OKENA

71 Second Ave., 3rd Floor

Waltham, MA 02451

Phone 781 209 3200

Fax 781 209 3199

StormWatch® 3.0

Evaluator's Guide

"Signature-based detection methods, which are already showing signs of extreme strain under current malicious code trends, will not be able to keep up with the new set of malicious-code risks created by the pervasive adoption and use of Web services and active content."

—John Pescatore and Arabella Hallawell, Gartner Research Note, August 31, 2001

"Look to Intrusion Prevention to cure the ills of Intrusion Detection."

–Richard Stienon, Gartner Group, "Security at the Speed of E-Business"

Contents

Introduction to the Evaluator's Guide

Welcome to the OKENA StormWatch 3.0 Evaluator's Guide. This guide is designed to help you understand the product design goals for StormWatch 3.0, as well as how to use StormWatch to establish a proactive security solution for your mission-critical business applications.

The Evaluator's Guide is recommended for any user who wishes to install StormWatch for the purpose of reviewing and evaluating the product. Additional configuration information is available (StormWatch Getting Started Guide and StormWatch Configuration Guide) for a more wide scale deployment on your StormWatch product CD.

The Guide consists of two parts:

- An introduction to StormWatch that provides background information and a high level overview of the product. This will help introduce StormWatch's unique security approach to new readers.
- A set of tests that allows evaluators to put the product through its paces, by running a broad range of real-world tests and observing the output.

By the end of the Guide, you should be familiar with the capabilities offered by StormWatch, and have "hands on" experience installing, configuring, and managing the product.

Behavior-based security

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 is an exponential 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).

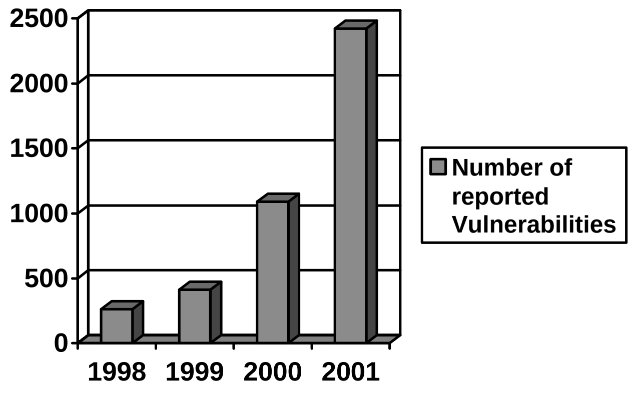


Figure 1-1. Increasing number of Vulnerabilities.

Signature-based security products will have to have a signature each of these to protect against attacks. Over the course of 2001, this would require 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.

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.

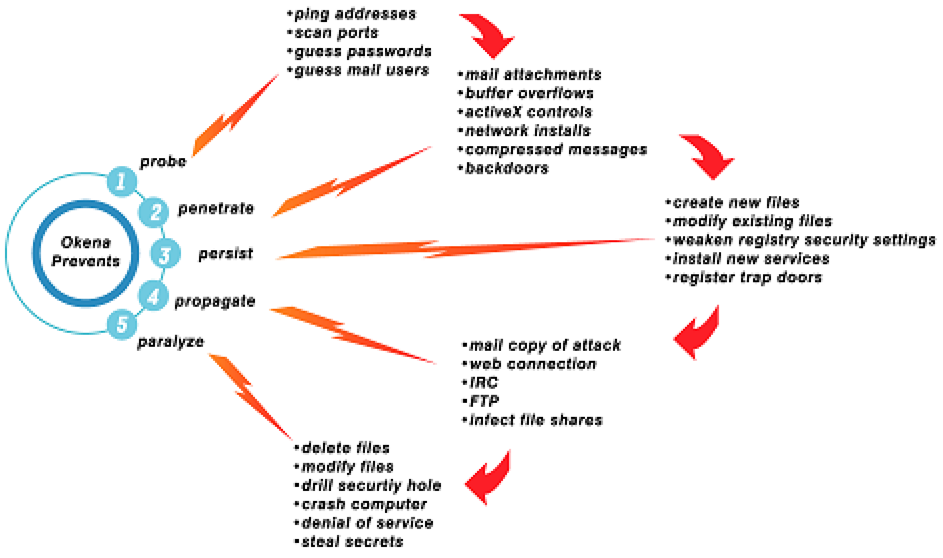


Figure 1-2. The Attack Lifecycle.

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 do – 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". If you look for attacks in the final three stages of the lifecycle, attacks will look very much like what we've seen for the last 15 years.

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 Overview

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. Our 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, our 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 their 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. 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.

[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.

StormWatch does not rely on, or even contain any signatures. Security is provided by policy rules defined on the Manager and distributed to the agents. All rules can be inspected, and the Manager 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 slide, 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.

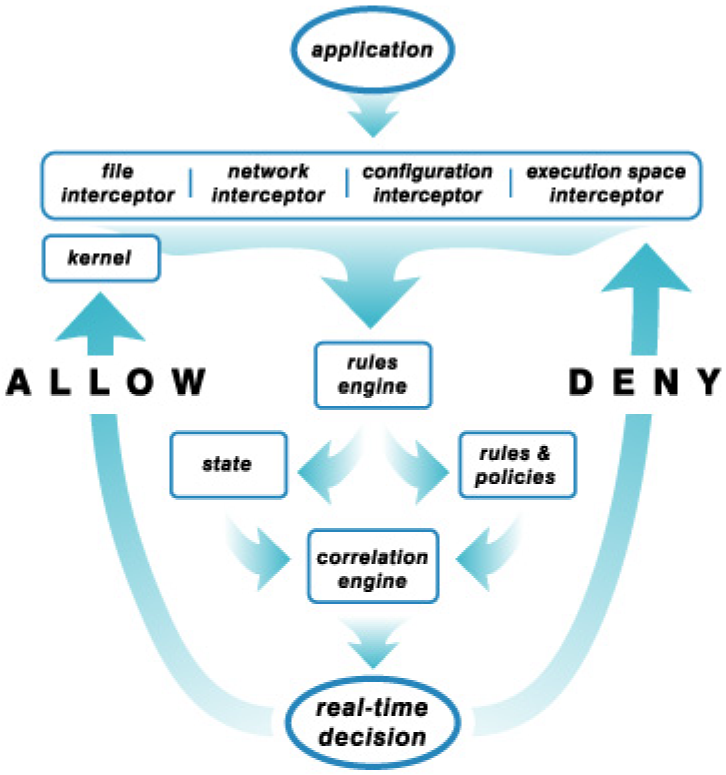


Figure 1-3. OKENA's INCORE architecture.

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.

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.

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

Because INCORE offers so many 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).

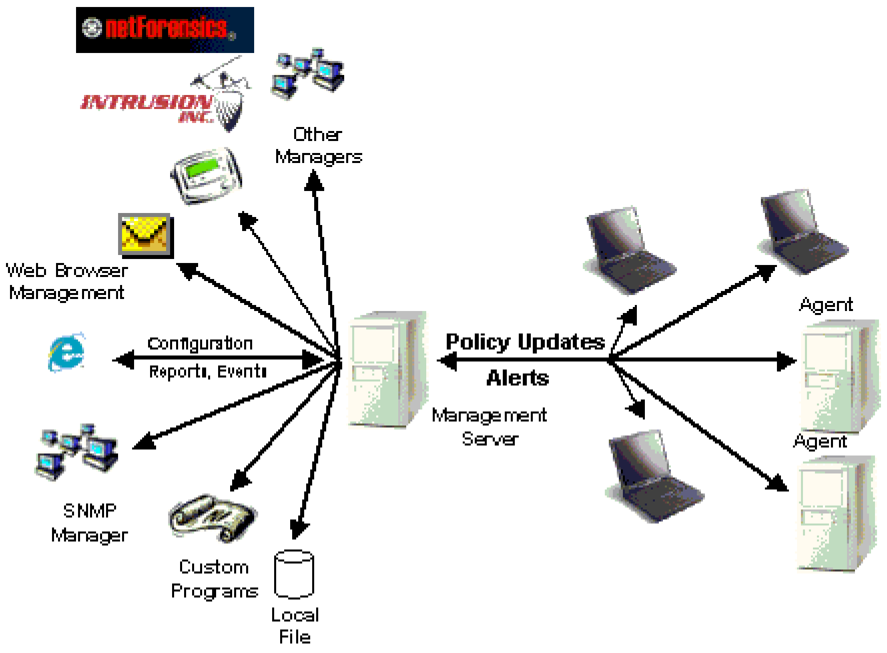


Figure 1-4. StormWatch Management Architecture.

The backend alert system is flexible. Events that are generated by agents can be sent via email, dial-up pageer, 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.

Evaluating StormWatch

This section of the document describes five distinct and concrete steps that you can take to see the power and ease of use that StormWatch offers. Each of these steps stands by itself, but build sequentially on one another. Doing each of these steps in sequence will provide a very complete, in-depth analysis and demonstration of the product.

Test 1: Systems protected within 30 minutes

StormWatch is very easy to install, typically taking no more than 30 minutes from the time that the StormSystem CD is placed in the CD drive to when agents are deployed and protecting remote systems. In fact, it is not even necessary to log into the Manager to have agents running on and protecting server or desktop computers. Agents proactively stop attacks and prevent misuse autonomously, and do so as soon as they are installed.

Other resources you might want to get:

- **StormWatch Getting Started Guide**, found on the StormSystem CD.
- **Various StormWatch policy documents**, found in the *PolicyDocs* folder of the StormSystem CD. OKENA provides PDF files describing each shipped policy categorized by the pre-configured group it is attached to. These documents explain the methodology used to create the policies shipped with StormWatch and may be useful in helping you to understand how StormWatch policies are crafted. They can also be used as a guide in helping you to develop your own policies.

What you need to do this test:

- **A computer to install the StormWatch Manager software on.** The StormWatch Manager is supported on Microsoft Windows 2000 server, with Service Pack 2 installed. Note that if only a few agents will be installed during these tests, it can be installed on Microsoft Windows 2000 Professional.
- **One or more computers to install StormWatch agents on.** For this test, it is recommended that Microsoft Windows server or desktop computers are used (either Windows NT 4.0 or Windows 2000). We will test Unix agents in a later step.
- **A computer that you can use to attack the StormWatch protected systems.** You may or may not want to do this, but this document offers some recommendations on tools you can use to attack your computers. Note that some tools run under Windows, while others run under Linux. You may need two computers, or one that will boot either Operating System.
- **IP communications between these systems.** Communications between agents and servers use secure HTTP (port 443). It is helpful to have access to the Internet to download exploits to exercise StormWatch's protection as well.
- **A valid software license key from OKENA.** Contact your local OKENA representative, or send an email message to sales@okena.com to get a key for your evaluation.

Things to keep in mind:

- The computer on which you are installing the StormWatch Management Console software should be placed in a physically secure, locked down location with restricted access.

- Do not install any software on the management console system that is not required by the product itself.
- You must have administrator privileges on the system on which you are installing the StormWatch Management Console to perform the installation.
- The management console system must have a static IP address.

Step 1. Install the StormWatch Manager.

Place the CD in the CD drive of the computer that will be the StormWatch Manager. If you have CD autorun enabled in the computer, the Manager installation will start automatically; if not, you will need to browser to the CD using Windows Explorer, and execute **setup.exe**.

