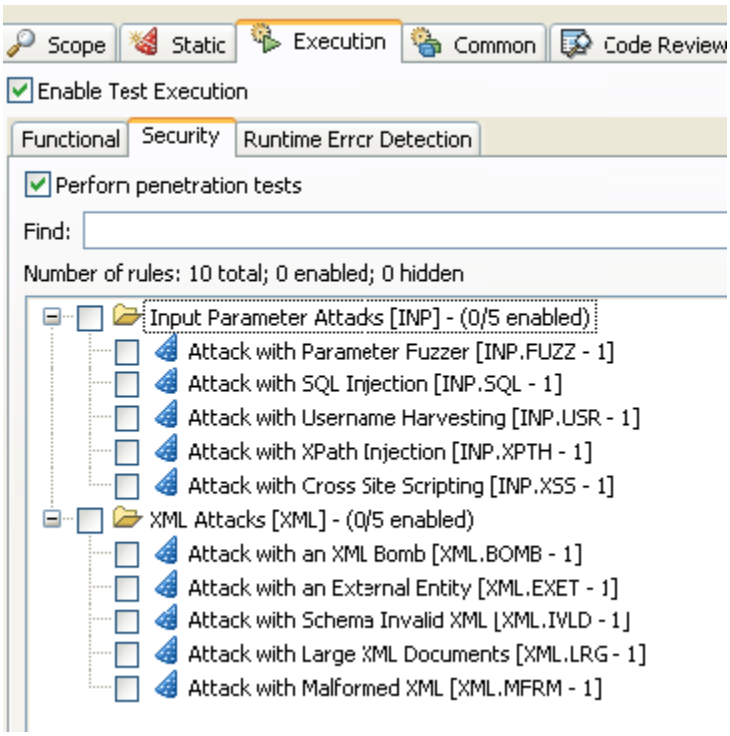
Penetration Testing

SOAtest's penetration testing can generate and run a variety of attack scenarios against your func-tional test suites.
Sections include:

Configuring Penetration Testing Attacks

To configure SOAtest to simulate attacks against your functional tests scenarios:

1. Choose **Parasoft> Test Configurations** to open the Test Configuration manager.
2. Click **New** to create a new Test Configuration.
3. Give the new Test Configuration a meaningful name.
4. Open that Test Configuration's **Execution> Security** tab.
5. Enable **Perform penetration testing**.
6. Use the rule tree to indicate which attacks you want to run.



Available Attacks

SOAtest can simulate the following attacks:

Attack	Description
Parameter Fuzzing	When input parameters to a Web service are not properly validated, it can lead to vulnerabilities in the underlying system. In native applications, buffer overflow attacks can occur when input parameter data sizes go unchecked. These vulnerabilities could cause system crashes or could even lead to unauthorized information being returned to the client application.
SQL Injections	When SQL statements are dynamically created as software executes, there is an opportunity for a security breach by passing fixed inputs into the SQL statement, making them a part of the SQL statement. This could allow an attacker to gain access to privileged data, login to password-protected areas without a proper login, remove database tables, add new entries to the database, or even login to an application with admin privileges.

Username Harvesting	A request that includes a wrong username or password should not be met with a response that indicates whether the username is valid or not; this would make it easier for an attacker to identify valid usernames, then use them to guess the passwords.
XPath Injections	XPath injections are similar to SQL injections in that they are both specific forms of code injection attacks. XPaths enable you to query XML documents for nodes that match certain criteria. If such a query is constructed dynamically in the application code (with string concatenation) using invalidated inputs, then an attacker could inject XPath queries to retrieve unauthorized data.
Cross-Site Scripting	Cross-site scripting problems occur when user-modifiable data is output verbatim to HTML. Subsequently, an attacker can submit script tags with malicious code, which is then executed on the client browser. This allows an attacker to deface a site, steal credentials of legitimate users, and gain access to private data.
XML Bombs	When using a DTD (Document Type Definition) within an XML Document, a Denial of Service attack can be executed by defining a recursive entity declaration that, when parsed, can quickly explode exponentially to a large number of XML elements. This can consume the XML parser resources, causing a denial of service.
External Entities	XML has the ability to build data dynamically by pointing to a URI where the actual data is located. An attacker may be able to replace the data that is being collected with malicious data. This URI can either be pointed to local XML files on the Web service's file system to make the XML parser read large amounts of data, to steal confidential information, or launch DoS attacks on other servers by having the compromised system appear as the attacker by specifying the URLs of the other servers.
Schema Invalid XML	A well-formed document is not necessarily a valid document. Without referencing either a DTD or a schema, there is no way to verify whether the XML document is valid or not. Therefore, measures must be taken to ensure that XML documents do, in fact, reference a DTD or schema.
Large XML Documents	Large payloads can be used to attack a Web service in two ways. First, a Web ser-vice can be clogged by sending a huge XML payload in the SOAP request, especially if the request is a well-formed SOAP request and it validates against the schema. Secondly, large payloads can also be induced by sending certain request queries that result in large responses.
Malformed XML	XML elements with malformed, unacceptable, or unexpected contents can cause the service to fail.

