



Web Application Vulnerabilities Detected and Analyzed by AppScan® 6.0

Application Tests

Application tests are unique to a specific application. Application tests are native to the specific programming and configuration of the application itself, not the underlying technologies. Unlike infrastructure checks that focus on the technological building blocks of the site (web server, application server, database server and CGI scripts), application tests *always* involve *dynamic* data consumed by the application, such as specific application paths, specific application parameter names, specific application parameter values and specific application cookies. An application test exploits a software bug at the business logic layer of a specific application. AppScan® 6.0 identifies application tests by learning the business logic of the application and then creates custom tests that are designed to probe the application's logic for vulnerabilities that a hacker can exploit.

The following matrix lists the various application test types, test techniques, validation rules, an evaluation of the application's responses and the business impact associated with each application test category.

Authorization: Credential/Session Prediction

Name	Description	Test Technique	Validation	Business Impact
Predictable Session Token	Web servers and/or applications use cookies in order to identify their users. Using predictable numeric values for a session cookie makes it possible to hijack another user's session.	Send request with different session identifier values	Determine whether a session was entered that belongs to a different user, or whether the session was terminated	<ul style="list-style-type: none"> • Identity theft • Session hijacking • User impersonation

Authorization: Insufficient Session Expiration

Name	Description	Test Technique	Validation	Business Impact
Session Not Invalidated After Logout	Test whether the session identifier is invalidated by the logout sequence	Perform logout and then request a resource that is only accessible to authenticated users	Compare the test response to the one AppScan received when it was authenticated (during the explore stage)	<ul style="list-style-type: none"> • Identity theft • Session hijacking • User impersonation
Permanent Cookie Contains Sensitive Session Information	Session information that is stored in permanent cookies may be exposed to other users	Extract session identifiers that were found during the explore stage and examine their attributes	Detect that session identifiers do not have the RAM attribute	<ul style="list-style-type: none"> • Identity theft • Session hijacking • User impersonation

Authorization: Session Fixation

Name	Description	Test Technique	Validation	Business Impact
External Session Identifiers Enforcement	In case the server has a "Permissive" behavior when it sets session identifier values, it is possible for an attacker to determine the cookie value for the victim and to use the same value to impersonate that user.	AppScan sends a session identifier via a URL parameter, with a value that will later be used for the validation.	<ul style="list-style-type: none"> • Did the application accept the suggested session identifier? • Did the response contain the session identifier? (search the SET-COOKIE response header) 	<ul style="list-style-type: none"> • Identity theft • Session hijacking • User impersonation

Client-side Attacks: Content Spoofing

Name	Description	Test Technique	Validation	Business Impact
HTTP Response Splitting	An application attack technique that enables web cache poisoning, cross-user defacement, page hijacking and cross-site scripting (XSS). This attack, and the derived attacks, are relevant to most web environments and are the	Attempt to manipulate the response's HTTP structure by using special characters as parameter values	Did the test response contain the new HTTP fragments (headers, new lines, etc.)?	<ul style="list-style-type: none"> • Identity theft • Site defacement (web cache poisoning) • Response hijacking

	result of the application's failure to reject illegal user input (containing malicious or unexpected characters).			
Phishing Through URL Redirection	Phishing is a general term for attempts to scam users into surrendering private information that will be used for identity theft.	When a parameter is suspected to contain a new address that the server is redirected to, it is modified to point to an external site.	AppScan detects whether a redirection to the external site was performed.	An attacker may cause the server to redirect requests to his own site and launch a phishing scam.

Client-side Attacks: Cross-site Scripting

Name	Description	Test Technique	Validation	Business Impact
Cross-Site Scripting	Insert client-side logic into the application's response to a user. Hackers use this page to steal valuable information (usernames, passwords, billing info, cookie data etc.) from unsuspecting site users.	<ul style="list-style-type: none"> • Insert scripts (client-side logic) into URL parameters • Insert scripts into hidden fields • Insert scripts into form fields • Insert scripts into the application URLs 	<ul style="list-style-type: none"> • Did the application accept the script? • If no, what was the response? Customized Error Page, standard 404, or other? • If yes, was the script embedded in the application's response and did it execute the script? 	<ul style="list-style-type: none"> • Identity theft • Session hijacking • User impersonation

