



Security & Oculan's OcuGuard Intrusion Detection Appliance:

A Technology Discussion

Oculan Corporation
May 2003

Rev. 2.0



Table of Contents

Introduction	3
Network Security Tools and Technologies	3
A Comprehensive Approach to Security	9
OcuGuard Technical Overview	9
Network Design and OcuGuard Deployment	12
Deploying Multiple OcuGuard Appliances	15
Integration	16
“Care and Feeding” of OcuGuards, Configurations, and Signatures	19
The Signature Update Process.....	20
Summary	21



Introduction

Security, to co-opt an old phrase, is not a destination, but a journey. And the journey is not a short one. All told, people have dedicated their lives to the pursuit of the amorphous “secure system” or “secure network”, only to find themselves constantly trying to stay one step ahead of the would-be intruder. This “journey” takes time, patience, and persistence, as well as a level of comfort with ambiguity. Security is about risk management—determining what is at stake, measuring both the likelihood and impacts of a potential loss, and addressing that possibility proactively. The quest for security does not occur in terms of black and white, but rather in shades of gray, and administrators will be forever tasked with determining which shades are acceptable and which are not.

This whitepaper will not attempt to make black and white from gray, but instead will focus on the processes and tools required to assemble and deploy a successful security solution. As with any technology-centric discussion of security, we'll spend a good amount of time discussing tools, but this does not preclude the necessity nor importance of having a well-defined policy in place. Once completed, this security policy will help you to define and fill the needs for specific tools addressing specific risks.

Network Security Tools and Technologies

There are three functional areas of security which must be addressed by any security policy: prevention, detection, and response.

- Prevention is the practice of making sure that access to data and resources are allowed only to those who are authorized, and is one of the more clear cut functional areas of security. Mechanisms such as system level authentication (e.g., passwords, secure tokens, biometrics, etc.) and network control (e.g., firewalls, etc.) fall into this category.
- Detection addresses the question: How do I know if my prevention tools haven't done what I had hoped or expected? It makes the basic assumption that complete disconnection from the world is not an option, and there is a need to sift the desired network traffic or resource requests from those that are undesirable. Virus scanners are the most prolific example of detection mechanisms, as email and applications are generally desirable, but those with destructive attachments or evil intent are not. Intrusion detection systems are one of the most rapidly emerging technologies in this category.
- Response is what happens after an incident is detected. This functional area spans both computer and human response, and if addressed thoroughly, will deal as much in process definitions as tool selections. Depending on the incident at hand, valid responses could range from ignoring the attack to immediately disabling a system to calling the FBI. Many virus scanners provide some response mechanisms, but usually, it's up to a human to make a conscious decision as to how to respond.

Virus detection tools are the most mature of available security products simply because they have been around for the longest time. Virus scanners usually have a component that sweeps



a filesystem, opening files and comparing them to a set of character or behavior patterns that uniquely identify a virus. These signatures can range from a simple text string match to a set of behaviors that are known to be malicious. Many virus scanners also have the ability to work by interception. This interception can either be in the form of a plug-in for an application that scans data before the application reads it, a network level proxy that downloads and scans mail for a mail reader, or at the operating system level by intercepting system calls to open a file. Upon detection of a virus, most products will notify the party or parties responsible for a system through a pop-up window, e-mail, or some other form of notification. Many will also block access to an infected file (thus the importance of scanning the data before the application sees it), some simply rename it, or remove it, while others will attempt to repair the file, removing the destructive components and restoring the file to its original, unaltered state.

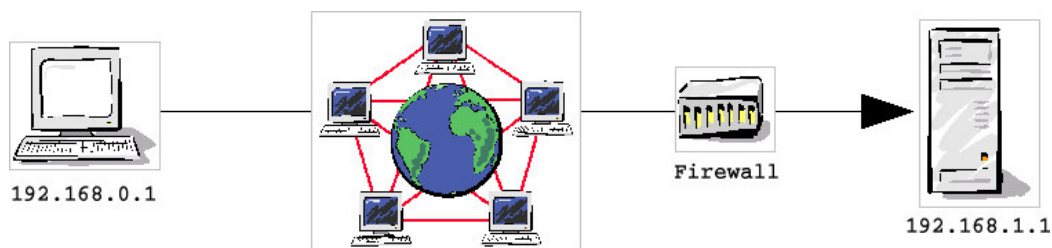
While most of us may have heard of firewalls, we don't often stop to think of just how inaccurate the name firewall really is. Firewalls are far from walls—their sole reason for existence is to let traffic through. They typically sit between some set of assets that require protection (usually an internal network with critical resources) and a network access point attached to a network and resources which are outside our zone of control (usually, the Internet). But the firewall's responsibility is not to keep traffic out, but rather to let select traffic in. Whether you approach firewall configuration by assuming that all traffic is desired and selectively eliminating traffic, or by assuming that all traffic is undesired and selectively allowing traffic, the sum remains the same: You are choosing to allow external traffic onto your protected internal network.

There are two main types of firewall, a hybrid of those types, and yet another type that has not seen much use. The first type of firewall is what's called a packet filter. A packet filter reads the IP headers in a packet to decide if the packet will be allowed to pass. The second type, a proxy firewall, actually reads a packet's payload to decide if it will forward the packet. Hybrid firewalls, our third type, combine the two technologies in an effort to get the best of both worlds: the speed of a packet filter combined with the security of a proxy. A final type of firewall that holds much promise detects anomalous traffic. These haven't seen much deployment as anomaly detection is difficult to implement and its processor demands can often affect performance.

The decisions made by a packet filter are usually based on one or more of six attributes: source IP address, source port, destination IP address and destination port, and sometimes packet length and packet frequency. A packet filter rule is usually comprised of a statement to either allow or deny the passing of packets which match some set of attributes. For example:

```
allow 192.168.0.0/24 port any outside -> 192.168.1.1/32 port 80 inside
```

This translates to: the 192.168.0.0/24 network is allowed access to the server 192.168.1.1 on port 80. If a packet arrives on the firewall's external (usually Internet facing) interface from the 192.168.0.0 network and is headed for 192.168.1.1 on port 80, it will be forwarded through the firewall and passed along on the inside network. Note that the contents of the packet are not checked, only the packet's header fields.



