NT IIS Vulnerabilities Involving Active Server Pages

Internet Research Project for GIAC LevelOne, SANS Institute
By
Nancy Gabriel

NT IIS (Microsoft Internet Information Server) is the built-in Web service of Windows NT Server 4.0 and the Windows 2000 Server. Microsoft claims around 25 percent of all Web sites are built on Microsoft servers, and that its share of e-commerce sites is closer to 50 percent.[1] Active Server Pages (version 2.0) were released in 1997 as part of IIS 4. They allow a remote client to take advantage of server-side scripting. Furthermore, ASP provides an array of objects and components which manage the interaction between the browser and the web server. Scripting languages such as VBScript and JScript are used to manipulate these objects. Version 2 of ASP was released in December of 1997 as part of IIS 4.[2]

The functionality of active server pages, especially when used with MS SQL Server to access a database, allows for powerful business applications on intranets and the Internet. However, the ease of interoperability can also expose the underlying host and network to abuse by intruders. The default configuration of IIS has created serious security vulnerabilities that have enabled compromises to the underlying host.[3] The results have been unrestricted access to databases (passwords and credit card numbers), access to web root files, defaced websites, access to other drives on the host, and denial-of-service attacks.

This report pulls together many of the recently exposed vulnerabilities and exploits involving NT IIS and active server pages. Protecting the server and website from these attacks depends firstly on the correct configuration of IIS and subsequent follow-ups with advisories and patches. Secondly, both webpage programmers and system administrators need to prevent unexpected characters , illegal pathnames and malformed URL's from reaching the server by using appropriate input field validation and perhaps a custom ISAPI filter.

The recent NT IIS/ASP vulnerabilities seem to fall loosely into several categories—those due primarily to DLLs or sample pages in the default NT IIS setup, those due to inadequate checking of form data in the ASP code itself, and problems with the web directory structure.

Sample scripts and problems within the ODBC/RDS components have been used to gain administrative access to vulnerable NT servers. Remote Data Service (RDS), a component of MDAC, is part of the default set-up for IIS 3.0 and 4.0. The Remote Data Service is designed to enable Web clients to issue client-based SQL queries to remote data resources hosted on the IIS Web server, using http. The remote client communicates with the DLL, msadcs.dll, on the server. ODBC (Open Database Connectivity) allows a program access to one or more relational databases using SQL. If a client fails to quote correctly the meta characters in a piece of data used in an SQL query, an attacker may be able to interfere with the tables in the database. The MS Jet database engine (which runs Access databases) allows an individual to embed VBA (Visual Basic for Applications) in string expressions, which may allow the individual to run commandline NT commands. This, combined with IIS running ODBC commands as system_local allow a remote attacker to gain full control of the system. The attacker can either use an existing Data Source Name (DSN), or can manually specify the location of a .mdb file on the server. Therefore, any default .mdb file or DSN on the vulnerable server may be used to launch the attack. The Windows NT 4.0 Option Pack installs several sample .mdb files in the /msadcs/Samples directory, and they were used to get a foothold into servers. A common exploit using the RDS vulnerability was to replace or deface corporate websites. Another exploit resulted in passwords being taken from a database.[4-7,25]

Another default DLL in NT is an ISAPI application, webhits.dll, included in the default setup of Index Server. Files that have the extension .htw are dispatched by webhits.dll. A vulnerability exists in webhits.dll that allows an attacker to break out of the web virtual root file system and gain unauthorized access to other files on the same logical disk drive, such as customer databases, log files or any file they know or can guess the path to. The same vulnerability can be used to obtain the source of Active Server Pages or any other server-side script file which often contain User IDs and passwords as well as other sensitive information. By appending a space (%20) to the end of the filename specified in the 'CiWebHitsFile' variable, and setting 'CiHiliteType' to 'Full' and 'CiRestriction' to 'None', it is possible to retrieve the unprocessed source of the file.[8]