Command Execution: Buffer Overflow

Name	Description	Test Technique	Validation	Business Impact
<ul style="list-style-type: none"> • Parameter Value Buffer Overflow • Cookie Buffer Overflow 	Insert values that contain enough characters to overflow the system's memory buffer. This can expose the application & network to a second wave of attacks (e.g., remote	<ul style="list-style-type: none"> • Insert "over-sized" character strings • In parameters in URL • In parameters in hidden fields • In form fields • Insert "over-sized" cookie values 	<ul style="list-style-type: none"> • Did the application accept the character string? • If no, what was the response? Customized Error Page, standard 404, or other? • If yes, did the application fail? • In addition, send the 	<ul style="list-style-type: none"> • Denial of service • Remote command execution on the server

	command execution) with more serious consequences.		original request again to make sure that the application is indeed not responsive.	
--	--	--	--	--

Command Execution: Format String Attack

Name	Description	Test Technique	Validation	Business Impact
Format String Remote Command Execution	The format string issue is a result of using client input as a formatting string input for C/C++'s data formatting functions (e.g. printf)	AppScan injects conversion characters (%f, %p, %n, etc) to 'format string' parameters.	<ul style="list-style-type: none"> • Did the application accept the character string? • If no, what was the response? Customized Error Page, standard 404, or other? • If yes, check if the formatting attempt was successful 	<ul style="list-style-type: none"> • Denial of service • Remote command execution on the server

Command Execution: LDAP Injection

Name	Description	Test Technique	Validation	Business Impact
LDAP Injection	LDAP Injection is an attack technique used to exploit websites that construct LDAP statements from user-supplied input.	Distort LDAP commands used to perform authentication or retrieve user data	<ul style="list-style-type: none"> • AppScan recognizes the LDAP error messages. • Did the application allow the injected characters? • If yes, what error did the application return? 	<ul style="list-style-type: none"> • Identity theft • Complete and unauthorized access to proprietary data • Corporate intelligence gathering

Command Execution: OS Commanding

Name	Description	Test Technique	Validation	Business Impact
<ul style="list-style-type: none"> • System Call Code Injection • Shell Command Injections 	<p>Some scripts use operating system calls to execute commands. Sometimes URL parameters are used as a part of the command.</p>	<p>Insert system commands into URL parameters</p>	<ul style="list-style-type: none"> • Did the application accept the character string? • Look for the command output in the response 	<ul style="list-style-type: none"> • Remote command execution on the server

Command Execution: SQL Injection

Name	Description	Test Technique	Validation	Business Impact
<ul style="list-style-type: none"> • SQL Injection • Login Page SQL Injection • HTTP Referer Header SQL Injection • SOAP Web Services SQL Injection • Cookie Poisoning SQL Injection 	<p>Insert Simple Query Language (SQL) commands into fields resulting in execution of those commands directly in the application's database</p>	<ul style="list-style-type: none"> • Manipulate SQL commands used to perform transactions • Bypass the authentication mechanism: set password parameter to ' OR '1='1 or append ' - - to the username • SQL Injection - MS-SQL xp_cmdshell shell command execution 	<ul style="list-style-type: none"> • AppScan recognizes the error messages of many different backend database systems. • Did the application allow the submission of SQL character(s)? • If no, what was the response? Customized Error Page, standard 404, or other? • If yes, did the application return an error message from the back-end database? 	<ul style="list-style-type: none"> • Identity theft • Complete and unauthorized access to proprietary data • Corporate intelligence gathering • Remote database command & query execution

