# Global Information Assurance Certification Paper

# Extending the Perimeter With OneSecure IDP

Austin Gilbert

GSEC v1.4 opt 2

October 18, 2002

## Abstract

*Due to the co-evolution of security defenses and offenses, there are ever increasing demands for Internet security. A successful network security analyst must always remain one step ahead of the game, falling behind can lead to devastating results. In an attempt to stay ahead of the game, security professionals must apply the principles of defense-in-depth, and multi-layered security, relying on one line of defense is always a poor strategy. In an effort to apply these principles on my organization's network, I installed OneSecure's Intrusion Detection and Prevention appliance outside of our firewall, the implementation includes the tuning of the IDP, the setup of a honey pot, and the discussion of the IDP's features. The implementation and results are presented in this paper.*

## Introduction

Technology continuously becomes more advanced and more cost effective as it evolves, presumably the benefits of Moore's Law. Dropping prices continue to drive the PC market to saturation, and every year more of the PCs sold are connected to the Internet. *All hail the glorious advances of technology!* One of the unfortunate drawbacks to these admirable advances includes the increase of Internet attacks on computer networks.

According to PricewaterhouseCooper's incident report for 2002, external attacks accounted for 48% to 64% of the worst security incidents that occurred during the year. They reported hefty increases in the amount of attacks occurring during the year, citing a 25% increase in virus infections, and a 10% increase in unauthorized system access. These numbers continue to climb every year despite the increased awareness in security and firewall deployment. This begs the question: how can we increase the assurance of our network fidelity?

Ten years ago, firewalls were an emerging technology, today they are a necessity. However, they do not provide absolute protection from threats. The co-evolution of security defenses, like firewalls, has been matched if not out-paced by the development of increasingly sophisticated network based attacks. Distributed port scanning, Distributed Denial of Service, Internet worms which drop root-kits onto vulnerable systems, and sophisticated email viruses are fairly commonplace events in this day in age.

Even positive technology development presents new security challenges. SOAP, for example, is a protocol independent messaging system, while intended for good, this protocol will no doubt become a major target for elite hackers. The problem that SOAP poses, right now, is that it isn't bound to a specific port number, so you can't simply block a port on your firewall. If there is a SOAP router attached to any protocols that you allow, then the SOAP messages will make it through. Unique problems, such as these, will continue to appear with the development and introduction of new technologies. Monitoring the contents of specific, problematic protocols is an area in which intrusion detection systems continue to improve; firewalls offering this feature are hit and miss.

### *Covering the Bases*

It is common knowledge that the first step to hacking a network is to obtain reconnaissance information. Typically the first step of reconnaissance involves a port scan of the network. This snapshot of which ports are open and closed narrows the attack scope potential hackers will take. Nearly everyone who is going to make a run at your network will perform a port scan of one kind or another. So it goes without saying, one of the ways to mitigate the Internet risk is to become invisible to scanning. Limiting the amount of intelligence information that the "enemy" has on your network can effectively lower the risk faced by being connected to the Internet.

While some firewalls do a good job of detecting network scanning, and blocking or dropping scan traffic, others do not perform this function at all. The network in question is set up with a pair of Nokia appliances running Checkpoint FW-1 version 4.1. All around a very robust and stable firewall, however, there is no integrated, real-time scan detection or prevention what-so-ever. There is a very good log viewing/querying tool that can be used to determine when a scan has occurred, and there are several home-grown scripts that add scan detection to the platform -- however, these scripts are not integrated into Checkpoint itself. Here, the primary purpose of a firewall is to protect vulnerable services from outside threats and to control access to the systems on a network [Wack]. This is a very traditional and wide spread view of what a firewall should be, although this paradigm is shifting to include more and more IDS functionality.

A key aspect in securing a network, is knowing whether or not the selected defenses are holding up to the attacks presented. A primary concern, to this end, is the firewall. Is it withstanding probes and attacks? [Wack]. How do we know for sure? Without an intrusion detection system in place, we have no reassurance that valid traffic (ie

traffic that is acceptable according to the firewall policy) is not malicious in nature. The fact firewalls pass malicious traffic on a semi-regular basis is why knocking over web-servers is a very popular activity; traffic to port 80 on a web-server is usually unrestricted. Clearly allowing HTTP traffic is a necessity, as well as allowing SMTP traffic for mail service, and traffic utilizing domain name services. An intrusion detection system (IDS) sheds light onto the nature of the incoming and outgoing traffic, giving system administrators one more advantage in protecting their systems.

