



Web Proxy

By



Table of Contents

| | | |
|----------|---|-----------|
| 1 | Introduction..... | 5 |
| 1.1 | What is SURU? | 5 |
| 1.2 | Method of operation..... | 5 |
| 1.3 | What makes it different from other web application proxies? | 5 |
| 2 | Setup | 5 |
| 2.1 | Installation | 5 |
| 2.2 | Fuzz DB location | 8 |
| 2.3 | Browser setup | 8 |
| 2.3.1 | Microsoft Internet Explorer setup..... | 8 |
| 2.3.2 | For Mozilla/Fox setup | 10 |
| 3 | The Interface..... | 11 |
| 3.1 | Overview | 11 |
| 3.2 | Proxy | 11 |
| 3.2.1 | Windows..... | 11 |
| 3.2.2 | Filters..... | 13 |
| 3.2.3 | Parameter control..... | 14 |
| 3.2.4 | Fuzzy logic trigger..... | 14 |
| 3.2.5 | Request functions | 14 |
| 3.3 | Recon..... | 15 |
| 3.4 | Misc | 16 |
| 3.5 | Config..... | 17 |
| 4 | Browsing..... | 18 |
| 4.1 | Normal browsing..... | 18 |
| 4.1.1 | What the entry in the Browse window means | 19 |
| 4.2 | Functions in the proxy tab..... | 20 |
| 4.3 | Sending requests through Suru | 21 |
| 4.4 | Editing requests..... | 21 |
| 4.4.1 | From the Response window..... | 21 |
| 4.4.2 | From the Request edit window | 22 |
| 4.5 | Replays | 23 |
| 5 | Fuzzing..... | 23 |
| 5.1 | Fuzz Scripts | 23 |
| 5.1.1 | Location..... | 23 |
| 5.1.2 | Format..... | 23 |
| 5.2 | Fuzzing How-to | 24 |
| 5.2.1 | Using fuzz scripts | 24 |
| 5.2.2 | About the Base Response | 27 |
| 5.2.3 | Using AI | 28 |

| | | |
|-------|---|----|
| 5.3 | The meaning in the Decoding the Result Field | 31 |
| 5.4 | AutoGroup and Tolerance Settings..... | 31 |
| 5.5 | Recalculate..... | 33 |
| 5.6 | Export..... | 33 |
| 5.7 | Note on Memory usage (The use of IgFLT and !Store) | 33 |
| 5.8 | Fine tuning using AI conditions | 34 |
| 6 | <i>Recon</i> | 36 |
| 6.1 | The interface..... | 36 |
| 6.2 | Right click menu..... | 37 |
| 6.3 | Job queue | 37 |
| 6.4 | How-To..... | 38 |
| 6.4.1 | Updating directories and files | 38 |
| 6.4.2 | Selecting a scan target | 39 |
| 6.4.3 | What do the results mean? | 39 |
| 6.5 | Using AI | 40 |
| 7 | <i>Misc</i> | 40 |
| 7.1 | Search and replace | 40 |
| 7.1.1 | Request search and replace | 40 |
| 7.1.2 | Response Search and Replace | 41 |
| 7.1.3 | Examples of Real-Life Uses: | 42 |
| 7.2 | Tools | 42 |
| 7.3 | Auto Relationship Discovery | 43 |
| 8 | <i>Configuration</i> | 44 |
| 8.1 | Saved sessions | 44 |
| 8.2 | Setting up Mozilla/Firefox as your request/replay browser | 44 |
| 9 | <i>Content extraction example</i> | 45 |

1 Introduction

1.1 What is SURU?

Suru is an in-line web proxy that can be used as a tool for web developers and security specialists. It can be used for debugging, testing and security assessment purposes for applications that use HTTP and HTTPS protocols. Suru's features include basic browsing interaction, on the fly request and response editing and fuzzy logic functionality (used for brute forcing and testing for web application flaws). The majority of its features cater for web application security reviews - namely directory and file mining, brute forcing, input sanitisation testing, directory indexing, web service testing, etc.

1.2 Method of operation

Suru is a MS Windows based application that runs as an application on a Windows host. By default it binds to a port on the local host from which browser requests are intercepted. The application can also be configured to receive and relay requests and responses from other systems.

1.3 What makes it different from other web application proxies?

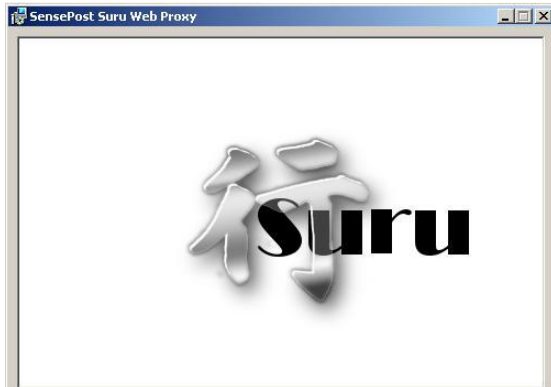
No other web application proxy has as much functionality as Suru. In web application assessments an analyst would typically require a number of tools, e.g. a web proxy, a brute forcer, an application to test for input sanitisation, a tool to for query manipulation, and a web site extractor/spider, etc. The majority of the tools required to audit a web application are included in Suru in a user friendly framework.

