HP DevInspect for Java Software

for the Windows® operating system

Software Version: 5.00

User Guide

Document Release Date: February 2008

Software Release Date: February 2008

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation.

Windows Vista is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

Adobe and Acrobat are trademarks of Adobe Systems Incorporated.

Documentation Updates

This guide's title page contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates, or to verify that you are using the most recent edition of a document, visit the following URL:

http://h20230.www2.hp.com/selfsolve/manuals

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

For information or assistance regarding DevInspect, contact customer support:

E-mail: spisupport@hp.com

Telephone: 866-774-2700

You can also visit the HP software support web site at:

www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides an efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the HP Software Support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels and HP Passport, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

http://h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Contents

1 Welcome to DevInspect for Java

Introduction

DevInspect for Java accelerates the construction and delivery of secure Web applications by identifying vulnerabilities without leaving the integrated development environment. For the first time, developers have the ability to build secure Web applications quickly and easily, without the need for specialized security knowledge and without the risk of slowing aggressive product release schedules.

Minimum System Requirements

Before installing DevInspect, make sure that your system meets the following minimum requirements:

- Windows XP SP2 or Vista
- 1 GB of RAM
- 1.2 GB of free disk space required (2 GB preferred)
- 1 GHz processor or better
- Microsoft .NET Framework 2.0
- Microsoft SQL Server Express SP1
- JDK 1.4.2 or above
- For plug-in versions: Eclipse 3.2 or 3.3, Rational Application Developer 6 or 7

The English-language version of Windows is required for successful installation and operation of DevInspect.

Main Features of DevInspect

DevInspect is designed to fit naturally with the way a developer works every day so that secure development becomes as familiar as coding and unit testing. Developers can secure their application and improve their security expertise during any phase of development without ever leaving the Java IDE.

Accuracy and Precision

- Combines source code analysis and black box testing to achieve unmatched confidence in the results.

- Analyzes security of application configuration.
- Bases vulnerability risk ratings on impact and probability.

Real-time Security View of Application

- Permits regular vulnerability data updates through SmartUpdate from expert security researchers at HP.
- Finds vulnerabilities exposed through third-party components.
- Incorporates advanced script parsing and interpreting for JavaScript, VB Script, and Flash.

Flexible Options

- Analyzes entire application or individual pages.
- Conducts automated scans.
- Allows customization of security policies.
- Detects several types of attacks, including SQL injection, cross-site scripting, and buffer overflow.

Education for Developers

- Includes detailed vulnerability description and exploit information.
- Enables sharing of security data between developers.
- Provides in-depth vulnerability reporting.

Integrated Security Tools

- HTTP Editor
- Web Proxy
- Web Form Editor
- Web Macro Recorder

Refer to Appendix A, DevInspect Tools, for information about using the integrated security tools.

Installing DevInspect

The DevInspect 5.0 for Java installation package is a single executable that you can obtain from the path specified in the e-mail message you received from HP. DevInspect can be installed in the following configurations:

- Standalone - This option installs Eclipse with the DevInspect features built in. You can choose this option no matter what Java IDE you are using, even if you already have Eclipse installed. This includes a full Eclipse 3.2 with WTP 1.5.
- Eclipse plug-in - This option installs DevInspect features into an existing installation of Eclipse 3.2 or higher.
- IBM Rational Software Development Platform plug-in - This option installs DevInspect features into an existing installation of IBM Rational Software Development Platform 6 and 7 products, including Rational Application Developer, Web Developer and Software Architect.

Install Using Installation Program

You can safely install DevInspect side by side with other HP products. To install, double-click on the installation package and follow the instructions. The installer will place DevInspect in the following locations by default.

Program files:

C:\Program files\HP\DevInspect for Java

Scan files and logs:

C:\Documents and Settings\ <username> \Application Data\HP\DevInspect\Eclipse\5.0

Install Using HP Site

DevInspect 5.0 for Java can also be installed from within your existing Eclipse installation through the HP update site if you are choosing the Eclipse or IBM Rational Software Development Platform plug-in options.

1. Click **Help > Software Updates > Find and Install**.
2. On the *Install/Update* dialog, select **Search for new features to install** and click **Next**.
3. If "HP Application Security Center" is not listed:
   a. Click **New Remote Site**.
   b. On the *New Update Site* dialog, in the **Name** box, enter "HP Application Security Center."
   c. In the **URL** box, enter one of the following, depending on your target Eclipse platform:

      Eclipse 3.2:

      http://updates.spidynamics.com/eclipse/eclipse-3.2/site.xml

      RSDP6:

      http://updates.spidynamics.com/eclipse/RSDP6/site.xml

RSDP7:

http://updates.spidynamics.com/eclipse/RSDP7/site.xml

d Click **OK**.

4 Make sure only “HP Application Security Center” is checked in the **Sites to include in search** list.

5 Click **Finish**.

6 On the Search Results panel, select DevInspect for Java (<target>) 5.0.x.x, where <target> indicates your platform.

7 Click **Next** and follow the on-screen instructions to complete the installation.

➤ When installing from the update site, the Eclipse Update Manager will automatically download and install .NET Framework 2.0 and SQL Server 2005 Express Edition, if needed.

2 Getting Started

Overview

DevInspect is an aggressive Web application analyzer that rigorously inspects your Web site or project for real and potential security vulnerabilities. This procedure is intrusive to varying degrees. Depending on which options you select, it can affect server and application throughput and efficiency.

Prepare Your System for Audit

During an audit of any type, DevInspect submits a large number of requests, many of which have “invalid” parameters. On slower systems, the volume of HTTP requests may degrade or deny access to the system by other users. Additionally, if you are using an intrusion detection system, it will identify numerous illegal access attempts.

To conduct a thorough assessment, DevInspect attempts to identify every page, form, file, and folder that composes your application. If you select the option to submit forms during a crawl of your site, DevInspect will complete and submit all forms it encounters. Although this enables DevInspect to navigate seamlessly through your application, it may also produce the following consequences:

- If, when a user normally submits a form, the application creates and sends e-mails or bulletin board postings (to a product support or sales group, for example), DevInspect will also generate these messages as part of its probe.
- If normal form submission causes records to be added to a database, then forms submitted by DevInspect will create spurious records.

During the audit phase of an assessment, DevInspect resubmits forms numerous times, manipulating every possible parameter to reveal problems in the applications. This will greatly increase the number of messages and database records created.

Helpful Hints

Create a Backup

For systems that write records to a back-end server (database, LDAP, etc.) based on forms submitted by clients, some DevInspect users, before auditing their production system, create a backup copy of their database and then reinstall it after the audit is complete. If this is not feasible, you can query your servers after the audit, searching for and deleting records that contain one or more of the default form values used by DevInspect. You can determine these values by using the Web Form Editor.

Consider E-mail Traffic

If your system generates e-mail messages in response to user-submitted forms, you might consider disabling your mail server. Alternatively, you could redirect all e-mails to a queue and then, following the audit, manually review and delete those e-mails that were generated in response to forms submitted by DevInspect.

Pace HTTP Traffic

DevInspect can be configured to send up to 100 concurrent HTTP requests before waiting for an HTTP response to the first request. The default maximum concurrent request count setting is 1 (when the auditor and crawler share connections). Increasing this setting will increase the speed of a scan. However, increasing the speed also increases the drain on system resources, as well as those of the server you are scanning. You can modify these settings through **Window** > **Preferences** > **Web Application Security** > **Requests**.

Configure the User Interface

Follow the suggestions below to configure the development environment for using DevInspect.

Task 1: Select the J2EE perspective

The J2EE perspective is recommended, but not required.

1 Click **Window** > **Open Perspective** > **Other**.

2 On the Open Perspective dialog, select **J2EE**.

3 Click **OK**.

Task 2: Select the Vulnerability Testing Console

1 Select the **Console** tab.

2 Click the **Open Console** drop-down list.

3 Select **Vulnerability Testing**.

Task 3: Open a Project (If scanning a project):

1 Open a Dynamic Web Project.

2 Make sure it is deployed to an integrated server.

Configure the DevInspect Scan Engine

You can tailor DevInspect to your specific development environment, allowing you to obtain optimum performance when analyzing your Web site or application. For detailed information and instructions, see Chapter 4, Workbench Preferences, and Chapter 5, Project Properties.

Create a Table of Values for Forms

Most Web applications contain HTML or JavaScript forms containing input controls (text boxes, buttons, drop-down lists, etc.). Users generally “complete” a form by modifying its input controls (such as entering text or checking boxes) before submitting the form to an agent for processing. Usually, this processing will lead the user to another page or section of the application. For example, after completing a login form, the user will proceed to the application's beginning page.

If DevInspect is to navigate through all possible links in the application, it must be able to submit appropriate data for each form. The Web Form Editor can help you build a table of control names and values that can be used for navigating your Web site. When DevInspect encounters a control, it will extract the associated value from the table and populate the control.

To access the Web Form Editor:

1. Click **Window** > **Preferences** > **Web Application Security** > **Advanced**.
2. Select the **Web Forms** tab.
3. Click **Launch Web Form Editor**.

Update Your SecureBase Database

HP engineers uncover new vulnerabilities almost every day. They develop attack agents to search for these malicious threats and then update our corporate database so that you will always be on the leading edge of Web application security. You should update your DevInspect platform each time you use it.

To download the latest adaptive agents, as well as vulnerability and policy information, click **DevInspect** > **Tools** > **SmartUpdate**.

3 Using DevInspect

Scan for Vulnerabilities

Use the following procedure to inspect a Web site or project.

1 To scan projects, open one or more dynamic Web projects (not necessary if scanning a remote site).

2 Do one of the following:

- Click the DevInspect icon on the toolbar.
- Click the **DevInspect** menu and select **Test for Vulnerabilities**.

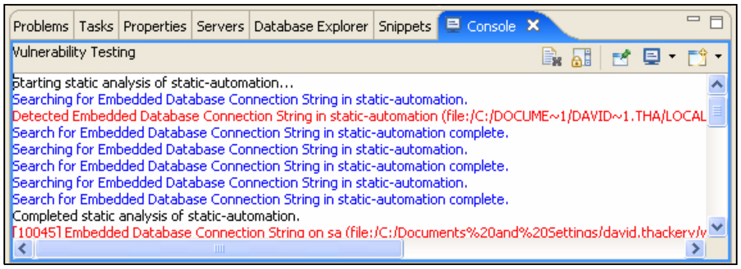
3 On the Choose Target dialog, select one of the following:

- **Test all reachable targets**: DevInspect will scan all URLs on all servers to which you are connected.
- **Test a specific target URL**: DevInspect will scan the site you enter or select. This option is valid only if you select **Enable Dynamic Analysis Engine** at the **Window > Preferences > Web Application Security** page.
- **Choose targets to test from the list**: DevInspect will scan the individual pages you select. If you have selected only **Enable Static Analysis Engine**, the list displays only the project (not its individual pages).

4 Click **OK**.

5 To view detailed information about the progress of the scan, open the **Console** view.

Note: Because several output consoles may be active, you may need to select **Vulnerability Testing** from the **Open Console** dropdown.



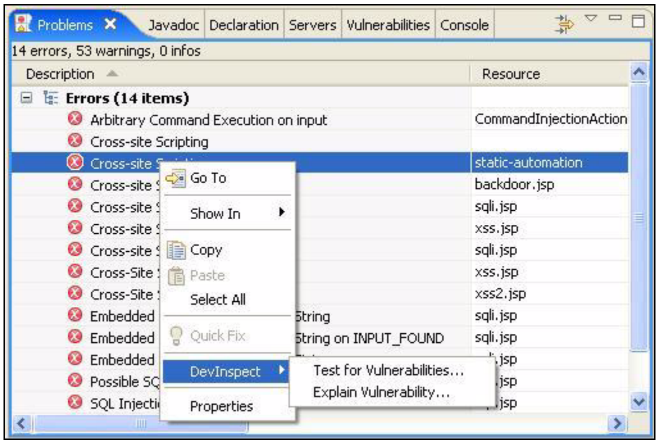
```
Starting static analysis of static-automation...
Searching for Embedded Database Connection String in static-automation.
Detected Embedded Database Connection String in static-automation (file:/C:/DOCUME~1/DAVID~1.THA/LOCAL
Search for Embedded Database Connection String in static-automation complete.
Searching for Embedded Database Connection String in static-automation.
Search for Embedded Database Connection String in static-automation complete.
Searching for Embedded Database Connection String in static-automation.
Search for Embedded Database Connection String in static-automation complete.
Completed static analysis of static-automation.
[10045] Embedded Database Connection String on sa (file:/C:/Documents%20and%20Settings/david.thackery/v
```

Status messages are color-coded according to the Console settings you selected as Workspace preferences (**Window > Preferences > Web Application Security > Console**). By default, the text announcing vulnerabilities appears red.

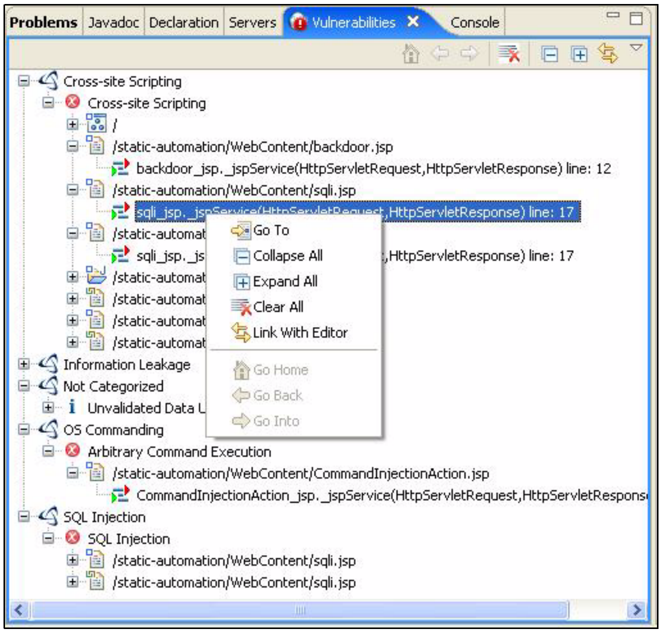
6 To view the list of detected errors, warnings, and informational items, open the **Problems** view. You may then right-click an item, select **DevInspect** from the context menu, and choose one of the following commands:

- **Test for Vulnerabilities**: Retests the selected URL or resource.

- **Explain Vulnerability**: Displays extensive and detailed information about the item, as well as recommendations for fixing vulnerabilities.



You can also use the Vulnerabilities view to detected vulnerabilities, which are grouped according to categories defined by the Web Application Security Consortium (WASC). When you expand a node (as illustrated below), the program lists all files in which the selected vulnerability was detected. Expand each of those nodes to see the vulnerability source (the point at which possibly tainted data enters the system) and its sink (the point at which the data is used). You can then double-click an item to open the source code in the editor (or right-click an item and select **Go To** from the context menu).


Problems
Javadoc
Declaration
Servers
Vulnerabilities
Console
Cross-site Scripting
Cross-site Scripting
/
/static-automation/WebContent/backdoor.jsp
backdoor_jsp._jspService(HttpServletRequest,HttpServletResponse) line: 12
/static-automation/WebContent/sqli.jsp
sqli_jsp._jspService(HttpServletRequest,HttpServletResponse) line: 17
/static-automat
sqli_jsp._js
,HttpServletResponse) line: 17
/static-automat
/static-automat
/static-automat
/static-automat
Information Leakage
Not Categorized
Unvalidated Data U
OS Commanding
Arbitrary Command Execution
/static-automation/WebContent/CommandInjectionAction.jsp
CommandInjectionAction_jsp._jspService(HttpServletRequest,HttpServletRespons
SQL Injection
SQL Injection
/static-automation/WebContent/sqli.jsp
/static-automation/WebContent/sqli.jsp
Go To
Collapse All
Expand All
Clear All
Link With Editor
Go Home
Go Back
Go Into

Distinctive icons identify the type of analysis that revealed the vulnerability, as well as the source and sink, as indicated by the following legend.

Icon	Description
	Vulnerability detected by dynamic analysis.
	Vulnerability detected by static analysis.
	Vulnerability detected by hybrid analysis.
	Vulnerability detected by correlated analysis.
	Line of code where possibly tainted data enters the system (source).
	Line of code where possibly tainted data is used (sink).
	Line of code where possibly tainted data enters the system and is used (sink = source).

Generate a Report

To create a report that encompasses all vulnerabilities detected during the scan:

1. Click **DevInspect > Generate Report**.
2. Select a scan, as designated by its timestamp.
3. Click **OK**.

Uninstall DevInspect

Follow the steps below to uninstall DevInspect.

1. Click **Help > Software Updates > Manage Configuration**.
2. Expand the **Eclipse** node.
3. Right-click **DevInspect for X** (where X is either WTP or Rational).
4. Select **Uninstall** from the shortcut menu.

4 Workbench Preferences

Overview

The preferences listed below represent the entire range of options that affect DevInspect functionality.

To configure these preferences:

1. Click **Window > Preferences**.
2. Click **Web Application Security**.
3. Configure the settings.
4. Expand the **Web Application Security** node to configure other settings.
5. Click **OK**.

The preferences are:

- Web Application Security
- AMP Connectivity
- Console
- Dynamic Analysis
  - Advanced
  - Allowed Hosts
  - Attack Exclusions
  - Authentication
  - Excluded Sessions
  - HTTP Parser
  - Requests
  - Scanner Proxy
  - Server Port Exclusions
- Licensing
- Policy
- Static Analysis

Web Application Security

The following preferences can be changed on the **Window > Preferences > Web Application Security** preference page.

Option	Description
Vulnerability warnings appear as errors	The scan engine classifies vulnerabilities as either Critical, High, Medium, Low, or Informational. To specify which of these classifications should be designated as errors on the Visual Studio Error List window, click the drop-down arrow and select one of the following: • **Never** (No vulnerabilities marked as errors) • **Critical** (Only critical vulnerabilities marked as errors) • **High or higher** (High and critical vulnerabilities marked as errors) • **Medium or higher** (Critical, high, and medium vulnerabilities marked as errors) • **Always** (All vulnerabilities marked as errors)
Logging Level	DevInspect logs activities at five different levels, ranging from most to least verbose in the following list: • **Debug**: Most verbose • **Information:** Informative events and those more severe • **Warning:** Warnings events and those more severe • **Error:** Errors and fatal events • **Fatal:** Fatal events only.
Save project vulnerability status	Select this option if you want to save the status of the Web site (as either "No Problems" or "Vulnerabilities") when you close.
Delete old scans on startup	If you select this option, DevInspect deletes all scan files (filename.spa) each time you launch Eclipse.
SmartUpdate when Eclipse starts	Select this option to check for updates automatically when starting Eclipse.
Enable static analysis engine	Allow DevInspect to perform static analysis.
Allow DevInspect to perform static analysis	Allow DevInspect to perform dynamic analysis.
DevInspect dialogs: Clear all "do not show again" settings	Certain DevInspect dialog boxes contain an option that allows you to prohibit the subsequent display of that dialog box. You can remove that prohibition by clicking **Clear**.

AMP Connectivity

The following preferences/properties can be changed on the **Window > Preferences > Web Application Security > AMP Connectivity** preference page.

You must have a license from HP that permits connecting with AMP.

Option	Description
Connect to AMP	Select this option to create a connection to the Assessment Management Platform (AMP).
AMP server	In the **Location** box, enter the URL or IP address of the AMP server.
Credentials	Enter a user name and password.
Options: Automatically transfer completed test results to AMP	If you select this option, DevInspect transmits the results to the AMP server as soon as the scan is completed. Otherwise, you must upload the file manually by selecting **DevInspect > Upload Results**.
Test Connection	Click this button to verify that the AMP server and credentials you specified are correct. This action does not connect your DevInspect application to AMP. It merely verifies that the path and credentials are valid.

Console

The following preferences can be changed on the **Window > Preferences > Web Application Security > Console** preference page.

Option	Description
Fixed Width Console	Set the width of console lines to a specific number of characters. Enabling this option allows you to specify the width. The default (and the minimum) is 80 characters.
Limit Console Output	Restrict the number of characters that the console can buffer. The default buffer size is 500,000 characters
Show Console Automatically	Display the Vulnerability Testing console when scanning a site.
Console Text Color Settings	Specify the types of messages displayed in the console and, if displayed, the colors for the text.

Dynamic Analysis

The following preferences/properties can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis** preference page.