### *Our Firewall is Swiss Cheese*

The stakes are high, and the risk is clear: Internet traffic is allowed to access our web-servers, email-servers, and our domain-name-servers. There is no reasonable control on many of these public services, nor is there a guarantee potential users are not malicious. Exploitation of any of these would result in huge loss of productivity, which means loss of revenue. Despite having a firewall in place, legitimate traffic, rather traffic posing as legitimate traffic, does make it through the firewall to its destination, leaving services vulnerable to zero day and other little-known exploitations. With a thorough understanding of this threat, and a risk to cost analysis in hand, the necessity of mitigation becomes absolutely lucid.

## Implementation

Knowing that something was needed to fill the void left by the firewall, and suspecting this void should be filled by an intrusion detection system, a way to justify the cost to management was also needed. The fastest way to justify an expense is to show the return on investment. That meant reports. This called for the ability to enumerate the real threat to our network, not just tally up false positives and script-kiddy-scans like many legacy intrusion detection systems. A way not only to detect attacks, but to stop them in progress, and to prevent them in the future. In essence, management had to be shown that trusting the firewall alone was not a prudent approach to security. Furthermore, they needed to be shown that adding an intrusion detection system to the line of defense was needed to insure the continued best practice of *multi-layered security*. This required demonstrating the existence of a very real threat: knowledgeable Internet hackers. In order to prove the Internet threat is real and to align with the principles of *defense-in-depth,* our network's perimeter would have to be extended. We needed something beyond the firewall; we needed an intrusion detection system.

The desire for an intrusion detection system raised many questions. What kinds of IDS systems are out there? Which variety should be use, host-based IDS (HIDS) or a network-based IDS (NIDS)? What are the advantages and disadvantages of each? If choosing a NIDS, should it be a passive sniffer based system or something more up-scale? What kind of features should it support? What kind of performance issues would this kind of system introduce? Where should the system be placed in the network?

Intrusion detection systems are classified by where they search for attack signatures. Systems searching in the flow of network traffic are, intuitively, referred to as network-based [ISS ]. Systems centered around file integrity checking [Turolla ] or systems which cruise log files looking for attack signatures and anomalies fall under the host-based designation [ ISS].

It should be noted that there are an increasing number of "next-generation" intrusion detection systems, which employ aspects of both network and host-based intrusion detection. Typically these parts are available separately, but can be integrated and managed with a central sever, as is the case with RealSecure 3.0. However, budget is always an issue in industry, and the cost for need requirements often dictate simpler stepping stone solutions. In our case, we were no where near meeting the labor requirements for installing and maintaining a regatta of host-based intrusion detection systems. The necessary lack of man power, significantly persuaded the use of a network intrusion detection system as the next step in the security evolution of the organization.

### *Host Based Intrusion Detection*

Host-based intrusion detection systems (HIDS) were initially developed in the early 1980's before networks were as prevalent as they are today [ISS]. In the simpler environment of that era, reviewing audit logs was fairly common practice, and an adequate means of dealing with security issues. Today, HIDS continue to provide a powerful forensic tool for investigating suspicious user activity, rook-kit cleanup, and general system auditing.

There are two basic categories of host-based intrusion detection: log scanners and integrity checkers [Turolla]. Log scanners include programs such as Swatch[1] (The Simple WATCHer ), Log Surfer[2], and LogSentry[3]. Rounding off the field on the integrity checking side are gems like: Tripwire[4,5], AIDE[6] ( Advanced Intrusion Detection Environment ), L5[7] , Fcheck[8], and Gog[9].

Typically, a full-fledged deployment of host-based intrusion detection

technologies would include a combination of file integrity checkers such as tripwire, and log auditing tools, such as SWATCH. The combinational usage of these technologies helps to increase security coverage, and ensure a wide range of coverage from attacks, as well as providing sufficient means for recovery, and after-the-fact analysis of security incidents.

The very fact that a HIDS is host-based makes it operating system dependent. This can be a huge disadvantage during deployment on heterogeneous networks. The heterogeneous nature of most large networks greatly increases the administrative effort needed to maintain wide-scale use of host-based intrusion detection systems, and consequently is one of the reasons most people avoid large scale usage of HIDS.

The major strength of host-based intrusion detection is stability. Once tuned, a host-based intrusion detection system (especially a file integrity checker) rarely produces false positives. Systems such as these provide absolute proof that there has been a breach. Traditionally, HIDS have not been thought of as real time systems, however, through the adaptation of operating system call hooks, HIDS can perform real time detection, and in some cases prevention. Entercept, a Cisco supported OEM product, is a good example of a HIDS that provides the ability to prevent file access to critical files, offering some protection from exploitation. The continually improving functionality of host-based intrusion detection makes it a competitive purchase when compared to the price of network-based intrusion detection systems.