The most recent DLL used to compromise the server is dvwssr.dll (and the corresponding client-side mtd2lv.dll). It is included with the FrontPage 98 extensions for IIS and shipped as part of the NT Option Pack. It includes an obfuscation string that manipulates the name of requested files. Knowing this string and the obfuscation algorithm allows anyone with web authoring privileges on the target host to download any .asp or .asa source on the system (including files outside the web root, through usage of the '../' string). This includes users with web authoring rights to only one of several virtual hosts on a system, allowing one company to potentially gain access to the source of another company's website if hosted on the same physical machine.[9,10]

Active Server Pages programs will often exchange information with a database, and may have authentication information embedded in their source code. For example, a Windows NT IIS web server may make network queries to a Microsoft SQL server from an ASP application. If the source code of the ASP contains usernames and passwords to the SQL server, then the SQL server may be directly accessed with a generic SQL client.[11] A synopsis of recent vulnerabilities used to view ASP source code follow (the exploits and solutions are found in each reference).

The File System Object (FSO) may be called from an Active Server Page (ASP) to display files that exist outside of the web server's root directory. FSO allows calls to be made utilizing "../" to exit the local directory path. An example of this syntax would be http://www.server.foo/showfile.asp?file=../../global.asa. This vulnerability could be used to view the source code of ASP files or stream data into other ASP files on the web server.[12]

The sample pages viewcode.asp, codebrws.asp and showcode.asp are present in all default installations of Site Server and IIS and could allow intruders to view sensitive or compromising information from that system. They are found in the /msadc/Samples directory. For instance, showcode.asp does inadequate security checking and allows anyone with a web browser to view the contents of any text file on the web server. This includes files that are outside of the document root of the webserver. The problem is the security check for form input does not test for the '..' characters within the URL. In 1997, with IIS 3.0, obtaining the ASP source code was as easy as replacing '.' in a URL with another escaped character, '%2e'. According to the L0pht, "Not only can the server-side scripting in .asp files be viewed by a remote user, but any text file on the system. For e-commerce servers, this puts transaction logs, credit-card numbers and customer information potentially at risk. There is even e-commerce shopping cart software that stores administrative passwords in the clear in text files."[13,14]

Active Server Pages with errors in them or their components can cause error messages to be served to the browser that include path information for included files used in the creation of the .asp file. These files can then be downloaded and may include sensitive information such as resource locations, website and network structure, and business models. A properly implemented and audited ASP will not exhibit this behavior.[15]

Other vulnerabilities in IIS involve the webserver directory structure. In one case, it was found that two separate web servers may be configured to share the same physical directory on an IIS directory. There may be instances where ASP information containing confidential information (from site A) is presented in ASP information served to a user from site B.[16] It was also found that server side processing of web pages can be bypassed under a specific set of conditions. If an .asp or similar file resides in a virtual directory whose name ends in a legal extension, the source code of the file may be sent to the client browser. IIS determines what action to take on a web document by parsing the URL for the filename extension. The first registered extension found in the string is used to make this decision. Therefore, if a request is made for the following file: /webroot/docs.htm/some.asp, and that file exists, IIS will parse the path and find the .htm extension in the virtual directory name docs.htm, determine that no preprocessing is required, and send the unprocessed source of the some.asp file.[17]

Even though patches and upgrades to IIS have closed a number of holes due to malformed URLs, they can still be exploited for a denial-of-service attack. A malicious user can request a document using a URL with a very large number of escaped characters arranged in a particular manner. The algorithm used to replace the escaped characters behaves inefficiently and ties up the server. There are ways to limit the length of URL accepted for processing by the server.[18]

All of the vulnerabilities described have solutions and some have links to Microsoft patches detailed in the reference links below. The best foundation is to set up the NT Server with security in mind from the beginning[19], and then configure the Information Server using all the best practices developed in light of known vulnerabilities.[20,21] Turning on only the necessary services, changing NT Registry settings from defaults where recommended by security advisories, setting directory permissions, and removing unnecessary DLLs and sample pages are some of the steps that can be taken. The subdirectories containing SQL scripts and application data, the DLL directory and the Include-files directory should be separate and not placed directly under the site directory. Do not use scripts in .inc files.[22] Keeping up with patches and advisories by joining a security users group or mailing list is also recommended. Following the thought process used to find the combination of server and form input weaknesses to carry out an attack is also instructional [25]—it will help you think defensively.

