

# Application Security: Countering The Professionals

Robin Layland

## Attacks are moving up the stack and getting more sophisticated. Are you ready to deal with them?

Security threats and attackers are turning professional. Network managers still need to stop the script-kiddies from defacing their websites, but it is becoming increasingly important to stop the professionals who want to steal valuable information. The new attackers search for vulnerabilities in the application and exploit these weaknesses. Attackers are bypassing the traditional network-layer firewall and IDS defenses; their exploits appear as legitimate traffic to the network layer defense, but hiding in the application layer are deadly attacks.

Working to counter these threats are a collection of security vendors, some established, some new. Well-known security vendors such as Cisco and Radware have been joined by new security players such as NetContinuum, Imperva, Citrix, Breach Security, Protegrity and ConSentry Networks. Additional players can be expected to join the field in 2007. Their goal is to take security to the next level and protect both the network and applications.

The new application threats come primarily from three areas:

- Viruses, worms, rootkits and malware.
- Attackers exploiting Web application vulnerability.
- Internal users gone bad or external attackers stealing valuable data.

Viruses and worms have been around for awhile. Malicious programs—malware—have become increasingly popular, and rootkits—i.e., malware that hides in the OS's kernel—are a new threat. The problem is that all these threats hide in the application payload, bypassing traditional network security, but nevertheless, enterprise managers need to ensure that these attacks are not spread by the network.

Complicating the situation is that the problem

really has to be viewed as two different types of threats. The first is viruses, worms, malware and rootkits that are known—security companies have seen them and created signatures for them. The second type is zero-day threats—those which are new, and for which no one has yet created a signature. Protecting against zero-day threats requires a different type of solution.

Attackers can also exploit Web applications. Some of the most common attacks include:

■ **Invalidated Input:** A URL may contain multiple parameters that tell the application what to display. The developer might create a parameter where the values are 1, 2 and 3. An attacker sends in a URL with the parameter coded to something other than 1, 2 or 3 and the application tries to process the invalid parameter, which causes it to take an action not intended, possibly opening up a vulnerability in the application.

■ **SQL and Command Injection:** This attack works by manipulating the Web parameters—the part following the `www.website.name` that the website sends back with the URL, i.e., all those numbers, letters, equal signs, etc. that you often see in your browser window. The vulnerability is created when an attacker inserts an SQL command in the parameter field, and the Web application believes the parameter is legitimate and executes the command, giving the attacker access to data. The attack is not limited to SQL but can be any command that the Web application or OS would execute. SQL is the most common; the general case is called a command injection.

■ **Buffer overflow:** This has long been a method of attacking applications, and Web applications carry on the tradition. The parameters in the Web request are expected to be a set length. An attacker codes a parameter longer than what the developer expects, and the extra part includes executable instructions. When the parameter is placed in storage, the extra-long part overwrites some of the existing application's code. If it overwrites instructions, the application will then execute the attacker's codes, allowing the attacker to direct the application to do what he or she wants.

Robin Layland is president of Layland Consulting, a firm that specializes in network architecture and new technology. He has more than 25 years' experience in enterprise networking as both a consultant and working at technical and management positions. He can be reached at [robin@layland.com](mailto:robin@layland.com)

■ **Cross-Site Scripting (XSS):** The attacker sends malicious script in the URL, forms or headers. This bad script can be sent from the legitimate server—for example it was placed by an attacker in a comment or message field sent to the server—or it can be accessed when the user clicks on a link in an email that has the script embedded in the URL. In either case, the user’s browser executes the script, which can give the attacker access to their user information or cookies, which allows the attacker to go to the sites later on as if he or she were a legitimate user.

■ **Broken Access Control:** Suppose a website requires users to authenticate themselves to gain access to the main website or some specific portion of the site. An attacker may be able to bypass this access control by learning the Web address of a page past the sign-on page, and entering the address. If the Web application is not written correctly, it may not realize the attacker has not been authorized, and it allows the attacker to go directly to a page in the secured area.

■ **Cookies:** An attacker modifies the cookie to allow him or her to change access controls, user ID or information, to allow access they should not have. Another problem with cookies is that attackers can snoop a user’s cookies that contain critical information that is not sufficiently secured.