Blind SQL Injection	Blind SQL Injection - run arbitrary SQL queries on the back-end database server	Sending requests whose vulnerable parameter (the parameter that gets embedded in the SQL query) is modified in such way that the response indicates whether the data is used in SQL query context or not.	The modification involves the use of an AND Boolean expression with the original string, which evaluates once as true and once as false. The result of the first request should be identical to the original result, whereas the result of the second request should be different.	<ul style="list-style-type: none"> • Identity theft • complete and unauthorized access to proprietary data • Corporate intelligence gathering • Remote database command & query execution
SQL Injection Command Execution	SQL Injection - MS-SQL xp_cmdshell shell command execution	Shell command execution through a built-in MS SQL function. It is possible to invoke this function with a shell command argument from a query context.	AppScan validates that the remote command was executed on the MS-SQL server by establishing a remote connection back from the MS-SQL server to the machine running AppScan, using the shell command 'tftp'.	<ul style="list-style-type: none"> • Identity theft • Unauthorized access to proprietary data • Theft of goods and/or services (eShoplifting) • Site defacement
	Oracle SQL Injection.	Run arbitrary SQL queries on the back-end database server.	AppScan validates that the SQL injection succeeded by forcing the Oracle database to establish an HTTP connection back from the Oracle server to the machine running AppScan, using the UTL_HTTP package.	<ul style="list-style-type: none"> • Identity theft • Complete and unauthorized access to proprietary data • Corporate intelligence gathering • Remote database command & query execution

Command Execution: SSI Injection

Name	Description	Test Technique	Validation	Business Impact
Server Side Directives File Retrieval	If server side include is enabled on the web server	AppScan embeds server side include directives such as	AppScan detects the returning	<ul style="list-style-type: none"> • Remote command execution on the server

	and the application embeds un-sanitized user input in the response, attackers may run arbitrary commands on the web server.	#include and #exec in parameter values.	message of the SSI test.	<ul style="list-style-type: none"> • Complete server compromise. Data can be downloaded, changed or deleted. The application can be changed, and the website can be defaced
--	---	---	--------------------------	--

Command Execution: XPath Injection

Name	Description	Test Technique	Validation	Business Impact
XPath Injection	An XPath Injection attack enables an attacker to extract a complete XML document used for XPath querying.	Send XPath query terminating strings as parameter values	Did the application return an XPath error message?	<ul style="list-style-type: none"> • Identity theft • Complete and unauthorized access to proprietary data • Corporate intelligence gathering

Information Disclosure: Directory Indexing

Name	Description	Test Technique	Validation	Business Impact
Forceful Browsing	Learn the directory structure of the specific web application. Force the application to display the contents of each directory on the site.	<ul style="list-style-type: none"> • Remove path name • Change path name • Add path name 	<ul style="list-style-type: none"> • Did the application accept the request for the guessed directory? • If no, what was the response? Customized Error Page, standard 404, or other? • If yes, then was it possible to access files found in the guessed directory? 	<ul style="list-style-type: none"> • Attack planning and intelligence gathering about site structure and security • Intellectual property theft

Information Disclosure: Information Leakage

Name	Description	Test Technique	Validation	Business Impact
Unencrypted __VIEWSTATE Parameter	Validate that the ViewState parameter is encrypted	Identify all scripts that use __VIEWSTATE parameters and validate that values are digitally encrypted	AppScan determines if the __VIEWSTATE value is encrypted according to its value.	Attack planning and intelligence gathering about site structure and security
Application Input Restrictions Bypass	By changing a parameter's value to a value beyond its designated range, an attacker can gain useful information about the behavior of the application, or even access confidential data.	<ul style="list-style-type: none"> • Increase parameter value • Decrease parameter value 	Did the application reach an undefined state? Did the script "break down" due to illegal and unexpected values?	<ul style="list-style-type: none"> • Confidential data exposed • Quality issues may lead to security breaches • Site may appear unprofessional
HTML Comments Sensitive Information Disclosure	Automatically retrieve revealing comments in the HTML source code left behind by application developers and site administrators.	Find all suspicious comments within HTML source code	Present suspicious comments for user review	Attack planning and intelligence gathering about site structure and security
IBM Net.Data Internal Variables Display	Attackers can view the values of internal variables which may disclose sensitive information such as pathnames, server names, usernames and passwords.	Substitutes variables that are embedded in user input	Search the test response for output of the injected variables	<ul style="list-style-type: none"> • Confidential data exposed • Attack planning and intelligence gathering about site structure and security
Application Logic Subversion	If an attacker probes the application by forging a request that toggles an On/Off or True/False value, the application may enter an undefined state that makes it vulnerable to attack.	Toggle parameter values (e.g., True->False, Yes->No, etc.)	Did the application reach an undefined state? Did the script "break down" due to illegal and unexpected values?	<ul style="list-style-type: none"> • Confidential data exposed • Quality issues may lead to security breaches • Site may appear unprofessional