2 Setup

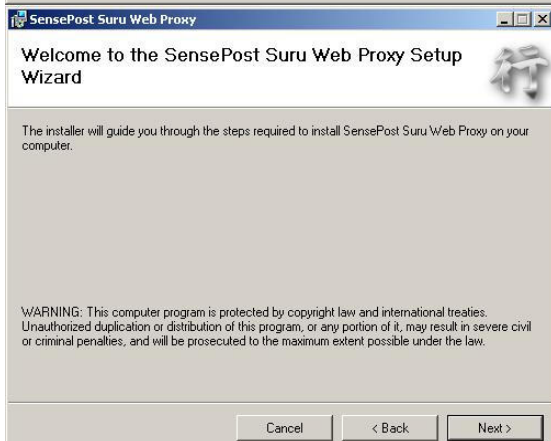
2.1 Installation

Prerequisite Note: Microsoft .Net framework v1.1 is required.

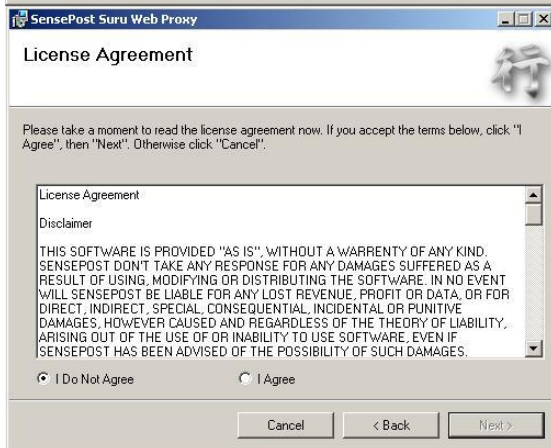
1. Suru comes pre-packaged in an MSI-style install shield which allows for an easier installation. To install: Unpack the Suru zip file and extract it to your file system, or simply browse to the zip archive, if your explorer supports it.
2. Select the `Setup.exe` application, which will initiate the installation process.
3. We recommend that you install Suru in the default directory as the location of the Fuzz database is hard coded within the application. If you choose to install folder other than the default, follow the steps in the next section "Fuzz DB Location" to locate the Fuzz DB folder location.
4. Follow the steps below to complete the installation process.



- Install start-up window
- Click *Next*



- Click *Next*



- License agreement screen
- Read, then select "I agree"
- Click *Next*



- Locate install directory by selecting "Browse". By default Suru installs to C:\Program Files\SensePost\Suru
- "Disk cost" will display your used/free disk space and how much Suru will use.
- Select who will use this application.
- Click *Next*

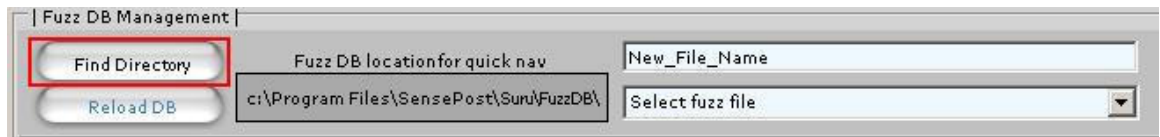
- Click *Next* to begin the installation

- Installation is complete
- Click *Close* to exit

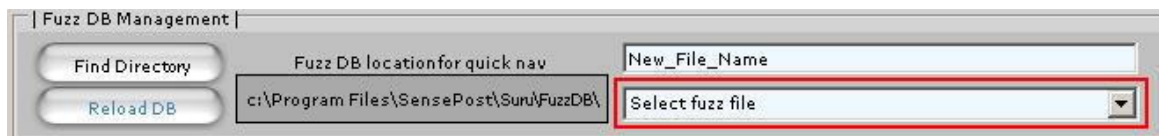
2.2 Fuzz DB location

If you have installed Suru in an alternative directory you will need to locate the "FuzzDB" directory. The "FuzzDB" is the store for the fuzzing scripts which are used during web application testing.

1. Launch Suru (Start>Programs>SensePost>Suru>Suru).
2. Select the "Config" tab.
3. Under the "Fuzz DB Management" section select "Find Directory".



4. This will open up a "Browse for folder" window. Locate the "FuzzDB", which will be in the installed location (<location>\Suru\FuzzDB).
5. Select "Reload DB".
6. Select the drop down list "Select fuzz file" to verify that the scripts have been located. If there are no script files listed, relocate the "FuzzDB" folder.



