HP DevInspect for .NET Software

for the Windows® operating system

Software Version: 5.1

User Guide

Document Release Date: June 2008
Software Release Date: June 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, Windows, and Windows XP 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: 1.800.633.3600; select option 2 for Software Support, enter your Service Agreement ID (SAID) number, choose option 1 for enterprise application software assistance, and then option 5 for former SPI Dynamics products.

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

To register for an HP Passport ID, go to:

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

Contents

1 Welcome to DevInspect

Introduction

DevInspect accelerates the construction and delivery of secure Web applications by identifying and fixing vulnerabilities without leaving the Visual Studio integrated development environment. For the first time, Microsoft Visual Studio developers can build secure Web applications quickly and easily, without the need for specialized security knowledge and without risking time-critical product release schedules.

DevInspect delivers the latest evolution in assessment technology, a Web application security product that adapts to any enterprise environment. As you initiate an assessment, DevInspect assigns “assessment agents” that dynamically catalog all areas of a Web application and report their findings to a main security engine that analyzes the results. DevInspect then launches audit engines to evaluate the gathered information and apply attack algorithms to locate vulnerabilities and determine their severity. With this smart approach, DevInspect continuously applies appropriate assessment resources that adapt to the specific application environment.

System Requirements

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

- Windows XP SP2 or Windows Server 2003 SP1
- 1 GB of RAM
- 150 MB of free disk space required (2 GB preferred)
- 1 GHz processor or better
- Microsoft Visual Studio 2005 Standard (or better) or Microsoft Visual Studio 2008 Standard (or better)
- Microsoft .NET Framework 2.0
- Microsoft SQL Server Express SP1 (or better)

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

The minimum screen resolution for DevInspect is 800 x 600. For best performance, use a screen display of 1024 x 768.

HP also recommends that you disable (or otherwise avoid using) URL rewriting when conducting dynamic analysis.

You should also set the virtual memory paging file size to 2046MB (Initial) and 4092 MB (Maximum). Use the following procedure to change this setting for the Windows XP operating system.

1. Click **Start** and select **Control Panel**.
2. Select **System**.
3. On the System Properties window, click the **Advanced** tab.
4. In the **Performance** group, click **Settings**.
5. On the Performance Options window, click the **Advanced** tab.
6. In the **Virtual Memory** group, click **Change**.
7. Select the **Custom Size** option.
8. In the **Initial size** box, enter 2046.
9. In the **Maximum size** box, enter 4092.
10. Click **Set**.

Main Features of DevInspect

DevInspect features the deepest and most intuitive security tool integration with Visual Studio. 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 Visual Studio IDE.

Accuracy and Precision

- Combines dynamic and source code analysis to achieve unmatched confidence in the results.
- Analyzes security of application configuration.
- Bases vulnerability risk ratings on impact and probability.
- Provides a real-time security view of an application.
- Permits daily vulnerability data updates through Smart Update 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 or step-mode (browser-based) scans.
- Allows customization of security policies.
- Complements Visual Studio code analysis.

SecureObjects Remediation

- Pinpoints areas of vulnerability in the application.
- Automatically fixes vulnerable code.
- Identifies and secures all applications.
- Corrects insecure application configuration.
- Fixes vulnerabilities exposed through third-party components.

Application Self-Defense

- Combines “white list” and “black list” validation.
- Detects and prevents several types of attacks, including SQL injection, cross-site scripting, and buffer overflow.
- Informs operations through ASP.NET health monitoring when attacks are detected.
- Protects against “brute force” attacks using BruteProtector validator.

Education for Developers

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

Visual Studio Integration

- Integrates seamlessly with Visual Studio as well as the Microsoft Team Foundation Server.
- Specifically targets the security of ASP.NET applications.
- Supports C#, Visual Basic, HTML, XML, SOAP, WSDL, JavaScript, VB Script.

Integrated Security Tools

- Web Macro Recorder
- Web Form Editor
- HTTP Editor
- Web Proxy
- SOAP Editor
- Regular Expression Editor
- Encoder/Decoder

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

New Features in Version 5.1

DevInspect for Visual Studio version 5.1 includes the following new features:

Correlation of Static and Dynamic Analysis

Hybrid Analysis combines both static (white-box) and dynamic (black-box) analysis. These results are now correlated to give users maximum assurance that vulnerabilities are real and allows developers to prioritize fixes more efficiently.

Scrubber Support

DevInspect now supports the use of predefined "scrubber" functions that protect against SQL injection and cross-site scripting. These ASP.NET framework functions are:

- Server.HtmlEncode(input)
- HttpUtility.HtmlEncode(input)
- RegEx.IsMatch

They are assumed to eliminate vulnerabilities in the code, but may still be found vulnerable through dynamic analysis. For this reason, hybrid analysis is the only way to confirm that the validation functions work as designed.

Static Analysis Engine Upgrade

- Five new data sink checks added for existing checks.
- Ten new input source methods validated with "Unvalidated Data Usage" checks

Features Released in Version 5.0

DevInspect for Visual Studio version 5.0 included the following new features:

Visual Studio 2008 Support

DevInspect can now be used in Visual Studio 2005 and Visual Studio 2008. This includes support for projects using Microsoft .NET Framework versions 2.0, 3.0, and 3.5.

Vista Support

DevInspect and its installer now support Windows Vista. Minor changes were made to handle development using IIS 7.

Upgraded Static Analysis Engine

Cross-site scripting and SQL injection can now be traced to the line of code. The Unvalidated Input Data check has been introduced to detect any use of data that has not been validated properly within the code.

Upgraded Dynamic Scan Engine

The dynamic scan engine used for black-box testing and shared among all ASC products has been updated to the latest version to improve speed, accuracy and latest security tests.

Licensing Improvements

Licenses can be deactivated, and a license expiration warning dialog is now displayed.

Universal Schema Format

DevInspect now exports scan data in Universal Schema format so that it can be imported into other HP ASC products.

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

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 form values used by DevInspect. You can determine these values from the Web Form Values properties of the Web site or page (in the DevInspect Explorer window).

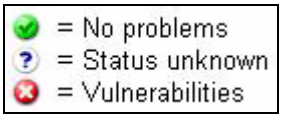
If for any reason you do not want DevInspect to crawl and attack certain directories, you must specify those directories using the Excluded URLs feature of DevInspect settings.

Configure the User Interface

Follow the steps below to configure Microsoft Visual Studio for using DevInspect:

1. Open a Web site or Web application project.
2. If the DevInspect Explorer window is not visible, click the **View** menu and select **DevInspect Explorer**.

The DevInspect Explorer displays a hierarchical (tree structure) view of your Web site's architecture. Each file or page is marked with an icon that designates its status as determined by DevInspect.



Initially, DevInspect designates all components as Status Unknown and marks them with the question mark. After analyzing a component, DevInspect marks it with the appropriate icon, as illustrated below.

DevInspect Explorer Toolbar

The DevInspect Explorer toolbar contains the following buttons.

Table 1 DevInspect Toolbar

Button	Name/Function
	Analyze Selected Web Site/Project: Builds the Web site, performs static analysis, and begins the dynamic analysis (vulnerability scan) of your application.
	Stop Analysis: Halts the scan and aborts the procedure.
	Export Results: Records the raw scan data in Universal Schema format and saves it in a compressed (.zip) file.
	Upload Scan to AMP: Transfers the scan to the AMP database.
	Create Report: Generates a report of the scan results and saves it in a PDF format.

Table 1 DevInspect Toolbar (cont'd)

Button	Name/Function (cont'd)
	Refresh: Refreshes the content of the DevInspect Explorer. Icons indicating the status of components are reset to "Unknown."
	Smart Update: Downloads the latest adaptive agents and software releases, as well as vulnerability and policy information.

Update SecureBase

You should update the product each time you use it. Hewlett-Packard engineers uncover new vulnerabilities nearly every day. They develop attack agents to search for these malicious threats and then update the corporate database so that you will always be on the leading edge of Web application security.

Follow the steps below to ensure that you have up-to-date information about DevInspect vulnerabilities:

1. On the toolbar in the DevInspect Explorer, click  .
2. If a more recent version of DevInspect is available for downloading from the Hewlett-Packard server, the Smart Update tool will ask if you want to install it (along with the database updates).
3. If updating only the SecureBase, Smart Update displays a dialog box listing the number and types of updates available. Click **Update**.

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's components. For detailed information and instructions, see Chapter 3, DevInspect Options.

Integrate with a Team Foundation Server

Use the following procedure to integrate DevInspect with a Microsoft Team Foundation Server.

These features, although not required, will enhance and maximize the centralized control and management of software vulnerabilities.

Task 1: Create a Project

Create a Team Foundation Server (TFS) team project using the process template "MSF for Agile Software Development - 4.0 (SPI)." If this template is not available:

1. Click the Windows **Start** button, point to **All Programs**, and select **HP**.

2 Select **DevInspect for .NET** and select **Load Process Template for 2005/2008**.

For existing projects, you should create a new team project and then change the bindings for source control on the existing project. Among other features, this introduces a "Vulnerability" work item to the TFS environment.

Task 2: Use a Check-In Policy

This feature allows you to create a check-in policy that monitors your project for security vulnerabilities.

1 On the Team Explorer panel within Visual Studio, right-click the team project name and select **Team Project Settings**.

2 From the submenu, select **Source Control**.

The *Source Control Settings* dialog appears.

3 Click the **Check-in Policy** tab.

4 Click **Add**.

5 Select the DevInspect Secure Code Policy and click **OK**.

The *Secure Code Policy Settings* dialog appears.

6 Select either **Low**, **Medium**, **High**, or **Critical**.

If you attempt to check in project items containing vulnerabilities that are rated by DevInspect as equal to or higher than the category you select, the TFS system displays a warning. You can cancel the process or you can override the policy and continue.

Task 3: Add Web Parts to TFS Portal

Follow the steps below to add Web parts to your Team Foundation Server portal.

1 On the Team Explorer panel within Visual Studio, right-click the team project name and select **Show Project Portal**.

2 On the Project Portal Web page, click **Modify Shared Page**.

3 From the cascaded menu, click **Add Web Parts** and then select Browse.

The Add Web Parts panel appears.

4 Select **Virtual Server Gallery**.

5 Drag DevInspect App View and DevInspect Summary, and drop them on the portal page.

The DevInspect App View shows the vulnerability status of all items, by project.

The DevInspect Summary displays all active, closed, and resolved vulnerabilities categorized by developer, project, or Web site.

Create a Table of Values for Forms

Most Web applications incorporate 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 entering a user name and password in 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.

How to Launch the Web Form Editor

Follow the steps below to launch the Web Form Editor:

1. Open a Web site or Web application project.
2. Can you see the DevInspect Explorer window or its tab?

   Yes: Go to Step 3.

   No: Click the **View** menu and select **DevInspect Explorer**.
3. Select the DevInspect Explorer window to display a hierarchical list of the site or project components.
4. Click the root.
5. In the **Properties** pane, select **Web Form Values** and click  .

   The Web Form Editor opens.
6. See Web Form Editor on page 53 for specific instructions.