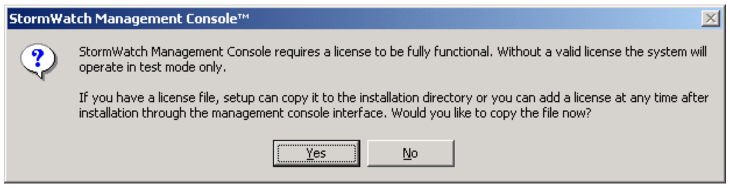
The setup program will prompt you for standard information like directories to use to install the software. It will also install and configure the database (Microsoft SQL Server runtime engine) and a web server (Apache) for use by the Manager. StormWatch configures these components for you.

Careful! The StormWatch Management Console cannot function properly if you have any other web server software running on the management console system. Having multiple web servers running on the same system causes port conflicts. If you already have a web server running on this computer, the StormWatch Manager will not be able to use the web server ports (80 and 443). The installation will abort, and you'll get a message that StormWatch cannot install the web server. If this happens, you will have to uninstall the existing web server and then rerun the setup StormWatch program, or find a new computer to install the Manager on – one that doesn't have a web server.

Tip: If you already have Microsoft SQL Server 2000 installed on the Manager computer, StormWatch will configure and use that, rather than install the runtime engine. While StormWatch will happily use SQL Server 2000, SQL Server version 7 is not supported. The StormWatch Manager installation will abort, and tell you that you need to uninstall SQL 7

Careful! You have to be logged on as administrator to install the agent kits.

Figure 1.1 shows the screen that you will see during the installation where StormWatch asks you for your license key. You can enter it directly, or browser to a file containing the key.


StormWatch Management Console™

StormWatch Management Console requires a license to be fully functional. Without a valid license the system will operate in test mode only.

If you have a license file, setup can copy it to the installation directory or you can add a license at any time after installation through the management console interface. Would you like to copy the file now?

Figure 1-1. Entering the License Key during installation.

Careful! If you don't have a key, StormWatch will function in *Intrusion Detection Mode*. Attacks will be detected, but not blocked. Adding a key later will let you change from IDS mode into protect mode. If you enter a license key when you install, agents will protect by default.

Step 2. Install Agent Kits on the computers you want to protect.

Now you need to log into the Windows desktop or server computer(s) that you want to protect. The easiest way to install agent kits is to use a web browser to directly retrieve the kit from the Manager. Since the Manager uses a web server for remote access, and since the agent kits are small, you can easily download and install the appropriate kit on the remote computer.

In your browser, type the URL of the Manager computer, e.g. *http://mystormwatchmanager.mydomain.com*. If you do not have DNS configured, you will need to type the Manager's IP address in the browser URL window, e.g. *http://10.0.1.17*. You can use either Internet Explorer or Netscape browsers to do this.

Figure 1.2 shows you the main web page. While you have the choice to log in, let's leave that for later. StormWatch allows you to protect agent computers without even logging in, so let's go straight to installing agent software. Click on the button that says "Agent Kits".

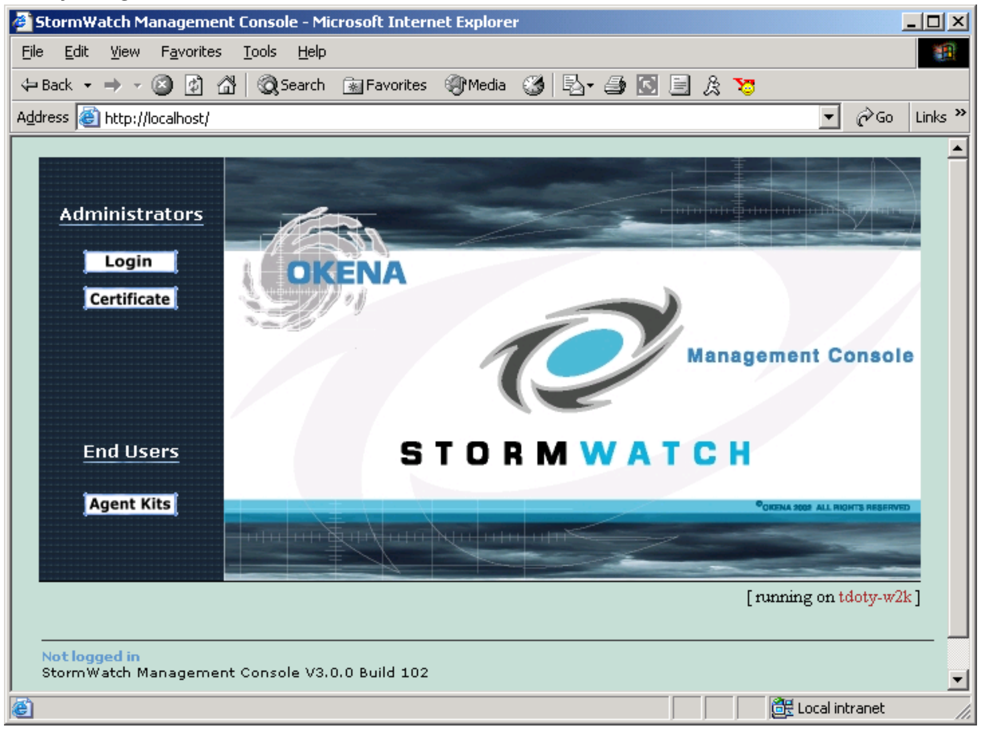

StormWatch Management Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/
Go Links
Administrators
Login
Certificate
End Users
Agent Kits
OKENA
Management Console
STORMWATCH
©OKENA 2002 ALL RIGHTS RESERVED
[running on tdoty-w2k]
Not logged in
StormWatch Management Console V3.0.0 Build 102
Local intranet

Figure 1-2. Getting Agent Kits.

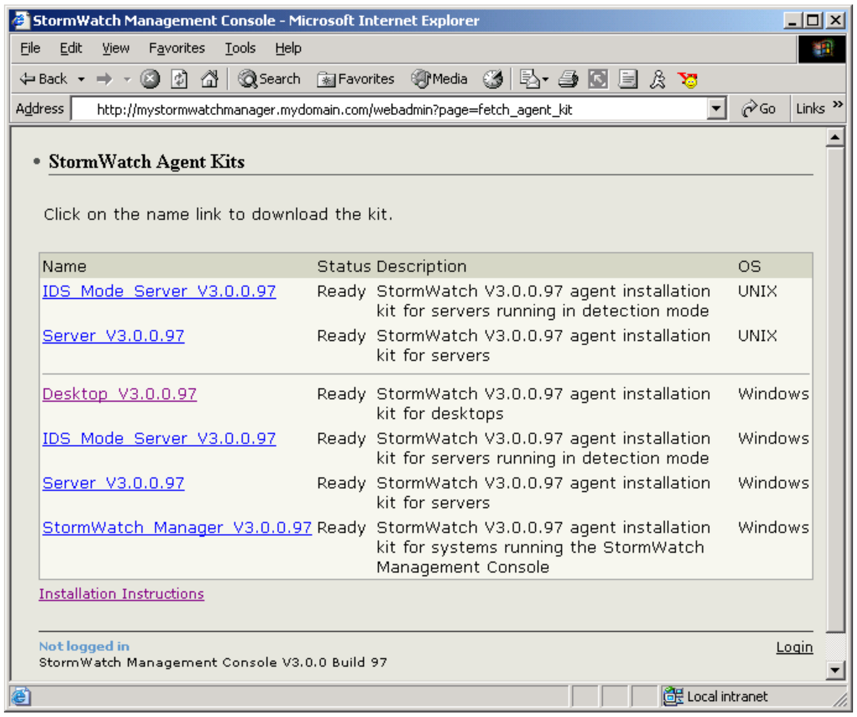
There are three types of agent kits that are automatically generated during the Manager installation:

- **Servers.** The security policy for these kits is optimized for server-class machines, running server-class applications. These kits provide the protection that you would want on most, if not all, of your servers. There is a different agent kit for Unix and Windows servers – Unix agent kits are on the top of the screen, and Windows agent kits are on the bottom. You can choose to install the agent kit in IDS Mode, by clicking on the "IDS Mode Server" kit.

- **Desktops.** This is the agent kit that is optimized to protect desktop computers running desktop applications. These kits provide the protection that you would want on most, if not all, of your desktops. Note that this agent kit is offered only for Windows computers.

- **StormWatch Manager.** This agent kit is optimized to protect the StormWatch Manager itself. OKENA strongly recommends that you install this agent kit on every StormWatch Manager.

Figure 1.3 shows the agent kit download screen. Right now, you'll want to click on the appropriate agent kit (server or desktop) that you want to install. It's fine to run the install straight from the Manager, or you can save the agent kit locally and run it from the local hard drive. Note that no interaction is required during the installation – the agent automatically does all local setup, and automatically registers itself with the Manager.


StormWatch Management Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://mystormwatchmanager.mydomain.com/webadmin?page=fetch_agent_kit
Go Links

StormWatch Agent Kits

Click on the name link to download the kit.

Name	Status	Description	OS
IDS Mode Server V3.0.0.97	Ready	StormWatch V3.0.0.97 agent installation kit for servers running in detection mode	UNIX
Server V3.0.0.97	Ready	StormWatch V3.0.0.97 agent installation kit for servers	UNIX
Desktop V3.0.0.97	Ready	StormWatch V3.0.0.97 agent installation kit for desktops	Windows
IDS Mode Server V3.0.0.97	Ready	StormWatch V3.0.0.97 agent installation kit for servers running in detection mode	Windows
Server V3.0.0.97	Ready	StormWatch V3.0.0.97 agent installation kit for servers	Windows
StormWatch Manager V3.0.0.97	Ready	StormWatch V3.0.0.97 agent installation kit for systems running the StormWatch Management Console	Windows

Installation Instructions

Not logged in
StormWatch Management Console V3.0.0 Build 97
Login

Figure 1-3. Agent Kit Download Screen.

Tip: There's no need to distribute encryption keys to the agents. The agent kit contains everything that the agent needs to securely communicate with the Manager. Since it uses the SSL encryption capability that your Operating System contains, you probably have strong encryption installed already.

Tip: When you're done installing the agent kit, you'll have to reboot. The reason for this is that StormWatch hooks many different locations in the kernel. You'll only need to do this once.

Congratulations! A StormWatch agent is now protecting your computer. Now, let's test it. The best way to test it is to attack it.

Step 3. Attack your system.

The proof of any security product is in how well it protects. StormWatch is designed to provide unmatched protection "out of the box", without any configuration being required. However, testing this requires that you actually try to attack the computer.

There are many tools available on the Internet that you can find to test your security. Table 1-1 lists a small selection of reputable sites that we believe offer high caliber tools for this purpose.

