



**КЗЗр Ош7**



## When it comes to the exploit du jour, an ounce of protection is worth a pound of detection. Host intrusion prevention products can help keep attackers at bay BY MIKE FRATTO

"EXPECT THE UNEXPECTED" may be a cliché, but it's a smart network security strategy nonetheless. Vendors and security professionals, despite claims to the contrary, cannot pinpoint with certainty the next threat to your virtual existence any more than you can. Of course, you can lower your exposure by installing security devices, such as firewalls, VPNs, virus scanners and IDSs, and implementing operational procedures, like OS hardening and enhanced authentication and access control. But the reality is that you must provide server access to employees and other users, and the risk of attack mushrooms when you open your door to the Internet at large.

Here are a few hard truths: All products have security vulnerabilities. Someone will find those vulnerabilities. Regardless of whether the discoverer tells the vendor about the vulnerability, publishes it or keeps it to him or herself, you're exposed to attack until the vendor issues a patch to close the hole and your administrators install said patch.

On the subject of patches, please repeat after us: "I must keep abreast of the patch cycle." (For a review of products to help you do that, see "PatchLink Helps Keep Windows Closed," at [www.nwc.com/1318/1318f3.html](http://www.nwc.com/1318/1318f3.html).) Still, you must test patches and hot fixes before applying them to production servers to make sure they don't break existing functionality, so even with a program in place, your servers

**36** EXECUTIVE SUMMARY **40** E-MAIL POLL RESULTS  
**48** REVIEW OF HOST INTRUSION PREVENTION PRODUCTS  
**52** HOW WE TESTED **54** VENDORS AT A GLANCE

could be days or weeks behind the patch-update curve.

Patches, while important, are reactive. The problem is precious little software is built with security in mind. General-purpose operating systems are designed primarily to manage multiple users and provide seamless access to system resources. Depending on the permissions bits or ACLs (access-control lists) placed on files and users' ownership or group affiliations, some will get more access than others. For example, the root user on Unix or the administrator in Windows NT/2000 have almost unfettered access to resources. More important, many system services run as highly privileged users. Each server requires only a limited, and often definable, set of resources to function properly, but there is no way to restrict services' access to required resources. Because these services provide direct access to your servers and, subsequently, other places in your organization, host OSs need to be protected from vulnerable programs.

### Preemptive Protection

**YOU NEED A WAY** to proactively protect your servers from malicious attacks. Host intrusion prevention, or HIP, is a way to do just that. Using a variety of different methods, HIP products restrict a program's or a user's access to system resources, safeguarding the underlying OS from attacks that take advantage of poorly written code.

For example, most attacks that give remote access do so by letting a server run commands on behalf of a remote user. It doesn't matter if the attack is a buffer overflow or a malformed URL—the salient point is that commands are run remotely. For example, on a Microsoft Internet Information Server (IIS) Web server with no service packs or hot fixes applied, there are way too many ways that a command shell can be invoked through *inetinfo.exe*, the IIS process (see "How We Tested Host Intrusion Prevention," page 52 for details). Yet, there is no reason for *inetinfo.exe* to be invoking a shell. Knowing that attacks often call a shell, and knowing that *inetinfo.exe* shouldn't be able to invoke a shell, we can now formulate a security policy. HIP products enforce that policy.

HIP is a relatively new security space, and while we didn't do a thorough security audit on the products we tested (see page 48), we did attack servers remotely, both as an attacker would do and while logged on as administrator or root. We could not successfully penetrate systems where HIP applications were installed and properly configured. Because we don't know what assaults attackers will dream up next, we won't go so far as to state that these products are attack-proof, but we are confident in saying that they do provide the protection they claim.

### A Kernel of Our Esteem

**MOST HIP WARES INSTALL** as kernel-level modules and shared library replacements. HIP works by trapping system calls from the application to the OS. By

## Executive Summary

# HOST INTRUSION PREVENTION

**T**he West Nile virus has been much in the news lately. Say an expensive vaccine was developed that might protect you from the bug and its mutations—the key word being *might*. Would you pay up? Or are you willing to bet that insect repellent and long sleeves will be enough to protect you on the off chance an infected mosquito decides you smell like a tasty treat? After all, even if you get West Nile you may pull through with no lasting damage ... or you may not.

If you've tried to sell your CFO on security technologies beyond basic firewalls and virus scanners, you see where we're going with this. We know from your feedback that despite new ROSI (return on security investment; see "Make Your Case," at [www.nwc.com/1318/1318f16.html](http://www.nwc.com/1318/1318f16.html)) numbers, the sell can be rough. Luckily, the HIP (host intrusion prevention) products we tested won't make too big a dent in your budget, even factoring in the sometimes considerable learning curve and time spent on customization. The premise behind HIP is that proactive is good, given the sorry state of security with most mainstream OSs. HIP products scrutinize system calls following your policy and restrict a program's or a user's access to system resources, shielding you from attacks that exploit shoddy coding.