Attack String Customization

To customize the attack strings used for various attacks, modify the .csv files in `[SOAtest_install_dir]/plugins/com.parasoft.xtest.libs.web_[version]\root\security.`

Configuring Runtime Error Detection for Hybrid Security Analysis

If your application has a Java backend and you want to apply runtime error detection in order to deter-mine if these attacks actually cause security breaches or other runtime defects, you should also configure runtime error detection as described in Performing Runtime Error Detection.

Be sure to configure both:

- The server.
- The Test Configuration that you will use to perform penetration testing (see Configuring Penetration Testing Attacks).

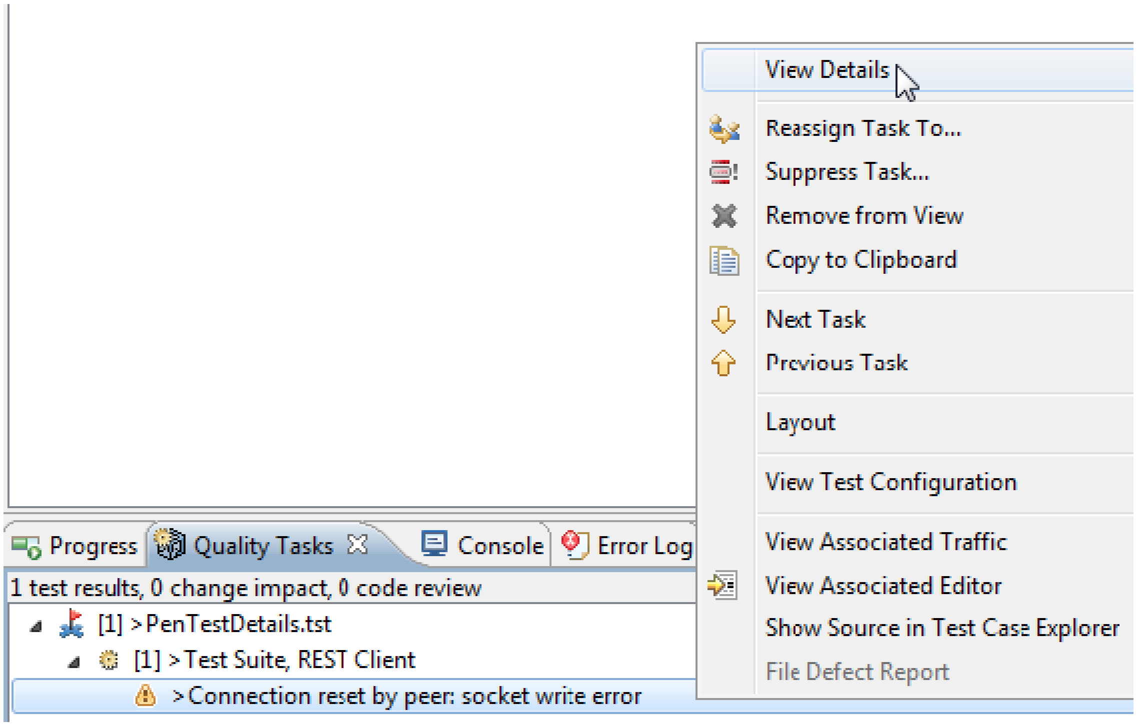
Executing Tests

To run the penetration tests:

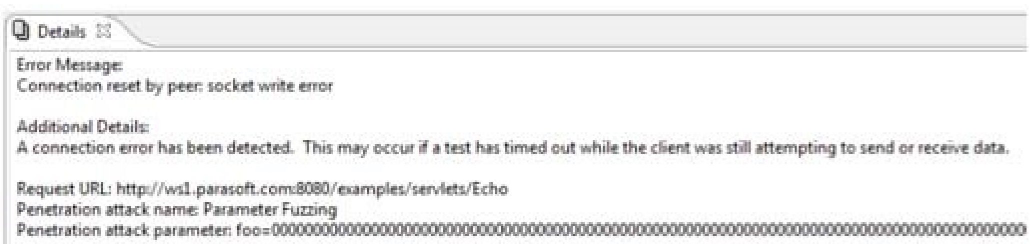
1. Select the test suite that you want to attack.
2. Run the Test Configuration that you designed for penetration testing (see Configuring Penetration Testing Attacks).

Reviewing and Validating Results

Results will be reported in the SOAtest tab and in any reports generated.

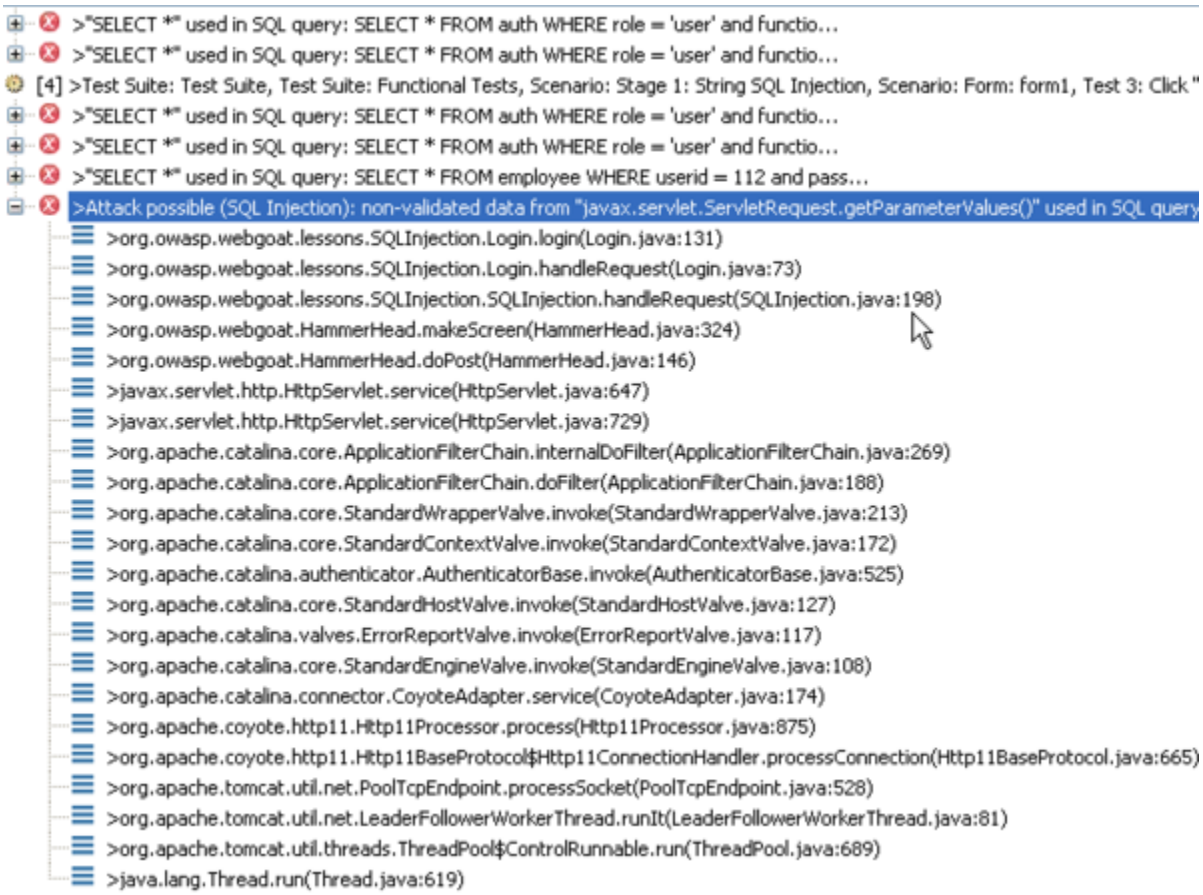


Double-clicking on this message (or right-clicking and choosing **View Details**) will open information about which attack caused the failure.