3 DevInspect Options

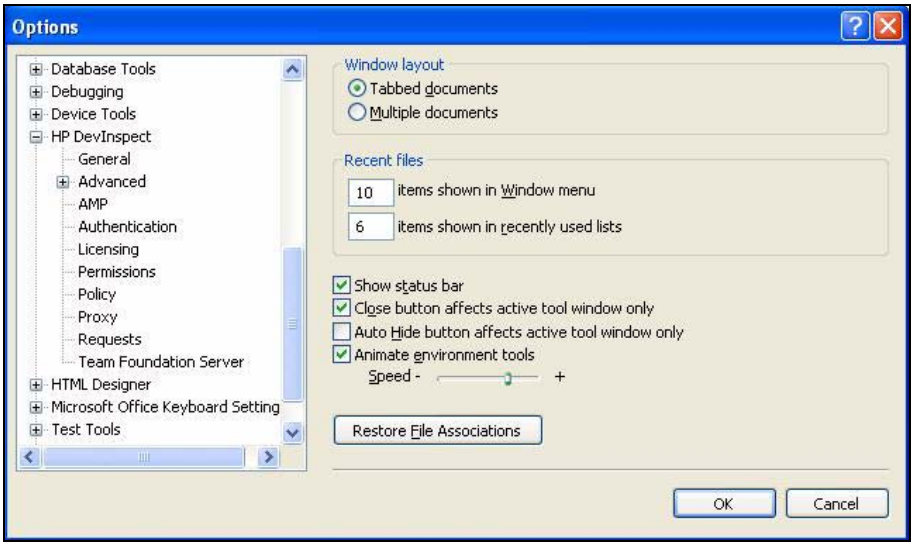
Overview

You can tailor DevInspect to your development environment by adjusting the options that control how your components are analyzed. These settings are categorized as follows:

- General
- Advanced
- AMP
- Authentication
- Licensing
- Permissions
- Policy
- Proxy
- Requests
- Team Foundation Server

Follow the steps below to access DevInspect options:

1. Click the Visual Studio **Tools** menu and select **Options**.
2. Expand the **HP DevInspect** node.
3. Select a settings category.



When you select a category in the left pane, related information displays in the right pane.

General Options

General options are described in the following table.

Table 2 General Options

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 options. By default, vulnerabilities are designated as warnings in the Error List. **Never**: No vulnerabilities marked as errors. **Critical Only**: Critical vulnerabilities marked. **High or higher**: High and critical marked. **Medium or higher**: Critical, high and medium marked. **Always**: All vulnerabilities marked.
Logging level	DevInspect logs activities at five different levels, ranging from most to least verbose: **Debug** — Most verbose **Info** — Informative events and those more severe **Warn** — Warnings events and those more severe **Error** — Errors and fatal events **Fatal** — Fatal events only You can also elect to turn off logging by selecting **None**.
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 Visual Studio startup	If you select this option, DevInspect deletes all scan database files each time you launch Visual Studio.
Delete now	Click this button to remove all files containing scan results.
Include remote scan results	Select this option to perform dynamic analysis of applications running in remote environments (that is, outside the current Visual Studio project) if your DevInspect license permits.

Advanced Options - Custom Request Elements

Custom request element options are described in the following table.

Table 3 Advanced Options - Custom Request Elements

Option	Description
Custom cookies	Use the Custom Cookies area to specify data that will be sent with the Cookie header in request messages sent by DevInspect to the server when conducting a vulnerability assessment. Follow the steps below to add a cookie: 1 From the Visual Studio **Tools** menu, select **Options**. 2 In the Options tree, select **HP DevInspect** and click **Advanced**. 3 Select **Custom Request Elements**. 4 To add a cookie, type or paste the cookie name/value pair in the **Value** box and click **Add**. The format is: CookieName=Value. 5 To delete a custom cookie, select it from the **Custom Cookies** area and click **Delete**.
Custom headers	Use the Custom Headers area 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. Follow the steps below to add a custom header: 1 From the Visual Studio **Tools** menu, select **Options**. 2 In the Options tree, select **HP DevInspect** and click **Advanced**. 3 Select **Custom Request Elements**. 4 To add a header, type or paste the header text in the **Value** box and click **Add**. 5 To delete a header, select it from the **Custom Headers** area and click **Delete**.

Advanced Options - File Not Found Detection

File-Not-found options are described in the following table.

Table 4 Advanced Options - 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.

Table 4 Advanced Options - File Not Found Detection (cont'd)

Option	Description
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 area 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. Follow the steps below to create a custom file-not-found signature: 1 In the **Signature** text box, type or paste a unique phrase that appears anywhere within the HTTP response message that you want to ignore. 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. 2 Click **Add**. 3 If necessary, repeat steps 1-2. You can specify as many 404 signatures as desired.

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.

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.

DevInspect 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: /\([\w\d]+\)/

Follow the steps below to add a state parameter:

1 Click **Add**.

2 Enter the name of the parameter in the **HTTP Parameter** box.

3 Select either **HTTP Query Data** or **HTTP Post Data**.

4 Click **OK**.

URL Path State

If your application determines state from certain components in the URL path, select **Determine State from URL Path** and add one or more regular expressions that identify those components. The defaults identify ASP.NET cookieless session IDs.

Follow the steps below to add a URL expression:

1 Click **Add**.

2 In the *URL State Detection Entry* dialog, enter a regular expression that identifies session state-keeping values that are part of the URL.

3 In the **Comments** box, enter a description of the state-keeping component.

4 Click **OK**.

Advanced Options - Overrides

These options accommodate the interface between DevInspect and two tools (the Web Form Editor and the Web Macro Recorder) when using the Vista operating system and Visual Studio 2008.

Table 5 Advanced Options - Overrides

Option	Definition
Use defaults (determined by project type and OS)	Select this option to incorporate logic that, when necessary, replaces “localhost” with the machine name
User defined	Select this option if and when a Microsoft update requires you to specify the equivalent of localhost (which translates to the loopback IP address 127.0.0.1 in IPv4).

Advanced Options - Web Forms

Web forms options are described in the following table.

Table 6 Advanced Options - Web Forms

Option	Definition
Submit forms	Select this check box if you want DevInspect to populate and submit forms encountered during an audit.

Table 6 Advanced Options - Web Forms (cont'd)

Option	Definition
Maximum submissions per form	Enter the maximum number of times that any single form will be submitted during DevInspect's scan. Note: Use the Web Form Editor to create or modify a list of input controls and associated values to be submitted during a scan. Refer to Recording Web Form Values on page 55 for more information.

AMP Options

This configuration information is used for integrating DevInspect into the Assessment Management Platform (AMP). If your DevInspect license does not allow connecting to AMP, these controls are not available.

When connected to AMP, you can upload scans to the AMP system. However, AMP controls which IP addresses may be scanned and also provides the SecureBase data and scanning policies to DevInspect.

AMP options are described in the following table.

Table 7 AMP Options

Option	Description
Connect to AMP	Select this check box to integrate with AMP.
AMP Manager URL	Enter the URL or IP address of the AMP Manager.
AMP Manager Client Authentication	Enter a user name (formatted as domain\username) and password, then click **Test** to verify the entry.
Enable Proxy	If DevInspect must go through a proxy server to reach the AMP manager, select **Enable Proxy** and then provide the IP address and port number of the server. If authentication is required, enter a valid user name and password.
Auto Upload Scans	If you select this option, DevInspect automatically uploads scan results to the AMP manager upon completion of the scan. If you do not select this option, you must click **Upload Scan to AMP**  in DevInspect Explorer to transfer the scan.
Scan Target	Select the AMP site associated with your project. If you are scanning a project, you can associate the URL "localhost" with a specific designated site in the AMP environment. When scanning a remote site (such as http://zero.webappsecurity.com), if you specify the scan target as "default," AMP will create a site, if necessary. Alternatively, you can pick a target that has been created specifically to accumulate scans of a particular site.

Authentication Options

Authentication options are described in the following table.

Table 8 Authentication Options

Option	Description
Authentication type	Select the type of server authentication required for the target Web site. • **None**: No authentication is required. • **Basic**: Use the industry-standard HTTP method for collecting user name and password information. 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. • **NTLM**: Use NTLM (NT LanMan) authentication. This process 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. • **Auto**: Allow DevInspect to determine the correct authentication type.
Credentials	If authentication is required, enter a user name and password.
Client Certificates	Client certificate authentication allows users to present client certificates rather than entering a user name and password. Follow the steps below to use client certificates. 1 Select **Enable** in the **Client Certificates** group. 2 Click **Select** to open the *Client Certificates* dialog box. 3 Choose a certificate. 4 Click **OK**.

Licensing Options

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 HP test site.

Activate

Follow the steps below to activate your installation:

1 In the **Activation Key** box, enter the activation key provided to your organization by HP.

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

3 Click **Update License**.

Request Trial License

Follow the steps below to request a trial license:

1. Complete all fields in the **Personal Information** section.
2. Click **Request Trial**.

Upon receiving your request, HP 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.

Deactivate

You can deactivate your DevInspect installation instance, allowing you to move the installation to another computer or re-image the computer without losing an allocated instance. If you uninstall DevInspect without first deactivating its installation instance, you will need to contact HP Support to have it reactivated.

1. Open DevInspect on the original computer.
2. Click **Tools** > **Options** > **HP DevInspect** > **Licensing**.
3. Copy the string in the **Activation Key** box. Write it down or save it to a file; you will need this to reactivate DevInspect.
4. Click **Deactivate**.
5. When a warning message informs you that you are about to remove the license information, click **Yes**.
6. Install DevInspect on the newer computer, or re-image the current computer.
7. Activate the installation instance by entering your activation key.

Permissions Options

Allowed Hosts

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 want 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

How to Add Allowed Hosts

Follow the steps below to add or delete allowed hosts:

1. Click ⊞ next to **Allowed Hosts** to expand the category.
2. Select **Allowed Hosts** and click ....

   The *String Collection Editor* window opens.
3. To add, type one or more allowed host entries (one per line).
4. To delete, double-click an entry and press the DELETE key (or right-click the entry and select **Delete** from the shortcut menu). Remove any blank lines that may remain.

Exclusions

Use this feature to prevent designated objects from being audited during a DevInspect vulnerability assessment.

How to Add Excluded Objects

Follow the steps below to add or delete excluded objects:

1. Click ⊞ next to **Exclusions** to expand the category.
2. Select **Cookies**, **Headers**, **Hosts**, **Paths**, **Postdata Parameters**, or **Query Parameters** and click ....

   The *String Collection Editor* window opens.
3. To add, type one or more entries (one per line). Format all entries as regular expressions
4. To delete, double-click an entry and press the DELETE key (or right-click the entry and select **Delete** from the shortcut menu). Remove any blank lines that may remain.

In the **Paths** category, you can specify a URL or file extensions (to exclude files of a certain type).

Policy Options

Policy options are described in the following table.

Table 9 Policy Options

Option	Description
Use standard policy	Select this option to use the standard scanning policy, without modification, provided by HP.
Use AMP policy	If you select this option, you can choose a policy that will be downloaded from AMP.
Use custom policy	Select this option to modify the scanning policy or to use a scanning policy that you previously modified and saved.

How to Modify Policies

Follow the steps below to modify and save a policy:

1. Select **Use custom policy**.
2. Click **Browse**.

   A standard file-selection window opens.
3. Type a name for the new policy in the **File name** box and click **Open**.
4. When prompted to confirm the creation of a new policy, click **Yes**.
5. Include (or exclude) an individual attack agent by selecting (or clearing) its associated check box.

Follow the steps below to load a modified policy:

1. Select **Use custom policy**.
2. Click **Browse**.

   A standard file-selection window opens.
3. Select the policy you want to use and click **Open**.

You can modify the policy, if required, by selecting (or clearing) the check boxes associated with each attack agent.

Proxy Options

Proxy options are described in the following table.

Table 10 Proxy Options

Option	Description
Make requests through a proxy	Select this option to access the Internet through a proxy server, and then enter the requested information.