*Network Intrusion Detection Systems*

Rather than relying on system logs or file integrity checking, network based intrusion detection systems utilize network traffic as their data source. There are two categories of network based intrusion detection systems: passive and active. Passive NIDS utilize sniffer technology to collect packet streams. It may do this by running in promiscuous mode, or it may utilize a spanning port on a switch to accomplish this. Typically passive NIDS are not run inline, therefore they usually only function as an early warning system for intrusions. However, through cooperation with other security measures it becomes possible for these out-of-line devices to extend protection similar to that of in-line intrusion detection systems. A fair illustration of this is: an intrusion detection system interconnected with a firewall via OPSEC protocols. In this case, the IDS can ensure that offending packet streams are blocked by notifying the firewall, the firewall then sets up a temporary rule to block the traffic. There are also solutions among the freeware solutions on unix and linux systems which do not support OPSEC

standards. Here, they take advantage of their firewall's command line interface, which provides the functionality necessary to block traffic. Active NIDS are different only in the respect that they can be placed in the flow of network traffic, with the ability to modify flow on its own accord. Active NIDS have more in common with firewalls than with sniffer based intrusion detection systems, and they will definitely become more and more popular.

Network based intrusion detection systems have several means of identifying potential threats. These include: pattern, expression, or bytecode matching, frequency or threasholding of events, correlations of lesser events, and statistical anomaly detection. Of these the last is the most unreliable and error prone. The OneSecure IDP adds a knowledge base of RFCs to help reduce the error in these signature based techniques. Generally speaking, one or more of these techniques are used almost universally by network-based intrusion detection systems.

There are few disadvantages to installing an intrusion detection system, whether passive or active. The worst consequence of running an IDS is the mild annoyance that can result from poor tuning. This is especially true, if the IDS is allowed to contact security administrators via email or pager. This drawback can be remedied with proper tuning. Tuning the intrusion detection system is the most demanding activity involved with the installation, and is the only other obviously drawback to NIDS. Experienced IDS users are sure to tell you that this is true for all kinds of intrusion detection systems, network-based and host-based. Basically you can prepare to get very intimate with the log viewer, because this is essentially the only tool available for tuning.

The benefits of having a network based intrusion detection system are fairly straight forward. Obviously, having a NIDS presents administrators with more information about traffic flow than they would have with out an IDS. Having an IDS gives administrators a heads up on the latest vulnerability trends. An increased frequency of probing on specific ports usually indicates a new exploit has been developed. The observance of peaks in scanning of a particular port should be followed up with Internet research for new exploits against this port. The fact that a NIDS sees all of the network traffic means that it can pick up on many things that host-based systems would never find. For example, a port scan is less likely to be picked up by a HIDS than a NIDS. This is especially the case when invalid packets are used for the scan, such as FIN packet. The host IP stack would generally drop an unsolicited FIN packet without logging the activity. On a good network-based system, however, the detection system would correlate multiple unsolicited SYNACK or FIN packets as a port scan. Again, this gives administrators

information that they typically wouldn't have before. This is also information which allows an intrusion detection system to proactively protect the network, because once a scan is recognized, not only can the scan be stopped, but any other traffic coming from the same origin can be blocked out as well.

*IDP to the Rescue*

When *Code Red* hit, almost everybody running IIS was vulnerable. In our organization, every single web-server running a vulnerable version of code was infected, and in turn, began to attack other systems on the web. Correcting the situation meant all hands on deck for a full day straight. The web servers had to be taken offline, restored from backups, patched, and then reconnected to the Net. If memory servers me correctly, the first patch for this problem did not work properly, and all the steps had to be repeated a second time. All in all this was a very costly problem. Fortunately no data was lost, but the combination of overtime labor hours, and loss of production was more than enough to justify more network protection. It was around this point in time when research into intrusion detection systems began. The prevailing fear generated from this event was: if a simple Internet worm could cause this kind of damage, what would the repercussions of a full network compromise be like? Several big budget security projects were approved as a result of *Code Red*, included among these allocations was one for an intrusion detection system.

Lacking enough dedicated staff to consider implementing host-based intrusion detection system (even a mediocre sized deployment of only critical severs was too overwhelming to consider ), the security group concluded that the next step should be a network-based intrusion detection system. The coverage provided by this system would help to "buy time" while administrative capabilities of host-based system increased to the point were our few hands could handle the installation and tuning process.

One of the difficulties with installing network-based intrusion detection is choosing a deployment location. Each potential location offers benefits and disadvantages, choosing a poor location can result in poor coverage of the network. Of course, this disadvantage can be eliminated by deploying multiple network intrusion detection systems, one to cover each vulnerable area: outside the firewall, one between the client PCs and the servers, one between the client and server network segments, etc. etc.

