WAF Testing Framework

Summary

WAF Testing Framework is a framework envisioned by Imperva, which enables a security stakeholder to evaluate the security controls put in place in front of the organizational Web applications. It is bundled with WebGoat by OWASP, so it provides a full, standalone testing environment.

The framework allows testing of the effectiveness of a Web Application Firewall (WAF) or any other Security Control that is put in place to protect the Web application from being attacked.

The WAF Testing Framework evaluates the security performance of the Security Control in order to highlight policies, rules, settings, etc., which may be resulting in either lax or excessive security. Unlike other Web Application Security testing tools that focus exclusively on generating attack traffic, this framework generates both attack traffic and legitimate traffic. This approach makes it possible to test the ability of a Security Control to detect malicious traffic and also to distinguish malicious traffic from good traffic. It provides a real world testing scenario in which the Security Control must block attack traffic, and avoid blocking good traffic (i.e., generating false positives). In addition, the WAF testing framework tests for sophisticated and stateful attacks such as cookie tampering and cookie injection. This ability is rarely included in other WAF testing methodologies, but is critical when evaluating Web application security solutions.

Note: A Security Control may be a WAF or any other Control that is able to protect a Web application from Web attacks, such as an Intrusion Prevention System or a Next-Generation Firewall.

Background

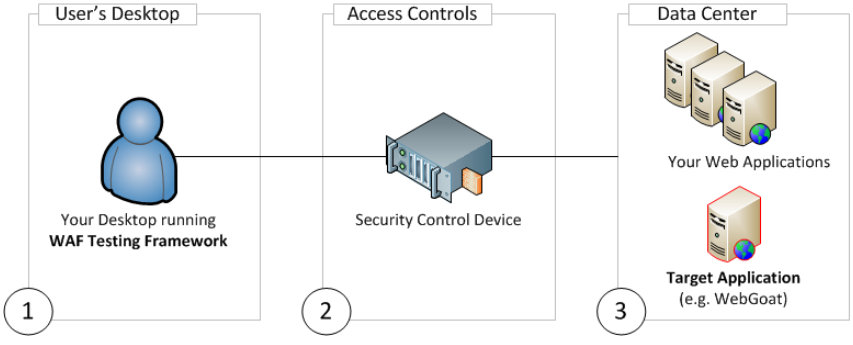
This document describes the methodology used in order to evaluate the Security Controls put in place in order to establish both the environment requirements and the test methodology.

Testing Environment

The WAF Testing Framework test environment is constructed out of several elements, all required in order to perform proper testing. We will describe the use of these components in the “Test Bench” section.

The components are:

1. **WAF Testing Framework** – the tool itself will generate the traffic for the benchmark.
2. **Tested Security Control** – a Security Control configured to protect the target application. Should be deployed in blocking mode. This may be a WAF or any other Control that is able to protect a web application from web attacks.
3. **Target Application** – this will be the test application used. The traffic from the WAF Testing Framework should be directed towards the test application. The default test base which is included in the tool was recorded with OWASP’s WebGoat. Using the framework with another application is possible by recording a new test set.



Test Bench

To run the benchmark, here are the prerequisites and deployment steps:

- Deploy WebGoat behind the Security Control you wish to evaluate (as described under “Target Application”)
  Make sure that it is running and that you have access to it from your web browser.
- Make sure that your environment meets the Diagram drawn above where the Target Application which you will run the tests against is on the other side of the Security Control you are evaluating.
- Execute the Framework and follow the instructions (as described under “Execution”)

Target Application

The target application which is bundled with WAF Testing Framework is OWASP WebGoat, a freely available and intentionally insecure Web application from OWASP. WebGoat should be deployed on a server which is protected by the tested WAF. The WAF Testing Framework executes HTTP requests against WebGoat, and evaluates which requests were blocked by the WAF.

NOTE: To ensure smooth operation of the WAF Testing Framework, use the version of WebGoat which is bundled with the WAF Testing Framework package.

To deploy WebGoat, copy the application archive to the appropriate server and extract it, then follow the platform specific instructions.

Windows:

WebGoat may be deployed on port 80 or 8080. Launch the appropriate executable accordingly:

`webgoat_8080.bat` <OR>

`webgoat.bat`

Linux:

Java 1.6 must be deployed to run WebGoat in a Linux environment. If Java 1.6 is not installed, download and extract it, and then point the JAVA_HOME environment variable to the appropriate directory.