Table 10 Proxy Options (cont'd)

Option	Description
Use proxy settings from Microsoft Internet Explorer	Select this option to import your proxy server information from Internet Explorer. If you select this option, the remaining options are not used.
Authentication	If your proxy server requires authentication, click ⊞ to expand the **Authentication** category (if necessary) and then enter a password and a user name.
Exceptions	If you do not need to use a proxy server to access certain IP addresses (such as internal testing sites), specify those sites here. 1 If necessary, click ⊞ to expand the **Exceptions** category. 2 Select **Bypass Addresses** and click ... 3 The *String Collection Editor* window opens. 4 To add, type one or more IP addresses that can be accessed without a proxy server (one per line). To delete, double-click an entry and press the DELETE key (or right-click the entry and select **Delete** from the shortcut menu). Remove any blank lines that may remain.
General	Click ⊞ to expand the **General** category (if necessary), and then enter the IP address and port number of your proxy server, and select a proxy server type. **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.

Requests Options

A requestor is the software module that handles HTTP requests and responses. Request options are described in the following table.

Table 11 Requests Options

Option	Description
Shared	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 configuration is suitable for use when maintaining state is not a significant consideration. You also specify the maximum number of concurrent requests that may be issued before DevInspect must wait for an HTTP response to the first request.

Table 11 Requests Options (cont'd)

Option	Description
Separate	If you select this option, the crawler and auditor use separate requestors. Also, the auditor's requestor associates a state with each thread, rather than having all threads use the same state. This method results in significantly faster scans. You also specify the maximum number of concurrent requests that can be created by each requestor before DevInspect must wait for an HTTP response to the first request. Increasing the maximum number of 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.
Request timeout interval	Specify how long DevInspect will wait for an HTTP response from the server. If this threshold is exceeded, DevInspect resubmits the request until reaching the retry count. If it then receives no response, DevInspect logs the timeout and issues the first HTTP request in the next attack series.
Retry count	Specify how many times DevInspect will resubmit a request to the server when receiving a "failed" response (which is defined as any socket error or request timeout).

If you notice numerous entries on the *Output* window showing requests timing out, you should reduce the maximum number of concurrent requests. 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.

Team Foundation Server

If you select **Automatically create vulnerability work items**, DevInspect will create work items of the type "Vulnerability" as vulnerabilities are detected during a scan. You must also be working with a Team Project (that is, the project must be checked into Team Foundation Server's source control). For instructions on integrating DevInspect with a Team Foundation Server, see Integrate with a Team Foundation Server on page 17.

4 Using DevInspect

DevInspect Overview

DevInspect accelerates the construction and delivery of secure Web applications and Web services by finding and fixing security vulnerabilities during development and protecting applications after deployment. DevInspect applies the most innovative vulnerability analysis and remediation techniques to pinpoint and correct application vulnerabilities before they can be released into production.

Using both static analysis and dynamic analysis audit engines, DevInspect examines your entire Web site (or specified discrete portions of it). It then creates a list of each vulnerability detected and provides detailed information about the dangers.

You retain total control of your source code by choosing whether or not to apply DevInspect's secure coding library or to use the reference information and examples to correct the code yourself. DevInspect also provides special validator objects that you can apply manually while coding your application, as well as a special module that will protect your application against "brute force" attacks.

Analyze a Web Site, Project, or Page

You can instruct DevInspect to analyze an entire Web site or Web application project (finding all vulnerabilities in the application) or a page (isolating an individual page for analysis).

1. Open a Web site or Web application project.
2. Can you see the DevInspect Explorer window or its tab?

   Yes: Go to Step 3.

   No: Click the **View** menu and select **DevInspect Explorer**.
3. Select the DevInspect Explorer window to display a hierarchical list of the site or application components.
4. Choose an area to analyze:
   - To target the entire site or application project, right-click the root.
   - To target a specific page, right-click the page.
5. Select **Analyze** from the shortcut menu.

   DevInspect builds and scans the Web site or application project. When analysis is complete, it lists all discovered vulnerabilities in the Error List window.

   Note: Whenever "[Correlated]" is appended to the vulnerability description, it means that DevInspect has detected the vulnerability through both static and dynamic analysis.

6 For AMP integrations only: If the option to “auto upload scans” is selected, you must select one of the following options:

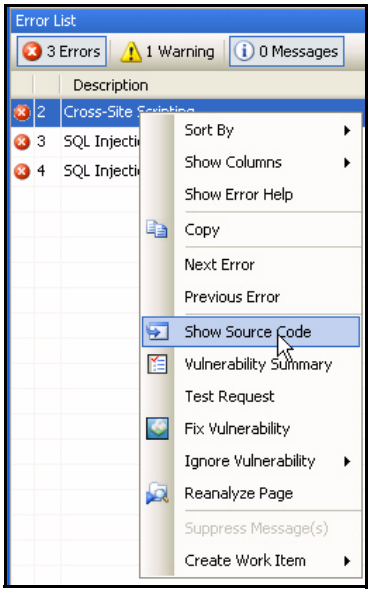
- Use default scan target <URL>

  DevInspect will upload the scan to an AMP site named for the URL you scanned. If a site by that name does not exist, AMP will create one.

- Specify scan target

  If you are scanning a project, you can associate the URL “localhost” with an AMP target site that has been created specifically to accumulate scans for that project.

7 Right-click an item in the Error List.



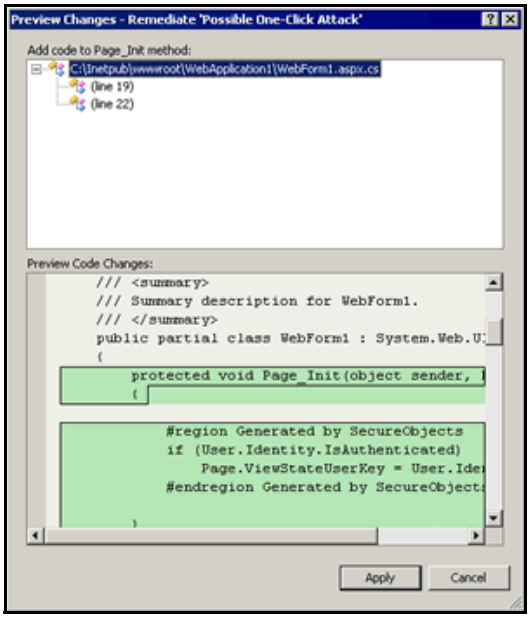
8 Select one of the following from the shortcut menu:

- **Show Source Code** — DevInspect displays in the document window the code associated with the vulnerability (and with the correction, if you previously selected **Fix Vulnerability**, described below).

- **Vulnerability Summary**—DevInspect displays extensive information about the vulnerability and identifies the affected files, specifying (where applicable) both the vulnerability source (the point at which possibly tainted data enters the system) and its sink (the point at which the data is used). From here, you may also instruct DevInspect to fix the vulnerability.

- **Test Request**—DevInspect launches the HTTP Editor and loads the selected HTTP request. You can then resend the request (or send an edited request) and view the HTTP response.

- **Fix Vulnerability**—DevInspect displays proposed modifications.



  To reject the proposed modifications, click **Cancel**.

  To accept the changes, click **Apply**. DevInspect generates code or adds a validator to the page. For validators, select the validator in the Document window and then modify its properties (if necessary) using the Properties window.

- **Ignore Vulnerability**—DevInspect removes the vulnerability from the Error List after you stipulate how often the problem should be ignored: This Time Only, This Page Only, or This Web Site Only.
- **Reanalyze Page**—DevInspect rescans the page on which the vulnerability was detected.
- **Create Work Item**—If you select **Vulnerability Work Item** from the cascading menu, DevInspect copies the vulnerability information to a vulnerability work item in the TFS work item store.

Secure Inputs for a Site, Project, or Page

DevInspect offers an alternative method for analyzing application inputs and securing them from malicious attacks.

Follow the steps below to secure an application or page.

1 Open a Web site or Web application project.

2 Can you see the DevInspect Explorer window or its tab?

  Yes: Go to Step 3.

  No: Click the **View** menu and select **DevInspect Explorer**.

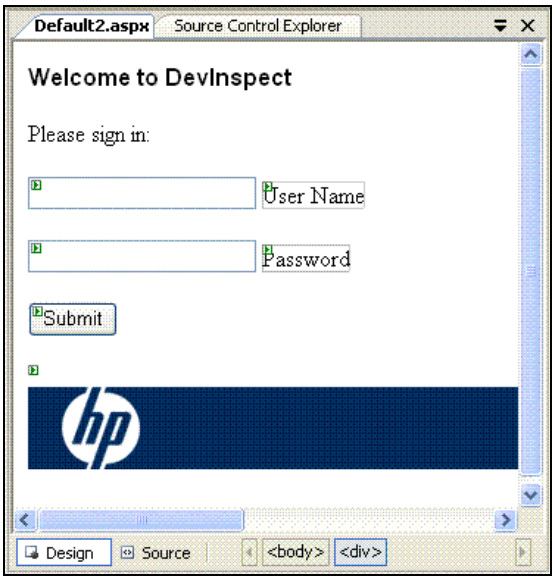
3 Select the DevInspect Explorer window to display a hierarchical list of the site or project components.

4 Choose an area to analyze:

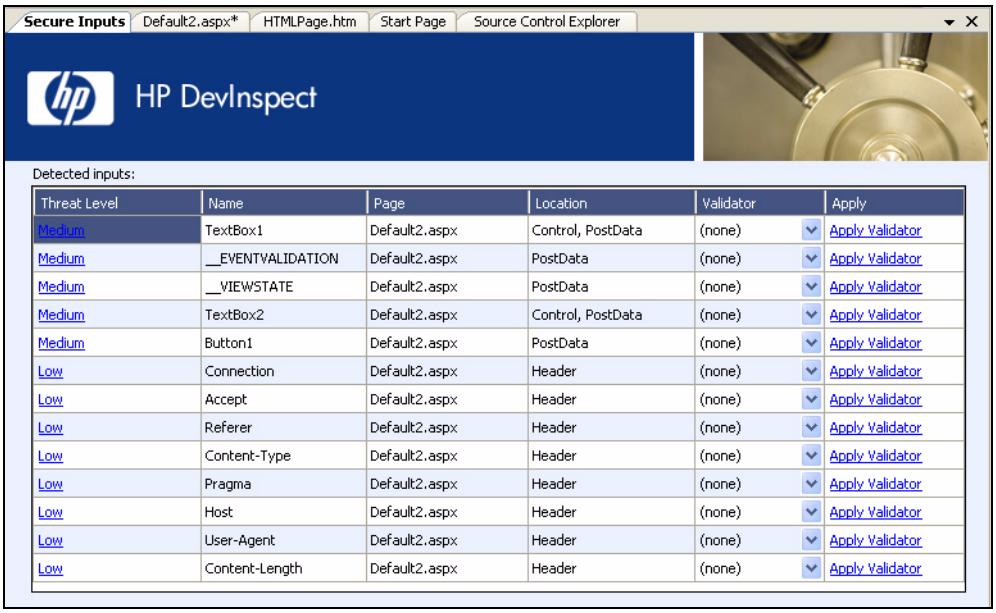
- To target the entire site or project, right-click the root and select **Secure Application Inputs** from the shortcut menu.
- To target a specific page, right-click the page and select **Secure Page Inputs**.

After analyzing the application or the page, DevInspect lists all detected inputs.