Tool	*Site*	*Comments*
nmap	http://www.insecure.org/	Most sophisticated network mapping and discovery tool. Does an excellent job of identifying the Operating System of the target device. Very commonly encountered.
Nessus	http://www.nessus.org/	A free, Linux-based, Open Source vulnerability scanner. Contains a very large list of current exploits for both Unix and Windows systems.
Windump	http://windump.polito.it/	High quality, free network packet and password capture tool. Windows version of Unix tcpdump
Etherpeek	http://www.wildpackets.com/	Good commercial packet sniffer.
Silentlog	http://packetstorm.decepticons.org/Win/SilentLog.zip	Keystroke logger with source code.
Pwdump2	http://razor.bindview.com/tools/files/pwdump2.zip	Allows encrypted password hashes to be dumped, even if the Windows 2000 system is protected (running SYSKEY)
Firehole	http://keir.net/firehole.html	Personal Firewall testing tool that uses DLL Injection
netcat	http://www.atstake.com/research/tools/	Among other features, can act as a remote login server on any port.
Command Shell	%Systemroot%\system32\cmd.exe (Windows) /bin/sh (Unix)	Lets you run commands from a command line

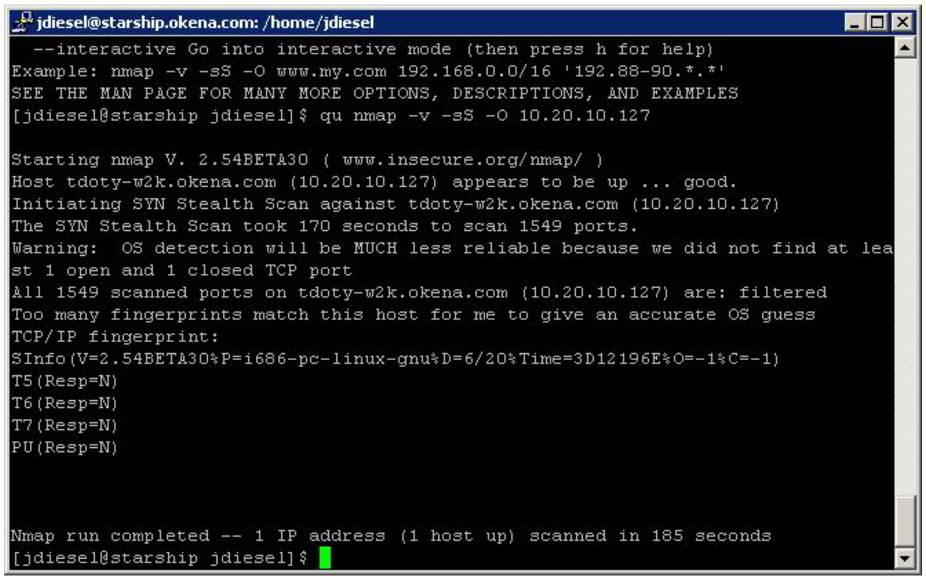
Table 1-1. Tools you can use to test StormWatch protection.

Each of these tools does a different task, and is used for a different purpose. The following sections show the purpose for each of the tools (i.e. what the tool does when StormWatch is not protecting the target), and the expected outcome when StormWatch is protecting the target.

Step 3a. Scanning with nmap.

nmap is a tool used to identify which devices are present on a network, and what Operating System and services they are running – indeed, the name stands for **N**etwork **Map**per. nmap works by sending a series of network probes to the target; the fact that the target responds identifies that it is there (and also which ports it is running), and the pattern of error messages returned identifies the OS. nmap is surprisingly accurate in identifying targets. It is frequently used at the initial stage of an attack or investigation, to determine what systems might respond to an attacker's exploits.

Expected outcome of nmap scans against StormWatch-protected systems: nmap is unable to identify the target operating system of systems running the default server or default desktop policies. nmap scans will appear to hang while its security tests timeout. nmap scans against systems not protected by StormWatch will report results very quickly. Figure 1-3 shows a screenshot of an nmap scan against a Windows system protected by StormWatch.



```
  --interactive Go into interactive mode (then press h for help)
Example: nmap -v -sS -O www.my.com 192.168.0.0/16 '192.88-90.*.*'
SEE THE MAN PAGE FOR MANY MORE OPTIONS, DESCRIPTIONS, AND EXAMPLES
[jdiesel@starship jdiesel]$ qu nmap -v -sS -O 10.20.10.127

Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Host tdoty-w2k.okena.com (10.20.10.127) appears to be up ... good.
Initiating SYN Stealth Scan against tdoty-w2k.okena.com (10.20.10.127)
The SYN Stealth Scan took 170 seconds to scan 1549 ports.
Warning:  OS detection will be MUCH less reliable because we did not find at lea
st 1 open and 1 closed TCP port
All 1549 scanned ports on tdoty-w2k.okena.com (10.20.10.127) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-linux-gnu%D=6/20%Time=3D12196E%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)


Nmap run completed -- 1 IP address (1 host up) scanned in 185 seconds
[jdiesel@starship jdiesel]$
```

Figure 1-3. Scanning a Windows system with nmap.

Careful! Disabling the "Cloak System" option (enabled by default in the *Default Server* and *Default Desktop* policies) will allow nmap to gather much more information.

Step 3b. Scanning with Nessus.

Nessus is an Open Source vulnerability scanner that runs on a Linux computer. It makes network connections to remote systems and runs many hundreds of security tests against them. Nessus is one of the few scanners that relies extensively on exploits to determine vulnerability – it typically does not rely on any "banner" information passed back from the remote service (e.g. "Sendmail 8.8.2 ready"), but instead actually tries to break into the service.

Nessus is well regarded in the security industry, and is frequently updated with new tests. It is frequently used to perform network security audits.

Expected outcome of Nessus scans against StormWatch-protected systems: Nessus will not detect any vulnerabilities in systems running the default server or default desktop policies. Nessus status screens will be blank, and its reports will show no vulnerability information. Nessus scans against systems not protected by StormWatch will typically report large numbers of "holes", "warnings", or "notes".

Careful! Disabling the "Cloak System" option (enabled by default in the *Default Server* and *Default Desktop* policies) will allow Nessus to gather much more information.

Step 3c. Capturing packets with Windump or dsniff.

Packet capture programs are sometimes referred to as packet "sniffers", after the popular Sniffer product sold by Network Associates. These utilities are used for a wide range of uses, from legitimate network troubleshooting to the much more shady password theft.

Windump is a Windows version of the popular tcpdump packet capture and analysis utility. Since it uses the tcpdump data format, add-on utilities that process tcpdump logs will also process Windump logs. This makes Windump a useful and popular packet capture utility. Dsniff is another popular packet capture program.

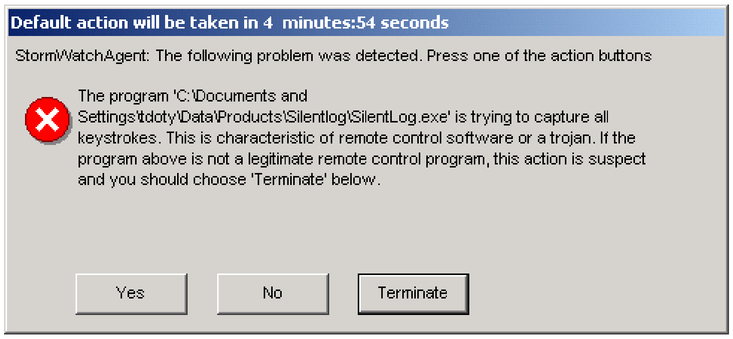
Expected outcome of running Windump on StormWatch-protected systems: StormWatch detects applications that perform unusual interactions with the NIC (more specifically, with the NDIS interface in Windows or with streams in Unix). These events will be intercepted either at boot time (Windows) or in real-time (Unix).

Tip: Since Windump interfaces with the NIC at boot time, you'll have to reboot the computer after installing Windump if you want to test StormWatch's capabilities.

Step 3d. Capturing keystrokes with Silentlog.

Silentlog is a "keystroke logger" program that silently captures all keyboard input and logs them to a file. Attackers often install keystroke loggers to capture passwords entered by users. Many Trojan horse programs include keystroke logging as a feature.

Expected outcome of running SilentLog on StormWatch-protected systems: StormWatch will generate a message saying that an attempt is being made to capture all keystrokes. The user will be able to select "Yes" (allow this action to take place), "No" (block this action but let the program continue to run), or "Terminate" (terminate the application that caused this event). By default, StormWatch will terminate the application trying to capture keystrokes if no selection is made within 5 minutes. Figure 1-4 shows this message box.


Default action will be taken in 4 minutes:54 seconds

StormWatchAgent: The following problem was detected. Press one of the action buttons

The program 'C:\Documents and Settings\tdoty\Data\Products\Silentlog\SilentLog.exe' is trying to capture all keystrokes. This is characteristic of remote control software or a trojan. If the program above is not a legitimate remote control program, this action is suspect and you should choose 'Terminate' below.

Figure 1-4. User queried about keyboard sniffing.

Tip: You can install SilentLog anywhere, on any drive, under any name. You could, for example, install it under the name IEXPLORE.EXE (the name used by the Internet Explorer web browser). What is important to StormWatch is not the name, but the action that the application takes.

Tip: Some programs trap keyboard input as part of their normal operations. The AOL Instant Messenger and Yahoo Messenger instant messenging clients are two examples of this.

Step 3e. Hijacking applications via DLL Injection.

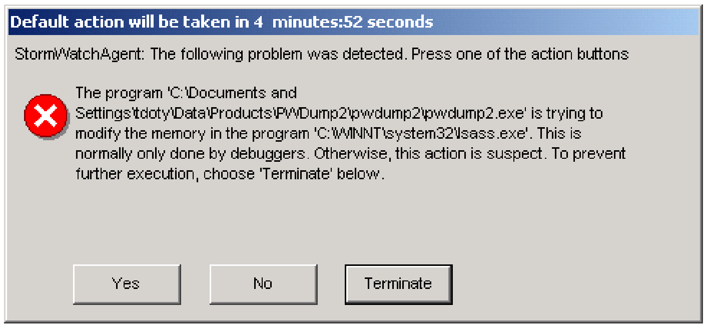
Common software routines are stored in collections called *libraries*. Microsoft Windows provides a built-in ability to load these libraries as required – this capability is called *Dynamically Linked Libraries*, and is abbreviated DLL. One form of attack is to insert (or "inject") a new and malicious DLL into a running application. This attack is called "DLL Injection."

A popular attack program that uses DLL injection is PWDUMP2. PWDUMP2 injects its code into an application that runs with elevated priviledge, in order to steal the encrypted password hashes saved in the Registry. These hashes are then analyzed by password cracking routines like Crack, l0phtcrack, or John the Ripper.

Another application that uses DLL injection is FIREHOLE, which is used to test whether personal firewall applications can "leak" - whether applications can make unauthorized outgoing connections.

Expected outcome of running PWDump2 or Firehole on StormWatch-protected systems: StormWatch detects applications that inject code into other running applications. As with our keyboard sniffing example, the user will see a pop-up box that identifies the application and asks whether the user wants to let this activity occur, block the application, or terminate the offending application. Figure 1-5 shows this dialog box. Note that while Figure 1-5 shows the result of running PWDump2, you would see a similar message if you used Firehole.

Tip: There are very few legitimate programs that use this technique. If you see activity like this – especially from downloaded programs – you should be extremely suspicious.


Default action will be taken in 4 minutes:52 seconds

StormWatchAgent: The following problem was detected. Press one of the action buttons

The program 'C:\Documents and Settings\tdoty\Data\Products\PWDump2\pwdump2\pwdump2.exe' is trying to modify the memory in the program 'C:\WINNT\system32\lsass.exe'. This is normally only done by debuggers. Otherwise, this action is suspect. To prevent further execution, choose 'Terminate' below.

Figure 1-5. User queried about DLL Injection activity.

Step 3f. Replacing critical portions of the Operating System (rootkit attacks).

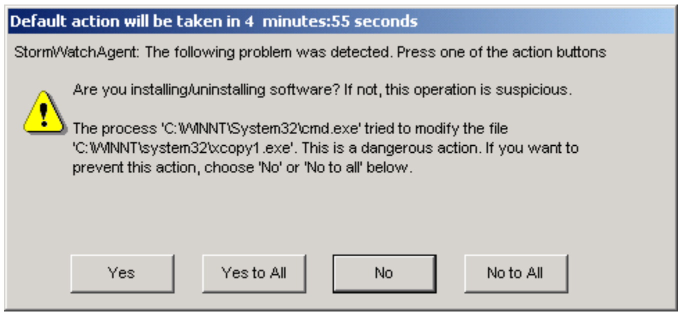
One of the classic attacks against Unix systems (dating from the 1980s or before) was to replace critical Operating System (OS) binaries. For example, the routine used by most Unix systems to authenticate users during login is the program /bin/login. Replacing this program with a different one that stores the passwords in a secret file is the classic subversion attack against Unix.

