A New Approach To Intrusion Detection: Intrusion Prevention

Executive Summary

Just as a sophisticated home security system might comprise cameras and sensors and monitoring equipment to watch for suspicious activity within the house – i.e., an unauthorized intruder -- so does an intrusion detection system alert IT system administrators to potential security breaches within a network environment. Like other traditional security technologies that have been assimilated into the broader scope of IT security planning, intrusion detection systems have emerged over the past five years as a growing requisite to the security paradigm.

However, even with the most well-planned and strategic implementation of these systems, either alone or in conjunction with other technologies, an ongoing variety of security attacks have shown repeatedly that such technologies are not doing the complete job of securing enterprise networks. These technologies are also accompanied by a substantial administrative burden that further compounds the problem.

This paper will examine the current state of the intrusion detection technology landscape, analyze critical weaknesses and provide an overview of OKENA StormWatch™ **Intrusion Prevention** security software, which serves as a realistic **host-based** replacement strategy for those processing environments that require proactive and preventive security measures in the face of attacks directed at today's open networks.

What is Intrusion Detection Technology and What Makes It Different From Other Technologies?

Intrusion detection technologies fall into two categories: *host-based* and *network-based*. Intrusion detection products are software and hardware products designed to monitor a device or network for malicious activity. Intrusion detection software is designed to compare network/resource activity to a list of signatures known to represent malicious activity. How does this fit into an overall security strategy? As organizations rely more and more heavily on Internet connectivity as a means of doing business, the burden placed on traditional security technologies becomes stronger and the associated stress points are more clearly defined.

Network Intrusion Detection Systems

Network Intrusion Detection Systems (NIDS) are dedicated software systems that "sit" on a network wire and analyze network packets. The data encapsulated in these packets is compared to a database of known attack signatures. If the data passing along the network does not match a known attack listed in the database, then that traffic continues without suspicion. However if the packet data matches a known attack, then some sort of response may be generated. Such a response might come in the form of an alert that is sent to a log file or perhaps even a page to a network administrator.

Host-based Intrusion Detection Systems

Host-based Intrusion Detection Systems (HIDS) have emerged as a result of historically monitoring audit log files. With traditional systems, administrators search through log files at the end of the day to detect any suspicious activity that had occurred during a defined time period. The negative aspects of this analysis became obvious in that the work itself is not only tedious, but also untimely. Newer HIDS offer a local agent performing the same scanning activity as an event occurs. As the event is logged to a log file, the local software agent installed on the resource checks to see whether that event matches any of those listed in its attack database. Some HIDS also have the ability to monitor application log files for evidence of additional attacks. HIDS also have the ability to monitor local files for any changes or modifications. When an event indeed matches an attack profile, the HIDS will send an alert and generate any one of a series of potential actions in hopes of deterring damage caused by malicious activity.

Limitations of Reactive IDS Technologies

All IDS technologies share the same critical flaw as other traditional information security technologies: they are *passive* and *reactive*. Because these solutions are predicated on signature detection, even when they are efficiently installed and administered, a variety of attacks will still cause damage to network resources and files on individual machines. There are several weaknesses endemic to both network and host-based intrusion detection systems.

Increasing Vulnerabilities

Signature-based security technologies were conceived in the mid-1990s, when there were few reported vulnerabilities. According to CERT, there were only 171 vulnerabilities announced in 1995. A signature-based product that was capable of being updated with a dozen new signatures a month did a relatively good job of providing security.

What we've seen since that time is a growing increase in reported vulnerabilities, which has transformed the landscape. The number of vulnerabilities reported each year is increasing exponentially, with no end in sight. Now consider mutations among these, which increase the count further. Figure 1-1 shows the increase in reported vulnerabilities in the late 1990s and early 2000s (source: CERT).

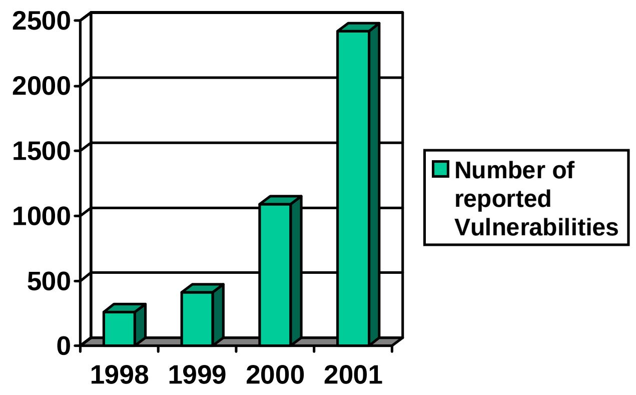


Figure 1-1. Increasing number of Vulnerabilities.