■ **Application Denial of Service (DoS):** This is an evolution of the DoS and distributed DoS (DDoS) attacks, where the attacker uses “legitimate” requests to flood an application, causing the application to be either overwhelmed or significantly slowed.

■ **Improper Error Handling:** Error messages from Web applications often include detailed internal error messages that include database dumps, error codes and trace information. This information can give attackers important clues about flaws in the site’s design that will allow them to craft better attacks.

■ **Configuration Management:** A wide range of sins allow attackers to gain access, including leaving accounts and access control with default passwords, unnecessary services left on, such as remote management, and improper files and directory permissions.

Attackers don’t always have to find weaknesses in the application, or plant malware. They act like normal users, only they are stealing critical business data or valuable information such as social security or credit card numbers. For example, an attacker may have an “inside” accomplice who gains access to a PC belonging to someone who is authorized to look up customer records to collect credit card numbers. The accomplice then sends the data to an attacker’s machine outside the corporation. From the network’s perspective, no violation has occurred; the traditional network security would not notice

a suspicious pattern, but would only see someone doing their job.

All the attacks listed here, along with new application attacks still being invented, are undetected by network firewalls and IDS/IPSs. The general reason is that the attacks are hidden in the application payload, and traditional network security only examines each packet as far as the headers. Attacks are spread over multiple packets and can only be detected by examining the entire transaction, or can only be found by looking at their behavior over time.

What is needed to catch and stop this new class of attackers is a security solution aimed at the application layer. The new application-layer threats place new demands on the network security architecture, changing the way network security examines the packets and requiring significant new functions.

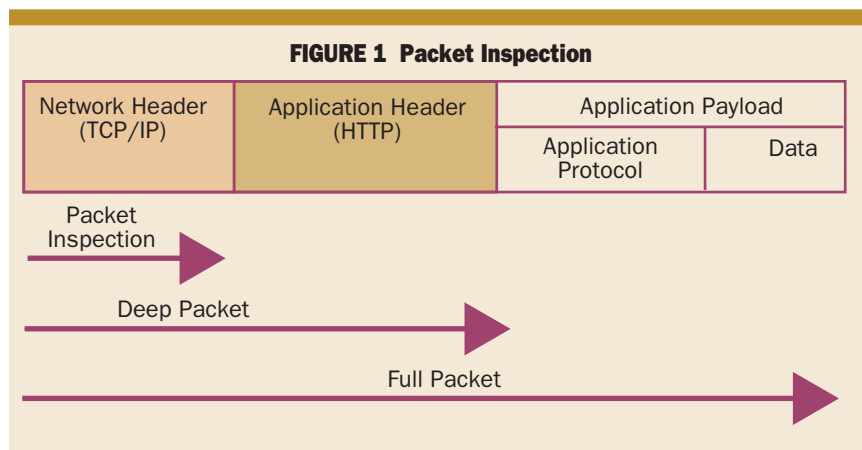
Besides catching the bad guy, there is another strong reason for implementing application security—it overcomes known problems in existing applications. When a security problem is found in a new application, deployment must be held up until the development team creates a fix, a process that can take a significant amount of time. And if a security problem is found in already-deployed applications, the problem is what to do until a fix is created—leave it up or take it down. Application security solutions allow the application to be deployed on schedule or left active, without the enterprise manager worrying about attackers exploiting the hole.

### Looking Very Deep Into The Packet

Network security always had to inspect the packets. The question is how deep the inspection had to go. Network firewalls only had to look at the TCP/IP header, along with preventing DoS attacks. Preventing some Web attacks requires extending the inspection to the application or HTTP header. This is generally referred to as deep inspection (Figure 1).

The problem with just examining the application header is that the attacks can be hidden in the application payload. A few examples of problems that can hide in the application payload:

**Many attacks are hidden in the application payload**



**More detailed scrutiny requires more detailed policies**

■ An attacker or internal user has collected credit card numbers and now needs to send them to an outside source. They send them in what appears to be a legitimate message.

■ Application DoS attacks can only be stopped by monitoring the application payload and noticing that the application requests are repetitive and from the same group of users or IP addresses.

■ Viruses, worms, malware, spyware and rootkits hide in the application payload. While all devices should have their own protection, the generally accepted defense-in-depth strategy requires that the network also hunt for and kill these attacks, requiring network equipment to examine the entire application payload.