Windows stores many of these routines in the SYSTEMROOT, which is typically found in the directory C:\WINNT\SYSTEM32. Not only are there executable routines here, but DLL libraries as well.

Many malicious programs attempt to modify or replace these programs. For our example, you can use a command shell (MS DOS Prompt) to manually replace or copy files. You can run a command shell from the start menu by selecting "Run" and typing "cmd.exe" in the dialog box. In the command shell, type "CD %SYSTEMROOT%" and then "cd system32" to change to your OS directory.

All executables and libraries in this directory are protected. For testing purposes, we will use a relatively unimportant one, XCOPY. Rather than deleting it, we will simply copy it to another filename. At the command prompt, type "copy xcopy.exe xcopy1.exe". When you see the query message, choose "No".

Expected outcome of OS replacement attacks against StormWatch-protected systems: You will see a query popup asking whether this activity is desired. As with other queries of this type, you can allow, deny, or terminate the activity. Figure 1-6 shows this query.


Default action will be taken in 4 minutes:55 seconds

StormWatchAgent: The following problem was detected. Press one of the action buttons

Are you installing/uninstalling software? If not, this operation is suspicious.

The process 'C:\WINNT\System32\cmd.exe' tried to modify the file 'C:\WINNT\system32\xcopy1.exe'. This is a dangerous action. If you want to prevent this action, choose 'No' or 'No to all' below.

Figure 1-6. User query about OS replacement activity.

Careful! We tried a trivial modification of the OS. You can try to actually delete critical files like SYSTEM.EXE or LSASS.EXE, and StormWatch will protect you. However, you should make sure that StormWatch is in **protection mode**, and that you are running the default protection policy (either *Default Servers* or *Default Desktops*). If you have disabled StormWatch, this will damage the Operating System.

Tip: There are times when users will want to add executables or DLLs to the system32 directory – for example, when installing device drivers. Query messages will allow users to install drivers, but prevent silent malicious installation of attack programs. That is why the query box asks if the user is installing software.

Step 3g. Unknown servers listening on high-number ports (backdoors).

A popular way to be able to connect to other systems is via an application listening (acting as a network server) on hard-to-find high number ports. If the attacker knows that he can connect to port 53,962, it is unlikely that most security defenses will be watching for this type of attack. *Backdoor* programs like this are included in most Trojan Horse applications.

netcat is a utility that can be used for many purposes, such as connecting to applications across the network and sending arbitrary data streams to it. It also provides the ability to listen (act as a server) on any port that you like (we've chosen port 23000 in the example below).

```
nc -l -p 23000 -t -e cmd.exe
```

Attackers would start a netcat listener on a high level port, and then use netcat (or telnet) on a remote system to connect to the listener:

```
telnet target.ip.address 23000
```

Expected outcome of unknown servers running on StormWatch-protected systems: The remote login session will be unable to connect to the netcat listener. The attacker may see a “Connection refused” message.

Careful! Connecting to yourself – for example, “telnet localhost 23000” – will not trigger StormWatch. As it turns out, many applications use this internal localhost address to communicate. StormWatch allows all traffic that is both generated and received locally. You have to try to break into the netcat backdoor remotely.

Key Point: Notice that until you actually run some attacks, StormWatch is very quiet. This is by intention: if your security system requires a lot of attention when you are **not** being attacked, all it is doing is making more work for you. StormWatch is designed not only to stop attacks, but also to provide a very low number of alerts for you to manage.

Test #1 Summary

We saw several key points in this test:

1. **StormWatch is installed easily**, with agents providing protection for servers and desktops "out of the box" in 30 minutes or less.

2. **StormWatch protects against a large number of attacks with the default policies.** There is no need to customize security policies to get protection, or even to log into the Manager.

3. **StormWatch does not generate a large number of alerts.** Unless you attack it, it will not give you many alerts that you have to manage.

Test 2: Protection is proactive – Stops the unknown attack

StormWatch differs substantially from traditional security products, in that it contains no signatures. Rather, it focuses on behavior, and blocks malicious or undesired behavior. By focusing on behavior, new and previously unknown attacks can be detected and blocked.

No new tests will be performed in this section – instead, we will analyze what happened in the tests we just ran, and determine that StormWatch protection will proactively stop undesired behavior.

What you need to do this test:

- **The StormWatch Manager used in the previous test.** The StormWatch agent(s) reported all malicious activity to the Manager. We will analyze these alerts in this test.

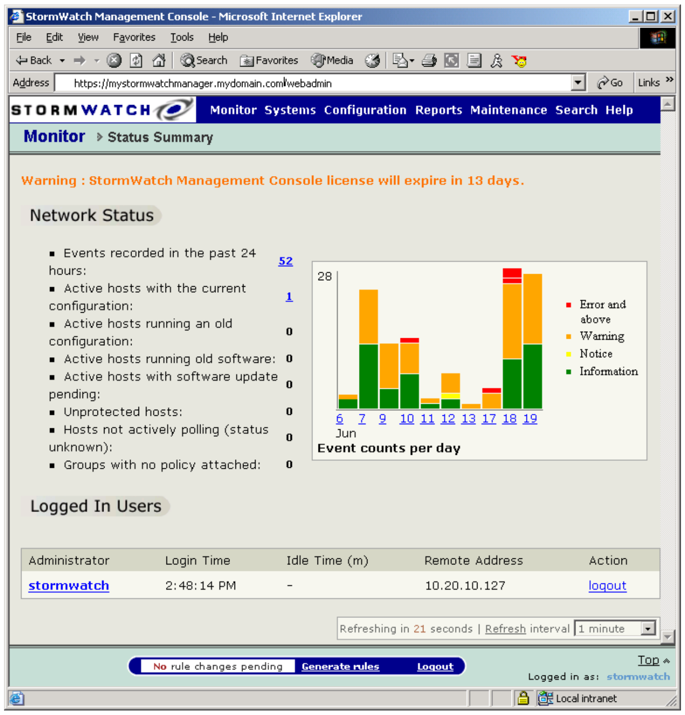
- **A computer to connect to the Manager.** The Manager user interface is provided in a Web Browser. You need a computer with a browser to access the Manager's user interface. Both Internet Explorer (5.x or higher) and Netscape (6.2) are supported. Note that you can use a browser on the Manager itself to connect to its user interface.

Step 1. Connect to the Manager's URL using the web browser.

Using your browser, connect to the Manager. If you are using a browser on the Manager computer, you can use the URL http://localhost. If you are using a browser on another computer, you will need to use a URL with the Manager's domain name or IP address. You'll see is a screen like what we saw earlier in figure 1-2.

Click the "Login" button, and type the user name and password that you defined when you installed the Manager. Note that your browser may notify you that it does not recognize the Manager's web server certificate. This is normal, since the Manager creates a unique certificate when it is installed. You can import the cert into your browser, but for now, it is fine to ignore this.

You will see the Status Summaryscreen shown in figure 2-1.


StormWatch Management Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://mystormwatchmanager.mydomain.com/webadmin
Go Links
STORMWATCH
Monitor Systems Configuration Reports Maintenance Search Help
Monitor Status Summary

Warning : StormWatch Management Console license will expire in 13 days.

Network Status

Events recorded in the past 24 hours:	52
Active hosts with the current configuration:	1
Active hosts running an old configuration:	0
Active hosts running old software:	0
Active hosts with software update pending:	0
Unprotected hosts:	0
Hosts not actively polling (status unknown):	0
Groups with no policy attached:	0

Logged In Users

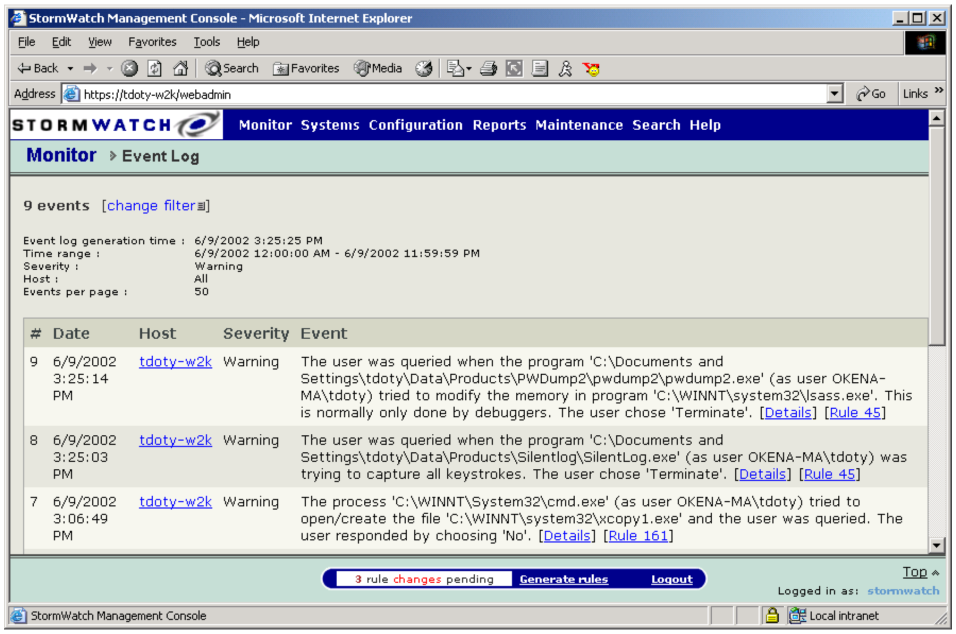
Administrator	Login Time	Idle Time (m)	Remote Address	Action
stormwatch	2:48:14 PM	-	10.20.10.127	logout

Refreshing in 21 seconds | Refresh interval 1 minute

Figure 2-1. StormWatch Status Summary Screen

Step 2. Open the Event Log.

The color coded Bar Graph contains a quick summary of the number and severity of the alerts that the Manager has received. As we can see, in this example, there are several high severity events. Clicking on the red portion of the bar takes us directly to the Event Log screen that provides a detailed breakdown of the events for each Severity Level, as shown in Figure 2-2.


StormWatch Management Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://tdoty-w2k/webadmin
Go Links
STORMWATCH
Monitor Systems Configuration Reports Maintenance Search Help
Monitor Event Log

9 events [change filter]

```
Event log generation time :  6/9/2002 3:25:25 PM
Time range :                 6/9/2002 12:00:00 AM - 6/9/2002 11:59:59 PM
Severity :                   Warning
Host :                       All
Events per page :            50
```

#	Date	Host	Severity	Event
9	6/9/2002 3:25:14 PM	tdoty-w2k	Warning	The user was queried when the program 'C:\Documents and Settings\tdoty\Data\Products\PWDump2\pwdump2\pwdump2.exe' (as user OKENA-MA\tdoty) tried to modify the memory in program 'C:\WINNT\system32\lsass.exe'. This is normally only done by debuggers. The user chose 'Terminate'. [Details] [Rule 45]
8	6/9/2002 3:25:03 PM	tdoty-w2k	Warning	The user was queried when the program 'C:\Documents and Settings\tdoty\Data\Products\Silentlog\SilentLog.exe' (as user OKENA-MA\tdoty) was trying to capture all keystrokes. The user chose 'Terminate'. [Details] [Rule 45]
7	6/9/2002 3:06:49 PM	tdoty-w2k	Warning	The process 'C:\WINNT\System32\cmd.exe' (as user OKENA-MA\tdoty) tried to open/create the file 'C:\WINNT\system32\xcopy1.exe' and the user was queried. The user responded by choosing 'No'. [Details] [Rule 161]

3 rule changes pending Generate rules Logout
Top
Logged in as: stormwatch
StormWatch Management Console
Local intranet

Figure 2-2. The Event Log screen.