With Runtime Error Detection Enabled

If you performed hybrid analysis (penetration testing + runtime error detection), errors detected will be reported as follows:



The same details will also be provided in reports.

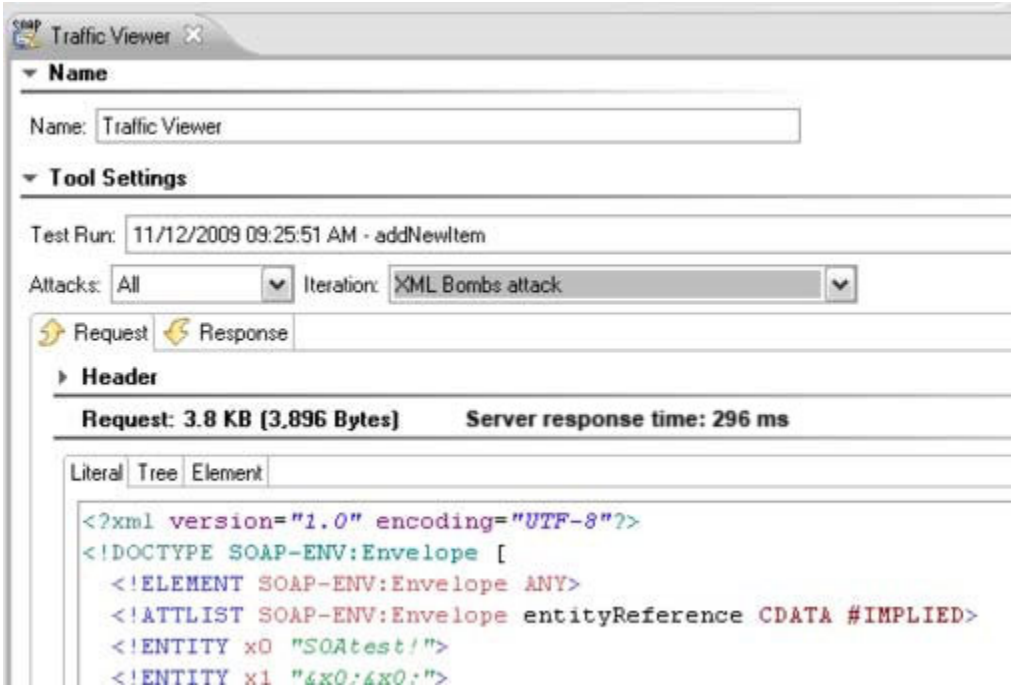
Note that SOAtest correlates each reported error with the functional test that was being run when the error was detected. This correlation between violations and functional tests allows you to trace each reported error to particular use cases against your application.

Additional Validation

Additional validation strategies can be applied to suit your needs. For example, you can chain Coding Standards, Search, or XML Validator tools to the test suite, inspect server logs manually, or run a script to parse these logs.

Viewing Attack Traffic

The Traffic Viewer for each test allows you to view attack traffic. Using the available **Attacks** and **Iteration** controls, you can display traffic for all attacks or for specific attack types, as well as focus on traffic for specific attack values.

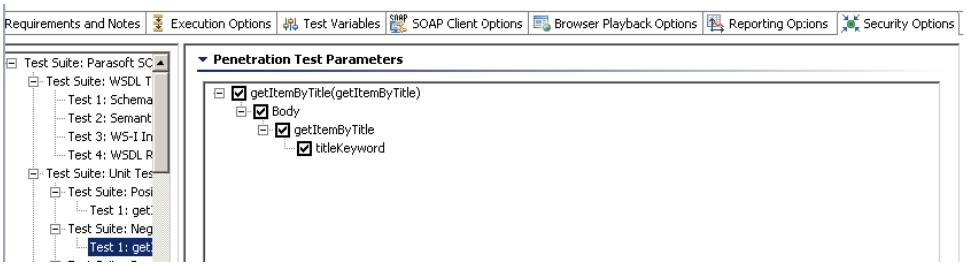


Configuring Attackable Parameters

By default, SOAtest will try to attack all of the available parameters represented in a selected test suite's SOAP Client, REST Client, Messaging Client, and Browser Playback tools.

To customize which parameters may be attacked:

1. Double-click the top-level test suite node for functional tests you want to attack.
2. Open the **Security Options** tab (on the far right).
3. Use the Penetration Test Parameter tree to indicate which parameters can be attacked.



4. Save the test suite configuration changes.