Graphical view of above packet filter rule

In contrast, proxy firewalls examine the contents of a packet before making the decision as to whether to forward it or discard the packet and log the attempt. To be able to read packet contents, the firewall needs to understand the relevant protocol and packet structure. That is, it must understand how to read an application's request for a web page or understand e-mail headers, hence the term 'application proxy'. A proxy will listen on a certain port on the external interface of the firewall. These ports are well-defined for most applications (e.g., ftp is port 21, smtp e-mail is port 25, http is port 80, etc.), so the proxy is bound to a particular port determined by which corresponding application it supports. As packets arrive, they are assembled and passed up the IP stack to the proxy, until the proxy receives the full request. The proxy then parses the request, checks it against a set of rules, and makes its decision. A 'rule' usually looks like "allow a request that matches this pattern to pass". Take for instance:

```
allow http://www.oculan.com/*
deny http://intranet.oculan.com/*
```

Since the proxy is associated with a specific application, it knows to listen on a particular interface and port, hence the simplicity of the rule (the interface and the port number are defined by the application, in this case HTTP on port 80). So, a web browser sending a request for `http://www.oculan.com/index.html` would be allowed through, but one for `http://intranet.oculan.com/index.html` would be denied. As a result of the proxy acting at the application level, the system on which the proxy runs first has to assemble the complete request, potentially across multiple packets and analyzing each of the packets along the way, and then analyzing the data contained within the packet to make its decision. Accordingly, proxy firewalls are much slower than packet filters.

Obviously, a proxy firewall can only function if it understands the application and corresponding protocols. Due to the sheer number of applications out there today, most proxy firewalls are configured to also function as packet filters when a proxy is not available. A hybrid also has the ability to make rules more tightly focused. For instance:

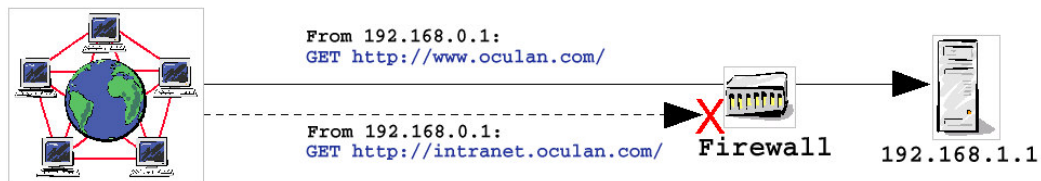
```
allow 192.168.0.0/24 port any outside ->
    192.168.1.1/32 port 80 inside http://www.oculan.com/*

deny 192.168.0.0/24 port any outside ->
    192.168.1.1/32 port 80 inside http://intranet.oculan.com/*

allow 192.168.2.0/24 port any outside ->
    192.168.1.1/32 port 80 inside http://intranet.oculan.com/*
```



So, a request from 192.168.0.0/24 to 192.168.1.1 on port 80 for `http://www.oculan.com/index.html` would be allowed through, but a request for `http://intranet.oculan.com/index.html` from the same network would be denied. The same request from the 192.168.2.0 network, however, would be specifically allowed by the final rule. Keywords such as `any` or `all` are often supported to allow for default behaviors.



Graphical view of above proxy rules

A new and promising firewall technology that is not yet widely deployed is based on anomaly prevention. There are two typical approaches: watching a network for a period of time and identifying a profile of “normal” behavior, or baseline, or by defining a set of behaviors that are always anomalous. The former takes some time to deploy, as it cannot simply be plugged into the network and be fully functional. And since most networks leverage similar technologies from a small subset of vendors, the latter is usually practical and an easily deployable solution. Examples of data that would always be anomalous would be things like an e-mail address that's four thousand characters long or an HTTP GET request to port 12345. Rather than build an entire firewall based on anomaly prevention alone, many vendors are working on adding this as an enhancement to packet filters or proxies.

Intrusion detection is a comparatively new field of technology. Intrusion Detection Systems (IDS), depending on the type, might include the packet analyzing power of a firewall, the pattern matching of virus scanners, or the intelligence promised by anomaly analysis engines.

There are two main types of intrusion detection technologies. One, a Host Based Intrusion Detection System (HIDS), resides on a host and watches for actions that seem suspicious. This could include monitoring a file or set of files for a change, attempting to access resources that are not allowed, deleting system files, or any number of other questionable occurrences. A Network-based Intrusion Detection System (NIDS) sits on the network, watching the traffic as it flows by. The NIDS will pick the packets up off the wire and examine them, comparing against either a set of signatures or a set of behaviors.

HIDS is almost always a piece of software installed on a system. Quite often it's installed by an end user, although many businesses are beginning to regularly deploy HIDS on their servers as well as their corporate desktop systems. This software usually has the ability to watch for file modifications, parse log files, or monitor system calls.

To keep track of file modifications, a HIDS will create a database of fingerprints of important files (e.g., `/etc/passwd` or the Windows registry) and will then sweep through the



system on some periodic interval, comparing the current fingerprint of the file against the fingerprint in the database. Some HIDS will even go so far as to keep backup copies and restore the original in the event of a change. Obviously, this cannot be done in real time and can prove computationally expensive if large numbers of files are to be monitored. It should be noted that the effectiveness of this method depends heavily on the mechanisms for creating the fingerprints and the storage of those fingerprints. A weak algorithm would be to store nothing more than the name of the file, its expected size, and the modification time. The behavior of an executable can be drastically altered without changing its size and the modification time can be altered after the fact by making modifications to the filesystem. A more thorough method is to use a strong hashing algorithm to create the file fingerprint. With a strong hash algorithm, the fingerprints of two sets of data will never match unless the data sets are identical. Storage of these fingerprints can also pose an interesting dilemma: even the best method of creating a file fingerprint is useless if the fingerprint storage is vulnerable. Were an attacker able to alter this storage undetected, the fingerprint for the file they wish to modify (or replace) can easily be altered, thus allowing the file modification to go undetected. If the fingerprint storage is compromised, the HIDS is rendered ineffective.

The log file functionality is meant to take advantage of the fact that most important applications keep a log of their activities, especially errors and application warnings. By reading the logs and comparing against a set of patterns, the HIDS can recognize suspicious activity. Log file monitoring can be enhanced by altering (or configuring) applications to log more information, however this can also cause a significant drain on system resources (both storage space and I/O time), possibly slowing the performance of the application itself. Of course, as new applications come along, signatures for their log files must be created. A common solution is to create a set of patterns to ignore and alarm on the rest, meaning that if an application does something unexpected, the HIDS will not ignore it. This does come at a cost of triggering many false alarms.

System call monitoring tends to be much more precise than the other two methods, but also tends to require a large set of signatures to be truly effective. By watching system calls, the HIDS can see such things as access to a set of resources that should not be accessed, a behavior that is abnormal (e.g., an e-mail that attempts to automatically run an application), or possibly do some level of string matching (watching for some application that attempts to modify a certain value in the Windows registry). This string matching can accomplish many of the same functions as file integrity monitoring, however this usually happens in near real-time and with much more granularity (i.e., it can watch for a particular change, rather than simply looking for whether or not a change occurred).