The Event Log shows all of the alerts that were received by the Manager. Some of these may be pop-up query messages that the user saw; others may relate to events that end users never saw. Note that if the alert is from a user query message, the alert will specify which action the user took (*Allow*, *Deny*, or *Terminate*).

Lets look at the alert generated when we ran PWDUMP2. You'll see an alert that says something like the following:

> The user was queried when the program 'C:\Documents and Settings\tdoty\Data\Products\PWDump2\pwdump2\pwdump2.exe' (as user OKENA-MA\tdoty) tried to modify the memory in program 'C:\WINNT\system32\lsass.exe'. This is normally only done by debuggers. The user chose 'Terminate'. [Details] [Rule 45]

Key Point: Notice that the activity was denied. The rule did not alert us about an attack that possibly was under way and that we might want to look into. Rather, the attack was intercepted and terminated. The alert is interesting for the administrator only in a historical sense – ***there is no need to have a room full of security operators watching consoles, looking for things to turn red***. Even if the agent cannot communicate with the console, the system will be protected with the last policy that the agent received.

Step 3. Click through to the rule that caused this alert.

Now let's look at why we couldn't copy the XCOPY.EXE program. You'll see an alert that says something like the following:

> The process 'C:\WINNT\System32\cmd.exe' (as user OKENA-MA\tdoty) tried to open/create the file 'C:\WINNT\system32\xcopy1.exe' and the user was queried. The user responded by choosing 'No'. [Details] [Rule 161]

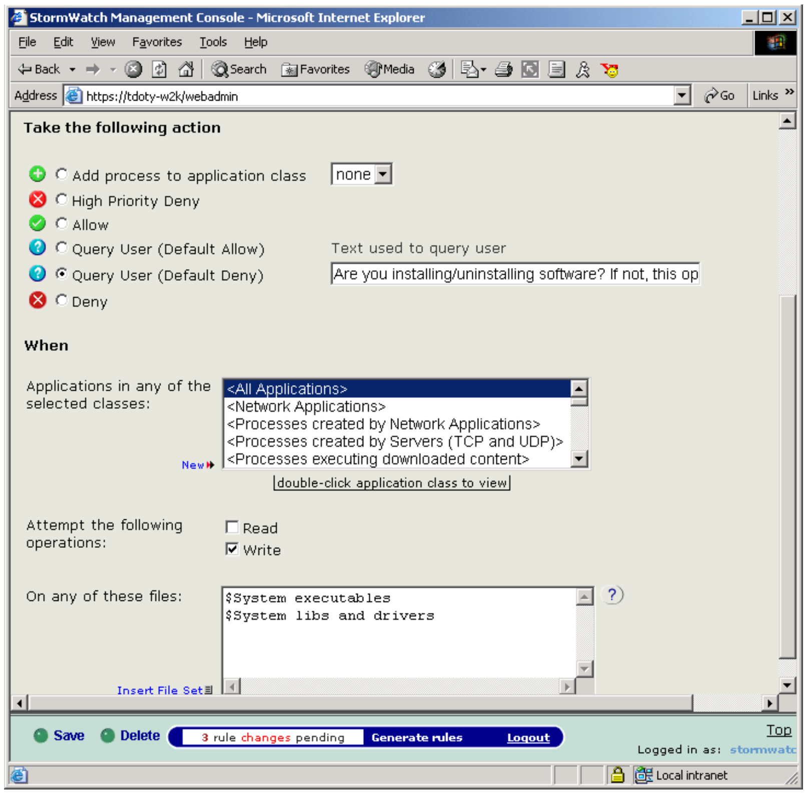
Let's take a look at rule 161, which controls this activity. Click on the link to the rule, shown in Figure 2-3. Note that we can tell several things from the rule screen:

- The user will be queried as to whether the activity is allowed or not.
- If the user doesn't make a selection, the default will be that the activity will be blocked.
- The rule will trigger when any application trying to write system executables, system libraries, or drivers.

Tip: Note that we have a couple pre-defined variables here: *System Executables* and *System libs and drivers*. OKENA has predefined these to relate to the specific files of these types for the various Operating Systems that StormWatch supports. These variables can be examined and modified if desired, but there is typically no need to do so.

Key Point: What is important to note here is that most of the rules that we've caused to trigger are generic in nature. Rather than a signature to look for a particular executable file, the rules look for ***types of activity*** that are not desired. What is important is not that cmd.exe (or a particular version of cmd.exe) that tried to modify a system executable; as you can see from the rule, ***any*** application trying to overwrite the Operating System will trigger ***exactly the same reaction***. The application could be a web browser that is executing malicious mobile code, or an email client program that executes a malicious attachment, or a web server attacked by a remote exploit. Since all applications are effected by this rule, the system is protected against all manner of attacks against the Operating System binaries.

This ability to block attacks based on behavior, rather than signatures, is the reason that we say that StormWatch is Proactive.


StormWatch Management Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://tdoty-w2k/webadmin
Go Links
Save Delete
3 rule changes pending
Generate rules
Logout
Top
Logged in as: stormwatc
Local intranet

Take the following action

- Add process to application class: none
- High Priority Deny
- Allow
- Query User (Default Allow)
- [✓ Query User (Default Deny)]
- Deny

Text used to query user: Are you installing/uninstalling software? If not, this op

When

Applications in any of the selected classes:
<All Applications>
<Network Applications>
<Processes created by Network Applications>
<Processes created by Servers (TCP and UDP)>
<Processes executing downloaded content>
New
double-click application class to view

Attempt the following operations:
☐ Read
☑ Write

On any of these files:
```
$System executables
$System libs and drivers
```
Insert File Set

Figure 2-3. The Rule Screen.

Test #2 Summary

We saw several key points in this test:

1. **StormWatch stops attacks.** All StormWatch alerts are of the form "I saw this activity which violates the policy you specified, so I stopped it". By the time the security administrator sees the alert on the Manager, the activity has already been stopped. **StormWatch does not provide Intrusion Detection, it provides Intrusion *Prevention*.**

2. **StormWatch protects against attacks that it has never seen before**, because the protective rules target malicious activity, not particular offending programs or network traffic. Any application that behaves in a way proscribed by the rules will be detected, and the activity stopped. Even applications that have never been seen before – or perhaps not even created when the rule was defined – will be controlled.

Test 3. StormWatch is easy to customize.

Now that we've seen the "out of the box" proactive protection that StormWatch can provide, let's see what is required to tune the default server and desktop policies to fit a local security environment. Since the rules are easily configured via a "point and click" interface, this will be relatively straightforward.

Tip: You will typically only do this to adapt the default policies to your environment. While every environment is a little different, there should be no reason to change many rules to get an exact fit.

What you need to do this test:

- **The StormWatch Manager used in the previous test.** We will modify some of the rules that caused alerts in the first test.
- **A computer to connect to the Manager.** Remember, you can use a browser on the Manager itself to connect to its user interface.
- **A computer running the StormWatch agent to test the customized policy.** We will repeat Step 3f from Test 1. It's best to use the same computer that you used in Test 1, but any computer running the same policy will work, too.

Step 1. Connect to the Manager's URL using the web browser, and go to the Event Log screen.

Using your browser, connect to the Manager. If you are using a browser on the Manager computer, you can use the URL http://localhost. If you are using a browser on another computer, you will need to use a URL with the Manager's domain name or IP address. Log in with the username and password you entered when installing the Manager software.

If you didn't start there on login, select "Event Log" from the "Monitor" menu. You should see a screen that looks like the one we saw in figure 2-2. You should see an alert from the time you tried to copy XCOPY.EXE to a different name in the system32 directory.

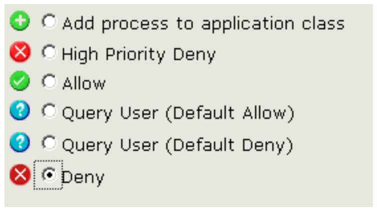
In the last test, we looked at the rule that stopped this. Now let's modify the rule.

Step 2. Jump to the rule.

This step is just like step 3 in the previous test. To view the rule, click on the link to the rule in the alert. We saw this rule earlier in figure 2-3.

Let's assume that our environment is a little different. We do not want users to be able to install drivers or update the Operating System – perhaps our IT organization take care of all driver and hotfix installation. Because the default policy allows the user to decide whether to do this (via the query pop-up messages), the default doesn't quite fit the way we work.

Let's change the rule to prohibit the action, rather than querying the user. Click the radio button next to "Deny", as shown in figure 3-1. We need to save the rule, and all you need to do now is click "Save" on the lower left hand side of the window. Congratulations – you've adapted a rule to your environment.



Add process to application class
High Priority Deny
Allow
Query User (Default Allow)
Query User (Default Deny)
[✓ Deny]

Figure 3-1. Customizing a rule.

Tip: Notice that we could have also chosen "High Priority Deny", which would have accomplished the same thing.

Step 3. Deploy the rule.

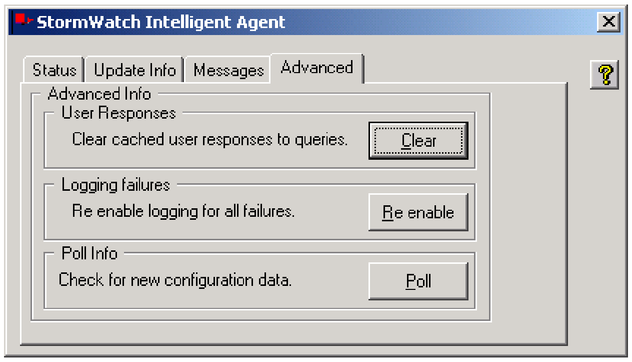
While we've successfully changed the rule, we need to get it distributed to the agents. Updating large numbers of agents can be a daunting challenge with many security products. Fortunately, StormWatch makes this easy.

First, we need to regenerate the rule. This causes the Manager to build a policy update package for every agent that uses the particular rule. Click "Generate", which you find in the middle of the bottom of the current screen. What you'll see is a screen that describes the change you are about to make. This is to give you a chance to change your mind before deploying changes to your running policies, and to verify that all the changes are what you want (you could have several rule changes generated all at the same time). Let's assume that we do in fact want to make this change. Go ahead and click "Generate" again.

You will see a few messages about what StormWatch does when it generates the rule. There really isn't anything you need to do here, and this usually only takes a few seconds.

That's it! The Manager has built updates for all agents that need them. As soon as the agent polls in, it will automatically get the modified policy and start enforcing it. Your agent will poll in after the default interval of 10 minutes. You can hurry this along by clicking on the red flag icon in the task bar of the system that is running the agent.

This opens up the local StormWatch agent GUI. Selecting the "Advanced" tab shows you the window in figure 3-2.


StormWatch Intelligent Agent

Status | Update Info | Messages | Advanced

Advanced Info

User Responses
Clear cached user responses to queries. [Clear]

Logging failures
Re enable logging for all failures. [Re enable]

Poll Info
Check for new configuration data. [Poll]

Figure 3-2. Using the local StormWatch agent GUI to poll.

Clicking "Poll" causes the agent to poll immediately. Now you should have the updated policy running on your agent.

Key Point: Updating large numbers of agents has traditionally been the hardest part of managing security. While security policies tend to change slowly, each change typically generally required a considerable amount of "Leg Work" to deploy. StormWatch is designed to completely eliminate policy update work – once rules are generated, the policy deploys itself.

Step 4. Repeat the attack from Test 1, Step 3f.