■ Security breaches can be from authenticated users who are doing something malicious. For example, using XML, Company A develops an automatic ordering system. The company sends an agreement to Company B, and someone at Company B responds by agreeing to the transaction but changes a field that shouldn't be changed—for example the price or number of items. Company A's application should be checking every field to make sure this doesn't happen, but in this case the application developers didn't think they needed to; they assumed that no one would change the field. Increasingly, network security needs to provide a second layer of defense, ensuring this and other types of changes don't occur.

These new threats mean that network security increasingly must examine the entire packet. This can be thought of as either full packet inspection or very deep packet inspection. Unfortunately, even this will not be enough. The XML example above requires that the network security examine the entire transaction—a transaction that is likely

spread over several packets requiring full transaction inspection. This means the network security equipment must assemble the packets, holding each one until it has the complete transaction, before it can send it on to the application.

Implementing full inspection creates several problems. It will require more storage and processing power. It also has the potential to slow the transaction down—how much depends on how the solution implements enforcement. If the security device that examines the packet or transaction provides the enforcement, then the data can't be forwarded until it has run it through the inspection process.

Another approach, used by Breach Security, is to allow the message to be forwarded, while the security device keeps a copy. With this system, servers have rules that roughly determine if the message should be examined; if the rules don't indicate that the message needs to be closely examined, the message is allowed to be forwarded to the application. If the rules do indicate there may be a problem, the message is held until the server is told to release it. Breach Security's application security device meanwhile examines a copy of the message for any problems. Once it has completed its analysis, it sends a message to the application server telling it to release the message (or kill it if there is a problem). The goal is to only delay messages that may have a problem instead of all messages. How much of a difference this makes in the real world is not clearly understood at this time.

The biggest problem is that this level of scrutiny requires more detailed policies. Increasingly, it is not enough to just have a policy about what someone can access. The policy will have to

**FIGURE 2 Network Security Architecture Functional Overview**

		Data Center	Edges	Client
<b>Correlate</b>	Admission Control/Compliance			•
	Authentication			•
	Encryption		•	•
	Network Firewall		•	•
	Access/Policy Enforcement			•
	Content Filtering		•	•
	IDS/IPS		•	•
	Virus, Worms, Malware & Spyware		•	•
	Signatures		•	•
	Zero-Day		•	•
	DoS & DDoS Prevention		•	
	Network Application		•	
	Web Application Firewall		•	
Anomaly Detection & Behavioral Modeling		•	•	

 New Security

understand what the applications are doing and what is allowed inside the application—what parameters are valid, what actions are acceptable, and by whom? The problem is that few companies have developed the personnel or the expertise to create policies at this level. Many security companies are helping by automatically creating policies which can then be reviewed and implemented. However, while automation can help, it can't do the detailed policy definition that will be required as professional attackers climb up the stack and electronic relationships become more prevalent.

Enterprises will have to develop the expertise to build the detailed policies that application security requires. Security groups will transform from maintaining equipment to understanding the business and writing policy that other groups owning the equipment, such as the networking and server departments, will enforce. This is good for a security professional, because the level of understanding required makes this a higher-level job and a more important one, and thus one that justifies a higher salary.

### **New Functional Requirements**

Combating threats at the application layer requires additional functionality. Figure 2 shows the full range of security functions network managers need to build into their network. Currently the network security portfolio includes the VPN functions of authenticating the user and encrypting the data; enforcing network layer-focused corporate policies such as access control; using client-based anti-virus solutions; IDS/IPS to search for known signatures of attackers; filtering Web requests to block access to unauthorized sites; and preventing network-level DoS or DDoS.

Application security does not eliminate the need for these existing functions. Instead, the application challenge adds several new functions, including admission control and compliance; extending virus, worm, malware, spyware and rootkit detection beyond known signatures to catch zero-day or brand new threats; stopping denial of service attacks aimed at applications; Web firewalls to protect against unique threats to Web applications; and anomaly detection and behavioral modeling to capture unique but bad actions by users.

