windows 98/ME/NT/2K. NTOP is freely available under GNU General Public License, but commercial support and customization are provided at http://support.ntop.com.

NTOP is entirely different from the Snort based NIDS examined earlier. NTOPs primary function is to provide detailed statistical data on network utilization. These statistics include the protocol and connections made be each monitored node on the network. By examining these statistics, the network administrator can easily spot unauthorized or suspicious network activity. (Deri 2000, 2002, 2003).

Capabilities:

NTOP uses the libpcap packet capture library to capture, or "sniff" network traffic in promiscuous mode. The program gathers detailed statistics on protocol usage, connection state, throughput, etc. on every host it identifies. The

[2] The reader is advised that the author has contributed some programming code to the NTOP project and has been an ardent proponent of its use for many years.

statistical data are stored in an RRD database[3].

Earlier versions of NTOP had a “rules engine” with which the administrator could configure NTOP to examine network data in much the same manner as does Snort, though the engine in NTOP was much less sophisticated. This ‘rules engine’ was removed in 2002 because the developers felt that it required too much processing power and it detracted from NTOPs overall stability. The development team also felt that their efforts would be better spent improving NTOP’s monitoring capabilities. The rule engine was replaced by a packet logging function. NTOP can be configured to log all “suspicious” packets to a file in tcpdump format. In NTOPs context, the word suspicious merely means that the packet didn’t fit NTOPs definition of normal network traffic. This could be anything from unknown protocols to odd port numbers. This log file can then be examined by a more powerful protocol analyzer such as ethereal. Unfortunately, my attempts to read this file failed. Ethereal would stop loading after only two packets with an error message stating that the packets were malformed. Clearly there was a problem with my implementation as most NTOP users have reported no problems with this facility. Fortunately, the packet logging function is not needed to the uses presented here.

Installation:

NTOP can be compiled from source or, in most cases, installed as a precompiled binary package. For this test the most recent RPM packages for NTOP (version 2.2.93) and the most recent version of rrdtool were downloaded from the NTOP web site. Download and installation required less than 15 minutes. Once downloaded the NTOP configuration file must be edited to match the user’s local needs. The configuration file format and options are clearly documented in both the main page and the online documentation. Once the configuration file is complete, NTOP must be run one time from a console terminal in order to set the administrative password. The entire configuration process required no more than 5 minutes. After this was completed, NTOP was launched in daemon mode and allowed to begin monitoring the network.

User Interface:

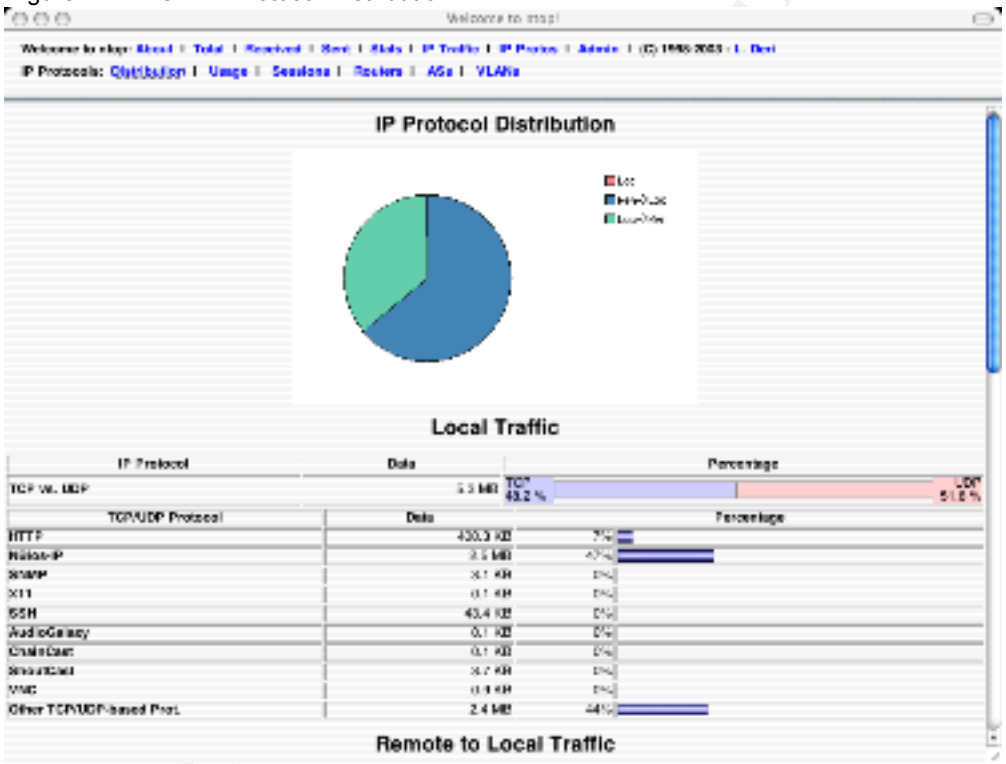
NTOP delivers data to the user through its own built in web server, available by default on port 3000 with an ssl version available on port 3001. These settings are configurable. The NTOP user interface provides a vast array of tables and graphs allowing the user to see everything from overall network utilization and throughput to the hour by hour use of a given protocol by a given host including all attempted and completed connections.

