Interested in learning more about security?

SANS Institute InfoSec Reading Room

A Design for Building an IPS Using Open Source Products

An IPS or Intrusion Prevention System can be an important component for protecting systems on a network. An IPS is based upon an IDS or Intrusion Detection System with the added component of taking some action, often in real time, to prevent an intrusion once detected by the IDS. This paper describes a design for an IPS built from all Open Source products and is based upon research done at the Illinois Institute of Technology's Rice campus. The goal of the research was to develop a design for an IPS that could be appli...

A Design for Building an IPS Using Open Source Products

Author:

Mike Smith

Contributors:

Sean Durkin

Kaebin Tan

8/10/2006

Abstract

An IPS or Intrusion Prevention System can be an important component for protecting systems on a network. An IPS is based upon an IDS or Intrusion Detection System with the added component of taking some action, often in real time, to prevent an intrusion once detected by the IDS. This paper describes a design for an IPS built from all Open Source products and is based upon research done at the Illinois Institute of Technology's Rice campus. The goal of the research was to develop a design for an IPS that could be applied to any small to medium sized network. This paper is written for technical integrators who are interested in building their own IPS without incurring software licensing costs.

Table of Contents

1. Introduction

At a high level an Intrusion Prevention System or IPS consists of the following components:

- Network Intrusion Detection System or NIDS to capture all network traffic flows, analyze the content of individual packets for malicious traffic and generate security events.
- A central rules engine that captures the security events and generate alerts based on the events received.
- A console to monitor events and alerts and control the NIDS
- A component that takes action based on the alerts and attempts to block the malicious traffic.

Following are some general techniques and IPS can employ to actively block or prevent malicious traffic:

- Injecting TCP reset packets into the attacker's connection to the victim system, thereby terminating the connection
- Reconfiguring routers and firewalls to block packets from the attacker's apparent location (IP address or site),
- Reconfiguring routers and firewalls to block the network ports, protocols, or services being used by an attacker, and
- In extreme situations, reconfiguring routers and firewalls to sever all connections that use certain network interfaces. [Bace, Mell, 2001]

For this design the Snort IDS: http://www.snort.org/ provided the base IDS system and rules engine, SnortSam http://www.snortsam.net/index.html a plug-in for Snort provided the IPS function and BASE an open source PHP application provided the console function. SnortSam uses the "Reconfiguring routers and firewalls" technique as its blocking mechanism.

The SnortSam product contains two components. The first component gets compiled into Snort as an output plug-in module. The output plug-in is enabled by adding a Snortsam keyword and a blocking action statement to any Snort rule. The output plug-in is then called when an intrusion is detected by a rule with the Snortsam statement. The second SnortSam component is a firewall agent that runs on firewalls, or on hosts with firewalls or on hosts that control firewalls. The agent receives a blocking request from the output plug-in then verifies the blocking request is valid and if valid reconfigures the firewall to block the IP address reported by the IDS for the time specified in the SnortSam statement.

The IPS design described in this paper integrates a distributed Snort IDS sensor with a SnortSam output plug-in and SnortSam agents running on Linux hosts with IPTables. With this configuration intrusions are detected at a network level and prevented at a host level.

The goals of this design was to develop a system that would monitor all campus network traffic, log any occurrences of packets containing signatures of malicious activity and protect campus servers from an internal or external attack in real time. The benefits this paper offers to the reader are as follows:

- Product Research and Selection. The paper describes what Open Source products were used to construct the IPS and why those products were selected.
- Sample hardware configurations for the system components.
- Design model that shows how the components are connected and the network flows between the components.
- Signature rules research and recommendations for tuning the signature rule set.

2. Components

The following sections describe the base sub-system components that make up the IPS and the criteria used to select them.

2.1. Operating System

One of the basic requirements of the design was to use Open Source products. Because IIT's hardware standard is Intel based servers our choices for OS were basically a Linux variant or FreeBSD.

The Snort project includes several written documents for installing and configuring the product. One of the best documents we found was written for a Linux installation. For the OS we selected the Fedora Core 4 OS.

Fedora Core 4 is distributed by the Fedora Linux Project http://fedora.redhat.com/, which is sponsored but not supported by RedHat. Fedora is managed by a steering committee and the OS is based on previous versions of the RedHat Linux. For us this meant Fedora, although not officially supported by RedHat, likely has many forums and other support communities available. Also, and most importantly, Fedora Core 4 can be downloaded free.

2.2. NIDS

The product selected to be the base NIDS system was Snort version 2.4.3. Snort is a NIDS that employs both a signature and anomaly based detection system that is based on a rules database. The choice to use Snort was an easy decision based on several factors. The Snort open source project is the most well known open source NIDS product available today. There are other open source IDS products available such as Prelude, Firestorm, LIDS Project to name a few, but none of these products seem to have the support structure or prevalence that Snort does.

For example, the Snort project includes a comprehensive Users Manual as well as a very specific installation guide. Additionally, the Snort project is supported by a commercial fee-based subscription service that provides rule updates as new threats are discovered. Although fee-based is not usually associated with Open Source, in this case the fee is not for software but for the service of receiving updated rules and ensures that the rules will be maintained on a regular basis. The current subscription fees from www.snort.org are $1,795 for an annual subscription, $495 for a quarterly subscription and $195 for a monthly subscription. Additionally, for those who do not wish to pay the subscription fee the rules will continue to be distributed freely with new Snort releases.

2.3. Console Logging, Alerting and Reporting

In the simplest configuration when Snort detects an intrusion and generates an alert Snort will log the alert in ASCII format to local log files. A more complex option is to configure Snort with an Output module that will log the alerts to a Database. From the DB other 3rd party "Console" products process the alerts display them in various views and formats and make the views accessible via browser. This is a very popular option with Snort configurations and an option we chose for its obvious benefits. There are two prevalent open source Snort Console products available today, ACID and Base.

ACID (Analysis Console for Intrusion Databases) is very outdated and not recommended. BASE (Basic Analysis and Security Engine) is written in PHP and based on the code from ACID. Therefore we chose Base 1.2.2 as our Snort Console.

2.4. IPS

Snort, in addition to being an IDS, can also be configured as an IPS. The IPS portion of Snort comes from its ability to be configured as an Inline IPS that integrates with IPTables on a Linux system. However, there are several issues with this option.

First, for the Snort Inline IPS to protect a network the Linux system running IPTables has to be the network gateway. Although running Linux with IPTables as a network gateway is a viable option generally appliance type routers from Cisco, Linksys or Netscreen are faster and require less maintenance and support than a Linux system running on a Server. Because of these limitations we decided to search for another IPS option.

What we found was SnortSam http://www.snortsam.net/index.html a Snort output plugin that provides a more flexible method for providing Intrusion Protection.