This step was where we tried to copy an executable into the system32 directory. That time, we say a pop-up message asking whether this was OK. This time, all we get is an "Access is denied" message in our command shell. The Manager has a new alert, but there was no end-user query.

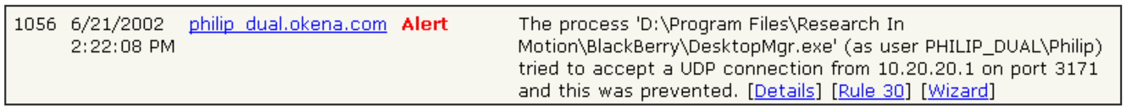
Step 5. The optional StormFront product provides a "Tuning Wizard".

StormWatch is one of several products in the StormSystem integrated system of products. StormFront is another StormSystem product that automates the process of analyzing behavior and creating or adapting StormWatch policies.

Tip: StormFront will also create entirely new policies that control applications, based on how the application behaves in your operational environment.

Careful! StormFront installs as a "snap in" to the StormWatch Manager. If you don't install StormFront, you won't be able to test this step. You can find StormFront on the StormSystem CD.

Once you have installed it (and entered a license key, if your original key didn't include StormFront), you will find new menu items that let you analyze applications of interest. Of more immediate interest, you will find that many of the alerts in the Event Log now have a new "click-through" option: "[Wizard]". The wizard helps you customize policies to allow events that are normal or expected in your environment. It walks you through the process of deciding if this is normal, whether it is normal for one system or more systems, and how you would like the policy to be updated. It analyzes your current policies and makes recommendations as to the most sensible manner of making the update. At the end of the process, it makes the appropriate changes for you. Figure 3-3 shows an alert with a link to the Wizard.

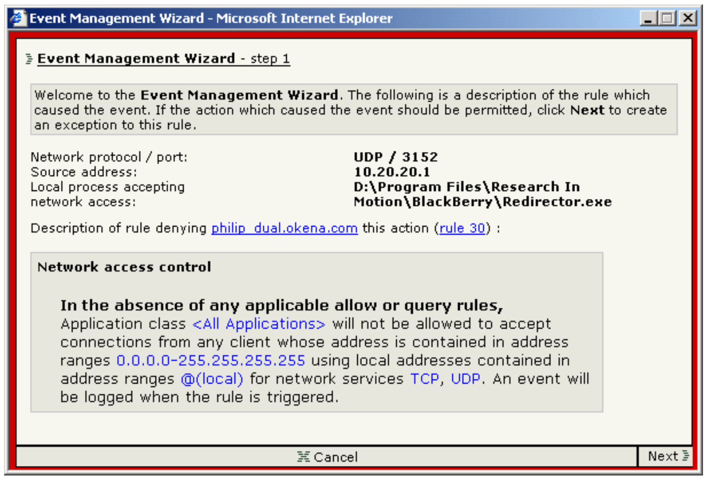


```
1056  6/21/2002    philip_dual.okena.com  Alert   The process 'D:\Program Files\Research In
      2:22:08 PM                                  Motion\BlackBerry\DesktopMgr.exe' (as user PHILIP_DUAL\Philip)
                                                  tried to accept a UDP connection from 10.20.20.1 on port 3171
                                                  and this was prevented. [Details] [Rule 30] [Wizard]
```

Figure 3-3. An alert with a link to the StrormFront policy Wizard.

Let's take a closer look at this alert. It seems that Philip has a new Blackberry wireless email device. The Blackberry software is attempting to communicate with the email server. This is blocked by the Default Desktop policy, which does not allow desktop applications to act as network servers. Users haven't complained that the application isn't working correctly, but it is hard to know precisely what the impact to the application might be.

After taking a closer look at the Blackberry application, the security administrator decides that this activity is low risk. He also anticipates that this is something that is likely to occur frequently, and that allowing this activity will result in more efficient operation. If there is a vulnerability in the Blackberry application – for example, perhaps a buffer overflow – the Default Desktop policy will still offer protection from attack. Therefore, let's use the Wizard to change the policy. Clicking on the "Wizard" link brings up the first screen of the Wizard, shown in Figure 3-4.



Event Management Wizard - step 1

Welcome to the **Event Management Wizard**. The following is a description of the rule which caused the event. If the action which caused the event should be permitted, click **Next** to create an exception to this rule.

Network protocol / port:	**UDP / 3152**
Source address:	**10.20.20.1**
Local process accepting network access:	**D:\Program Files\Research In Motion\BlackBerry\Redirector.exe**

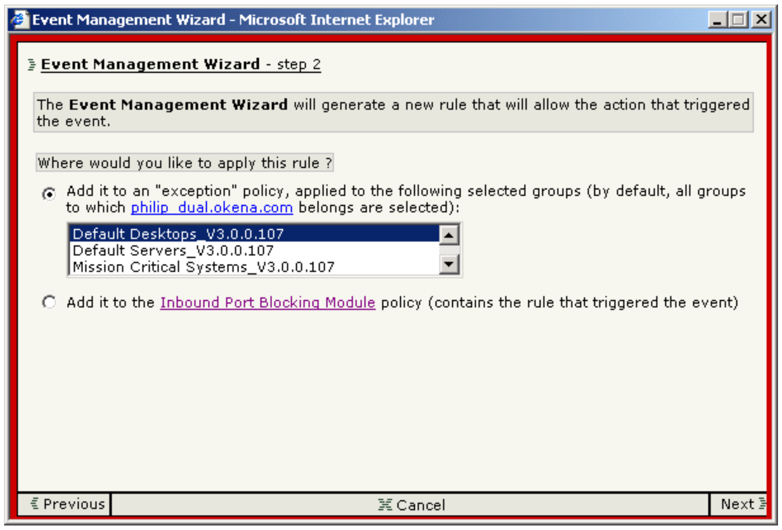
Description of rule denying philip_dual.okena.com this action (rule 30) :

Network access control

In the absence of any applicable allow or query rules, Application class <All Applications> will not be allowed to accept connections from any client whose address is contained in address ranges 0.0.0.0-255.255.255.255 using local addresses contained in address ranges @(local) for network services TCP, UDP. An event will be logged when the rule is triggered.

Figure 3-4. The StormFront Policy Wizard.

This screen shows the action that was blocked, and gives a description of the rule that blocked it. If you want, you can click through to the rule itself like we did in Test 2, but for now, let's just go to the next screen, shown in Figure 3-5.


Event Management Wizard - Microsoft Internet Explorer

Event Management Wizard - step 2

The **Event Management Wizard** will generate a new rule that will allow the action that triggered the event.

Where would you like to apply this rule ?

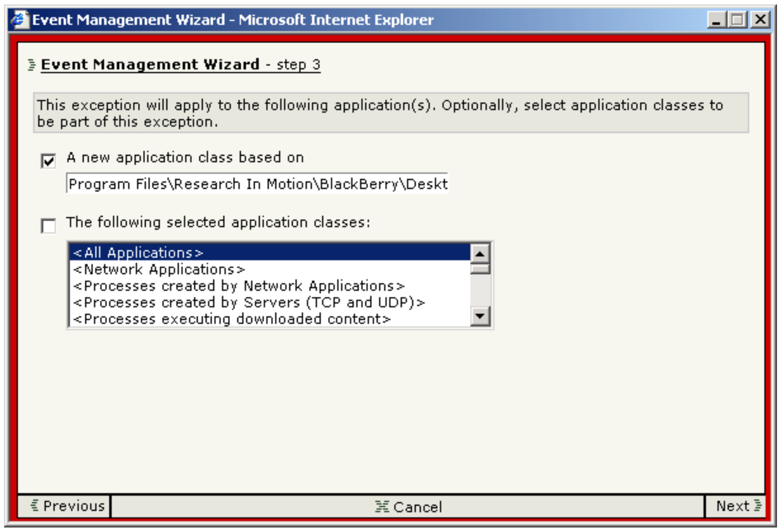
(●) Add it to an "exception" policy, applied to the following selected groups (by default, all groups to which philip_dual.okena.com belongs are selected):

Default Desktops_V3.0.0.107
Default Servers_V3.0.0.107
Mission Critical Systems_V3.0.0.107

( ) Add it to the Inbound Port Blocking Module policy (contains the rule that triggered the event)

Figure 3-5. Deciding how to update the policy.

The Wizard now shows your options on how to update the policy. One way is to actually change the policy module – the building block used by one or more policies – that contains this rule. Notice, however, that in this case the Wizard recommends creating an "Exception" policy that will be included in the groups of systems that are affected. The group affected here (in other words, the group that contains the agent that generated the alert) is the Default Desktops group. Click "Next" to continue.


Event Management Wizard - Microsoft Internet Explorer

Event Management Wizard - step 3

This exception will apply to the following application(s). Optionally, select application classes to be part of this exception.

[✓] A new application class based on

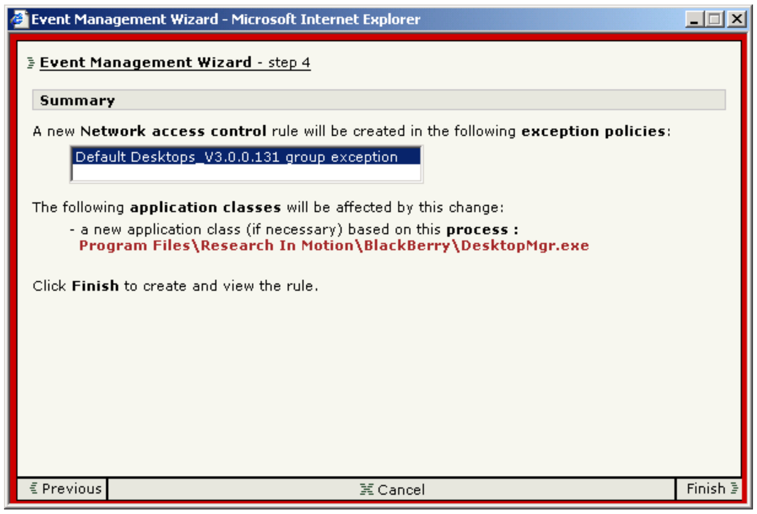
Program Files\Research In Motion\BlackBerry\Deskt

[ ] The following selected application classes:

<All Applications>
<Network Applications>
<Processes created by Network Applications>
<Processes created by Servers (TCP and UDP)>
<Processes executing downloaded content>

Figure 3-6. Selecting the appropriate applications.

The Wizard now asks whether this exception applies only to the application in question – the application that caused the rule to block the action – or whether it applies to other types of applications. Note that StormWatch has a large number of applications that it knows about (applications that use the network, etc), and these are displayed in the Wizard. However, the Wizard recommends that the exception be applied more strictly – only to the Blackberry application itself. Click "Next".


Event Management Wizard - Microsoft Internet Explorer

Event Management Wizard - step 4

Summary

A new **Network access control** rule will be created in the following **exception policies**:

Default Desktops_V3.0.0.131 group exception

The following **application classes** will be affected by this change:

- a new application class (if necessary) based on this **process** :
Program Files\Research In Motion\BlackBerry\DesktopMgr.exe

Click **Finish** to create and view the rule.

Figure 3-7. Summary of policy changes.

The Wizard now shows a summary of what you've decided to do. If you click "Finish," StormFront builds or updates the appropriate policies. You'll have to click the "Generate Rules" button on the Manager to update the live policies, and agents will get the updated policies the next time they poll in. If you click "Cancel" in the Wizard, it discards your changes before committing them.

Tip: StormFront's analysis capability is very useful for forensics investigation. If your Network IDS reports traffic to a particular high-numbered port on a host, you can use StormFront to analyze what application is listening on that port and what it does. StormFront collects and summarizes all resource access requests that the application makes. It can even build a policy to "sandbox" the application, if you like. This ability to closely investigate mysterious applications can greatly speed up the security administrator's ability to identify undesired activity.