Example: For the following page …



… the Secure Page Inputs function detected the following inputs:



Threat Level	Name	Page	Location	Validator	Apply
Medium	TextBox1	Default2.aspx	Control, PostData	(none)	Apply Validator
Medium	__EVENTVALIDATION	Default2.aspx	PostData	(none)	Apply Validator
Medium	__VIEWSTATE	Default2.aspx	PostData	(none)	Apply Validator
Medium	TextBox2	Default2.aspx	Control, PostData	(none)	Apply Validator
Medium	Button1	Default2.aspx	PostData	(none)	Apply Validator
Low	Connection	Default2.aspx	Header	(none)	Apply Validator
Low	Accept	Default2.aspx	Header	(none)	Apply Validator
Low	Referer	Default2.aspx	Header	(none)	Apply Validator
Low	Content-Type	Default2.aspx	Header	(none)	Apply Validator
Low	Pragma	Default2.aspx	Header	(none)	Apply Validator
Low	Host	Default2.aspx	Header	(none)	Apply Validator
Low	User-Agent	Default2.aspx	Header	(none)	Apply Validator
Low	Content-Length	Default2.aspx	Header	(none)	Apply Validator

5 To assign a validator to an input:

a Click the **Validator** drop-down in the row associated with an input.

b Select a validator.

c Click **Apply Validator**.

Apply Validation Manually

With a project Web page opened in a Visual Studio code/document window, use the following procedure to add a SPIValidator object manually.

1 Is the Visual Studio Toolbox visible?

Yes: Go to Step 2.

No: Select **Toolbox** from the **View** menu.

2 From the DevInspect tab of the toolbox, drag a SPIValidator object onto the form and drop it near a control.

3 Select the SPIValidator object and, in the Properties window, set the properties.

4 To log messages to the Windows Event Log, double-click the SPIValidator control.

Edit a Validator

Follow the steps below to edit a SPIValidator object.

1 In the document window, click the tab associated with the page on which the SPIValidator resides.

2 Click the SPIValidator.

3 Use the Properties window to modify the following properties.

All properties except ControlToValidate and AllowPattern are optional.

ControlToValidate

Select a control, or enter the name of an HTTP header or cookie.

ErrorMessage

Modify the text of the message that will display if user input fails the validation test.

DataTypeToValidate

Select either QueryString, PostString, HTTPHeader, Cookie, or AjaxEventHandler. Leave blank for normal validation of a visible control on the form. Change not recommended.

AllowPattern

You can select a regular expression pattern from the SecureObjects library or you can create a custom pattern.

Use the following procedure to apply a SecureObjects pattern:

1 Click the ellipsis button  to display the Choose an Allow Pattern dialog.

2 Select a validation expression from the **Pattern** list.

3 If necessary, edit the error message.

You can edit the error message both here and in the ErrorMessage property.

4 To test for validity, click **More**. The **User Input** box contains sample input for the pattern selected. Enter an input string. If the pattern allows the input, a check mark appears on a green disk; if the pattern denies the input, an X appears on a red disk.

5 When finished, click **OK**.

Use the following procedure to create and apply a custom pattern:

1 Click the ellipsis button [...] to display the *Choose an Allow Pattern* dialog.

2 Using the **Pattern** list, select one of the SecureObjects patterns to use as a template or select **Custom Pattern**.

3 Click **Modify**.

4 On the *Modify an Existing Pattern* dialog, select one of the following and then click **OK**:

- **Use the modified pattern on this validator only**; your modifications will be applied to the validator, but will not be saved in the SecureObjects library.
- **Reuse the modified pattern with a new name** and create a name for the pattern in the **New name** box; the validator will be added to the SecureObjects library.

5 In the **MinLength** box, type or select the minimum number of characters in the input

6 In the **MaxLength** box, type or select the maximum number of characters in the input.

7 In the **Error msg** box, type the message to display when the input fails validation.

8 If necessary, click **More>>** to extend the dialog box.

9 In the **Expression** box, modify the regular expression that will be used to validate the input.

10 (optional) In the **Replace** box, specify one or more substrings of the parsed input.

11 To test for validity, enter an input string in the **User Input** box. If the pattern allows the input, a check mark appears on a green disk; if the pattern denies the input, an X appears on a red disk.

12 When finished, click **OK**.

DenyPattern

This feature is optional and is used primarily to create a log event when input is denied because it matches a selected pattern.

1 Click the ellipsis button [...] to display the Choose Deny Pattern(s) dialog.

2 Select one or more of the following patterns:

- **Buffer Overflow**: prevents hacker efforts to transfer excess data to a buffer causing overflows into another buffer.
- **Cross-Site Scripting**: prevents cross-site scripting attacks.
- **Directory Traversal**: prevents attempts to map your entire web-site hierarchy and directory structure.
- **SQL Injection**: prevents SQL injection attacks.
- **URL**: prevents certain attacks that use technically legal URL syntax.

3 Click **OK**.

ReplacePattern

If the AllowPattern contains a custom regular expression, this property contains the replacement pattern to use when calculating the contents of Value. If using a predefined AllowPattern, this field is ignored.

This property allows you to parse user input into substrings that are programmatically accessible. For example, the "U.S. Zip Code" regular expression for the allow pattern is:

```
(\d{5})(?:\s?-?\s?(\d{4}))?
```

This expression contains two substrings, as defined by the enclosing pairs of parentheses:

- (\d{5})
- (\d{4})

The notation $1 refers to the first substring (the five-digit ZIP code) and $2 refers to the second substring (the four-digit ZIP+4 code). A programmatic reference to the four-digit substring would be:

```
TextBox1_validator.Values[2]
```

The notation $0 (which is the default for most replace patterns) concatenates all the grouped substrings in the expression.

MaxLength

Enter the maximum number of characters that the input string may contain.

If you set the MaxLength property for both the input control and this validator object, the lower value will be enforced.

Analyze a Portion of the Web Site or Project

The Browse-and-Analyze method (also known as Step Mode) opens a browser and allows you to navigate to whatever sections of your application you choose to visit. DevInspect records information only about those resources that you encounter while manually navigating the site. This feature is used most often to enter a site through a logon page or to define a subset or portion of the application that you want to investigate.

Follow the steps below to browse and analyze a Web site:

1. Open a Web site or Web application project.
2. Can you see the DevInspect Explorer window or its tab?

   Yes: Go to Step 3.

   No: Click the **View** menu and select **DevInspect Explorer**.
3. Select the DevInspect Explorer window to display a hierarchical list of the site or project components.
4. Right-click the Web site root and select **Browse and Analyze** from the shortcut menu.
5. Using a browser opened by DevInspect, navigate to those sections of the Web site that you want to investigate.
6. When finished, close the browser.

Once you finish navigating through the site, you can audit the results to assess the security vulnerabilities related to that portion of the site that you recorded.

Modify Scanning Properties

You can tailor DevInspect to meet the specific and individual requirements of your development environment by modifying the properties associated with a Web site/project or a page.

Follow the steps below to modify DevInspect scanning properties.

1 Open a Web site or Web application project.

2 Can you see the DevInspect Explorer window or its tab?

Yes: Go to Step 3.

No: Click the **View** menu and select **DevInspect Explorer**.

3 Select the DevInspect Explorer window to display a hierarchical list of the site or project components.

4 Do you want to modify scanning properties for the entire Web site/project?

Yes: Click the root.

No: Click a page (to modify properties for a specific page).

A list of properties appears in the Properties window.

5 (For Web site/project only) Click ⊞ next to **Authentication** to expand the list.

- **Authentication Script** — You can use the Web Macro Recorder to create or edit a script that includes a user name and password (or other credentials) that authenticate the user to the system. DevInspect will use that script to enter your Web site and (optionally) navigate through your application. Select this property and click the ellipsis button to open the Web Macro Recorder. You can then create a script or simply load (and optionally edit) a previously recorded script.
- **Use Authentication Script** — To instruct DevInspect to use the selected authentication script while scanning your site, select True.

6 Click ⊞ next to **Checks** to reveal the **Ignored Checks** item. DevInspect deploys a number of attack groups when scanning your site or project. Each attack group contains individual modules (called attack agents or checks) that "check" your Web site for vulnerabilities. You can elect to turn off individual checks, instructing DevInspect to ignore (not conduct) these attacks.

To exclude or include checks:

a In the Ignored Checks property value column, click [...].

The Ignored Checks window opens.

b Click ⊞ next to a category to expand the list of checks.

c To exclude a check (or a category), select the check (or category) in the Active Checks list and click [>>]. The selection is moved to the Ignored Checks list.

d To include an excluded check (or a category), select the check (or category) in the Ignored Checks list and click [<<]. The selection is moved to the Active Checks list.

7 (For Web site only) Click ⊞ next to **Forms** to reveal the **Web Form Values** item. Use this feature to create, edit, or load a list of Web form input controls (and their associated values) that DevInspect will use to complete and submit forms during an audit of your page or site. See Web Form Editor on page 53 for more information.

8 (For site or project only) Click ⊞ next to **Permissions** to expand the list. Use this section to exclude specific cookies, form values, or query parameters from being audited by DevInspect.

a In the property value column of Excluded Cookies, Excluded Form Values, or Excluded Query Parameters, click ...

The String Collection Editor window opens.

b To add, type one or more entries (one per line). Format all entries as regular expressions.

c To delete, double-click an entry and press the DELETE key (or right-click the entry and select **Delete** from the shortcut menu). Remove any blank lines that may remain.

9 Click ⊞ next to **Status** to expand the list. The read-only Status properties indicate:

- The date and time the project was last analyzed.
- The current security status of the project item.
- The URL of the Web site.

Protect Against 'Brute-Force' Attacks

The DevInspect BruteProtector can inoculate a page against brute-force attacks. The BruteProtector is intended for use with production versions of your application. Because it will also protect against the simulated attacks that DevInspect conducts when analyzing your application, it should not be applied until you are ready to release.

Follow the steps below to add a BruteProtector to a page.

1 Can you see the Toolbox?

Yes: Go to Step 2.

No: Select **Toolbox** from the **View** menu.

2 From the DevInspect tab of the toolbox, drag a BruteProtector object onto the page and drop it.

3 Select the BruteProtector object.

4 In the Properties window, set the properties as described in the following table.

Table 12 BruteProtector Properties

Property	Definition
ID	Programmatic name of the control.
AutoDelay	If you select **Yes**, then DevInspect adds an incremental artificial time delay before responding to an HTTP request when a brute-force attack has been detected.

Table 12 BruteProtector Properties

Property	Definition
Delay	The number of seconds that should be added incrementally to the period that DevInspect waits before sending an HTTP response when a brute-force attack has been detected.
EnableViewState	If you select **True**, the control saves its state for use in round-trips.
ProtectedInputs	Click … and select which input controls should be protected (usually a password text box and a user name text box).
RequestCountThreshold	The maximum number of times you will allow a client to submit non-valid data during the time span specified by the RequestTimespanThreshold. The default is 5 attempts.
RequestTimespanThreshold	The period during which the number of non-valid submissions will be compared to the RequestCountThreshold. The default is 5 minutes.

5 In the Properties window, click the Events icon  .

6 In the AttackDetected property, select AttackDetected.

7 To enable brute force protection, add an app setting element in your web.config file, with a key of “disableBruteForceProtection” and a value of “false.” App settings are found in the <appSettings> element, under the <configuration> element in the web.config. For example:

```
<configuration>
   <appSettings>
      <add key="disableBruteForceProtection"
      value="false"/>
   </appSettings>
...
```

Integrate ASP.NET Health Monitoring

DevInspect supports the ASP.NET health monitoring feature by detecting malicious attacks on your deployed application and sending this event data to one or more health monitoring providers that you specify.

To implement health monitoring for DevInspect events, add the following snippet to your configuration settings. In this example, if a DevInspect validator detects a malicious input attack or if the BruteProtector detects a “brute force” attack, Web event data would be sent to the EventLogProvider.

```
<!-- This snippet should be placed in the configuration\system.web
```

```
section of a web.config file. -->
<configuration>
   <system.web>
      <healthMonitoring enabled="true">
         <eventMappings>
            <add name="SecureObjects Security Events"
type="SPI.SecureObjects.SecurityAuditEvent,SPIWebControls"/>
         </eventMappings>
         <rules>
            <add name="Security Event Subscription"
eventName="SecureObjects Security Events" provider="EventLogProvider"/>
         /rules>
      </healthMonitoring>
```

Create Reports

After analyzing your Web site or page, you can obtain a comprehensive report containing the results of DevInspect's analysis.