> SnortSam is a plugin for Snort. SnortSam itself consists of two pieces – the output plugin within snort and an intelligent agent that runs on the firewall, or a host near the firewall. The plugin allows for automated blocking of IP addresses on following firewalls:
>
> - Checkpoint Firewall-1
> - Cisco PIX firewalls
> - Cisco Routers (using ACL's or Null-Routes)
> - Former Netscreen, now Juniper firewalls
> - IP Filter (ipf), available for various Unix-like OS'es such as FreeBSD
> - FreeBSD's ipfw2 (in 5.x)
> - OpenBSD's Packet Filter (pf)
> - Linux IPchains
> - Linux IPtables
> - Linux EBtables
> - WatchGuard Firebox firewalls
> - 8signs firewalls for Windows
> - MS ISA Server firewall/proxy for Windows
> - CHX packet filter
>
> The SnortSam agent provides a variety of capabilities that go beyond other automated blocking mechanisms, such as:
>
> - White-list support of IP addresses that will never be blocked.
> - Time-override list.
> - Maximum block time ceiling as well as minimum block time definition for reporting entities.
> - Flexible, per rule blocking specification, including rule dependent blocking time interval.
> - A SID filter list of allowed or denied SIDs based on reporting entity.
> - Misuse/Attack detection engine (including roll-back support) that attempts to mitigate the risk of a self-inflicted Denial-Of-Service in the IDS-Firewall integration.
> - Repetitive (same IP) block prevention with customizable window to improve performance.
> - TwoFish encrypted communication between Snort and the SnortSam agent.
> - True OPSEC support using the Checkpoint SDK (opsec plugin).
> - Block tracking and block expiration for firewalls that don't support timeouts.
> - Multi-threading for faster processing and simultaneous block on multiple devices.
> - File logging and email notification of events. [Knobbe, 2006a]

SnortSam is an Open Source product that can be downloaded for free and integrates with recent Snort 2.4.x versions. The output plugin is called by adding a command to any of the Snort IDS rules in which you would like to enable intrusion protection. Because of its flexibility and integration capabilities with Snort we chose SnortSam 2.4.3 as the IPS product for our design.

3. System Design

The public network at Rice consists of multiple switches connected in a daisy chain and configured as a single public VLAN with a class C public IP address block. Student lab computers and the campus servers like DHCP, DNS, Web Servers, etc. connect to this public VLAN. There are also two private networks connected to the public network. The private networks are single switches with one VLAN and a private IP address block. The campus wireless network is one private network and the staff network is the other private network. The private networks connect to the public network through NAT Firewalls and only allow outbound traffic from the private networks.

The basic design approach for the IPS is a distributed design that consists of two systems. The first system is a Snort IDS with a promiscuous port on the public network with a SnortSam IPS output plug-in enabled. This is the Distributed IDS/IPS. The second system a Snort IDS with a SnortSam IPS output plug-in enabled, BASE, Apache, MySQL installed. This is the Central IDS/IPS. The servers requiring protection are running SnortSam FW agents with IPTables enabled. These are the distributed agents.

A distributed design was selected to ensure the system performed well and to help protect the integrity of the system. The following sections provide more detail regarding these design characteristics. Figure 1 below shows the Rice network design and how the Central IDS/IPS the Distributed IDS/IPS and a campus server running the SnortSam FW connect to the campus network.

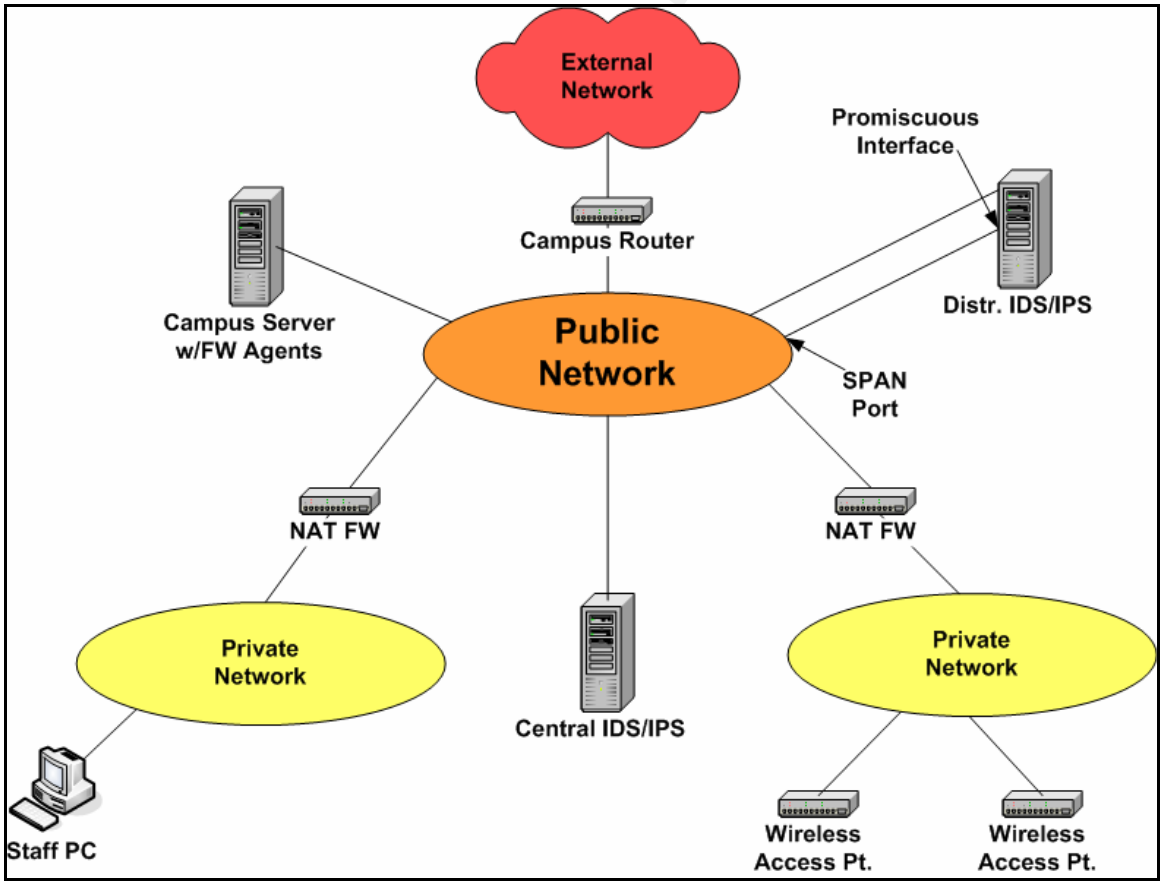

Figure 1: System Design

3.1. IDS & IPS Integration

The installation package for Snort includes a comprehensive set of rules for detecting attacks. The Snort rules provide the signatures of the known attacks and provide the basis for the IDS alert process. To enable a SnortSam block to occur for a particular attack the Snort rule for that attack must be modified to include the call to the SnortSam output plug-in. This is the integration point between Snort and SnortSam. The following section shows an example original Snort rule and the same rule with the SnortSam statement added.