We loaded up Argus' PitBull LX and Protector, CA's eTrust Access Control, Entercept's Web Server Edition, Harris' STAT Neutralizer, Okena's StormWatch and StormFront, and WatchGuard's ServerLock and AppLock/Web, and after months of poking and prodding, we picked Okena StormWatch as our Editor's Choice. However, rivals are on track to steal Okena's thunder. We'll be watching, so stay tuned.

## SAMPLE COST OF DEPLOYING A HIP PRODUCT (to protect a Web app server)

Cost of server	\$1,800
Cost of training	\$35 per hour x 40 hours = \$1,400
Cost of developing and deploying policies	\$35 per hour x 20 hours = \$700
Total cost	\$3,900

trapping the system calls before they get to the kernel, the HIP application can process the call against its policy and either deny it or allow the call to continue. None of the products we tested required us to install a secure OS. Argus Pitbull doesn't trap system calls; rather, it modifies the system-call code to incorporate its policy enforcement. In any case, a system call is first processed by the HIP product and compared against its policy. If the call is allowed, it's passed through to the kernel. Otherwise it is denied, and the application is notified. Individual rules also can be logged so administrators can track application or user activity. Policies explicitly define what resources, such as files and network ports, an application or user may access and how the application or user can access them, for example, read, write or execute. HIP products that implement user-based access control can even take power away from the root and administrator.

### Three-Way No Calling

**HIP VENDORS HAVE IMPLEMENTED** three basic models: behavior-based policies, signature-based policies and user-based policies. While you can achieve similar functionality with any model, the general concepts are vastly different. In all three

cases, it's important to understand that these products protect the OS from attacks that exploit applications. They won't stop attacks that operate by interacting with an application, such as someone trying to manipulate your database through SQL injection

## HIP products implementing user-based access control can take power away from the root and administrator.

by fiddling with your Web application, nor will they stop servers from accessing files to which they should have access.

Behavior-based models require you to define the normal behavior during an application's run time. For example, Apache on Linux needs to read and write files in its directory root, read its configuration file, read and execute its own executable, read shared libraries, read files from webroot, execute cgi-bin files and bind to Ports 80 and 443. A properly behaving server doesn't require any access beyond that, so a policy that limits its access blocks all abnormal behavior. Of course, what is normal for one server may not be

# COST OF DOWNTIME

(break-in using unknown buffer overflow attack against Web server)

## Total productivity lost

Total downtime

Time to assess and repair damage: **49** hours

Total productivity lost: **49** hours x **30%** time users lost x **500** users = **7,350** hours

## Cost of downtime (Total productivity lost x percentage of staff) x hourly rate

Employees with annual salary of \$45,000

**(7,350** hours x **15%** of staff) x **\$22.50** per hour = \$ **24,806.25**

Employees with annual salary of \$30,000

**(7,350** hours x **30%** of staff) x **\$15.00** per hour = \$ **33,075.00**

Employees with annual salary of \$20,000

**(7,350** hours x **55%** of staff) x **\$10.00** per hour = \$ **40,425.00**

## Total cost for downtime

**\$98,306.25**

*[It's not hard to believe the claims of high dollar loss because of attacks when you factor in downtime and repair.]*

normal for another, so each server must be profiled and its normal behavior thoroughly understood before you can define an enforceable policy.

## Sign Here, Please

**SIGNATURE-BASED HIP** products use signatures to detect and block malicious activity. Of course, signature products suffer the same problems as IDSs and virus scanners—they are only as current as their latest updates. But used in conjunction with behavior-based HIP, signatures provide another layer of security by being able to selectively block attacks at the application layer, and they also aid in attack identification because an attack alert message can provide a name, rather than alerting on a generic “buffer overflow.”

Finally, user-based models are similar to user-based access control, where user name or group membership is used in conjunction with ACLs on system objects to determine if access is allowed or denied. This model works well in the Unix world, where services can be run under unique user IDs. However, in Windows NT, where critical services, such as *inetinfo.exe*, can run as

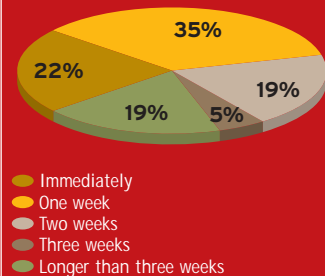
System\Local, it's a different story; you can't effectively limit an activity that is running as System\Local because other services running as that account require so much access. The lack of individual user IDs is also problematic.

As for policies, HIP products fall into two categories: those limited to predefined policies and those that allow customization. In general, HIP products with predefined policies are easier to install and manage, provided your applications are supported. These packages use discovery or rely on an administrator to customize the configuration for each installation. For example, if you put your webroot in a nonstandard location, the HIP application will need to know about it. The obvious limitation is that you can't protect anything beyond your supported applications.

The second category of products offers methods for securing any application, provided you can define the necessary resources and the types of access for each resource. Customization offers the flexibility to support any service your business needs, and you're not tied to a vendor's limited set of supported

### E-MAIL POLL

How long does it typically take your IT organization to implement server patches provided by vendors?