Follow the steps below to create a report.

1. Open a Web site or Web application project.
2. Can you see the DevInspect Explorer window or its tab?

   Yes: Go to Step 3.

   No: Click the **View** menu and select **DevInspect Explorer**.
3. Select the DevInspect Explorer window to display a hierarchical list of the site or project components.
4. In the DevInspect Explorer, click the Create Report icon  .
5. Enter a name and location for the report, or accept the defaults supplied by DevInspect.
6. Click **Save**.

Export Scan Data

Use the Export function to save information collected during a DevInspect analysis. The data is formatted as XML. If DevInspect detects characters that would prevent an XML file from opening properly, it replaces those character strings with the message "Error exporting data." This typically occurs when the target site contains Flash (.swf) files.

1. Open a Web site or Web application project.
2. Can you see the DevInspect Explorer window or its tab?

   Yes: Go to Step 3.

   No: Click the **View** menu and select **DevInspect Explorer**.

3 Select the DevInspect Explorer window to display a hierarchical list of the site or project components.

4 In the DevInspect Explorer, click the Export Results icon  .

5 Enter a name and location for the report, or accept the defaults supplied by DevInspect.

6 Click **Save**.

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
- SOAP Editor
- Regular Expression Editor
- Encoder/Decoder

This appendix describes each tool and provides a detailed explanation of how these tools can be used within your environment to enhance the security of the products you develop.

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 options. For example, if you specify a URL in the Excluded URL option, 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. In the DevInspect Explorer, select the root Web site.
2. In the Properties window, select the Authentication Script behavior.
3. Click the Browse button 

   The Web Macro Recorder appears.
4. Select **New** from the **File** menu (or click the New icon on the toolbar).
5. Click **Launch Browser**.
6. Using the browser's Address bar, enter or select a URL.
7. 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.
8. 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.
9. When finished, click the **Pause** button (on the recorder) or close the browser.
10. (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.

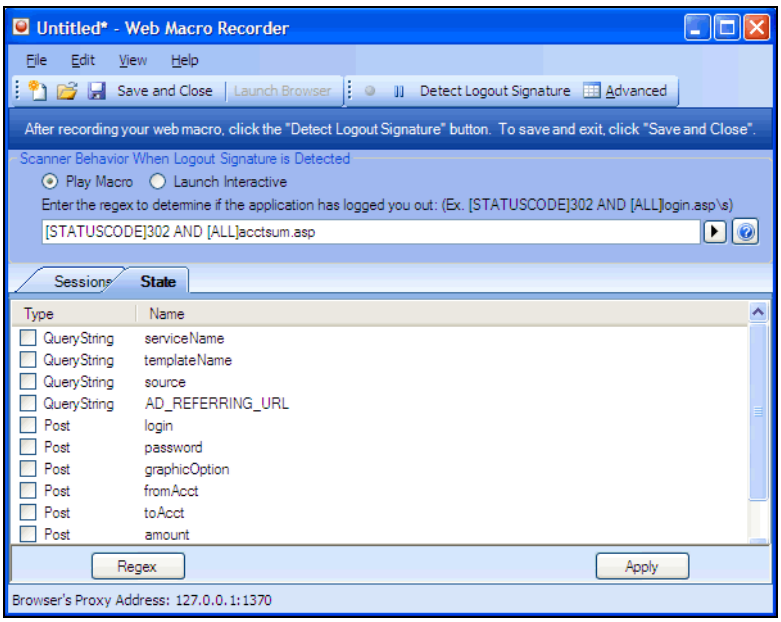
c If the Macro Recorder is unable to determine the logoff signature, try clicking on other URLs.

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 52 for more information.

11 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)

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



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**.

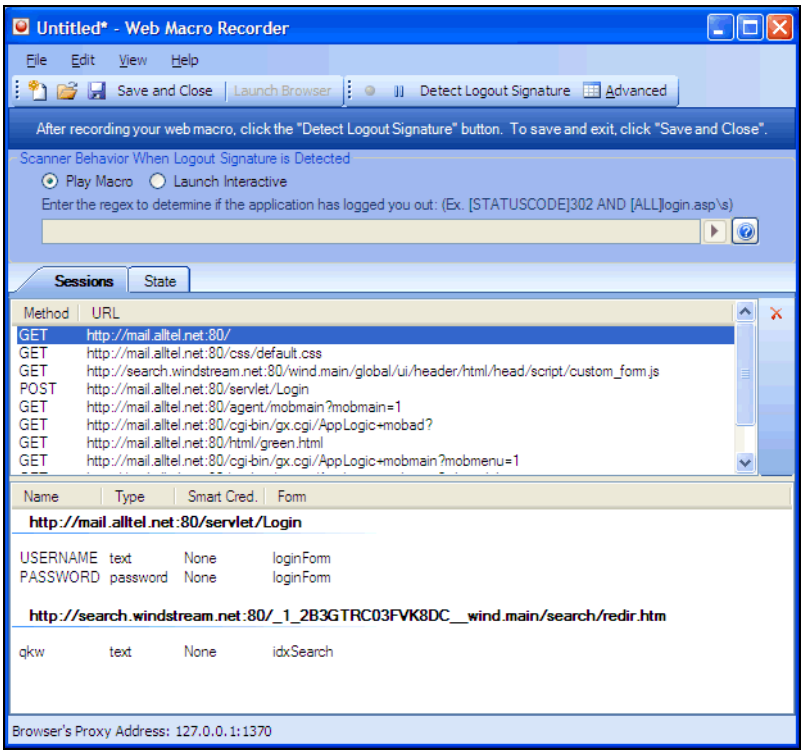
13. To save the macro, select **Save** or **Save As** from the **File** menu.

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.



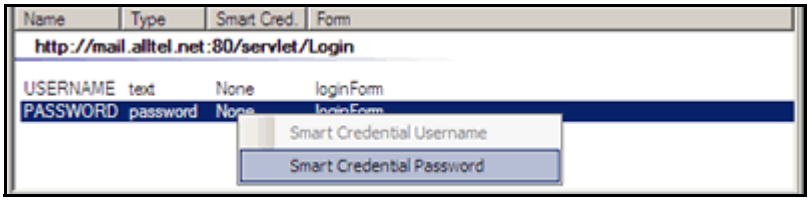
In this example, the form and the controls were rendered by the following HTML statements:

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

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

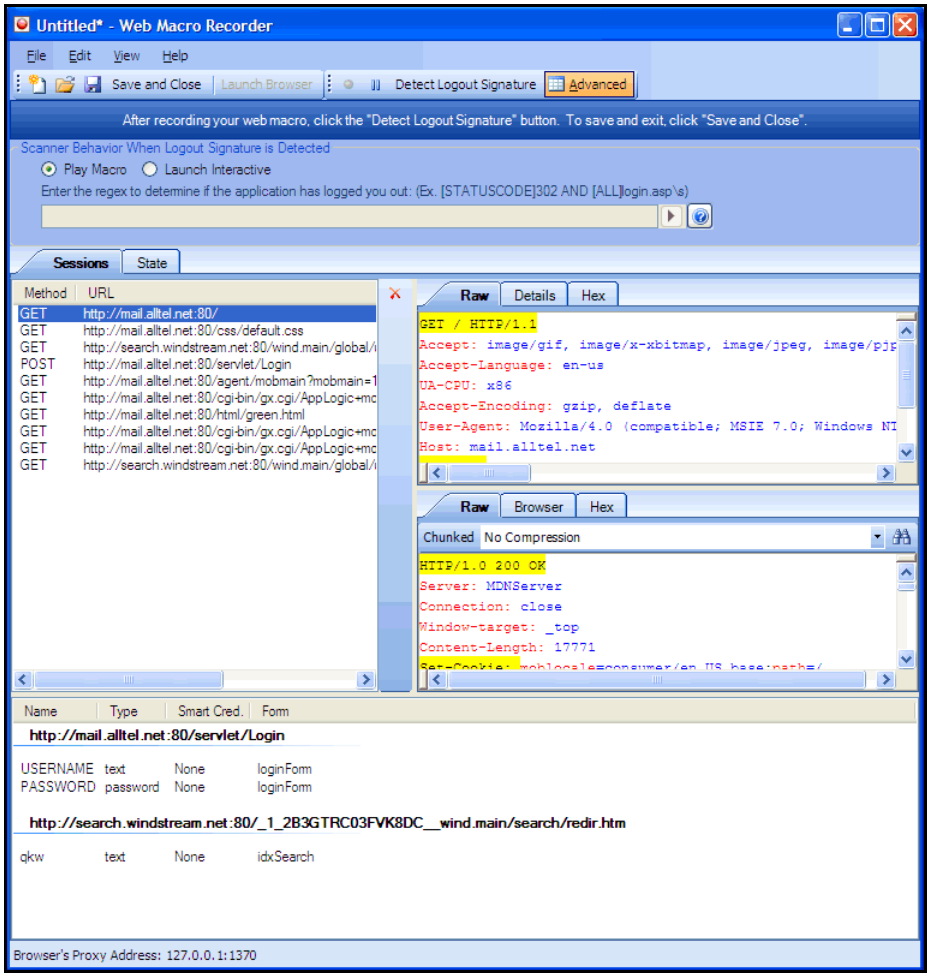
<input type=”password” size=”16” name=”PASSWORD”>

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.



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.



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.

Web Macro Recorder 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]
[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 the following 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:

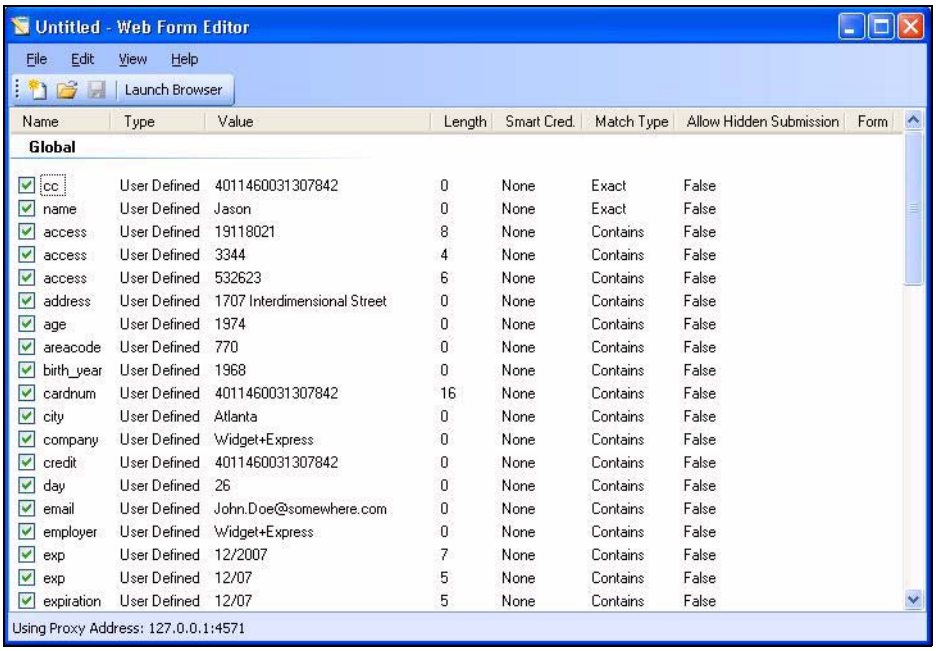
- Create the list manually
- Record the values as you navigate through the application

Manually Creating a Web Form List

Use the following procedure to create a Web Form list manually.

