Securing Microsoft Web Applications - A Guide for Systems Administrators

Matt Pogue

Securing Microsoft Web Applications - A Guide for Systems Administrators
Matt Pogue
February 18, 2003

Introduction

With the recent explosion of the Internet, e-commerce, and web-based applications, an Internet presence is now essential for many companies. Consumers have come to expect, and in some cases demand, the ability to interact with an organization via the web. As a result of this trend, many organizations are scrambling to deploy not only static content to the web, but also feature-rich applications that allow users to purchase goods and services, interact with customer support, manage their accounts, and perform many other functions. However, many times security and development "best practices" take a back seat to ease-of-use and speed to market.

In addition, most systems administrators rarely have an opportunity to interact with development teams as applications are being developed. As systems administrators, one of our primary duties is maintaining the integrity and security of our systems and networks. However, even the most hardened of systems can be quickly compromised by exploiting an insecure application that is running on it. Nowhere is this more evident than on the web.

The purpose of this paper is to provide systems administrators with a high-level overview of some of the major security considerations surrounding web applications that utilize Microsoft's Internet Information Server, SQL Server and Component Object Model (COM+), as well as links to in-depth technical information that expands upon the high-level topics discussed here. I will also discuss considerations for writing secure code, implementing secure DNS services, and packet filtering/proxy configurations. Finally, I will highlight and explain the need for more interaction between systems administrators and development staff during the initial planning and design phases of the development cycle.

Step One - Interact with Development Staff

Interaction with development staff during the initial planning and design phases of the development cycle is crucial to furthering a systems administrator's knowledge of how the application handles security, authentication, and access to protected resources. Most developers have an extremely limited knowledge of security mechanisms and procedures, and as a result, many applications are found to be lacking even the most basic security controls. It is not necessary for systems administrators to be involved in all phases of application development, but it is important to know what resources the application will need to access (e.g. SQL servers, Directory Services, files, UNC shares, etc.) and what authentication/access control mechanisms are being used to facilitate this access. Discussing these issues with developers during the planning phase of the development cycle will ensure proper security procedures are implemented throughout the application, and as any developer can tell you, it's much easier to plan and implement proper authentication mechanisms up front than to change existing mechanisms near the end of the cycle.

Step Two - Build a secure foundation

Hardening the network in front of your applications as well as the servers that run them is a crucial step in the deployment process. There are many different approaches to web application development, but in most cases, network and server configuration must be taken into consideration before the development process begins. Specific processes for building servers and configuring network devices will vary by organization and are beyond the scope of this document, however, the importance of this step cannot be underestimated. Some considerations at this stage include:

- Secure the base operating system installation (this includes removing all unnecessary services and applying all applicable service packs and hot fixes). This step may also involve applying a Local Security Policy. If your organization has a standard template for IIS servers, this should be applied. If your organization has no standard template in place, the National Security Agency (NSA) provides a set of templates for securing Windows 2000 servers at http://nsa1.www.conxion.com/win2k/download.htm.
- Secure IIS. This process can be automated to some extent with Microsoft's IISLockdown tool (see http://www.microsoft.com/windows2000/downloads/recommended/iislockdown/default.asp for more information about this tool). However, keep in mind that this tool is not the "silver bullet" for securing IIS and should always be examined in a test environment before deployment. Your organization may provide a checklist of some sort that defines standard IIS configurations or you may wish to use the NSA's template for IIS, which is located at http://nsa1.www.conxion.com/win2k/download.htm. You may also wish to consider encrypting or encoding your website's source code stored on the webserver's local disk. The "4 Guys From Rolla" website (http://www.4guysfromrolla.com) provides an excellent article on implementing encryption in Active Server Pages at http://www.4guysfromrolla.com/webtech/110599-1.2.shtml. The purpose of this article is a demonstration of encrypting query strings since these are visible in the address box of the browser, however, as the article mentions it would be possible to extend this example to encrypting entire pages on disk. Keep in mind that while providing an extra layer of security, you will take a performance hit due to the need to decrypt on every page request; and the more complicated your encryption scheme, the larger the performance hit. Microsoft provides their own implementation of this functionality via the Script Encoder component (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/SeconScriptEncoderOverview.asp). Keep in mind that neither of these methods can be considered 100% secure and will not stop the determined hacker from viewing your source code, but they will at least make it more difficult.
- Know your DNS strategy. For example, who will host DNS services for the site? If this function already exists in your organization, your existing DNS servers will be the ideal location. If you require a new DNS installation, then appropriate

steps should be taken to secure the DNS service. Detailed instructions on securing DNS are beyond the scope of this document, but those seeking more information will find an excellent guide to securing Windows 2000 DNS on the NSA's website at http://nsa1.www.conxion.com/win2k/download.htm. A template for securing BIND DNS on Linux systems can be found at http://www.linuxsecurity.com/resource_files/server_security/bind-8.x.txt.