[3] For more information on RRD or Round Robin Database please see http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool/

The authors of NTOP have taken considerable effort to provide an intuitive interface to this vast wealth of data. The NTOP screens are well laid out and very easy to navigate. In nearly all cases, data are provided in both tabular and graphical form, allowing the user to find specific data points or spot trends very quickly.

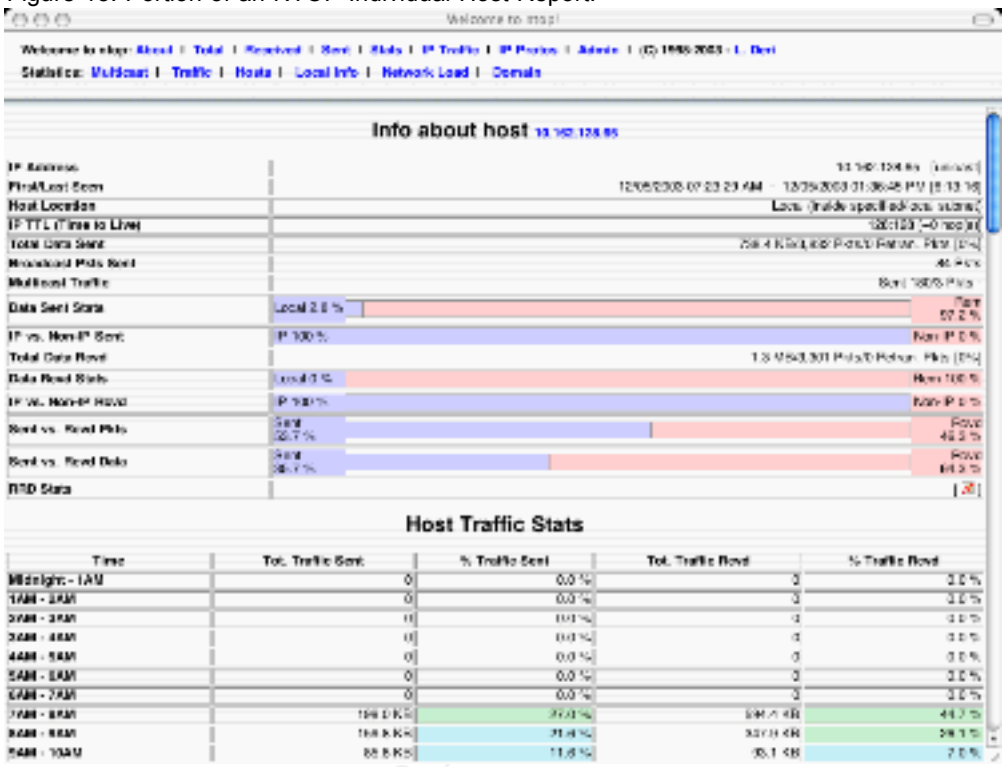
Several NTOP screen shots are included here to illustrate its capabilities. However, space simply does not allow for the complete review of all of the possibilities provided by NTOP:

Figure 14: NTOP IP Protocol Distribution.