When accepting form input with active server pages, think of how an assembled URL or database query could be corrupted (accidentally or purposefully) and use scripting to screen these cases out. Some of these weaknesses are inherent with the use of any input forms. Check for escaped characters, the pipe command, unquoted input, or '../' in a pathname string, depending on what application the input is used for. Some examples of user input screening are given in the references.[23-25]

When the Web server receives a request for a file, the Internet Information Server first checks the registered ISAPI filters. An ISAPI filter is a DLL that exports two or three specific functions that can then be called by IIS. The filter is loaded on the server and can be installed so that all requests for all applications on the server are monitored or only requests for a single application are monitored. Active Server Pages rely on an ISAPI filter to catch .asp files prior to returning anything to the Web browser. These files are then processed by the ISAPI filter, which strips out all "<%%>" tags, compiles VB Script, and invokes any components called, while making Application and Session Objects available during the processing of all Active Server scripts. All of this occurs prior to the Web server returning the results to the Web browser. ISAPI filters can be used for custom authentication and logging for an application receiving user input.[26,27] It may be possible to use the logging utility for detecting repeated failed attempts to connect to an application such as SQL Server.

The security vulnerabilities and resulting exploits noted above involve both weaknesses in the default setup of NT IIS *and* inadequate screening of user input data from active server pages. Therefore, the solution must involve a collaborative effort between both system and software people. One idea is to get the system administrator and web developers together to produce a coding convention specifically addressing the known security vulnerabilities. It could be reviewed and updated in light of new exploits. The same interoperability that enables powerful applications also creates a complex system—what one person can get their arms around the entire structure? Even the hackers need to collaborate to figure out what the various DLLs in IIS are for. Eventually, through experimentation and consultation, someone hits on the right mode of access and exploits a vulnerability. That "someone" can be a corporate team, proactively closing security holes…or someone else.

Bibliography

[1] Brown, Kent. "DNA 2000: Opening new Windows" Nov. 1999. URL: http://www.devx.com/upload/free/features/entdev/1999/11nov99/cv1199/cv1199.asp (4/29/00).

[2] Programmers Resource.com "What are Active Server Pages (ASP)?" URL: http://www.programmersresource.com/articles/whatisasp.asp (4/29/00)

[3] Security Focus "Locking Down IIS, HTR and RDS Attacks, patches, and vulnerabilities" last updated Dec. 6 1999. URL: http://www.securityfocus.com/frames/index.html?focus=microsoft (4/29/00)

[4] Shipley, Greg. “Securing Windows NT Server” Apr. 3, 2000. URL: http://www.nwc.com/1106/1106ws12.html (4/29/00)

[5] Puppy, Rain Forest. “RDS/IIS 4.0 Vulnerability and exploit” 1999. URL: http://www.wiretrip.net/rfp/p/doc.asp?id=1&iface=2
“Update to ODBC/RDS vulnerabilities” Sep. 21, 1999. URL: http://www.wiretrip.net/rfp/p/doc.asp?id=4&iface=2
“Defending against RDS attacks”. URL: http://www.wiretrip.net/rfp/p/doc.asp?id=29&iface=2 (4/29/00)

[6] Cheney, Kirk. “Vulnerability in Microsoft Data Access Components (MDAC) for Internet Information Server (IIS)” Apr. 18, 2000. URL: http://www.sans.org/infosecFAQ/MDAC.htm (4/28/00)

[7] Puppy, Rain Forest and Astley, Matthew. “Advisory: NT ODBC Remote Compromise” May 25, 1999. URL: http://www.securityportal.com/list-archive/bugtraq/1999/May/0216.html (4/29/00)

[8] Litchfield, David. “Webhits.dll buffer truncation” Jan. 26, 2000. URL: http://ntbugtraq.ntadvice.com/default.asp?pid=36&sid=1&A2=ind0001&L=NTBUGTRAQ&P=R4201 and “MS Index Server ‘%20’ ASP Source Disclosure Vulnerability” Mar. 30, 2000. URL: http://www.securityfocus.com/bid/1084 (4/29/00)