- Know your packet filtering strategy. Chances are, you or your ISP already has a firewall in place and you are probably already blocking all ports except tcp port 80 (HTTP), and tcp port 443 (SSL). However, with the recent threat of worms targeting IIS such as CodeRed and Nimda, simple packet filtering is not enough. There are several "proactive" approaches that an organization can take to protect themselves against these types of worms. One approach is to use a reverse proxy, such as Apache web server with the mod_proxy module (http://httpd.apache.org/docs-2.0/mod/mod_proxy.html#forwardreverse), to proxy incoming HTTP requests. Another approach is to use a commercial application such as Eeye Digital Security's SecureIIS Application Firewall (http://www.eeye.com/html/Products/SecureIIS/index.html) as a wrapper for IIS.

Step Three - Secure the Back-End

The back-end data store is the crown jewel of most e-commerce and customer-centric dynamic sites, and should be treated as such. Stolen customer data from insecure databases is a very newsworthy item and it seems that new stories about data theft appear almost weekly. A web application built on an insecure database is like a house built on a foundation made of mud. Eventually, the house will collapse, it's just a matter of time. Some considerations at this stage include:

- If the application needs to access an SQL database, where is this server located? Does it reside in your organization's DMZ, in the corporate "trusted" LAN, or perhaps even on the same server as the web application? If it resides on a separate machine from the web application, what protocol is used for access? For environments where data encryption and/or data authentication is desired, consider using Windows 2000's IPSec in Transport mode to provide data encryption (ESP protocol) and/or data authentication (AH protocol) for all communications between the IIS server and the SQL server. Is the firewall configured properly to allow access to the server from the web server but restrict other, unwanted, access? What type of authentication is used to access the database? In Microsoft's SQL Server, your choices are Windows NT Authentication or Mixed-Mode Windows NT and SQL Server authentication. According to Microsoft,

> "Windows Authentication has certain benefits over SQL Server Authentication, primarily due to its integration with the Windows NT 4.0 and Windows 2000 security system. Windows NT

4.0 and Windows 2000 security provides more features, such as secure validation and encryption of passwords, auditing, password expiration, minimum password length, and account lockout after multiple invalid login requests"[1].

- If you are directly responsible for securing the SQL server, ensure that the base operating system is up to date with all the latest service packs, hot fixes, and patches. Ensure that the SQL Server application is up to date with all the latest services packs and patches. Recently, a worm (identified as "Kaiten", "W32/Voyager", or "W32/cblade.worm") was released that targets Microsoft SQL Servers that are configured with a blank password for the "sa" account (see CERT Incident Note IN-2001-13 at http://www.cert.org/incident_notes/IN-2001-13.html for more details). Above all else, restrict network access to the SQL server and do not employ a blank "sa" password!
- If the application will access a database, where is the authentication information (e.g. username/password/database server address) stored? Many developers are careless with this information. It is not uncommon to examine source code and find authentication information plainly visible in hidden HTML form tags and passed clear-text in URL query strings! For example, IIS applications have the ability to declare application- and session-level variables in the global.asa file located in the root of the site's documents folder. Many developers, as a matter of habit, place an application-level variable that defines the database connection object in this file. However, this is a dangerous practice. Several recent vulnerabilities allow an attacker to potentially view the source of files within the web root (see CERT advisory CA-2001-12 at http://www.cert.org/advisories/CA-2001-12.html and CERT Vulnerability Note VU#35085 at http://www.kb.cert.org/vuls/id/35085 for two examples). Two viable alternatives to this method include storing the connection string in the IIS metabase[2] or in the Windows registry. Either method is more secure than storing this information in the global.asa file, however, the latter will require a custom COM+ object for accessing the Windows registry .
- What database permissions are required? Many functions may require only read-only access. If this is the case, proper steps should be taken to ensure that the user being authenticated has access permissions no greater than what is required. For the purposes of logging, consider using two different accounts for read and write access to the database.

Is your CEO prepared to explain to your customers why their credit card numbers are now available on many popular hacker sites and newsgroups? If not, then proper implementation of the above procedures can help to ensure that he or she doesn't have to!

