

External Vulnerability Assessment

-Technical Summary-

Prepared for:

ABC ORGANIZATION

On

March 9, 2008

Prepared by:

AOS Security Solutions

Table of Contents


Executive Summary.....	3
Discovered Security Vulnerabilities.....	3
HTTP Windows 98 MS/DOS device names DOS	3
Database Server Error Message	4
IIS Global Server Variables Disclosure (global.asa.bak).....	4
Microsoft ASP.NET or ASP Unicode Conversion Cross-Site Scripting	5
Cross-Site Scripting	6
SQL Injection (confirmed).....	7
Backup File (.old)	8
WebLog Administrative Access Bypass.....	9
Backup File (Appended .bak)	9
Admin Section Must Require Authentication	10
Logins Sent Over Unencrypted Connection.....	11
Possible Username or Password Disclosure.....	12
Appendix A – External Scan Reports	13
Appendix B – Web App Penetration Testing Documents	13

Executive Summary

Alexander Open Systems (AOS) performed an external security vulnerability assessment of ABC ORGANIZATION. An external security vulnerability assessment looks at the security of devices such as firewalls, servers and routers that provide services on the Internet. It also covers application layer assessments on any web based services externally facing.

The security assessments performed by AOS follow a standard assessment methodology beginning with reconnaissance, vulnerability enumeration and penetration testing for validation. AOS performs these assessments with the least possible impact to the organization. This means our assessment tools have been throttled back as to not consume customer internet bandwidth as well as our assessments are done at a mutually agreeable time that is least impacting to the organization. The following section is the findings of AOS's vulnerability assessment.

Discovered Security Vulnerabilities

The following is a list of discovered security vulnerabilities, a description of the problems associated with each vulnerability, recommendations for mitigating the vulnerability and the procedure to complete the recommended mitigating action. The vulnerabilities have been rated with by the overall security risk to the systems at ABC ORGANIZATION. Above each vulnerability is a multi-colored bar with a black arrow such as this:  The red segment indicates the vulnerability is of the highest risk to the systems at ABC ORGANIZATION. The dark green segment indicates the vulnerability is the lowest risk to the organization with the red being the highest risk.



HTTP Windows 98 MS/DOS device names DOS

Problem:	It was possible to freeze or reboot Windows by reading a MS/DOS device through HTTP, using a file name like CON\CON, AUX.htm or AUX.
Recommendation:	upgrade your system or use a HTTP server that filters those names out.
Affected Servers:	<ul style="list-style-type: none"> 24.28.199.152



Database Server Error Message

Problem:	The most common cause of an unhandled exception is a failure to properly sanitize client-supplied data that is used in SQL statements. They can also be caused by a bug in the web application's database communication code, a misconfiguration of database connection settings, an unavailable database, or any other reason that would cause the application's database driver to be unable to establish a working session with the server. The problem is not that web applications generate errors. All web applications in their normal course of operation will at some point receive an unhandled exception. The problem lies not in that these errors were received, but rather in how they are handled. Any error handling solution needs to be well-designed, and uniform in how it handles errors. For instance, assume an attacker is attempting to access a specific file. If the request returns an error File not Found, the attacker can be relatively sure the file does not exist. However, if the error returns "Permission Denied," the attacker has a fairly good idea that the specific file does exist. This can be helpful to an attacker in many ways, from determining the operating system to discovering the underlying architecture and design of the application.
Recommendation:	Removing Detailed Error Messages
Procedure:	Find instructions for turning off detailed error messaging in IIS at this link: http://www.microsoft.com/windows2000/en/server/iis/default.asp?url=/windows2000/en/server/iis/htm/core/iiercst.htm Use the following syntax to suppress error messages on an Apache server. Syntax: ErrorDocument <3-digit-code> Example: ErrorDocument 500 /webserver_errors/server_error500.txt
Affected Servers:	<ul style="list-style-type: none"> • http://zero.webappsecurity.com/ ON • 24.28.199.152



IIS Global Server Variables Disclosure (global.asa.bak)