In general, HIDS have a few problems. First, management of HIDS software and configurations across many systems in an environment is problematic at best. Most of the packages today have matured from software that was developed to handle intrusion detection on a single host, not the potential hundreds of machines on which they are deployed in corporate enterprises. Centralized configuration and event management is typically an afterthought, with many vendors charging large sums for this functionality. As the need for HIDS grows, vendors will begin to address the management issue. However, they can never get around a single, fundamental flaw: intrusion detection software can end up running on a host that's been compromised. If the host is compromised, it cannot be trusted, nor can the software that is supposed to alert when an intrusion occurs. The opportunities to alter log files, wipe signature databases, disable or replace the HIDS software itself are significant,



and in most cases, the risk is simply too great to accept. While HIDS can show a great deal of very detailed information about a host, it cannot stand alone.

Network intrusion detection systems take a different, more manageable approach. NIDS are configured to listen for data on the network, looking for potentially detrimental behavior. As with firewalls, there are two basic network intrusion detection approaches, and a hybrid of the two. One method is to compare the network traffic against a set of signatures (patterns), looking for matches. Another method is to compare traffic against a set of behaviors that are known to be unusual. Hybrids implement both technologies, but vendors usually select one method for primary detection and leverage only a subset of the other.

Signature matching systems (sometimes also referred to as pattern matching IDS) work exactly as their name implies. As traffic passes by the network interface of the device, it runs the packet through a set of signatures looking for a match. Once a match is found, an action is taken. This action can range from appending to a log file to paging an administrator to remotely manipulating firewall rules. Signatures usually are combinations of patterns in the headers and payload of the packet. Since signature-based NIDS have the entire packet available, a particular signature can range from generic to very specific. For instance, a rule could look for all packets from 192.168.3.5 to 192.168.8.10 on port 80 with the PSH and ACK flags set containing the strings 'GET' and '/etc/passwd'. Another rule could simply look for all connections from anywhere to anywhere on port 12345. Rules can even be so specific as to look for a certain IP sequence number. Signature matching NIDS are quite flexible, but suffer from the same core problem as virus scanners: if there is no defined signature for an attack, the attack will go unnoticed. For signature based NIDS to be effective, their signatures must be kept up to date with the latest information and signature definitions available.

The other main type of network intrusion detection is anomaly detection. Similar in functionality to the firewall technology of the same name, these NIDS can function in a couple ways. The device can be set up to 'learn' what is normal traffic in a given network and trigger on anything abnormal, or it can be given a set of behaviors which are always suspicious. The learning mode obviously takes a while and is also prone to false alarms. Simply installing the latest demo of a new application may cause a perceived deviation from the norm, thus triggering an alarm. The behavioral model is quite similar to signature based IDS, except it usually has some level of application awareness. A rule for this sort of NIDS could look something like 'alarm on an e-mail address more than five hundred characters long'. In the end, behavior based NIDS have the same issues as signature based systems and anomaly-based firewalls: they must be kept up to date, and they must have significant processing resources at their disposal.

Hybrid NIDS usually take the form of application aware signature based systems. Quite often, data is spread across multiple packets. A web request can be long enough to not fit in a normal size packet, especially if it's been crafted by an attacker. A signature based NIDS that's unable to reassemble the individual packets into a complete request will be unable to match a signature, thus the attacker slips by unnoticed. So, the NIDS must be smart enough to know how to reassemble the packets into a request before it attempts to pattern match. This requires the application awareness of an anomaly based IDS to be able to know where the request begins and ends.



NIDS are usually appliances or computers that have been heavily secured, making the system much more difficult for an intruder to compromise. Since an intruder is much less likely to be able to break into this system they must focus beyond it, increasing the likelihood that they will be detected and unable to cover their tracks. Added to this inherent level of security is the current practice of replicating some amount of data between the NIDS sensors and their management station(s). Of course, even the best IDS is useless if the database of signatures is vulnerable. If an attacker can alter the database undetected, then they can simply replace the signature of the file they wish to modify (or replace) with a signature that will match their new file. With this database change, the IDS will never know that a file has changed.

A Comprehensive Approach to Security

A **comprehensive security solution** consists of some mix of the above tools and an organizational security policy. While a virus scanner may protect a desktop system from a particular bit of malicious code, it cannot protect a web server from the hordes of would-be, automated attackers. An intrusion detection system will happily send up a flag when it spots unusual behavior, but it cannot prevent the behavior from occurring in the first place¹. A complete security plan must define the resources the organization considers of value and the degree of protection those resources merit. The solution will likely include some mix of virus detection, firewalls, and intrusion detection.

A common question is “I already have a firewall, why do I need an IDS?” Recalling the firewall information, it is important to understand that the majority of firewalls in use today are of the packet filter type. They only look at the headers, not the payload. Even proxy firewalls err on the side of permissiveness. A packet filtering firewall will let a request right through to a web server, without ever noticing that the packet contains 'GET /etc/passwd'. An IDS, on the other hand, will quickly trigger on this request, notifying an administrator. A proxy firewall would probably not allow the request through, but since a false alarm from a proxy firewall could potentially mean a loss of business traffic, proxies tend to be a bit more lenient than an IDS. A false alarm from an IDS is an annoyance--a false alarm from a firewall could mean lost revenue.

OcuGuard Technical Overview

Oculan's OcuGuard is a network intrusion detection system in appliance form. The core of the system, the detection engine, is a robust, signature-based NIDS that adopts critical features from application-aware systems. It works by monitoring network traffic for packets and attempting to match each packet against a set of conditions (e.g., a signature). If a match occurs, an alert is triggered.

An OcuGuard is essentially a sensor. It runs headless and has no direct user interface. An Oculan OpticNerve serves as the user interface for one or more OcuGuard appliances. When

¹ The industry is beginning to acknowledge an emerging set of *intrusion prevention* technologies, but these are very complex and costly technologies, fraught with all the problems associated with automatically shutting off network accesses based on what a computer *thinks* is appropriate. Oculan considers these technologies academically interesting, but not yet ready for “prime time”, especially in the SMB market.



an OcuGuard generates an alert, it is sent to the OpticNerve to be processed. The OpticNerve then decides if/how to notify the user of this event. There are four possibilities based on a configuration generated by the user: ignore the event, display the event in an event viewer, e-mail an administrator, or page an administrator. These are usually configured in such a way as to only page an administrator for the highest priority events.