Signature-based security products will have to have a signature for each of these to protect against attacks. Over the course of 2001, this would have required 6 new signatures EVERY DAY. We can anticipate that the number is now approaching a dozen new signatures a day. Clearly, signature-based security is unable to keep up with the situation. The exponential increase in reported vulnerabilities suggests that there likely will NOT be a signature for a given attack.

Managing Alerts and False Positives

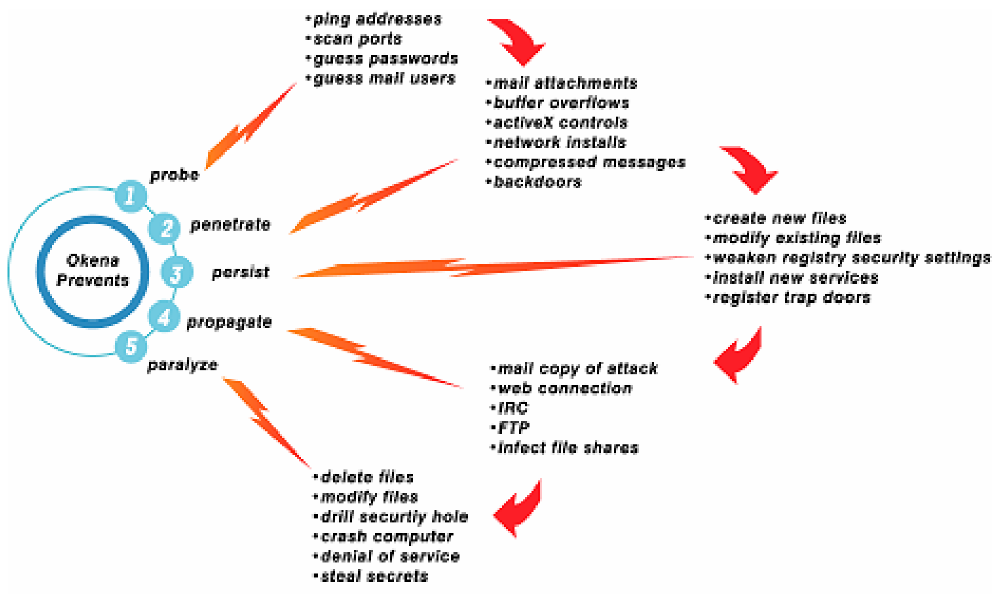
Pattern and signature matching technologies are prone to a high number of false positives and false negatives. In other words, the signatures that they deploy generate false alarms that indicate possible malicious activity when, in fact, appropriate host and network behavior is taking place. These false positives crowd administrative consoles, reduce the effectiveness of the product and the administrator and force the admin to respond to unwarranted security alerts.

A Different Approach

A different approach is called for – one that can even detect attacks against vulnerabilities that have not been published, or even discovered. Interestingly, there is a commonality across attacks that can be used to do precisely this.

The Attack Lifecycle

All attacks will follow the same logical progression, as shown in **Figure 1-2**:



1. Vulnerable targets are identified in the **Probe** phase. The goal of this phase is to find computers that can be subverted.
2. Exploit code is transferred to the vulnerable target in the **Penetrate** phase. The goal of this phase is to get the target executing the exploit code via some attack vector like a buffer overflow.
3. Once an exploit has been successful, the exploit code tries to make itself persistent on the target. The goal of the **Persist** phase is to ensure that the attacker's code will be running and available to the attacker even if the target system reboots.
4. Now that an attacker has a beachhead in the organization, it is time to extend this to other targets. The **Propagate** phase looks for vulnerable neighboring machines that the exploit code can spread to.
5. Only in the **Paralyze** phase is actual damage done. Files are erased, systems are crashed, and Distributed Denial Of Service (DDoS) attacks are launched.