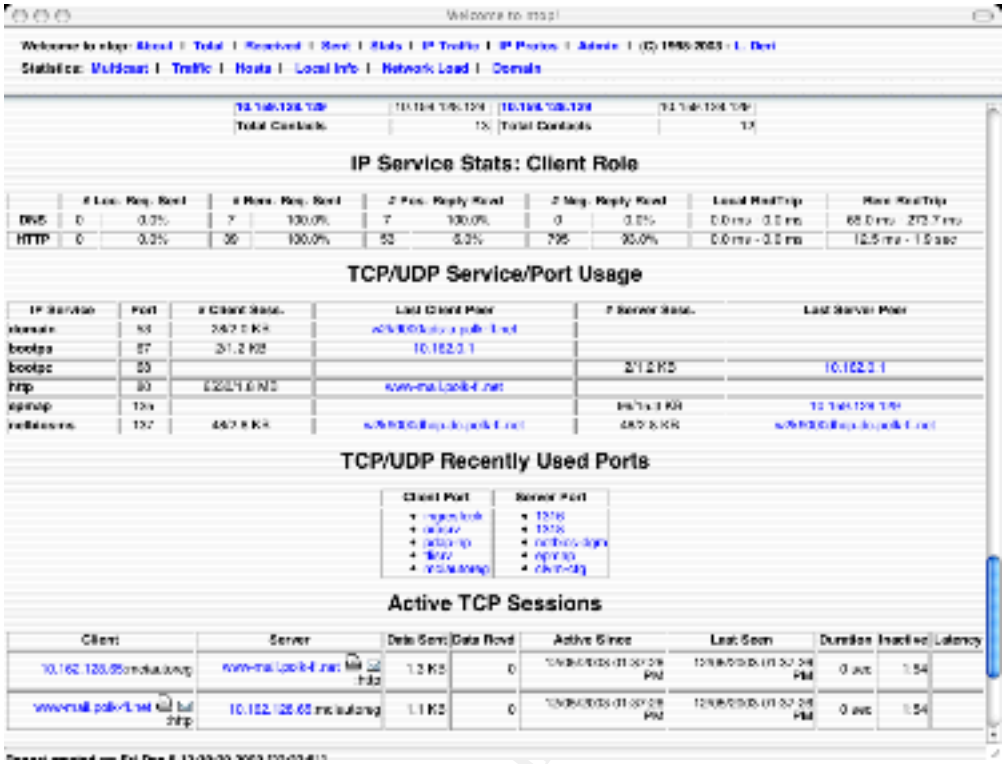
Source: The Author

Figure 15: Portion of an NTOP Individual Host Report.