[9] Serer, Al and Puppy, Rain Forest. “MS IIS FrontPage 98 Extensions Filename Obfuscation Vulnerability” Apr. 14, 2000. URL: http://www.securityfocus.com/bid/1108 (4/29/00)

[10] Puppy, Rain Forest. “A back door in Microsoft FrontPage extensions.” URL: http://www.wiretrip.net/rfp/p/doc.asp?id=45&iface=2 (4/29/00)

[11] Gula, Ron. “The top 14 things your ethical hackers for hire didn’t test.” Jul. 1999. URL: http://www.securitywizards.com/papers/pentest.txt (4/29/00)

[12] Geisbert, Gary. “The File System Object (FSO) may be called from an Active Server Page” Feb. 11, 1999. Updated Apr. 11, 2000. URL: http://www.securityfocus.com/bid/230 (4/28/00)

[13] Pond, Weld. “Web users can view ASP source code and other sensitive files on the web server” May 7, 1999. URL: http://www.l0pht.com/advisories/showcode.txt and an earlier exploit, http://www.l0pht.com/advisories/asp.txt (1997) (4/28/00)

[14] Diederich, Tom. “Microsoft issues Net server security bulletin” May 5, 1999. URL: http://www.idg.net/crd_internet_73976.html (4/29/00)

[15] Walsh, Jerry. “NT IIS ASP VBScript Runtime Error Viewable Source Vulnerability” Feb. 9, 2000. URL: http://www.securityfocus.com/bid/978 (4/29/00)

[16] Hamilton, Ivan. “ASP Caching Bug” Jan. 27, 1999 updated Apr. 11, 2000. URL: http://www.securityfocus.com/bid/195 (4/29/00)

[17] Hunger, Adam. “Microsoft IIS Virtual Directory Naming Vulnerability” Dec. 21, 1999. URL: http://www.securityfocus.com/bid/882 (4/29/00)

[18] Microsoft Security Bulletin (MS00-023): Escaped Characters Vulnerability Last updated Apr. 12, 2000. URL: http://www.microsoft.com/technet/security/bulletin/fq00-023.asp (4/29/00)

[19] “Windows NT Security: Step-by-Step”, published by the SANS Institute. Available through URL: http://www.sans.org/newlook/publications/ntstep.htm (4/29/00)

[20] “MS Internet Information Server 4.0 Security Checklist” Nov. 3, 1999. URL: http://www.microsoft.com/TechNet/iis/technote/iischeck.asp (4/28/00)

[21] Enfield, Paul. “Implementing a Secure Site with ASP” 1997 with updates to 2000. URL: http://www.microsoft.com/technet/IIS/technote/security.asp

And The Microsoft Security site. URL: http://www.microsoft.com/technet/security/default.asp (4/28/00)

[22] Meade, John. "Planning for ASP" 1999. URL: http://www.microsoft.com/technet/iis/planasp.asp (4/28/00)

[23] Forristal , Jeff. "Maintaining Secure Web Applications" Mar. 20, 2000. URL: http://www.nwc.com/1105/1105ws1.html, page 4. (4/29/00)

[24] Hui, Vernon. "Microsoft Beefs up VBScript with Regular Expressions" May 10, 1999. URL: http://msdn.microsoft.com/workshop/languages/clinic/scripting051099.asp (4/29/00)

[25] Puppy, Rain Forest. "How I hacked PacketStorm" URL: http://www.wiretrip.net/rfp/p/doc.asp?id=42&iface=2 (4/29/00)

[26] NetLine ASP Online Book 2000. URL: http://mogwai.netline.be/forem/asp/ch10.htm (4/27/00)

[27] Reilly, Douglas. "From: Inside Server-Based Applications—Chapter 11: ISAPI Filters (a sample chapter online)" 2000. URL: http://mspress.microsoft.com/prod/books/sampchap/1547.htm#167 (4/29/00)