There is a major dividing line between the **Penetrate** and the **Persist** stages. The first two stages are highly subject to mutation (the footprint of the attack is continually changing). They are also highly subject to being hidden from defenses via "Evasion Techniques" like Unicode encoding of web strings or overlapping packet fragments. Since attack identification at the Penetrate stage sometimes involves a certain amount of interpretation (guessing how the target computer will handle the network packet), it tends to be a large generator of false alarms.

The last three stages, in contrast, are highly stable over time. There are a limited number of malicious activities that an attacker can perform – modify the operating system, add a new user account, open up an outgoing network connection, delete files. This list has remained remarkably unchanged: the Morris Worm of 1988 did the same types of damage as the NIMDA Worm of 2001. Also, because activities like modification of operating system binaries are highly remarkable and unusual events, it is much easier to identify attacks accurately at these stages**.**
In short, if you try to identify attacks at the early stages of the lifecycle, each attack will look different, and you will be caught in the signature "update race". Security should be offered through all stages of the attack and especially the last three where the damage takes place. Providing this defense-in-depth solution is the driving force behind OKENA's behavior based technology.

The vulnerability situation is clearly painful – bad and getting worse. Signature-based host security has not kept pace with the threat, and is falling further behind. OKENA offers a radical departure from this failed technology – by analyzing the ***behavior*** that is used to attack and damage computer systems.

Types of Behavior

Instead of focusing on attacks, behavior-based security focuses on preventing malicious activity on the host. By focusing on ***behavior***, damaging activity can be detected and blocked *no matter which attack was used*. For example, one security-relevant behavior might be *Web servers adding a new user account to the system.* There are many possible types of attacks (viewed from the *Penetrate* stage*)* that might cause this. A signature-based security system would need a signature for all of these. A behavior-based system would not.

Security-relevant behaviors come in three classes:

- **Malicious Activity.** This type of behavior is what is typically seen in the final three stages of the attack lifecycle. Examples include unauthorized modifications to the operating system or deletion of files. Since malicious activity is always undesired, security at this level is very inexpensive to deploy. Little or no environment tuning is required to stop malicious activity.
- **Policy-relevant Activity.** This is activity that, while not necessarily malicious, is undesired in your environment. For example, you may wish to prevent users from downloading files using Instant Messenger, because of the concern that they are not being scanned by the corporate email virus scanner. This "everything not expressly prohibited is allowed" approach allows customers to rapidly implement policy directives from senior management.
- **Application Wrapping.** For extreme levels of security, it is possible to entirely lock down an application. **Only known safe behavior is allowed for these applications.** By enforcing "good" behavior, anything outside the scope of this good behavior, whether a new attack or simply some kind of error, can be addressed very effectively. Basically, this "Everything not expressly permitted is prohibited" approach offers the highest possible level of security. Security at this level will require more tuning effort. However, few systems will require (or justify) this strict level of control.

Most organizations are happy to stop malicious activity, and never extend behavior-based controls to include policy-related activity or application wrapping.

StormWatch™ Intrusion Prevention for Servers and Desktops

OKENA StormWatch provides uncompromising intrusion prevention security to your enterprise by deploying intelligent agents on desktops and servers that defend against the proliferation of attacks across networks.

StormWatch takes a unique and truly proactive approach to preventing intrusions. Unlike existing security solutions that are attack centric (reliant on databases of known attack signatures) or user centric (user access control) StormWatch is **behavior centric;** by focusing on the behavior of mission critical applications, StormWatch protects enterprises against **known** but more importantly **unknown** security risks. This provides four key advantages to users:

1. **Desktop & Server Intrusion Prevention.** Prevention, not detection is what is needed to get ahead of the security 'update race'.
2. **Dramatically increased accuracy allows prevention.** If you can't trust the security product to sort the good from the bad, you can never let it actually stop anything. With the unacceptably high levels of false positives (false alarms) generated by most security products, most of what would be stopped would be legitimate. StormWatch's automatic correlation combined with its "defense in depth" results in near-zero false positives.
3. **Behavior-based, not signature-based.** Stops new attacks that attempt the same old malicious activity.
4. **Zero Update.** OKENA default security policies get you out from under the update race.

The first point is that StormWatch doesn't just detect attacks, it stops them. As IT departments struggle to accomplish more, and make do with less automatic prevention of attacks is no longer a desirable option, it is critical. OKENA technology is designed such that everything malicious is stopped. Rather than offering an overburdened administrator an alert suggesting that an attack might be in progress and that he should check to make sure that his host hasn't been compromised, StormWatch alerts are always of the form "I saw this bad thing and I stopped it. You might want to check it out if you have some spare time, but don't worry – it's been stopped". Every alert has this form.