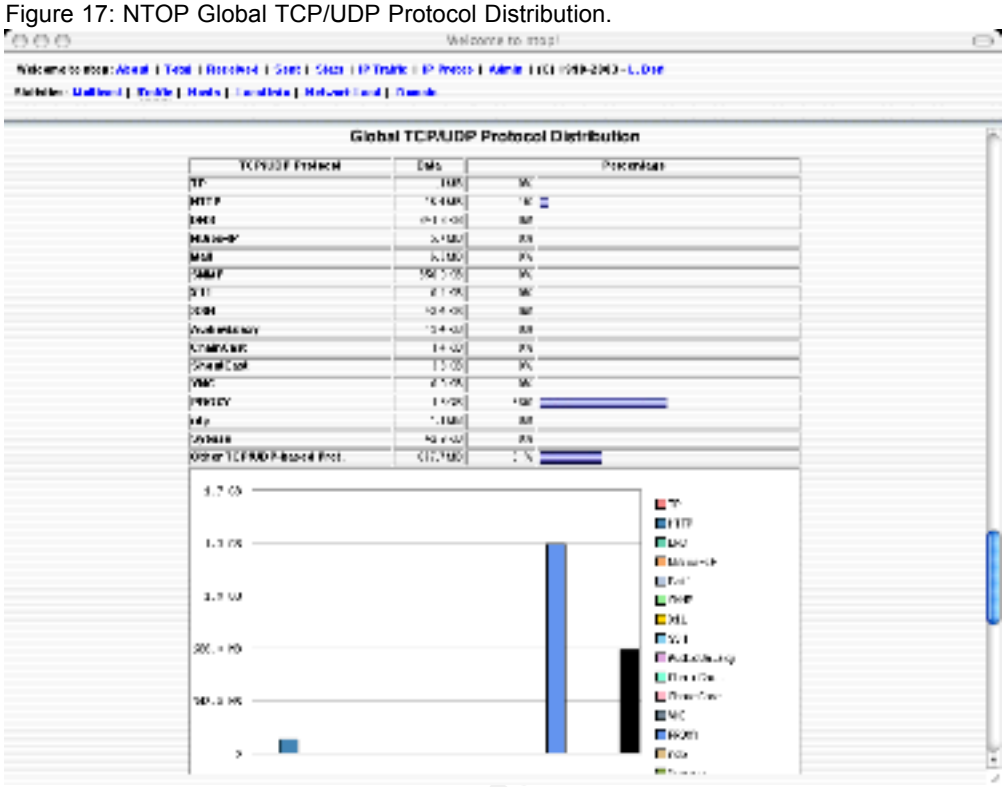
Source: The Author

Figure 16: Another Part of NTOP Individual Host Report.



Source: The Author

Figure 17: NTOP Global TCP/UDP Protocol Distribution.



Source: The Author

Results:

With the single exception of the glitch in the “suspicious packets” file, NTOP performed flawlessly. It gathered detailed statistics on the network as a whole as well as all hosts that it identified. The detail was such that after spotting port scan attempts with Snort, The exact ports and hosts involved could be viewed in NTOP. NTOP was also able to identify several other interesting pieces of information such as when and from which computer the night custodian was accessing the Internet and to which web sites the involved computer connected. In another case, NTOP not only identified a host using peer to peer file sharing to down load mp3 files, but even gave the titles of those files.

Though NTOP lacks some of the advanced capabilities of a full fledged NIDS, it clearly enhances the network administrators abilities to monitor activity of all kinds on the network. Its extensive statistics and intuitive interface allow the user to quickly drill down to relevant details. As such, NTOP makes an excellent adjunct to other network intrusion detection systems.

Prelude

Introduction:

According to it's web site, Prelude (available at http://www.prelude-ids.org) bills itself as a "a new innovative Hybrid Intrusion Detection system designed to be very modular, distributed, rock solid and fast."

Capabilities:

Prelude is very similar to Snort in its claimed capabilities. In fact, prelude uses the exact same rules file format as Snort, and comes with Snorts own default rule set. Prelude claims to be able to handle a greater level of traffic due to a more efficient design. This claim could not be verified. Prelude also adds simple host based integrity checking.

Installation:

Prelude is available in RPM (RedHat Package Manager) format for those Linux distributions which utilize it. Other distributions must compile from source. Both the RPMs and source compilation were tested with identical results. In either case, once the binary files are in place, a MySQL database must be created from a sql script included in the download. Preludes configuration files must be updated with host names and addresses of the sensor and database server. The appropriate rules files must be installed. Finally, a sensor must be configured to communicate with the manager by authenticating it to the manager application using a one time password. All of these steps were completed without difficulty.

Results:

The test results for Prelude were a complete disappointment. After successful installation and configuration the program launched without errors. Messages on the console output indicated that the sensor had launched successfully and had bound itself to the network interface. It also indicated that it had successfully connected to the database. However, after allowing the program to run overnight, it had not logged even one packet! There were no error messages of any kind on the console output or in the system logs. Double checking all configuration files produced no errors either. The network interfaced was double checked to insure that it was in promiscuous mode. Finally the program was relaunched. Again, all console output indicated correct operation, but no packets were logged. The test was considered a complete failure.