Problem:	An information disclosure vulnerability has been detected on an IIS server. During the assessment, the global.asa.bak file was retrieved from the web server. This file is most likely a backup copy of the global.asa file, which is a text-based file that consists of server-side script that defines application- or session-level variables that users will use throughout their web session. The global.asa file may contain a database server name, user name, database password, and database name. If exploited by a remote attacker, the sensitive system information in the file would be disclosed. Recommendations include removing the vulnerable file.
Recommendation:	Removing Detailed Error Messages
Procedure:	A common characteristic of a secure web application is a clean webroot. The more clutter that fills a webroot, the higher the chances are that a piece of information that might be of value to an attacker has been overlooked. The following recommendations will help to ensure that you are maintaining

	<p>a secure web application.</p> <ul style="list-style-type: none"> • Webroot Security Policy: Implement a security policy that prohibits backing up source code in the webroot. • Cleanup Script: Create a script that periodically deletes common backup files (.old, .bak, .tmp, .old, .arc, .orig, .backup, and so on) from your webroot. • Temporary Files: Many tools and html editors automatically create temporary or backup files in your web root. Be careful when editing files on a production server that you are not inadvertently leaving backup or temporary copies of those files in the webroot. • Test Server: Do not use your production server for testing out new scripts. Instead, create a testing server for that purpose. • Default Installations: Often, a lot of unnecessary files and folders are installed by default. For instance, IIS installations include demo applications. Be sure that you remove any files or folders that you are not actively using.
Affected Servers:	<ul style="list-style-type: none"> • http://zero.webappsecurity.com/global.asa.bak ON • 24.28.199.152



Microsoft ASP.NET or ASP Unicode Conversion Cross-Site Scripting

Problem:	<p>A Unicode conversion Cross-Site Scripting (XSS) vulnerability was found. This vulnerability is due to an input validation error in the filtration of special HTML characters supplied as Unicode characters. If exploited, an attacker could craft a malicious link containing arbitrary HTML or script code to be executed in a user's browser. Recommendations include modifying the web.config file to use only Unicode code page for output or filtering full-width ASCII characters from all non-trusted data sources.</p> <p>The application fails to properly validate Unicode characters in the "Request Validation" and "HttpServerUtility.HtmlEncode" security mechanisms. If exploited, an attacker could control the Web browser of other Web users who view the page by embedding malicious HTML tags and JavaScript. An attacker could use this technique to steal sensitive information such as credit card numbers, usernames, passwords, files, and session identifiers from the Web users.</p>
Recommendation:	Modify the web.config file to use only Unicode code page for output.
Procedure:	<ul style="list-style-type: none"> • To do this, add the following lines to your web.config file: <pre><configuration> <system.web> <globalization responseEncoding="utf-8" /> </system.web> </configuration></pre> <p>If you cannot use Unicode, have your developers to filter full-width ASCII characters from all non-trusted data sources, such as user input, HTTP</p>

	<ul style="list-style-type: none"> headers, some components output, and other data. http://www.securityfocus.com/bid/12574
Affected Servers:	<ul style="list-style-type: none"> http://zero.webappsecurity.com/attack.aspx?test=%uff1cscript%uff1ealert('vulnerability')%uff1c/script%uff1e ON 24.28.199.152



Cross-Site Scripting

Problem:	<p>Cross-Site Scripting vulnerabilities were verified as executing code on the web application. Cross-Site Scripting occurs when dynamically generated web pages display user input, such as login information, that is not properly validated, allowing an attacker to embed malicious scripts into the generated page and then execute the script on the machine of any user that views the site. In this instance, the web application was vulnerable to an automatic payload, meaning the user simply has to visit a page to make the malicious scripts execute. If successful, Cross-Site Scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be mistaken for those of a valid user, compromise confidential information, or execute malicious code on end user systems. Recommendations include implementing secure programming techniques that ensure proper filtration of user-supplied data, and encoding all user supplied data to prevent inserted scripts being sent to end users in a format that can be executed.</p>
Recommendation:	<p>Cross-Site Scripting attacks can be avoided by carefully validating all input, and properly encoding all output</p>
Procedure:	<ul style="list-style-type: none"> Fixes for Cross-Site Scripting defects will ultimately require code based fixes. <p>SPI Dynamics Cross-Site Scripting Whitepaper http://www.spidynamics.com/spilabs/education/whitepapers/CrossSiteScripting.html</p> <p>OWASP Cross-Site Scripting Information http://www.owasp.org/documentation/topten/a4.html</p> <p>Microsoft http://support.microsoft.com/default.aspx?scid=kb;EN-US;q252985</p> <p>Microsoft Anti-Cross-Site Scripting Library V1.0 http://www.microsoft.com/downloads/details.aspx?familyid=9a2b9c92-7ad9-496c-9a89-af08de2e5982&displaylang=en</p> <p>CERT http://www.cert.org/advisories/CA-2000-02.html</p> <p>Apache http://httpd.apache.org/info/css-security/apache_specific.html</p> <p>Netscape</p>