Most attacks these days have a unique signature. It's often easy to look at a stream and realize that a request for cmd.exe is rarely of a business purpose. Likewise, the code to cause a buffer overflow, and execute something malicious in its aftermath usually does not change, either. The overall attack may change, but small, distinct pieces are still likely to be recognizable. The OcuGuard looks for a specific pattern based on the signatures of the attack code. Sometimes it's a matter of looking for a request, sometimes it is the response to a particular request that causes an alarm.

Many attacks against web servers are found simply by looking for a particular request in a URL. Quite often, web servers are exploited by tricking them into passing user specified arguments to an application, which then cannot handle the input. This can be things like bogus query strings that cause a CGI to run a command and return the results to the attacker, or it could be a request that is too long for the application to parse. The OcuGuard has a long list of signatures that can detect attempts at specific web server attacks, attacks of specific third party applications that run on many web servers, and also reconnaissance that is simply looking to see if a particular server is vulnerable, leaving it open to a subsequent attack.

Web servers are the most attacked processes these days, but DNS and FTP servers are not far behind. The OcuGuard has many signatures that detect reconnaissance and attacks against these systems and more. For example, *bind* (a popular, yet often vulnerable, DNS server) version queries often indicate an attacker making a list of exploitable systems to come back and attack later. There are also signatures that match against tools known for attempting to upload and/or hide illegal software on unsuspecting FTP servers.

Another hot button (though less so in the wake of self-propagating attack code – *worms*) is detection of distributed denial of service (DDoS) attacks. The tools for DDoS quite often have unique signatures both in the attacks they send and in the communication between the control stations (the masters) and the systems that actually do the attacking (zombies). The OcuGuard has the ability to detect both of these cases, thus alerting when an organization is undergoing a distributed denial of service attack as well as when an organization's assets are being used to attack a third party.

Up until now, the OcuGuard has been discussed as a signature based NIDS, while it's actually more of a hybrid. To fully meet the IDS needs of the small-to-midsized business, the OcuGuard must have some level of application awareness. Again, take the web request example. In order to perform a pattern match against a particular HTTP request, the sensor has to know what part of a packet (or packets) it should match against. While this still sounds like a signature match, one needs to understand that there are many ways to encode a HTTP request:

```
GET /www/scripts/..%255c../winnt/system32/cmd.exe?/c+dir
```

```
GET /www/scripts/..%252f../winnt/system32/cmd.exe?/c+dir
```



```
GET /www/scripts/..\%35%63../winnt/system32/cmd.exe?/c+dir  
GET /www/scripts/..\%35c../winnt/system32/cmd.exe?/c+dir  
GET /www/scripts/..\%25%35%63../winnt/system32/cmd.exe?/c+dir
```

In the first request, %255c decodes to \ in the Unicode character set. In the second, %252f decodes to /. The rest fall into the category of 'double decoding'. In the third example, %%35%63 first decodes to %5c, which then decodes to \. In the next, %%35c first decodes to %5c, which then becomes \. In the final example, %25%35%63 first becomes %5c (%25 decodes to '%') and again, this becomes %5c, our familiar backslash character from earlier in the example.

All of the above requests are taken from a server under attack from the Nimda worm. This worm attacks Microsoft Internet Information Server (IIS) web servers and exploits a vulnerability in the Unicode decoding. This exploit allows applications to be run that are outside of the server's document folder. So, on a vulnerable system, the above would allow the execution of cmd.exe and would print the contents of a directory. Without some level of understanding of the stream, this sort of encoding could slip by. As an example, a request that looks like the following:

```
GET/ww/..\%255c../winnt/system32/%63%6d%64%2e%65%78%65%253F%252f%63%2b%64%69%72
```

Once it reaches the webserver and is decoded, it looks strangely familiar:

```
GET /www/..\../winnt/system32/cmd.exe?/c+dir
```

To be successful in detecting this attack, a NIDS must have enough protocol awareness to be able to decode the request, so it can apply its signatures. Once the appliance has the capability of decoding such requests, a simple signature that looks for the string 'cmd.exe' can trigger an alarm. Oculan's OcuGuard leverages this level of packet "intelligence", decoding the request before applying signatures. It should be noted that a simple signature could be written for the above case, but there are thousands of possible combinations of encoded and unencoded characters.

Another area in which the OcuGuard deviates from strict signature analysis is in the area of portscans. In order to detect a portscan, some level of state must be kept across multiple connections in order to group them into a single event. It should be noted that specific packets can trigger a portscan alert simply because those packets are usually only generated by portscan tools (specifically ones that set odd flags and attributes in an effort to discern the operating system of the scanned host).

Once an attack has been detected, the OcuGuard will send a message to the OpticNerve with all the relevant information. The OpticNerve is configured to react to the incoming events based on their priority. The end user is provided with an interface that allows them to determine the reaction to an event when it arrives.



While on the surface, paging an administrator seems to be an effective mechanism to communicate critical system information, one must be careful not to over-communicate. The typical response to a flood of pager messages is either to turn off the pager, or to turn off the device that generates the events, which in this case is the OcuGuard. Obviously, this leads to a much lessened level of security, as events can end up being ignored. This is an area in which the integration with the OpticNerve is quite useful. The OpticNerve has capabilities in place to prevent a flood of messages being sent to a pager, while preserving enough information to be useful. The OpticNerve can provide both singular event information, or in the case of some concentration of events in a short time span, aggregate events can be generated. This reduces the overall number of pages, while keeping the critical information readily available to the administrator.

Network Design and OcuGuard Deployment

Sensor location is very important with a NIDS and Oculan's OcuGuard is no different. An OcuGuard can only detect attacks in traffic it can see. OcuGuard appliances watch for traffic at a layer 2 level (that is, they listen to traffic at an ethernet level), but watch for attacks at layer 3 (IP) and higher. Remember that ethernet is inherently a broadcast medium. In general, all traffic is sent to all systems. However, switches have fundamentally changed this behavior. The purpose of a switch is to make sure that a host receives only traffic that it is supposed to receive. While helpful to increase available bandwidth, this makes it difficult for an OcuGuard to see all the traffic.

The OcuGuard is what's referred to as a promiscuous listener. When set on a network, it watches all traffic that passes, not just traffic destined specifically for the OcuGuard itself.

As an aside, all OcuGuard appliances have two interfaces (a third, currently unused, is reserved for future development). One is used for communications with the OpticNerve. The OcuGuard, upon initial network configuration, will establish a connection to the OpticNerve and set up an encrypted channel for sending events (updates are downloaded via a slightly different method that relies on the same protocol, but establishes a connection "on-demand"). The encrypted channel mandates that the OcuGuard must authenticate itself to the OpticNerve, and that the OcuGuard will verify that the OpticNerve it is communicating with is in fact the system with which it should be communicating. Oculan has implemented this channel using SSHv2 (Secure Shell version 2).

