

Realtime
publishers

"Leading the Conversation"

The Essentials Series

IT Compliance Volume II

sponsored by

SECURE[®]
COMPUTING

by Rebecca Herold

Reducing Attack Exposure for Internet-Facing Applications

by Rebecca Herold, CIPP, CISSP, CISA, CISM, FLMI

April 2007

Build In Security

The more software you have and the more options that are available for client machines to communicate with your software, the less secure your networks and data.

 Increasing complexity increases vulnerabilities.

If you have just one non-secured application, you open a potential attack path that may be exploited, circumventing all the other security you've implemented on your other servers and possibly allowing a nice little pathway through your firewall via that application vulnerability in ways that other direct attacks upon the firewall would fail. Figure 1 shows how, with even the strongest firewalls and applications, just the existence of one application could allow an attacker to wreak havoc throughout your network and your organization by shutting down access to the systems upon which you depend to conduct business.

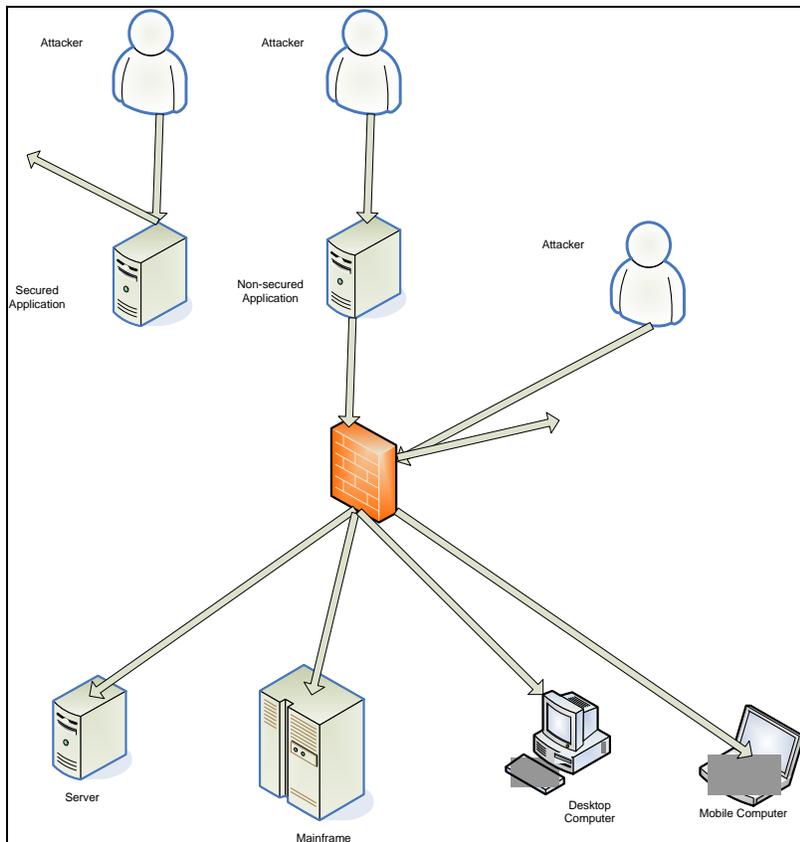


Figure 1: Attack paths through one application vulnerability.

Applications security is fundamentally a software engineering problem. It must be addressed in a systematic way throughout the entire software development life cycle. During this process, the software engineers and developers must minimize access to data and network resources using built-in granular controls to maximize application security.

Attack Methods

Lack of thorough and thoughtful security engineering can leave many vulnerabilities within an application. A small sample of attacks that exploit application vulnerabilities include:

Type of attack	Specific exploit
Denial of Service Attacks	Invalid packet lengths
	Flooding with random MAC addresses
	Malformed packet headers
Spoofing Attacks	Spoofing Internet headers
	Invalid Source IP with correct ports
	Race condition responses
Firewall Rule Set Bypass Attacks	Random UDP data to bypass rules
	Port scanning detection
	Connection prediction
Exploit	Send mail Header Overflow
	WFTPD Overflow
	Wu-IMAPD Overflow
Brute Force	Massive GET/POST requests
	Anomaly detection
Hijacking Sessions	XSS redirection (UNICODE)
Directory Transversal	Try “..” and other traversal methods
	UNICODE traversal methods
	Mixed strings

The most fundamental security action to help protect the enterprise from Internet-based attacks is to design your Internet-facing application to disable all capabilities not relevant to your organization’s use of the application. Another important defense not utilized nearly enough is to disallow all types of access by default and allow only those necessary based upon roles.