Source: NETWORK COMPUTING E-Mail Poll, 194 respondents

applications. The downside is that application profiling and policy development are complex and time-consuming processes because you have to cycle through resource and access discovery, policy development and deployment, and testing, and repeat until the policy is defined to allow only the necessary access and the application runs without problems.

Clearly, there is a trade-off between ease of management and flexibility. If you go with a general-purpose HIP, budget to send your administrator to training, or at least give him or her time to learn the systems. We spent between 25 and 40 hours per product just learning the ins and outs, and we consider ourselves moderately skilled in many host OSs. We also used some handy references, which you can find in our "Recommended Reading" list, below. While programming skills are not required to effectively profile applications and develop policies, the more you know about how the host OS and applications work, the better off you'll be.

## What's It Gonna Do for Me?

IT'S EASY TO QUANTIFY THE PRACTICAL VALUE of HIP products. Attacks against the OS and applications are no longer viable, so you're not at the mercy of host-level vulnerabilities. It's also easy to show the direct cost benefit of investing in this technology. The huge claims of financial loss attributed to attacks like Code Red and Nimda probably are more accurate than not when you take into account not only system downtime but affected workers' lost productivity. Many reports have systems down for days, in some cases even weeks. Also factor in the costs associated with stopping work on ongoing projects, which now will be late, in turn affecting other business processes. For organizations that had their online systems taken down, the loss spiraled even higher.

To get an accurate assessment of costs associated with a break-in and the savings that would result if that attack had no impact, you need to formalize, in terms of time and money, costs associated with critical IT systems. Unfortunately, according to Forrester Research, 60 percent of companies say they can't even quantify the loss due to security incidents, and 52 percent don't know how to quantify the cost of responding to incidents.

That means that, to make the business play, you may have to do some leg work to gather this data. Be sure to factor in the cost to determine the cause of the incident, assess the damage and repair the damage.

To do this, determine the number of workers affected and for how long, arrive at average salaries along logical groups, and then total it up to see how much an attack cost you in productivity. If your server

is used to generate revenue, such as a customer order system, estimate the cost of lost business during the incident and after.

We developed a sample worksheet to illustrate these principles. We assumed an attack on a Web server was successful, and the attacker had control of the server. The breach was noted, and the server was taken offline. A backup server was available, but because it was identical to the running server, putting it online was too risky. The time to assess the damage, including the vulnerability, took 34 hours (based on the results of the HoneyNet Forensic Challenge, [project.honeynet.org/challenge/](http://project.honeynet.org/challenge/)) and another 15 to restore and repair the server (our experience). Out of a user population of 1,000, roughly 500 users spent a significant amount of time, 30 percent of their workdays, using the server. While the server was down, no work got done with the

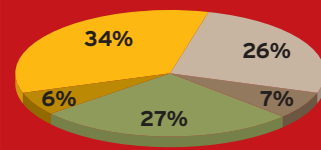
application. We estimated three classes of users, from data input to managerial functions, and assigned a population to each. After totaling the server downtime, the amount of time lost for employees and the hourly rate for each group, we came up with a staggering \$98,306 for the incident.

Of course, if your organization hasn't had to deal with a security breach, your cost of an intrusion is an exercise in speculation. The point is, there are hard costs associated with break-ins, and the savings from one blocked intrusion are just too large to be ignored.

Deploying HIP is not often simple. Depending on the vendor, it could take from a few hours to several days just to learn the product and develop the solution. The more feature-rich applications we tested took us as long as 40 hours to learn and another 20 to deploy and test an effective policy. Multiply that by an hourly rate of a security admin making on average \$70,000—that's about \$35 per hour, \$2,100 for 60 hours—add the cost of software, at a median of \$1,800, and the cost of protection is a steal at \$3,900.

### E-MAIL POLL

What is your level of awareness of host-intrusion-prevention software?



- Not at all aware
- Aware but haven't evaluated
- Currently evaluating
- Plan to evaluate
- Implementing now or soon

Source: NETWORK COMPUTING E-Mail Poll, 194 respondents

## RECOMMENDED READING

**Inside Microsoft Windows 2000, Third Edition** (Microsoft Press, 2000)

**Linux Kernel Programming, Third Edition** (Addison Wesley Professional, 2002)

**Smashing the Stack for Fun and Profit**, by Aleph One, [www.phrack.com/show.php?p=49&a=14](http://www.phrack.com/show.php?p=49&a=14)

**A Buffer Overflow Study: Attacks and Defenses**, by Pierre-Alain Fayolle & Vincent Glaume, [www.securiteam.com/securityreviews/5IPOG1P8AM.html](http://www.securiteam.com/securityreviews/5IPOG1P8AM.html)

**The Honeynet Project Forensic Challenge Results**, [project.honeynet.org/challenge/results/](http://project.honeynet.org/challenge/results/)