StormWatch addresses attacks at every stage of the attack lifecycle. By analyzing and correlating information at every stage, it is possible to distinguish truly malicious from seemingly suspect (but in reality perfectly normal) traffic. It can also detect attacks even when data packets are encrypted – even encrypted with SSL[1]. The result is a dramatic reduction in false positive alerts – typically by a factor of 40 or more.

Most Host-based IDS (HIDS) systems suffer notoriously from false alarms. One HIDS vendor recommends (in its documentation) that you have no more than 20-30 sensors reporting to a single management console. The reason isn't because the console cannot handle more than that many sensors, but rather because the human operator will be entirely overwhelmed by the volume of alerts. Is the system actually under that many attacks? Not at all – the problem is that the signatures are inaccurate.

[1] Technically speaking, the SSL encryption routines are compiled into the server or client programs, so decryption is done by the application, not the protocol stack. This means that a Host IDS system analyzing data at the TDI layer (a "Network Node IDS") will only see encrypted traffic.

Even if the sensor could be configured to block traffic, it generates so many false positive alerts that nobody trusts it to prevent the attack.

StormWatch does not rely on, or even contain any signatures. Security is provided by policy rules defined on a management console and distributed to intelligent agents residing on critical server and desktops requiring security. All rules can be inspected, and the Management console even provides a human language description of what the policies do. The policy rules focus on behavior – and as we saw in the attack lifecycle illustration (Figure 1-2), there are in reality relatively few malicious behaviors. A single rule that stops the Operating System from being overwritten will prevent many new and unknown exploits, because the exploits rely on overwriting the OS.

Lastly, because security is behavior and policy based, there are no signature updates that need to be distributed to the agents. This "Zero Update" architecture will have a dramatic cost saving over the lifetime of the product. Consider what must be done to update signatures on several HIDS protected critical servers: the signature has to be obtained from the vendor, it has to be tested in the lab to ensure that it doesn't crash applications running on the host, it has to be deployed to the servers, and (sometimes) it has an unanticipated side effect that wasn't seen during testing and has to be removed. Each of these actions represents time that the administrators have to spend obtaining, validating, and deploying the update. The more updates there are, the higher this cost will be. If updates are infrequent, this cost will be lower but obviously the host will be insecure for a longer period.

The StormWatch Single Agent

With a policy-based system like StormWatch, multiple security problems can be solved depending on how the policies are configured. StormWatch addresses several major security areas:

1. Hardening for servers and desktops, including File System/OS Lockdown and integrity Baseline.
2. Buffer Overflow and network attack protection.
3. Web Server protection.
4. Desktop Distributed Firewall.
5. Malicious Code Sandbox**,** protecting against attacks received from Java, Javascript, ActiveX, or other "Mobile Code" types**.**

This "Agent Consolidation" provides an incredible Return On Investment for customers – we'll discuss that a little later.

StormWatch Agent Architecture

StormWatch has application visibility because of its location alongside the kernel (in the same fashion as anti-viral or distributed firewall products which are also unobtrusive to the OS). All system calls to file, network and configuration resources can be intercepted using our patent-pending **INCORE** (**IN**tercept **CO**relate **R**ules **E**ngine) technology. More important than interception of calls however is the subsequent correlation carried out by StormWatch. This correlation and the resultant understanding of an application's behavior is what allows the software to truly prevent new intrusions.

When an application needs access to system resources, it makes an Operating System call to the kernel. INCORE intercepts these OS calls, and compares them with a cached policy which was centrally defined on the Manager (and downloaded by the agent when it polled the Manager). It correlates this particular OS call with others made by that application/process, and correlates these events to detect malicious

activity. If the request does not violate policy, it is passed to the kernel for execution. If the request does violate policy, it is blocked (not passed to the kernel), an appropriate error message is passed back to the application, and an alert is generated and sent from the agent to the Manager.
For example, a web server is serving HTML web pages. As incoming web requests are received, the web server generates file system I/O and network packet I/O requests. As long as these are within the bounds of the policy (example: web server applications have read access to web pages), no security events are generated. If a known attack (say, a UNICODE directory traversal attack, hidden via SSL encryption) tries to make the application act outside this policy (example: open a command shell like CMD.EXE), the request will violate policy and will be blocked. An error message like 404: Not Found will be generated to the remote user.