1. In DevInspect Explorer, select the root Web site.
2. In the Properties window, select the Web Form Values behavior.
3. Click the Browse button 

The *WebForm Editor* window appears.



4 To load a previously created Web form values file, select **Open** from the WebForm Editor's **File** menu.

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

6 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 (shortcut) menu.
- To modify a Web form value, right-click an entry and select **Modify** from the shortcut (shortcut) menu.

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

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

8 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).

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

10 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.

11 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**.

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

13 If necessary, you can assign additional attributes by right-clicking an entry and using the shortcut (shortcut) 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 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…

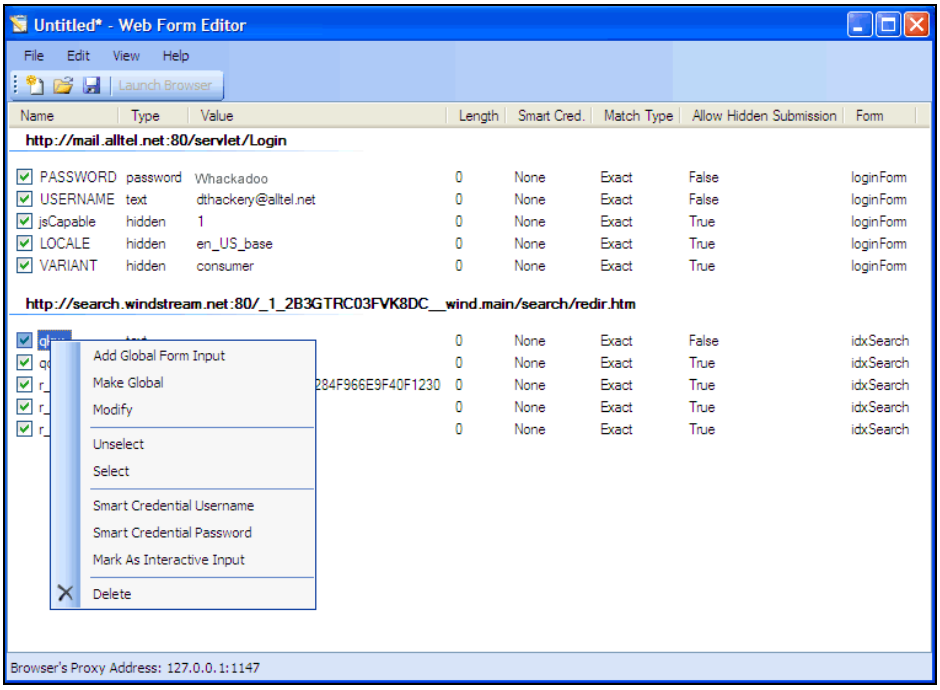
<form name="loginForm" action="/servlet/Login" method="POST">

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

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

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

…and the user entered his name and password.



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

If you elect to submit forms during a scan (see Advanced Options - Web Forms on page 25), DevInspect will use the Web form values file you specify in the Web Form Values behavior of the Properties window for the target Web site.

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 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 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 13 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 13 Rules for Matching Web Form Values (cont'd)

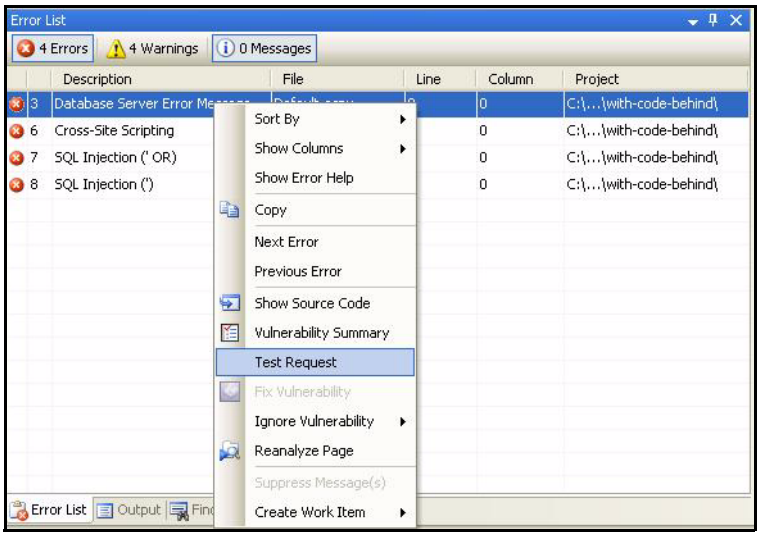
	Partial Match 5. Name value included in field name. Length allows wildcard.	A Web form value in the file partially matches the field name found. All characters in the Web form value match a portion of the Web page field name and the field length for the record allows for submission to any field length (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.

Use one of the following procedures to launch the HTTP Editor:

- Click the Windows **Start** button and select **All Programs > HP > HP Security Toolkit > HTTP Editor**.
- When using Web Proxy, select a request from the list of displayed sessions and click the HTTP Editor icon (or right-click the request and select **HTTP Editor** from the context menu).
- Within Visual Studio, right-click an HTTP request in the Error List and select **Test Request** from the context menu, as depicted in the following illustration.



Panes

Request Viewer Pane

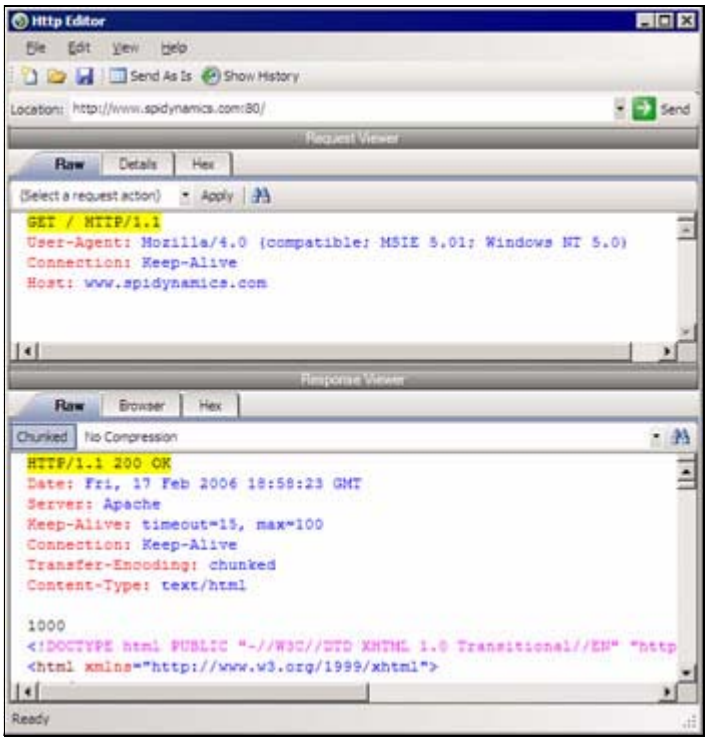
The Request Viewer pane contains the HTTP request message, which you can view in three different formats using the following tabs:

- **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.
- **XML** - Displays any XML content in the message body (Note: This tab appears only if the request contains XML-formatted data).

Response Viewer Pane

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.
- **XML** - Displays any XML content in the message body (Note: This tab appears only if the response contains XML-formatted data).



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

In normal mode, if you edit the message body of the request, the HTTP Editor recalculates the content length and substitutes the appropriate value in the Content-length header. However, when using the Send As Is option, the HTTP Editor does not modify the content length. You can force this recalculation before sending the request by selecting **Change Content-Length** and clicking **Apply**.

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**.

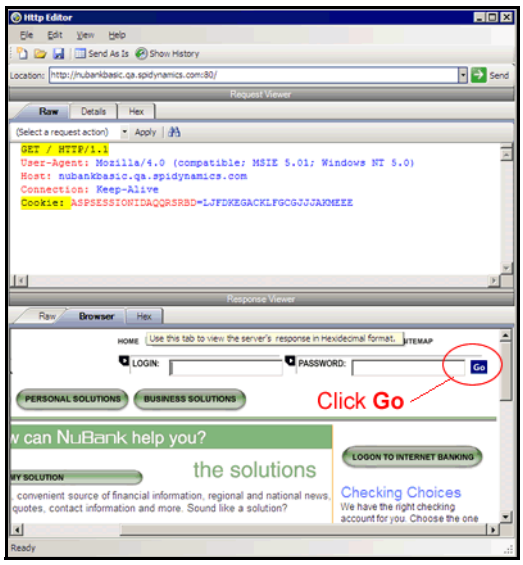
HTTP Editor Settings

Follow the steps below to configure HTTP Editor settings:

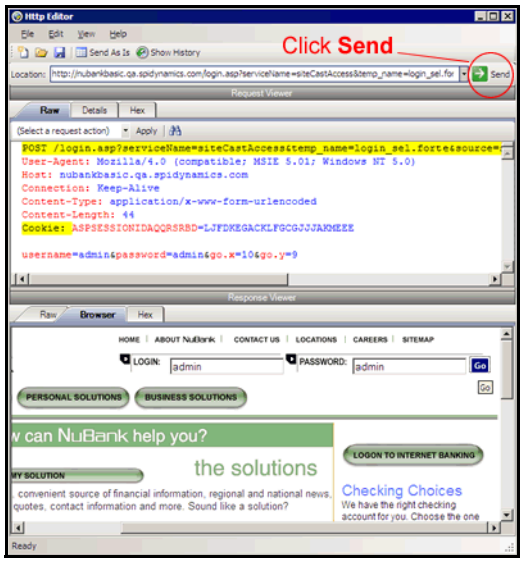
1. Click the **Edit** menu and select **Settings**.
2. Select the **Options** tab.
3. In the **Request** group:
   a. If you select **Send As Is**, the HTTP Editor will not modify the request, regardless of any other settings you may select. This allows you to send a purposely malformed message. Authentication 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**.



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**.



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 Visual Studio **Tools** menu, click **Options**.
2. On the Options window, expand the **HP DevInspect** node (if necessary) and select **Proxy**.
3. Click **Launch Web Proxy**.

   The *Web Proxy* window opens.

4. Click  or select **Start** from the **Proxy** menu.

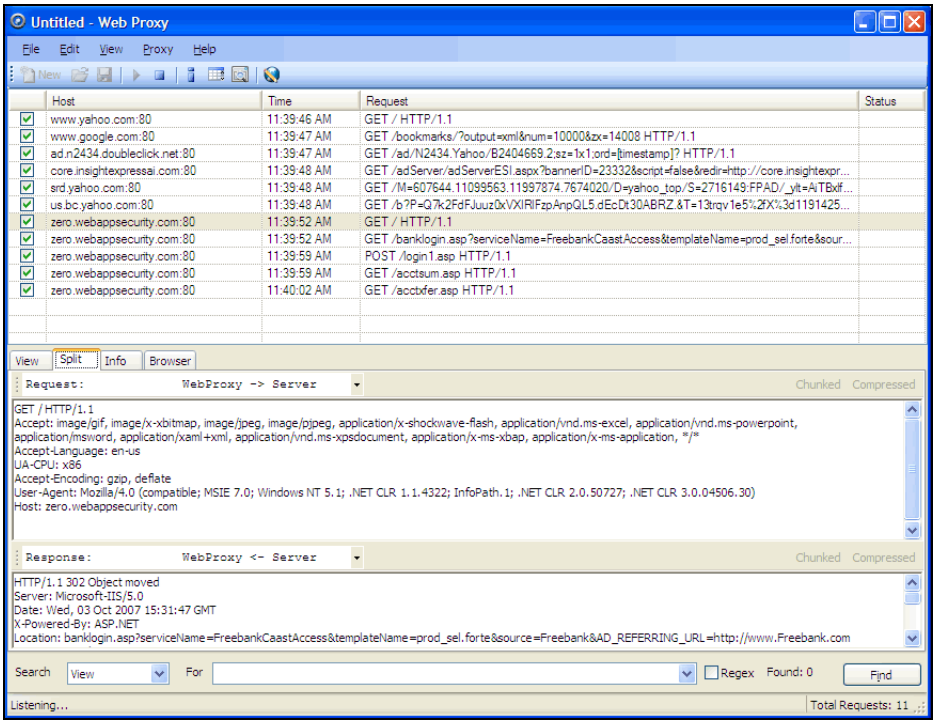
   "Listening" displays in the Web Proxy status bar.