> Original Rule:
> $HOME_NET any (msg:"SCAN nmap XMAS"; flow:stateless; flags:FPU,12; reference:arachnids,30; classtype:attempted-recon; sid:1228; rev:7)
>
> Changed Rule:
> $HOME_NET any (msg:"SCAN nmap XMAS"; flow:stateless; flags:FPU,12; reference:arachnids,30; classtype:attempted-recon; sid:1228; rev:7; **fwsam: src, 5 minutes;**)

This tells SnortSam to send a command to the SnortSam firewall agent to block the source IP of the attacker for 5 minutes. This is the mechanism that enables the IPS function in this design.

3.2. Network Integration

A key factor in the ability of the Distributed IDS/IPS to detect intrusions is the promiscuous Ethernet interface. When in promiscuous mode the Ethernet adapter will read all frames from the physical media as opposed to non-promiscuous mode where only frames containing the adapter's MAC address or broadcast frames are read. When Snort is configured to listen on the promiscuous interface the Snort IDS will receive all frames traversing the Ethernet LAN. However, switched networks use VLANS and all ports within a VLAN do not see all the frames. A switch uses a MAC address table to send unicast frames to the specific port where the Ethernet adapter with the matching MAC address is connected and only sends broadcast frames to all ports in the VLAN.

Switched networks use port mirroring to enable traffic monitoring on the VLAN. Port mirroring sends a copy of all frames received or sent from a port or a VLAN to a specific port on the switch. For Cisco switches port mirroring is called SPAN and the SPAN port is where the IDS/IPS promiscuous interface should be connected. SPAN stands for Switched Port Analyzer. Following is a definition of SPAN from www.cisco.com: "SPAN extends the monitoring capabilities of existing network analyzers into a switched Ethernet environment. SPAN mirrors the traffic at one switched segment onto a predefined SPAN port. A network analyzer attached to the SPAN port can monitor traffic from any of the other switched ports."

When configuring SPAN you specify source port(s) or source VLAN that you want to copy frames from, then you specify whether you want to copy inbound, outbound or both and then you specify a destination port (SPAN port) where the copied frames will be sent. For switch environments that consist of multiple interconnected switches RSPAN or Remote SPAN can be used to copy frames across switches to a destination SPAN port. Following are the recommended configuration options:

- Network: Public
- Port Type: RSPAN
- Source: Public Network VLAN
- Traffic Direction: Both

With the above configuration options the system can detect both internal and external attacks against any host in the public campus network. However, this configuration will generate the most SPAN traffic and thus incur the most overhead on the switches.

If switch processing capacity is an issue an alternative configuration would be to configure a SPAN port as follows:

- Network: Public
- Port Type: SPAN
- Source: Router switch port
- Traffic Direction: Both

With this configuration any attacks being launched externally against a campus host would be detected and any attacks being launched from within the campus against external hosts would also be detected. This is the configuration implemented at IIT's Rice campus.

3.3. Performance

At a high level the system consists of four primary functions:

- Monitoring the networks for malicious activity (IDS).
- Blocking malicious activity (IPS).
- Logging the alerts generated by the IDS and the blocking request by the IPS.
- Serving web pages showing the IDS alerts and the IPS blocking requests.

As whole the system is very network and disk I/O intensive with the monitoring of the flat public LAN being the most network I/O intensive function. In order to ensure the system performs well we decided to dedicate one system for monitoring the public network and to centralize the alert logging and web page serving to a second system.

The first system is the Distributed IDS/IPS that runs Snort and SnortSam, monitors all the inbound and outbound traffic on the public network and generates Snort alerts and SnortSam actions. The Distributed IDS/IPS has two interfaces one is promiscuous with no IP address assigned and connects to the SPAN port on the public network switch as described in previous section. The second interface is non-promiscuous, has an IP address assigned and connects to the public network. The second interface is used for sending data to the distributed agents and Central IDS/IPS and for managing and monitoring the system remotely. The Distributed system's primary function is monitoring the public campus network, sending alerts to the Central IDS/IPS and sending SnortSam blocking requests to the distributed SnortSam agents.

The second system is the Central IDS/IPS which runs Snort, SnortSam, BASE, MySQL and Apache. The central IDS/IPS receives and logs all Snort alerts and SnortSam actions and serves BASE and SnortSam log web pages. The central IDS/IPS has one interfaces that is promiscuous by default, has an IP address assigned and connects to the public network. This interface is will receive MySQL traffic and SnortSam alerts from the Central IDS/IPS. The Central system's primary task is receiving MySQL logging requests and SnortSam blocking requests from the Distributed IDS/IPS and serving the Base and SnortSam log web pages.

3.4. System Integrity

Any attacks detected by the Snort IDS trigger the SnortSam output plugin to send blocking requests to all the SnortSam FW agents over the network. The blocking requests will contain the source IP of the IDS/IPS and the IP frame the source IP of the attacker in the data field. This poses a risk if the hacker is sniffing the network and sees his IP in the payload of a packet. The hacker would then know his actions have been detected and would also learn the IP address of the IDS/IPS. In order to eliminate this risk all of the SnortSam blocking requests are encrypted. SnortSam comes with built in encryption and uses the twofish symmetric encryption algorithm. The symmetric key that is used between the SnortSam output plugin and the SnortSam FW agents is stored on the SnortSam conf files.

This option lists Snort sensors that SnortSam is accepting packets from.

You can specify hostname, IP address, IP address and network mask, and optionally an encryption key used configured for that host or network.

Examples:

accept 10.10.0.0/16, officepassword

accept snort1, hostpassword

accept 192.168.1.1 [Knobbe, 2006b]

Both of the IDS/IPS servers also run IPTables and SnortSam firewall (FW) agents so they can protect themselves from any attacks. The following section provides more details regarding how the SnortSam FW agents and IPTables provide protection at the host level.

3.5. Host Protection

This design uses Linux Hosts running IPTables, which is a host firewall that comes with Linux. SnortSam includes an IPTables FW agent that can dynamically update the IPTables firewall rules to block malicious traffic when it is detected by the Distributed IDS/IPS for a specified period of time. Through this technique servers and other hosts on a network can open ports as needed in order to provide services such as http, sql, telnet, ssh, etc. but still be protected from attackers by the IPS.

The SnortSam FW agent running on the hosts has a conf file that is used to configure the agent. The conf file tells the SnortSam agent which Distributed IDS/IPS to accept blocking requests from and includes a common keyword that is used by the encryption algorithm. The conf file can also be used to tune the SnortSam FW agent specifically for the host it's protecting. This is covered in more detail in section 4 below.

3.6. Network Flows

Figure 2 shows the network flows between the IDS/IPS components, network/system administrators and the host FW agents.