Given the budget, we could only choose one deployment location. The consensus in the security group was to place the intrusion detection

system outside of the firewall. While this left us unsure of exactly what was creeping past the firewall, it would have the greater benefit of producing fantastic report fodder by showing exactly how many times a day people were probing our network. Overall, we needed the fear factor on our side if management was to approve the money for this purchase, having these reports in-hand was essential to gaining approval. Showing them how many intrusion attempts are made each hour is powerfully persuasive. Being able to break threats out into categories of severity also makes effective arguments. The presentation goes something like this:

> "Here is the number of times a day we get port scanned, here are the number of buffer overflows run against our production web servers by the day, and here is the number of hackers we lured into the honey pot. Of the three the last is the real threat. This last group represents a real person with a brain -- thinking -- and snooping around the back-country of our network. With an intrusion detection system, we could eliminate 90% of these threats, reduce port scanning by two thirds, and stop all known web attack scripts in their tracks. Plus, have a very good idea of how many real threats are being made on our network."

The concern that legitimate traffic, web traffic for example, will cause significant damage to a network will always be a powerful influence towards implementing an intrusion detection system.

The traditional signature based network intrusion detection system is prone to false positives. Somebody sends you an email containing the phrase 'Gee WIZ', and you end up with an alert for the sendmail WIZ attack. Things have improved since the first generation detection systems. However, top of the line systems such as the OneSecure IDP do identify attacks by their signatures, albeit in a more accurate and advanced way. With the WIZ attack for example, the signature is only triggered if the WIZ signature is detected in the command field. It accomplishes this by not just sniffing SMTP traffic, but by actually understanding the RFC for this traffic. This is a unique approach, and was one of the key influences in selecting this particular intrusion detection system for installation. There are some caveats with this technique. They did too good a job adhering strictly to the RFCs, almost nobody else managed this level of exactness. In fact, most of the initial false positives flooding the logs after installation were protocol anomalies. On the other hand, the IDP was the only IDS that could recognize and drop all SNMP buffer overflow attacks that surfaced in 2001[10], simply because the traffic generated by these attacks did not conform to the RFC for SNMP traffic.

Word to the wise: do not believe the vendor when they say "No false positives". It just simply isn't the case. Anyone having worked with a variety of intrusion detection systems, will undoubtedly understand that

false positives will occur, even on a well tuned system. The thing that helps security administers maintain their sanity is how well they can tune a particular system. The OneSecure IDP offers very granular control in a very user friendly format, which was a much welcomed feature.

*IDP Features*

Non-technical features included fabulous 24 hour support. It is a good feeling to know that if there is a crisis requiring outside assistance, there will be someone there. Beyond that, the attack signatures for the IDP platform are updated about once a week on average. The only caveat with the update system is that they are only manually applied in version 1.6 (although there was talk about automatic updates in version 2.0 ).

Having freshly updated attack signatures is of little use if they are wrong and cannot be corrected. Signatures also turn out to be useless when there is a new threat on the block. Neither of these are problems with the OneSecure appliance. The signatures are not contained inside a black-box, giving administrators the ability to modify the current signatures, and create new ones if the need arises. One note, any changes that are made to the signatures provided by OneSecure will be over-written once the new signatures are downloaded and installed. This means you can make changes, but they will essentially be undone after you "update" the signatures from OneSecure.

The OneSecure IDP provides six basic software features: scan detection, back-door detection, SYN protection, a honey-pot, exploit detection, and a good log viewer. The settings for SYN, scan, and back-door detection are configurable via thresholds on the IDP. There are also separate rule sections for back-doors, SYN attacks, exploits, and the honey pot. This allows maximum configuration of the system, and allows administrators the ability to setup exceptions to general rules, or to monitor given attack signatures on specific destinations. It is, in fact, the same kind of granular that a firewall has over network traffic.

There are also a couple of advantages of going with an appliance based solution, such as the IDP. The operating system has already been harden, saving the security administrators tons of work. The intrusion detection software has been tightly integrated into the operating system yielding increased performance. Plus, this particular platform can be run inline as a bridge device. This means the only way to crack this system is from the management side of the box. Placing the management interface on an out-of-band network drastically reduces the possibility that an outsider will compromise this machine.

The further advantage of running this system as an inline device, is that the intrusion detection system can drop traffic in the same fashion a firewall would. After tuning the device to an acceptable level, changing a few small settings in the rule base, allows offending traffic to be dropped. Therefore, once the system has been tuned, and all of the appropriate signatures loaded, attacks like *code red* can be stopped, period. That includes the ability to prevent inbound and outbound attacks. Real zealots will appreciate the ability to prevent any traffic not complying with RFC standards. This is not recommended due to the fact large percentages of legitimate traffic break these rules on a regular basis.