Step Four - Secure and Configure COM+ Objects

If you are like most systems administrators, you are probably asking, "What exactly is COM+?" Without going into detail that only a programmer could appreciate, COM+ is essentially an extension of Micorosoft's Component Object Model. Its purpose is to facilitate the development of business applications on the Microsoft platform and serves as Microsoft's answer to application development using Enterprise JavaBeans. COM+ also provides a way for applications to interact with various services such as Microsoft Transaction Server, Microsoft Message Queue Server, and Microsoft Distributed Transaction Coordinator.[3] COM+ objects can be coded using multiple languages and are implemented as compiled .exe's or .dll's.

In addition to the above, COM+ also implements its own security model. The COM+ security model provides a means to insulate developers from the underlying Windows and RPC security mechanisms so that as these technologies evolve, applications can immediately benefit from improvements in these technologies.[4]

From the administrator's perspective, COM+ means extra security considerations and configuration. COM+ object properties are configured separately from IIS by way of the Component Services MMC snap-in. Administrators can add COM+ applications which can each have multiple components associated with them. Some considerations for this step are as follows:

- How is security implemented within the application? COM+ implements a role-based security model that can be implemented either declaratively or programmatically.

  > "Declarative security means that you set up security outside the program code using the Component Services snap-in. The code does not do any work on the security front. Security is handled entirely by COM+. Programmatic security means that you write code to perform security checks and act appropriately based on the results. The two methods are not mutually exclusive; they can be used together if required."[5]

  The important thing to remember is that either method can be used, but programmatic security should be used with caution, and role checking should be done based on caller, not component.
- What services, applications, or filesystems does the application need access to? The approach to this should be to grant only the minimum permissions the application requires to run. A good practice would be to create an NT user to serve as the application's identity. Obviously, a component that does little more than send email notifications should not run in the local Administrator identity. Having a dedicated user for application authentication will also facilitate meaningful logging on the system.
- Has the application been thoroughly tested? Thorough testing is a crucial, but often overlooked, step in the development cycle. Buffer overflow vulnerabilities have historically been some of the most devastating ever discovered, and

regardless of the fact that they are some of the oldest known vulnerabilities, they still account for up to 50% of today's exploits.[6] A buffer overflow “is when a program allocates a block of memory of a certain length and then tries to stuff too much data into the buffer, with the extra overflowing and overwriting possibly critical information crucial to the normal execution of the program.”[7] However, by avoiding "dangerous" system calls that perform no bounds checking and performing proper bounds checking on all homegrown code before deployment to your live site, you can greatly mitigate the risk of buffer overflows.

COM+ application and component security is a complex topic and could easily fill several papers of this size. Guy Eddon has written an excellent article on the topic for the Microsoft Systems Journal. This article provides an excellent starting point for further exploration of COM+ security. The text of that article can be found at http://www.microsoft.com/msj/defaultframe.asp?page=/msj/1199/comsecurity/comsecurity.htm&nav=/msj/1199/newnav.htm.

References:

[1] Microsoft. "Administering SQL Server." MSDN Library http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/sql/ad_security_47u6.htm (29 Nov. 2001)

[2] Carpe, Thomas C. "Securing ASP Data Access Credentials Using the IIS Metabase." http://www.asp101.com/articles/thomas/metabase/default.asp (29 Nov. 2001)

[3] whatis.com. "COM+." http://whatis.techtarget.com/definition/0,289893,sid9_gci211825,00.html (29 Nov. 2001)

[4] Eddon, Guy. "The COM+ Security Model Gets You Out of the Security Programming Business." Microsoft Systems Journal November 1999 http://www.microsoft.com/msj/defaultframe.asp?page=/msj/1199/comsecurity/comsecurity.htm&nav=/msj/1199/newnav.htm (30 Nov. 2001)

[5] Shohoud, Yasser. "Programming COM+ Security." Middle Tier http://security.devx.com/upload/free/features/vcdj/2000/05may00/mt0500/mt0500.asp (30 Nov. 2001)

[6] McGraw, Gary, Viega, John. "Make Your Software Behave: Learning the Basics of Buffer Overflows." IBM DeveloperWorks March 2000 http://www-106.ibm.com/developerworks/library/overflows/ (3 Dec. 2001)

[7] Litchfield, David. "Exploiting Windows NT 4 Buffer Overruns, A Case Study: RASMAN.EXE." http://www.atstake.com/research/reports/wprasbuf.html (6 Dec. 2001)