SUMMARY OF RESULTS

Table 1 on the nex page provides a summary of the test results for the NIDS solutions examined here.

Table 1: Summary of Test Results.

	PureSecure	MIDAS v2.0	ACID	NTOP	Prelude
Events Logged per 24 hrs	21,000[4]	149	15,000[4]	n/a[3]	0
Unique Events per 24 hrs	100[5]	2	100[5]	n/a[3]	0
Ease of Installation[1]	4 Automated install script works well with little user intervention. Single daemon process requires no configuration.	1 No automated install of any kind. 6 daemon processes require manual configuration with little to no feedback.	4 Download and install well documented and straightforward.	5 Download and install simple and well documented.	3 Download and install well documented but require multiple steps with not automation.
Ease of Configuration[1]	4 Automated Snort configuration & Verification of Snort rules.	2 Automated Snort Configuration Failed. No Rules Verification.	3 ACID console requires minimal manual installation, but provides no Snort configuration of any kind.	4 One configuration file must be manually edited. Some advanced reports require one time configuration.	3 Configuration well documented but require multiple steps with no documentation.
Reporting Capabilities	Extensive, built in, automated report configuration.	Minimal	Extensive, built in, requires manual report configuration.	Extensive, built in, some advanced reports require initial configuration.	n/a[2]
Support	Excellent Documentation available in PDF format. No support offered for Personal Edit Phone and Email support available for commercial version.	Online forum and mailing list with little traffic. In some cases days past between posts and responses.	Online forum and mailing lists with high traffic. Well written online documentation.	Online forums and mailing with high traffic. Well written man page, but user documentation minimal and out of date. Paid phone, email and consulting support available.	Online forum and mailing list. Moderate traffic.

NOTES: 1. Scale of 1 to 5, 5 being easiest, 1 being most difficult
2. Unable to test due to lack of data.
3. Not applicable to this test.
4. Average rounded to nearest thousand.
5. Estimate. Number of unique events is highest on first day of testing and steadily declining there after.

Implementation:

At the completion of the evaluation phase both NTOP and PureSecure were the clear leaders. The decision was made to roll out the NIDS based on these two products. A test system was installed at the main switch of the campus using the "poor man's ethertap" approach illustrated earlier. and allowed to run for three full months. During that time, the Snort rules were tuned and additional "pass" rules were added to the Snort configuration to tell Snort to ignore normal traffic that would otherwise generate false positive alerts. After two weeks of operation, the test system was generating less than a dozen alerts daily with only an average of two to three false positive alerts. The test system was successful in finding several workstations within the campus that were infected with various viruses and malware. It was also successful in alerting the operator to port scans coming in from other campuses.

NTOP also proved itself to be highly effective. After running for several weeks NTOP had provided an excellent graphical view of normal network traffic. Once the author became familiar with this normal "base line" it was very easy to pick out exceptions, such as spikes in usage of a given port. These anomalies could then be investigated by looking at detailed reports for individual hosts provided by NTOP or by applying filters to look for specific traffic. New Snort rules based on the traffic patterns supplied by NTOP could also be applied, though in practice, this was rarely necessary as NTOP was nearly always capable of pointing out the source of the abnormal traffic. In one example, NTOP not only pointed to a workstation running a peer-to-peer file sharing program, but even listed the titles of the mp3 files downloaded by the user.

At the end of the three month test the author met with district officials to review the results. All agreed that despite the difficulties of not having access to the switches, the test had been a resounding success. However, there was considerable worry over the wording in the PureSecure license and just exactly what constituted "non commercial use".[4] Since the district could not afford to pay for a license from Demarc should they be found to be out of compliance, the decision was mode to drop PureSecure and the author was asked to reimplement the test using only open source software.