Of all the features provided by the OneSecure IDP appliance, the most intriguing and useful feature turned out to be the honey pot capabilities. Not only does the honey pot act as a decoy to attract malicious traffic away from you production network, but it can also be used as a proactive tool to prevent future attacks. The idea is to stop the attack while in its infancy, the reconnaissance phase. To accomplish this, you must find an effective way to identify and cut off port scanning.

*Setting Up the Honey Net*

The honey pots were setup in the IDP after selecting two Internet routed IP addresses, and giving them juicy DNS names ( e.g. payroll.my.org, accounting.my.org ). These two host were completely virtual, the IDP answers on any port that we decide to have the honey pots mimic. The honey pot was configured to answer on a few of the most popularly scanned ports, per http://www.dshield.org. Cleverly, enough ports were selected to make the server looked like a hardened Internet machine, but not so many as to make it an obvious target. Decoy number one pretends to run ftp, http, and https. This is complimented by decoy number two, which is running LDAP, Microsoft SQL, NTP, and SSH. Of the two decoys, number two was created to be more interesting. Decoy number one was essentially setup to help pick-off people who were scanning our fleet of web servers.

As usual, while tuning the IDP, the default behavior is to log and allow the offending traffic. After observing a few hits on the honey pot, the rules were adjusted to drop the offending traffic. In both setups, the destination address field was the virtually hosted honey pot. Before allowing traffic to be dropped, however, the source field in the rule was modified to exclude any of our organizations NAT'd addresses, or any of our external IP addresses. Previously, any address connecting to the honey pot would have been blocked. This was changed as a safety

feature to ensure that none of our internal users ever get locked out *accidentally*, despite trying to reach a host they shouldn't be.

On the IDP, there are several actions that can be taken due to a rule firing. The choices are:

IDP Notify
IDP Drop
IDP Close
OPSEC Notify
OPSEC inhibit
OPSEC inhibit and close
OPSEC inhibit drop
OPSEC inhibit, drop, and close

*IDP Notify* takes no future action against the source, except to log traffic from the offender. *IDP Drop* drops future connections based on the blocking options box. In the blocking options box you can choose to block based on several values: the source, the source subnet, the source/service pair, the source/destination/service triplet, the source subnet/service pair, and the source subnet/destination/service triplet. Plus you have a timeout parameter, which tells the IDP how many seconds the offender should be punished for. This was configured to block offenders for 30 minutes; long enough to frustrate hackers, short enough that legitimate users will not lose too much time should they find themselves blocked by the IDP. *IDP Close*, closes all future connections based on the blocking options box. *OPSEC Notify* tells an OPSEC compliant system, like a Checkpoint firewall, to log the offender's future traffic. *OPSEC Inhibit* tells an OPSEC system to close future connections matching the blocking options box. *OPSEC Inhibit and close* tells the OPSEC system to close the current connection and future connections per the parameters of the blocking options box. *OPSEC Inhibit and Drop* tells the OPSEC systems to close future connections and drop the current connection per the blocking options box. And finally, the *OPSEC inhibit, drop, and close* option tells OPSEC systems to drop future connections and close the current connections matching the blocking options box.

All of the rules in the honey pot were configured to allow the IDP appliance to drop offending traffic, rather than rely on OPSEC communications which would have increased the difficulty of the installation.

### *Honey Pot Benefits*

What have we accomplished by setting up honey pots with virtual hosts? By dropping traffic from any source that hits a mimicked service, we can

effectively cut off many port scans. In fact, any port scan that grazes across the faked ports is terminated immediately. Comparing the quantity of scans logged before the honey pot was initialized, with the number of scans after the honey pot was initiated revealed a major advantage to this tactic. The number of completed port scans was reduced by two thirds. This certainly cuts out the scanners who blindly scan our entire address range for every single open port. Limiting the quantity of host scanned will limit the number of targets a hacker has to pick from. The logical argument follows: fewer targets, fewer known vulnerable services, fewer hacking attempts.

*Pushing the Envelope*

The benefits of the honey pot prove to be well worth the fifteen minute setup, but we can do better by using another feature of the honey pot. At first, the presence of this feature was surprising, but its usefulness should be immediately recognized. The OneSecure IDP honey pot has the ability to imitate single (or multiple) ports on production hosts. That is, the IDP can fake virtual ports on production servers, just as easily as it mimics entire IP addresses. Therefore, it can make a production server appear as if a specified port is open, when in fact the port is closed. What does this mean in terms of security posture for our network? It means that anyone stumbling across mimicked ports on our network will have their traffic blocked for thirty minutes. This is extremely useful, because we can blanket our entire IP block from being blindly scanned. Obviously, skilled and persistent hackers, will eventually figure out which ports to avoid while scanning our network, however, we will have become virtually invisible to the average script-kiddy, and this is a tremendous advantage. Reducing risk is the name of the game in security, by faking a couple of ports on each host, we can drastically reduce the number of completed port scans against our network, and thereby reduce the number of hackers coming around as well.