Suppose an attacker were to try an unknown, never-before-seen attack like a buffer overflow attack. Again, this could be hidden via SSL encryption or evasion techniques. The Execution Space interceptor will detect the application violating its own or another's execution space/environment. In this case, it would detect code executing from data space, and block the execution. Because this behavior violates policy, no update would be needed to block the new attack – thus the name "Zero Update".
INCORE supports four interceptors:

- File System (this is easy to understand – all file read or write requests are intercepted)
- Network (packet events at the driver (NDIS) or transport (TDI) level).
- Configuration (read/write requests to the registry on Windows or to rc files on Unix)
- Execution Space.

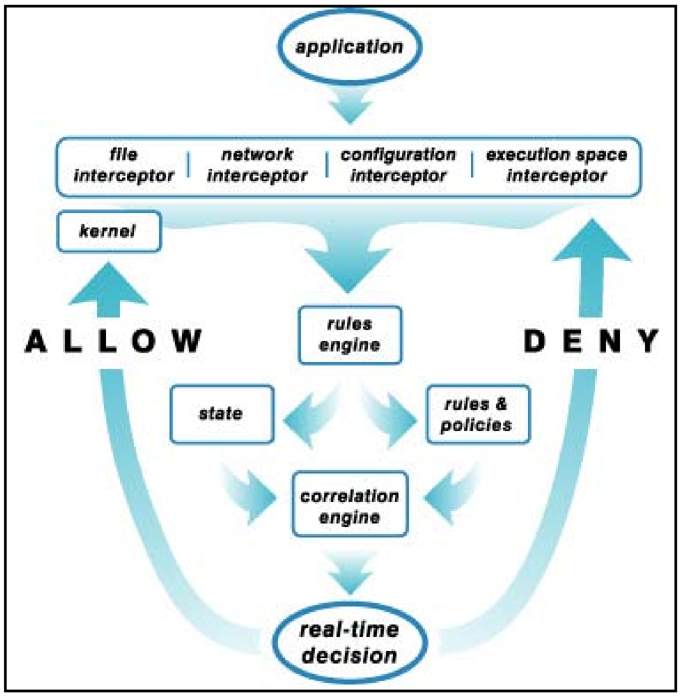


Figure 1-3. OKENA's INCORE™ architecture.
The execution space interceptor is worth spending some time explaining. While this does not refer directly to a specific resource like file or network, it deals with maintaining the integrity of each application's dynamic runtime environment. Requests to write to memory not owned by the requesting application will be detected and blocked by this interceptor. Attempts by one application to inject code (for example, by injecting a shared library like a DLL) into another process will also be detected and blocked. For example, the getadmin exploit on NT used a DLL injection technique to escalate the user's privilege level – StormWatch would block this attack. Lastly, buffer overflow attacks are detected by this interceptor. The result is that not only is the integrity of dynamic resources like file system and configuration preserved, but the integrity of highly dynamic resources like memory and network I/O are also preserved.

Because StormWatch intercepts the File, Network, Configuration and Run-Time operations and compares them to policies, StormWatch is able to track the state of that application. Combinations and sequences of file, configuration, and network operations constitute the application's behavior. When an application attempts an operation, StormWatch checks the operation against its policy and also correlates the policy for this operation against the application's maintained state. This enables the agent to make real-time allow or deny decisions within the context of the overall behavior and reduce the number of false positives associated with traditional, non-correlating behavior blocking schemes.

A StormWatch policy is a collection of rules that IT assigns to each server and desktop. These **application-centric access control rules** (the rules are not based on users or IDs) provide safe access to required resources. OKENA provides policies that enterprises can implement out-of-the-box or use as models for customized policy development.

StormWatch immediately provides important intrusion detection and prevention features for servers and distributed firewall functionality. These out of the box solutions provide protection against classes of vulnerabilities as well as policies for common applications such as Microsoft SQL Server, Office, Instant Messenger and IIS Web servers. These policies can and should be deployed immediately with minimal configuration to protect your most critical servers and desktops.