5. Open your Web browser and manually navigate to 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.

6. When you have browsed to all necessary pages, return to Web Proxy and click  or select **Stop** from the **Proxy** menu.

7 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.



8 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.

9 To resend a request (with or without editing), select it from the list of displayed sessions and click the HTTP Editor icon (or right-click the request and select **HTTP Editor** from the context menu).

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 77 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 14 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.

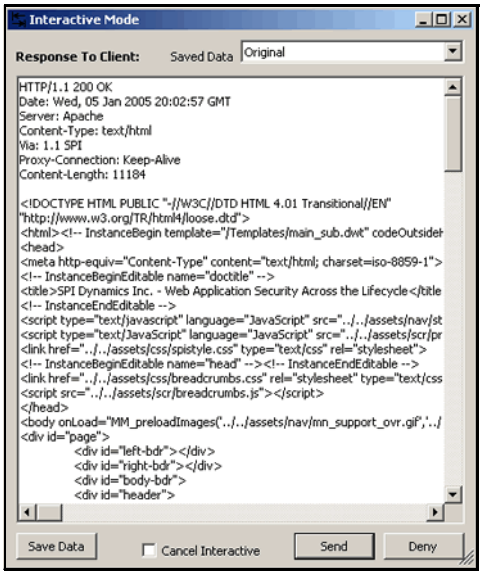
7 Click the drop-down arrow and select a color with which to highlight the data, if found.

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.



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  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.

SOAP Editor

Web services are programs that communicate with other applications (rather than with users) and answer requests for information. Most Web services use Simple Object Access Protocol (SOAP) to send XML data between the Web service and the client Web application that initiated the information request. Unlike HTML, which only describes how Web pages are displayed, XML provides a framework to describe and contain structured data. The client Web application can readily understand the returned data and display that information to the end user.

A client Web application that accesses a Web service receives a Web Services Definition Language (WSDL) document so that it understands how to communicate with the service. The WSDL document describes what programmed procedures the Web service includes, what parameters those procedures expect, and the type of return information the client Web application will receive.

SOAP uses HTTP and XML as the means to exchange information so that programs on one platform can communicate with a program on the same or a different operating system. Use the SOAP Editor to generate SOAP requests automatically, and to manually edit SOAP requests and responses. You can also create and save a file containing values that should be submitted by an HP scanner when conducting a Web services assessment.

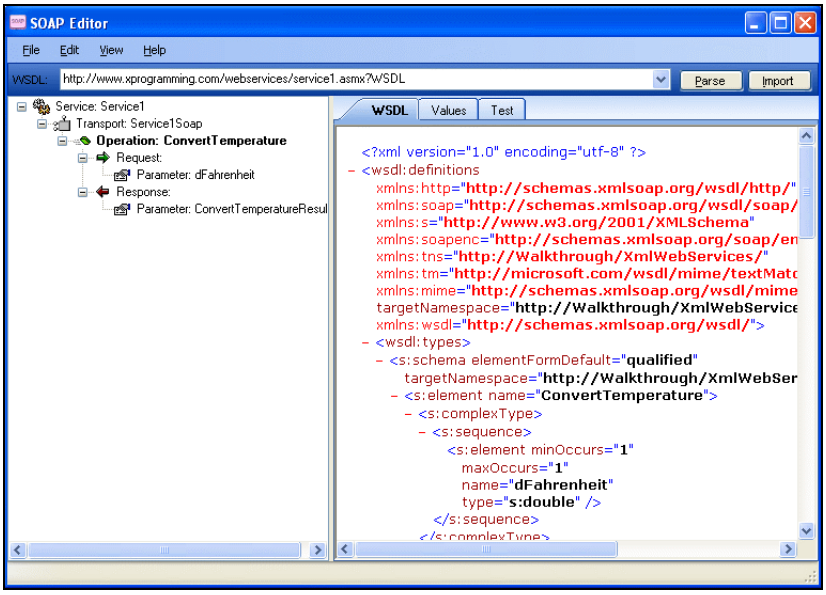
Submitting SOAP Requests

To submit SOAP requests:

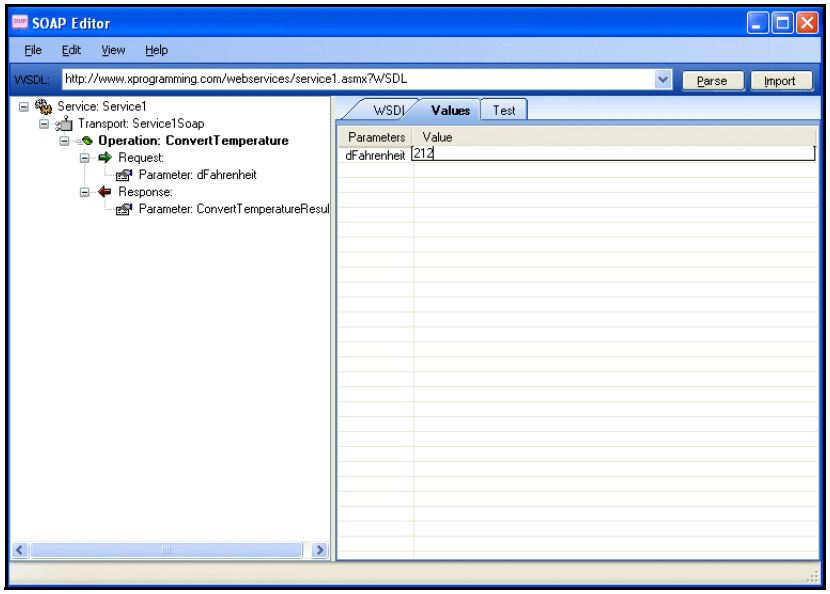
1. From the **Tools** menu, click **Soap Editor**.
2. Do one of the following:
   - In the **WSDL** box, type or select the URL of the WSDL site (example: http// :www.myprogramming.com/webservices/service1.asmx?WSDL).
   - Click **File** > **Import WSDL** (or click **Import**) and select a WSDL file that you previously saved locally (using the **File** > **Export WSDL** feature).

   If authentication is required, or if SOAP requests need to be made through a proxy server, see SOAP Editor Settings on page 81 for more information.
3. Click **Parse**.

The SOAP editor lists all discovered operations.



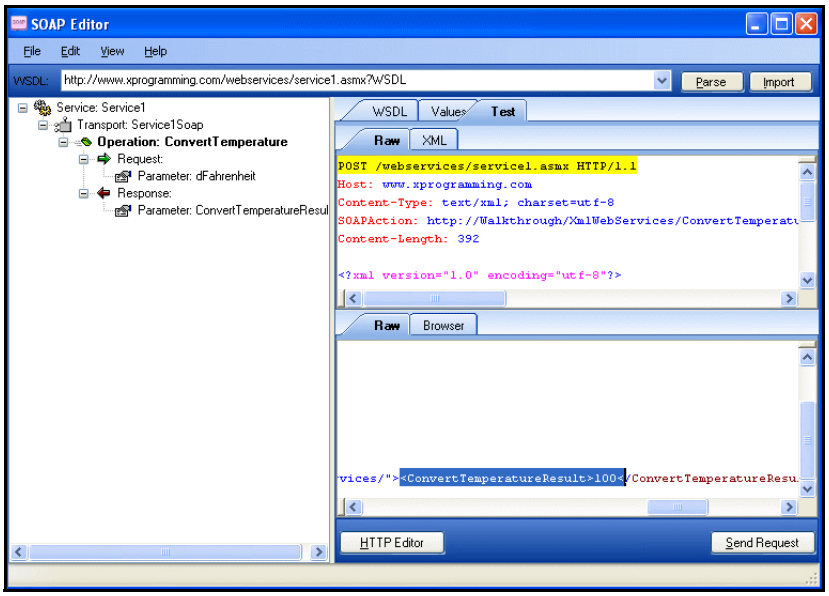
4 Click the **Values** tab and enter the Request parameters you want to submit.



5 Click the **Test** tab.

6 Select an operation. If necessary, edit the raw request on the Request Viewer **Raw** tab (the top pane). View the schema by clicking the **XML** tab.

7 Click **Send Request** to submit the request using the values you entered on the **Values** tab. The Response Viewer (the lower pane) displays the server response.



8 (Optional) Click **HTTP Editor** (at the bottom of the Response Viewer) to view and edit the raw HTTP request. See HTTP Editor on page 62 for more information.

9 To save the values in a file that DevInspect can use when conducting a Web service assessment, click the **File** menu and select **Save Values**.

When performing a Web service assessment, DevInspect crawls the WSDL site and submits an arbitrary enumeration value for each parameter in each operation. It then audits the site by attacking each parameter in an attempt to detect vulnerabilities such as SQL injection.

You can tailor these attacks to your WSDL by creating a file containing specific values that should be submitted, and by selecting the DevInspect option to "Auto-fill SOAP messages during crawl" (select **Edit** > **Default Scan Settings** and then select the Method category in the Scan Settings group). This feature is especially useful when an operation requires submission of a password, license number, or other specific parameters.

You can create one Values file for each WSDL, or you can create a file containing values for all WSDLs that you intend to scan.

If you create individual files, be sure to clear all values before selecting a different WSDL (click the **File** menu and select **New Values**).

SOAP Editor Settings

Use the SOAP Editor's settings to add proxy server and authentication information for requests made via the SOAP Editor.

1 From the SOAP Editor's **Edit** menu, select **Settings**.

The SOAP Editor displays the *Settings* window, which contains two categories in the left pane:

- **Authentication**
- **Proxy**

2 Select a category and enter the information described below.

3 When finished, click **OK**.

Authentication

If the WSDL site does not require authentication, select **None**. Otherwise, select one of the following and enter your credentials:

- **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.

Proxy

Select one of the following options:

- **Direct Connection (proxy disabled)** - Select this option if you are not using a proxy server.
- **Use Internet Explorer proxy settings** - Select this option to import your proxy server information from Internet Explorer.
- **Explicitly configure proxy** - Select this option to access the Internet through a proxy server, using the configuration information you provide. To avoid using a proxy server for certain sites, enter the URL or IP address of those sites in the **Bypass proxy for** box. If you explicitly configure a proxy, you may also configure a proxy for HTTPS.

SOAP Editor Menus

The SOAP Editor contains the following menus:

File

- **Open Values** - Load a Values file into the SOAP Editor. Use this function to edit a Values file or to add values for a different WSDL.
- **Save Values** - Create a file containing values you specify for request parameters.
- **New Values** - Clear all values in the SOAP Editor.
- **Import WSDL** - Load a WSDL file that you previously exported.
- **Export WSDL** - Save the current WSDL file to a local location.
- **Exit** - Close the SOAP Editor.

Edit

Settings - Create or edit SOAP Editor settings.

View

Word Wrap - Adjust lines of text to fit within the available viewing area.

Help

- **SOAP Editor Help** - Open the Help file to the default topic.
- **Index** - Open the Help file, displaying the index pane.
- **Search** - Open the Help file, displaying the search pane.
- **About SOAP Editor** - Open a window that displays information about the SOAP Editor.