Some very commonly probed SMB ports, which would never be allowed on external machines, were deviously chosen as the honey ports: 137 and 139. Port 137 was chosen, mostly because it is heavily scanned on our network. Scans in the logs after the initial honey pot setup, were scans that probed port 137 on our entire IP block. Being that port 137 was not emulated on our decoy machines, these scans were allowed to complete unimpeded, setting up a faked port on every host was the obvious solution to this problem. Port 139 was chosen, because anyone trying to connect to this port from the outside is not our friend. The Benefits to mimicking these ports are two fold. One, it knocks out even more of the scanning than just the honey pots alone. Two, should, for some reason, the firewall become mis-configured to allow SMB traffic,

the host will still be protected because the IDP will answer up on these ports instead of the actual host.

Again, the further reduction of information given away to hackers represents an increase in network security. Even if it only protects from ankle-biters, the improvement is worth the investment. Truly skilled hackers will find their way into any network. The key is to make it as difficult as possible and to have a sound audit trail for forensic analysis, in the event of a compromise.

## Discussion

The installation of the OneSecure IDP has greatly enhanced the security posture of our organization. Not having an Intrusion Detection System is like running through a forest blindfolded, you know there is a chance your going to run into a tree, but you will never see it coming. Putting an IDS in place is akin to removing the blind fold, suddenly you have a very clear picture of who and what is knocking at your door, hopefully giving you enough awareness to get out of the way. Seeing what kinds of attacks are bouncing off our web, email, and domain-name servers regularly is imperitive for disaster readiness. If nothing else, observing the quantity of attempts made will motivate the local IT personnel to maintain current patches and hotfixes on production servers.

As with most intrusion detection systems installations, tuning the systems is the most labor intensive aspect of the initial setup. The OneSecure IDP, fortunately, has a very user friendly log viewer, which allowed a very quick tuning. Each rule has the option to setup a packet capture, along with any alarms; seeing the actual traffic is imperative for deciding whether an alarm is a false positive or not. Tuning the IDP on our 3500 node network took approximately twenty-six man hours. This is actually on the low end of the spectrum; typically the initial tuning of an intrusion detection system will take around forty hours to accomplish. After blatant false positives have been removed, and the system is running smoothly, finer tuning can be undertaken. The fine tuning of the system took between twelve and twenty hours. It was less of a priority, because once the system was tuned within reasonable thresholds we could manually sort through which alarms were meaningful and which were not. On the initial setup, determining the difference between meaningful alarms and flaky alarms is very difficult. Imagine, trying to fill a tea cup from Niagra Falls -- without over filling it. This is the feeling of doing the initial tuning outside of the firewall. It is a balancing act, trying to reduce the number of alarms without throwing out too many real alarms in the process. Having the system inside the firewall would have been a very different experience. There would have been far fewer protocol anomalies and alerts caused by port scanning,

however there would have been less information to use in management reports as well.

There are two methodologies toward tuning an intrusion detection system. On one hand, a person may want the lowest possible number of false negatives, on the other the lowest possible number of false positives. There are benefits and compromises on both sides. Those preferring low rates of false negatives, must endure much higher rates of false positives. Personally, the other side of the fence is preferable: low rates of false positives, with higher rates of false negatives. High rates of false negatives mean that sometimes attacks will make it through the perimeter without detection, however, if an alarm is going off every thirty seconds, then the system is not practically functional. Besides, these attacks were making it through before the intrusion detection system was in place, now, at least some of the attacks are being logged and dropped. The reasoning behind having low rates of false negatives is somewhat better suited for disease detection in the medical field. The idea being, not to miss anything "bad", in this case, not to miss reporting on any particular attack. In the field of network security, were each alarm will result in time spent investigating, the preferable method lies squarely with low rates of false positives.

*Honey Pot*

To reiterate, the use of the honey pot improves the security posture of our network by reducing the reconnaissance information available to hackers. Information is power. By reducing the amount of information that can be gained by a wide spread port scan, we have significantly reduced our Internet risk. We have also reduced the level of "noise" on our intrusion detection system. Mostly the honey pot and honey ports will prevent unskilled hackers from running rampant on our network. This means that the IDS logs are now filled with smaller quantities of more serious attack patterns that can be investigated vigorously by security personnel, rather than larger quantities of meaningless port scans run by teenagers for fun after school. Here, we are implementing the honey pots to help enumerate the real threat to our security.