Key Point: While tuning policies to your environment is not a complex task, automation allows lower-level administrators to be effective in updating policies. Because StormFront guides the process, policies are updated sensibly. Using StormFront ensures that policies remain well-structured and maintainable over time.

Test #3 Summary

We saw several key points in this test:

1. **StormWatch is simple to customize.** The Manager GUI provides hotlinks to the pertinent rules, making it easy to identify which rules need modification to fit your local policy.

2. **StormFront provides a Wizard to make customization even easier.** The Wizard helps automate more complex customization tasks, so that even junior level security administrators can quickly and sensibly adapt StormWatch to their environment.

Test 4. StormWatch runs on Unix as well as Windows.

So far, we've been doing all of our testing on Windows systems. Now let's see how StormWatch performs in a heterogeneous Windows and Unix environment.

What you need to do this test:

- **The StormWatch Manager used in the previous tests.** We will do some testing on a Solaris system.
- **A computer Sunning Solaris.** StormWatch supports Solaris 8 systems.
- **A computer to connect to the Manager.** Remember, you can use a browser on the Manager itself to connect to its user interface. You can also use a Netscape 6.2 browser on the Solaris computer to connect to the Manager.

Step 1. Install Agent Kits on Unix computer.

You'll need to log into the Solaris computer(s) that you want to protect. The easiest way to install agent kits is to use a Netscape web browser to directly retrieve the kit from the Manager, just like we did on Windows computers in Test 1.

Tip: There is a GNU utility called *wget* that will download an agent kit from a Manager, when run from a command line. Systems that do not have web browsers, or where installation is desired to be scripted will find this utility handy. You can find a compiled wget Solaris 8 binary at http://www.sunfreeware.com/sol8right.html.

Tip: You can also save the Solaris agent kit on a server, and use FTP to distribute it to your target computer.

Careful! Unlike Windows systems, StormWatch does not offer a "Default Desktops" agent kit for Solaris. It is assumed that all Solaris systems are servers, so only a "Default Servers" agent kit is provided by default. Note that the "IDS Mode Server" agent kit is also available for Solaris systems. As with the equivalent kit for Windows servers, this will alert, but not block activity.

In your browser, type the URL of the Manager computer, just like you did in Test 1. The Solaris agent kit is not an InstallShield executable – rather, it is a Solaris package. You install the agent kit just like you install any other Solaris package.

Careful! Just like you need to be logged in with administrator privilege to install the agent kit on Windows computers, you have to be logged in as root to install the agent kit on Solaris computers.

We'll assume that you've used a Netscape browser to download the agent kit to your Solaris system. Once it's downloaded, you need to extract the files from the *tar* archive:

```
# tar xf StormWatch-Unix_Kit-setup.tar
```

The installation will be familiar to any Solaris system administrator: you use *pkgadd* to install the agent:

```
# pkgadd -a OKENAswa/reloc/cfg/admin -d .
```

Careful! While pkgadd installs and configures the agent, you will need to reboot for the agent to begin protecting the system. The reason for the reboot is that the while the agent doesn't replace any portions of the Operating System, it "shims in" to the OS in a number of places.

Step 2. Attack your system.

Just like we did in Test 1, we need to attack the Solaris system to see how StormWatch provides protection. You can use many of the tools discussed earlier.

Step 2a. Scanning with nmap or Nessus.

Just as port scans are used against Windows targets, they are used against Unix targets.

Expected outcome of Nessus or nmap scans against StormWatch-protected systems: Because the Solaris Operating System works very differently from Windows, the results of scanning your Solaris system are different then when you scanned your Windows system. Because the Solaris *inetd* process services all incoming connections (before handing them off to the appropriate service), nmap and Nessus will report that the Solaris system has a number of open ports. This will be true even if StormWatch is blocking incoming connections from the scanning host. While it may seem surprising that blocked ports are reported as open, this is an artifact of the way Unix handles network connections – in fact, this is exactly what is seen when using other security tools like TCP Wrappers or xinetd.

Tip: The StormWatch agent will report all port scans to the StormWatch Manager. The Manager will also correlate these reported portscan events with similar events received from other agents to detect distributed port scans (port scans where many devices are scanned at the same time.

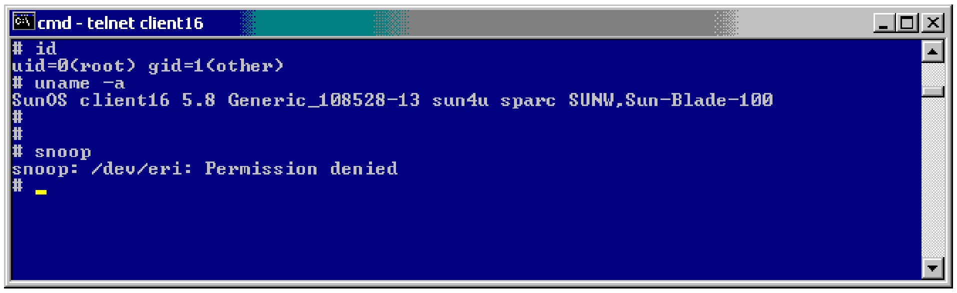
Tip: Unlike TCP Wrappers, StormWatch provides centralized control of which systems will be blocked. If you like, you can even define "Hosts.Allow" and "Hosts.Deny" variable names to hold allowed or blocked systems, and enforce this globally. Unlike TCP Wrappers, you can also centrally control which applications are allowed to accept connections. For example, you could globally prohibit the use of telnet to prevent the sending of unprotected passwords over the network.

Tip: If you use the Solaris system as the scanning computer, StormWatch prevents the execution of both nmap and Nessus. Since both nmap and Nesus send many non-standard packets, they need to communicate directly with the network interface. StormWatch prohibits this, so they will not run on a system protected by StormWatch. If you try, you will get a message similar to the following generated by nmap (note how nmap is confused, and thinks that you are scanning localhost):

```
Starting nmap V. 2.54BETA28 ( www.insecure.org/nmap/ )
pcap_open_live: /dev/eri: Permission denied
There are several possible reasons for this, depending
on your operating system:
LINUX: If you are getting Socket type not supported,
try modprobe af_packet or recompile your kernel with
SOCK_PACKET enabled.
*BSD:  If you are getting device not configured, you
need to recompile your kernel with Berkeley Packet
Filter support.  If you are getting No such file or
directory, try creating the device (eg cd /dev;
MAKEDEV <device>; or use mknod).
SOLARIS:  If you are trying to scan localhost and
getting '/dev/lo0: No such file or directory',
complain to Sun.  I don't think Solaris can support
advanced localhost scans.  You can probably use "-P0 -
sT localhost" though.
```

Step 2b. Capturing packets with snoop.

One of the first things that an attacker tried once he gains access to a Unix system is to start a packet sniffer. While there are many sniffer programs available, they all cause the NIC to change into "promiscuous" mode (i.e. receive and decode all packets seen on the network, not just those addressed to the system. For the purpose of this test, we've used *snoop*, a packet sniffer which comes bundled with Solaris.


cmd - telnet client16

```
# id
uid=0(root) gid=1(other)
# uname -a
SunOS client16 5.8 Generic_108528-13 sun4u sparc SUNW,Sun-Blade-100
#
#
# snoop
snoop: /dev/eri: Permission denied
# _
```

Figure 4-1. Capturing packets with a packet sniffer.

Expected outcome of packet sniffing against StormWatch-protected systems: The StormWatch Default Server policy prevents the NIC from changing into promiscuous mode. Different programs will handle the "access denied" OS signal differently, but Figure 4-1 shows the result when snoop is run. Note that the user trying to run snoop is root. Root normally has unrestricted access to all system resources; this example shows that even the root account can be prevented from performing dangerous tasks.

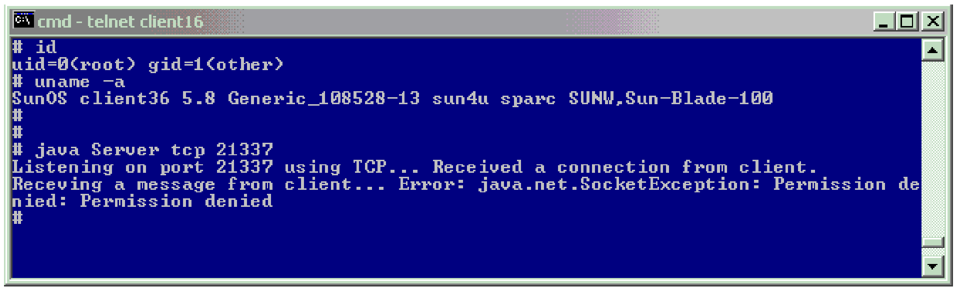
Step 2c. Installing a backdoor on a high-numbered port.

One of the most popular utilities for running backdoor services is *netcat.* As we saw earlier, you can use netcat to run a backdoor telnet-like service on any port you like with the following command (this example causes netcat to listen on port 23000):

```
nc -l -p 23000 -t -e /bin/sh
```

Unfortunately for our testing purposes (but good for security), we've seen that netcat won't even run on a Solaris system protected by StormWatch, because it tries to hook directly into the NIC streams interface. Therefore, we have to use a different program to test StormWatch. We chose to write a small Java program to do this, because Java is widely available (especially on Solaris systems). Also, many people know how to write small Java programs, so the number of potential attackers who can do this is large.

Expected outcome of running backdoor programs against StormWatch-protected systems: While the program may start up and believe that it is listening for incoming connections, (Figure 4-2 shows us running a backdoor on port 21337), the StormWatch Default Server policy will prevent the remote client from connecting to the backdoor program.


cmd - telnet client16

```
# id
uid=0(root) gid=1(other)
# uname -a
SunOS client36 5.8 Generic_108528-13 sun4u sparc SUNW,Sun-Blade-100
#
#
# java Server tcp 21337
Listening on port 21337 using TCP... Received a connection from client.
Receving a message from client... Error: java.net.SocketException: Permission de
nied: Permission denied
#
```

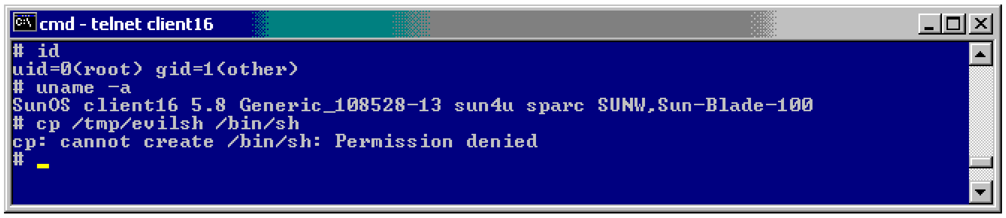
Figure 4-2. Running a backdoor program.

Tip: The Java source code for this backdoor is listed in Appendix B.

Step 2d. Overwriting the Operating System (rootkit attacks).

As we said earlier, replace critical Operating System (OS) binaries is one of the classic computer attacks. Any command shell (/bin/sh, /bin/csh) will let you try to replace critical system binaries.

Expected outcome of running backdoor programs against StormWatch-protected systems: As shown in Figure 4-3, we attempted to overwrite the command shell program /bin/sh with one of our own, /tmp/evilsh. As you can see from the example, the StormWatch Default Server policy prevented this even though we were logged in as root.



```
# id
uid=0(root) gid=1(other)
# uname -a
SunOS client16 5.8 Generic_108528-13 sun4u sparc SUNW,Sun-Blade-100
# cp /tmp/evilsh /bin/sh
cp: cannot create /bin/sh: Permission denied
# _
```

Figure 4-3. Trying to Trojan the Operating System binaries.