Option	Description
Enable smart assessment	Smart Assessment is an “intelligent” feature that discovers the type of server that is hosting the Web site and checks for known vulnerabilities against that specific server type. For example, if you are scanning a site hosted on an IIS server, DevInspect will probe only for those vulnerabilities to which IIS is susceptible. It would not check for vulnerabilities that affect other servers, such as Apache or iPlanet.
Disable path truncation	Path truncation attacks are requests for known directories without file names. This may cause directory listings to be displayed. DevInspect truncates paths, looking for directory listings or unusual errors within each truncation. For example, if a link consists of http://www.site.com/folder1/folder2/file.asp, then truncating the path to look for http://www.site.com/folder1/folder2/ and http://www.site.com/folder1/ will cause the server to reveal directory contents or will cause unhandled exceptions.
Case-sensitive request and response handling	Select this option if the server at the target site is case-sensitive to URLs.
Compress response data	If you select this option, DevInspect saves each HTTP response in a compressed file on the database.
Maximum crawl-audit recursion depth	The Crawl Depth value determines how deeply DevInspect traverses the hierarchical levels of your Web site. If set to 1, DevInspect drills down one level; if set to 2, DevInspect drills down two levels; and so on. The maximum value is 1000.
Parse and execute JavaScript and VBscript	DevInspect always follows links defined by HTML (using the <a href> tag). You can also elect to force DevInspect to crawl links defined by JavaScript or VisualBasic script. By default, this option is enabled. Choosing to parse script can significantly increase the amount of time required for DevInspect to crawl a site. If you want to increase the speed at which DevInspect conducts a crawl while parsing script, choose not to display images via your browser settings.
Parse Flash	If you select this option, DevInspect will analyze Flash files, Adobe's vector graphics-based resizeable animation format.
Limit maximum single URL hits to	Use this field to limit the number of times a single link will be followed during a crawl. Sometimes, the configuration of a site will cause a crawl to loop endlessly through the same URL.

Option	Description
Limit maximum crawl folder depth to	This value determines how deeply DevInspect traverses the hierarchical levels of your Web site. If set to 1, DevInspect drills down one level; if set to 2, DevInspect drills down two levels; and so on. The maximum recursion level is 1,000.
Limit maximum crawl count to	This feature restricts the number of HTTP requests sent by the crawler and should be used only if you experience problems completing a scan of a large site.

Dynamic Analysis - Advanced

The following preferences/properties can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis > Advanced** preference page. You can:

- Create custom cookies
- Create custom headers
- Configure file-not-found detection
- Configure Web form settings and create a file containing Web form values

Custom Cookies

Use the Custom Cookies view to specify data that will be included with the Cookie header in request messages sent by DevInspect to the server when conducting a vulnerability assessment.

To add a custom cookie:

1. Click **New**.
2. Type or paste the cookie name/value pair in the **Add a new cookie** box.

   The format is: CookieName=Value.
3. Click **OK**.

To delete a custom cookie:

1. Select an item from the **Custom Cookies** list.
2. Click **Remove**.

To edit a custom cookie:

1. Select an item from the **Custom Cookies** list.
2. Click **Edit**.
3. Edit the cookie definition.
4. Click **OK**.

Custom Headers

Use the Custom Headers view to add headers to be included with each audit DevInspect performs. For example, you could add a header having a value such as “You are being attacked by Consultant ABC” that would be included with every request sent to your company's server when DevInspect is auditing that site. You can add multiple custom headers.

To add a custom header:

1. Click **New**.
2. Type or paste the header text in the **Add a new header** box.
3. Click **OK**.

To delete a custom header:

1. Select an item from the **Custom Headers** list.
2. Click **Remove**.

To edit a custom header:

1. Select an item from the **Custom Headers** list.
2. Click **Edit**.
3. Edit the header definition.
4. Click **OK**

File-Not-Found Detection

Option	**Description**
Maximum File-Not-Found Page Size	If the server returns a 404 status code and if the size of that response is larger then the size set in this configuration, DevInspect will falsely flag 404 pages as potentially hazardous. If this occurs, then increase the size and rescan the site.
Automatically Detect Custom File-Not-Found Pages	Some Web sites do not return a status “404 Not Found” when a client requests a resource that does not exist. Instead, they may return a status “200 OK” but the response contains a message that the file cannot be found. Select this check box if you want DevInspect to detect these “custom” file-not-found pages.
Matching Threshold Percentage	DevInspect attempts to detect custom file-not-found pages by sending requests for resources that cannot possibly exist on the server. It then compares each response and measures the amount of text that differs between the responses. For example, most messages of this type have the same content (such as “Sorry, the page you requested was not found”), with the possible exception being the name of the requested resource. If you select the Automatically detect check box, you can specify what percentage of the response content must be the same. The default is 90 percent.

Option	Description
Custom File-Not-Found Signatures	Use this view to add information about any custom 404 page notifications that your company uses. If your company has configured a different page to display when a 404 error occurs, add the information here. False positives can result in DevInspect from 404 pages that are unique to your site. **To add a custom signature:** 1 Click **New**. 2 In the **Add a new signature** box, type or paste a unique phrase that appears anywhere within the HTTP response message that you want to ignore. This area does not require a URL. Instead, insert a phrase that appears on each 404 page. For example, if your page states, “Sorry, this site does not exist,” enter that phrase. This will prevent a false positive from resulting whenever your company's unique 404 page is encountered. 3 Click **OK**. **To delete a custom signature:** 1 Select an item from the **Custom Headers** list. 2 Click **Remove**. **To edit a custom signature:** 1 Select an item from the **Custom File-Not-Found Signatures** list. 2 Click **Edit**. 3 Edit the signature text. 4 Click **OK**.

Web Forms

Most Web applications contain HTML forms composed of input controls (text boxes, buttons, drop-down lists, etc.). Users generally “complete” a form by modifying these input controls (such as entering text or checking boxes) before submitting the form to an agent for processing. Usually, this processing will lead the user to another page or section of the application. For example, after completing a logon form, the user will proceed to the application's beginning page. If DevInspect is to navigate through all possible links in the application, it must be able to submit appropriate data for each form.

Option	Description
Submit forms	Select this option if you want DevInspect to populate and submit forms encountered during an audit.
Maximum submissions per form	Enter the maximum number of times that any single form will be submitted during DevInspect’s scan.
Launch Web Form Editor	Click this button to start the Web Form Editor, which you can use to create or modify a list of input controls and associated values that will be submitted during a scan. See Recording Web Form Values on page 65 for instructions.

Dynamic Analysis - Allowed Hosts

The following preferences can be changed on the **Window** > **Preferences** > **Web Application Security** > **Dynamic Analysis** > **Allowed Hosts** preference page.

Use the Allowed Host settings to add domains to be crawled. If your Web presence uses multiple domains, add those domains here. For example, if your primary domain is “HPexample.com,” and if the domains “HPexample2.com” and “HPexample3.com” are part of your Web presence, add those domains here if you wanted to include them in the crawl or audit.

If you use different host servers with unique names (such as “HP.example.com” and “HP2.example.com”), add the additional addresses here to include them in your crawl or audit. Only the first server or domain added to the tree will be included in any DevInspect reports that you generate.

You can also use this feature to scan any domain containing the text you specify. For example, suppose you specify www.myco.com as the scan target and you enter “myco” as an allowed host. As DevInspect scans the target site, if it encounters a link to any URL containing “myco,” it will pursue that link and scan that site's server, repeating the process until all linked sites are scanned. For this hypothetical example, DevInspect would scan the following domains:

- www.myco.com:80
- contact.myco.com:80
- www.myco.com:443
- ethics.myco.com:80
- contact.myco.com:443
- wow.myco.com:80
- mycocorp.com:80
- www.interconnection.myco.com:80

To add an allowed host:

1. Click **New**.
2. Enter the host name in the **Add a new host** box.
3. Click **OK**.

To delete an allowed host:

1. Select an item from the **Allowed Hosts** list.
2. Click **Remove**.

To edit an allowed host:

1. Select an item from the **Allowed Hosts** list.
2. Click **Edit**.
3. Edit the allowed host definition.
4. Click **OK**.

Dynamic Analysis - Attack Exclusions

The following preferences/properties can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis > Attack Exclusions** preference page or the Project > Properties > **Web Application Security** > Attack Exclusions page.

Use this feature to prevent designated objects from being audited during a DevInspect vulnerability assessment. Format all entries as regular expressions.

Option	Description
Cookies	Use the Cookies view to identify cookies that should not be attacked.
Headers	Use the Headers view to identify HTTP headers that should not be attacked.
POST Data Parameters	Use the POST Data Parameters view to identify POST data parameters that should not be attacked.
Query Parameters	Use the Query Parameters view to identify query parameters that should not be attacked.

To add an excluded object to the list:

1. Select **Cookies**, **Headers**, **Post Data Parameters**, or **Query Parameters**.
2. Click **New**.
3. Enter a regular expression that represents the object.
4. Click **OK**.

To delete an excluded object from the list:

1. Select **Cookies**, **Headers**, **Post Data Parameters**, or **Query Parameters**.
2. Select an item from the list.
3. Click **Remove**.

To edit an excluded object:

1. Select **Cookies**, **Headers**, **Post Data Parameters**, or **Query Parameters**.
2. Select an item from the list.
3. Click **Edit**.
4. Edit the allowed host definition.
5. Click **OK**.

Dynamic Analysis - Authentication

The following preferences can be changed on the **Window** > **Preferences** > **Web Application Security** > **Dynamic Analysis** > **Authentication** preference page.

Option	Description
Authentication type	Select the type of server authentication required for the target Web site. • **None**: Authentication is not required. • **Automatic**: DevInspect determines the correct authentication scheme, based on the server's response. Although this method is reliable, it requires additional HTTP sessions each time DevInspect must establish a connection with the server. To minimize processing time, you should select this method only when you are not certain which authentication scheme is used by the target server. • **Basic**: The Web browser displays a dialog box for a user to enter a previously assigned user name and password. • **Digest**: The Windows Server 2003 operating system implements the Digest Authentication protocol as a security support provider (SSP), a dynamic-link library (DLL) that is supplied with the operating system. • **NTLM**: Use NTLM authentication for servers running IIS. If NTLM authentication is enabled, and DevInspect has to pass through a proxy server to submit its requests to the Web server, DevInspect may not be able to crawl or audit that Web site. Use caution when configuring DevInspect for scans of sites protected by NTLM. After scanning, you may want to disable the NTLM authentication settings to prevent any potential problem.
Credentials	If authentication other than None or Automatic is required, enter a user name and password. Leave both fields blank to use the credentials of the current user.
Client Certificates	Client certificate authentication allows users to present client certificates rather than entering a user name and password. **To use a certificate:** 1 Select the **Identify myself using the following certificate** check box. 2 Click **Browse**. 3 Select an entry from the **Certificate Store** list. 4 Select a certificate from the **Certificate** list. 5 Click **OK**.

Option	Description
Automatic Login	A macro is a recording of the HTTP requests and responses that are generated when you navigate through a Web site or application using the Web Macro Recorder tool. You can then instruct DevInspect to use this recording to enter your Web site and (optionally) navigate through your application. • Use a Login Macro This type of macro is used primarily for Web form authentication. It incorporates logic that will prevent DevInspect from terminating prematurely if it inadvertently logs out of your application. To use a login macro, select this check box, then click the drop-down arrow and select a macro from the list. Note: When recording this type of macro, be sure to select **Enable Check For Logout** and then specify the application's log-out signature • Use a Startup Macro This type of macro is used most often to focus on a particular subsection of the application. It specifies URLs that DevInspect will use to navigate to that area. It may also include login information, but does not contain logic that will prevent DevInspect from logging out of your application. To use a startup macro, select the check box, then click the drop-down arrow and select a macro from the list. To record a macro: 1 Click **Manage Macros**. 2 On the *Manage Macros* window, click **New**. 3 When the Web Macro Recorder appears, use it to create a macro. See Web Macro Recorder on page 56 for additional information.

Dynamic Analysis - Excluded Sessions

The following preferences can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis > Excluded Sessions** preference page.

Use this feature to prevent designated objects from being audited during a DevInspect vulnerability assessment. Format all entries as regular expressions.

Option	Description
Extensions	Use the **Extensions** view to identify file types that should be excluded. DevInspect will not request files of the type you specify. Note: The extensions you specify will also be replicated on the Content Types preference page (**Window > Preferences > General > Content Types**) in the category “Excluded from Vulnerability Testing.”
Hosts	Use the **Hosts** view to identify hosts that should be rejected. You must use a regular expression. Examples: • Enter a string such as logout. If that string is found in any portion of the URL, the URL will be excluded. Using the logout example, DevInspect will exclude URLs such as logout.asp or applogout.jsp. • If you enter “ /myApp / ” then DevInspect will exclude all resources in the myApp directory, such as http://www.test.me /myApp /filename.htm. • If you enter “ /W3SVC[0-9]*/ ” then DevInspect will exclude the following directories: http://www.test.me /W3SVC55/ http://www.test.me /W3SVC5/ http://www.test.me/W3SVC550/
URLs	Use the **URLs** view to identify URLs that should be rejected. DevInspect will not send any HTTP requests to the URLs you specify. For example, to ensure that you ignore and never send requests to any resource at Microsoft.com, enter the following regular expression. Microsoft\.com Notice that the period (or dot) is preceded by a backslash, indicating that the next character is special (i.e., it is not the character used in regular expressions to match any single character except a newline character). Note: You should always reject any URL that deals with logging off the site, since you don’t want to exit the application before the scan is completed.

Option	Description
MIME Types	Multipurpose Internet Mail Extensions (MIME) is a specification for formatting non-ASCII messages so they can be sent over the Internet. The Content-Type header indicates the type and subtype of the message content, as in the following example: Content-Type: text/plain The combination of type and subtype is generally called a MIME type (also known as Internet media type). Examples include: • text/html • image/jpeg • image/gif • audio/x-wave • audio/mpeg • video/mpeg • application/zip DevInspect will not process files associated with any MIME type you specify.

To add an object to the list:

1. Select **Extensions**, **Hosts**, **URLs**, or **MIME Types**.
2. Click **New**.
3. Enter a regular expression that represents the object.
4. Click **OK**.

To delete an object from the list:

1. Select **Extensions**, **Hosts**, **URLs**, or **MIME Types**.
2. Select an item from the list.
3. Click **Remove**.

To edit an object:

1. Select **Extensions**, **Hosts**, **URLs**, or **MIME Types**.
2. Select an item from the list.
3. Click **Edit**.
4. Edit the entry.
5. Click **OK**.

Dynamic Analysis - HTTP Parser

The following preferences can be changed on the **Window** > **Preferences** > **Web Application Security** > **Dynamic Analysis** > **HTTP Parser** preference page.

If your application uses URL rewriting or post data techniques to maintain state within a Web site, you must identify which parameters are used. For example, a PHP4 script can create a constant of the session ID named SID, which is available inside a session. By appending this to the end of a URL, the session ID becomes available to the next page. The actual URL might resemble the following:

.../page7.php?PHPSESSID=4725a759778d1be9bdb668a236f01e01

Because session IDs change with each connection, an HTTP request containing this URL would create an error when you tried to replay it. However, if you identify the parameter (PHPSESSID in this example), then DevInspect will replace its assigned value with the new session ID obtained from the server each time the connection is made.

Similarly, some state management techniques use POST data or URL expressions to pass information. For example, the HTTP message content may include userid=slbhkelvbkl73dhj. In this case, “userid” is the parameter you would identify.

Note: You need to identify parameters only when the application uses URL rewriting or posted data to manage state. It is not necessary when using cookies.

Option	Description
URL Expressions	One technique used by developers and Web servers to maintain session state is URL encoding. Instead of storing the session identifier as jsessionid in a cookie, the server places the session ID as a parameter in every HTML link in a Web page. In the following example, “1234567” is the session information: http://www.onlinestore.com/bikes/(1234567)/index.html The regular expression for identifying the parameter would be: /\(([\w\d]+\)/ **To create a regular expression that searches the URL for URL encoding:** 1 Select the **Determine State from URL Path** check box. 2 Click **New**. 3 In the **Add a new signature** box, enter a regular expression. 4 Click **OK**.

Option	Description

POST Data	The HTTP POST method is another alternative to using cookies for maintaining session state, as illustrated in the following example subroutine snippet.
	`<FORM METHOD=POST ACTION="DataEntry.asp"`
	`NAME=DataEntryForm>`
	`<P>Enter your name`
	`<INPUT TYPE="TEXT" NAME=FullName>`
	`<BR>Enter your credit card number`
	`<INPUT TYPE="TEXT" NAME=CreditCard>`
	`</P>`
	`<!-- Keeps track of the information by using`
	`the hidden HTML form variable Next Page. -->`
	`<INPUT TYPE="HIDDEN" NAME=NextPage VALUE=2>`
	`<INPUT TYPE="SUBMIT" VALUE="Next ->"`
	`NAME=NextButton>`
	`</FORM>`
	To specify POST data parameters:
	1 Click **New**.
	2 In the **Add a new POST data parameter** box, enter a parameter.
	3 Click **OK**.

Query Parameters	A query parameter is appended to the URL in the HTTP request as a variable. In the following example, the parameter is PHPSESSID:
	`…/`
	`page7.php?PHPSESSID=4725a759778d1be9b36f01e01`
	To specify query parameters:
	1 Click **New**.
	2 In the Add **a new query parameter** box, enter a parameter.
	3 Click **OK**.

Dynamic Analysis - Requests

The following preferences can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis > Requests** preference page.

Requestor Performance

DevInspect can be configured to send up to 100 concurrent HTTP requests before waiting for an HTTP response to the first request. Using a high maximum number of concurrent requests will increase the speed of a scan, but might also exhaust your system resources as well as those of the server you are scanning.

Option	Description
Connection Pooling	• Shared Connection Pool If you select this option, the crawler and the auditor use a common requestor when scanning a site, and each thread uses the same state, which is also shared by both modules. This is suitable for use when maintaining state is not a significant consideration. For this option, you must also specify the maximum number of concurrent requests. • Independent Connection Pool If you select this option, the crawler and auditor use separate requestors. Also, the auditor's requestor associates a state with each connection, rather than having all connections use the same state. You also specify the maximum number of connections that can be created for each requestor. Each requestor can be configured to send up to 25 concurrent HTTP requests before waiting for an HTTP response to the first request
Error Handling	• Limit maximum response size to Select this option to limit the size of accepted server responses, and then specify the maximum size (in bytes). • Timeout Specify how long DevInspect will wait for an HTTP response from the server. If this threshold is exceeded, DevInspect resubmits the request until a response is received or until reaching the retry count that you specify in the **Maximum Retries** box. If it then receives no response, DevInspect logs the timeout and issues the first HTTP request in the next attack series. • Maximum Retries Specify how many times DevInspect will resubmit an HTTP request after receiving a "failed" response (which is defined as any socket error or request timeout).

Connectivity

There may be occasions during a scan when a Web server fails or becomes too busy to respond in a timely manner. You can specify the conditions under which DevInspect should terminate a scan.

Option	Description
Stop Scan if Loss of Connectivity Detected	If you select this option, DevInspect will terminate the scan whenever it loses the connection to the server, as determined by the parameters you select below:.
Consecutive “single host” retry failures	If you select this option, enter the number of consecutive timeouts permitted from one specific server. If DevInspect exceeds this limit, it will terminate the scan.
Consecutive “any host” retry failures	If you select this option, enter the total number of consecutive timeouts permitted from all hosts. If DevInspect exceeds this limit, it will terminate the scan.
Nonconsecutive “single host” retry failures	If you select this option, enter the total number of nonconsecutive timeouts permitted from a single host. If DevInspect exceeds this limit, it will terminate the scan.
Nonconsecutive “any host” request failures	If you select this option, enter the total number of nonconsecutive timeouts permitted from all hosts. If DevInspect exceeds this limit, it will terminate the scan.

If you notice numerous entries on the console showing requests timing out, you should reduce the maximum number of connections. While most servers can handle a large number of requests, servers in development environments sometimes have limitations on their licensing that allow only five or fewer users to be connected at a single time. In such cases, you should reduce the maximum concurrent request count to be less than 5. Failing to do so may mean that DevInspect does not accurately crawl or audit the site because requests are being rejected by the server.

Dynamic Analysis - Scanner Proxy

The following preferences can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis > Scanner Proxy** preference page.

To scan a target site through a proxy server, select **Enable proxy** and provide the requested information.

Option	Description
Proxy host	Enter the URL or IP address of the proxy server
Proxy port	Enter the port number used by the proxy server
Use SOCKS	Select this option to use SOCKS4 protocol for handling TCP traffic through a proxy server
Enable proxy authentication	If your proxy server requires authentication, enter the qualifying user name and password

Dynamic Analysis - Server Port Exclusions

The following preferences can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis > Server Port Exclusions** preference page.

Use this feature to specify ports that DevInspect should not scan.

To specify a port that should be excluded:

1 Click **New**.

2 On the Excluded Ports window, enter a port number.

3 Click **OK**.

To remove an entry from the list of excluded ports:

1 Select a port.

2 Click **Remove**.

Licensing

The following Window preferences can be changed on the **Window > Preferences > Web Application Security > Licensing** preference page.

Use this form to enter an activation (license) key and activate your installation of DevInspect.

If you have not purchased DevInspect, you can request an activation key that will be valid for a limited time and will allow you to scan the Hewlett-Packard test site.

To activate your installation:

1 In the **Activation Key** area, enter the activation (license) key provided to your organization by Hewlett-Packard, developers of DevInspect.

2 In the **License Service URL** box, enter the fully qualified URL of the licensing service. The default is https://licenseservice.spidynamics.com/.

3 Click **Update License**.

To request a trial license:

1 Complete all fields in the Personal Information section.

2 Click **Request Trial**.

Upon receiving your request, Hewlett-Packard will send you an e-mail containing an activation key. Return to this Licensing Options page and follow the instructions above for activating your installation.