# Conclusions

The OneSecure experience has been a good one. Examining the level of network security awareness prior to installing the intrusion detection system, with the level of awareness and security after the installation, reveals an extreme level of improvement. Prior to installing the system, there was no record of any port scans occurring. Any kind of port scan completed successfully, easily giving away which protocols were allowed

to reach our publicly addressable host. Code Red attacks continually banged against our web servers, and SMTP buffer offer-flow attempts were run against our mail servers on a daily basis. Non of these attacks were recorded or prevented. After installing the intrusion detection system, these attacks were not only detected and logged, but in most cases stopped. The most impressive improvement came from the ability to use the honey pot to block scanning activity. The less information given away to an attacker, the better the security posture becomes. The fact that the honey pot could be used to prevent effective scanning of our network, reduced the number of port scans drastically: from a half dozen scans an hour, to a half dozen successful scans per day. The only caveat that permitted any scans to complete is the fact that the honey pot failed to prevent most FIN scanning. The vendor has been notified of this flaw, and hopefully will have a fix in place in the next revision of the system. Other improvements in security posture include the prevention of several known buffer overflows, viruses, back-doors, and other generally annoying network activity.

Overall, placing the intrusion detection system outside the firewall has been an eye opening experience. Surely placing the appliance behind the firewall would have made it magnitudes easier to tune, however, the ability to enumerate threats against our network would have been crippled. The system would functionally operate in the same way: anything dangerous passing through the firewall would be stopped. The decision on where to place the system has the potential to become a religious battle, just remember that placing it in either location will enhance network security.

### *Next-Generation Intrusion Detection*

The term firewall probably isn't going anywhere. In the future however, the companies who have developed successful intrusion detection systems will be the front runners in the industry. Tomorrow's Checkpoint, will be the company who gains the most market shares in the intrusion detection market today. Why is this? Evolution. With an inline intrusion detection system like OneSecure's IDP, it is possible to drop traffic arbitrarily based on port, protocol, source or destination. It is not necessary for traffic to trigger an attack signature for the traffic to be blocked or dropped. Intrusion detection systems that can perform the basic functions traditionally handled by firewalls will become more and more common, until one day (in the very near future) the two systems will be completely integrated and managed from a single client program. The few obvious aspects of the OneSecure IDP that currently prevent its use as a firewall replacement are the lack of VPN termination and network address translation (NAT). Some

organizations don't use their firewalls for VPN termination, and have VPN concentrators or other VPN termination points already, for them, the path for replacing the firewall slightly clearer. For others, VPN termination functionality on the intrusion detection system will eventually be filled out through the integration of the IDP with ScreenOS from Netscreen[11], a company which recently acquired OneSecure. The integration with ScreenOS should flesh out a usable NAT implementation as well. Using the IDP as both firewall and intrusion detection system has a two fold benefit: reclamation of hardware used for the current firewall, and integration of functionality for security administrators. The second benefit is worth its weight in gold, it means not having to maintain multiple sets of security policies for an organization. It means only opening one application to trouble shoot Internet connectivity issues, rather than having to open one console for firewall administration and another for the intrusion detection system.

## Footnotes

1. http://www.oit.ucsb.edu/~eta/swatch
2. http://www.cert.dfn.de/eng/logsurf
3. http://www.psionic.com/products
4. http://www.tripwire.com
5. http://www.tripwire.org
6. http://www.cs.tut.fi/~rammer/aide.html
7. ftp://coast.cs.purdue.edu/pub/tools/unix/L5
8. http://www.geocities.com/fcheck2000/fcheck.html
9. http://www.multimania.com/cparisel/gog
10. http://www.cert.org/advisories/CA-2002-03.html
11. http://www.netscreen.com/main.html

## References

Frederick, Karen Kent. "Evaluating Network Intrusion Detection Signatures, Part 1." Infocus. 10 Sept 2002.
9 Oct 2002. <http://online.secuirtyfocus.com/infocus/1623>

"Information Security Breaches Survey 2002." PricewaterhouseCoopers. 9 Oct 2002.
<https://www.security-survey.gov.uk/isbs2002_detailedreport.pdf>

"Network-vs. Host-based Intrusion Detection: A Guide to Intrusion Detection Technology."
Internet Security Systems. 2 Oct 1998. 5 Oct 2002.
<http://documents.iss.net/whitepapers/nvh_ids.pdf>