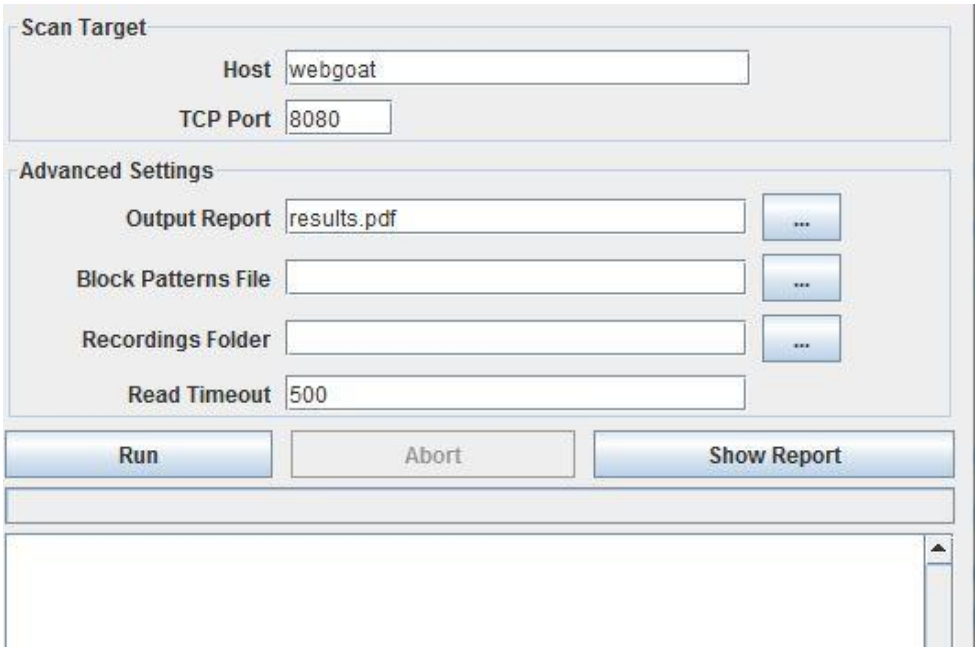
To launch WebGoat on Linux:

`./webgoat.sh start8080` <OR>

`./webgoat.sh start80`

Execution

The WAF Testing Framework software is deployed as a GUI application on Windows. To run the tool, launch the executable named "WAFTesting.exe".



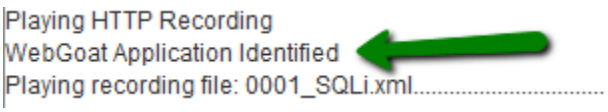
The following configuration options are available in the GUI:

- **Host:** the host name or IP of the target Web application, i.e. WebGoat.
- **Port:** the port of the target Web application.
- **Output Report:** the file name for the output report.
- **Block Patterns File:** in case that the tested security control (i.e. Web Application Firewall) uses an error page to indicate that a request were blocked, use this field to specify the error page configuration location. A sample file named "custom_blocking.xml" is included in the package. See "Error Page Configuration" below for more information on configuring an error page. Note that this field is not required for Imperva's SecureSphere.
- **Recordings Folder (optional)**: path to a directory containing custom input XML recording files. The directory may contain any number of input files. An example of an input XML file is provided with this framework.
- **Read Timeout**: the timeout for a response from the server. Note that requests that are timed out are considered as blocked by the tested device.

After completing the configuration, click the **Run** button to execute the benchmark. During the execution, the progress bar and notification area will indicate on the benchmark progress. The **Abort** button may be used to terminate the execution prematurely. Once the execution is done, click on the **Show Report** button to show the result report.

Note that in case the framework is used with WebGoat, a message would appear indicating that the WebGoat application was identified correctly. If the message does not appear, please double check the deployment and configuration.

In any case of unexpected results, refer to the *error.log* and *trace.log* files in the working directory.



Error Page Configuration

The WAF Testing Framework operates by sending requests to the target Web application and checking if they were blocked by the tested WAF. The verification is performed by comparing the request to a pattern which indicates that the WAF sent back an error page.

The blocking pattern for Imperva SecureSphere is included within the framework. For WAFs other than SecureSphere, the user can configure a custom blocking pattern(s) in the form of an XML file. The file includes a string pattern and response code which corresponds to the WAF error page. For example, the following XML contains two error indications:

```
<blocking-indications>
    <error-page>
        <status-code value="403"/>
        <page-content><![CDATA[You don't have permission to access]]></page-content>
    </error-page>
    <error-page>
        <status-code value="200"/>
        <page-content><![CDATA[Request rejected]]></page-content>
    </error-page>
</blocking-indications>
```

Summary

The WAF Testing Framework enables organizations to fully test their chosen Web Security Control by analyzing its capabilities to analyze both legitimate and illegitimate traffic, providing a comprehensive report of both False Positives and False Negatives, and therefor provides a real benchmark to the chosen security control and its capabilities.

For questions, please contact waftf@imperva.com.

Imperva
3400 Bridge Parkway, Suite 200
Redwood City, CA 94065
www.imperva.com