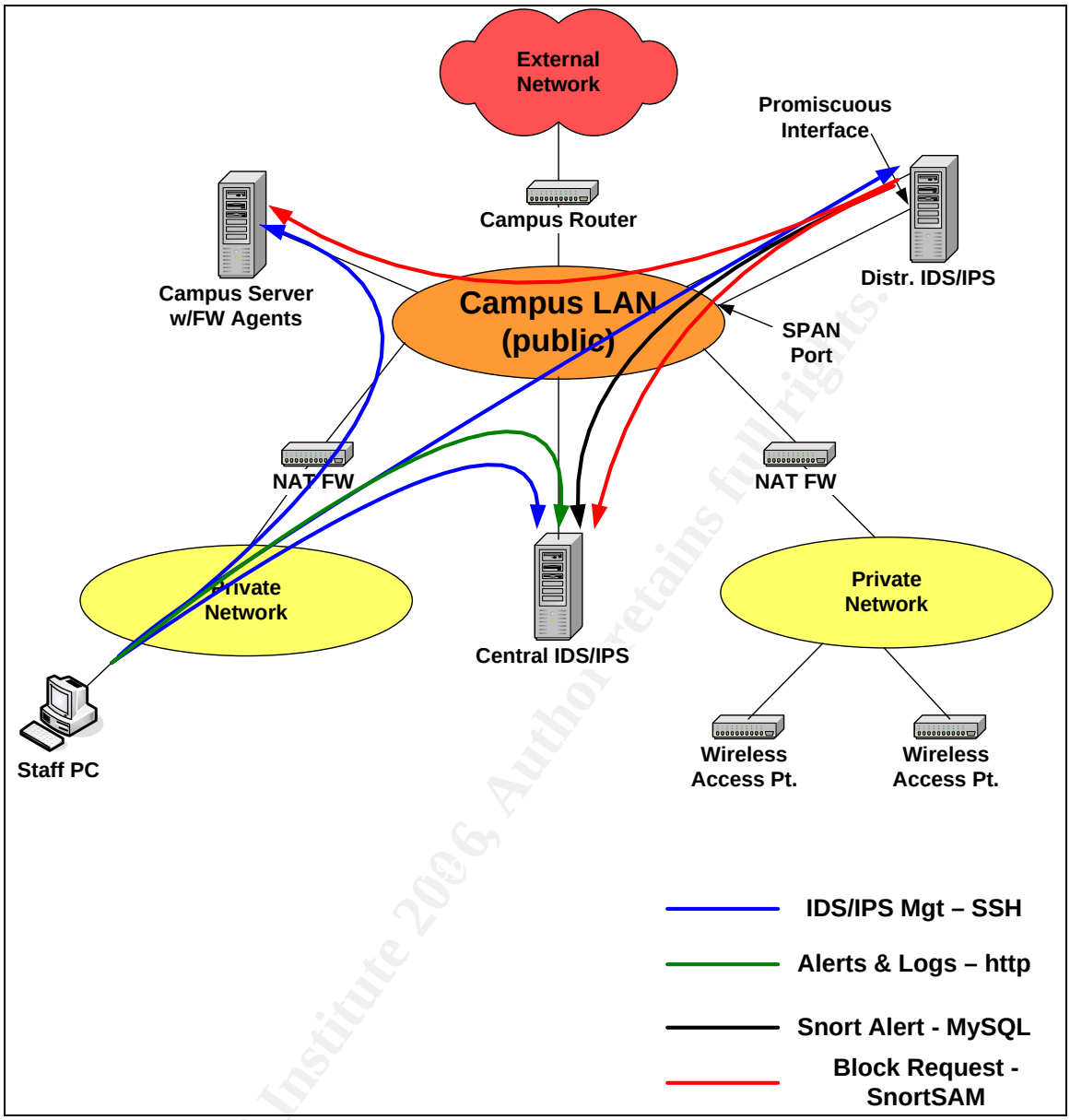

Figure 2: Network Flows.

3.7. IPS in Action Flows

The following figures show the flows that occur during an event where malicious traffic is detected on the network.

Figure 3 shows an attacker launching an attack.

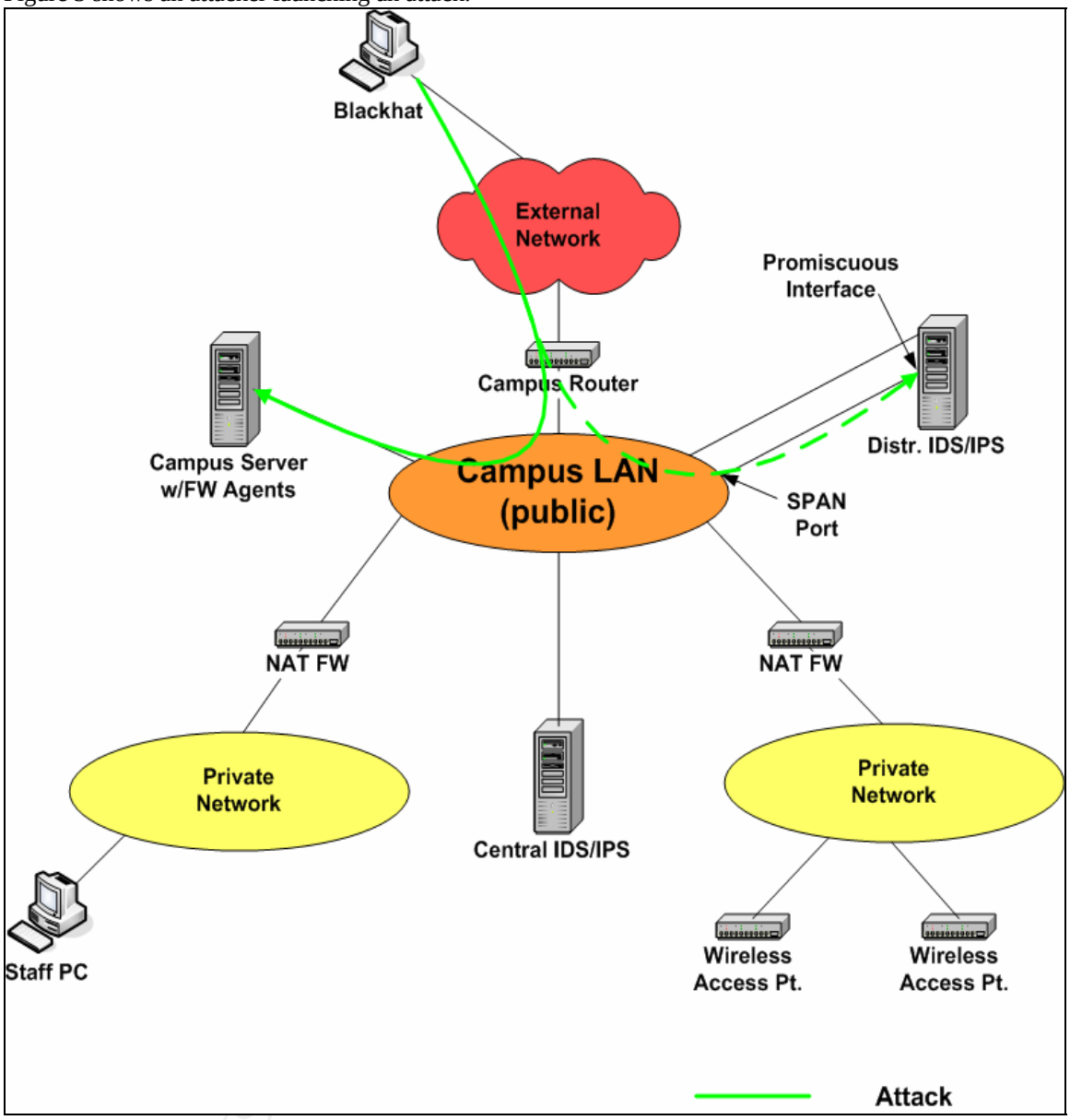

Figure 3: Attacker launching a port scan attack.