The first new function is admission control and compliance. The idea is that, before the device is allowed onto the network, the network checks the device to ensure that the user has up-to-date virus protection at the correct patch level, is in compliance with all the network standards, and is who they say they are. The network does this either by checking with an agent already on the device, or by sending down a short-lived agent to perform the check. If everything is OK, then the device connects to the network. If something is wrong or not up to standards, the network manager must decide what to do about it. This can range from

denying complete access, allowing the user only limited access to applications—such as only publicly available ones—or correcting the problem.

The benefit of network access control is that it stops machines with vulnerabilities that can be exploited by attackers from getting access to the network. The potentially negative aspect is that it can stop legitimate users from doing their jobs. To avoid harming productivity in this manner, enterprises should implement efficient means of updating end users' virus protection and patching, and make sure this supports a wide range of end devices, from PCs to newer handheld devices. A good implementation will also periodically monitor the end user device if it stays connected for a long period of time, to prevent an already-connected device from going bad.


Vendors give admission control different names. Cisco uses the term Network Admission Control, while Microsoft uses Network Access Control. Implementations of the concept also vary. Cisco implements it within their switch, Juniper as part of the VPN structure, and Microsoft as part of the new OS Vista. New players like ConSentry Networks provide a standalone box that can also act as a concentrator switch.

The Figure 2 columns labeled Datacenter, Edge and Client address where the functionality is needed. For example, admission control needs to be present in the edge devices; edge can mean the LAN concentrator in the campus network; at the branch office, the VPN device, wireless or Internet gateways. Admission control also is needed in the client. This, as noted above, can be a permanent piece of software in the client, or software sent down when the client connects to the network. Datacenter refers to the equipment fronting the servers and the functions, wherever servers are located and need to be protected.

*Zero-day detection and anomaly detection and behavioral modeling* are related. Both are looking for unusual actions. For example, it is not uncommon for a client, upon startup, to initiate a large number of connections to different applications simultaneously. What is unusual is for a client to start a very large number of connections, with a fair number of them failing. This is the behavior of a worm trying to connect out and using a directory, such as Outlook's, to spread.

Additionally, a client is expected to act like a client. For example, a customer service agent would normally retrieve customers' information. What is unusual is for the customer service agent to send out a file that contains a large amount of customer information. The same logic can be applied to servers—they generally should not be asking for customer information from another server. Zero-day, anomaly detection and behavior modeling are techniques for looking at unusual behavior when no known signature exist.

How well developed are these techniques? Vendors are starting to be able to catch actions



**If clients aren't acting like clients, or servers like servers, this may indicate a problem**



**New packages  
must provide  
adequate  
reporting without  
generating  
excessive false  
positives**

related to viruses, worms, spyware and malware. They are also starting to catch when credit card numbers or social security numbers are being sent. How well they catch other behavior varies widely from vendor to vendor, but the good news is that the vendors are working hard to improve their capabilities, and advances can be expected over the next several years.

*Web application firewalls* is one area where vendors have made significant advances. There are many ways to attack Web applications; the bulleted list near the opening of this article presented a partial list. Web application firewalls plug these holes, many of which are created by sloppy application design—for example, not checking if the user has returned an invalid or too-long parameter opens a hole. Other holes are harder to detect, such as when someone has tampered with a cookie.

The different techniques used by application firewalls can be classified in four broad groups:

- White list
- Monitoring actions
- Tampering
- Signatures

A *white list* specifies valid parameters, including valid values and their lengths. The application firewall knows what parameters are valid for each page, and if it detects any parameter outside the range, it takes action. This is effective in stopping many attacks, including invalid parameters, cross-site scripting, SQL and command injection and buffer overflow.

The biggest issue surrounding the white list is how to create it. One technique is for the application developers to give network management a list of all valid parameters. The problem with this approach is that it requires the network manager to know about every application, and to get the developers to supply all the information, something that rarely happens in the real world. Additionally, any time the application makes changes, there must be a notification process before the application can go live.

Vendors have taken several approaches to get around this problem. Citrix's application firewall monitors the Web application and creates a list of all the parameters it has seen. Network managers then meet with developers and ask them if the list is complete. This is an improvement over the first approach, but is still difficult to do in a large organization.