Policy

The following preferences can be changed on the **Window > Preferences > Web Application Security > Policy** preference page.

A policy is simply a checklist of threats that may exist in your application. You can select all of them or a subset. When you test your application for vulnerabilities, DevInspect launches attack agents and audit engines specifically designed to search for these threats.

Policy properties are grouped into five categories, as defined by the Web Application Security Consortium. Each category contains one or more threat classifications. Expand each node and select the vulnerabilities for which DevInspect will test.

- **Authentication**: This category of attacks focuses on a Web site's method of validating the identity of a user, service, or application.
- **Authorization**: This category of attacks focuses on a Web site's method of determining if a user, service, or application has the necessary permissions to perform a requested action.
- **Client-Side Attacks**: This category of attacks focuses on the trust established between a Web site and its users. When a user visits a Web site, trust is established between the two parties. The user expects valid content and does not expect to be attacked by the Web site.
- **Command Execution**: This category of attacks focuses on the alteration of user-supplied to execute remote commands on the Web site.
- **Information Disclosure**: This category of attacks focuses on ways of getting a Web site to reveal system-specific information, such as the type of software, version numbers, and patches installed.

Static Analysis

The following preferences can be changed on the **Window > Preferences > Web Application Security > Static Analysis** preference page.

DevInspect conducts static analysis by examining the bytecode in class files, tracing the data flow from its source (the point at which possibly tainted data enters the system) to its sink (the point at which the data is used). If the analysis does not encounter a user-defined method for sanitizing the data, then DevInspect considers the process vulnerable and assigns an appropriate threat class defined by the Web Application Security Consortium (WASC).

Scrubbers

A scrubber is a method (written by the developer or provided by a third party) that examines user input and prohibits the introduction of data that could otherwise threaten the security of a Web site or application. These threats include SQL injection, cross-site scripting, information leakage, command injection, and various other Web application security attack classes.

You must identify the scrubbers that you have incorporated into your project.

1. Click **Add**.

   The *Scrubbers* window opens.

2. In the **Select method** box (at the top), type the first few letters of the method name.
3. Select one or more methods.
4. Click **OK**.

Note: You can also add a scrubber in any perspective that displays methods. Simply select a method, then click the **DevInspect** menu and select **Add to scrubbers** (or right-click a method and, from the context menu, select **DevInspect > Add to scrubbers**).

Exclusions

There is no benefit to applying static analysis to all packages, especially the Swing toolkit or Abstract Windows Toolkit (AWT) or those packages dealing with images or audio. These are excluded by default. You may choose to exclude other packages, as well.

To exclude a package from static analysis:

1. Click **Add**.

   The *Exclusions* window lists all packages in the Java runtime that are not currently excluded.

2. (optional) In the **Select package** box (at the top), type the first few letters of a package name.

3. Select one or more packages.

4. Click **OK**.

5 Project Properties

Overview

The properties listed below constitute a subset of the global workspace (Window) preferences, allowing you to easily apply different settings to a specific project.

To configure these properties:

1. Select a project in the Project Explorer
2. Click **Project > Properties**.
3. Expand the **Web Application Security** node.
4. Select a property listed below.
   - Advanced
   - Allowed Hosts
   - Attack Exclusions
   - Authentication
   - Excluded Sessions
   - Requests
5. To use the same settings configured for the workspace, select **Use workspace settings**.
6. To configure project settings that differ from the workspace settings:
   a. Select **Use project settings**.
   b. Configure the setting. For detailed descriptions, refer to the corresponding sections in this chapter.
7. Click **OK**.

Advanced

The following preferences/properties can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis > Advanced** preference page. You can:

- Create custom cookies
- Create custom headers
- Configure file-not-found detection
- Configure Web form settings and create a file containing Web form values

Custom Cookies

Use the Custom Cookies view to specify data that will be included with the Cookie header in request messages sent by DevInspect to the server when conducting a vulnerability assessment.

To add a custom cookie:

1. Click **New.**
2. Type or paste the cookie name/value pair in the **Add a new cookie** box.

   The format is: CookieName=Value.
3. Click **OK**.

To delete a custom cookie:

1. Select an item from the **Custom Cookies** list.
2. Click **Remove**.

To edit a custom cookie:

1. Select an item from the **Custom Cookies** list.
2. Click **Edit**.
3. Edit the cookie definition.
4. Click **OK**.

Custom Headers

Use the Custom Headers view to add headers to be included with each audit DevInspect performs. For example, you could add a header having a value such as “You are being attacked by Consultant ABC” that would be included with every request sent to your company's server when DevInspect is auditing that site. You can add multiple custom headers.

To add a custom header:

1. Click **New.**
2. Type or paste the header text in the **Add a new header** box.
3. Click **OK**.

To delete a custom header:

1. Select an item from the **Custom Headers** list.
2. Click **Remove**.

To edit a custom header:

1. Select an item from the **Custom Headers** list.
2. Click **Edit**.
3. Edit the header definition.
4. Click **OK**

File-Not-Found Detection

Option	Description
Maximum File-Not-Found Page Size	If the server returns a 404 status code and if the size of that response is larger then the size set in this configuration, DevInspect will falsely flag 404 pages as potentially hazardous. If this occurs, then increase the size and rescan the site.
Automatically Detect Custom File-Not-Found Pages	Some Web sites do not return a status “404 Not Found” when a client requests a resource that does not exist. Instead, they may return a status “200 OK” but the response contains a message that the file cannot be found. Select this check box if you want DevInspect to detect these “custom” file-not-found pages.
Matching Threshold Percentage	DevInspect attempts to detect custom file-not-found pages by sending requests for resources that cannot possibly exist on the server. It then compares each response and measures the amount of text that differs between the responses. For example, most messages of this type have the same content (such as “Sorry, the page you requested was not found”), with the possible exception being the name of the requested resource. If you select the Automatically detect check box, you can specify what percentage of the response content must be the same. The default is 90 percent.
Custom File-Not-Found Signatures	Use this view to add information about any custom 404 page notifications that your company uses. If your company has configured a different page to display when a 404 error occurs, add the information here. False positives can result in DevInspect from 404 pages that are unique to your site. **To add a custom signature:** 1 Click **New**. 2 In the **Add a new signature** box, type or paste a unique phrase that appears anywhere within the HTTP response message that you want to ignore. This area does not require a URL. Instead, insert a phrase that appears on each 404 page. For example, if your page states, “Sorry, this site does not exist,” enter that phrase. This will prevent a false positive from resulting whenever your company's unique 404 page is encountered. 3 Click **OK**. **To delete a custom signature:** 1 Select an item from the **Custom Headers** list. 2 Click **Remove**. **To edit a custom signature:** 1 Select an item from the **Custom File-Not-Found Signatures** list. 2 Click **Edit**. 3 Edit the signature text. 4 Click **OK**.

Web Forms

Most Web applications contain HTML forms composed of input controls (text boxes, buttons, drop-down lists, etc.). Users generally “complete” a form by modifying these input controls (such as entering text or checking boxes) before submitting the form to an agent for processing. Usually, this processing will lead the user to another page or section of the application. For example, after completing a logon form, the user will proceed to the application's beginning page. If DevInspect is to navigate through all possible links in the application, it must be able to submit appropriate data for each form.

Option	Description
Submit forms	Select this option if you want DevInspect to populate and submit forms encountered during an audit.
Maximum submissions per form	Enter the maximum number of times that any single form will be submitted during DevInspect’s scan.
Launch Web Form Editor	Click this button to start the Web Form Editor, which you can use to create or modify a list of input controls and associated values that will be submitted during a scan. See Recording Web Form Values on page 65 for instructions.

Allowed Hosts

The following preferences can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis > Allowed Hosts** preference page.

Use the Allowed Host settings to add domains to be crawled. If your Web presence uses multiple domains, add those domains here. For example, if your primary domain is “HPexample.com,” and if the domains “HPexample2.com” and “HPexample3.com” are part of your Web presence, add those domains here if you wanted to include them in the crawl or audit.

If you use different host servers with unique names (such as “HP.example.com” and “HP2.example.com”), add the additional addresses here to include them in your crawl or audit. Only the first server or domain added to the tree will be included in any DevInspect reports that you generate.

You can also use this feature to scan any domain containing the text you specify. For example, suppose you specify www.myco.com as the scan target and you enter “myco” as an allowed host. As DevInspect scans the target site, if it encounters a link to any URL containing “myco,” it will pursue that link and scan that site's server, repeating the process until all linked sites are scanned. For this hypothetical example, DevInspect would scan the following domains:

- www.myco.com:80
- contact.myco.com:80
- www.myco.com:443
- ethics.myco.com:80
- contact.myco.com:443
- wow.myco.com:80

- mycocorp.com:80
- www.interconnection.myco.com:80

To add an allowed host:

1. Click **New**.
2. Enter the host name in the **Add a new host** box.
3. Click **OK**.

To delete an allowed host:

1. Select an item from the **Allowed Hosts** list.
2. Click **Remove**.

To edit an allowed host:

1. Select an item from the **Allowed Hosts** list.
2. Click **Edit**.
3. Edit the allowed host definition.
4. Click **OK**.

Attack Exclusions

The following preferences/properties can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis > Attack Exclusions** preference page or the Project > Properties > **Web Application Security** > Attack Exclusions page.

Use this feature to prevent designated objects from being audited during a DevInspect vulnerability assessment. Format all entries as regular expressions.

Option	Description
Cookies	Use the Cookies view to identify cookies that should not be attacked.
Headers	Use the Headers view to identify HTTP headers that should not be attacked.
POST Data Parameters	Use the POST Data Parameters view to identify POST data parameters that should not be attacked.
Query Parameters	Use the Query Parameters view to identify query parameters that should not be attacked.

To add an excluded object to the list:

1. Select **Cookies**, **Headers**, **Post Data Parameters**, or **Query Parameters**.
2. Click **New**.
3. Enter a regular expression that represents the object.
4. Click **OK**.

To delete an excluded object from the list:

1. Select **Cookies**, **Headers**, **Post Data Parameters**, or **Query Parameters**.
2. Select an item from the list.
3. Click **Remove**.

To edit an excluded object:

1. Select **Cookies**, **Headers**, **Post Data Parameters**, or **Query Parameters**.
2. Select an item from the list.
3. Click **Edit**.
4. Edit the allowed host definition.
5. Click **OK**.

Authentication

The following preferences can be changed on the **Window > Preferences > Web Application Security > Dynamic Analysis > Authentication** preference page.

Option	Description
Authentication type	Select the type of server authentication required for the target Web site. • **None**: Authentication is not required. • **Automatic**: DevInspect determines the correct authentication scheme, based on the server's response. Although this method is reliable, it requires additional HTTP sessions each time DevInspect must establish a connection with the server. To minimize processing time, you should select this method only when you are not certain which authentication scheme is used by the target server. • **Basic**: The Web browser displays a dialog box for a user to enter a previously assigned user name and password. • **Digest**: The Windows Server 2003 operating system implements the Digest Authentication protocol as a security support provider (SSP), a dynamic-link library (DLL) that is supplied with the operating system. • **NTLM**: Use NTLM authentication for servers running IIS. If NTLM authentication is enabled, and DevInspect has to pass through a proxy server to submit its requests to the Web server, DevInspect may not be able to crawl or audit that Web site. Use caution when configuring DevInspect for scans of sites protected by NTLM. After scanning, you may want to disable the NTLM authentication settings to prevent any potential problem.
Credentials	If authentication other than None or Automatic is required, enter a user name and password. Leave both fields blank to use the credentials of the current user.

Option	Description
Client Certificates	Client certificate authentication allows users to present client certificates rather than entering a user name and password. **To use a certificate:** 1 Select the **Identify myself using the following certificate** check box. 2 Click **Browse**. 3 Select an entry from the **Certificate Store** list. 4 Select a certificate from the **Certificate** list. 5 Click **OK**.
Automatic Login	A macro is a recording of the HTTP requests and responses that are generated when you navigate through a Web site or application using the Web Macro Recorder tool. You can then instruct DevInspect to use this recording to enter your Web site and (optionally) navigate through your application. • Use a Login Macro This type of macro is used primarily for Web form authentication. It incorporates logic that will prevent DevInspect from terminating prematurely if it inadvertently logs out of your application. To use a login macro, select this check box, then click the drop-down arrow and select a macro from the list. Note: When recording this type of macro, be sure to select **Enable Check For Logout** and then specify the application's log-out signature • Use Separate Startup Macro This type of macro is used most often to focus on a particular subsection of the application. It specifies URLs that DevInspect will use to navigate to that area. It may also include login information, but does not contain logic that will prevent DevInspect from logging out of your application. To use a startup macro, select the check box, then click the drop-down arrow and select a macro from the list. To record a macro: 1 Click **Manage Macros**. 2 On the *Manage Macros* window, click **New**. 3 When the Web Macro Recorder appears, use it to create a macro. See Web Macro Recorder on page 56 for additional information.

Excluded Sessions

The following preferences can be changed on the **Window** > **Preferences** > **Web Application Security** > **Dynamic Analysis** > **Excluded Sessions** preference page.

Use this feature to prevent designated objects from being audited during a DevInspect vulnerability assessment. Format all entries as regular expressions.

Option	Description
Extensions	Use the **Extensions** view to identify file types that should be excluded. DevInspect will not request files of the type you specify. Note: The extensions you specify will also be replicated on the Content Types preference page (**Window** > **Preferences** > **General** > **Content Types**) in the category "Excluded from Vulnerability Testing."
Hosts	Use the **Hosts** view to identify hosts that should be rejected. You must use a regular expression. Examples: • Enter a string such as logout. If that string is found in any portion of the URL, the URL will be excluded. Using the logout example, DevInspect will exclude URLs such as logout.asp or applogout.jsp. • If you enter " /myApp / " then DevInspect will exclude all resources in the myApp directory, such as http:// www.test.me /myApp /filename.htm. • If you enter " /W3SVC[0-9]*/ " then DevInspect will exclude the following directories: http://www.test.me /W3SVC55/ http://www.test.me /W3SVC5/ http://www.test.me/W3SVC550/
URLs	Use the **URLs** view to identify URLs that should be rejected. DevInspect will not send any HTTP requests to the URLs you specify. For example, to ensure that you ignore and never send requests to any resource at Microsoft.com, enter the following regular expression. Microsoft\.com Notice that the period (or dot) is preceded by a backslash, indicating that the next character is special (i.e., it is not the character used in regular expressions to match any single character except a newline character). Note: You should always reject any URL that deals with logging off the site, since you don't want to exit the application before the scan is completed.

Option	**Description**
MIME Types	Multipurpose Internet Mail Extensions (MIME) is a specification for formatting non-ASCII messages so they can be sent over the Internet. The Content-Type header indicates the type and subtype of the message content, as in the following example: Content-Type: text/plain The combination of type and subtype is generally called a MIME type (also known as Internet media type). Examples include: • text/html • image/jpeg • image/gif • audio/x-wave • audio/mpeg • video/mpeg • application/zip DevInspect will not process files associated with any MIME type you specify.

To add an object to the list:

1. Select **Extensions**, **Hosts**, **URLs**, or **MIME Types**.
2. Click **New**.
3. Enter a regular expression that represents the object.
4. Click **OK**.

To delete an object from the list:

1. Select **Extensions**, **Hosts**, **URLs**, or **MIME Types**.
2. Select an item from the list.
3. Click **Remove**.

To edit an object:

1. Select **Extensions**, **Hosts**, **URLs**, or **MIME Types**.
2. Select an item from the list.
3. Click **Edit**.
4. Edit the entry.
5. Click **OK**.

Requests

The following preferences can be changed on the **Window** > **Preferences** > **Web Application Security** > **Dynamic Analysis** > **Requests** preference page.

Requestor Performance

DevInspect can be configured to send up to 100 concurrent HTTP requests before waiting for an HTTP response to the first request. Using a high maximum number of concurrent requests will increase the speed of a scan, but might also exhaust your system resources as well as those of the server you are scanning.

Option	Description
Connection Pooling	• Shared Connection Pool If you select this option, the crawler and the auditor use a common requestor when scanning a site, and each thread uses the same state, which is also shared by both modules. This is suitable for use when maintaining state is not a significant consideration. For this option, you must also specify the maximum number of concurrent requests. • Independent Connection Pool If you select this option, the crawler and auditor use separate requestors. Also, the auditor's requestor associates a state with each connection, rather than having all connections use the same state. You also specify the maximum number of connections that can be created for each requestor. Each requestor can be configured to send up to 25 concurrent HTTP requests before waiting for an HTTP response to the first request
Error Handling	• Limit maximum response size to Select this option to limit the size of accepted server responses, and then specify the maximum size (in bytes). • Timeout Specify how long DevInspect will wait for an HTTP response from the server. If this threshold is exceeded, DevInspect resubmits the request until a response is received or until reaching the retry count that you specify in the **Maximum Retries** box. If it then receives no response, DevInspect logs the timeout and issues the first HTTP request in the next attack series. • Maximum Retries Specify how many times DevInspect will resubmit an HTTP request after receiving a “failed” response (which is defined as any socket error or request timeout).

Connectivity

There may be occasions during a scan when a Web server fails or becomes too busy to respond in a timely manner. You can specify the conditions under which DevInspect should terminate a scan.

Option	Description
Stop Scan if Loss of Connectivity Detected	If you select this option, DevInspect will terminate the scan whenever it loses the connection to the server, as determined by the parameters you select below:.
Consecutive "single host" retry failures	If you select this option, enter the number of consecutive timeouts permitted from one specific server. If DevInspect exceeds this limit, it will terminate the scan.
Consecutive "any host" retry failures	If you select this option, enter the total number of consecutive timeouts permitted from all hosts. If DevInspect exceeds this limit, it will terminate the scan.
Nonconsecutive "single host" retry failures	If you select this option, enter the total number of nonconsecutive timeouts permitted from a single host. If DevInspect exceeds this limit, it will terminate the scan.
Nonconsecutive "any host" request failures	If you select this option, enter the total number of nonconsecutive timeouts permitted from all hosts. If DevInspect exceeds this limit, it will terminate the scan.

If you notice numerous entries on the console showing requests timing out, you should reduce the maximum number of connections. While most servers can handle a large number of requests, servers in development environments sometimes have limitations on their licensing that allow only five or fewer users to be connected at a single time. In such cases, you should reduce the maximum concurrent request count to be less than 5. Failing to do so may mean that DevInspect does not accurately crawl or audit the site because requests are being rejected by the server.

A DevInspect Tools

Overview

The DevInspect platform offers a robust set of testing tools. These are:

- Web Macro Recorder
- Web Form Editor
- HTTP Editor
- Web Proxy

Web Macro Recorder

This tool allows you to create or edit a macro that DevInspect will use to enter your Web site and (optionally) navigate through your application.

Subsequently, when you instruct DevInspect to use the macro, the scanner will crawl and audit only the target URL and those servers specified as allowed hosts (in the scanner's option settings). This allows you to pass through a single sign-on application such as SiteMinder or MS Passport without assessing its server.

Any activity you record in a macro will override the DevInspect settings. For example, if you specify a URL in the Excluded URL setting, and then you actually navigate to that URL when creating a macro, DevInspect will ignore the exclusion when it crawls and audits the site.

Web Macro Recorder Menus

The Web Macro Recorder menu bar contains the menus described in the following sections.

File Menu

The **File** menu contains the following commands:

- **New** - Deletes all information from previous sessions.
- **Open** - Loads a file containing a Web macro.
- **Save** - Saves the Web macro.
- **Save As** - Saves the Web macro.
- **Exit** - Closes the Web Macro Recorder.

Edit Menu

The **Edit** menu contains the following commands:

- **Cut** - Deletes selected text and saves it to the clipboard.
- **Copy** - Saves the selected text to the clipboard.
- **Paste** - Inserts text from the clipboard
- **Edit with HTTP Editor** - Launches the HTTP Editor, allowing you to edit the HTTP request for the selected session.
- **Delete Session** - Removes the selected session from the macro.
- **Record** - Resumes recording after pausing.
- **Pause** - Halts the recording; to resume, click **Record**.
- **Find** - Displays a window that allows you to search for text that you specify.
- **Settings** - Allows you to configure parameters for the Web Macro Recorder.

View Menu

The **View** menu contains the following commands:

- **Launch Browser** - Launches the default browser.

- **HTTP Editor** - Launches the HTTP Editor.
- **Toolbars** - Displays or hides the Recorder toolbar.
- **Filter Rules** - Excludes the recording of any page containing the resource type that you select.
- **Advanced** - Displays panes for viewing the HTTP request and response messages.

Help Menu

The **Help** menu contains the following commands:

- **Web Macro Recorder Help** - Opens the Help file with the **Contents** tab active.
- **Index** - Opens the Help file with the **Index** tab active.
- **Search** - Opens the Help file with the **Search** tab active.
- **About Web Macro Recorder -** Displays information about the Web Macro Recorder.