	http://channels.netscape.com/ns/browsers/security.jsp SecurityFocus.com http://www.securityfocus.com/infocus/1768
Affected Servers:	<ul style="list-style-type: none"> • Several pages are affected (see raw data for full details) ON • 24.28.199.152



SQL Injection (confirmed)

Problem:	Critical SQL Injection vulnerabilities have been identified in the web application. SQL injection is a method of attack where an attacker can exploit vulnerable code and the type of data an application will accept
Recommendation:	Each method of preventing SQL injection has its own limitations. Therefore, it is wise to employ a layered approach to preventing SQL injection, and implement several measures to prevent unauthorized access to your backend database.
Procedure:	<ul style="list-style-type: none"> • Restrict Application Privileges: Limit user credentials so that only those rights the application needs to function are utilized. Any successful SQL Injection attack would run in the context of the user's credential. While limiting privileges will not prevent SQL Injection attacks outright, it will make them significantly harder to enact. • Strong SA Password Policy: Often, an attacker will need the functionality of the administrator account to utilize specific SQL commands. It is much easier to "brute force" the SA password when it is weak, and will increase the likelihood of a successful SQL Injection attack. Another option is not to use the SA account at all, and instead create specific accounts for specific purposes. • Consistent Error Messaging Scheme: Ensure that you provide as little information to the user as possible when a database error occurs. Don't reveal the entire error message. Error messages need to be dealt with on both the web and application server. When a web server encounters a processing error it should respond with a generic web page, or redirect the user to a standard location. Debug information, or other details that could be useful to a potential attacker, should never be revealed. Application servers, like WebSphere, often install with error messages or debug settings enabled by default. Consult your application server's documentation for information on suppressing those error messages. • Stored Procedures: If unused, delete SQL stored procedures such as master..Xp_cmdshell, xp_startmail, xp_sendmail, and sp_makewebtask. • Parameterized Queries: SQL Injection arises from an attacker's manipulation of query data to modify query logic. The best method of preventing SQL Injection attacks is thereby to separate the logic of a query from its data. This will prevent commands inserted from user input from being executed. The downside of this approach is that it can have an impact on performance, albeit slight, and that each query on the site must be structured in this method for it to be completely effective. If one query is inadvertently bypassed, that could be enough to leave the application vulnerable to SQL Injection. The following code

	<p>shows a sample SQL statement that is SQL injectable.</p> <ul style="list-style-type: none"> • For More information see: • Microsoft http://msdn.microsoft.com/msdnmag/issues/04/09/SQLInjection/default.aspx http://support.microsoft.com/default.aspx?scid=kb;en-us;302570 <p>SQLSecurity.com http://www.sqlsecurity.com/DesktopDefault.aspx</p> <p>OWASP http://www.owasp.org/index.php/SQL_Injection</p>
Affected Servers:	<ul style="list-style-type: none"> • Several pages are affected (see raw data for full details) ON • 24.28.199.152



Backup File (.old)

Problem:	<p>A serious vulnerability in your web application has been detected due to the presence of a backup file on your server. Often, old files are renamed with an extension such as .old to distinguish them from production files. The source code for old files that have been renamed in this manner and left in the webroot can often be retrieved. At a minimum, an attacker who retrieves this file would have all the information contained in it, whether that be database calls, the format of parameters accepted by the application, or simply information regarding the architectural structure of your site. Recommendations include implementing a security policy within your organization that prohibits backing up web application source code in the webroot.</p>
Recommendation:	<p>Ensure that your developers are not leaving things of value to a potential attacker publicly available via your web application.</p>
Procedure:	<p>For Security Operations: A common characteristic of a secure web application is a clean webroot. The more clutter that fills a webroot, the higher the chance that a piece of information that might be of value to an attacker has been overlooked. The following recommendations will help to ensure that you are maintaining a secure web application.</p> <ul style="list-style-type: none"> • Webroot Security Policy: Implement a security policy that prohibits backing up source code in the webroot. • Cleanup Script: Create a script that periodically deletes common backup files (.old, .bak, .tmp, .old, .arc, .orig, .backup, and so on) from your webroot. • Temporary Files: Many tools and html editors automatically create temporary file or backup files in your web root. Be careful when editing files on a production server that you are not inadvertently leaving a backup or temporary copy of that file in the webroot. • Test Server: Do not use your production server for testing out new