Positive Security Models vs. Negative Security Models

There are generally two methods currently used to defend against all types of application server attacks; the negative security model and the positive security model.

Negative Security Model

The typical method used by most security products and organizations is the negative security model. This approach basically identifies and disallows the specific types of traffic and access attempts already known to be threatening while allowing all other requests and access attempts. Antivirus and intrusion detection and prevention systems are classic examples of this approach. They both check requests and traffic flows against attack signatures. This type of approach used to work fairly well. However, new threats are emerging more frequently than ever before, and their number will continue to increase. This will result in considerably less time on an ongoing basis to react to the steadily increasing new attacks over time.

 A negative security model tries to figure out if there is something dangerous in the traffic. Negative security models are easy to start working with but you cannot create a foolproof rule set. Negative security rules are written only for known problems, exploits, and Web application attacks.

Positive Security Model

The positive security model denies all types of access and allows only those specific access capabilities to specific roles based upon authorizations. For example, consider the dozens of application-specific proxies that exist, such as application filtering for email, Citrix, Oracle, SQL, VoIP, Web, and other commonly used Internet protocols. When the positive security model is used, each proxy is configured according to your organization's unique use of the applications. This configuration then becomes the standard against which all traffic trying to use the application is checked.

Application-specific filters enable organizations to define very explicitly how the applications can be used based upon group rules. The firewall can then allow very granularly only those access capabilities based upon only the permitted use of these group rules. Such granular role-based group access controls help to ensure that suspicious traffic that does not conform to your corporate rules can be filtered and prevented from entering your enterprise, even for zero-day threats.

 Organizations need to establish granular role-based group access based upon the business' information security policies, business group responsibilities, leading industry practices, and RFC compliance.

The positive security model approach allows only legitimate, acceptable traffic elements and denies everything else.

 The positive security model tries to ensure the incoming data is safe to use, and that the incoming requests are safe. This is a much more effective goal than allowing all traffic and trying to detect all the bad stuff.

The positive security model is based upon sound security practices that have existed for decades, but were somehow lost along the way within most security product development efforts. A positive security model mirrors business application logic. So, whenever the application changes, the security model must be updated. For example, if you have an application variable such as “account” that will be all numerals, then you should have a rule defined that will check and ensure that the contents of “account” are all numerals or they will be disallowed. If your “account” changes to be all alphabetic, then you will also need to change your check to allow only all alphabetic “account” values to be allowed. This example is very simplistic, but it demonstrates the common sense approach of the positive security model.

 Although more secure, the positive security model is also more difficult to deploy because it needs to be custom configured to the specific application being protected.

Providing Granular Access to Applications

The positive security model utilizes granular access controls to application capabilities. Access controls can be made on a very specific level to any user, program, or process that requests permission to data or tries to perform specific business process activities. The access control subject (for example, the user ID), the access control object (for example, a specific database), or a combination of the two can define the access control authorizations.

The access control subject can be based upon:

- The time of day or day of request
- The location from where the access control subject authenticated
- Password or token utilized
- And so on

An individual access control subject may have different rights assigned to specific passwords that are used during the authentication process

The access control object can be based upon:

- Classification of the data content of the object
- Transaction restrictions
- And so on

The access control subject may be restricted from accessing all or part of the data within the access control object because of the type of data that may be contained within the object

The attributes of a subject are often referred to as privilege attributes or sensitivities. When these attributes are matched against the control attributes of an object, the privilege is either granted or denied.