Figure 4 shows the SnortSam blocking requests from the SnortSam output plugin going to all of the SnortSam FW agents. In this figure the distributed IDS/IPS detects the attack (green) and sends Block Requests (red) to servers (and other key systems).

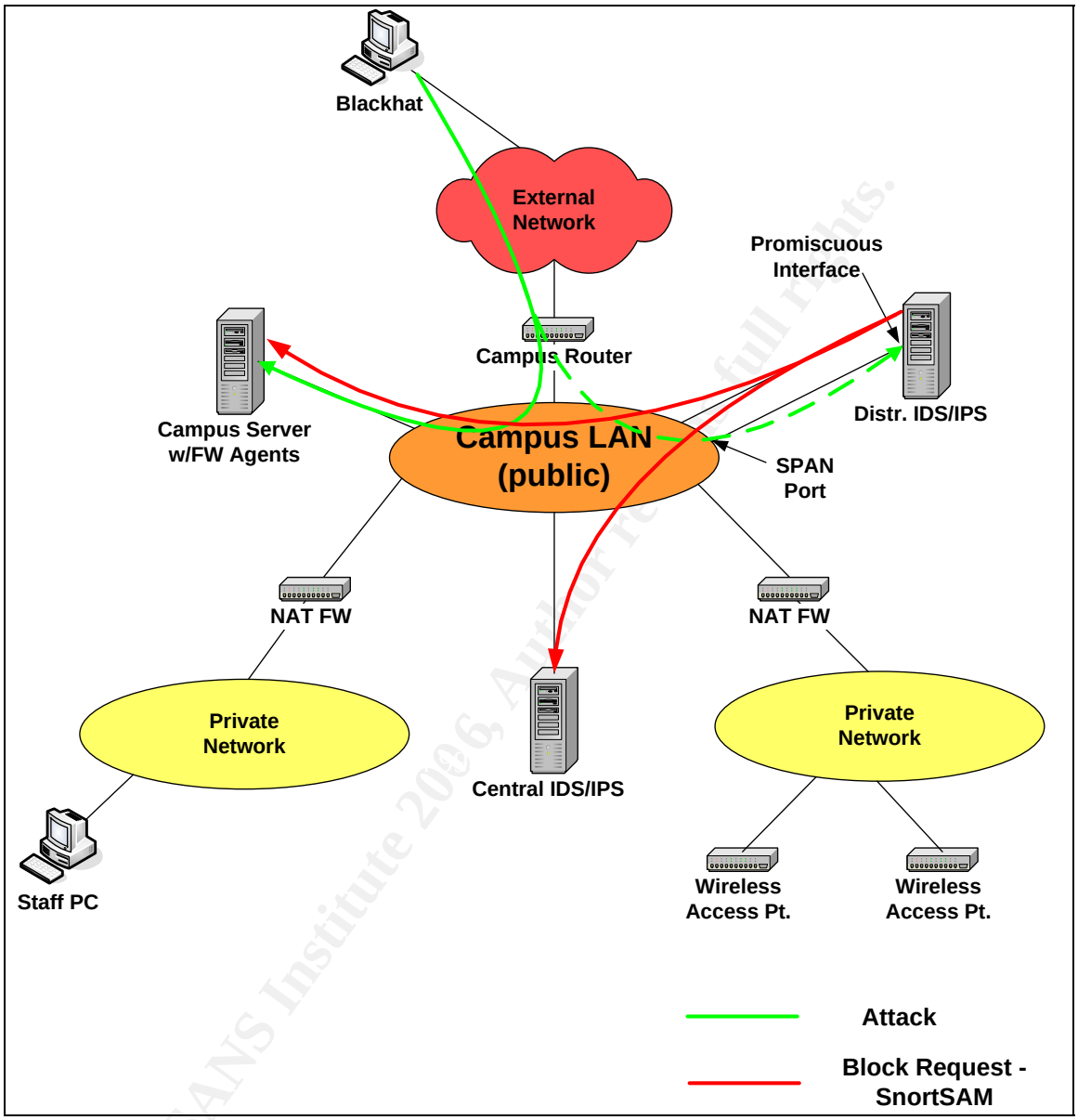

Figure 4: SnortSam blocking requests.

Figure 5 shows the host IPTables blocking the source IP of the attacker.