This led to a reevaluation of the entire project. At this point nearly a year had passed since the initial inception of the project and the decision was made to test the newest release of MIDAS in the hope that it's shortcomings had been addressed and would suffice to take the place of PureSecure in the test system.

In early 2004 the author was able to attend the SANS GSEC training class in

[4] Though the school district itself is a non-profit orginization, there was concern that since the network managers within the district are full time, salaried employees, this could be interpreted as requiring a commercial license for PureSecure.

Orlando Florida. Upon returning from this event, the author installed and tested MIDAS v2.2. MIDAS, by this time, had undergone significant change and enhancement. The web based user interface was completely overhauled and enhanced. Significantly, many aspects of its operation could now be configured through the web interface. The earlier failures of the system to log alerts did not occur during this installation and MIDAS was able to launch snort and collect data with no difficulties.

This newer version of MIDAS still requires the administrator to manually edit the configuration files for the various components which would make it somewhat more difficult to roll out at the district level. Also, there is still no provision for verifying the Snort configuration as in PureSecure, thus the administrator is still required to to verify the Snort configuration manually. These tasks can be partially automated through the use of shell scripts and do not present a significant impediment to roll out.

The test system was put back into operation with MIDAS v2.2 in place of PureSecure. System performance was on par with the earlier PureSecure based system. In fact, some of the district network managers who viewed the system preferred the new MIDAS web interface to the PureSecure interface. Several made comments that it was both more intuitive and had better response time (though this was not unanimous, see figure 18.).

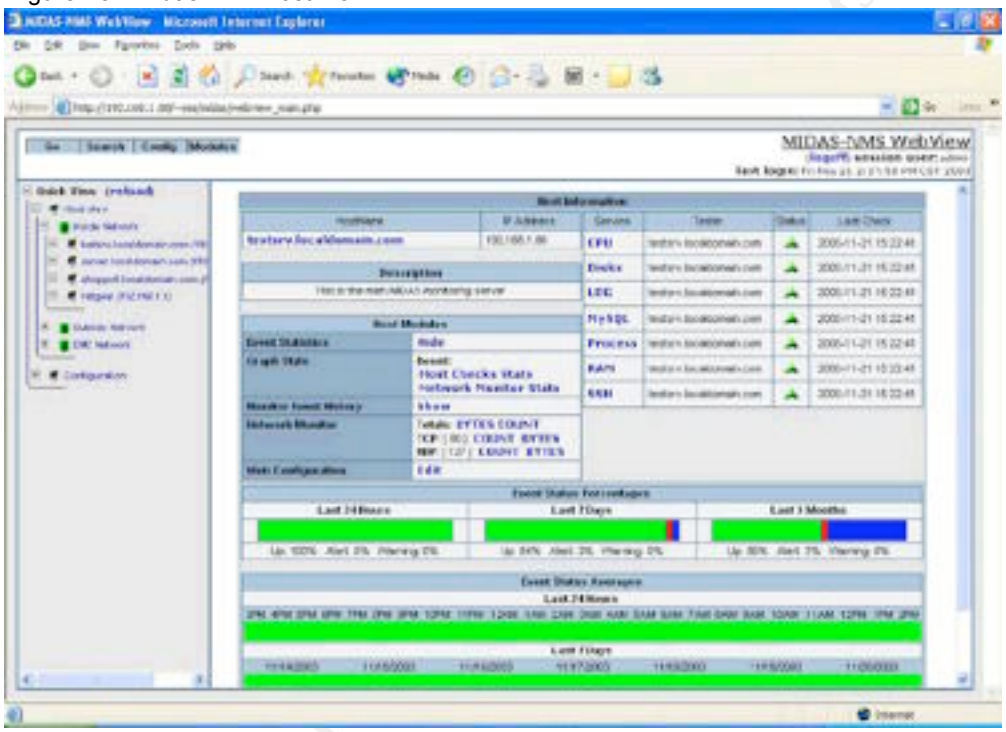
As a result of the SANS training, the author became aware of a fundamental flaw in the implementation. This flaw is best illustrated by an incident in which a student had gained access to a VNC password and had used it to access a desktop in the campus' Media Center. Fortunately, the individual involved was quickly identified and the passwords were immediately changed. Equally fortunate was the fact that the student was only able to access other student workstations as the teacher workstations used different passwords. However, because the NIDS was designed to look only at traffic flowing out of and into the network at the router, there was no trace of this incident within either NTOP or the Snort logs.