Creating a Macro

Follow the steps below to create a macro:

1. Start the Web Macro Recorder.
2. Select **New** from the **File** menu (or click the New icon on the toolbar).
3. Click **Launch Browser.**
4. Using the browser's Address bar, enter or select a URL.
5. You can exclude the recording of responses containing certain objects:
   a. Select **Filter Rules** from the Macro Recorder's **View** menu.
   b. Select the check box associated with a resource type to exclude the recording of any page containing the selected type. Clear the check box to remove the prohibition.
6. If your application uses Web form authentication, enter a valid user name and password, and then submit the data (usually by clicking a button such as **Log On**, **Go**, **Submit**, etc.).

   Although the primary purpose for recording an authentication macro is to enable DevInspect to log on to your site, you may continue navigating through your application. The Web Macro Recorder will continue recording all the HTTP traffic.
7. When finished, click the **Pause** button (on the recorder) or close the browser.
8. (Optional) Create a logout signature.

   If DevInspect encounters a hyperlink to another resource, it will navigate to that URL and continue its assessment. If it follows a link to a logoff page (or if the server automatically “logs off” a client after a certain number of minutes), the scanner will not be able to visit additional resources where the client is required to be logged on. When this inadvertent logoff occurs, the scanner must be able to log on again without user intervention. This process hinges on the scanner’s ability to recognize when it is no longer logged on.
   a. From the list of visited URLs, select one that you accessed after logging on. Do not select the URL where you actually logged on.

b Click **Enable Check for Logout**.

The Web Macro Recorder attempts to create a regular expression that it will use to search server responses for indications that the client has logged off.

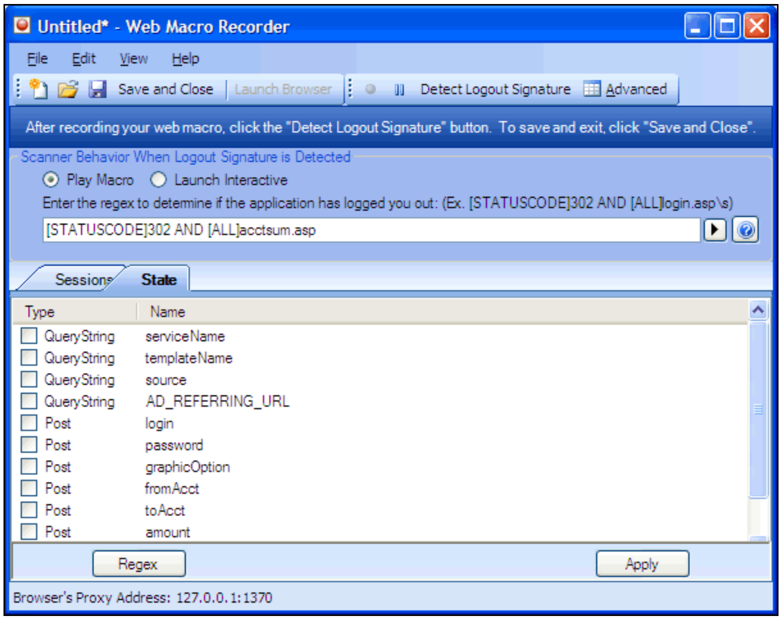
c If the Macro Recorder is unable to determine the logoff signature, try clicking on other URLs.

d If the Macro Recorder is still unable to determine a signature, you can manually enter one. In the **Regex** box, type a regular expression that identifies a unique text or phrase that occurs in the server's HTTP response when a user logs off or when a user who is not logged on requests access to a protected URL. For example, if your server returns a message such as “Have a nice day” when a user logs off your application, then enter “Have\sa\snice\sday” as the regular expression (“\s” is used in regular expressions to designate a space). A scanner can also detect that it has logged off if the server sends a specific message in response to the scanner's attempt to access a password-protected URL. For example, the server may respond with a status code of “302 Object moved.” In this case, “[STATUSCODE]302 AND [ALL] http://login.myco.com/config/mail?” might be a typical regular expression. See Tips for Using Regular Expressions on page 62 for more information.

9 Specify which action the scanner should take if it detects that it has logged out of the application:

- Run the macro, or
- Launch the Interactive mode (which will allow you to manually log back in)
- Launch the Interactive mode (which will allow you to manually log back in)

10 (Optional) If your application uses URL rewriting or post data techniques to maintain state within a Web site, select the **State** tab.



After recording your web macro, click the "Detect Logout Signature" button. To save and exit, click "Save and Close".

Scanner Behavior When Logout Signature is Detected

● Play Macro ○ Launch Interactive

Enter the regex to determine if the application has logged you out: (Ex. [STATUSCODE]302 AND [ALL]login.asp\s)

[STATUSCODE]302 AND [ALL]acctsum.asp

Type	Name
□ QueryString	serviceName
□ QueryString	templateName
□ QueryString	source
□ QueryString	AD_REFERRING_URL
□ Post	login
□ Post	password
□ Post	graphicOption
□ Post	fromAcct
□ Post	toAcct
□ Post	amount

Browser's Proxy Address: 127.0.0.1:1370

You must identify which parameters are used for state management. For example, a PHP4 script can create a constant of the session ID named SID, which is available inside a session. By appending this to the end of a URL, the session ID becomes available to the next page. The actual URL might look something like the following:

.../page7.php?PHPSESSID=4725a759778d1be9bdb668a236f01e01

Because session IDs change with each connection, a recorded macro containing this URL would create an error when you tried to replay it. However, if you identify the parameter (PHPSESSID in this example), then the scanner will replace its assigned value with the new session ID obtained from the server each time the connection is made.

Similarly, some state management techniques use post data to pass information. For example, the HTTP message content may include userid=slbhkelvbkl73dhj. In this case, “userid” is the parameter you would identify to the Web Macro Recorder.

Note: You need to identify parameters only when the application uses URL rewriting or posted data to manage state. It is not necessary when using cookies.

The Web Macro Recorder can identify potential parameters if they occur as posted data or if they exist within the query string of a URL. However, if your application embeds session data in the URL as extended path information, you must provide a regular expression to identify it. In the following example, “1234567” is the session information:

http://www.onlinestore.com/bikes/(1234567)/index.html

The regular expression for identifying the parameter would be:

```
/\([\w\d]+\)/
```

a To enter a regular expression, click **Regex** and then use the Regular Expression Editor to create an expression.

b To identify parameters, select a parameter in the **Type/Name** list (such as “login” in the preceding illustration).

c Click **Apply**.

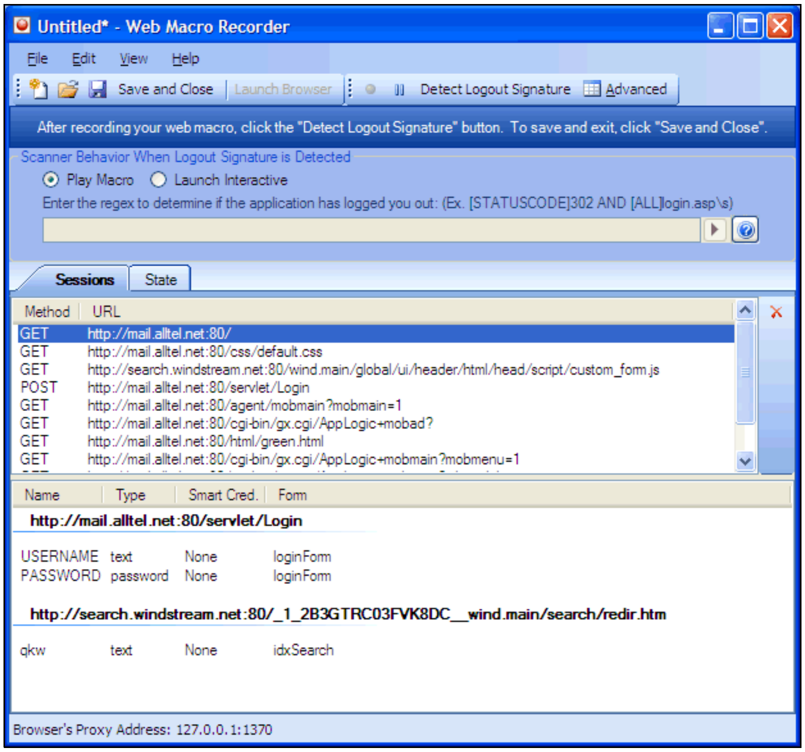
11 To save the macro, select **Save** or **Save As** from the **File** men.

Editing a Macro

As you navigate through the target Web site, the Web Macro Recorder transcribes each session, displaying on the **Sessions** tab the method and URL associated with each HTTP request sent to the server.

1 Select a request in the **Method/URL** list.

If the associated HTTP response includes “text” or “password” input controls, their name and type are displayed in the lower pane.



After recording your web macro, click the "Detect Logout Signature" button. To save and exit, click "Save and Close".

Scanner Behavior When Logout Signature is Detected

● Play Macro ○ Launch Interactive

Enter the regex to determine if the application has logged you out: (Ex. [STATUSCODE]302 AND [ALL]login.asp\s)

Method	URL
GET	http://mail.alltel.net:80/
GET	http://mail.alltel.net:80/css/default.css
GET	http://search.windstream.net:80/wind.main/global/ui/header/html/head/script/custom_form.js
POST	http://mail.alltel.net:80/servlet/Login
GET	http://mail.alltel.net:80/agent/mobmain?mobmain=1
GET	http://mail.alltel.net:80/cgi-bin/gx.cgi/AppLogic+mobad?
GET	http://mail.alltel.net:80/html/green.html
GET	http://mail.alltel.net:80/cgi-bin/gx.cgi/AppLogic+mobmain?mobmenu=1

Name	Type	Smart Cred.	Form
http://mail.alltel.net:80/servlet/Login			
USERNAME	text	None	loginForm
PASSWORD	password	None	loginForm
http://search.windstream.net:80/_1_2B3GTRC03FVK8DC__wind.main/search/redir.htm			
qkw	text	None	idxSearch

Browser's Proxy Address: 127.0.0.1:1370

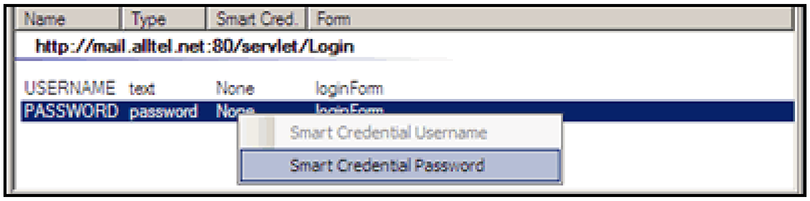
In this example, the form and the controls were rendered by the following HTML statements:

```
<form name="loginForm" action="/servlet/Login" method="POST">

<input type="text" size="16" name="USERNAME" value="">

<input type="password" size="16" name="PASSWORD">
```

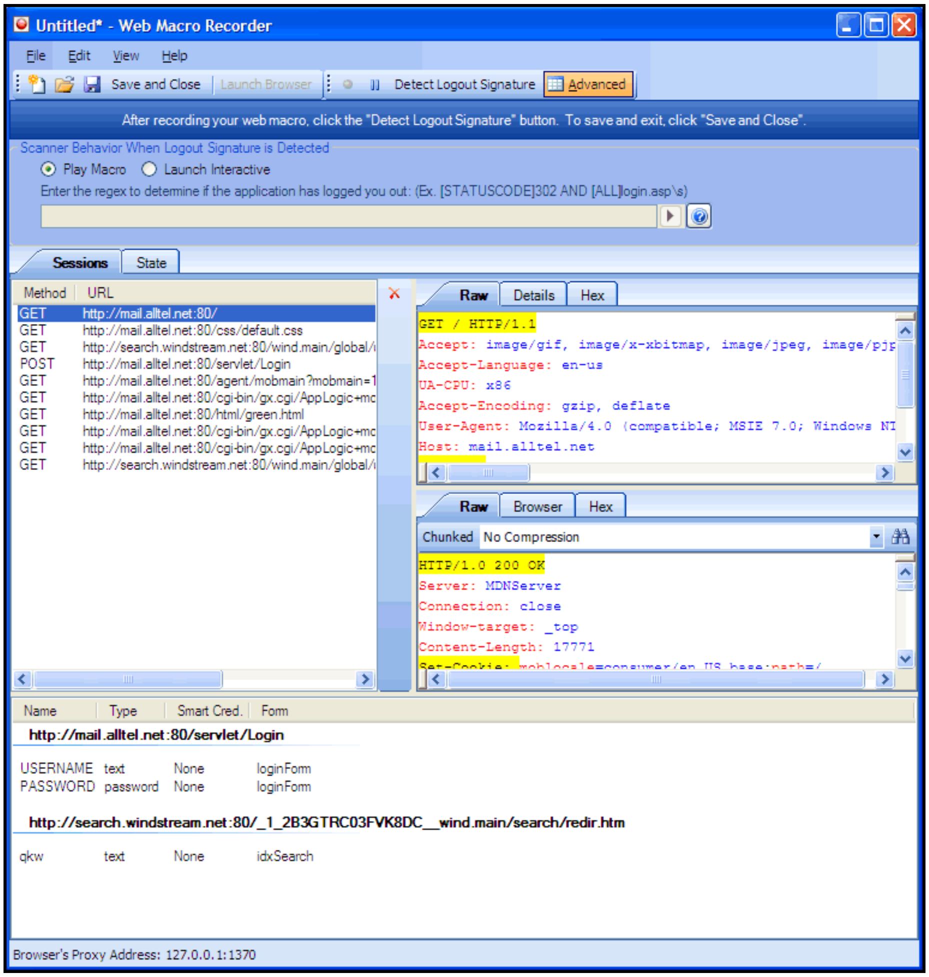
2 You can designate a control as a “Smart Credential” user name or password. Right-click the control name and select an option from the shortcut menu, as shown below.



Name	Type	Smart Cred.	Form
http://mail.alltel.net:80/servlet/Login			
USERNAME	text	None	loginForm
PASSWORD	password	None	loginForm

If you start an assessment using a macro that includes Smart Credentials, then when you scan the page containing the input elements associated with these entries, DevInspect will substitute the password specified in the Authentication options (or, if no user name is specified, the name of the current Windows user). This allows you to create the macro using your own user name and password, yet when someone else runs the scan using this macro, DevInspect will submit that user's name and password.

3 If you click the **Advanced** button, the Macro Editor displays the contents of the HTTP request and response in separate panes.



After recording your web macro, click the "Detect Logout Signature" button. To save and exit, click "Save and Close".

Enter the regex to determine if the application has logged you out: (Ex. [STATUSCODE]302 AND [ALL]login.asp\s)

Method	URL
GET	http://mail.alltel.net:80/
GET	http://mail.alltel.net:80/css/default.css
GET	http://search.windstream.net:80/wind.main/global/i
POST	http://mail.alltel.net:80/servlet/Login
GET	http://mail.alltel.net:80/agent/mobmain?mobmain=1
GET	http://mail.alltel.net:80/cgi-bin/gx.cgi/AppLogic+mc
GET	http://mail.alltel.net:80/html/green.html
GET	http://mail.alltel.net:80/cgi-bin/gx.cgi/AppLogic+mc
GET	http://mail.alltel.net:80/cgi-bin/gx.cgi/AppLogic+mc
GET	http://search.windstream.net:80/wind.main/global/i

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjp
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
Host: mail.alltel.net
```

Chunked No Compression