Assign Access Privileges to Subjects Using Roles

The configuration of privileges in access control for an individual subject or user provides the most granularity. However, in enterprises with hundreds or thousands or more users, assigning and maintaining the granularity becomes a huge management burden. By giving multiple subjects similar permissions based upon their business roles, such as job titles or department teams, this granularity can still be achieved with more simplified access control administration.

Configure Access Privileges to Objects Using Roles

The configuration of privileges to access an individual object provides maximum granularity. It is not uncommon today for the number of objects within an access control system to number in the tens or even hundreds of thousands. Although configuring access to individual objects results in the maximum control, this granularity can quickly become an enormous administration burden.

It is a common practice to assign the appropriate permissions to a directory, then each object within the directory will inherit the respective parent directory permissions. By incorporating multiple objects with similar permissions or restrictions within a group or directory, the granularity is maintained but the administration of the access control system is simplified.

-  Stateful inspection firewalls only open and close ports; they are not role-based. Leaving ports wide open or completely closed does not provide for role-based granular access controls.
-  Network-layer firewalls have little awareness of the applications they are supposed to be protecting, so their connection controls cannot be granularly customized to each client's unique use of their own applications.

By implementing application access in granular ways, organizations can improve security and better monitor traffic. This can be achieved through the use of application-layer proxies. Granular access controls through the application's connections can allow for different roles to be given different types of access.

-  Implementing application-layer proxies significantly increases security. Stateful inspection firewalls only open and close ports.

Restricting Applications Capabilities

The concept of using granular access controls can also be used to restrict the capabilities of applications that will result in improved security. For example, you can restrict access to Internet-facing applications by screening all access requests through a tightly configured application gateway. By restricting application access in a granular method based upon roles, you can also prevent sensitive data leakage.

Firewall type	Characteristics
Application layer proxy	<ul style="list-style-type: none">• Communicates directly with the untrusted Internet requester to determine whether or not to allow the access through to the application server.• The only way an attack is going to get through to the internal application server is if there is an error in the proxy's logic, or the attacker invents a new attack that fits within the proxy's idea of 'acceptable traffic' for that protocol.
Network layer	<ul style="list-style-type: none">• Allows untrusted requestors to have a direct packet flow to the internal application server if the port is open.• Once the source/destination/origin has been deemed acceptable, the external client traffic is allowed straight through, unmodified, and passed directly to the internal system behind the firewall.

A major problem with application-layer proxies is that they are considered too slow. To address this issue, it is effective to remove the applications capabilities that you don't need so that the application-layer proxy does not need to spend more time checking those capabilities. This not only speeds the access request processing but also helps to prevent data leakage through capabilities you don't really need.

Get Rid of What You Don't Need

For example, what if you do not need to allow application users access to databases for some specific predefined storage procedures? Most organizations leave those capabilities intact. However, if you do not limit the access appropriately, this leaves the risk that an attacker could do something bad through that capability, such as mounting an SQL injection attack to retrieve, manipulate, or destroy data. By removing the capabilities that you do not need to perform your business activities, you lessen the chance that your business data will be inappropriately accessed, stolen, or worse.

Consider another example. Many applications have many different predefined privileges based upon user group or role authorization. However, often organizations do not assign roles granularly enough, lumping an entire department under one role because it is easier and quicker to do. That one role, however, is often one that can do a very wide range of actions leading to inadequate separation of duties privileges. As a result, there is no accountability or ability to perform per user authorization. There is also the very real possibility that fraud can occur by one of the users figuring out that, for instance, he can not only request a check but also approve the request and indicate to which address (one he has set up) it should be sent. Bottom line: If you do not need some of the application's capabilities, get rid of them.

Internet-Facing Applications Security Improvement Checklist

So, as a quick review, when developing and deploying Internet-facing applications:

- Incorporate security into the application throughout the entire development process
- Use a positive security model to establish access capabilities
- Provide granular access to applications
- Restrict applications' capabilities; disable those you do not need

These four actions will dramatically improve your applications' security capabilities and help to protect your business data.