Key Point: StormWatch offers the ability to secure Unix and Windows systems, whether they are desktops (Windows) or servers (Windows or Unix). This simplifies training, reduces the number of console systems that are required to provide security, and eliminates the need for dataq consolidation between products.

Test #4 Summary

We saw several key points in this test:

1. **StormWatch protects Unix systems.** Just like we did with Windows, we deployed agents that protected a Solaris system.

2. **StormWatch manages Unix and Windows agents from a single console.** The StormWatch Manager lets you control both Windows and Unix systems from a single management point.

Summary

Congratulations! There's a lot to see in any new product, and we've seen a lot during our evaluation of StormWatch. While any introduction can only cover the highlights, we've seen several key points:

1. **StormWatch is installed easily**, with agents providing protection for servers and desktops "out of the box" in 30 minutes or less.

2. **StormWatch protects against a large number of attacks with the default policies.** There is no need to customize security policies to get protection, or even to log into the Manager

3. **StormWatch does not generate a large number of alerts.** Unless you attack it, it will not give you many alerts that you have to manage.

4. **StormWatch stops attacks.** All StormWatch alerts are of the form "I saw this activity which violates the policy you specified, so I stopped it". By the time the security administrator sees the alert on the Manager, the activity has already been stopped. **StormWatch does not provide Intrusion Detection, it provides Intrusion Prevention**.

5. **StormWatch protects against attacks that it has never seen before**, because the protective rules target malicious activity, not particular offending programs or network traffic. Any application that behaves in a way proscribed by the rules will be detected, and the activity stopped. Even applications that have never been seen before – or perhaps not even created when the rule was defined – will be controlled.

6. **StormWatch is simple to customize.** The Manager GUI provides hotlinks to the pertinent rules, making it easy to identify which rules need modification to fit your local policy.

7. **StormFront provides a Wizard to make customization even easier.** The Wizard helps automate more complex customization tasks, so that even junior level security administrators can quickly adapt StormWatch to their environment.

8. **StormWatch protects Unix systems.** Just like we did with Windows, we deployed agents that protected a Solaris system.

9. **StormWatch manages Unix and Windows agents from a single console.** The StormWatch Manager lets you control both Windows and Unix systems from a single management point.

Appendix A. Frequently Asked Questions

How does StormWatch integrate with the kernel of my Windows system?
StormWatch uses the same interfaces that are currently used by anti-virus and personal firewall software manufacturers. These interfaces include file system filter drivers and the Network Driver Interface Specification (NDIS).

What is the performance impact incurred by running StormWatch?
Because StormWatch does not scan packets for content (unlike AV and IDS products) the performance impact is less than 2%.

Do I need to replace any system programs?
Because StormWatch intercepts system calls at the operating system level, there is no need to replace any system programs.

What is the impact to StormWatch when a new service pack is released?
Like all software companies, OKENA will perform a thorough QA process to maintain currency with the latest Microsoft issued patches and service packs that will interact with StormWatch software.

What is the impact of StormWatch to newly installed executables on a system?
Provided that the new software does not change the behavior of existing StormWatch protected applications, there is no expected impact.

Do I need to develop policies in order to start using StormWatch?
StormWatch ships with 'out of the box' agent kits and policies for default desktop and server systems. This makes StormWatch immediately useful for securing your environment. Most organizations start with these policies, because they are designed to stop malicious activity. It also ships with a number of default policies for popular Microsoft applications such as IIS, SQL Server, and Office, for use by organizations that want to implement policy-based control over application behavior. These can be supplemented by new, user-defined policies as a StormWatch implementation proceeds in scope.

What administrative overhead accompanies StormWatch policy development and deployment?
StormWatch application-centric policies are easy to deploy because they don't need to be assigned to the user population, only to hosts; they can be simply evaluated without impacting production environments, through the use of test mode; and they can be disseminated simply through the enterprise by using host groups and agent polling.

What Sort Of Correlation Does StormWatch Provide?
StormWatch utilizes real-time correlation at both the agent and the global level. This provides greater accuracy for decision-making at the agent level and enables security to be dynamically adapted across the enterprise in reaction to events that occur on distributed hosts. StormWatch real-time correlation enables the following functionality: Correlation of network scans (ping scans and port scans) Correlation and dynamic quarantining of files based on email worm events Correlation of OS events from host event logs Correlation of events received from AV scanners (files can be added to StormWatch dynamic quarantine list).

Do StormWatch Default Policies Help With System Hardening?
Buffer Overflow protection, port scan detection, network worm protection, and Trojan detection are pre-configured rules that you can add to your policies in the same way you add other rules. These are basic system hardening features that should be applied in most cases.

What do the StormWatch Default Policies do?
OKENA provides PDF files (in the *PolicyDocs* folder on the product CD) describing each shipped policy categorized by the pre-configured group it is attached to. These documents

explain the methodology used to create the policies shipped with StormWatch and may be useful in helping you to understand how StormWatch policies are crafted. They can also be used as a guide in helping you to develop your own policies.

How Does StormWatch Protect Against Buffer Overflows?

When malicious code attempts to overrun buffers, StormWatch can detect and prevent the accessing of system functions by code executing in data or stack space. This functionality was the key to preventing notorious buffer overrun attacks like Code Red and Nimda. Note that StormWatch provides automatic protection to all applications from both Stack and Heap overflow attacks.

How Do StormWatch Policies Protect My Applications Against SYN Floods?

SYN floods results in half open connections on the server. An abundance of half open states on a server can prevent legitimate connections from being established. The StormWatch policies help to prevent the proliferation of half open states. You should apply SYN flood protection to servers within your enterprise, keeping them up and running and able to provide resources should a SYN flood attack occur.

How Do StormWatch Policies Protect My Applications Against Port Scans?

Using port scan protection in a policy causes the intelligent agent on a system to log an event when an attempt is made to scan the system for an open port. This can warn you if someone is mapping out your system in preparation for an attack. The intelligent agent also gathers information on the source IP addresses perpetrating a scan and it reveals the source address of the latest scan attempt. If scans are detected across several machines, StormWatch correlates against these events and generates an additional event to warn of this correlation.

How Do StormWatch Policies Protect Against The Propagation Of Network Worms?

When a worm of this type is received through email and executed by unsuspecting users, it generally attempts to send copies of itself to all entries in the email address book of the user. In doing this, the worm modifies registry keys, writes its own script files, and modifies existing files. When this type of suspicious activity is detected, the intelligent agent queries the user, informing the user of this activity. When the user elects to stop the system action on the desktop where the worm was received, the network worm is prevented from propagating itself. The pre-configured StormWatch network worm protection rule also correlates a series of suspicious events across multiple machines.

How Do StormWatch Policies Protect My Applications Against Trojan Horses?

The included Trojan detection rule lets you enable several different types of Trojan detection. Trapping of keystrokes by network applications. (Detect and prevent applications that attempt to capture system keystrokes.) Accessing memory owned by other applications (Detect and prevent applications that attempt to interfere with the memory space of other applications.) Stealing local passwords (Detect and prevent applications that attempt to steal local system passwords.) Downloading and invoking executable file. (Detect and prevent applications that download executables and immediately attempt to execute them. This is a type of buffer overrun attack that takes the form of and email attachment executable file) Downloading and invoking ActiveX controls (Detect and prevent malicious applications acting like a Web browser)

Appendix B. Java code for backdoor program

OKENA created this java application for use in testing StormWatch in the OKENA test laboratory. We have included it here as an example of what an attacker might use as a “back door” on a server.

Please note the following:

- This program is provided as an example only, and is not supported.
- To run the program you need to have Java Runtime Environment (JRE) installed on your machine. You can download it from http://java.sun.com/. Note that most Solaris builds include this by default.
- To run the executable you would execute this from the command line (note that this is case sensitive; you must type "Server" and not "server"):

  # java Server [protocol] [port]   ===>CASE SENSITIVE!, so you must type "Server" and not "server"

  example:

  # java Server tcp 25000

- Whenever in doubt of the arguments, just execute it without any arguments to get the usage statement.

```
/**  Server.java
 *   Version: 1.0
 *   Author: Veronika Orlovich
 *     Date: 4/01/02
 */
import java.io.*;
import java.net.*;
import java.util.*;

/**
 * This program takes 2 command line arguments from the user
 * 1. Protocol to be used:  TCP or UDP
 * 2. Port number: 0 to 65,535
 *
 *
 * The server listens on a port and protocol specified by the user
listening
 * for a client connection.  Once client connected, it receives and
displays
 * a text message that the client sent.
 *
 */

public class Server
{
      private final static int MINPORT = 0;
      private final static int MAXPORT = 65535;

      public static void main (String[] args) throws Exception
      {
            String protocol;  // protocol
            int port;         // port number

            //extract and authenticate command line arguments
            try
            {
                  protocol=args[0];
                  port=Integer.parseInt(args[1]);


                //Check protocol name specified for invalid name
                if (protocol.equalsIgnoreCase("TCP"))
                     tcpConnect(port);
                else
                     if (protocol.equalsIgnoreCase("UDP"))
                           udpConnect(port);
                   else
                   {
                         System.out.println("Error: Invalid protocol name
used!");
                         System.out.println("Exiting...");
                           System.exit(0);
                   }

               //check port number in range
               if (port < MINPORT || port > MAXPORT)
               {
                     System.out.println("Error: Port number is out of
range!");
                     System.out.println("Exiting...");
                     System.exit(0);
               }
            }catch (Exception e)
            {
                  System.out.println("Usage: Server [protocol] [port]");
                System.out.println("Where - [protocol] TCP or UDP only");
                  System.out.println("      - [port] is the port number
from 0 to 65,535");
                  System.out.println("Example: Server TCP 80 ");
                  System.exit(0);
            }
      }//end of main

      public static void tcpConnect(int port)
      {
            String message;

            try{
                //create a TCP socket for the client to ’knock’ on
                  ServerSocket listenSocket = new ServerSocket(port);
                  System.out.print("Listening on port " + port + " using
TCP... ");
                  do //loop waiting for a connection from client
                  {
                        Socket connectionSocket=listenSocket.accept();
                        System.out.println("Received a connection from
client.");

                        //establish in/out streams
                        BufferedReader inFromClient=new
BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));
```

```
                        DataOutputStream outToClient=new
DataOutputStream(connectionSocket.getOutputStream());

                        //read the message from client
                        System.out.print("Receving a message from
client... ");
                        message=inFromClient.readLine();
                        System.out.println("Done.");

                        //print message received from client
                        System.out.print("FROM CLIENT: ");
                        System.out.println(message);

                        //close the connection
                        System.out.print("Closing connection to client...
");
                        connectionSocket.close();
                        System.out.println("Done");
                  } while (true);
            }catch(Exception e)
            {
                  System.out.println("Error: " + e);
            }


      }//end of tcpConnect
      public static void udpConnect(int port)
      {
            try{
                  //create a UDP socket to listen on
                  DatagramSocket serverSocket = new DatagramSocket(port);
                  System.out.println("Listening on port " + port + "
using UDP.");

                  byte[] receiveData = new byte[1024];
                while(true)
                  {
                        DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);

                        //receive a packet from client
                        System.out.print("Waiting for a datagram
packet...");
                        serverSocket.receive(receivePacket);
                        System.out.println("Received.");

                        //extract message received from client
                        String message = new
String(receivePacket.getData());
                        System.out.println("FROM CLIENT: " + message);

                  }//end of while
            }catch (Exception e)
            {
                  System.out.println("Error: " + e);
            }
      }
```

```
} // end of server class
```