```
HTTP/1.0 200 OK
Server: MDNServer
Connection: close
Window-target: _top
Content-Length: 17771
Set-Cookie: moblocale=consumer/en_US base:path=/
```

Name	Type	Smart Cred.	Form
http://mail.alltel.net:80/servlet/Login			
USERNAME	text	None	loginForm
PASSWORD	password	None	loginForm
http://search.windstream.net:80/_1_2B3GTRC03FVK8DC__wind.main/search/redir.htm			
qkw	text	None	idxSearch

Browser's Proxy Address: 127.0.0.1:1370

4 You can also edit an HTTP request if, for example, you need to change or remove headers, or edit passwords or user names. Simply right-click a session and select **Edit** from the shortcut menu to launch the HTTP Editor.

Settings

Follow the steps below to modify the Web Macro Recorder settings:

1 Click the Web Macro Recorder **Edit** menu and select **Settings**.

2 In the **Proxy Listener** group, select a local IP address.

The Web Macro Recorder serves as a proxy that handles HTTP traffic between a browser and a target Web site. By default, it uses the local IP address 127.0.0.1 and any available port. However, you can specify a different IP address and port (if you clear the **Automatically Assign Port** check box).

3 Select **Save Files in clear text** if you do not want to save macros in an XML format using Base 64 encoding (which is the default). Saving files in clear text allows you to read the XML tags. The actual data, however, is not rendered in ASCII format and is not human readable.

4 Select **Always on Top** to keep the Web Macro Recorder displayed on your screen when you switch programs or windows.

Tips for Using Regular Expressions

HP engineers have developed and implemented extensions to the normal regular expression syntax. When building a regular expression, you can use the following tags and operators:

Regular Expression Tags:

[BODY]
[STATUSCODE]
[STATUSDESCRIPTION
[HEADERS]
[HEADER:<header name>]
[COOKIES]
[ALL]

Regular Expression Operators:

AND
OR
NOT
[ ]
( )

Note: You must include a space (ASCII 32) before and after an "open" or "close" parenthesis.

Examples

- To detect a response in which the status line contains a status code of "200" and having the phrase "logged out" somewhere in the message body, use this regular expression:

  [STATUSCODE]200 AND [BODY]logged\sout

- To detect a response indicating that the requested resource resides temporarily under a different URI (redirection) and having a reference to the path "/Login.asp" anywhere in the response, use the following:

  [STATUSCODE]302 AND [ALL]Login.asp

- To detect a response containing either (a) a status code of "200" and the phrase "logged out" or "session expired" anywhere in the body, or (b) a status code of "302" and a reference to the path "/Login.asp" anywhere in the response, use the following regular expression:

  ([STATUSCODE]200 AND [BODY]logged\sout OR [BODY]session\sexpired) OR ([STATUSCODE]302 AND [ALL]Login.asp)

  Note the spaces before and after the parenthesis.

- To detect a redirection response where "login.aspx" appears anywhere in the Location header, use the following regular expression:

  [STATUSCODE]302 AND [HEADER:Location]login.aspx

- To detect a response containing a specific string (such as "Please Authenticate") in the Reason-Phrase portion of the status line, use the following regular expression:

  [STATUSDESCRIPTION]Please\sAuthenticate

Web Form Editor

Most Web applications contain forms composed of input controls (text boxes, buttons, drop-down lists, etc.). Users generally "complete" a form by modifying its controls (such as entering text or checking boxes) before submitting the form to an agent for processing. Usually, this processing will lead the user to another page or section of the application. For example, after completing a login form, the user will proceed to the application's beginning page.

Some sites contain many different forms for completing a variety of transactions. If DevInspect is to navigate through all possible links in the application, it must be able to submit appropriate data for each form.

With the Web Form Editor, you can create or modify a file containing the names of all input controls and the associated values that need to be submitted during a scan of your Web site. These entries are categorized by URL, so even if different controls on different pages have the same name, the Web Form Editor can discriminate between them. Alternatively, you can designate a form entry as "global," meaning that its value will be submitted for any input control having the same name attribute, regardless of the URL at which it occurs.

During a scan, if DevInspect encounters an input control whose name attribute is not matched in the file you create, it will submit a default value (12345).

For server authentication (logging in to a server with a user name and password), you can enter values here or on the **Authentication** tab of the *Settings* window.

➤ If you are using a proxy server, the WebForm Editor will not use the default settings from DevInspect. You must first configure Internet Explorer to use the desired proxy.

There are two ways to create a list of form values:

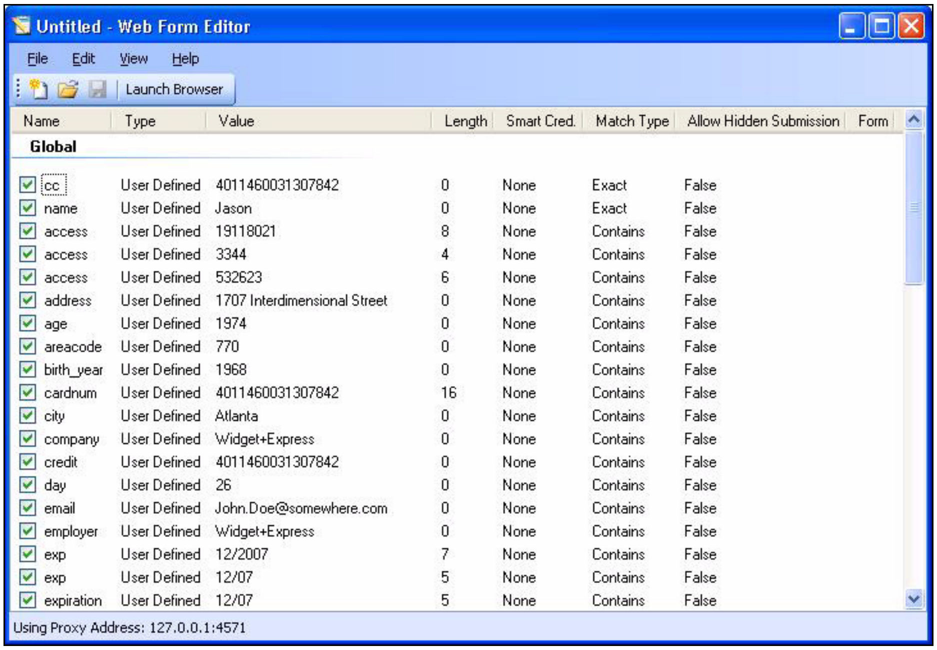
- Create the list manually
- Record the values as you navigate through the application

Manually Creating a Web Form List

Use the following procedure to create a WebForm list manually.

1 From the **Tools** menu, select **WebForm Editor**.

The *WebForm Editor* window appears.



Name	Type	Value	Length	Smart Cred.	Match Type	Allow Hidden Submission	Form
Global							
☑ cc	User Defined	4011460031307842	0	None	Exact	False	
☑ name	User Defined	Jason	0	None	Exact	False	
☑ access	User Defined	19118021	8	None	Contains	False	
☑ access	User Defined	3344	4	None	Contains	False	
☑ access	User Defined	532623	6	None	Contains	False	
☑ address	User Defined	1707 Interdimensional Street	0	None	Contains	False	
☑ age	User Defined	1974	0	None	Contains	False	
☑ areacode	User Defined	770	0	None	Contains	False	
☑ birth_year	User Defined	1968	0	None	Contains	False	
☑ cardnum	User Defined	4011460031307842	16	None	Contains	False	
☑ city	User Defined	Atlanta	0	None	Contains	False	
☑ company	User Defined	Widget+Express	0	None	Contains	False	
☑ credit	User Defined	4011460031307842	0	None	Contains	False	
☑ day	User Defined	26	0	None	Contains	False	
☑ email	User Defined	John.Doe@somewhere.com	0	None	Contains	False	
☑ employer	User Defined	Widget+Express	0	None	Contains	False	
☑ exp	User Defined	12/2007	7	None	Contains	False	
☑ exp	User Defined	12/07	5	None	Contains	False	
☑ expiration	User Defined	12/07	5	None	Contains	False	

Using Proxy Address: 127.0.0.1:4571

The WebForm Editor loads a prepackaged default file.

a To load a different file, select **Open** from the WebForm Editor's **File** menu.

b To create a new file, select **New** from the **File** menu

2 Do one of the following:

- To add a Web form value, right-click anywhere in the Web Form Editor's work area and select **Add Global Form Input** from the shortcut (pop-up) menu.
- To modify a Web form value, right-click an entry and select **Modify** from the shortcut (pop-up) menu.

The *Add User-Defined Input* or the *Modify Input* window appears.

3 In the **Name** box, type (or modify) the name attribute of the input element.

4 In the **Length** box, enter either:

- the value that must be specified by the size attribute, or
- zero, for input elements that do not specify a size attribute.

For example, to submit data for the following HTML fragment…

<INPUT TYPE="password" NAME="accessID" MAXLENGTH="6">

…you must create an entry consisting of accessID (Name) and specify a size of "6" (Length).

5 In the **Value** box, type the data that should be associated with the input element (for example, a password).

6 Use the **Match** list to specify how the scanner should determine if this entry qualifies to be submitted for a particular input control. The options are:

- **Exact** - The name attribute of the input control must match exactly the name assigned to this entry.

- **Starts with** - The name attribute of the input control must begin with the name assigned to this entry.
- **Contains** - The name attribute of the input control must contain the name assigned to this entry.

7 Programmers sometimes use input controls with type= “hidden” to store information between client/server exchanges that would otherwise be lost due to the stateless nature of HTTP. Although the Web Form Editor will collect and display the attributes for hidden controls, the scanner will not submit values for hidden controls unless you select **Allow Hidden Submission**.

8 Click **Add** (or **Modify**).

9 If necessary, you can assign additional attributes by right-clicking an entry and using the shortcut (pop-up) menu.

- To remove an entry, choose **Unselect**. This clears the check mark and removes the entry from processing, but does not delete it from the file.
- To activate an entry, choose **Select**. This creates a check mark and includes the entry for processing.
- To delete an entry, choose **Delete**.
- To designate an entry as a smart credential, select either **Smart Credential Username** or **Smart Credential Password**.

  When recording Web form values, you will often encounter a log-on form requiring you to enter a user name and password. You can safely use your own user name and password, provided that you designate those entries as “Smart Credentials” before saving the file. Your actual password and user name are not saved.

  When scanning the page containing the input control associated with this entry, the scanner will substitute the password specified in the product's Authentication options. This would be a known user name and password that does not require security. Alternatively, if no user name or password is specified, the scanner will submit the string “FormFillText.”
- If you select **Mark As Interactive Input**, the scanner will pause the scan and display a window prompting the user to enter a value for this entry (if the scan options include the settings **Prompt For Web Form Values During Scan** and **Only Prompt Tagged Inputs**).

  It is not necessary to tag passwords with **Mark As Interactive Input**.

Recording Web Form Values

The Web Form Editor serves as a proxy that handles HTTP traffic between a browser and a target Web site. By default, it uses the local IP address 127.0.0.1 and any available port. However, you can specify a different IP address and port by selecting **Settings** from the **Edit** menu.

Use the following procedure to capture names and values of input controls on a Web site.

1 To create a list of form values, select **New** from the **File** menu (or click the New icon on the toolbar).

2 To add form values to an existing list, select **Open** from the **File** menu (or click the Open icon on the toolbar) and choose a file using the standard file-selection dialog.

3 Click **Launch Browser**.

4 Using the browser's **Address** bar, enter or select a URL and navigate to a page containing a form.

5 Complete the form and submit it (usually by clicking a button such as **Log In**, **Submit**, **Go**, etc.).

6 Navigate to additional pages and submit forms until you have traversed all the links you wish to follow.

7 The Web Form Editor displays a list of name and value attributes for all input controls found in all forms on the pages you visited.

For example, the last two entries in the following illustration were derived from the following HTML fragment…

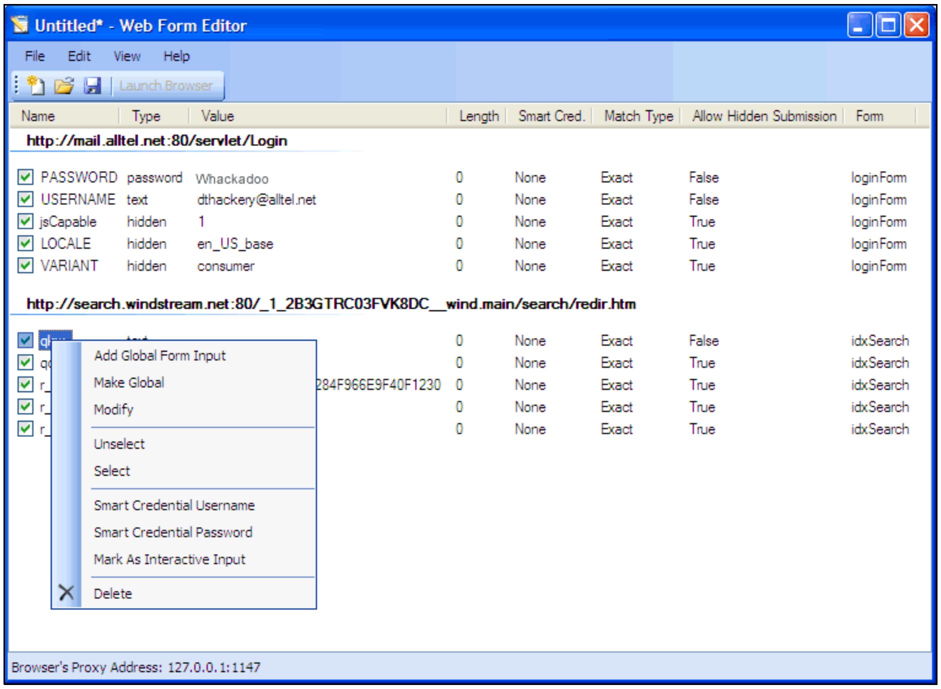
```
<form name="loginForm" action="/servlet/Login" method="POST">

<input type="password" size="16" name="PASSWORD">

<input type="text" size="16" name="USERNAME" value="">

<input type="SUBMIT" value="Submit"></form>
```

…and the user entered his name and password.


Untitled* - Web Form Editor
File
Edit
View
Help
Launch Browser
Add Global Form Input
Make Global
Modify
Unselect
Select
Smart Credential Username
Smart Credential Password
Mark As Interactive Input
Delete

Name	Type	Value	Length	Smart Cred.	Match Type	Allow Hidden Submission	Form
http://mail.alltel.net:80/servlet/Login							
☑ PASSWORD	password	Whackadoo	0	None	Exact	False	loginForm
☑ USERNAME	text	dthackery@alltel.net	0	None	Exact	False	loginForm
☑ jsCapable	hidden	1	0	None	Exact	True	loginForm
☑ LOCALE	hidden	en_US_base	0	None	Exact	True	loginForm
☑ VARIANT	hidden	consumer	0	None	Exact	True	loginForm
http://search.windstream.net:80/_1_2B3GTRC03FVK8DC__wind.main/search/redir.htm							
☑ q			0	None	Exact	False	idxSearch
☑ q			0	None	Exact	True	idxSearch
☑ r_		284F966E9F40F1230	0	None	Exact	True	idxSearch
☑ r_			0	None	Exact	True	idxSearch
☑ r_			0	None	Exact	True	idxSearch

Browser's Proxy Address: 127.0.0.1:1147

If necessary, you can modify items by right-clicking an entry and using the shortcut (pop-up) menu.

8 Do one of the following:

- To edit an entry, select **Modify**.
- To add an entry, select **Add Global Form Input**. A Global entry is one not associated with a specific URL.
- To remove an entry, choose **Unselect**. This removes the entry from processing, but does not delete it from the file.
- To delete an entry, choose **Delete**.

- To designate an entry as a smart credential, select either **Smart Credential Username** or **Smart Credential Password**.
- To force the scanner to pause and display a window prompting the user to enter a value for this entry, select **Mark As Interactive Input**.

  When a scanner encounters an HTTP or JavaScript form, it will pause the scan and display a window that allows you to enter values for input controls within the form, provided that the scanner's option to **Prompt For Web Form Values** is selected. However, if the scanner's option to **Only Prompt Tagged Inputs** is also selected, DevInspect will not pause for user input unless a specific input control has been designated **Mark As Interactive Input** (except for passwords, which always cause the scanner to pause for input).

9. From the **File** menu, click **Save** or **Save As**.

Importing a Web Form File

You can import a file that was designed and created for earlier versions of DevInspect and convert it to a file that can be used by the current Web Form Editor.

1. From the **File** menu, select **Import**.

   The *Convert Web Form Values* window appears.
2. Click the ellipses button next to **Select File To Import**.
3. Using a standard file-selection window, locate the XML file created by an earlier version of the Web Form Editor.
4. Click the ellipses button next to **Select Target File**.
5. Using a standard file-selection window, specify a file name and location for the converted file.
6. Click **OK**.

Scanning with a Web Form File

There is no setting to specify which Web Form file should be used to submit values during a scan. If you elect to submit forms, DevInspect will use either the prepackaged default data set or the one last loaded into the Web Form Editor.

Web Form Editor Settings

Follow the steps below to modify the Web Form Editor settings:

1. Click the **Edit** menu and select **Settings**.
2. Select either the **General** or **Proxy** category and enter the settings described in the following sections.
3. Click **OK**.

General

Proxy Listener

The Web Form Editor serves as a proxy that handles HTTP traffic between a browser and a target Web site. By default, it uses the local IP address 127.0.0.1 and any available port. However, you can specify a different IP address and port by selecting Settings from the Edit menu.

To avoid the possibility of specifying a port that is already in use, select Automatically Assign Port.

Proxy

Use these settings to access the Web Form Editor through a proxy server.

Direct Connection (proxy disabled)

Select this option if you are not using a proxy server.

Auto detect proxy settings

Not enabled.

Use Internet Explorer proxy settings

Select this option to import your proxy server information from Internet Explorer.

Configure a proxy using a PAC file

Not enabled.

Explicitly configure proxy

Select this option to access the Internet through a proxy server, and then enter the requested information

1 In the Server box, type the URL or IP address of your proxy server, followed (in the **Port** box) by the port number (for example, 8080).

2 Select a protocol for handling TCP traffic through a proxy server: SOCKS4, SOCKS5, or standard.

   Important: Smart Update is not available if you use a SOCKS4 or SOCKS5 proxy server configuration. Smart Update is available only when using a standard proxy server.

3 If your proxy server requires authentication, enter the qualifying user name and password.

4 If you do not need to use a proxy server to access certain IP addresses (such as internal testing sites), enter the addresses or URLs in the **Bypass Proxy For** box. Use commas to separate entries.

HTTPS Proxy Settings

For proxy servers accepting HTTPS connections, select **Specify Alternative Proxy for HTTPS** and provide the requested information.

Web Form Editor Menus

The Web Form Editor menu bar contains the menus described in the following sections.

File Menu

The **File** menu contains the following commands:

- **New** - Deletes all data.
- **Open** - Loads a file containing a Web form values.
- **Import** - Loads an XML file designed and created for earlier versions of DevInspect and convert it to a file that can be used by the current Web Form Editor
- **Save** - Saves the Web form values.
- **Save As** - Saves the Web form values.
- **Exit** - Closes the Web Form Editor.

Edit Menu

The **Edit** menu contains the following command:

- **Settings** - Allows you to configure parameters for the Web Macro Recorder.

View Menu

The **View** menu contains the following command:

- **Launch Browser** - Launches the default browser.

Help Menu

The **Help** menu contains the following commands:

- **Web Macro Recorder Help** - Opens the Help file with the **Contents** tab active.
- **Index** - Opens the Help file with the **Index** tab active.
- **Search** - Opens the Help file with the **Search** tab active.
- **About Web Form Editor** - Displays information about the Web Form Editor.

Web Form Logic

When crawling a Web application and submitting Web form values, DevInspect analyzes the entries in the Web form values file to determine if a value should be submitted. The logic for determining a match is represented in the following table, ordered from “most preferred” to “least preferred.”

Table 1 Rules for Matching Web Form Values

Page-specific form values	Exact Match. Name exact match. Length exact match.	The specific Web page, Web form name, and value length detected on the crawled Web page exactly match a single record in the webformvalues.xml selected for the scan.
	Partial Match. Name-only match. Length allows wildcard.	The specific Web page and Web form name detected on the crawled Web page match a single record in the webformvalues.xml selected for the scan. The field length associated with that form value allows for submission to any field input length (wildcard field length match).
Global form values	Exact Match. Name exact match. Length exact match.	The Web form name and value length detected on the crawled Web page match a single record in the Global Web form values section of the webformvalues.xml selected for the scan.
	Partial Match 1. Name exact match. Length allows wildcard.	The Web form name detected on the crawled Web page exactly matches a form name found in the global values section of the webformvalues.xml selected for the scan. The field length associated with that form value allows for submission to any field input length (wildcard field length match).
	Partial Match 2. Field name starts with Name value. Length exact match.	A Web form value in the file partially matches the field name found. All characters in the Web form value match the beginning of the Web page field name and the field length detected on the crawled Web page match the record in the Global Web form values section of the webformvalues.xml selected for the scan.
	Partial Match 3. Field name starts with Name value. Length allows wildcard.	A Web form value in the file partially matches the field name found. All characters in the Web form value match the beginning of the Web page field name and the field length for the record allows for submission to any field length (wildcard field length match).
	Partial Match 4. Name value included in field name. Length exact match.	A Web form value in the file partially matches the field name found. All characters in the Web form value match a portion of the Web page field name and the field length for the record allows for submission to any field length (wildcard field length match).

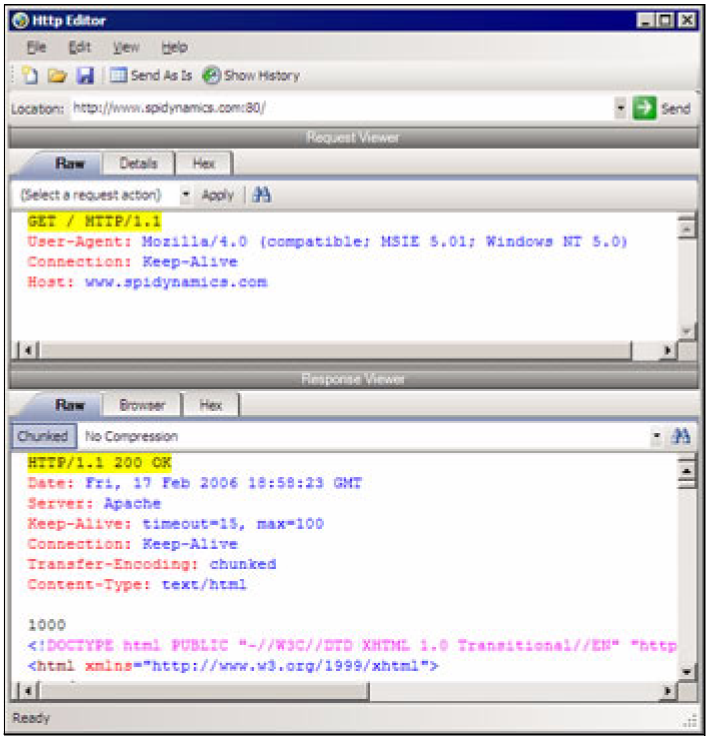
Table 1 Rules for Matching Web Form Values (cont'd)

	Partial Match 5. Name value included in field name. Length allows wildcard.	A Web form value in the file partially matches the field name found. All characters in the Web form value match a portion of the Web page field name and the field length for the record allows for submission to any field length (wildcard field length match).
No match	Field name has no exact or partial matches to Web form values.	No Web form value match was found. Submit the specified default value (Default).
No default value	The Web form values file has no default value specified.	No Web form value match was made and the default value is not in the webform values file. Submit “not found.”

HTTP Editor

Use the HTTP Editor to create or edit requests, send them to a server, and view the response either in raw HTML or as rendered in a browser. The HTTP Editor is a manual hacking tool that requires a working knowledge of HTML, HTTP, and request methods.

To set proxy and authorization parameters, if necessary, select **Settings** from the **File** menu.



Location: http://www.spidynamics.com:80/

No Compression

```
GET / HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Connection: Keep-Alive
Host: www.spidynamics.com
```

```
HTTP/1.1 200 OK
Date: Fri, 17 Feb 2006 18:58:23 GMT
Server: Apache
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html