How INCORE makes technology converge and StormWatch unique

Looking at INCORE, we can see how StormWatch can truly act as a Best-of-Breed Intrusion Detection/Prevention agent, and a file integrity monitoring agent, and an application sandbox. Combining use of the interceptors (based on the policy defined on the Manager) lets us create the needed security application. The ability to "Mix and Match" interceptors – based on centrally defined policy rules – allows OKENA to rapidly implement new security capabilities.

Desired Security Capability	Network Interceptor	File System Interceptor	Configuration Interceptor	Execution Space Interceptor
Distributed Firewall	✓			
Intrusion Detection	✓	✓		✓
Application Sandbox		✓	✓	✓
Network Worm Prevention	✓	✓		✓
System Hardening		✓	✓	

In reality, the default policies that ship in StormWatch implement all of these security applications (Distributed Firewall plus IDS plus application sandbox etc), so customers don't have to create their own policies. Of course, users can create or change policies easily via the GUI, but the default policies provide all of these protections at once.

Because INCORE offers different interceptors, StormWatch is able to deliver many different security capabilities.By using particular combinations of the INCORE interceptors, we can effectively "create" traditional security capabilities. For example, since traditional Distributed Firewall products deal with address and port blocking, the Network interceptor will provide this capability. Operating System Hardening or Integrity Enforcement looks for changes to critical files or registry keys, so these two interceptors will give us a system hardening capability

StormWatch Management Architecture

StormSystem uses an agent-Manager (server) architecture, where policy is created and modified on the manager and automatically distributed to all agents. The Manager uses a secure web interface to the management GUI, allowing management from anywhere in the enterprise without the use of insecure remote access methods. Agents poll the manager at periodic intervals for policy changes or new software updates, and send alerts to the Manager in real-time. All agent-manager communications is secure and uses standards based protocols (secure web, SSL, https, TCP port 443).

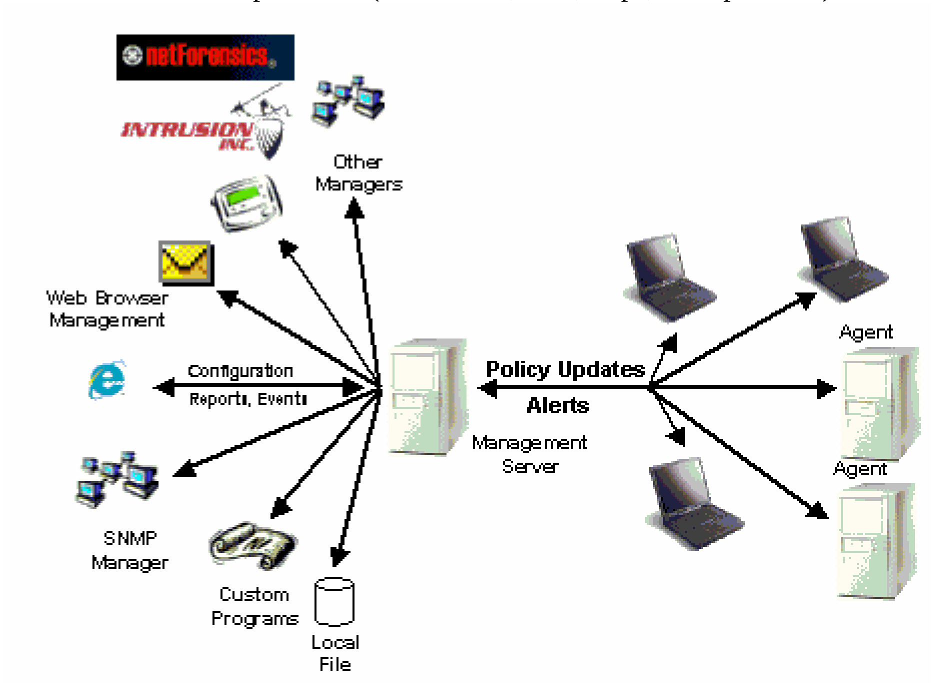


Figure 1-4. StormWatch Management Architecture.

The backend alert system is flexible. Events that are generated by agents can be sent via email, dial-up pager, SNMP trap, written to a local file, and/or cause a custom program to execute. Several OKENA customers have successfully integrated StormWatch with their SNMP-based network management systems such as Unicenter and OpenView, and StormSystem integrates with third party management consoles like netForensics and Intrusion, Inc.