The listening, "promiscuous" interface of the OcuGuard is plugged into a network where it will have the most visibility into network traffic. Again, an OcuGuard cannot detect attacks in traffic that it cannot see. This interface has no IP address and no IP protocol stack. It cannot be directly addressed (while the management interface can be addressed and should be placed on a protected network), making the system more secure and making it much more difficult for an attacker to either detect or attack an OcuGuard.

As previously mentioned, LAN switches make it difficult for an OcuGuard to see all the traffic that one might be interested in monitoring. There are various ways to address the problem at hand. The most common method leverages the fact that most manageable switches have the ability to reflect traffic from one port to another, effectively duplicating the traffic from one layer 2 network to another. Cisco calls this *SPAN*, other vendors use terminology like traffic reflection or port mirroring. Another common option is to plug the



OcuGuard and the device to be monitored into a shared media device (e.g., a hub). This device would then need to be uplinked to another network device to provide connectivity for the remainder of the business. While this option may be appealing for cost reasons, it can introduce bandwidth limitations, specifically the 4 Mbps effective throughput of shared media, 10Mbps ethernet. However, it is common to deploy the OcuGuard directly inside a firewalled Internet connection in which the limiting factor for bandwidth is introduced by the WAN link itself, usually at 1.5 Mbps or less.

What happens when there's more than one LAN segment? There are two options: one is to have multiple OcuGuard appliances, one for each segment, or alternatively, you can opt is to monitor at choke points. Careful thought needs to be given to this decision as neither is perfect for every environment. However, Oculan's recommended "80% solution" is to approach network choke points first, then expand as necessary.

For maximum visibility, it's hard to beat one OcuGuard per Layer 2 LAN segment, with each reporting back to a central OpticNerve. Since an OcuGuard can only alert on traffic that it can see, with one OcuGuard per LAN segment, visibility is provided into all of the segments, and thus all the traffic. An example where this might be useful is if an organization were concerned more with internal attacks than external attacks. If an OcuGuard were placed only in the LAN segment which hosts the main servers, any attacks from one client network to another would go unnoticed. Another likely example in today's distributed networks with centralized management would be a company that has remote facilities (perhaps subsidiaries or other divisions, perhaps branch offices). Many of these organizations have servers and/or client systems with important information at these remote locations. An OcuGuard located at the central facility will provide no visibility into attacks that occur within a remote facility. If an organization desires to monitor for attacks within a remote facility, then an OcuGuard should be installed at that remote facility.

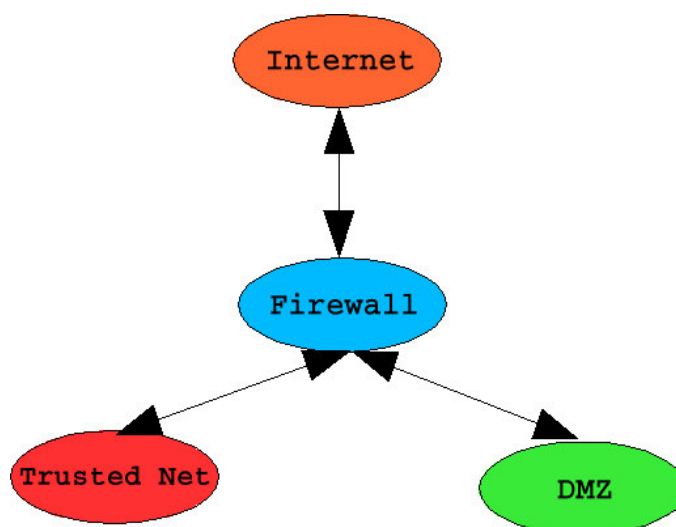
Since Oculan's architecture supports multiple OcuGuard appliances reporting back to a central OpticNerve, all remote OcuGuards can be managed from one interface. A side benefit to having one OcuGuard per LAN segment is that each OcuGuard can be individually tuned for the specific environment. If a segment has no Linux-based systems, then the Linux specific signatures can be disabled for only that specific segment, while leaving them on for a server segment. This helps reduce the number of false alarms an administrator needs to deal with.

It is likely, however, that it may not be feasible to install an OcuGuard for each physical segment, usually due to budgetary constraints. In those cases, a decision must be made as to where to place the OcuGuard(s) to achieve maximum visibility, and in turn, maximum return on investment. This again is network architecture dependent. A common approaches leverages an OcuGuard at each choke point. If the OcuGuard were placed between a LAN and an Internet connection, all attacks originating from the Internet would then be detected, but a client computer attacking a computer on a different internal network would not be detected. In the network case where all clients are on one segment and servers are on another segment, the choke point is the router between the servers and clients. An OcuGuard could be deployed such that it watches all the traffic between the router and servers. Thus, all attacks from the clients against the servers would be detected.



Note: if the servers are all connected to the same switch, it might be tempting to plug the OcuGuard into the switch and mirror all traffic to all the servers to the OcuGuard port. While seemingly a good idea, it's doomed to failure. It's possible, if not likely, for twenty servers to be heavily taxed at the same time, possibly saturating their own individual links. That's potentially four gigabits per second, assuming that systems are in full duplex mode. The maximum bandwidth available to the port in which the OcuGuard is plugged is 200 Mbps (100Mbps Full Duplex). Four gigabits of data will not fit down a 100Mbps link, so packets will inevitably be dropped at the switch. However, if you mirror traffic from the router's switch port only you avoid overloading both the switch (which is now taxed by additional mirroring requests) and the OcuGuard. By deploying the OcuGuard in concert with the network design, leveraging the existence of choke points, these situations can be avoided.