1000
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http
<html xmlns="http://www.w3.org/1999/xhtml">
```

Panes

The Request Viewer pane contains the HTTP request message, which you can view in three different formats using the following tabs:

- **Raw** - Depicts the line-by-line textual format of the request message.
- **Details** - Displays the header names and field values in a table format.
- **Hex** - Displays the hexadecimal and ASCII representation of the message.

The Response Viewer pane contains the HTTP response message, which you can also view in three different formats using the following tabs:

- **Raw** - Depicts the line-by-line textual format of the response message.
- **Browser** - Displays the response message as rendered in a browser.
- **Hex** - Displays the hexadecimal and ASCII representation of the response message.

HTTP Editor Menus

File Menu

The **File** menu contains the following commands:

- **New Request** - Deletes all information from previous sessions and resets the Location URL.
- **Open Request** - Allows you to load a file containing an HTTP request saved during a previous session.
- **Save Request** - Allows you to save an HTTP request.
- **Save Request As** - Allows you to save an HTTP request.
- **URL Synchronization** - When selected, any characters you type into the Address combo box are added to the Request-URI of the HTTP request line.
- **Send As Is** - Allows you to send a malformed HTTP request. Note that most standard HTTP proxies cannot process non-compliant HTTP requests.
- **Exit** - Closes the HTTP Editor.

Edit Menu

The **Edit** menu contains the following commands:

- **Cut** - Deletes selected text and saves it to the clipboard.
- **Copy** - Saves the selected text to the clipboard.
- **Paste** - Inserts text from the clipboard
- **Find** - Displays a window that allows you to search for text that you specify.
- **Settings** - Allows you to configure request, authentication, and proxy parameters for the HTTP Editor.

View Menu

The View menu contains the following commands:

- **Show History** - Displays a pane listing all HTTP requests sent.
- **Word Wrap** - Causes all text to fit within the defined margins.

Help Menu

The Help menu contains the following commands:

- **HTTP Editor Help** - Opens the Help file with the Contents tab active.
- **Index** - Opens the Help file with the Index tab active.
- **Search** - Opens the Help file with the Search tab active.
- **About HTTP Editor** - Displays information about the HTTP Editor.

Request Actions

The following options are available from the **Request Action** list in the Request Viewer pane.

PUT File Upload

The PUT method requests that the enclosed entity be stored under the supplied Request-URI.

To write a file to a server:

1. Select **PUT File Upload** from the drop-down list on the Request Viewer pane.
2. In the text box that appears to the right of the list, type the full path to a file
   - or -
   Click the Open Folder icon and select the file you want to upload.
3. Click **Apply**. This will also recalculate the content length.

Change Content-Length

If, by editing the request, you change the length of the message body but do not change the field value of the Content-length header, the server will return an error. To avoid this problem, select **Change Content-Length** and click **Apply**. The HTTP Editor will recalculate the content length and substitute the appropriate value.

URL Encode/Decode Param Values

The specification for URLs (RFC 1738, Dec. '94) limits the use of characters in URLs to a subset of the US-ASCII character set. HTML, on the other hand, allows the entire range of the ISO-8859-1 (ISO-Latin) character set to be used in documents, and HTML4 expands the allowable range to include the complete Unicode character set as well. To circumvent this limitation, you can encode non-standard letters and characters for display in browsers and plug-ins that support them.

URL encoding of a character consists of a "%" symbol, followed by the two-digit hexadecimal representation of the ISO-Latin code point for the character. For example:

- The asterisk symbol (*) = 42 decimal in the ISO-Latin set
- 42 decimal = 2A hexadecimal
- URL code for asterisk = %2A

You can use URL encoding to bypass an intruder detection system (IDS) that inspects request messages for certain keywords using only the ISO-Latin character set. For example, the IDS may search for "login" (in ISO-Latin), but not "%4C%4F%47%49%4E" (the URL-encoded equivalent).

To substitute URL code for parameters throughout the entire message, select **URL Encode Param Values** and click **Apply**.

To translate URL-encoded parameters to ISO-Latin, select **URL Decode Param Values** and click **Apply**.

Unicode Encode/Decode Request

The Unicode Worldwide Character Standard includes letters, digits, diacritics, punctuation marks, and technical symbols for all the world's principal written languages, using a uniform encoding scheme. Incorporating Unicode into client-server applications and Web sites offers significant cost savings over the use of legacy character sets. Unicode enables a single software product or a single Web site to be targeted across multiple platforms, languages and countries without re-engineering. It allows data to be transported through many different systems without corruption.

To translate the entire request message into Unicode, select **Unicode Encode Request** and click **Apply**.

To translate the entire request message from Unicode into ISO-Latin, select **Unicode Decode Request** and click **Apply**.

Create MultiPart Post

The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. You can attempt to upload data by manipulating a POST request message.

To insert data from a file:

1. Select **Create MultiPart Post** from the **Action** list on the Request pane.
2. In the text box to the right of the **Action** list, type the full path to a file
   - or -
   Click the Open Folder icon and select the file you want to insert.
3. Click **Apply**.

Response Actions

The area immediately below the tabs on the Response Viewer pane contains three controls:

- a **Chunked** button
- a **Content Coding** drop-down list
- a button that launches the **Find In Response** dialog, allowing you to search the response for the text string you specify.

Chunked

If a server starts sending a response before knowing its total length, it might break the complete response into smaller chunks and send them in series. Such a response contains the “Transfer-Encoding: chunked” header. A chunked message body contains a series of chunks, followed by a line with “0” (zero), followed by optional footers and a blank line. Each chunk consists of two parts:

- A line with the size of the chunk data, in hex, possibly followed by a semicolon and extra parameters you can ignore (none are currently standard), and ending with CRLF.
- The data itself, followed by CRLF.

Content Codings

If the HTTP response contains compressed data, you can decompress the data using one of the options from the list.

- GZIP - A compression utility written for the GNU project.
- Deflate - The “zlib” format defined in RFC 1950 [31] in combination with the “deflate” compression mechanism described in RFC 1951 [29].

Editing and Sending Requests

Follow the steps below to edit and send a request.

1. Modify the request message in the Request Viewer pane.

   To change certain features of the request, select an item from the **Action** list and click **Apply**.

2. Click **Send** to send the HTTP request message.

   The Response Viewer pane displays the HTTP response message when it is received.

3. To view the response as rendered in a browser, click the **Browser** tab.
4. You can prepare your next HTTP request using the HTML or JavaScript controls rendered on the **Browser** tab. To use this feature, you must select the **Interactive Navigation** option (click the **File** menu and select **Settings**).
5. To save a request, select **Save Requests** from the **File** menu.

Searching for Text

Follow the steps below to search for text in the request or response

1. Click 🔍 in either the Request Viewer or Response Viewer pane.
2. Using either the *Find in Request* or *Find in Response* window, type or select a string or regular expression.
3. If using a regular expression as the search string, select the **Regex** check box.
4. Click **Find**.

Settings

Follow the steps below to configure HTTP Editor settings:

1. Click the **Edit** menu and select **Settings**.
2. Select the **Options** tab.
3. In the **Request** group:

   a. If you select **Send As Is**, the HTTP Editor will not modify the request, regardless of any other settings you may select. This allows you to send a purposely malformed message. Authentication and proxy settings are disabled when using this option.

   b. If you select **Manipulate Request**, the HTTP Editor will modify requests to accommodate the following parameters:

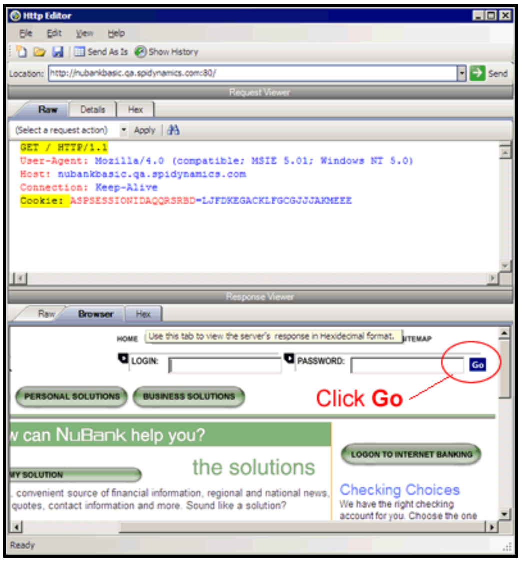
      — **Apply State** — If your application uses cookies, URL rewriting, or post data techniques to maintain state within a session, the HTTP Editor will attempt to identify the method and modify the response accordingly.

      — **Apply Proxy** — If you select this option, the HTTP Editor will modify the request according to the proxy settings you specify.

4. In the **Navigation** group, select either **None**, **Interactive**, or **Browser Mode**.

You can view the server's response as rendered in a browser by selecting the **Browser** tab in the Response Viewer (the lower pane). If the **Interactive** feature is enabled, you can prepare your next HTTP request using the HTML or JavaScript controls rendered in the browser.

For example, using the logon page at nubankbasic.qa.spidynamics.com (shown below), you could enter a user name ("admin") and password ("admin"), and then click **Go**.

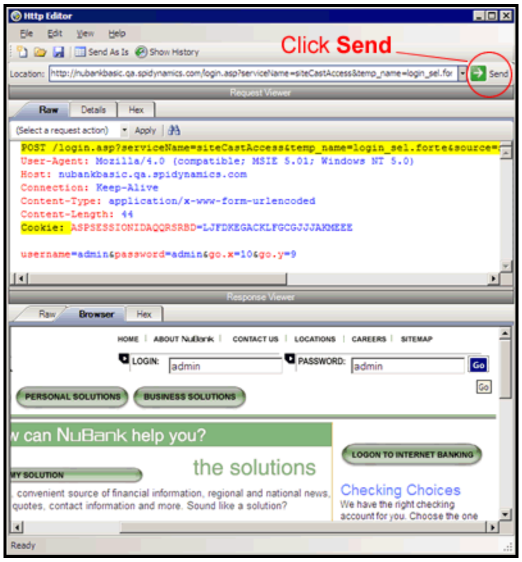

Http Editor
File Edit View Help
Send As Is
Show History
Send
Request Viewer
Raw
Details
Hex
(Select a request action)
Apply
Response Viewer
Raw
Browser
Hex
Use this tab to view the server's response in Hexidecimal format.
HOME
ITEMAP
LOGIN:
PASSWORD:
Go
PERSONAL SOLUTIONS
BUSINESS SOLUTIONS
v can NuBank help you?
the solutions
MY SOLUTION
convenient source of financial information, regional and national news, quotes, contact information and more. Sound like a solution?
LOGON TO INTERNET BANKING
Checking Choices
We have the right checking account for you. Choose the one
Ready

Location: http://nubankbasic.qa.spidynamics.com:80/

```
GET / HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: nubankbasic.qa.spidynamics.com
Connection: Keep-Alive
Cookie: ASPSESSIONIDAQQRSRBD=LJFDKEGACKLFGCGJJJAKMEEE
```

Click **Go**

The HTTP Editor formats the request (which uses the POST method to the login1.asp resource) and displays it in the Request Viewer, as illustrated below. You could then edit the logon message (if required) or simply send it to the server by clicking **Send**.


Http Editor
File Edit View Help
Send As Is
Show History
Send
Request Viewer
Raw
Details
Hex
(Select a request action)
Apply
Response Viewer
Raw
Browser
Hex
HOME | ABOUT NuBank | CONTACT US | LOCATIONS | CAREERS | SITEMAP
LOGIN:
PASSWORD:
Go
PERSONAL SOLUTIONS
BUSINESS SOLUTIONS
v can NuBank help you?
the solutions
MY SOLUTION
convenient source of financial information, regional and national news, quotes, contact information and more. Sound like a solution?
LOGON TO INTERNET BANKING
Checking Choices
We have the right checking account for you. Choose the one
Ready

Click **Send**

Location: http://nubankbasic.qa.spidynamics.com/login.asp?serviceName=siteCastAccess&temp_name=login_sel.for

```
POST /login.asp?serviceName=siteCastAccess&temp_name=login_sel.forte&source=
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: nubankbasic.qa.spidynamics.com
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 44
Cookie: ASPSESSIONIDAQQRSRBD=LJFDKEGACKLFGCGJJJAKMEEE

username=admin&password=admin&go.x=10&go.y=9
```

LOGIN: admin
PASSWORD: admin

If you select the **Browser Mode** option, then Interactive mode is enabled, but the HTTP Editor will send the request immediately, without first placing it in the Request Viewer and allowing you to edit it.

5 Select **Enable Active Content** to allow execution of JavaScript and other dynamic content.

6 Click the **Authentication** tab and select an authentication method:

- **None** - Select this option if the site does not require authentication.
- **Automatic Authentication** - This allows Web Brute to determine the correct authentication type.
- **HTTP Basic Authentication** - This is a widely used, industry-standard method for collecting user name and password information. Normally, a Web browser displays a window for a user to enter a previously assigned user name and password, also known as credentials. The Web browser then attempts to establish a connection to a server using the user's credentials.
- **NTLM Authentication -** NTLM (NT LanMan) is an authentication process that is used by all members of the Windows NT family of products. Like its predecessor LanMan, NTLM uses a challenge/response process to prove the client's identity without requiring that either a password or a hashed password be sent across the network.

7 If authentication is required, provide the following information:

- User name
- Password (and then retype it in the **Confirm Password** field)

8 To force the HTTP Editor to submit the specified user name and password when it encounters forms with password fields, select **Submit these credentials to forms with password input fields**.

9 Click the **Proxy** tab and select one of the following options:

- **Direct Connection (proxy disabled)** - Select this option if you are not using a proxy server.
- **Auto detect proxy settings** - If you select this option, DevInspect will use the Web Proxy Autodiscovery Protocol (WPAD) to automatically locate a proxy autoconfig file and use this to configure the browser's Web proxy settings.
- **Use Internet Explorer proxy settings** - Select this option to import your proxy server information from Internet Explorer.
- **Configure a proxy using a PAC file** - Select this option to load proxy settings from a Proxy Automatic Configuration (PAC) file. Then specify the file location in the **URL** box.
- **Explicitly configure proxy** - Select this option to access the Internet through a proxy server, using the configuration information you provide. If you select this option, you may also select HTTPS Proxy Settings for proxy servers accepting https connections.

10 Click **OK**.

Web Proxy

Web Proxy is a stand-alone, self-contained proxy server that you can configure and run on your desktop. With it, you can monitor traffic from DevInspect, a browser, or any other tool that submits HTTP requests and receives responses from a server. It is a tool for debugging and penetration assessment; you can see every request and server response while browsing a site.

You can also create a Startup macro or a Login macro that you can use with DevInspect or the Access Management Platform (AMP).

Before using Web Proxy with your browser, you must configure your browser's proxy settings. If using Internet Explorer:

1. Click the **Tools** menu and select **Internet Options**.
2. Click the **Connections** tab.
3. Click **LAN Settings**.
4. On the *LAN Settings* window, select **Use a Proxy Server for your LAN** and enter the address and port of the proxy server you want to use. By default, Web Proxy uses your local host settings (127.0.0.1:8080).

You should also configure Microsoft Internet Explorer to use HTTP 1.1 through proxy connections. On Internet Explorer:

1. Click the **Tools** menu and select **Internet Options**.
2. Click the **Advanced** tab.
3. In the “HTTP1.1 settings” section, select **Use HTTP 1.1 through proxy connections**.

Using Web Proxy

Follow the steps below to use Web Proxy with a browser:

1. From the **Tools** menu, click **Web Proxy**.

   The *Web Proxy* window opens.

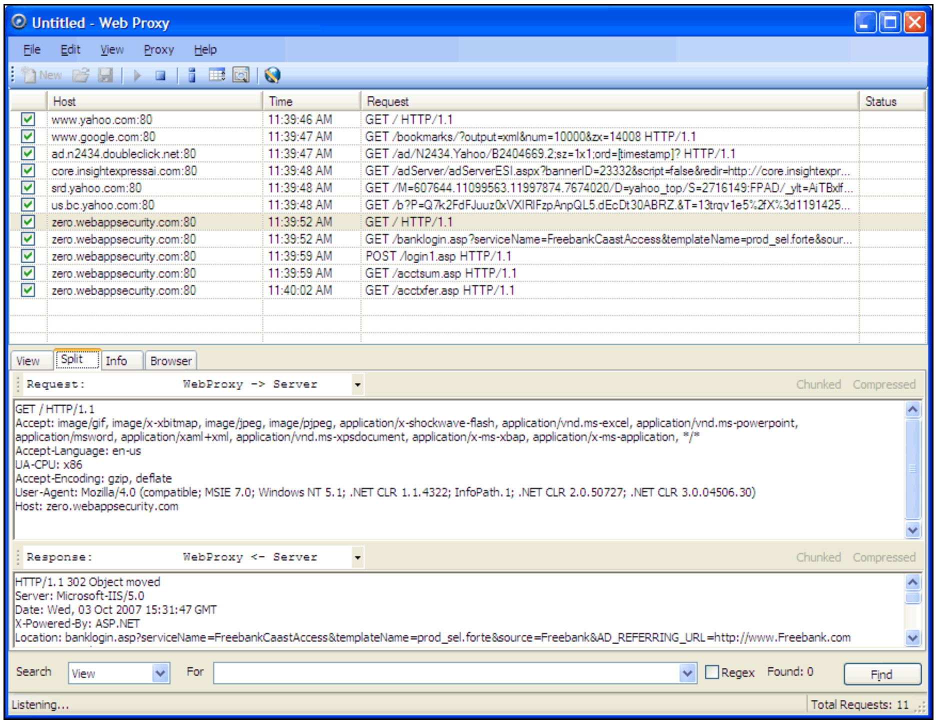
2. Click ▶ or select **Start** from the **Proxy** menu.

   “Listening” displays in the Web Proxy status bar.

3. Open your Web browser and manually navigate the site for which you want to view requests/responses.

   If Web Proxy receives a request for a certificate from a Web Server, it displays a dialog asking you to locate the certificate. The program then caches your selection on a “per server” basis. Therefore, if you subsequently want to use a different certificate for a particular server, you must clear the cache by stopping and then restarting Web Proxy.

4. When you have browsed to all necessary pages, return to Web Proxy and click ■ select **Stop** from the **Proxy** menu.
5. Each session (a request and matching response) you recorded is listed in the top pane. To view the actual HTTP message, select an entry. The message appears in the bottom frame. By default, the **View** tab is selected.



	Host	Time	Request	Status
☑	www.yahoo.com:80	11:39:46 AM	GET / HTTP/1.1	
☑	www.google.com:80	11:39:47 AM	GET /bookmarks/?output=xml&num=10000&zx=14008 HTTP/1.1	
☑	ad.n2434.doubleclick.net:80	11:39:47 AM	GET /ad/N2434.Yahoo/B2404669.2;sz=1x1;ord=[timestamp]? HTTP/1.1	
☑	core.insightexpressai.com:80	11:39:48 AM	GET /adServer/adServerESI.aspx?bannerID=23332&script=false&redir=http://core.insightexpr...	
☑	srd.yahoo.com:80	11:39:48 AM	GET /M=607644.11099563.11997874.7674020/D=yahoo_top/S=2716149:FPAD/_ylt=AiTBxlf...	
☑	us.bc.yahoo.com:80	11:39:48 AM	GET /b?P=Q7k2FdFJuuz0xVXIRIFzpAnpQL5.dEcDt30ABRZ.&T=13trqv1e5%2fX%3d1191425...	
☑	zero.webappsecurity.com:80	11:39:52 AM	GET / HTTP/1.1	
☑	zero.webappsecurity.com:80	11:39:52 AM	GET /banklogin.asp?serviceName=FreebankCaastAccess&templateName=prod_sel.forte&sour...	
☑	zero.webappsecurity.com:80	11:39:59 AM	POST /login1.asp HTTP/1.1	
☑	zero.webappsecurity.com:80	11:39:59 AM	GET /acctsum.asp HTTP/1.1	
☑	zero.webappsecurity.com:80	11:40:02 AM	GET /acctxfer.asp HTTP/1.1	

```
Request:            WebProxy -> Server
```

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint,
application/msword, application/xaml+xml, application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-application, */*
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; InfoPath.1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
Host: zero.webappsecurity.com
```

```
Response:           WebProxy <- Server
```

```
HTTP/1.1 302 Object moved
Server: Microsoft-IIS/5.0
Date: Wed, 03 Oct 2007 15:31:47 GMT
X-Powered-By: ASP.NET
Location: banklogin.asp?serviceName=FreebankCaastAccess&templateName=prod_sel.forte&source=Freebank&AD_REFERRING_URL=http://www.Freebank.com
```

6 To change the format in which the message is displayed, select one of the tabs (**View**, **Split**, **Info**, or **Browser**).

When using the **View** or **Split** tabs, the **Chunked** and **Compressed** buttons are enabled if a response is either chunked-encoded or compressed. This allows you to view the original response received by Web Proxy as well as the de-chunked or decompressed response.

7 To edit a request, select it from the lists of displayed requests and click **Request Editor**.

Use the **File** menu to save selected requests to an xml file and later load them for analysis. You can also save a sequence of requests as a Web macro that you can use when conducting a DevInspect scan. All **File** menu commands apply to "check-marked" requests.

Click the top of any column to sort the requests by that selection. For example, to sort the requests by the time they were made, click the top border of the **Time** column.

➤ You must stop Web Proxy when you want to change Web Proxy settings.

Creating a Web Macro

You can use either the Web Macro Recorder or Web Proxy to create a Start macro or a Login macro.

A Start macro is used most often to focus on a particular subsection of an application. It specifies URLs that an HP scanner will use to navigate to the area. It may also include login information, but does not contain logic that will prevent the scanner from logging out of your application.

A Login macro is used for Web form authentication, allowing the scanner (or the AMP sensor) to log in to an application. You can also incorporate logic that will prevent the scanner from logging out of your application.

Follow the steps below to create a macro using sessions captured by Web Proxy:

1. Select the sessions you want to include in the macro by placing a check mark in the left column.
2. Click the **File** menu and select **Create Web Macro**.
3. (Optional) On the *Create Web Macro* dialog, select **Enable Check For Logout** and then enter a regular expression that identifies a unique text or phrase that occurs in the server's HTTP response when a user logs out or when a user who is not logged in requests access to a protected URL.

   Background: During a normal scan, the scanner begins crawling your site at the home page. If it encounters a link to another resource (usually through an <A HREF> HTML tag), it will navigate to that URL and continue its assessment. If it follows a link to a logout page (or if the server automatically "logs out" a client after a certain number of minutes), the scanner will not be able to visit additional resources where the client is required to be logged in. When this inadvertent log-out occurs, the scanner must be able to log in again without user intervention. This process hinges on the scanner's ability to recognize when it is no longer logged in.

   In some applications, if the user logs out (by clicking a button or some other control), the server responds with a unique message, such as "Have a nice day." If you specify this phrase as the server's logout signature, the scanner searches every response message for this phrase. Whenever it detects the phrase, the scanner attempts to log in again by sending an HTTP request containing the username and password.

   The scanner can also detect that it has logged out if the server sends a specific message in response to the scanner's attempt to access a password-protected URL. For example, the server may respond with a status code of "302 Object moved." If the scanner knows specifically what to look for in this response, the program will recognize that it has been logged out and can re-establish a logged-in state.

   Using the background example (above), if your server returns a message such as "Have a nice day" when a user logs out of your application, then enter "Have\sa\snice\sday" as the regular expression ("\s" is used in regular expressions to designate a space). A more likely example is where the server returns a 302 status code and references a new URL. In this case, "[STATUSCODE]302 AND [ALL]http://login.myco.com/config/mail?" might be a typical regex phrase.
4. Enter a name in the **Save macro as** box.
5. Click **OK**.

Web Proxy Menus

The Web Proxy menu bar contains the menus described in the following sections.

File Menu

The **File** menu contains the following commands:

- **New** - Deletes all session information.
- **Open** - Loads a proxy session file.
- **Save** - Saves the recorded sessions.
- **Save As** - Saves the recorded sessions.
- **Exit** - Closes Web Proxy.