XML External Entity File Disclosure	Attempt to retrieve system files through DTD external entities	Define an "external entity" which value is taken from an external source (e.g., system file)	<ul style="list-style-type: none"> • Did the XML parser accept the request? • If yes, did the parser return the requested system file? 	<ul style="list-style-type: none"> • Unauthorized access to proprietary information • Theft of goods or services (eShoplifting)
-------------------------------------	--	--	--	---

Information Disclosure: Path Traversal

Name	Description	Test Technique	Validation	Business Impact
File Parameter Alteration	CGI scripts often include parameters that specify a file that is displayed or used as a template. If the filename provided to the script is not validated by the application, an attacker could manipulate this parameter and request other files residing on the server.	Use the path traversal technique (set value to <code>..\..\..\filename.ext</code>) to retrieve files outside of the virtual directory	Check the response to see if the requested file was retrieved successfully	Server files become viewable to the attacker on Windows operating systems. These files may include: databases, customer lists and other vital data.
Poison Null Byte Files Retrieval	A user is able to force the script to use any file by adding <code>%00</code> (Null byte) to the end of a desired file.	Use a null byte (<code>%00</code>) to retrieve files outside of the virtual directory, or files that were not meant to be used by the application	Check the response to see if the requested file was retrieved successfully	Server files become viewable to the attacker on Windows operating systems. These files may include: databases, customer lists and other vital data.

Information Disclosure: Predictable Resource Location

Name	Description	Test Technique	Validation	Business Impact
Directory Guessing	Guess the presence of standard directories that can contain important files. These directories typically have names like <code>.../admin/...</code> .	Use extensive list of directory names that are created by default when installing web application technologies to search for unprotected	<ul style="list-style-type: none"> • Did the application accept the request for the guessed directory? • If no, what was the response? Customized Error Page, standard 404, or other? • If yes, then was it possible to access files 	<ul style="list-style-type: none"> • Attack planning and intelligence gathering about site structure and security • Intellectual property theft

		proprietary content and files	found in the guessed directory?	
Temporary File Download	Web servers usually associate Common Gateway Interface (CGI) filename extensions with a handler. Older versions with different extensions (e.g., BAK) may be viewed as is -- revealing source code as the default handler simply returns the file as plain text.	<ul style="list-style-type: none"> • Set file extension to various old extensions (BAK, OLD, SAV, etc.) • Append old extensions to file names 	Check the response to see if the requested file was retrieved successfully with script source code in it	Script source code may be exposed, which may help the attacker to reverse-engineer the web application.

Logical Attacks: Abuse of Functionality

Name	Description	Test Technique	Validation	Business Impact
Insecure Indexing	Search engines typically run on the server and may have access to local files which should not be accessible to remote users. In these cases, it is possible to exploit the accessibility rights these search engines have and read files and documents that were otherwise impossible to access.	Modify a search script input parameters to contain words that are commonly used in certain file types	Check if the search results contain the file name or type	<ul style="list-style-type: none"> • Sensitive files that should not be accessible may be read by a remote user. • Unauthorized access to proprietary data • Corporate intelligence gathering
Application Debug Mode Enforcement	Obtain unauthorized administrative access to the application and site through "backdoors" left open by developers and administrators	Add common debug and backdoor values to scripting URLs	<ul style="list-style-type: none"> • Did the application accept the request with the debug/backdoor option turned on? • If yes, then did AppScan access the application in "debug" mode? 	<ul style="list-style-type: none"> • Attack planning and intelligence gathering about site structure and security • Full control of the web application and site

Unsigned __VIEWSTATE Parameter	Validate that the ViewState parameter is signed.	Identify all scripts that use the __VIEWSTATE parameter and validate that values are digitally signed	AppScan determines if the __VIEWSTATE value is signed according to its value.	Attack planning and intelligence gathering about site structure and security
eShoptlifting	Web-based shopping applications which pass price data as hidden fields in HTML forms are often compromised when attackers submit the form with a reduced price. If the modified price is accepted by the application, an attacker is able to purchase goods or services for a fraction of the price.	AppScan detects parameters that are suspected to contain product prices and sets an alternative price.	Examining the test response reveals whether the new price added by AppScan was accepted by the application	Theft of goods and/or services
Email Parameter Spoofing	If a link contains a complete email address that is used to receive messages generated by that link, the email address can be changed and the server can be used to distribute email messages to third parties.	AppScan detects parameters that are suspected to contain email addresses and sets a different email address in those parameter values.	Examining the test response reveals whether the new email address added by AppScan was accepted by the application	It is possible to send emails through your web application, using spoofed email addresses.