StormWatch Correlation

Events are correlated locally on the agent, as well as globally on the Manager. This results in an extraordinary increase in accuracy when compared to signature-based Host IDS (HIDS) systems.
Local correlation greatly reduces the “false alarm” (technically, the ”False Positive”) problem that plagues HIDS. By analyzing a number of different actions, it is possible to identify malicious activity with very high precision. One example of this is Network Worm propagation. If we were to analyze each of the activities performed by the worm in isolation, everything appears benign: a file is downloaded; the file executed; the process opens the Outlook address book; the process sends email. Each of these is normal, valid behavior that is seen many times a day. Put together in sequence, the behavior becomes sinister and destructive. By correlating these events, the agent is able to block worm activity without alerting many times a day. A Host IDS that did not correlate these events would have to generate alerts like “Application A opened the Address Book” or “A program was downloaded and ran”. Each of these alerts would have to be analyzed by the human operator to sort normal events from actual attacks.

Trojan Horse detection is another example, where activity like hooking the keyboard, communicating via IRC, and other activity is analyzed in a broader context to detect malicious activity without generating excessive false alerts. Similarly, the ability of the agent to classify applications based on their behavior is a key capability in controlling them. For example, StormWatch will consider a program as an "Email Client" if it acts like an email client (uses outgoing connections for POP, IMAP, or SMTP, for example). There is no need to hard code executable names (OUTLOOK.EXE) into a list to be able to control them with rules like "Email clients cannot make outbound HTTP connections, because corporate policy prohibits 'web bugs' 'wiretapping' email sessions".

Global correlation is similar, but correlates events received from many different agents. By looking at events across the enterprise, attacks that might have been missed will be detected (we call this a "False Negative" situation, and is generally considered to be much worse than False Positive). Attackers who send only a few (sometimes only one) packet to each host in the enterprise have traditionally been able to map the entire network while evading detection. Using global correlation, these "distributed scans" are automatically detected by the Manager. If several agents detect a common program trying to propagate via email, the Manager will add the program to the Global Quarantine List. When agents poll, they will receive the updated quarantine list. Even if they have not yet been attacked by the worm, the worm's executable files will be placed "off limits".

The StormWatch Return On Investment

What makes StormWatch different from other security solutions is that it's a truly proactive approach. Stopping the unknown attack is much harder than stopping the known and everyday attack, yet it is the unknown attack that causes the most damage and requires the most effort to recover.

Organizations looking to secure their systems using multiple products (for example, a Distributed Firewall and Tripwire) will find that StormWatch is much less expensive to buy (one product, not two), much easier to deploy (one agent, not two), and much easier to operate (one product to train staff, one console to monitor).

Since StormWatch also protects servers and desktops, customers protecting both will experience further operational economies of scale. And since StormWatch protects not only Windows, but Unix systems as well, there is yet another layer of operational cost savings. Protecting Unix and Windows servers and desktops with other products (Distributed Firewalls, Host IDS, malicious code sandbox, audit consolidation tools) could take a dozen different products – or you could use StormWatch.
Lastly, there are no signatures to test and deploy. Host IDS systems with frequent updates would result in administrators continually deploying new updates in the test lab to ensure compatibility with production servers. This requirement for high, ongoing personnel cost is totally absent with StormWatch. Its " Zero Update" architecture eliminates the admin burden of HIDS signature updates, testing, and possible roll-backs.

Application-centric intrusion prevention represents a new category or philosophical approach toward protecting today's expansive networks and enterprise environments. Traditional IDS technologies can actually impose barriers and intrusive procedural stumbling blocks to network and user activity, which impact performance and ultimately result in undue administrative burdens and frustration. Behavior enforcement strategies within the context of intrusion prevention security take a holistic approach toward the evolution of attacks. The result is a proactive defense based on application behavior that prevents damage caused by network and file attacks

More Information

For more information about how StormWatch protects against variety of attacks and malicious intrusions, please review the Frequently Asked Questions document, located on the OKENA Web site (**www.okena.com**) under the Products section heading.

Or, please contact OKENA for an exclusive presentation which further illustrates a true, real-world attack scenario that took advantage of the holes left by traditional security technologies. Call OKENA at (781) 209-3200 or (301) 896-9388 to schedule a personal demonstration. Or log into **www.okena.com** and click on OKENA's Webinar on the OKENA home page.