In most cases, a network will be configured as noted in the following diagram:



Obviously, the ports on the firewall will serve as choke points, throttling higher speed internal traffic down to the speed of the Internet connection, which in the case of most small-to-midsized businesses is usually capped at DS-1 speeds (1.544 Mbps)². If the OcuGuard is deployed outside the firewall, the administrator will be inundated with security events to address. While the event load may be intellectually interesting, it's quite often unmanageable, not to mention that it's simply not good design. By deploying outside the firewall, you are undoing the benefit of using a firewall to process and reduce the amount of traffic that will be allowed in your network. And you have gained little beyond that — the same threats will continue to be recognized, but they'll likely be buried in a mass of portscans and other threats which otherwise may have been blocked by the firewall, obscuring that which is truly a risk to your internal infrastructure. By sacrificing this

² As more and more businesses move away from dedicated connections and frame relay-based Internet access and toward xDSL and Cable Modems, 1.544Mbps is still a reasonable rule-of-thumb speed to assume for Internet access. The OcuGuard appliance is engineered to handle well beyond these bandwidth demands.



information (albeit interesting), you allow the firewall to do its job and in turn, monitoring traffic after the firewall provides a much smaller subset of information that must be handled—by both you and the OcuGuard appliance. If your firewall provides a logging facility (as most today do), it would not take much effort to correlate the firewall logs with the OcuGuard events to get a more clear picture of an attack. Most of the time, in the above configuration, the firewall does not allow hosts on the Internet to initiate a connection with hosts on the trusted net. Many organizations will place all of their client systems and any servers that will never be publicly visible in the trusted network, while placing systems that often require externally-initiated connectivity, such as web, mail, FTP, DNS servers in the DMZ. The firewall is configured to allow connections to a restricted set of services on the systems in the DMZ (for instance, only allowing port 80 traffic to the web server, but not the other systems in the DMZ). So, if one wanted only to monitor for attacks from the Internet, a single OcuGuard in the DMZ would suffice. This could also monitor for attacks from the trusted network against the servers in the DMZ. And it leverages the firewall's ability to process and filter traffic. However, there are many signatures in the OcuGuard that would detect if a system on the network were trying to communicate with (or respond to) an external, malicious host, as is common in the event that a *backdoor* or a distributed denial of service (DDoS) tool were installed. If there were only one OcuGuard in the above network, and it was deployed in the DMZ, these events would not be detected. While relying on the firewall for blocking traffic between the Internet and the trusted net is a solid design decision, it also assumes that the firewall's Internet connection is the trusted net's only path to the Internet, or worse yet, the Internet's only path to the trusted net. Multiple firewalls, perhaps by an extranet, an internal WAN, or direct connections to other networks, will greatly complicate the deployment considerations and decisions associated with the OcuGuard. An organization should keep an eye on any networks or users outside their zone of control, even if they are trusted business partners, to make sure that they are not attacking (Thanks to email-distributed zombie code and rapidly proliferating worms, more and more organizations are unwittingly participating in DDoS attacks). To address these concerns, an OcuGuard would need to be placed at these other network access choke points as well.

Obviously, placing OcuGuards only at choke points is less expensive, but it does trade off the savings versus visibility. The actual physical deployment of OcuGuards is a critical component in maximizing their effectiveness, and in many cases, can help to justify a network re-design effort. It would be much better to have one OcuGuard on each physical network, but these decisions should be made based on a security plan which analyzes the risks and justifies the funding to address them. The OcuGuards, coupled with the OpticNerve, provide the sorts of tools and features the small-to-midsized business needs to effectively and efficiently maintain a networked infrastructure. Augmented with the signature-update service (included as part of the support contract), OpticNerve and OcuGuard together provide an extremely cost-effective way to address both network/systems management needs, as well as intrusion detection, without adding additional staff (or further burdening existing staff) to provide the “care and feeding” mandated by a product-only solution.

Deploying Multiple OcuGuard Appliances

We have already discussed multiple OcuGuard deployments, but how does it really work in the real world? Remember, the system includes at least one OcuGuard sensor appliance reporting to one OpticNerve management appliance. The OpticNerve is Oculan's central



appliance, providing network management capabilities, event logging, user interface, reporting, and integration with notification systems, as well as management of Oculan's other appliances.

As the integration point in the Oculan offering, the OpticNerve is positioned to be the one point in the infrastructure where all other services and appliance, including the OcuGuard, will report its operational state, configuration, and event information. All OcuGuards in an organization are normally configured to report their events back to a centralized OpticNerve so that in the case of multiple OcuGuards, an operator need reference only one web interface to drill into security events across an entire enterprise.

The OpticNerve also provides the user interface for configuration of the OcuGuard(s) in an organization. There are two basic ways to configure an OcuGuard through the GUI. One method is to configure a particular appliance with the exact settings desired, creating one complete configuration per OcuGuard in an organization. Another method involves using template configurations. With each configuration created, it can be quickly applied to, and if need be, modified for other appliances. For instance, one could create a configuration for a Windows network, and then use that same configuration for each other OcuGuard that monitors similar Windows-based networks throughout an organization. Since each configuration automatically becomes a template, it can be reused at any time to configure any other OcuGuards. As a result of this central interface for viewing events from multiple OcuGuards, as well as a tool for their configuration, the workload for multiple sensors is greatly reduced.

Integration

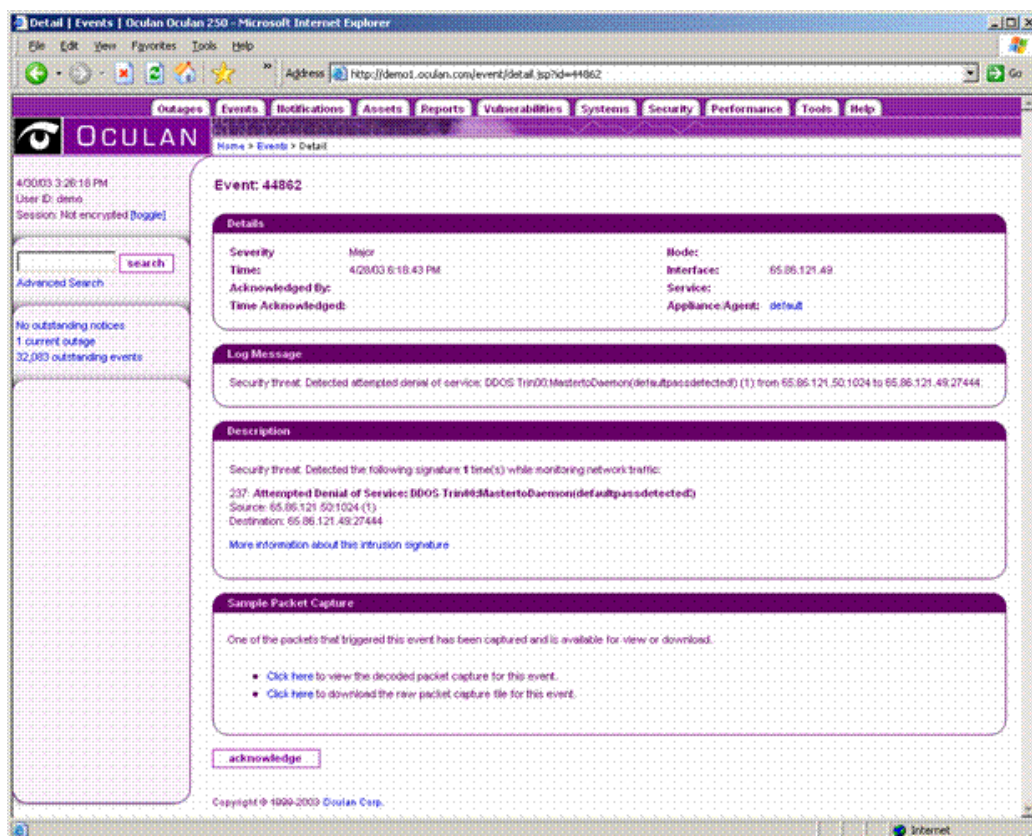
Data presentation is perhaps the most important features of a NIDS, or for that matter, any security product. If the information is not presented in a way that makes it quick and easy to comprehend and understand, then it might as well not be there at all. The OpticNerve's event browser was designed in a way to provide all event information in a way that an operator can easily understand, and further, can easily manipulate to provide a focused view of just the data needed for the task at hand. In the screen capture below, you see the standard format of events in the OpticNerve's event browser.