Regular Expression Editor

A regular expression is a pattern that describes a set of strings. Regular expressions are constructed similarly to mathematical expressions by using various operators to combine smaller expressions. Only advanced users with a working knowledge of regular expressions should use this feature.

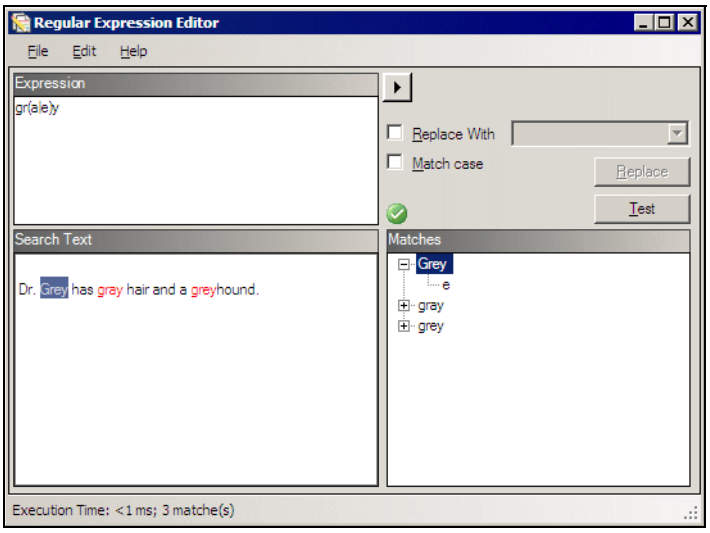
Testing a Regular Expression

Use the Regular Expression Editor to verify regular expressions.

Follow the steps below to use the Regular Expression Editor:

1 Click the Windows **Start** button and select **All Programs > HP > DevInspect for .NET > Regular Expression Editor**.

The *Regular Expression Editor* window opens.



2 In the **Expression** box, type or paste a regular expression that you believe will find the text for which you are searching.

For assistance, click  to reveal a list of objects you can insert by clicking. These include:

- Metacharacters
- Regular expressions that define a URL and an IP address
- Regular expression extensions that you can use to restrict your search to certain areas of an HTTP message

3 In the **Comparison Text** box, type (or paste) the text through which you want to search.

Alternatively, you can load an HTTP request or response message that you previously saved using the HTTP Editor. To do so:

a Click the **File** menu and select **Open Request**.

The Request file is actually a session containing data for both the HTTP request and response.

b Using the standard file-selection window, choose the file containing the saved session.

c Select either **Request** or **Response**.

d Click **OK**.

4 To find only those occurrences matching the case of the expression, select the **Match Case** check box.

5 If you want to substitute the string identified by the regular expression with a different string:

a Select the **Replace With** check box.

b Type or select a string using the drop-down combo box.

6 Click **Test** to search the target text for strings that match the regular expression. Matches will be highlighted in red.

7 If you selected the **Replace** option, click **Replace** to substitute all found strings with the replacement string.

Regular Expressions

Special characters and sequences are used in writing patterns for regular expressions. The following table describes some of these characters and includes short examples showing how the characters are used.

Table 15 Characters Used in Regular Expressions

Character	Description
\	Marks the next character as special. /n/ matches the character “n”. The sequence /\n/ matches a linefeed or newline character.
^	Matches the beginning of input or line. Also used with character classes as a negation character. For example, to exclude everything in the content directory except /content/en and /content/ca, use: /content/[^(en|ca)].*/.* . Also see \S \D \W.
$	Matches the end of input or line.
*	Matches the preceding character zero or more times. /zo*/ matches either “z” or “zoo.”
+	Matches the preceding character one or more times. /zo+/ matches “zoo” but not “z.”
?	Matches the preceding character zero or one time. /a?ve?/ matches the “ve” in “never.”
.	Matches any single character except a newline character.
[xyz]	A character set. Matches any one of the enclosed characters. /[abc]/ matches the “a” in “plain.”
\b	Matches a word boundary, such as a space. /ea*r\b/ matches the “er” in “never early.”
\B	Matches a nonword boundary. /ea*r\B/ matches the “ear” in “never early.”
\d	Matches a digit character. Equivalent to [0-9].

Table 15 Characters Used in Regular Expressions (cont'd)

Character	Description
\D	Matches a nondigit character. Equivalent to [^0-9].
\f	Matches a form-feed character.
\n	Matches a linefeed character.
\r	Matches a carriage return character.
\s	Matches any white space including space, tab, form-feed, and so on. Equivalent to [\f\n\r\t\v]
\S	Matches any nonwhite space character. Equivalent to [^ \f\n\r\t\v]
\w	Matches any word character including underscore. Equivalent to [A-Za-z0-9_].
\W	Matches any nonword character. Equivalent to [^A-Za-z0-9_].

Regular Expression Extensions

Hewlett-Packard 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]
- [ALL]

Regular Expression Operators

- AND
- OR
- NOT
- [ ]
- ( )

Examples

To detect a response in which (a) the status line contains a status code of "200" and (b) the phrase "logged out" appears anywhere in the message body, use the following 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 that you must include a space (ASCII 32) before and after an "open" or "close" parenthesis; otherwise, the parenthesis will be erroneously considered as part of the regular expression.

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

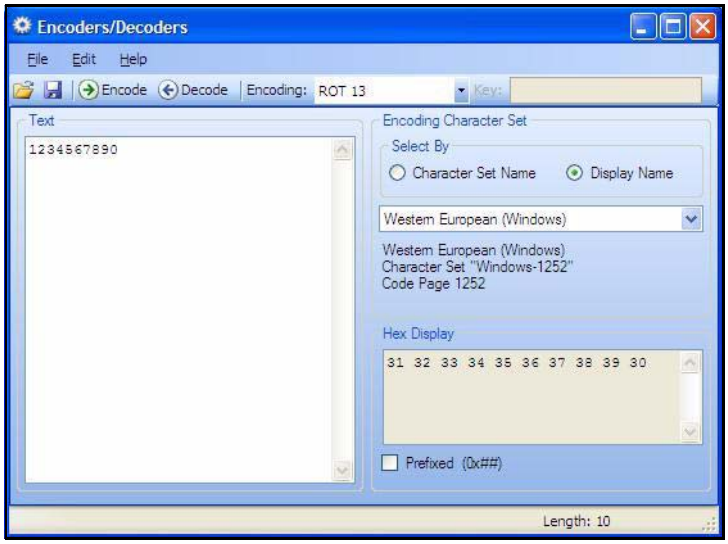
[STATUSCODE]302 AND [HEADER:Location]login.aspx

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

Encoders/Decoders

This tool allows you to encode and decode values using Base64, hexadecimal, MD5, and other schemes. You can also encode a string into a Unicode string and use special characters in URL construction. During the analysis of your scan results, when you encounter a string that you suspect is in an encoded or encrypted format, you can simply copy the string, paste it into the Encoders/Decoders tool, and then click **Decode**.



Encoding a String

Follow the steps below to encode a string:

1. Type (or paste) a string into the **Text** area, or load the contents of a file by selecting **Open** from the **File** menu.
2. Select an encoding character set using either the **Character Set Name** or the **Display Name**.
3. Select a cipher type from the **Encoding** list. For more information, see Encoding Types on page 89.
4. If necessary, type a key in the **Key** box.
5. Click **Encode**.

   The **Text** area displays the encoded string; the **Hex Display** area displays the hexadecimal value of each character in the encoded string (formatted in the character set that you select).

Decoding a String

Follow the steps below to decode a string:

1. Type (or paste) a string in the text area, or load the contents of a file by selecting **Open** from the **File** menu.
2. Select an encoding character set using either the **Character Set Name** or the **Display Name**.
3. Select a cipher type from the **Encoding** list.

4 If necessary, type a key in the **Key** box.

5 Click **Decode**.

You can also use DevInspect's encoding and decoding capabilities in the HTTP Editor. Right-click while editing a session to access encoding and decoding options.

Manipulating Encoded Strings

The encoded form of a string may contain characters that are non-printable. This often occurs when using a hash-based encoding scheme or any encoding scheme that requires a key. Since non-printable characters cannot be copied to the Windows clipboard, you cannot simply copy from or paste into the Encoder/Decoder. However, there are three methods you can use to work around this limitation:

- The easiest method is to copy the hexadecimal representation of the string, as it appears in the right pane. Then open a second Decoder tool, click the **View** menu, and select **Paste Hex**. Then select the appropriate scheme from the **Encoded Type** list, enter the original key in the **Key** box, and click **Decode**.
- You can also save the encoded string to a file and, when you want to decode it, use the **Open** command from the **File** menu to load it into the Encoder tool.
- Also, after encoding the string using the chosen encoding method and key, you can encode the resulting string using the base 64 method; then copy the string to the clipboard, paste the clipboard contents, decode using base 64, and decode again using the original method and key.

Encoding Types

The Encoder/Decoder allows you to select the encoding types described below.

- 3DES is a mode of the DES encryption algorithm that encrypts data three times (the string is encrypted, then the encryption is encrypted, and the resulting cipher text is encrypted a third time). The key must be 8-16 characters.
- Base64 encodes and decodes triplets of 8-bit octets as groups of four characters, each representing 6 bits of the source 24 bits. Only characters present in all variants of ASCII and EBCDIC are used, avoiding incompatibilities in other forms of encoding.
- Blowfish is an encryption algorithm that can be used as a replacement for the DES algorithm.
- DES (Data Encryption Standard) is a widely-used method of data encryption that can use more than 72 quadrillion different private (and secret) encryption keys. Both the sender and the user must use the same private key.
- Hex is hexadecimal.
- MD5 produces a 128-bit "fingerprint" or "message digest" of whatever data you enter.
- RC2 is a variable key-size block cipher designed by Ronald Rivest. It has a block size of 64 bits and is about two to three times faster than DES in software.
- RC4 is a stream cipher designed by Ronald Rivest. It is a variable key-size stream cipher with byte-oriented operations. Used for file encryption in products such as RSA SecurPC and also used for secure communications, as in the encryption of traffic to and from secure Web sites using the SSL protocol.

- ROT13 is a simple Caesar cipher used for obscuring text by replacing each letter with the letter thirteen places down the alphabet.
- SHA1 is Secure Hash Algorithm, a one-way hash function developed by NIST and defined in standard FIPS 180. SHA-1 is a revision published in 1994; it is also described in ANSI standard X9.30 (part 2).
- SHA-256 uses 256-bit encryption.
- SHA-384 uses 384-bit encryption.
- SHA-512 uses 512-bit encryption.
- ToLower changes upper-case letters to lower-case.
- ToUpper changes lower-case letters to upper-case.
- TwoFish is an encryption algorithm based on an earlier Blowfish.
- Unicode provides a unique number for every character, regardless of the platform, program, or language.
- URL creates values that can be used for URL-encoding non-standard letters and characters for display in browsers and plug-ins that support them.
- XHTML encapsulates the entered data with text tags: <text>data</text>
- XOR performs an Exclusive OR operation. You must provide a key. If the length of the key string is only one character, that character is ORed against each character in the encode/decode string.

Prefixed

C and languages with a similar syntax (such as C++, C#, Java and JavaScript) prefix hexadecimal numerals represented with “0x” (for example, 0x5A3). The leading zero allows the parser to recognize a number, and the “x” stands for hexadecimal.

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 16 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 16 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 16 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.

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.

Scrubbers

The Microsoft ASP.NET framework provides three scrubbing functions designed to protect against SQL injection and cross-site scripting. These functions are Server.HtmlEncode(input), HttpUtility.HtmlEncode(input), and RegEx.IsMatch.

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 user name 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.

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).

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 path name 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 "user name" 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

T

U

V

W

X

Z