Edit Menu

The **Edit** menu contains the following commands:

- **Select All** - Selects all sessions.
- **Clear Selected** - Deletes all selected sessions.
- **Settings** - Allows you to configure parameters for Web Proxy.

View Menu

The **View** menu contains the following commands:

- **Standard** - Displays only the standard toolbar.
- **Search** - Displays the standard and search toolbars.

Proxy Menu

The **Proxy** menu contains the following commands:

- **Start** - Starts Web Proxy.
- **Stop** - Stops Web Proxy.
- **Interactive** - Enables the interactive mode, which allows you to view requests and responses as the messages arrive at Web Proxy. See Web Proxy Interactive Mode on page 86 for more information.

Help Menu

The **Help** menu contains the following commands:

- **Web Macro Recorder Help** - Opens the Help file with the **Contents** tab active.
- **Index** - Opens the Help file with the **Index** tab active.
- **Search** - Opens the Help file with the **Search** tab active.
- **About Web Macro Recorder** - Displays information about the Web Macro Recorder.

Web Proxy Tabs

Each HTTP session (a single request and the associated response) is listed in the top pane of Web Proxy. When you select a session, Web Proxy displays information about the session in the lower pane. The information displayed depends on which tab you select.

Table 2 Web Proxy Tabs

Tab	Description
View	Use the **View** tab to select which HTTP messages you want to inspect. Options available from the drop-down list immediately below the tab are: **Session**: view the complete session (both request and response) **Request from browser to Web Proxy**: view only the request made by the browser to Web Proxy **Request to server from Web Proxy**: view only the Web Proxy request to the server **Response from server to Web Proxy**: view only the server response to Web Proxy **Response to browser from Web Proxy**: view only the Web Proxy response to the browser
Split	Click the **Split** tab to create two information areas for a single session. For example, you could show the HTTP request message created by the browser (in one area) and the HTTP response generated by the server (in the second area). You can cut, paste, and copy the raw request, and right-click to see a shortcut menu of encoding options. However, you cannot save an edited request from the Web Proxy tool. Use the HTTP Editor to save an edited request.
Info	Use the **Info** tab to view detailed information about the requests. Information includes the number of forms found, header information, and the properties of the page.
Browser	Click the **Browser** tab to view the response as formatted in a browser.

Web Proxy Settings

Click the **Edit** menu and select **Settings**.

➤ You cannot change settings while Web Proxy is running. Select **Stop** from the **Proxy** menu, change settings, and then restart Web Proxy.

Task 1: Configure General Settings

1 Select the **General** tab.

2 In the **Proxy Listener Configuration** group, enter an IP address and port number. By default, Web Proxy uses address 127.0.0.1 and port 8081, but you can change this if necessary.

Both Web Proxy and your Web browser must use the same IP address and port. If using Internet Explorer, click the **Tools** menu and select **Internet Options**; click the **Connections** tab and click **LAN Settings**; on the *LAN Settings* window, select **Use a Proxy Server for your LAN** and enter the address and port of the proxy server you want to use.

To configure Web Proxy on your host to be used by another host, you will need to change the value of the Local IP Address. The default address of 127.0.0.1 is not available to outside hosts. If you change this value to your workstation's current IP address, remote stations can use your workstation as a proxy.

3 Use the **Do Not Record** option to create a regular expression filter that prevents files of specific types from being handled by Web Proxy. The most common types are already excluded as defaults, but other types (MPEG, PDF, etc.) can also be excluded. The purpose is to allow you to focus on HTTP request/response lines and headers by removing clutter from the message.

4 When using the interactive mode, you can force Web Proxy to pause when it:

- Receives a request from the client.
- Receives a response from the server.
- Finds text that satisfies the search rules you create (using the **Flag** tab).

If you select any of these options, Web Proxy will continue only when you click the **Allow** button.

5 In the **Logging** group, select the type of items you want to record in the log file and specify the directory in which the log file should be maintained. If you elect to record requests and/or responses, you can also choose to convert and log the data using Base 64 encoding. This can be useful when responses contain binary data (such as images or flash files) that you want to examine.

- Raw Request refers to the HTTP message sent from the client to Web Proxy.
- Modified Request refers to the HTTP message sent from Web Proxy to the server.
- Raw Response refers to the HTTP message sent from the server to Web Proxy.
- Modified Response refers to the HTTP message sent from Web Proxy to the client.

6 In the **Advanced HTTP Parsing** group, select the character set that you assume is used for encoding the target site.

Task 2: Configure Proxy Servers Settings

1 Click the **Proxy Servers** tab.

Use this area to add one or more proxy servers through which Web Proxy will route all its requests. Distributing the attack across multiple servers makes detection and counter-measures more difficult, thus mimicking how a hacker might attempt to avoid an intrusion detection system.

If you use multiple proxy servers, Web Proxy will "round-robin" the requests (i.e., Web Proxy will sequence through the list of proxy servers, sending the first request to the first server, the second request to the second server, and so on).

2 In the **Proxy Address** box, type the IP address of the server through which you want to route Web Proxy requests.

3 Specify the port number in the **Proxy Port** box.

4 Select the type of proxy (standard, SOCKS4, or SOCKS5) from the **Proxy Type** list.

5 If this proxy server requires authentication, select an authentication type and enter your authentication credentials in the **Username** and **Password** boxes.

6 Click **Add** to add the server and display its IP address in the **Available Proxy Servers** list.

You can also import a file containing a list of proxy servers by clicking **Import**. The file containing proxy information must be formatted as follows:

- Each line contains one record followed by a line feed and carriage return.
- Each field in the record is separated by a semicolon.
- The fields appear in the following order: address;port;proxytype;user name;password.
- The user name and the password are optional. However, if authorization is not used, you must include two semicolons as placeholders.

Examples:

128.121.4.5;8080;Standard;magician;abracadabra

127.153.0.3;80;socks4;;

128.121.6.9;443;socks5;myname;mypassword

7 If you do not need to use a proxy server to access certain URLs (such as internal testing sites), you can specify one or more hosts in the **Bypass Proxy List** area.

a Click **Add** in the **Bypass Proxy List** group.

The *Bypass Proxy* window appears.

b Enter the host portion of the HTTP URL that should be bypassed.

Do not include the protocol (such as http://).

For example, to bypass a proxy server for this URL

http://zero.webappsecurity.com/Page.html

enter this string

zero.webappsecurity.com

or this string

zero.*

You can also enter an IP address. Note that Web Proxy will not resolve host names to IP addresses. That is, if you specify an IP address and the HTTP request actually contains that numeric IP address, then Web Proxy will bypass a proxy server for that host. However, if the HTTP request contains a host name that resolves to the IP address that you specify, Web Proxy will still send the request to a proxy server.

c Click **OK**.

Task 3: Configure Search-and-Replace Settings

1 Click the **Search and Replace** tab.

Use this tab to create rules for locating and replacing text or values in HTTP messages. This feature provides a highly flexible tool for automating your simulated attacks. Some suggested uses include:

- Masking sensitive data, such as user names and passwords
- Appending a cookie to each request
- Modifying the Accept request-header field to add or delete media types that are acceptable for the response
- Replacing a variable in the Request-URI with a cross-site scripting attack

2 Click **Add** to create a default entry in the table.

3 Click the **Search Field** column of the entry.

4 Click the drop-down arrow and select the message area you want to search.

5. In the **Search For** column, type the data (or a regular expression representing the data) you want to find.
6. In the **Replace With** column, type the data you want to substitute for the found data.
7. Repeat this procedure to create additional search rules.

   The request/response rules are applied sequentially, in the order in which they appear. For example, if a rule changes HTTPS to SSL, and if a subsequent rule then changes SSL to SECURE, the result will be that HTTPS is changed to SECURE.

Task 4: Configure Flag Settings

1. Click the **Flag** tab.

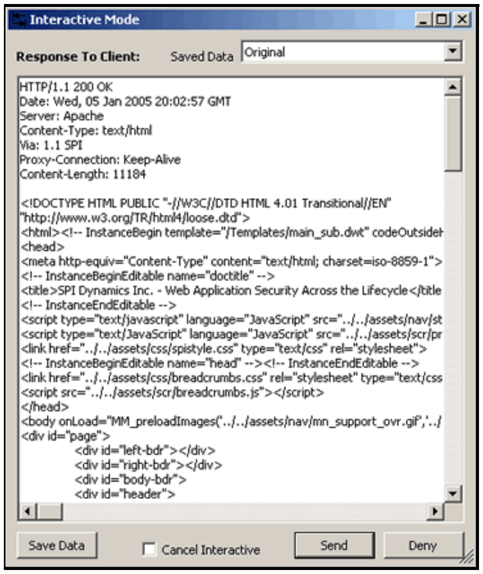
   This feature allows you to find and highlight keywords in requests or responses.
2. Click **Add** to create a default entry in the table.
3. Click the **Search Field** column of the entry.
4. Click the drop-down arrow and select the message area you want to search.
5. In the **Search** column, type the data (or a regular expression representing the data) you want to find.
6. Click the **Flag** column of the entry.
7. Click the drop-down arrow and select a color with which to highlight the data, if found.

Web Proxy Interactive Mode

Use Interactive mode to view each browser request and each server response as the messages arrive at Web Proxy. The message will not continue toward its destination until you click **Allow**. This permits you to modify the message before it is delivered.

You can also prevent the message from being sent to the server by clicking **Deny**.

Using the **Proxy** tab on the *Web Proxy Settings* window, you can force Web Proxy to pause after each request, after each response, or after locating specific text in either the request or response.



```
HTTP/1.1 200 OK
Date: Wed, 05 Jan 2005 20:02:57 GMT
Server: Apache
Content-Type: text/html
Via: 1.1 SPI
Proxy-Connection: Keep-Alive
Content-Length: 11184

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><!-- InstanceBegin template="/Templates/main_sub.dwt" codeOutsideH
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<!-- InstanceBeginEditable name="doctitle" -->
<title>SPI Dynamics Inc. - Web Application Security Across the Lifecycle</title
<!-- InstanceEndEditable -->
<script type="text/javascript" language="JavaScript" src="../../assets/nav/st
<script type="text/JavaScript" language="JavaScript" src="../../assets/scr/pr
<link href="../../assets/css/spistyle.css" type="text/css" rel="stylesheet">
<!-- InstanceBeginEditable name="head" --><!-- InstanceEndEditable -->
<link href="../../assets/css/breadcrumbs.css" rel="stylesheet" type="text/css
<script src="../../assets/scr/breadcrumbs.js"></script>
</head>
<body onLoad="MM_preloadImages('../../assets/nav/mn_support_ovr.gif','../
<div id="page">
        <div id="left-bdr"></div>
        <div id="right-bdr"></div>
        <div id="body-bdr">
        <div id="header">
```

Follow the steps below to turn on interactive mode:

1 Click the **Proxy** menu and select **Stop**.

2 Click the **Proxy** menu and select **Interactive**
-or-

click i on the toolbar.

3 Click the **Proxy** menu and select **Start**.

When Web Proxy is in Interactive mode, a check mark appears next to the Interactive command on the **Proxy** menu and the Interactive icon is backlit. Clicking the icon or selecting the command will toggle the Interactive mode on or off.

B HTTP Status Codes

Introduction

The following list of status codes was extracted from the Hypertext Transfer Protocol version 1.1 standard (rfc 2616). You can view the complete standard at http://www.w3.org/Protocols/rfc2616/rfc2616.html.

Table 3 HTTP Status Codes

Code	Definition
100	Continue
101	Switching Protocols
200 OK	Request has succeeded
201 Created	Request fulfilled and new resource being created
202 Accepted	Request accepted for processing, but processing not completed.
203 Non-Authoritative Information	The returned metainformation in the entity-header is not the definitive set as available from the origin server, but is gathered from a local or a third-party copy.
204 No Content	The server has fulfilled the request but does not need to return an entity-body, and might want to return updated metainformation.
205 Reset Content	The server has fulfilled the request and the user agent should reset the document view which caused the request to be sent.
206 Partial Content	The server has fulfilled the partial GET request for the resource.
300 Multiple Choices	The requested resource corresponds to any one of a set of representations, each with its own specific location, and agent-driven negotiation information (section 12) is being provided so that the user (or user agent) can select a preferred representation and redirect its request to that location.
301 Moved Permanently	The requested resource has been assigned a new permanent URI and any future references to this resource should use one of the returned URIs.
302 Found	The requested resource resides temporarily under a different URI.
303 See Other	The response to the request can be found under a different URI and should be retrieved using a GET method on that resource.
304 Not Modified	If the client has performed a conditional GET request and access is allowed, but the document has not been modified, the server should respond with this status code.

Table 3 HTTP Status Codes (cont'd)

Code	Definition
305 Use Proxy	The requested resource MUST be accessed through the proxy given by the Location field.
306 Unused	Unused.
307 Temporary Redirect	The requested resource resides temporarily under a different URI.
400 Bad Request	The request could not be understood by the server due to malformed syntax.
401 Unauthorized	The request requires user authentication. The response MUST include a WWW-Authenticate header field (section 14.47) containing a challenge applicable to the requested resource.
402 Payment Required	This code is reserved for future use.
403 Forbidden	The server understood the request, but is refusing to fulfill it.
404 Not Found	The server has not found anything matching the Request-URI.
405 Method Not Allowed	The method specified in the Request-Line is not allowed for the resource identified by the Request-URI.
406 Not Acceptable	The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request.
407 Proxy Authentication Required	This code is similar to 401 (Unauthorized), but indicates that the client must first authenticate itself with the proxy.
408 Request Timeout	The client did not produce a request within the time that the server was prepared to wait.
409 Conflict	The request could not be completed due to a conflict with the current state of the resource.
410 Gone	The requested resource is no longer available at the server and no forwarding address is known.
411 Length Required	The server refuses to accept the request without a defined Content-Length.
412 Precondition Failed	The precondition given in one or more of the request-header fields evaluated to false when it was tested on the server.
413 Request Entity Too Large	The server is refusing to process a request because the request entity is larger than the server is willing or able to process.
414 Request-URI Too Long	The server is refusing to service the request because the Request-URI is longer than the server is willing to interpret.
415 Unsupported Media Type	The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.

Table 3 HTTP Status Codes (cont'd)

Code	Definition
416 Requested Range Not Satisfiable	A server should return a response with this status code if a request included a Range request-header field (section 14.35), and none of the range-specifier values in this field overlap the current extent of the selected resource, and the request did not include an If-Range request-header field.
417 Expectation Failed	The expectation given in an Expect request-header field (see section 14.20) could not be met by this server, or, if the server is a proxy, the server has unambiguous evidence that the request could not be met by the next-hop server.
500 Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.
501 Not Implemented	The server does not support the functionality required to fulfill the request. This is the appropriate response when the server does not recognize the request method and is not capable of supporting it for any resource.
502 Bad Gateway	The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.
503 Service Unavailable	The server is currently unable to handle the request due to a temporary overloading or maintenance of the server.
504 Gateway Timeout	The server, while acting as a gateway or proxy, did not receive a timely response from the upstream server specified by the URI (e.g., HTTP, FTP, LDAP) or some other auxiliary server (e.g., DNS) it needed to access in attempting to complete the request.
505 HTTP Version Not Supported	The server does not support, or refuses to support, the HTTP protocol version that was used in the request message.

C Attacks and Methodologies

Introduction

A Web application includes not only the code that creates your Web site, but also the architectural components necessary to make a Web site available and useful to the public. When considering Web application security, you must account for all the components that work together to create a Web site, not just the visible face presented to the world at large.

DevInspect is a standalone software package that can analyze any Web application, identify potential security flaws, and supply you with the latest information necessary to resolve security issues before unauthorized users are able to capitalize on them. In an ever-changing, dynamic environment such as the Web, having a security tool that's always up to date is an absolute necessity. With this in mind, DevInspect's design team engineered the software to automatically update its built-in knowledgebase of known successful hacking methodologies every time it's used. The software will then emulate these methodologies against the applications to be tested. This knowledgebase is gathered from HP's security experts, as well as a wide variety of leading third-party security organizations and analysts.

When new methods of attack are discovered, DevInspect is ready with same-day upgrades to its SecureBase™ vulnerabilities database. DevInspect employs the following list of attacks and key methodologies when assessing the security vulnerabilities of your Web application:

- Parameter Manipulation
- Path Manipulation
- WebServer Assessment
- Site Searching
- Application Mapping
- Brute Force Authentication Attacks
- Content Investigation
- Known Attacks

Parameter Manipulation

Parameter manipulation involves tampering with URL parameters to retrieve information that would otherwise be unavailable to the user. Parameter manipulation modifies, adds or removes parameter names and/or arguments. Basically, any input can be modified. Parameter manipulation attacks can be used to achieve a number of objectives, including disclosure of files above the Web root, extraction of information from a database, and execution of arbitrary operating-system level commands. These attacks are directed specifically toward query strings, postdata, headers, and cookies.

Query strings

Web applications often use query strings as a simple method of passing data from the client and the server. Query strings are a way to add data calls to a hyperlink, and then retrieve that information on the linked page when it is displayed. By manipulating query strings, an attacker can easily steal information from a database, learn details about the architecture of your Web application, or possibly execute commands on your Web server. When conducting an audit, DevInspect implements advanced query string manipulation to determine the application's vulnerability to query string manipulation.

Postdata

Since manipulating a query string is as easy as typing text in the address bar of a browser, many Web applications rely on the POST method coupled with the use of forms rather than GET to pass data between pages. Since browsers normally don't display POST data, some programmers are lulled into thinking that it is difficult or impossible to change the data, when in fact the opposite is true. DevInspect will determine your application's susceptibility to attacks that rely on the POST method of parameter manipulation.

Headers

Both HTTP requests and responses utilize headers to deliver information about the HTTP message. A developer may not consider HTTP headers as areas of input, even though many Web applications will log headers such as the "referrer" or "user-agent" to a database for traffic statistics. DevInspect will intercept header information, and attempt to pass different parameter values during an audit.

Cookies

Many Web applications use cookies to save information (for example, user ID's and timestamps) on the client's machine. By changing these values, or "poisoning" the cookie, malicious users can gain access to the accounts and information of other users. As well, attackers can also steal a user's cookie and gain direct access to the user's account, bypassing the need to enter an ID and password or other form of authentication. DevInspect will list all cookies discovered during a scan, and attempt to change their parameters during an audit.

In general, parameter manipulation attacks belong to one of the following four categories:

- Injection
- Overflow

- Addition
- Deletion

Parameter Injection

Parameter injection attacks replace an argument value with an attack string.

Example:

The string

http://www.site.com/webapp.asp?ValidParameter=ValidArgument

will be changed to

http://www.site.com/webapp.asp?ValidParameter=AttackString

These attempts to manipulate parameters associated with a URL are usually directed to the following areas:

Command Execution

Command execution attack strings are composed of special characters combined with operating system-level commands that will be run if the Web application uses the string in a call to an operating system command without first parsing out the special characters.

Example:

In response to the following request

http://www.site.com/article.pl?id=;id;

a server might send data such as

uid=99(nobody) gid=99(nobody).

SQL injection

SQL injection attack strings are composed of fragments of SQL syntax that are executed on the database server if the Web application uses the string when forming a SQL statement without first parsing out certain characters.

Example: '

(SELECT TOP 1 name FROM sysobjects WHERE 1=1)+'

Directory traversal-Directory traversal attack strings are expressions that will cause the Web application to display the contents of files above the webroot if the Web application uses the string to specify a file location without first completely parsing out traversal characters.

Example:

In response to the following request

http://www.site.com/article.asp?id=../../../../../boot.ini

a server might send data similar to

:\boot.ini file contents

Cross-site scripting

This issue occurs when dynamically generated Web pages display input that is not properly validated. This allows an attacker to embed malicious JavaScript into the generated page, enabling him to execute the script on the machine of any user that views the malicious page. Any site that allows users to post text messages can be vulnerable to an attack such as this. This vulnerability is commonly seen on:

- Search engines that repeat back the search keyword that was entered
- Error messages that repeat back the string that contained the error
- Forms that are filled out where the values are later presented to the user
- Web Message boards that allow users to post their own messages.

An attacker who uses cross-site scripting successfully might compromise confidential information, manipulate or steal cookies, create requests that can be mistaken for those of a valid user, or execute malicious code on the user systems.

Example:

```
'<script>window.open('http://www.website.com')</script>
```

Abnormal input

Abnormal input attack strings are composed of characters that can cause unhandled exceptions in Web applications where unexpected input is not parsed out. Unhandled exceptions often cause error messages to be displayed that disclose sensitive information about the application's internal mechanics. Source code may even be disclosed.

Example:

```
%00
```

Hidden content access

Hidden content access strings are names or numbers that will cause a Web application to display hidden content, such as an administrative interface or unused portions of the website.

Example:

Given the query

http://www.site.com/index.php?ArticleID=2

if valid values for ArticleID range from 1-30, then changing the query to

http://www.site.com/index.php?ArticleID=99

might reveal a hidden portion of the website not normally accessed by users.

Untrusted application access

Untrusted application access will change existing values that are Boolean or single-digit numbers to other values in an attempt to gain access to an authenticated portion of the website.