	<p>scripts. Instead, create a testing server for that purpose.</p> <ul style="list-style-type: none"> • Default Installations: Often, a lot of unnecessary files and folders are installed by default. For instance, IIS installations include demo applications. Be sure that you remove any files or folders that you are not actively using. <p>For Development: Under no circumstances should source code be backed up in a zip file or renamed with a different extension and left available on the webroot. While the solution is black and white in this instance, it does indicate the need for adopting programming standards that include a secure design component. Take pains to ensure information that could be utilized by an attacker is not included within your web application.</p>
Affected Servers:	<ul style="list-style-type: none"> • http://zero.webappsecurity.com:80/rootlogin.old • http://zero.webappsecurity.com:80/rootlogin.asp.old ON • 24.28.199.152



WebLog Administrative Access Bypass

Problem:	<p>phpWebLog version 0.4.2 (with PHP versions below 4.0 rc1).</p> <p>In common.inc.php, \$CONF is not properly initialized as an array; this allows users to alter the contents stored inside it. The alteration of the content allows attackers to bypass the administrative authentication.</p>
Recommendation:	<p>Since currently there is no known information about this vulnerability, the recommendation is to remove this file if it is not needed for the production server.</p>
Procedure:	<p>Remove file if not needed</p>
Affected Servers:	<ul style="list-style-type: none"> • http://zero.webappsecurity.com:80/scripts/weblog • 24.28.199.152



Backup File (Appended .bak)

Problem:	<p>A serious vulnerability has been detected in your web application due to the presence of a backup file on your server. Often, old files are renamed with an appended extension such as .bak to distinguish them from production files. The source code for old files that have been renamed in this manner and left in the webroot can often be retrieved. At a minimum, an attacker who retrieves this file would have all the information contained in it, whether that be database calls, the format of parameters accepted by the application, or simply information regarding the architectural structure of your site. Recommendations include implementing a security policy within your organization that prohibits backing up web application source code in the webroot.</p>
----------	---

Recommendation:	Ensure that your developers are not leaving things of value to a potential attacker publicly available via your web application.
Procedure:	<p>For Security Operations: A common characteristic of a secure web application is a clean webroot. The more clutter that fills a webroot, the higher the chance that a piece of information that might be of value to an attacker has been overlooked. The following recommendations will help to ensure that you are maintaining a secure web application.</p> <ul style="list-style-type: none"> • Webroot Security Policy: Implement a security policy that prohibits backing up source code in the webroot. • Cleanup Script: Create a script that periodically deletes common backup files (.old, .bak, .tmp, .old, .arc, .orig, .backup, and so on) from your webroot. • Temporary Files: Many tools and html editors automatically create temporary file or backup files in your web root. Be careful when editing files on a production server that you are not inadvertently leaving a backup or temporary copy of that file in the webroot. • Test Server: Do not use your production server for testing out new scripts. Instead, create a testing server for that purpose. • Default Installations: Often, a lot of unnecessary files and folders are installed by default. For instance, IIS installations include demo applications. Be sure that you remove any files or folders that you are not actively using. <p>For Development: Under no circumstances should source code be backed up in a zip file or renamed with a different extension and left available on the webroot. While the solution is black and white in this instance, it does indicate the need for adopting programming standards that include a secure design component. Take pains to ensure information that could be utilized by an attacker is not included within your web application.</p>
Affected Servers:	<ul style="list-style-type: none"> • http://zero.webappsecurity.com:80/pindex.asp.bak • http://zero.webappsecurity.com:80/default.asp.bak • http://zero.webappsecurity.com:80/confirm.asp.bak • http://zero.webappsecurity.com:80/rootlogin.asp.bak • http://zero.webappsecurity.com:80/login.asp.bak • http://zero.webappsecurity.com:80/admin/help.cgi.bak <p>ON 24.28.199.152</p>