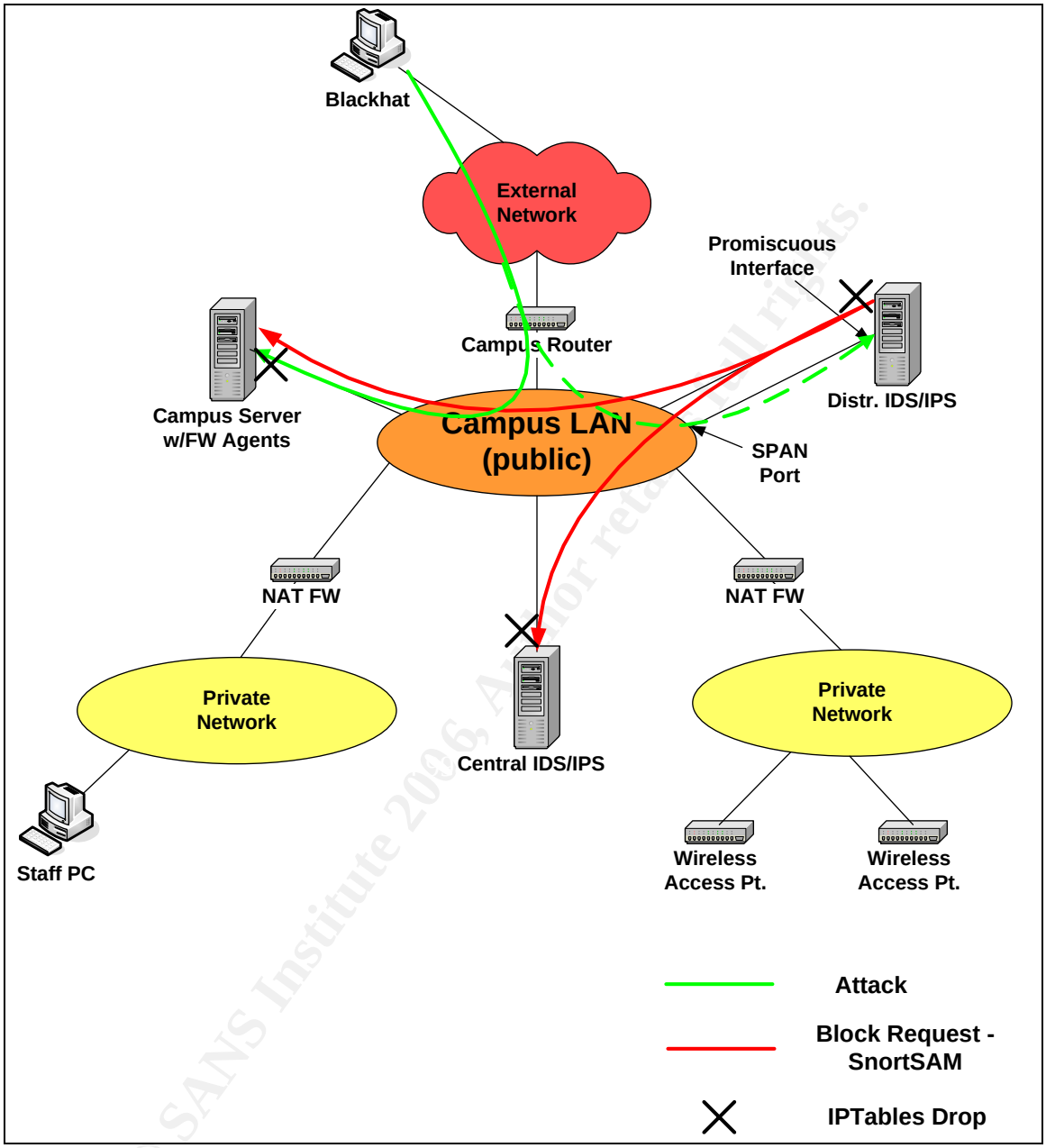

Figure 5: Host IPTables blocking the source IP of the attacker.

Figure 6 shows the MySQL alert logging.

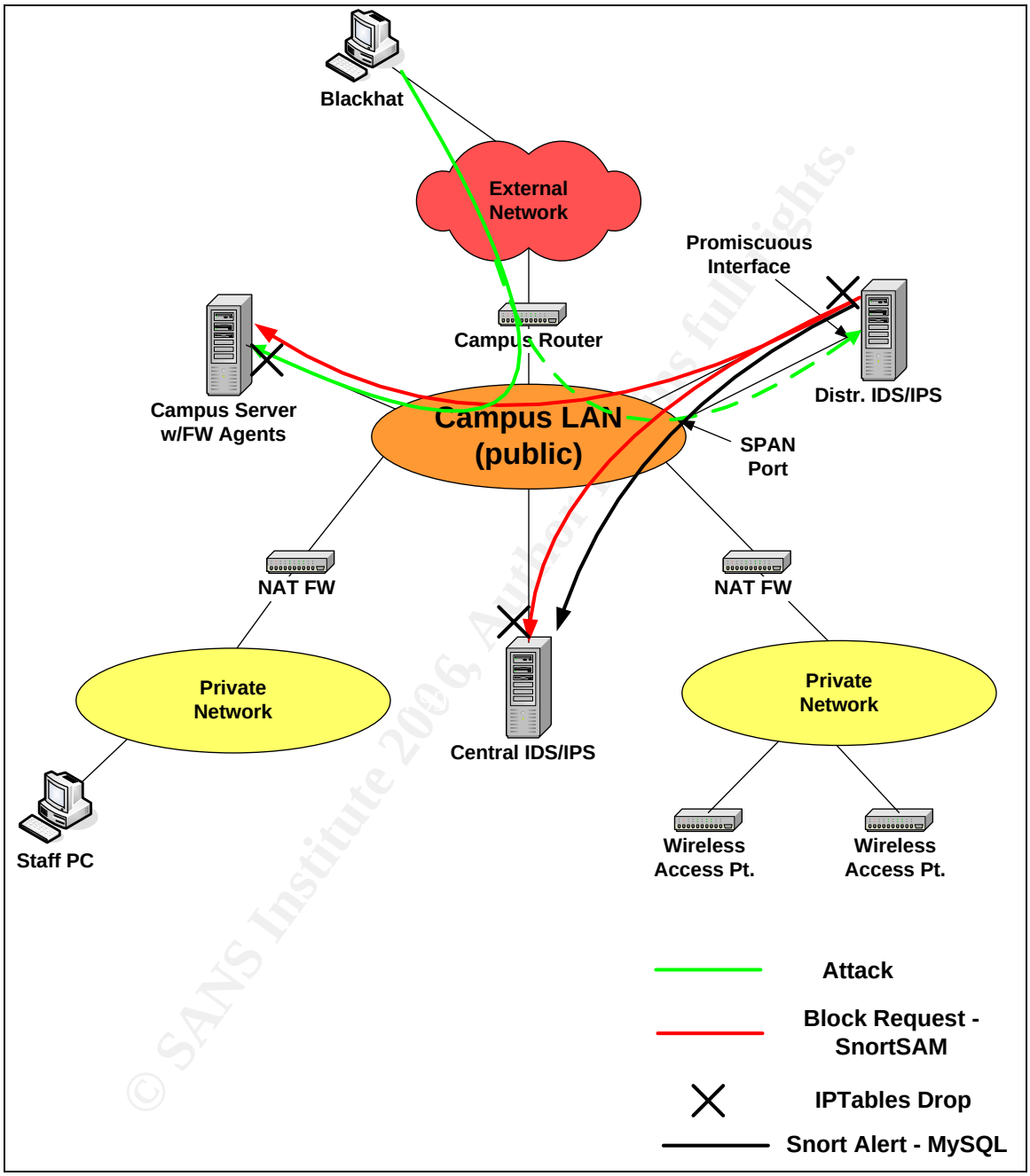

Figure 6: Snort Alert MySQL DB write request.

3.8. Logging and Reporting

The Central IDS/IPS will be handling the logging of all the Snort IDS Alerts to a MySQL DB. The Snort installation comes with a DB schema for MySQL that was used. The alerts in the DB are then made

available for viewing via browser through the BASE application. Figure 7 below shows the BASE home page.