ID	Severity	Title	Device	Interface	Service	App	Ack
47153	Major	4/25/03 9:11:55 PM		65.85.121.49		default	<input type="checkbox"/>
Security threat: Detected attempted denial of service: DDOS Trind00 Master00/Owner00/default/pass/affected00(1) from 65.86.121.50:1034 to 65.86.121.49:27444							
47153	Major	4/25/03 9:11:41 PM		65.85.121.49		default	<input type="checkbox"/>
Security threat: Detected attempted denial of service: DDOS shell handler to agent (1) from 65.86.121.50:1024 to 65.86.121.49:18755							
47153	Major	4/25/03 9:11:25 PM		65.85.121.49		default	<input type="checkbox"/>
Security threat: Detected attempted denial of service: DDOS mtstream handler ping to agent (1) from 65.86.121.50:65835 to 65.86.121.49:10498							
47142	Major	4/25/03 9:11:50 PM		65.85.121.49		default	<input type="checkbox"/>
Security threat: Detected attempted denial of service: DDOS Stackelshant client-check-gag (1) from 192.168.0.78 to 65.85.121.49							
44028	Major	4/25/03 6:36:33 AM		65.85.121.50		default	<input type="checkbox"/>
Security threat: Detected attempted denial of service: DDOS MSDTC attempt (5) from 144.9.72.134:443 to 65.85.121.50:3372							
44892	Major	4/25/03 6:18:47 PM		65.85.121.49		default	<input type="checkbox"/>
Security threat: Detected attempted denial of service: DDOS Trind00 Master00/Owner00/default/pass/affected00(1) from 65.86.121.50:1034 to 65.86.121.49:27444							
44893	Major	4/25/03 6:18:30 PM		65.85.121.49		default	<input type="checkbox"/>
Security threat: Detected attempted denial of service: DDOS shell handler to agent (1) from 65.86.121.50:1024 to 65.86.121.49:18755							
44893	Major	4/25/03 6:18:21 PM		65.85.121.49		default	<input type="checkbox"/>
Security threat: Detected attempted denial of service: DDOS mtstream handler ping to agent (1) from 65.86.121.50:65835 to 65.86.121.49:10498							
44892	Major	4/25/03 6:18:40 PM		65.85.121.49		default	<input type="checkbox"/>
Security threat: Detected attempted denial of service: DDOS Stackelshant client-check-gag (1) from 192.168.0.78 to 65.85.121.49							
44893	Major	4/25/03 3:32:28 PM		65.85.121.50		default	<input type="checkbox"/>
Security threat: Detected attempted denial of service: DDOS MSDTC attempt (1) from 63.121.29.7:80 to 65.86.121.50:3372							

OpticNerve Event Browser Screen Capture

You can readily see that the information provided gives a fair amount of information, including the severity of the event, the date and time it was received, the address of the interface involved in the event, and a description of the alarm. However, there are several active navigation controls within the event itself. All event browsers are outfitted with Oculan's Query-by-Example technology, which when enabled, adds additional controls to the browser. These controls, in the form of "plus" and "minus" signs, allow you to refine the current view by adding a search criteria equivalent to the current value (the "plus" sign) or removing all occurrences of the current value (the "minus" sign). This functionality allows you to quickly and easily restrict the current view to include only that current value (e.g., only Critical events, only address 209.116.71.166, etc.) or exclude that current value (e.g., don't display any events from node 209.116.71.166). Additionally, the event id, the numeric identifier in the second column from the left, serves as a hot link to additional details regarding the event. In the case of event 7812, that link would take you to a page similar to one pictured in the screen capture below.



The OpticNerve's Event Detail View

The top portions of the detail screen provide the same information you saw within the event browser, allowing you to immediately understand the association between the event log and the detailed description being displayed here. Note also that the description provides not only a breakdown of the event type received and the four critical IP headers, but is augmented with hypertext references (hot links) to external sites which provide additional background information on the threat. This information usually includes critical details such as why the signature was triggered, what it means, and in many cases, how to resolve the issue (e.g., what service pack to download, what firewall rules to consider, etc.)

Additionally, the OcuGuard appliances retains the original network packet that generated the alert, passing it on to the OpticNerve where it is retained along with the threat information for either online display or download for further forensic or legal purposes.

The other interface to security events is through some notification mechanism. These are typically e-mail or pager alerts sent to the operator to inform them that one or more high priority events has occurred. These notifications exist as a cue to the operator so they know they need to check the events on the OpticNerve. Enough information is supplied in the notification to allow the operator decide the relative importance and time-sensitivity of responding to the event. The idea is to present the operator with enough information in their pager notification to decide if they should get out of bed at 4 AM, or if they can wait to go in at a respectable hour.



A common issue with systems that provide an interface to notification via pager/text message is that they send events as they receive them, as quickly as they can. In the case of security events, this could result in hundreds of pages within a very short timeframe. Many operators would simply turn off their pager after the first few, potentially missing important, subsequent notifications. The OpticNerve's notification system was designed to keep the operator from being flooded under a huge load of events, even if many OcuGuards are involved.

The OcuGuard is based on an asynchronous messaging mechanism, which allows it to be “event driven”. This means, when the OcuGuard needs to communicate with the OpticNerve, it sends an event in a native format that the OpticNerve immediately understands. Further, an OcuGuard only sends events as needed. If there has been nothing that requires alerting, the OcuGuard sends no events. Also, events are sent to the OpticNerve from the OcuGuard, rather than the OpticNerve polling the OcuGuard on some regular basis. This minimizes network traffic as communication only occurs as needed, not all the time.