Logical Attacks: Denial of Service

Name	Description	Test Technique	Validation	Business Impact
<ul style="list-style-type: none"> XML Parser DTD Parameter Entities Denial of Service SOAP Array Overflow 	Denial of Service attempts against various XML parsers	<ul style="list-style-type: none"> XML DTD entity blowup XML DTD parameter entities blowup XML parser attribute blowup 	<ul style="list-style-type: none"> Did the server accept the request? If no, what was the response? If yes, did the parser crash? 	<ul style="list-style-type: none"> Denial of service Remote command execution on the server

Logical Attacks: Insufficient Process Validation

Name	Description	Test Technique	Validation	Business Impact
Accessible Protected Resource	An insufficient process validation makes it possible for attackers to access protected resources. If the server does not enforce the access restriction mechanism, unauthorized access becomes possible.	This issue is tested by requesting a protected resource without the proper session identifiers.	The server should reject the request instead of returning the resource content.	<ul style="list-style-type: none"> Unauthorized access to proprietary data Corporate intelligence gathering

Application Privacy Tests

Name	Description	Test Technique	Validation	Business Impact
<ul style="list-style-type: none"> Unencrypted Login Request Unencrypted Password Parameter Unencrypted Sensitive Data 	Tests how the application handles private information, such as passwords, sensitive content (e.g., SSN, credit card numbers)	<ul style="list-style-type: none"> Identify insecure login requests Identify scripts that pass sensitive user information in an insecure way 	AppScan determines if sensitive user information has passed unencrypted.	<ul style="list-style-type: none"> Identity theft Sensitive user information gathering
Missing Secure Attribute in Encrypted Session (SSL) Cookie	Test cookie security and privacy issues	Identify secure cookies (cookies with the secure-bit turned on) which were not passed over SSL encryption	AppScan determines which cookies are secure cookies (have the 'secure' attribute), but are not passed over SSL	<ul style="list-style-type: none"> Identity theft Session hijacking User impersonation

Application Quality Tests

Name	Description	Test Technique	Validation	Business Impact
Application Error	Security tests that may not necessarily be considered application vulnerabilities but could still indicate susceptibility to manipulation or misuse.	<ul style="list-style-type: none"> Remove parameter value Increase parameter value Decrease parameter value Insert hazardous character(s) Append new values to existing ones Toggle parameter values (e.g. True->False, Yes->No, etc.) 	Did the application reach an undefined state? Did the script "break down" due to illegal and unexpected values?	<ul style="list-style-type: none"> Quality issues may lead to security breaches Site may appear unprofessional

Infrastructure Tests

Infrastructure tests are vulnerabilities that are found in any of the CGI scripts, web servers, application servers, or database servers used to build web applications. In general, an infrastructure test is an unintended consequence of either flawed design or development of the web application technology. It can also be the product of a misconfiguration of the web application technology once it is installed as a part of a website. When technologies containing infrastructure tests are used in web applications, the web application is, by extension, vulnerable to attack from the web (usually through Ports 80 and 443).

AppScan contains all current infrastructure tests that are associated with Common Vulnerabilities and Exposures (CVEs) and published in CERT and Security Focus. These tests are updated frequently – once a week on average. Following is a summary of the types of infrastructure test checks performed by AppScan:

Infrastructure Test Type	% of Total AppScan Infrastructure Tests
Infrastructure tests	~90%
Smart infrastructure tests*	~10%
Total infrastructure tests	Over 1,800

* *Smart Infrastructure Tests* are unique to AppScan and improve the reliability of results. In contrast to the one-step approach of other types of infrastructure test scanners (CGI, Network, and Database), AppScan takes a two-step approach to scanning for CGI scripts known to contain an infrastructure vulnerability. As a result, it is much more likely to find CGIs that contain infrastructure vulnerabilities no matter where they are installed within a site. First, AppScan explores the site and learns its structure. In the process, it finds all directories that

contain scripts. Based on this information, AppScan creates custom requests for the scripts known to contain an infrastructure vulnerability based within every scripting directory on the site. This is referred to as “Smart Infrastructure Tests.”