Example:

Change this query

http://www.site.com/index.php?LoggedIn=false

to

http://www.site.com/index.php?LoggedIn=true

Format string attack

A format string attack will test the application for improper format handling on user input.

Example:

Changing this query

http://www.site.com/search.cgi?query=contact

to

http://www.site.com/search.cgi?query=%x%x%x%x

will break the application and allow the user to gain control of the webserver.

Parameter Overflow

Parameter overflow attacks supply Web applications with extremely large amounts of data in the forms of parameter or cookie header arguments or parameter names. If a Web application is programmed in such a manner that it cannot appropriately handle unexpected and extremely large amounts of data, it may be possible to execute arbitrary operating system-level code or cause a denial-of-service condition.

Numeric overflow

A numeric overflow increases an existing digit value to extremely high values in order to invoke an application error.

Example:

Changing this query

http://www.site.com/index.php?ArticleID=2

to

http://www.site.com/index.php?ArticleID=2147483648

will attempt to exceed the value of a signed integer.

String overflow

A string overflow increases the length of an alphabetic parameter in order to invoke an application error.

Example:

Changing this query

http://www.site.com/webapp.asp?ValidParameter=ValidArgument

by adding several thousand characters (as below)

http://www.site.com/webapp.asp?XXXXXXX…XXX=ValidArgument

will attempt to exceed the value of a signed integer.

Parameter Addition

Parameter addition attacks insert new parameters into an HTTP request. Parameter addition attacks can be used to gain access to restricted or undocumented application features, manipulate internal application settings.

Application debug/backdoor modes

These parameters are often undocumented application features that are added by programmers in order to assist with quality assurance. Debug and backdoor modes access can lead to disclosure of sensitive information about the internal mechanics of the Web application or even administrative control.

Example:

http://www.site.com/
webapp.asp?ValidParameter=ValidArgument&debug=true

Internal parameter specification

These attacks define parameters and arguments in the HTTP request that are only supposed to be set inside of the Web application. If the Web application sets the variable specified in the parameter to the value specified in the argument, internal application settings will be altered.

Example:

http://www.site.com/
webapp.php?ValidParm=ValidArgument&DBUsername=sa

Parameter Deletion

Parameter deletion removes a parameter value from query data or cookie headers. If the Web application relies on the presence of the removed parameters for basic functionality, and unhandled exception may occur. Unhandled exceptions often cause error messages to be displayed that disclose sensitive information about the application's internal mechanics. Source code may even be disclosed.

Path Manipulation

Path manipulation attacks construct or modify the Request-URI section of the HTTP request in order to gain access to files above the webroot, bypass authorization settings, display directory listings or display file source.

Path truncation

Path truncation attacks are requests for known directories without filenames. This may cause directory listings to be displayed.

Example:

Given the following URL

http://www.site.com/folder1/folder2/file.asp

truncating the path to look for

http://www.site.com/folder1/folder2/

and

http://www.site.com/folder1/

may cause the webserver to reveal directory contents or to cause unhandled exceptions.

Case sensitivity

Case sensitivity attacks change the case of the characters in the filename in an attempt to change the manner in which the request is processed. If the Web application performs a case-sensitive string comparison for authorization or processing purposes this may defeat authorization settings or cause source code to be disclosed.

Character encoding

Character-encoding attacks substitute encoded equivalents of characters in a request for a known resource. If the Web application performs a string comparison for authorization or processing purposes using the encoded URI without parsing the encoded characters first, authorization settings maybe defeated or source code may be disclosed.

- **Unicode**: The Unicode Worldwide Character Standard includes letters, digits, diacritics, punctuation marks, and technical symbols for all the world's principal written languages, using a uniform encoding scheme. DevInspect submits strings that have been converted to their Unicode equivalent, and attempts to gain unauthorized authentication credentials through this manipulation.
- **Hex-encode**: Hex-encoding involves replacing strings with their hexadecimal equivalent. DevInspect submits hexadecimal-encoded strings, and attempts to gain unauthorized authentication credentials through this manipulation.

MS-DOS 8.3 Short Filename

These attacks convert the filenames to the MS-DOS 8.3 format. If the Web application performs a string comparison for authorization or processing purposes using the MS-DOS 8.3 filename without first converting it to its FAT32/NTFS equivalent, this may defeat authorization settings or cause source code to be disclosed.

```
Example:
The file named
longfilename.asp
would become
longfi~1.asp
```

Directory traversal

These attacks are expressions in the URI that will cause the Web server to display the contents of files above the webroot if the Web application uses the string to specify a file location without first completely parsing out traversal characters.

```
Example:
../../../../../boot.ini
```

Character stripping

These attacks add special characters to a URI that the server or application may parse out. If the server or application uses the URI in a string comparison for authorization or request processing without first stripping out the special characters authorization settings may be defeated and source code may be disclosed.

- Character append attacks add a special character to the end of a file or directory name. For example, file.asp would become file.asp%00.
- Character prepend attacks add a special character to the beginning a file or directory name. For example, /protecteddirectory/file.asp would become /%00protecteddirectory/file.asp.
- Buffer truncation attacks add a large number of characters to a filename and extension followed by another extension. If a Web application determines how the request should be processed based on the file extension, it will protect against a buffer overflow by truncating the request at a certain length and then stripping out the special characters, in that order. Authorization settings may be bypassed and source code may be displayed using the following example:

```
file.asp%20%20%20%20%20%20%20[many %20s].txt
```

Webserver Assessment

During a Web server assessment, DevInspect tests your proprietary Web server for vulnerabilities utilizing information gathered during a site search and other applied methodologies. Protocol and extension implementation analysis is used to determine what services the server offers, whether or not they conform to established standards for these services, and details regarding their implementation. As Web server configurations are responsible for serving content and launching applications, damage from an attack on an unprotected proprietary Web server can include denial of service, the posting of inappropriate messages or graphics on the site, deletion of files, or damaging code or software packages being left on the server.

HTTP compliance

HTTP compliance testing assesses the Web server or proxy server for proper compliance to HTTP/1.0 and HTTP/1.1 rules. This testing consists of attacks such as sending a data buffer larger then the marked length (buffer overflows). Servers are tested to see if they properly sanitize data by mixing and matching various methods and headers that are never seen within a normal request and determining if the Web server handles the requests properly. These attacks can determine if a Web server or Web device complies with HTTP specifications and can also uncover unknown vulnerabilities.

WebDAV compliance

WebDAV allows users to place and manipulate files in a directory on your Web server. DevInspect will ascertain whether or not WebDAV privileges can be exceeded and manipulated on your Web server.

SSL strength

SSL strength identification determines the encryption level accepted by a Web server. This can be important to ensure that secure clients do not connect at an encryption level lower then the expected standard, and that data is being properly encrypted to prevent its interception.

Certificate analysis

DevInspect analyzes the SSL certificate for improper properties such as unknown CA certificate analysis or expired time.

HTTP Method Support

DevInspect determines what HTTP methods are supported by the Web server.

Site Searching

This can be considered the information gathering stage, much as an attacker would learn as much as possible about your Web application before launching an attack. Site search is used to locate resources such as documents, applications and directories on the server that are not intended to be viewed by Web users. Disclosure of such resources can result in the disclosure of confidential data, information about internal server and application configurations and settings, administrative access to the site, and information and application source code. DevInspect determines the availability of the following items, among others, to users of your Web application.

Test files

Test files often contain information that can be used to implement an attack. For example, authenticated test scripts that have been left on the server could provide an attacker with the location of sensitive areas of your site.

Administrative interfaces

Administrative interfaces are applications that network administrators often place on a network to conduct remote maintenance.

Program dumps

When a program terminates prematurely, it often leaves a dump file on the server (an image of system memory when the program stops executing). Attackers will often break an application through various methods and then retrieve important information from a dump file.

Application logs

Several software applications leave default application logs that detail the installation of the product. Application logs can reveal important information about the architecture of your Web application, including the location of hidden areas.

Installation documentation

Certain software packages place comprising information in default installation documentation that is left available on the server.

Backup files

Network administrators and developers often leave backup files and scripts on the Web server. These files commonly contain information that can be used to breach a site's security. Backup file search involves replacing extensions on files, and then looking for older or backup versions stored on the site. For example, an attacker who finds hi.asp might search for hi.old and hi.back, and retrieve the script's source code.

Site statistics pages

A Site Statistics page can be used to determine information about who is visiting your site. However, it can also reveal information that an attacker can use in formulating an attack, such as the location of other areas of your site.

Application Mapping

DevInspect exposes and follows all known (and unknown) links located on your site. This creates a baseline for vulnerability checking and application testing.

Crawl

One of the most important elements of discovering the security vulnerabilities of your Web application is in mapping its internal structure. A DevInspect crawl completely maps a site's tree structure. In essence, a crawl runs until no more links on the URL can be followed.

Automatic form filling

DevInspect can be configured to submit data automatically for any form encountered during a crawl (for example, if a page requires entry of a telephone number, etc.).

SSL support

DevInspect can crawl any site that uses SSL and determine whether data is being properly encrypted and protected.

Proxy support

A proxy server can be used to ensure network security, provide adequate caching purposes, and regulate administrative control. DevInspect can crawl sites that use a proxy server, and check for vulnerabilities specifically related to that configuration.

Client certificate support

A certificate is a statement verifying the identity of a person or the security of a Web site. Attackers will attempt to alter the values of client certificates to gain unauthorized access to your Web application.

State management

State is a property of connectivity. HTTP is a stateless protocol; no concept of session state is maintained by HTTP when handling client-server communications. DevInspect determines if any cookies used on your Web application are secure (are they set to expire, properly handled, etc.), and if session IDs are managed securely.

Directory enumeration

Directory enumeration lists all directory paths and possibilities on the application server, including hidden directories that could possibly contain sensitive information. DevInspect uses a database of known folders (such as admin, test, logs, etc.) and hidden areas discovered during a crawl when composing a directory enumeration listing.

Brute Force Authentication Attacks

This test determines if users are employing usernames and passwords that an unauthorized intruder might be able to guess easily. For example, it will discover if an authorized user is accessing a Web site by entering a username of “customer” and a password of “password”; the Web administrator could then warn that user about the susceptibility and suggest changing the password and/or username.

Web Brute will attempt a “brute force” attack of three authentication types: basic HTTP, NTLM, and forms on Web pages.

Content Investigation

Content Investigation involves searching through content discovered during a site search to determine what information available to users of your Web application should remain private. DevInspect searches for the following items when conducting Content Investigation (although by no means a comprehensive list), and will determine each item’s potential level of exploitation.

- **Spam Gateway Detection**: Spam gateways are e-mail Web applications that allow the client to specify the location of the mail recipient via hidden form inputs or parameters.
- **Client-Side Pricing**: Client-side pricing is a Web application flaw that allows the client to specify item pricing via hidden form inputs or parameters.
- **Sensitive Developer Comments**: Developer comments in HTML often reveal sensitive information about an application’s internal mechanics and configuration. For example, something as seemingly innocuous as a comment referencing the required order of fields in a table could potentially give an attacker a key piece of information needed to crack the security of a site.
- **WebServer/Web Package Identification**: DevInspect will identify all services and banners on the Web server, and ascertain the vendors and version numbers of all available software packages used by your Web application. This is accomplished through header evidence, link evidence, and default/template page evidence.
- **Absolute Path Detection**: DevInspect detects if a fully qualified pathname was able to be discovered anywhere within an application. Certain vulnerabilities can only be exploited if the attacker has the fully qualified pathname.
- **Error Message Identification**: Often, error messages will reveal more than they were designed to do. For example, pages containing /servletimages/logo2circle.gif are default template BEA Weblogic error pages. An attacker forearmed with that knowledge can customize his attack to take advantage of that server’s inherent vulnerabilities.

- Permissions Assessment: DevInspect will determine what level of permissions (such as uploading files to the Web server, editing data, traversing directories, etc.) are available in different areas of your Web application, and then determine the best way to remedy any inherent security vulnerabilities.
- **Session Hijacking**: DevInspect sends multiple requests in order to gain multiple sessions and analyze the session ID within the URLs or cookies for weaknesses.
- **Non-Restrictive Search Engine**: Determining if a search engine's search scope is unrestricted. Many search engines have access to search the entire webroot or the entire drive of a Web server. If this is available, a search for "admin" or "adduser" will return links to the administrative portion of the Web site.

Known Attacks

Known attacks include all exploitable holes and bugs in Web servers, applications, and other third-party components that have been published, posted, or otherwise communicated. Most of these vulnerabilities have existing patches, but hackers will exploit systems where patches have not been installed in a timely fashion. Known attack information is included in all other methodologies. DevInspect relies on a proprietary database, which contains fingerprints of known attacks dating back to the birth of the World Wide Web. DevInspect downloads checks for new risks and exploits each time customers run Smart Update, ensuring that the product is always at the forefront of hacking expertise.

Glossary

Audit

To assess your Web application for security vulnerabilities.

Authentication

Identity verification, typically through the use of logon passwords. Authentication precedes authorization. DevInspect supports Basic, NTLM, and Web-based (form) authentication.

Authorization

Access control. After a user has been authenticated (proven their identify, typically via a logon password), the operating system or application must identify what resources the user can access during this session, and authorize access accordingly.

Banner

Server identification. An attacker will grab banners to determine the make and model of the server operating system, and use that information when formulating an attack against the vulnerabilities of that software package.

Basic Authentication

A widely used, industry-standard method for collecting user name and password information.

- The Web browser displays a dialog box for a user to enter a previously assigned user name and password, also known as credentials.
- The Web browser then attempts to establish a connection to a server using the user's credentials.
- If a user's credentials are rejected, the browser displays an authentication dialog box to re-enter the user's credentials. Internet Explorer allows the user three connection attempts before failing the connection and reporting an error to the user.
- If the Web server verifies that the user name and password correspond to a valid user account, a connection is established.

The advantage of Basic authentication is that it is part of the HTTP specification and is supported by most browsers. The disadvantage is that Web browsers using Basic authentication transmit passwords in an unencrypted form. By monitoring communications on your network, an attacker can easily intercept and decode these passwords using publicly available tools. Therefore, Basic authentication is not recommended unless you are confident that the connection between the user and your Web server is secure.

Canonicalization

Sanitizing data by not accepting improper input. For example, stripping special characters from a request before processing it.

Cross-site scripting

This issue occurs when dynamically generated Web pages display input that is not properly validated. This allows an attacker to embed malicious JavaScript into the generated page, allowing the attacker to execute script on the machine of any user that views the malicious page. Any site that allows users to post text messages can be vulnerable to an attack such as this.

Client

A client is the requesting program or user in a client/server relationship. For example, the user of a Web browser is effectively making client requests for pages from servers all over the Web. The browser itself is a client in its relationship with the computer that is getting and returning the requested HTML file. The computer handling the request and sending back the HTML file is a server.

Crawl

The process by which DevInspect identifies the structure of the target Web site. This is usually followed by an audit, which is the actual vulnerability assessment. A crawl and an audit, when combined into one function, is termed a scan.

Cookie

Cookies are information stored by a server on a client for future use (such as user preferences, configuration information, etc.). Cookies appear in two basic forms, as individual files or as records within one contiguous file. Often, there are multiple sets, the result of multiple browsers being installed in differing locations. In many cases, it is the forgotten cookies that contain the revealing information that you would prefer others not see.

HTTP

Hyper Text Transfer Protocol. HTTP is the set of conventions that governs how HTML documents are transmitted and received across the World Wide Web. When browsing Web sites, your Web browser is a client program that makes requests (for example, that a certain Web page be displayed) from a Web server somewhere on the Internet. An important element of HTTP is in how servers (the computers hosting the Web applications, in this instance) handle requests from clients (remote computers connecting to the server via the World Wide Web). A session can be defined as the matched pair of a client request and a server response. HTTP is a stateless protocol-no concept of session state is maintained by HTTP when handling client-server communications. While that sounds complicated, it is really quite simple when broken down. Each request made by a client is handled individually by a server. Multiple requests made by the same client are each treated as unique by the responding server. In other words, the server does not attempt to maintain a connection with the client at any time.

IDE

Integrated Development Environment; a programming environment integrated into an application.

IDS

Intrusion Detection System. This type of system supplements perimeter security applications (such as a firewall) and identifies attacks that have passed through those defenses.

Image map

In Internet development, an image map is a graphic defined so that different areas of the image are linked to different destinations.

Login Macro

This type of macro is used for Web form authentication. You can also incorporate logic that will prevent DevInspect from terminating prematurely if it inadvertently logs out of your application.

NTLM

NTLM (NT LanMan) is an authentication process that is used by all members of the Windows NT family of products. Like its predecessor LanMan, NTLM uses a challenge/response process to prove the client's identity without requiring that either a password or a hashed password be sent across the network.

Parameter

An item of information, such as a name, a selection, or a number, passed to a program by another program or an end-user.

Policy

The set of vulnerability checks and attack methodologies that DevInspect will deploy against a Web application.

Proxy server

In Internet terminology, a proxy server is one that serves as an intermediary between a workstation user and the actual Internet. Requests for Internet services made by the client (the workstation) must pass through the proxy server, as also do the Web server responses. A proxy server can be used to ensure network security, provide adequate caching purposes, and regulate administrative control.

Query string

The extra bit of data in the URI after the question mark that is used to pass variables. The query string is used to transfer data between the client and the server. Web applications often use query strings as a simple method of passing data from the client and the server. Query strings are a way to add data calls to a hyperlink, and then retrieve that information on the linked page when it is displayed. By manipulating query strings, an attacker can easily steal information from a database, learn details about the architecture of your Web application, or possibly execute commands on your Web server.

Scan

A generic term for the assessment of a Web site or enterprise. The actual task may be either a crawl, audit, or a combined crawl and audit.

Server

In the Web application client/server model, a server (a program housed in a computer) uses HTTP to serve files that form Web sites to users. The user's system contain an HTTP client (e.g. the Web browser) that forwards requests to the Web server, which responds with the appropriate data. Two leading Web servers are Apache and Microsoft's Internet Information Server (IIS).

Session

A session is a matched set containing both the client request and server response. For Internet applications, each session is associated with a particular port.

Session hijacking

Allows an attacker to masquerade as another user and gain access to Web service without having to authenticate. By using session hijacking, an attacker has access to the Web application with permissions of the original user.

Session ID

Generally, successful authentication credentials stored so that a user does not have to enter them repeatedly. Since the session ID can be used instead of a username and password combination, an attacker who discovers and provides a valid session ID in a request could perform session hijacking or replay attacks.

Session state

HTTP sessions are stateless. HTTP is a stateless protocol-no concept of session state is maintained by HTTP when handling client-server communications. While that sounds complicated, it is really quite simple when broken down. Each request made by a client is handled individually by a server. Multiple requests made by the same client are each treated as unique by the responding server. In other words, the server does not attempt to maintain a connection with the client at any time.

Sink

In source code, the point at which possibly tainted data is used.

SOAP

Simple Object Access Protocol. SOAP uses HTTP and XML as the means to exchange information so that programs on one platform (for example, Windows XP) can communicate with a program on the same or a different operating system (such as Linux).

Source

In source code, the point at which possibly tainted data enters the system.

SQL injection

The act of passing SQL code not intended by the developer into an application. For example, problems can arise when a developer does not protect against potentially malicious input such as an apostrophe, which could close the SQL string and give the user unintended system and application access.

Start Macro

This type of macro is used most often to focus on a particular subsection of the application. It specifies URLs that DevInspect will use to navigate to that area. It may also include login information, but does not contain logic that will prevent DevInspect from logging out of your application.

Step mode

Step mode captures each click followed on the site and then develops the site structure. Once you have completed clicking through the site, click **Audit** to assess the security vulnerabilities of the site.

String

A block of values or symbols, such as a character string (a sequence of alphanumeric characters), or a binary string (a sequence of binary values).

Trojan

A Trojan horse attack is the programming equivalent of the wooden horse given to the city of Troy. Seemingly benign data or programming is used to hide malicious or harmful code in such a way that it can instigate its chosen form of damage without your knowledge.

URI

Uniform Resource Identifier. According to the World Wide Web Consortium, Internet space is populated by many points of content. URIs are the method used to locate any given point of content on the Internet, whether it be a Web page, a video or music file, a program, or a graphic image. A URL (Uniform Resource Locator) is a particular form of URI, and is used as a designation for a Web page address. Typically, a URI describes:

- The process used to access the content
- The specific computer that stores the content
- The specific name of the content (i.e. the file name) on the computer

For example, the URI

> http://www.spidynamics.com/graphics/home/spi_logo.gif

designates a file that can be accessed via HTTP that is housed on a computer named "www.spidynamics.com" (which can be mapped to a unique Internet address). In the directory structure of that computer, the file is located at the pathname of /graphics/home/.

URL

Uniform Resource Locator. An HTTP URL can be for any Web page or individual file.

Webroot

In a computer file system organized in a hierarchical or tree structure, the root directory is the directory that includes all other directories (i.e. C:\). For Web sites, the webroot is the uppermost level of the tree hierarchy of the site.

Web form authentication

Many Web applications contain HTML forms that a user must complete successfully before being allowed to access the remainder of the application. Typically, the user types a "username" in a single-line text input control and "password" in a password control, and then submits the form to a server-based agent for processing.

Index