Other vendors such as NetContinuum, Breach Security and Imperva also monitor Web applications and automatically develop a list. This approach does not require meeting with the developers, but is open to problems if a parameter is rarely used, requiring a long learning time. It also opens the system up to learning a bad parameter if the attack is launched during the learning period. Imperva mitigates the problem by allowing network managers to tell the security device to only

learn from trusted IP addresses. Ultimately, there is no perfect solution, and good security procedures will likely include a combination of auto-learning and reviewing the results.

*Monitoring actions* means learning what is allowed and making sure users stick to the rules. For example, if a section of the website requires a user to be authenticated or enter a password, then no user should be able to bypass this step. For example, if a user enters a page address beyond the credentials checkpoint without being authenticated, then the application firewall stops their access. Additionally, the application firewall can monitor what is being sent out and if, for example, an error page has information that would allow an attacker to improve their attack, the application firewall can remove that information.

*Tampering* looks for changes that are not allowed. For example, suppose a cookie is placed on a user's machine. An attacker could change the cookie and when the Web application retrieves the cookie it could open a hole for the attacker to enter the Web application. Web application firewalls can monitor cookies, pages and parameters for inappropriate changes and correct the problem. NetContinuum encrypts cookies to keep them safe.

Web application firewalls also learn the signatures of common attacks and take preventative action.

Web application firewalls are a powerful tool to protect Web applications, and when combined with anomaly detection, behavioral modeling and zero-day prevention, can stop most current Web attacks. What makes all these approaches, including the existing ones, even more effective is when information from all methods is combined and collated. Cisco and other vendors currently do this by collecting information from the different platforms in a management station.

An interesting addition to this approach is taken by Imperva. Imperva combines many of the functions shown in Figure 2. The problem is that many times each individual function does not signal that it has definitely found a problem, but instead reports that there may be a problem—not a strong enough warning to take action, but strong enough to raise concern. The problem has always been that if network management takes action based on the maybes, the false positive rate gets out of hand.

Imperva's solution is to have each function report the degree to which it is sure. If several functions all report they are concerned and that there may be a problem, then Imperva says there is enough reason to expect something is wrong and takes action. As attackers launch new attacks, this approach could help turn gray situations into black and white.

Correlating actions of a user over time is another way application security devices can provide protection. For example, a user who is probing the network and application may not do enough on

any one probe to be noticed, but the cumulative effect of all their probes could be noticed and the user blocked.

### **Bump On The Highway**

Application security vendors are making good progress countering the threats, but there are speed bumps in their way. The first is encrypted traffic. Application security devices need to see the packets or transactions in clear text. If the encryption is done before the application security device, then the device will be unable to detect the threats.

Terminating the connection, decrypting the traffic and then re-encrypting it is one way around the problem, and is what vendors such as NetContinuum are doing. Imperva and Breach Security have a slightly different take. They only decrypt the traffic, examine it and make a decision. They keep the encrypted version in memory, and if they find no problem with the packet, they forward the encrypted version.

Another problem is that implementing a full network security scheme as outlined in Figure 2. is not simple. No vendor has a one-box solution. Putting together the entire solution requires several boxes or blades. Managing and correlating the data from all the different solutions is only partially possible. Having multiple boxes could also

increase the latency problem. While the industry is moving forward, a unified solution is more than a year away.

Application security is new to the market. Many improvements will be made in the coming years. Much of the technology, such as behavioral modeling, is just being developed, and has a while before it can catch all the major threats.

The key for enterprise managers is to take application security seriously and start developing a strategy which will include solutions from the vendors as well as development of more detailed policies□



**Much of this technology is just emerging**

#### **Companies Mentioned In This Article**

Breach Security ([www.breachsecurity.com](http://www.breachsecurity.com))  
Cisco ([www.cisco.com](http://www.cisco.com))  
Citrix ([www.citrix.com](http://www.citrix.com))  
ConSentry Networks ([www.consentry.com](http://www.consentry.com))  
Imperva ([www.imperva.com](http://www.imperva.com))  
Juniper ([www.juniper.net](http://www.juniper.net))  
Microsoft ([www.microsoft.com](http://www.microsoft.com))  
NetContinuum ([www.netcontinuum.com](http://www.netcontinuum.com))  
Protegrity ([www.protegrity.com](http://www.protegrity.com))  
Radware ([www.radware.com](http://www.radware.com))