Since the campus uses a switched network, a mirror port on the switch would be necessary to allow monitoring of the campus' internal traffic. Unfortunately, the district department responsible for maintaining and configuring all switches and routers had declined to participate in the test. For this reason it was not possible to monitor internal campus traffic. This incident clearly illustrates how organizational politics and lack of "buy in" by important stake holders can compromise a project.

In order to demonstrate the ability of the system to scale at the district level, MIDAS/Snort sensors were placed at each school within the campus. These sensors were configured to log back to the main console. With their

configurations properly tuned these sensors generated negligible network traffic and proved to be an additional burden to the network infrastructure. The MIDAS web interface made it easy to update the Snort configurations for each sensor from one central console. However, since no funding was available for this test, the computers used for the sensors had to be pulled from service after approximately two weeks and put into classroom service.

Figure 18: Midas v2.2 Host view.



Source: http://midas-nms.sourceforge.net/screenshots.html

Conclusion:

The choice of the “best” NIDS is not as simple as looking at tables of data and reviews. Each IDS has its own strengths and weaknesses. To be effective, these strengths and weaknesses must be matched up against the type and amount of network traffic expected and the skill level of the network administrator. One thing is clear, while host based and network based IDS’s represent two divergent views on intrusion detection, these views are complimentary. The most effective protection will be achieved by utilizing both host based and network based intrusion detection.

In this project, PureSecure clearly stood out among the others. However, ACID also proved itself to be a viable alternative for smaller networks or for those who

do not qualify for the noncommercial version of PureSecure and cannot afford the commercial license (starting at approximately $2,300.00).

MIDAS v2.2 proved itself to be an able substitute for PureSecure even if it lacked some of the more advanced features. MIDAS also has the benefit of being under active and continued development and supports a very active user community.

Prelude failed to collect any meaningful data in this test and proved both difficult to administer and configure. It should be pointed out that this program has been used successfully on other networks. More than likely, the problems in this product lie in undiscovered platform and/or library incompatibilities. This suggests that considerable development effort is needed.

NTOP, while not purely a NIDS program showed itself to be an excellent adjunct to any network security and monitoring solution and was also praised for it's ability to help enforce acceptable use policies as well as it's ability to show anomalies in network traffic.

At the conclusion of the project, the test system was demonstrated for district officials and the results reviewed. The system received high praise and was submitted as a model for roll out across the district. Several network managers at other campuses voiced their enthusiasm for this project. However, by the conclusion of the 2003/2004 school year, no action had been taken concerning the proposal.

As of June 2004 the system remained in service at the author's campus and remained effective even though the author was only able to devote a few hours of his time to it each week. The number of alerts generated has continued to decline as incoming scans and alerts from computers at other campuses were reported to district officials. Thus, even this one IDS running at only one campus has had a district wide effect.

As a final point, it must be stated emphatically that no intrusion detection system is complete without some form of host based integrity checking. This point was driven home by the recent attacks against the Debian Linux project (Schulze). Three of their servers were successfully attacked and compromised by an unknown attacker who exploited a buffer overflow in the Linux kernel. In this case the debian.org NIDS failed to catch the intrusion, but the host based IDS[5] on each server, in combination with the system logs did!

[5] Debian.org utilizes AIDE, Advanced Intrusion Detection Environment, an open source alternative to Tripwire with equivalent functionality. Available at http://aide.sourceforge.net.