Thomas, Benjamin D. "Intrustion Detection Primer" Linuxsecurity. 13 March 2002. 9 Oct 2002.
<http://www.linuxsecurity.com/feature_stories/feature_story-8.html>

"Top 10 Target Ports." DShield.org Distributed Intrusion Detection System. Euclidian Consulting. 17 Oct 2002.
9 Oct 2002. <http://www.dshield.org/topports.html>

Turolla, Stefano. "Host Based Intrusion Detection." European Organization for Nuclear Research.
25 April 2000. 9 Oct 2002.
<http://wwwinfo.cern.ch/dis/security/general/tools/host_based.html>

Wack, John. "Logging and Statistics on Network Use, Misuse." Why Firewalls. 9 Feb 1995. 9 Oct 2002.
<http://csrc.nist.gov/publications/nistpubs/800-10/node37.html>

Last Updated: July 24th, 2017

# Upcoming Training


| | | | |
|---|---|---|---|
| SANS Prague 2017 | Prague, Czech Republic | Aug 07, 2017 - Aug 12, 2017 | Live Event |
| SANS Boston 2017 | Boston, MA | Aug 07, 2017 - Aug 12, 2017 | Live Event |
| SANS Salt Lake City 2017 | Salt Lake City, UT | Aug 14, 2017 - Aug 19, 2017 | Live Event |
| Community SANS Omaha SEC401* | Omaha, NE | Aug 14, 2017 - Aug 19, 2017 | Community SANS |
| SANS New York City 2017 | New York City, NY | Aug 14, 2017 - Aug 19, 2017 | Live Event |
| Community SANS Trenton SEC401 | Trenton, NJ | Aug 21, 2017 - Aug 26, 2017 | Community SANS |
| Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style | Virginia Beach, VA | Aug 21, 2017 - Aug 26, 2017 | vLive |
| SANS Chicago 2017 | Chicago, IL | Aug 21, 2017 - Aug 26, 2017 | Live Event |
| SANS Virginia Beach 2017 | Virginia Beach, VA | Aug 21, 2017 - Sep 01, 2017 | Live Event |
| SANS Adelaide 2017 | Adelaide, Australia | Aug 21, 2017 - Aug 26, 2017 | Live Event |
| Community SANS Pasadena SEC401 @ NASA | Pasadena, CA | Aug 23, 2017 - Aug 30, 2017 | Community SANS |
| Mentor Session - SEC401 | Minneapolis, MN | Aug 29, 2017 - Oct 10, 2017 | Mentor |
| SANS Tampa - Clearwater 2017 | Clearwater, FL | Sep 05, 2017 - Sep 10, 2017 | Live Event |
| SANS San Francisco Fall 2017 | San Francisco, CA | Sep 05, 2017 - Sep 10, 2017 | Live Event |
| Mentor Session - SEC401 | Edmonton, AB | Sep 06, 2017 - Oct 18, 2017 | Mentor |
| SANS Network Security 2017 | Las Vegas, NV | Sep 10, 2017 - Sep 17, 2017 | Live Event |
| Mentor Session - SEC401 | Ventura, CA | Sep 11, 2017 - Oct 12, 2017 | Mentor |
| Community SANS Albany SEC401 | Albany, NY | Sep 11, 2017 - Sep 16, 2017 | Community SANS |
| Community SANS Dallas SEC401 | Dallas, TX | Sep 18, 2017 - Sep 23, 2017 | Community SANS |
| Community SANS Columbia SEC401 | Columbia, MD | Sep 18, 2017 - Sep 23, 2017 | Community SANS |
| SANS Baltimore Fall 2017 | Baltimore, MD | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| SANS Copenhagen 2017 | Copenhagen, Denmark | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| Community SANS Boise SEC401 | Boise, ID | Sep 25, 2017 - Sep 30, 2017 | Community SANS |
| Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style | Baltimore, MD | Sep 25, 2017 - Sep 30, 2017 | vLive |
| Community SANS New York SEC401 | New York, NY | Sep 25, 2017 - Sep 30, 2017 | Community SANS |
| Rocky Mountain Fall 2017 | Denver, CO | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| SANS London September 2017 | London, United Kingdom | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| Community SANS Sacramento SEC401 | Sacramento, CA | Oct 02, 2017 - Oct 07, 2017 | Community SANS |
| SANS DFIR Prague 2017 | Prague, Czech Republic | Oct 02, 2017 - Oct 08, 2017 | Live Event |
| Community SANS Charleston SEC401 | Charleston, SC | Oct 02, 2017 - Oct 07, 2017 | Community SANS |
| Mentor Session - SEC401 | Arlington, VA | Oct 04, 2017 - Nov 15, 2017 | Mentor |