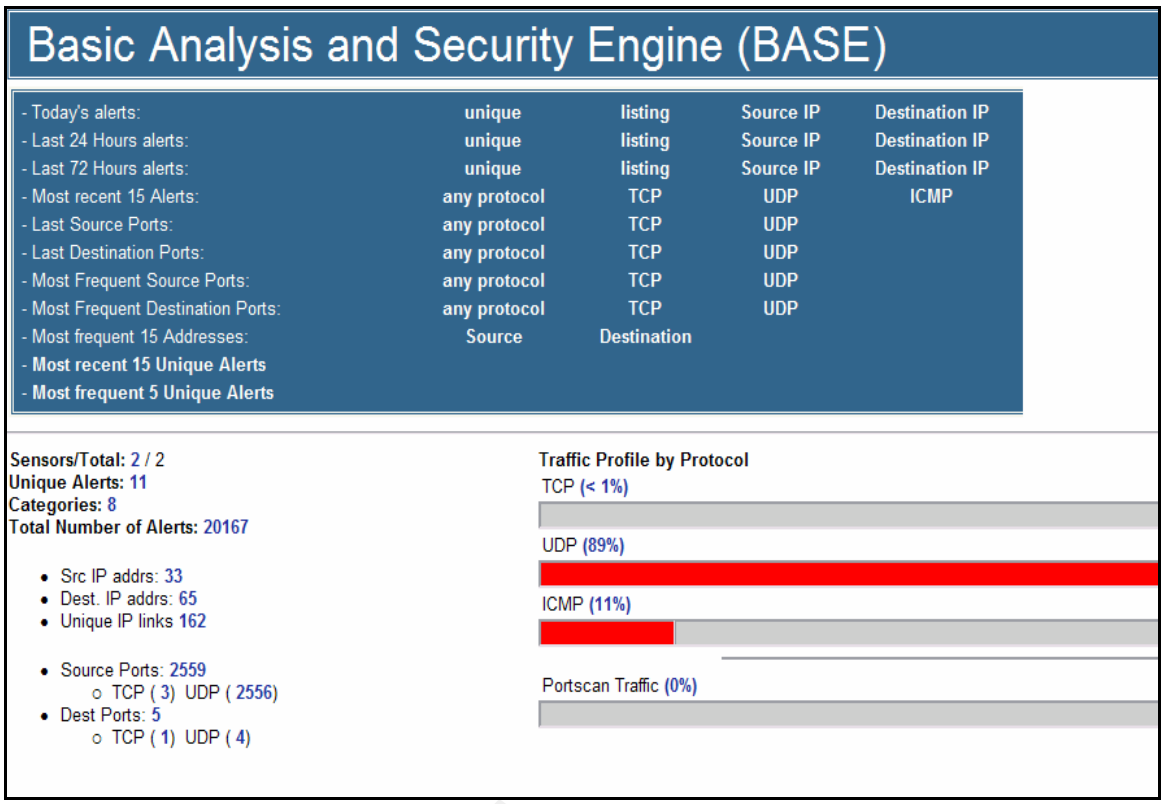

Figure 7: BASE Home Page

The SnortSam logging will be done locally on each system. However, the distributed IDS/IPS will forward the SnortSam blocking requests to all of the SnortSam FW Agents including the Central IDS/IPS. These forwarded blocking requests will then get logged locally on the Central IDS/IPS. With the central IDS/IPS the SnortSam logs will be written to an Apache data directory and made available for viewing via a browser.

In order to ensure the confidentiality of the log data only https is allowed and the Apache basic authentication is used.

3.9. System Configurations

The following tables provide system level information about the Central IDS/IPS and Distributed IDS/IPS. This information is generic and not meant to provide implementation specific details but can be used to develop specific implementation documentation.

Hardware				
System	**CPU**	**RAM**	**Hard Drive**	**Network Interface Cards**
Central IDS/IPS	Pentium 4 Class or compatible equivalent	256 MB	16 GB	2

Distributed IDS/IPS	Pentium 4 Class or compatible equivalent	256 MB	16 GB	3

Table 1: Hardware Implementation

Software							
System	**OS**	**IDS**	**IPS**	**DB**	**Web**	**Appl**	**Other**
Central IDS/IPS	Fedora Core 4	Snort 2.4.4	SnortSam 2.5	MySQL 4.1	Apache 2.0	BASE 1.2	ADODB 4.6.2
Distributed IDS/IPS	Fedora Core 4	Snort 2.4.4	SnortSam 2.5	MySQL 4.1	N/A	N/A	N/A

Table 2: Software Implementation

Network Interfaces				
System	**ETH0**	**ETH1**	**ETH2**	**Ports Opened**
Central IDS/IPS	IP Address - Yes Promiscuous – Yes	N/A	N/A	TCP:22 - SSH TCP:3306 - MySQL TCP:898 - SnortSam TCP:443 – https
Distributed IDS/IPS	IP Address - Yes Promiscuous – No	IP Address - No Promiscuous – Yes	N/A	TCP:22 - SSH

Table 3: Network Interface Implementation

For the servers requiring protection the implementation details will vary greatly. The only requirements for these systems are the use of IPTables and the installation of the SnortSam Firewall agent which is the same agent that is running on the IDS/IPS systems.

3.10. IDS & IPS Configuration

The configuration of the IDS and IPS components are required to ensure the IPS can effectively protect the specific network and server environment where it is running. The following steps are high level and meant to provide an approach for configuring the IDS and IPS components. More specific details regarding how to specifically implement these customizations can be found in the product documentation.

The first step with the Snort configuration is to decide which type of attacks should be monitored by the IDS. The Snort IDS comes with a full set of rules containing the attack signatures for all known attacks as well as other types of traffic an administrator may want to monitor. The rules are contained in ".rules" files that are specific to a type of attack. For example, "scan.rules", "ftp.rules", "telnet.rules", "dos.rules" to name just a few. The "snort.conf" tells Snort which rules files to use at startup time. The default "snort.conf " file includes the rules and the associated pre-processor statements for most major attacks and is the configuration used for the IIT Rice campus implementation. The rule set also includes signatures that allow administrators to monitor for specific network policy rules. For example, there are rules that will generate alerts for chat, port, p2p, multimedia, etc. The policy rules are not enabled by default.

The second step is to determine what attacks should be blocked by SnortSam when detected by Snort. The SnortSam IPS blocking requests are called by adding a "fwsam" statement to the individual Snort rules. See section 3.1 for examples of Snort rules with the fwsam statement added. One option is to add the "fwsam" statements to all the enabled Snort rules. However, it is recommended that you only add the "fwsam"

statements that coincide with the servers you're trying to protect. For the IIT Rice implementation we added the "fwsam" statement to each rule within the following Snort rules files:

- web-attacks.rules
- scan.rules
- dos.rules
- icmp.rules
- mysql.rules

The third step is to determine what hosts you want to protect on your network. As shown in the flow diagrams earlier in the document the Distributed IDS/IPS will send the blocking requests to these servers. This is enabled through "forward" statements in the Snortsam.conf file on the Distributed IDS/IPS. Each host to be protected will require a "forward" statement that includes the IP address of the server and the encryption password to be used between the Distributed IDS/IPS and the server.

The last step is to compile a "white list" of server IP addresses that should not be blocked. The white list is implemented in the snortsam.conf file and tells SnortSam to never block certain source ip addresses.

> This adds the host or network to the white-list of hosts/networks that will never be blocked.
> Blocking request for hosts on this list are ignored.
> Examples:
>
> dontblock a.root-servers.net
> dontblock 192.168.10.0/24 [Knobbe, 2006c]

This is important to ensure that SnortSam does not send blocking requests that could block legitimate traffic. The "white list" should contain the IP addresses of campus servers like DNS and possibly any administrator PC's. The list should always include the addresses of the Distributed IDS/IPS and the Central IDS/IPS. For example, consider a situation where the hacker has gained knowledge about the IPS system and spoofs the IP addresses of the Distributed IDS/IPS. Without the "white list" the Distributed IDS/IPS would send a blocking request to all the hosts it's protecting and the Central IDS/IPS telling these systems to block its IP Address. Once this occurs the IPS system is effectively down no other blocking requests will be accepted by the systems until the original block request expires. At this point the hacker can change his IP address and proceed with his attack.