Sample of Major Vendors, Platforms, and Technologies Checked for Infrastructure vulnerabilities by AppScan 6.0*

- Apache
- BEA WebLogic
- Extropia
- IBM WebSphere
- IBM Net.data
- Jakarta Tomcat
- Java Servlets
- Lotus Domino
- Macromedia ColdFusion
- Macromedia Jrun
- Microsoft FrontPage
- Microsoft IIS
- Microsoft Index Server
- Microsoft Personal Web Server
- Microsoft Site Server
- Microsoft Exchange (OWA)
- Microsoft .NET
- MySQL
- Netscape Enterprise Server
- Novell
- OpenSSL
- Oracle Application Server
- Perl
- PHP
- SAP
- SalesLogix
- Sun - iPlanet
- Sun - Java Hotspot
- Sun One
- Sun Cobalt
- Unix
- Vignette products
- Visnetic Website
- Zeus
- Zope

* A list of Third-Party Misconfiguration and infrastructure tests for specific vendor, platform or technology is available upon request.

About Watchfire

Watchfire provides software and services to manage online risk. More than 300 enterprise organizations and government agencies, including AXA Financial, SunTrust, Boots PLC, Health and Human Services, Social Security Administration and Dell rely on Watchfire to monitor, manage, improve and secure all aspects of the online business including security, privacy, quality, accessibility, corporate standards and regulatory compliance. Watchfire’s alliance and technology partners include IBM Global Services, PricewaterhouseCoopers, TRUSTe, Microsoft, Interwoven, EMC Documentum and Mercury. Watchfire is headquartered in Waltham, MA. For more information, please visit www.watchfire.com.

Watchfire, WebXM, AppScan, PowerTools, the Bobby Logo and the Flame Logo are trademarks or registered trademarks of Watchfire Corporation. All other products, company names, and logos are trademarks or registered trademarks of their respective owners.

References

1. Credential/Session Prediction

http://www.webappsec.org/projects/threat/classes/credential_session_prediction.shtml

2. Insufficient Session Expiration

http://www.webappsec.org/projects/threat/classes/insufficient_session_expiration.shtml

3. Session Fixation

http://www.webappsec.org/projects/threat/classes/session_fixation.shtml

4. Content Spoofing

http://www.webappsec.org/projects/threat/classes/content_spoofing.shtml

5. Cross-site Scripting

http://www.webappsec.org/projects/threat/classes/cross-site_scripting.shtml

6. Buffer Overflow

http://www.webappsec.org/projects/threat/classes/buffer_overflow.shtml

7. Format String Attack

http://www.webappsec.org/projects/threat/classes/format_string_attack.shtml

8. LDAP Injection

http://www.webappsec.org/projects/threat/classes/ldap_injection.shtml

9. OS Commanding

http://www.webappsec.org/projects/threat/classes/os_commanding.shtml

10. SQL Injection

http://www.webappsec.org/projects/threat/classes/sql_injection.shtml

11. SSI Injection

http://www.webappsec.org/projects/threat/classes/ssi_injection.shtml

12. XPath Injection

http://www.webappsec.org/projects/threat/classes/xpath_injection.shtml

13. Directory Indexing

http://www.webappsec.org/projects/threat/classes/directory_indexing.shtml

14. Information Leakage

http://www.webappsec.org/projects/threat/classes/information_leakage.shtml

15. Path Traversal

http://www.webappsec.org/projects/threat/classes/path_traversal.shtml

16. Predictable Resource Location

http://www.webappsec.org/projects/threat/classes/predictable_resource_location.shtml

17. Abuse of Functionality

http://www.webappsec.org/projects/threat/classes/abuse_of_functionality.shtml

18. Denial of Service

http://www.webappsec.org/projects/threat/classes/denial_of_service.shtml

19. Insufficient Process Validation

http://www.webappsec.org/projects/threat/classes/insufficient_process_validation.shtml