Admin Section Must Require Authentication

Problem:	This policy states that any area of the website or web application that contains sensitive information or access to privileged functionality such as remote site administration requires authentication before allowing access. The URL http://zero.webappsecurity.com:80/admin/help.cgi has failed this policy.
----------	---

Recommendation:	Restrict access to important directories or files.
Procedure:	<p>Apache: Security Tips for Server Configuration Protecting Confidential Documents at Your Site Securing Apache - Access Control</p> <p>IIS: Implementing NTFS Standard Permissions on Your Web Site</p> <p>Netscape: Controlling Access to Your Server</p> <p>General: Password-protecting web pages Web Security</p>
Affected Servers:	<ul style="list-style-type: none"> • http://zero.webappsecurity.com:80/admin/help.cgi • http://zero.webappsecurity.com:80/admin/WS_FTP.LOG • http://zero.webappsecurity.com:80/admin/help.cgi.BAK • http://zero.webappsecurity.com:80/admin/help.cgi.bak <p>ON</p> <ul style="list-style-type: none"> • 24.28.199.152



Logins Sent Over Unencrypted Connection

Problem:	<p>Any area of a web application that possibly contains sensitive information or access to privileged functionality such as remote site administration functionality should utilize SSL or another form of encryption to prevent login information from being sniffed or otherwise intercepted or stolen. The URL http://zero.webappsecurity.com:80/stats/ has failed this policy. Recommendations include ensuring that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.</p> <p>An attacker who exploited this design vulnerability would be able to utilize the information to escalate their method of attack, possibly leading to impersonation of a legitimate user, obtaining proprietary data, or simply executing actions not intended by the application developers</p>
Recommendation:	Any area of a web application that possibly contains sensitive information or access to privileged functionality such as remote site administration functionality should utilize SSL or another form of encryption to prevent login information from being sniffed or otherwise intercepted or stolen.
Procedure:	<p>For Security Operations: Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.</p> <p>For Development: Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.</p>

Affected Servers:	<ul style="list-style-type: none"> • http://zero.webappsecurity.com:80/stats/ • http://zero.webappsecurity.com:80/login.asp.bak • http://zero.webappsecurity.com:80/stats/Default.asp • http://zero.webappsecurity.com:80/login/login.asp • http://zero.webappsecurity.com:80/login.asp <p>ON</p> <ul style="list-style-type: none"> • 24.28.199.152
-------------------	---



Possible Username or Password Disclosure

Problem:	<p>A username or password was found. Unknown application testing seeks to uncover new vulnerabilities in both custom and commercial software. Because of this, there are no specific patches or descriptions for this issue. Please note that this vulnerability may be a false positive if the page it is flagged on is technical documentation. Recommendations include removing the information from the production server, or otherwise restricting access.</p> <p>This information could allow an attacker to access sensitive applications and information on a site, or to perform functions according to the privilege level of the login information. Gaining information critical to the success of escalated attacks would also be a likely impact of exploitation.</p>
Recommendation:	Remove the information from the web server, if possible, or otherwise restrict access.
Procedure:	Remove the information from the web server, if possible, or otherwise restrict access.
Affected Servers:	<ul style="list-style-type: none"> • http://zero.webappsecurity.com:80/pcomboindex.asp • http://zero.webappsecurity.com:80/pindex.asp.bak • http://zero.webappsecurity.com:80/pformresults.asp • http://zero.webappsecurity.com:80/pcomboindex.asp • http://zero.webappsecurity.com:80/pcomboindex.asp • http://zero.webappsecurity.com:80/pcomboindex.asp • http://zero.webappsecurity.com:80/pcomboindex.asp • http://zero.webappsecurity.com:80/login.asp.BAK • http://zero.webappsecurity.com:80/pcomboindex.asp • http://zero.webappsecurity.com:80/pcomboindex.asp • http://zero.webappsecurity.com:80/pcomboindex.asp • http://zero.webappsecurity.com:80/pcomboindex.asp • http://zero.webappsecurity.com:80/pindex.asp.BAK <p>ON</p> <ul style="list-style-type: none"> • 24.28.199.152

Appendix A – External Scan Reports

Appendix B – Web App Penetration Testing Documents