The white list is a weakness. If an attacker spoofs the IP from the white list then SnortSam will not attempt to block the attack, however the IDS would log the attack. The only defense in this situation is to monitor the Snort database log and search for white list IP's.

3.11. IDS Rule Tuning & Thresholds

After implementation you can observe the types of alerts being sensed on your network and start tuning the snort rules. When the system fist comes online you may notice particular alerts occurring repeatedly. From the BASE console you can click on the alerts to learn more about them. Some alerts may be benign like a multicast "bad traffic" alert and if occurring often these alerts can make it difficult to spot more malicious alerts. When this occurs you can tune the Snort rules and set thresholds for these alerts instead of disabling the alert altogether.

Thresholds are enabled in Snort through the threshold.conf file by defining the thresholds for particular alerts. You then configure Snort to use threshold.conf by editing the snort.conf file. Following is an example threshold for an alert from threshold.conf.

threshold gen_id 1, sig_id 2189, type limit, track by_src, count 1, seconds 3600

This tells Snort to only log the alert with the signature ID 2189 every 3600 seconds or once an hour.

4. Initial Implementation Observations

The system design described in this document has been implemented at IIT's Rice Campus and as of June 1st 2006 the system has been running in a production mode for one month. During this 1st month the system has logged approximately 250,000 Snort IDS alerts and has issued 237 SnortSam IPS blocking requests. The following table provides a summary of the types of network traffic that triggered the IPS blocking requests.

Snort Attack Signature ID	Attack Description	Total number of blocking requests	Number of blocking requests (unique source address)
1330	Attempted wget command access via web	18	6
1331	Attempted uname command access via web	1	1
1333	Attempted id command access via web	19	5
1344	Attempted cc command access via web	1	1
1356	Attempted perl access via web	8	7
1365	Attempted rm command access via web	19	19
1367	Execution of a "mail" command attempted via HTTP.	19	11
1408	A TCP packet having a large payload was detected. Possible DOS MSDTC Attack	8	1
1444	TFTP Get	76	1
1774	WEB-PHP bb_smilies.php access attempt to exploit a known vulnerability	4	2
1917	SCAN UPnP service discover attempt	27	5
2329	MS-SQL probe response overflow attempt - exploit known vulnerability	9	1
2523	DOS BGP spoofed connection reset attempt	6	6
3456	MYSQL 4.0 root login attempt	22	4
		237	70

Table 4: SnortSam IPS Activity Summary

5. Conclusion

This design and its successful implementation have shown that you can build an effective IPS without incurring any software license or maintenance fees. The design described in this document is suitable for small to medium networks and will provide protection for Linux hosts running IPTables, but could easily be modified to suit specific implementations. For example, the SnortSam IPS can also be configured to work with network firewall agents to provide protection at a network level. Also, the design can scale to support more complex network environments by adding additional Distributed IDS/IPS systems.

References

Bace, Rebecca and Mell, Peter. Intrusion Detection Systems. [online] http://csrc.nist.gov 17 August 2001. [Cited November 2005]. Available from: http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf

Knobbe, Frank. SnortSam Web Site. [online] http://www.snortsam.net 2006. [Cited June 2006]. Available from: http://www.snortsam.net/index.html and http://www.snortsam.net/files/snortsam/docs/README.conf

Last Updated: June 20th, 2016

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Salt Lake City 2016	Salt Lake City, UTUS	Jun 27, 2016 - Jul 02, 2016	Live Event
SANS Cyber Defence Canberra 2016	Canberra, AU	Jun 27, 2016 - Jul 09, 2016	Live Event
MGT433 at SANS London Summer 2016	London, GB	Jul 07, 2016 - Jul 08, 2016	Live Event
SANS London Summer 2016	London, GB	Jul 09, 2016 - Jul 18, 2016	Live Event
SANS Rocky Mountain 2016	Denver, COUS	Jul 11, 2016 - Jul 16, 2016	Live Event
SANS Delhi 2016	Delhi, IN	Jul 18, 2016 - Jul 30, 2016	Live Event
SANS San Antonio 2016	San Antonio, TXUS	Jul 18, 2016 - Jul 23, 2016	Live Event
SANS Minneapolis 2016	Minneapolis, MNUS	Jul 18, 2016 - Jul 23, 2016	Live Event
SANS San Jose 2016	San Jose, CAUS	Jul 25, 2016 - Jul 30, 2016	Live Event
Industrial Control Systems Security Training	Houston, TXUS	Jul 25, 2016 - Jul 30, 2016	Live Event
SANS Vienna	Vienna, AT	Aug 01, 2016 - Aug 06, 2016	Live Event
SANS Boston 2016	Boston, MAUS	Aug 01, 2016 - Aug 06, 2016	Live Event
Security Awareness Summit & Training	San Francisco, CAUS	Aug 01, 2016 - Aug 10, 2016	Live Event
DEV531: Defending Mobile Apps	San Francisco, CAUS	Aug 08, 2016 - Aug 09, 2016	Live Event
SANS Portland 2016	Portland, ORUS	Aug 08, 2016 - Aug 13, 2016	Live Event
SANS Dallas 2016	Dallas, TXUS	Aug 08, 2016 - Aug 13, 2016	Live Event
DEV534: Secure DevOps	San Francisco, CAUS	Aug 10, 2016 - Aug 11, 2016	Live Event
Data Breach Summit	Chicago, ILUS	Aug 18, 2016 - Aug 18, 2016	Live Event
SANS Alaska 2016	Anchorage, AKUS	Aug 22, 2016 - Aug 27, 2016	Live Event
SANS Bangalore 2016	Bangalore, IN	Aug 22, 2016 - Sep 03, 2016	Live Event
SANS Chicago 2016	Chicago, ILUS	Aug 22, 2016 - Aug 27, 2016	Live Event
SANS Virginia Beach 2016	Virginia Beach, VAUS	Aug 22, 2016 - Sep 02, 2016	Live Event
SANS Brussels Autumn 2016	Brussels, BE	Sep 05, 2016 - Sep 10, 2016	Live Event
SANS Adelaide 2016	Adelaide, AU	Sep 05, 2016 - Sep 10, 2016	Live Event
SANS Northern Virginia - Crystal City 2016	Crystal City, VAUS	Sep 06, 2016 - Sep 11, 2016	Live Event
SANS Network Security 2016	Las Vegas, NVUS	Sep 10, 2016 - Sep 19, 2016	Live Event
SANS London Autumn	London, GB	Sep 19, 2016 - Sep 24, 2016	Live Event
SANS ICS London 2016	London, GB	Sep 19, 2016 - Sep 25, 2016	Live Event
Digital Forensics & Incident Response Summit	OnlineTXUS	Jun 23, 2016 - Jun 30, 2016	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced