



WAVSEP Scan Log



Scanner	Scan Description
N-Stalker 2009 Free Edition	<p>I initiated the scan with the “Cross Site Scripting Only” policy, activated the optimize button, optimized the scan for J2EE, defined the spider “max per node” feature as 64 and initiated the scan on the following URLs:</p> <p>http://localhost:8080/wavsep/RXSS-Detection-Evaluation-GET/index.jsp http://localhost:8080/wavsep/RXSS-Detection-Evaluation-POST/index.jsp http://localhost:8080/wavsep/index-false.jsp</p> <p>The scanner successfully crawled all the URLs.</p>
Priamos	<p>I executed the tool’s search injection feature on the following URLs:</p> <p>http://192.168.110.1:8080/wavsep/index-sql.jsp http://192.168.110.1:8080/wavsep/index-false.jsp</p> <p>The tool crawled nearly all the URLs, but did not find a single vulnerability, regardless of what I attempted.</p>
LoverBoy	<p>I configured an http proxy (to see if it is working), increased the timeout for exploitation to 60 seconds and for crawling to 35 minutes, checked the “scan for vulnerabilities” checkbox, checked the logging checkbox and chose a name for the log file, checked the “include MySQL” in the included vulnerability scanners. I also tried the same settings without proxy configuration, and on several initial URLs:</p> <p>http://192.168.110.1:8080/wavsep/index-sql.jsp http://192.168.110.1:8080/wavsep/index-false.jsp</p> <p>I even transferred the content of index-sql.jsp and index-false.jsp into the index.jsp file, and scan from the initial point of access:</p> <p>http://192.168.110.1:8080/wavsep/ http://192.168.110.1:8080/wavsep</p> <p>The tool failed the crawling process over and over. For some reason, it only managed to begin scanning the default pages of tomcat, but still failed crawling through my pages (with or without proxy).</p>
crawlfish	<p>I disabled the “restrict to folder” checkbox (the crawling process only seemed to succeed with this feature disabled), increased the max crawl cap to 100, and scanned the following URLs:</p> <p>http://192.168.110.1:8080/wavsep/index-xss.jsp http://192.168.110.1:8080/wavsep/index-false.jsp</p> <p>Even though the scanner always crashed at the end of the scan, I was still able to see the scan results with the error window in the background.</p> <p>The tool successfully crawled all URLs, did not detect any POST vulnerabilities (probably because it didn’t submit any forms).</p>
openAcunetix	<p>The tool did not crawl or locate any exposures, regardless of the URLs or parameters defined.</p>
PowerFuzzer	<p>The tool didn’t manage to crawl the xss/false/sql index URLs, so I solved the problem by copying the content from each one of them into the index.jsp file, and executed the scan in front of the root directory:</p> <p>http://192.168.46.2:8080/wavsep</p>
Web Injection Scanner (WIS)	<p>When I attempted to use this tool against a vulnerable .Net application, I only managed to cause it to crawl using the following command on a full URL (no vulnerabilities were identified):</p> <p>wis http://192.168.52.129/asp/asp/login.aspx</p> <p>In the current benchmark the tool seemed to work, but failed to crawl the application or locate any exposures. I tried a couple more options, but still got the same results. Among the options attempted:</p> <p>wis http://192.168.46.2:8080/wavsep/index-sql.jsp wis http://192.168.46.2:8080/wavsep/index-false.jsp wis http://192.168.46.2:8080/wavsep/ (after copying the content of index-sql.jsp into index.jsp) wis http://192.168.46.2:8080/wavsep wis “http://192.168.46.2:8080/wavsep/” wis “http://192.168.46.2:8080/wavsep/index-sql.jsp” wis http://192.168.46.2:8080/wavsep/SInjection-Detection-Evaluation-GET-500Error/Case1-InjectionInLogin-String-LoginBypass-WithErrors.jsp?username=textvalue&password=textvalue2 wis “http://192.168.46.2:8080/wavsep/SInjection-Detection-Evaluation-GET-500Error/Case1-InjectionInLogin-String-LoginBypass-WithErrors.jsp?username=textvalue&password=textvalue2”</p>

Scanner	Scan Description
Mini MySqlatOr	<p>I attempted to start crawling the index-sql.jsp page, and found out that the tool was not able to crawl the application in this manner, so I copied the content of the files index-sql.jsp & index-false.jsp to index.jsp, and scanned the folder root URL: <code>http://192.168.46.2:8080/wavsep/</code> (The tool successfully crawled all the pages). I then selected all the parameters in the "SQL Injection Finder" tab, and initiated the scan.</p>
iScan	<p>The tool was not able to scan non-standard ports for some reason (8080), so I defined the web site as an upstream proxy in burp (192.168.46.2:8080) and burp as an outgoing proxy in burp. The following URLs were scanned using iScan (through burp): <code>http://localhost/wavsep/index-sql.jsp</code> <code>http://localhost/wavsep/index-xss.jsp</code> <code>http://localhost/wavsep/index-false.jsp</code> The scanner was not able to scan the application, and always declared that the URL scanned was not found, so I investigated further and found the problem: it seems that the scanner did not support any upper cases URLs, and in fact, sent URLs only in lower case format, a behavior that caused my tomcat server to respond with 404 messages. As a result, I have decided that in its current condition, this tool could not be used to test a wide range of technologies, and thus, I will postpone its assessment to future benchmarks.</p>
Gamja	<p>The tool does not support any configurable options. I used the following commands to execute the scanner: <code>perl gamja.pl http://192.168.46.2:8080/wavsep/index-xss.jsp</code> <code>perl gamja.pl http://192.168.46.2:8080/wavsep/index-sql.jsp</code> <code>perl gamja.pl http://192.168.46.2:8080/wavsep/index-false.jsp</code></p>
ScreamingCSS	<p>I placed the "wget" file distributed with the scanner "Gamja" in the same directory of the tool. I executed the scan using the following commands: <code>perl screamingCSS.pl -e -i -v http://192.168.46.2:8080/wavsep/index-xss.jsp >> SCSS_wavsepXSS_Scan_report.txt</code> <code>perl screamingCSS.pl -e -i -v http://192.168.46.2:8080/wavsep/index-false.jsp >> SCSS_wavsepFalse_Scan_report.txt</code> The tool managed to crawl all the application URLs.</p>
Grabber	<p>The tool can be configured using a configuration file or using command line arguments. Initially I tried using the configuration file (pretty simple), but I also executed the tool with the following command: <code>grabber --spider 1 --sql --bsql --backup --include --javascript --session --xss --url http://192.168.46.2:8080/wavsep/index-xss.jsp >> scanlog_WavSepXSS.log</code> The tool ignored direct references to index URLs, so I copied the content of index-xss.jsp, index-sql.jsp and index-false.jsp into the index.jsp page (one at a time), and executed the scan using the following command: <code>grabber --spider 1 --sql --bsql --xss --url http://192.168.46.2:8080/wavsep</code> For some bizarre reason, even though the tool successfully crawled all URLs, it did not scan any GET parameters, and instead, only scanned forms (POST parameters). Since all the false positive tests were implemented as GET URLs, I had to create special index pages, so that grabber will be able to scan the false positive cases.</p>
VulnDetector	<p>After trying to use the tool a couple of times I realized it will not scan any non standard ports, so I set up burp to listen to localhost port 80, and then defined port forwarding in burp for the actual target server (192.168.46.2:8080). After verifying that the port forwarding works fine, I altered all the "asp" extensions and texts in the source code to "jsp", copied the content of the different index pages to index.jsp, edited the python code and defined <code>http://localhost/wavsep/</code> as the tested site (and "localhost" as the domain), made sure the XSS and SQL flags were true, set the check level to 3, manually created the log directory and files (the tool fails to execute if the files do not exist) and started the scan using the command <code>"c:\Python25\python.exe vulndetector-0.0.2pa.py"</code>. When the scan didn't work I tried executing the same scan with python 2.4, and/or using different configurations & target URLs, but with no success. It is important to mention that the tool did manage to scan different simple sites, but for some reason, was not able to scan the current test site.</p>
SQID (SQL Injection Digger)	<p>I initiated a scan in front of the index pages with the following commands (burp proxy was used to verify the tool is working properly): <code>sqid -m crawl --crawl http://192.168.46.2:8080/wavsep/index-sql.jsp -a -v -D sqid.db -P localhost:9999</code> <code>sqid -m crawl --crawl http://192.168.46.2:8080/wavsep/index-false.jsp -a -v -D sqid.db -P localhost:9999</code> The tool crawled some of the public URLs, but found nothing. A further examination (BURP) showed that the tool did not perform ANY tests on the pages found. Scanning through a URL file caused the tool to access the various URLs (while ignoring the proxy definitions), but again, no exposures were found (assuming they were actually tested).</p>
ZAP	<p>I enabled all the injection plugins in ZAP and executed the tool in-front each individual directory.</p>

Scanner	Scan Description
Paros Proxy	<p>I enabled all the injection plugins in Paros and executed the tool in-front of the following URLs: http://localhost:8080/wavsep/index-xss.jsp http://localhost:8080/wavsep/index-sql.jsp http://localhost:8080/wavsep/index-false.jsp</p> <p>Performing additional scans in the same session results in the discovery of additional vulnerabilities, particularly in SQL-200-Valid (case 1(2nd), case 4). It is currently unclear whether the reason for the inconsistency is the scanner or the behavior of the vulnerable pages.</p>
WebScarab	<p>I used the spider feature of WebScarab to crawl the various application pages, accessed the XSS/CRLF tab, marked all the URLs (CTRL+A) and pressed the "check" button, so the tool will try and confirm vulnerabilities. Only URLs with GET parameters were suspected as being vulnerable (I verified that the behavior persists with one additional version of WebScarab - 20090427).</p>
Uber Web Security Scanner	<p>The initial pages were configured in the scan configuration file - uwss.conf (which was surprisingly easy to use), and the scan was executed using the following commands: <pre>python uwss -f uwss.conf -c python uwss -f uwss.conf -r xml</pre> The following URLs were scanned: http://192.168.1.100:8080/wavsep/index-xss.jsp http://192.168.1.100:8080/wavsep/index-sql.jsp http://192.168.1.100:8080/wavsep/index-false.jsp</p> <p>The tool successfully crawled all the application pages (although I'm not sure forms were submitted during this process, or even if the tool is capable of submitting forms).</p>
Wapiti	<p>I executed the scan multiple times without the backup, nikto and htaccess options: <pre>python wapiti.py http://192.168.1.100:8080/wavsep/index-xss.jsp --scope domain --module "all-nikto-backup-htaccess" --underline --verbose 2 --reportType html --output reportsWavsepXss</pre> The results provided consist of the exposures detected in all the individual scans, which were performed in front of the following URLs: http://192.168.1.100:8080/wavsep/index-xss.jsp http://192.168.1.100:8080/wavsep/index-false.jsp http://192.168.1.100:8080/wavsep/index-sql.jsp</p> <p>The tool used a lot of time-based injection detection payloads which obviously fulfilled their purpose in detecting vulnerable locations; it is however, important to mention that time-based detection that does include syntax verification might cause plenty of false positives on slow systems (and thus, the scanner should verify the successful time-based injections at least twice, and in addition, retransmit the syntax that caused the delay with characters that will make it invalid, in order to verify that the delay is actually caused by that syntax).</p>
SQLiX	<p>The tool wasn't able to scan the application when directed to a specific file name, so I had to copy the content of the internal index page (index-sql, index-false) to the index.jsp file (it solves the problem) and initiate the scan in front of the application's root directory. The scan was eventually executed with the following command: <pre>Perl SQLiX.pl -crawl http://192.168.1.100:8080/wavsep/ -all -v=2</pre> The tool didn't manage to scan the false positive case "7", and crashed every time it did (so I simply removed the link from the directory index page before completing the scan).</p>
WSTool	<p>I manually configured the ws_init.php (by editing it), enabled almost all the optional features (except "exception URL" and 2XX, 3XX, 4XX errors), increased the check limit to 600, and finally, initiated the scan using the following commands: <pre>php ws_main.php 192.168.1.100 8080 GET /wavsep/index-xss.jsp >> reportXSS.html php ws_main.php 192.168.1.100 8080 GET /wavsep/index-sql.jsp >> reportSQL.html php ws_main.php 192.168.1.100 8080 GET /wavsep/index-false.jsp >> reportFalse.html</pre> The scanner successfully crawled all URLs.</p>

Scanner	Scan Description
arachni	<p>According to the recommendations in the website, I used ruby 1.9.2, and installed the application using the ruby gems mechanism.</p> <p>I initially tested version 0.2.4, using the web interface, and the following configuration:</p> <p>Configuration changes: Disabled Audit-Cookies, Reduced Concurrent HTTP Connections to 5 (from 20)</p> <p>*Plugins enabled when testing RXSS/False RXSS exposures:*</p> <p>XSS, XSSPath, ResponseSplitting, UnvalidatedRedirect, XSS in HTML element event attribute, XSS in HTML "script" tag, XSS in HTML tag, XSSURI</p> <p>*Plugins enabled when testing SQLi/False SQLi exposures:*</p> <p>Code injection (timing), Blind (rDiff) SQL Injection , SQLInjection, Code injection , Blind (timing) SQL injection, OS command injection , XPathInjection , OS command injection (timing)</p> <p>In addition, I tried scanning the individual directories using the dedicated plugins (SQLi/XSS).</p> <p>During the scans, I found several minor bugs that caused the blind SQLi detection accuracy of the tool to be inconsistent, and contacted Tasos Laskus (the author) to notify him on the bugs.</p> <p>He quickly started mitigating the issues, and after a few iterations (and a few sleepless nights), sent me the new release, arachni-0.3, which had much better accuracy... as the final results demonstrate (The RXSS results are nearly identical in both versions – 99% vs 100%).</p> <p>When using the upgraded version, I used the following command-line commands to scan the individual directories:</p> <pre>arachni http://[ip]:[port]/wavsep/[xss-directory]/index.jsp -m xss* -gp arachni http://[ip]:[port]/wavsep/[sqli-directory]/index.jsp -m sqli* --http-req-limit=1 -gp</pre> <p>Note that the verifications performed with the updated version of arachni (0.3) improved the results of the SQL Injection detection in 20 test cases (significant), but where executed using a single thread, to a void inconsistency in the time based detection module, at least, as much as possible.</p>
Xcobra	<p>The lack of documentation made it pretty difficult to locate the dependencies for this tool. I tested it on a Fedora VM with python 2.6.x & PyGTK (and additional libraries). The GUI loaded successfully, but each scan performed with the tool ended immediately without any findings, even though I enabled all the plug-ins. I tried to execute the scanner in an additional couple of ways:</p> <p>First, I tried to refer the GUI to the public root URL; it did not seem to crawl the application, so I tried the same with the login page; still, no results. Finally I tried executing the scanner (via the GUI) by loading a target file containing the URLs, but even though the URLs were all loaded successfully, no vulnerabilities were found.</p> <p>Similar scans were performed using the command line version of the tool, without any changes in the results (apart from exceptions and notifications).</p> <pre>python xcobra.py -i WavsepScan.txt -d 7 -v</pre> <p>The tool did not find vulnerabilities (it seemed to be working, since it did not throw exceptions).</p>
XSSS	<p>I installed the dependencies, gave the perl script execution permission, and then executed the tool in front of the root URL of the public section of the application, while enabling the form scanning feature (the public section should be completely covered by the default crawling depth of the tool, which is 5, so no additional configuration changes were made). I used the following execution commands:</p> <pre>./xss --forms http://192.168.1.101:8080/wavsep/index-xss.jsp ./xss --forms http://192.168.1.101:8080/wavsep/index-false.jsp</pre> <p>The tool did not manage to crawl the application pages, but I eventually managed to scan the directories using the following commands:</p> <pre>./xss http://192.168.1.101:8080/wavsep/RXSS-Detection-Evaluation-GET/ ./xss --forms http://192.168.1.101:8080/wavsep/RXSS-Detection-Evaluation-POST/ ./xss http://192.168.1.101:8080/wavsep/RXSS-FalsePositives-GET/</pre>

Scanner	Scan Description
Secubat	<p>After a long and weary installation & configuration process of MSSQL 2005 Express, MSSQL management studio and the tool itself, I started the tool and configured the following values: Max Runtime 10:10:00, Max Pages/Domain 500, Max Depth 5, This domain only – enabled, Enable attacking – enabled, Attack this crawling run – enabled, All Plugins Enabled (3 XSS, 1 SQL)</p> <p>The crawling and scanning process was executed in front of <code>http://localhost:8080/wavsep/index-xss.jsp</code>, but the crawling process did not succeed when executed on the main index or the root directory (even after the <code>index.jsp</code> page was populated with links), so I tried referring the scanner to the individual root of each vulnerable directory.</p> <p>When that didn't work, I crawled the application first (and verified that the process succeeded by viewing the report), and then executed the scan on the appropriate crawling results.</p> <p>The tool successfully crawled all the URLs (eventually, while requiring me to scan each directory independently); however, the tool got stuck in every scan, and required me to restart upon each failure, but one of the errors gave me a lead to the reason – it seems that the attack plugins ignored the 8080 port the application resided on and accessed port 80 instead, so I used burp to channel the communication from port 80 to port 8080, and re-initiated the crawl and the scan on the different directories, as if they resided on port 80. (<code>http://localhost:8080/wavsep/RXSS-Detection-Evaluation-GET</code>, etc)</p> <p>The Max Pages/Domain was automatically reduced to 99, and in addition, regardless of what I did, the scanning phase always stopped without performing any attacks (only crawling requests appeared in burp log, without any attack patterns). I verified that the MSSQL database used by the tool was populated with crawling information (and it has been), tried executing it in front of different URLs, including index pages and folders, attempted it with and without burp's port 80 forwarding, but to no avail.</p> <p>Eventually, I decided to try an old copy I had, and to scan the POST pages first, and finally, I was rewarded for my efforts, and the tool finally worked! (Same configuration, but the maximum test time was set to 30 minutes and the max pages to 80). The tool was eventually used to scan all the POST URLs in the project (all the GET tests failed): <code>http://localhost/wavsep/RXSS-Detection-Evaluation-POST/index.jsp</code> <code>http://localhost/wavsep/SInjection-Detection-Evaluation-POST-500Error/index.jsp</code> <code>http://localhost/wavsep/SInjection-Detection-Evaluation-POST-200Error/index.jsp</code> <code>http://localhost/wavsep/SInjection-Detection-Evaluation-POST-200Valid/index.jsp</code> <code>http://localhost/wavsep/SInjection-Detection-Evaluation-POST-200Identical/index.jsp</code></p> <p>In addition, I altered my false positive test cases so they'll support POST as well, and executed the scanner against them.</p>
sqlmap	<p>I crawled the application's SQL vulnerable pages and false positive pages through burp, while creating a separate burp-formatted log file for each individual directory. I supplied the burp-generated log files to sqlmap as the target source, and used sqlmap's config file to configure the following scan variables:</p> <p>Changes made to the configuration file of sqlmap (<code>sqlmap.conf</code>): <code>logFile</code> (list in the stable version) = <code>BurpLogs\Burp-WAVSEP-[log-name].log</code>, <code>os</code> = windows, <code>level</code> = 5, <code>risk</code> = 3, <code>tech</code> = BEUST (default), <code>extensiveFp</code> = True, <code>getBanner</code> = True, <code>forms</code> = True, <code>freshQueries</code> = True</p> <p>On occasion, I also played around with the following definitions: "<code>batch = True</code>" (to make the tool behave more like a traditional scanner), "<code>flushSession = True</code>", "<code>dropSetCookie = True</code>" and "<code>dbms = mysql</code>" (in <code>sqlmap.conf</code>).</p> <p>The scans were executed against each individual directory, using the following command: <code>C:\Python26\python.exe sqlmap.py -c sqlmap.conf</code></p> <p>The stable version (0.9) tool did very well when used to scan GET variables, but seemed to ignore POST variables (even when the form parsing feature was active), so I updated my current version to the latest version in the SVN (revision 4209). The latest revision required a few more configuration changes (since several sections were a bit different), but managed to scan POST variables as well.</p> <p>The tool scanned all the URLs, performed initial tests on all GET/POST/cookie/user-agent values, and detected most of the vulnerable test cases (!), with nearly zero false positives (only detected one false positive test case, even though it was only vulnerable to XSS, and did not contain any SQL code); however, it only detected certain test cases (4,7,8) if mysql was predefined as the dbms, and testing headers caused the time-based detection modules to affect the detection results of other testcases (only in the SVN version, probably due to connection pool/tomcat resource consumption).</p>
aidSQL	<p>In order to test aidsq, I used a backtrack 5 station with php5.3.2 installed, made sure that php5-cli was installed and installed php5-curl myself.</p> <p>I downloaded the latest stable release (02062011), edited the "aidsq.conf" file, made sure that jsp is in the list of interesting extensions, changed the "immediate-mode" definition to "no", and left the rest of the defaults without change.</p> <p>I executed aidsq against each individual directory using the following command: <code>./aidsq --url=http://IP:port/wavsep/[directory-name]/index.jsp</code></p> <p>The tool crawled all the URLs, seemed to scan them properly, and immediately exploited vulnerabilities that were found.</p> <p>I also tried testing the latest SVN version (r255, updated in May 2011), but had errors similar to the errors I got in the first benchmark, and since it was a few days before the current benchmark release, I didn't have time to find a solution for the issue.</p>

Scanner	Scan Description
XSSploit	<p>I tried installing the scanner in multiple operating systems (ubuntu 9, fedora 11, windows xp), but for some reason, whenever I started executing it on an application, the spider would not work (http://192.168.1.100:8080/wavsep/index-xss.jsp). At the last deadline, a few hours before releasing the article, I decided to try again. I uninstalled every python installation an module from my windows XP vm, deleted any trace of the word “python” from the registry, and installed python 2.5, pyOpenSSL, wxPython and pyCurl (the site claims that only python 2.5 and wxPython are required, but I got carried out). In addition, I defined burp proxy to forward traffic from port 80 to port 8080, and crawled & scanned the following URLs:</p> <p>http://localhost/wavsep/index-xss.jsp http://localhost/wavsep/RXSS-Detection-Evaluation-GET/index.jsp http://localhost/wavsep/RXSS-Detection-Evaluation-POST/index.jsp http://localhost/wavsep/RXSS-FalsePositives-GET/index.jsp http://localhost/wavsep/index-false.jsp</p> <p>This time, the spider feature that did work, and so did the scanner... So it seems like being stubborn has some advantages after all...</p> <p>The scanner successfully crawled all URLs and submitted all the forms, but when I started analyzing, various exceptions were presented in the tool’s console (particularly on page 32), and the scan was never truly “completed” (except in the false positive URLs); it was however, possible to view the results that were already discovered by the tool.</p> <p>I repeated the process with various URLs, with identical results and problems.</p>
Burp Suite Professional	<p>I defined Firefox to use burp as a proxy, accessed the index-xss.jsp, index-sql.jsp and index-false.jsp index files, used burp to crawl them all and verified that all the URLs were successfully crawled.</p> <p>I then verified that all the scan plugins were enabled, and executed the scan on the entire domain (localhost).</p> <p>The tool did not manage to scan all the URLs properly in the first run, so I tried scanning only the SQL vulnerable URLs;</p> <p>The results were inconsistent, probably due to WAVSEP’s lack of database connection pool.</p> <p>I disabled all the attack entry points except for GET & POST parameters (cookie, header, etc), and reduced the amount of threads from 10 to 5, disabled the “intelligent attack selection feature”, made sure that multiple items are not removed, and rescanned the URLs, and eventually, made sure that there won’t be any database connection failures by scanning each directory individually, and that method proved as a more reliable source of information for the purpose of the evaluation.</p> <p>Note:</p> <p>It is important to mention that the inconsistency did not occur in the case of reflected XSS exposures, and was limited to SQL exposures, and thus, is related to the lack of connection pool in WAVSEP, and NOT to the tool itself.</p>
Cenzic Hailstorm Professional	<p>After testing the tool with a few policies, I consulted with the relevant personal at cenzic, and they recommended the OWASP 2010 policy for the purpose of the test.</p> <p>I initially attempted to scan the main index URLs, but after I started receiving inconsistent results and a few scans were abruptly stopped (while only having WAVSEP’s lack of connection pool to blame), I decided to try the safe method and separately scan each individual directory, using the OWASP 2010 policy (with a URL limit set to 500).</p> <p>In order to verify the consistency of the results, I attempted to scan each directory multiple times and compare the results.</p> <p>The tool did a pretty decent job in detecting cross site scripting exposures (although some of the exploits provided by the tool did not work for the given vulnerable case), but I had a debate on how to interoperate his SQL injection results.</p> <p>The tool provides 3 categories for potential SQL Injection exposures: Blind, Disclosure and Error (the latter being a “lead” more than a verified exposure);</p> <p>Many SQL Injection exposures detected by the tool were classified under the SQL Error term (plugin), so I couldn’t out rule the results, even though they weren’t “verified”, however, it also classified several false positive test cases under this term... so I had a debate on whether I should count these cases as false positive cases or not.</p> <p>I eventually decided to take the “merciful” approach and treat these cases as a valid behavior, due to the classification of “SQL Error” (which is correct, but does not pinpoint on a vulnerability), due to the wide usage of this plugin (P.S – Task: limit the usage of SQL Exceptions in false positive test cases in the future versions of WAVSEP... it complicates things :/)</p> <p>Note:</p> <p>The tool did not contain a plugin for Time-Delay based SQL injection detection (verified with the relevant personal at cenzic, at the date of the test)</p>
WebInspect	<p>I tested WebInspect version 9.10.78.0, Engine Version 4.05.99, and updated the plugins on 28/07/2011 (I also tested the version the previous 9.0.351.1, SecureBase 4.04.00, on 27/04/2011, and the results were identical).</p> <p>In order to minimize the risk of database connection failures, I decided to scan each directory individually.</p> <p>I initially started with the “standard” policy, which provided incomplete results, and missed a lot of occurrences;</p> <p>I decided not to take any risks, and used the “All Checks” policy, while validating each scan twice, and scanning again with either the “SQL Injection” policy or the “Cross Site Scripting” policy (depending on the scanned directory), while limiting the scans to the currently scanned directory and choosing a “thorough” coverage method.</p> <p>The results were consistent in all scans.</p> <p>The test went pretty smooth, but the tool did crash on several occasions (always when initiating a new scan, after several scans were already performed).</p>

Scanner	Scan Description
Sandcat Free Edition	<p>I initially tried scanning with all the plugins enabled, but Sandcat appeared to be stuck on various brute force tests (there was no progress for a long period of time, and the "skip check" feature didn't work.). I tried to disable the server checks first, and then a couple of more, but again, the scanner got stuck (for over 30 minutes, until I eventually gave up), and unlike Sandcat version 3.7, I didn't get any message that stated that the delay was intentional. The same behavior repeated itself in every version of the tool I tried -3.9.3, 3.9.4, 3.9.4.5, 4.0rc1 and 4.0.0.1 (excluding 3.7, which works fine in spite of the delays, but is obsolete).</p> <p>I eventually decided to enable only a handful of relevant features before each test.</p> <p>The SQL Injection vulnerable pages were scanned with the following features enabled: database disclosure, command execution, SQL Injection, Xpath Injection, miscellaneous (http://localhost:8080/wavsep/index-sql.jsp); the crawling process detected all the pages, and the scan completed successfully.</p> <p>The SQL Injection false positive pages were scanned with a similar configuration, and the tool's behavior was similar as well (http://guardian:8080/wavsep/SInjection-FalsePositives-GET/index.jsp).</p> <p>The RXSS vulnerable pages were scanned with the following features enabled (both individually and altogether): CRLF Injection, Cross Frame Scripting, Cross Site Scripting, Directory Traversal, File Inclusion.</p> <p>The GET RXSS, POST RXSS and false positive RXSS were individually scanned, and the scanning process seemed to work properly in terms of detection, but always got stuck near the end of the RXSS GET scans (after discovering all the possible vulnerabilities). The crawling processes always managed to detect all the pages. The following URLs were used in the XSS scan:</p> <p>http://guardian:8080/wavsep/RXSS-Detection-Evaluation-GET/index.jsp http://guardian:8080/wavsep/RXSS-Detection-Evaluation-POST/index.jsp http://guardian:8080/wavsep/RXSS-FalsePositives-GET/index.jsp</p>
XSSer	<p>Although the tool was very easy to install and use (a GUI with a wizard, a huge improvement compared to the previous version user experience), I was having difficulties interpreting the results, but eventually found the section that lists the actual vulnerabilities.</p> <p>I used the built in wizard to scan the relevant directories, and scanned each directory multiple times, with several configurations, to ensure the result consistency.</p> <p>The tool was only able to crawl and scan the GET XSS directory, but that was a known limitation.</p> <p>During the scan, the tool sometimes ignored the fact that I did not select the launch option, and launched firefox instances anyway (probably due to the "automatic" definition in the configuration), and furthermore, the crawling feature did not generate consistent results, and constantly failed...</p> <p>The overall results are actually unified results of the test cases that the tool was able to find, whenever it was able to crawl the tested URLs.</p>
ProxyStrike	<p>I initially crawled the main XSS url (http://localhost:8080/wavsep/index-xss.jsp), after enabling all the scan plugins, increasing the scan threads to 7 (for each plugin), and enabling the "crawl using plugins" feature, but for some reason, the tool only crawled the POST URLs, and when I performed another crawling process on the GET directory, I only discovered that the bug persists.</p> <p>I eventually used Paros as an external spider the uses ProxyStrike as an outgoing proxy, and that solved my problem.</p> <p>Overall, the following URLs were crawled and scanned:</p> <p>http://localhost:8080/wavsep/index-xss.jsp http://localhost:8080/wavsep/index-sql.jsp http://localhost:8080/wavsep/RXSS-FalsePositives-GET/index.jsp http://localhost:8080/wavsep/SInjection-FalsePositives-GET/index.jsp</p> <p>The SQL injection included injection classification and database identification.</p> <p>The XSS results did not include any proof of concept, but the XML report generated stated which special characters the tool managed to use in the context of each field. The tool did not validate that the characters returned are sufficient to construct an exploit in the given HTML scope, and thus, requires the tester to perform manual validation.</p> <p>The XSS feature can be very useful as a complementary tool (similar to using Watcher and X5s), but not as the main tool (due to the large amount of false positives). On the other hand, the SQL injection feature is amazing (0 false positives! good detection rates in cases where error messages and exceptions are not presented), and should be used whenever possible (preferably with an external crawler, or while manually crawling).</p> <p>There is however a scoping problem that should be addressed here, since any request to any server will be scanned using this tool (there is no feature that enables defining domain restrictions), and as a result, the proxy may attack external web sites (when safebrowsing requests are performed by the browser, plugin update, etc). Therefore, this tool should be used with care (when burp is used as the external spider, the tester can configure it to transfer to the outgoing proxy only requests to specific domain, a feature that can mitigate this problem).</p>
WebCruiser Free Edition	<p>The spider successfully crawled all the URLs. I verified the scanning results by scanning several subdirectories directly, receiving identical results.</p>

Scanner	Scan Description
Grendel Scan	<p>The test was executed with the following configuration:</p> <p>In the test module selection I enabled both the GET and POST checkboxes in the Spider Form Baseline configuration, checked almost all the other features (except the spider URL regex and the search engine recon), and went over and configured each one (GET and POST enabled).</p> <p>I enabled all the Information disclosure Plugins, CRLF Injection, Directory Traversal, Generic Fuzzing and all XSS & SQLi plugins. I set the XSS testing aggression to High in the "XSS - query" and the "ErrorXSS" plugins, I enabled "SQL Tautologies" optional plugin.</p> <p>I defined burp-proxy as an outgoing proxy to make sure that the tool is working properly.</p> <p>The following URLs were scanned:</p> <p><code>http://localhost:8080/wavsep/index-xss.jsp</code> <code>http://localhost:8080/wavsep/index-sql.jsp</code> <code>http://localhost:8080/wavsep/index-false.jsp</code></p> <p>The tool successfully crawled all URLs, and even submitted values in POST parameters (something that came as a surprise to me since previous checks I performed showed that POST parameters were ignored by this tool).</p> <p>The initial scan discovered SQL injection vulnerabilities, but did not locate any XSS vulnerabilities, so I performed another scan in which the fuzzing and SQL injection plugins were disabled; that still didn't solve the problem, so I disabled all plugins except the XSS and spider plugins (and defined the XSS plugins with Medium aggression levels), and finally obtained the relevant missing results.</p>
JSKY Free Edition	<p>I enabled all the XSS, SQLi and LFI/RFI plugins, marked the "URLs are case sensitive" checkbox in the scan wizard and did not configure exclusions (the entire site is public).</p> <p>The scan was executed in front of the following URLs:</p> <p><code>http://192.168.1.100:8080/wavsep/index-xss.jsp</code> <code>http://192.168.1.100:8080/wavsep/index-sql.jsp</code> <code>http://192.168.1.100:8080/wavsep/index-false.jsp</code></p> <p>The tool managed to crawl all the URLs, detected vulnerabilities in GET parameters, but did not detect vulnerabilities in POST parameters.</p>
Scrawler	<p>I executed the scanner in front of the following URLs:</p> <p><code>http://192.168.1.100:8080/wavsep/index-sql.jsp</code> <code>http://192.168.1.100:8080/wavsep/index-false.jsp</code></p> <p>The scanner successfully crawled all URLs.</p>
Oedipus	<p>I used burp to create a log file by activating the spider feature on the various application pages.</p> <p>The following links were crawled:</p> <p><code>http://192.168.1.100:8080/wavsep/index-full.jsp</code> <code>http://192.168.1.100:8080/wavsep/index-sql.jsp</code> <code>http://192.168.1.100:8080/wavsep/index-xss.jsp</code> <code>http://192.168.1.100:8080/wavsep/index-false.jsp</code></p> <p>I executed the <code>o_analyzer</code> on the log file generated by burp, and scanned the application using the output of this analysis:</p> <pre>ruby o_analyzer.rb -f WavsepForOedipus.log -t burp http://192.168.1.100 ruby o_scanner.rb -f 03Dec2010151028.oedipus.192.168.1.100\input.oedipus -w all -y 127.0.0.1:9999 -p 8080 http://192.168.1.100</pre> <p>During the scan I experienced a number of memory leaks that caused the tool to crash (particularly in case 11 of 500Error and 200Error), and it took some effort and adjustments to complete the scan (deleting Case11 URLs from the input.oedipus file, increasing the RAM allocated to the VM, etc).</p> <p>The tool's results in the SQLi-200-Valid and SQLi-200-Identical cases were inconsistent, and changed from scan to scan.</p>

Scanner	Scan Description
NTOSpider	<p>NTOSpider was configured in the following manner: The scope was defined with “stay on port”, “restrict to directory” and “include default pages”; the attack policy selected was “Full Attacks “, and a server load was set to “Light”.</p> <p>I also included the “advanced attack” plugin in the attack configuration, and removed the authentication tests (since there isn’t any relevant vulnerabilities in WAVSEP).</p> <p>I eventually disabled the following plugins in the scans (it took too long – 2 hours+ per directory): OS Commanding, Arbitrary File Upload, Web Service Analysis, Authentication Testing (but still enabled the “Advanced Attacks”)</p> <p>After each scan, I verified that the tool successfully crawled all the URLs, and in addition, rescanned the directory while enabling only the directory specific plugins (SQLi/RXSS).</p> <p>During the tests, I have come across a severe bug in the POST RXSS detection mechanism, which caused the detection accuracy of RXSS vulnerable POST parameters to be less than half as efficient as the GET RXSS detection accuracy; Since the difference was extreme, and did not reflect the tool's capabilities, and since I notified other vendors that had similar problems, I contacted NTO's support and reported the issue.</p> <p>NTO quickly fixed the problem and supplied me with a fixed version (5_4_133), which was only used to re-scan the RXSS-POST test cases (apparently, a mechanism that was supposed to prevent identical forms from being repeatedly tested was responsible for the problem)</p>
ParosPro	<p>The scans were executed with the following plug-in groups enabled: Miscellaneous, Information Gathering, Injection, Client-Browser, Cross Site Scripting.</p> <p>Each directory was scanned separately.</p>
Acunetix WVS (Commercial Edition)	<p>I used the Default policy in the scans (and verified the results with other policies), while playing around with the tech-optimization feature (J2EE or none), and scanning each individual directory using both extensive & heuristic modes.</p> <p>The tool seemed to successfully scan all the test cases, without any exceptional incidents.</p> <p>Since the tested commercial version of Acunetix was older than the tested free version (20110608 vs 20110711), and since the results of the upgraded free version were actually better than the older commercial version I had tested, I changed the results of the commercial tool to match the ones of the new free version (from 22 to 24 in both the GET & POST RXSS detection scores).</p>
JSky (Commercial Edition)	<p>The scans were executed with the following configuration: "stay on the same dir", "Urls are case sensitive", "extract urls from javascript", 8 threads, "Crawl first and then scan", "Deeply execute javascript", and the "All" scan Policy (while occasionally repeating the scans with designated policies - e.g. XSS / SQLi).</p>
WebCruiser Enterprise Edition	<p>Each individual directory in WAVSEP was scanned separately, while enabling all the available plug-ins (SQLi, XSS, XPath – enabled by default).</p>
Vega	<p>I enabled all the scan plugins, and then attempted to scan each individual directory, only to find out that the spider module did not work properly (the tree-view presented all the pages as gray links, as if they were parsed from the index file but not accessed).</p> <p>I eventually activated the proxy module (there is a simple to use play button in the proxy tab) and used the crawler of burp proxy to crawl the links, while defining Vega as an upstream proxy.</p> <p>It did the trick, and although the tool had plenty of minor bugs, the results were interesting, and well worth the effort (open source and false positive free is a rare combination, after all).</p> <p>Most of the tests were performed with all the plugins enabled, but from time to time, I verified the consistency of the results by scanning directories while using only the dedicated test plugin (either SQLi or XSS).</p>
SandcatCS	<p>The scans were executed against each individual directory, using the following command: sandcatCS localhost:8080 -sn:SESSIONNAME -hm:appscan -surl:/wavsep/[directory]/index.jsp</p> <p>The results were verified by performing additional scans using the “xss” and “sqlinj” policies.</p>

Scanner	Scan Description
Netsparker (Commercial Edition)	<p>I originally tested version 1.9.0.5 against the benchmarking platform. The scanning process was performed with 5-8 threads, and with the entire list of scan plugins enabled (usually using the full scan policy, and on occasion, scans with a single thread). Additional verification were performed in-front of each directory, while using only the exposure-specific relevant plugins (XSS, RFI, Redirection & Http Header Injection for testing the RXSS vulnerable pages, and all the SQLi plugins for testing the SQL Injection vulnerable pages).</p> <p>During the tests, I have come across a bug in the POST RXSS detection mechanism, which caused the detection accuracy of RXSS vulnerable POST parameters to be less than half as efficient as the GET RXSS detection accuracy; Since the difference was extreme, and did not reflect the tool's capabilities, and since I notified other vendors that had similar problems, I contacted Netsparker support and reported the issue.</p> <p>Netsparker support quickly contacted me, and recommended I'll disable the URL rewrite crawling feature (I accessed "Settings > Crawling " and unchecked the "Heuristic URL Rewrite Support" feature), and that seemed to solve the problem (I'll try and remember that for the next benchmark). Version 1.9.0.5 had a solid 98.5 detection percent in SQL Injection and a 63.6 detection percent in XSS detection.</p> <p>A week before the benchmark planned release date (and two weeks after I tested their latest stable release – 1.9.0.5), Netsparker CEO contacted me and offered me to test their new version (2.0.0.0) which was released on the same day, using the same configuration I used for 1.9.0.5 (URL Rewrite Disabled, etc). I agreed, and the newly released version appeared to have a dramatic increased accuracy in Reflected XSS detection, an increase which was represented in the results presented for this tool.</p> <p>Note – the false positive SQL injection exposures detected by Netsparker were classified as "possible" and not as verified exposures; however, the case is the same with other scanners (such as Cenx Hailstorm and WebInspect), and since the positive accuracy calculation of almost all the scanners included these values, reporting a "potential" exposure counted as a false positive exposure for the purpose of the benchmark.</p>
safe3wvs	<p>I executed the tool against each individual directory, while enabling the following plugins: XSS, SQLi, all of the crawling plugins (JS Analysis, Form Submission, Auto-Redirect).</p> <p>The free version seemed to ignore POST parameters, even when the form submission flag was acticated.</p>
Nessus	<p>Nessus had 44739 plugins at the time of the test, and the number surely grew by now.</p> <p>Nessus was executed using the default "Web App Tests" policy, with the following changes: Port range – 8080 (only), Disabled the safe checks flag, marked "Consider unscanned ports as closed", TCP Scan.</p> <p>The following web application specific configuration was used throughout the test: WebAppTests and custom policies: RunTime of 300 minutes Tests: CGI Abuses/CGI Abuses: XSS/ Web Servers (and sometimes - Enable all plugins) Web Application Test Settings: Enable web application tests, All combinations (Extremely slow), Http Parameter Pullotion – enabled (look for all), Test Embedded Web Server, Enable CGI Scanning (global variable settings)</p> <p>In order to cause Nessus to scan only one WAVSEP directory each time, I added an index.html file to the web server root, and included both a link & JS redirection to the directory that was the target of the scan; the value of the link & JS code was altered to redirect to another directory, after each scan was completed... I only discovered the "Web Mirroring" configuration by the end of the test (would have been much easier to use this feature to obtain the same result)...</p> <p>The following plugins detected XSS vulnerabilities: CGI Generic HTML Injections, CGI Generic Cookie Injection Scripting, CGI Generic Cross Site Scripting Vulnerability (quick test), CGI Generic Cross Site Scripting Vulnerability (extended test)</p> <p>The following plugins detected SQL Injection vulnerabilities: CGI Generic SQL Injection Vulnerability, CGI Generic SQL Injection (blind, time based), CGI Generic SQL Injection (blind), CGI Generic SQL Injection Vulnerability (2nd pass),</p> <p>And in one scenario: CGI Generic Command Execution Vulnerability (time-based)</p> <p>The following plugins were responsible for some of the false positives: CGI Generic SQL Injection Vulnerability (Parameters Names), CGI Generic 2nd Order SQL Injection Detection (potential), CGI Generic SQL Injection Vulnerability (HTTP Headers), CGI Generic SQL Injection detection (potential, 2nd order, 2nd pass)</p>

Scanner	Scan Description
IBM Rational AppScan	<p>I contacted IBM Rational Appscan to get an evaluation version, and support for properly configuring & executing the product. After receiving the latest evaluation version of the product (version 8.0.0.3 – which was a limited iFix version that addressed several bugs that might have affected the scan), I configured the tool according to the detailed recommendations I received from the relevant personal at IBM Rational AppScan:</p> <p>Configuration Values: Communication and Proxy – Number of Threads -> 1 Communication and Proxy ->Timeout -> 5 (instead of 15; Optional – On Occasion) Automatic Form Fill-> User & Password Defined Test Options -> Disabled Adaptive Testing (URL Rewrite Prevention) Advanced Configuration-> Similarity Threshold ->99</p> <p>XSS Scanning Plugins: Cross Site Scripting (All Variants) Link Injection Phishing through Frames Phishing Through URL Redirection PHP Remote File Inclusion</p> <p>SQL Injection Scanning Plugins Authentication Bypass via SQL Injection SQL Injection (All Variations) SQL Injection Command Execution (All Variations) (I did not use SQL Injection using Declare / Cast / Exec)</p> <p>I attempted various configurations throughout the scans, scanned each directory multiple times, and performed separate scans using the time-based SQL Injection detection plugins (The time based attacks were executed both separately & alongside the rest of the SQLi Plugins, and executing them separately did not improve the results in this case).</p>
Sandcat Pro	<p>I tested each individual directory, using with multiple configuration, while using the following plugins:</p> <p>General->Common Exposures Injection->CRLF Injection Injection->Cross Frame Scripting Injection->Cross-Site Scripting (XSS) Injection->File Inclusion Injection->Miscellaneous</p> <p>Injection->SQL Injection Injection->XPath Injection Injection->Miscellaneous</p>
N-Stalker 2012 Free Edition	<p>I downloaded and tested N-Stalker 2012 shortly before publishing the article. I updated all of the tool's databases on the 31/07/2011 (to build 7.1.1.106), and used the XSS Scan policy to separately scan each of the XSS-vulnerable directories in WAVSEP. I tried scanning using various configurations, with and without optimizations. The result I got was probably the most surprising I've seen in this research: When N-Stalker released the 2012 free version, they announced that the free version will have reduced RXSS detection capabilities, and they truly meant what they said... N-Stalker 2009 FE (the previous version free edition released by N-Stalker) was a fantastic tool, In spite of all of its restrictions (100 URL limit, etc), which had an RXSS detection accuracy of 40 out of 66 test cases, one of the best scores in my previous benchmark (and furthermore... this result was also false positive free! a result that few tools matched!)... N-Stalker 2012 however, can only detect 8 out of 66 cases, less than a quarter of the previous version!!! True, it's not subject to the same restrictions, and it has some nice features, and furthermore, removing features from a free product is a legitimate business oriented decision of any commercial enterprise... It doesn't really fall into the category of good or bad, but I guess I'm still surprised by the move.</p>

Scanner	Scan Description
Damn Small SQLi Scanner (DSSS)	<p>DSSS (Damn Simple SQLi Scanner) is a proof of concept tool designed to prove that the accuracy of commercial vendors can be beaten with less than 200 lines of python code... written by Miroslav Stampar, one of the authors of sqlmap.</p> <p>Even though the tool didn't support POST parameters (only GET parameters were supported, and only at a crawling depth of 1), it still provided damn good results for less than 100 lines of python code.</p> <p>I used python 2.6.5 to execute DSSS, and executed the tool using the following command: python dsss.py --url=[URL]</p>
Watobo	<p>The application was crawled and tested in two different methods: a single scan performed after crawling the main index page via Burp (while channeling the communication to WATOBO via the proxy chaining feature), and several individual crawling & scanning operations that were similarly performed against each individual directory.</p> <p>In the first scenario, the crawling process was initiated on the following URLs, while activating the XSS, SQL and LFI plugins (separately for each scope): http://localhost:8080/wavsep/index-xss.jsp http://localhost:8080/wavsep/index-sql.jsp http://localhost:8080/wavsep/index-false.jsp</p> <p>In the second scenario, the crawling & scanning processes were performed on each directory individually, while verifying that all the access points in each directory were located.</p> <p>I tried scanning with & without the "smart scan" feature, and used 8 max persistent requests.</p>
WebSecurityfy	<p>The scan was separately initiated in front of the XSS vulnerable pages (http://localhost:8080/wavsep/index-xss.jsp), the SQL vulnerable pages (http://localhost:8080/wavsep/index-sql.jsp) and the false positive pages (http://localhost:8080/wavsep/index-false.jsp).</p> <p>The results were consistent, and were obtained using Websecurityfy 0.8 (final).</p>
SkipFish	<p>I scanned the application using version 2.02b, without any dictionary or bruteforce features, and initiated the scan using the following commands: ./skipfish -LV -W /dev/null -Y -o [OutputDirName] -g 5 -m 5 http://[ip]:[port]/wavsep/[wavsep-dir]/index.jsp</p> <p>Even though the results were better than the last test results, the tool provided no indication that it actually crawled all the URLs, so I created a URL list using burp (copied the links to a file), and scanned using the following command: ./skipfish -LV -W /dev/null -Y -o [OutputDirName] -g 5 -m 5 @[URL-File-Name] ./skipfish -LV -W /dev/null -Y -o [OutputDirName] -g 1 -m 1 @[URL-File-Name]</p> <p>The results were identical, even though I provided all the URLs in a file, so I'm not sure the problem is related to crawling issues (an assumption made since the report does not contain all the URLs)</p>
Netsparker Community Edition	<p>A bug in the XSS verification mechanism of the scanner caused a VBScript msgbox popup to appear each time the "msgbox" method was used in the page, and worse, the scanner seemed to be stuck after several msgbox dialogs appeared (the same bug appeared in the previous version of the tool); Instead of trying to "click" my way through the test (like I did with the previous version), I decided to alter the code that caused the bug in the tool; I replaced the VBScript function "MsgBox" with "Document.Write" in test cases 13,14,15,20,21,25,26 and 29 (16 cases total) of WAVSEP's RXSS-GET & RXSS-POST source code to avoid the VBScript pop up bug.</p> <p>After re-initiating the scan on the updated version of WAVSEP (1.0.1), various exceptions started to appear when the scanner got to 70%-80% (appeared on multiple scans), so I documented the exceptions, paused the scan and re-initiated it, and repeated the procedure until the tool completed the scan (3-4 times).</p> <p>In order to verify the exception consistency, I configured the tool to use 8 threads and only the XSS detection plug-in, and re-activated the scan again on the RXSS URLs, but exceptions still occurred when the scan got to about 80 percent; furthermore, for some reason, the detection of RXSS case 16 (GET) was inconsistent.</p> <p>After testing the RXSS detection accuracy and the false positive ration, I met with some serious difficulties when I tried scanning the SQL Injection test cases; the results were very inconsistent, and the tool "missed" many test cases that its previous versions successfully detected.</p> <p>After further investigation, I discovered that the blame was myne... WAVSEP's database access was not implemented with a connection pool, and thus, there were many database connection failures, resulting in false negatives for many tools; The problem was solved by reducing the amount of threads and scanning each directory individually. Further re-verification showed consistent results, and the scanning method was replicated for the rest of the tools in the benchmark (toDo: implement connection pool for the next version of WAVSEP).</p> <p>After scanning each directory individually, the tool provided far better results (although there still was inconsistency in the results of erroneous injections, especially in verified reported exposures).</p> <p>The overall results were slightly better than the results of the previous versions of Netsparker, but it is currently unknown if the inconsistencies and accuracy improvement derive from the changes in the software, the test lab or different method of test execution (most probably).</p>

Scanner	Scan Description
W3AF	<p>I used W3AF built in web spider to crawl the URLs in each directory, and executed a scan with the following plugins enabled: webspider, xss, sqli, blindSqli, remoteFileInclude, globalRedirect, phishingVector, eval</p> <p>The tool did report several exceptions, but the most of the scans were completed successfully, that is until I started scanning the 200-Valid test cases.</p> <p>The GUI kept freezing and presenting exceptions whenever I used w3af to scan 200-Valid test cases (blind sql injection test cases), so I eventually directed the output to an html report, waited for the scan to complete (apparently, the scanning process continued, even though the GUI freezes), and analyzed the results in the HTML report.</p> <p>I had to scan some of the directories several times, since for some bizarre reason, the "sqli" plugin did not seem to work if the output related detection plugins were disabled (xss,redirect,phishing).</p> <p>The results were a bit less accurate than the last version tested, since the eval injection plugin missed case 19 (and other test cases) in all the POST based categories (bug?).</p>
Andiparos	<p>The application was crawled and tested in two different methods: a single scan performed after crawling the main index page, and several individual crawling & scanning operations that were performed against each individual directory.</p> <p>The scanning process was performed after enabling all the plugins in the following categories: HTML Injection, SQL Injection, Miscellaneous .</p> <p>The output word "SQL" seems to be a key word in the detection process, and every appearance of it causes the tool to identify the page as vulnerable (and as a result it was removed from the static HTML of all the tested pages).</p> <p>The tool is still prone to various bugs, such as failure to generate reports, issues related to the alerts tab auto-refresh and unstable spider that sometimes requires several crawling operations</p>
Acunetix WVS Free Edition	<p>I initially tested version 7.0-20110608, but discovered a bug in the free edition that caused the tool not to present any results after scanning vulnerable pages (the commercial edition was not prone to the same bug, even though the version was identical). I reported this bug to Acunetix, and they released the fixed version after a week or two (which also added support for detecting 2 additional test cases).</p> <p>Since the tested commercial version of Acunetix was older than the tested free version (20110608 vs 20110711), and since the results of the upgraded free version were actually better than the older commercial version I had tested, I changed the results of the commercial tool to match the ones of the new free version (from 22 to 24 in both the GET & POST RXSS detection scores).</p> <p>The test was initiated in front of each individual directory using the default configuration, a disabled port scanner feature, without any optimizations, and while scanning with both Extensive & Heuristic modes (separate scans).</p> <p>All the URLs were crawled successfully, and the Extensive & Heuristic scans found identical results.</p>