“Care and Feeding” of OcuGuards, Configurations, and Signatures

Oculan's OcuGuard enjoys significant differentiation from other intrusion detection products on the market today. First, while other products mandate training for deployment and configuration, and in many cases, dedicated staff to support custom integration and operations, the OcuGuard does not. The administration of an OcuGuard was designed to be minimal, and with a rules-engine maintaining signatures on your behalf, there is little ongoing work necessary, save when there are major network changes in your environment. Additionally, the user interfaces were designed to be as intuitive and easy-to-use as possible—a feature that other vendors seem to have ignored. Leveraging the OcuGuard as an information source and the OpticNerve as a user interface (which provides event formatting for readability, comprehension, and dynamic links to external sites), the information provided is very comprehensible and typically, the user interface requires only a few minutes of “poking around” before the end-user is up and running. The design goal for the integration mandates the timely receipt of information that can be distilled for use by a network or systems administrator — not necessarily a security expert.

One distinct advantage of the OcuGuard is cost-effectiveness. The appliance is extremely cost-effective, especially as compared to its very expensive competition. In a market where it is not uncommon to have to purchase IDS sensors, administrative software, a reporting tool, and integration services, all with their own individual price tags, Oculan's comprehensive approach to delivering a solution instead of a suite of products immediately becomes appealing. And OcuGuards can quickly be added to, removed from, or moved within an organization's infrastructure.

And, under Oculan's support plan, the OcuGuard receives software updates³ as part of the package. Further, the updates can be automatically installed and configured, at the user's discretion. Similar to Microsoft's *Windows Update* functionality, you can configure your OpticNerve to automatically check if updates are available, automatically download those updates, or automatically download and install the updates.

³ Within its software release series, commonly referred to as *point releases*.



One of the major features of the OcuGuard support package is the signature update capabilities. Without the latest signatures, an IDS will begin to miss more current attacks. Since the current attacks are always changing, the signatures absolutely must be kept up to date. As new signatures are adopted or built by Oculan's in-house security team, they are made available for download (or automatic download, which is more often the case) for OcuGuards deployed in the field. The OpticNerve, which handles the software update for all Oculan appliances, normally connects with Oculan's download servers once a day.

Oculan's goal with the OcuGuard appliance is to enable network and systems administrators with the information they need to attain a level of security awareness. While we're not turning administrators into security experts, we are providing them the critical information so that they can make informed security decisions about their environment. This is accomplished through a series of consistent, intuitive user interfaces, *wizard*-based configuration, and ongoing rule-based auto-administration of the appliance(s) itself.

The actual OcuGuard configuration utility is not a complex rule editor nor a scripting language, but a simple set of checkboxes allowing the user to decide which services on which operating system they're interested in watching. For instance, if an environment contains Microsoft Windows 2000 servers that are running Internet Information Server, a simple checkbox enables all the signatures for that service. If a network contains only Linux based servers, all Microsoft signatures can be switched off with a few clicks of the mouse, thus cutting down on the number of irrelevant events and potential false positives. And since the configuration is rules-driven, as new signatures are made available, they'll be automatically applied or ignored, based on the initial "check box" configuration completed by the installing technician.

While a certain level of knowledge is needed to understand the events on a network, there are advantages to having a network person in charge of security events (especially intrusion detection). An administrator that knows and understands the network will be more able to decide how an attack impacts the network and the systems on that network. The administrator will also be able to look at a network outage along with a denial of service signature and quickly understand why the network outage occurred. The network administrator will also have a strong knowledge of the systems installed on a network, as well as being responsible for determining the best location for OcuGuard sensors.

The Signature Update Process

The Oculan OcuGuard signature database and the team charged with its maintenance are driven by three key factors: accuracy, minimization of false alarms, and timeliness. We strive to provide the customer with accurate signatures that ensure maximum visibility of attacks without triggering false alarms. The Oculan security team makes every effort to provide updates in a timely manner, as traffic signatures are always a moving target.

As with most security companies, Oculan's primary source of signatures is the security community. As a group, the security community is quite open with respects to current and past attacks. Oculan monitors the many websites and mailing lists that post exploits, traffic patterns, and signatures. Our security team then takes this information back to our labs and tests. Given signatures, we validate them to make sure they do what they should and do not trigger many false alarms (although false alarms cannot be totally eliminated). Given traffic



patterns, we load them up on our test systems and create signatures from there. Given exploit tools, we load these up and let them go, all the while keeping exact logs of the traffic created. From these traffic logs, we create our signatures.

To completely simulate attack scenarios, our labs are populated with 'victim' systems (e.g., systems which serve as targets of attacks), OcuGuard appliances, systems dedicated to traffic collection and analysis, and 'attackers' (e.g., systems that are used to launch attacks for which we need to create signatures). We then iterate through various tests and record the effects, capturing every bit that crosses the wire for later replay. After a few runs of the attack against various victims, we analyze the data collected. We seek out identifying characteristics of the packets, such as the packet contents or the sequence numbers. Given that the majority of the attacks are only unique at the packet content level, we spend a lot of time there. We look for strings like '/bin/sh' and the string that leads up to it, especially looking for the "NOP sled" (i.e., packet "padding" used in buffer overflow attacks) that accompanies many exploits.

After new signatures are tested, we integrate them into our current set of signatures to make sure that there are no overlaps or conflicts. We also verify the alert messages sent, as well as track down and link to external sources of more information, where available. We then categorize each signature so that the rules engine can process it appropriately.

Once complete, we upload the signatures to our central server. This server is polled once a day by OpticNerves (if configured to auto-check for updates), and if any new files are found, they can be downloaded to the OpticNerve. Any OcuGuards connected to that particular OpticNerve will poll the OpticNerve and, in turn, download their updates within five minutes from the local machine.

Summary

Many small-to-midsized businesses (less than 5000 employees) have suddenly found themselves in an awkward position. Email and the Internet have mandated that the company has Internet access, which in turn requires the introduction of new technologies, new traffic, and new security concerns. At the same time, the number of threats to the unprotected Internet newcomer are growing exponentially. Meanwhile, legislation has either already been adopted, or is being proposed (e.g., Gramm-Leach-Bliley, HIPAA) which mandates that businesses take reasonable care in protecting their environments and in turn, their customers' data. Without a security plan that addresses how to manage these new technologies, deploy them, keep them up to date, and leverage their information to avoid hiring additional staff, the organization is doomed to suffer security-related losses or worse yet, business losses.

Oculan's pre-integrated product family provides the ability to manage networks and systems, as well as analyze traffic for threats that have made it past the firewall. The tools are built by Oculan's team, deployed by Oculan's family of technology partners, and the information provided to the end user is distilled to a point that little if any additional training is necessary.

Security is not a destination, but a journey, and it can often be a scary one, especially with limited staff, tools, and resources at your disposal. Don't opt to travel alone.