Bibliography

Mick Bauer, "**Intrusion Detection for the Masses"**
Linux Journal, July 01, 2001
http://www.linuxjournal.com/article.php?sid=4718

Mick Baur, "**Stealthful Sniffing, Intrusion Detection and Logging"**
Linux Journal, October 01, 2002

Kumar Das, "**Protocol Anomaly Detection for Network-based Intrusion Detection"**
January 5, 2002
http://www.sans.org/rr/papers/index.php?id=349

Luca Deri, "**Ntop: a Lightweight Open-Source Network IDS"**
Finsiel S.p.A. Centro Serra, University of Pisa. September 2000.

Luca Deri, "**Passively Monitoring Networks at Gigabit Speeds Using Commodity Hardware and Open Source Software"**
NETikos S.p.A. 2002

Luca Deri, Stefano Suin, and Gaia Maselli,
"**Design and Implementation of an Anomaly Detection System: an Empirical Approach"**
Proceedings of Terena TNC 2003, Zagreb, Croatia, May 2003.

Ido Dubrawsky, "**Freeware Intrusion Detection Tools"**
Sys Admin Magazine, August 2001
http://www.samag.com/documents/s=1147/sam0108o/0108o.htm

Victor R. Garza and Joseph L. Roth, "**Inspecting the Inspectors"**
Infoworld Magazine, August 23, 2004: 39-47

Nalneesh Gaur, "**Snort: Planning IDS for Your Enterprise"**
Linux Journal, July 11, 2001
http://www.linuxjournal.com/article.php?sid=4668

Roger Grimes, "**The New Security: Many Threats Many Solutions"**
Infoworld Magazine, August 30, 2004: 43-47

Allison Hrivnak,
"**Host Based Intrusion Detection: An Overview of Tripwire and Intruder Alert**
January 29, 2002
http://www.sans.org/rr/papers/index.php?id=353

Chris Kuethe, "**Homebrew Intrusion Detection Systems"**
Sys Admin Magazine, August 2001
http://www.samag.com/documents/s=1147/sam0108m/0108m.htm

Stuart McClure, Joel Scambray and George Kurtz, **Hacking Exposed, 4th Edition**
Osborner/McGraw Hill, New York 2003

William Metcalf,
"**A Practical Guide to Running SNORT on Red Hat Linux 7.2 and Management Management Using IDS Policy Manger MySQL"**, April 2, 2002
http://www.sans.org/rr/papers/index.php?id=360

Rich Paredes, "**Detecting and Removing Trojan Horses on Linux"**
Sys Admin Magazine, August 2002
http://www.samag.com/documents/s=7467/sam0208e/0208e.htm

Michael Rash, "**Detecting Suspect Traffic"**
Linux Journal, November 01, 2001
http://www.linuxjournal.com/article.php?sid=4876

Danny Rozenblum, "**Understanding Intrusion Detection Systems"**
August 9, 2001
http://www.sans.org/rr/papers/index.php?id=337

Martin Schulze, "**Debian Investigation Report After Server Compromises"**
Linux Today, Dec 2, 2003
http://www.linuxtoday.com

J.S. Sherif; R. Ayers; T.G. Dearmond,
"**Intrusion detection: the art and the practice. Part I."**
Information Management & Computer Security, August 27, 2003

J.S. Sherif; R. Ayers, "**Intrusion detection: methods and systems. Part II.** "
Information Management & Computer Security, OCE 22, 2003

Jeffrey Slonaker, "**An Overview of PureSecure™"**
May 12, 2003
http://www.sans.org/rr/papers/index.php?id=1036

Peering Over the Firewall
Jeffrey L. Taylor
Linux Journal, July 23, 2003
http://www.linuxjournal.com/article.php?sid=6985

Kristy Westphal, “**Snort — A Look Inside an Intrusion Detection System”**
Sys Admin Magazine, Septermber 2000
http://www.samag.com/documents/s=1161/sam0009f/0009f.htm

Kristy Westphal, “**SNAREing Intruders in Linux”**
Sys Admin Magazine, August 2002
http://www.samag.com/documents/s=7467/sam0208a/0208a.htm