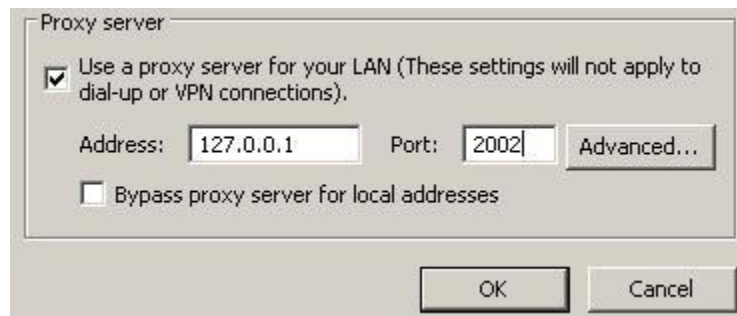
2.3 Browser setup

How to set your browser to use Suru as a proxy.

2.3.1 Microsoft Internet Explorer setup

For a LAN

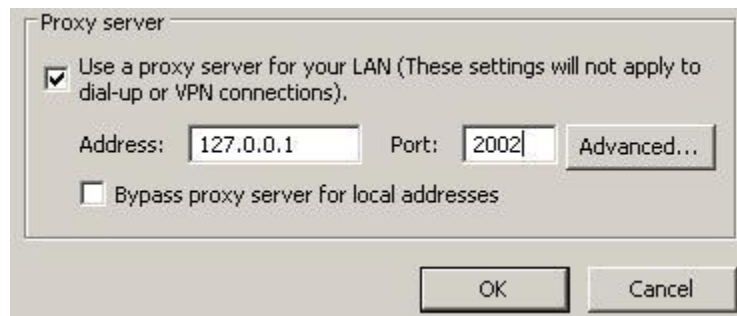
1. Open IE.
2. Select "Tools>Internet options" from the menu bar.
3. Select the "Connections" tab.
4. Click the "LAN settings" button.
5. In the "Proxy Server" section select the check box "Use a proxy server ...".



6. In the Address field enter in 127.0.0.1; In the Port field enter in 2002 (the default port used by Suru. See the Config section on how to change to an alternative port number).

For Dial-up

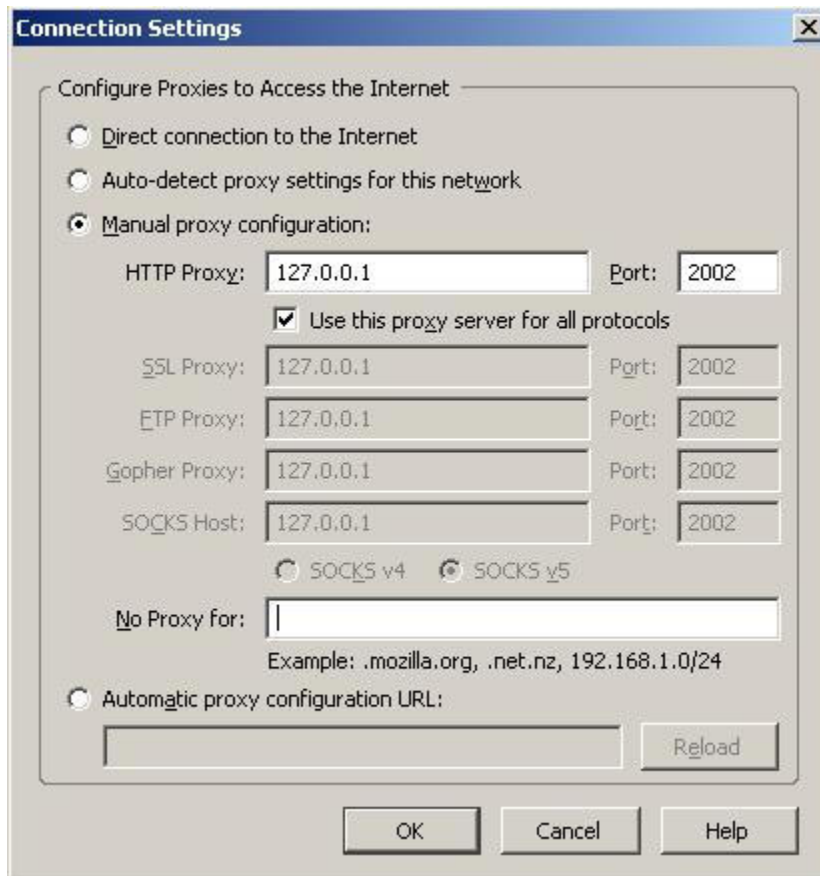
1. Open IE.
2. Select "Tools>Internet options" from the menu bar.
3. Select the "Connections" tab.
4. Select your dial-up connection under the "Dial-up and Virtual Private Network settings" section.
5. Click the "Settings" button.
6. In the "Proxy Server" section select the check box "Use a proxy server ...".



7. In the Address field enter in 127.0.0.1; In the Port field enter in 2002 (the default port used by Suru. See the Config section on how to change to an alternative port number).

2.3.2 For Mozilla/Fox setup

1. Open Firefox
2. Select "Tools>Options" from the menu bar
3. Click "Connection Settings" in the "General" tab
4. Select "Manual proxy configuration"



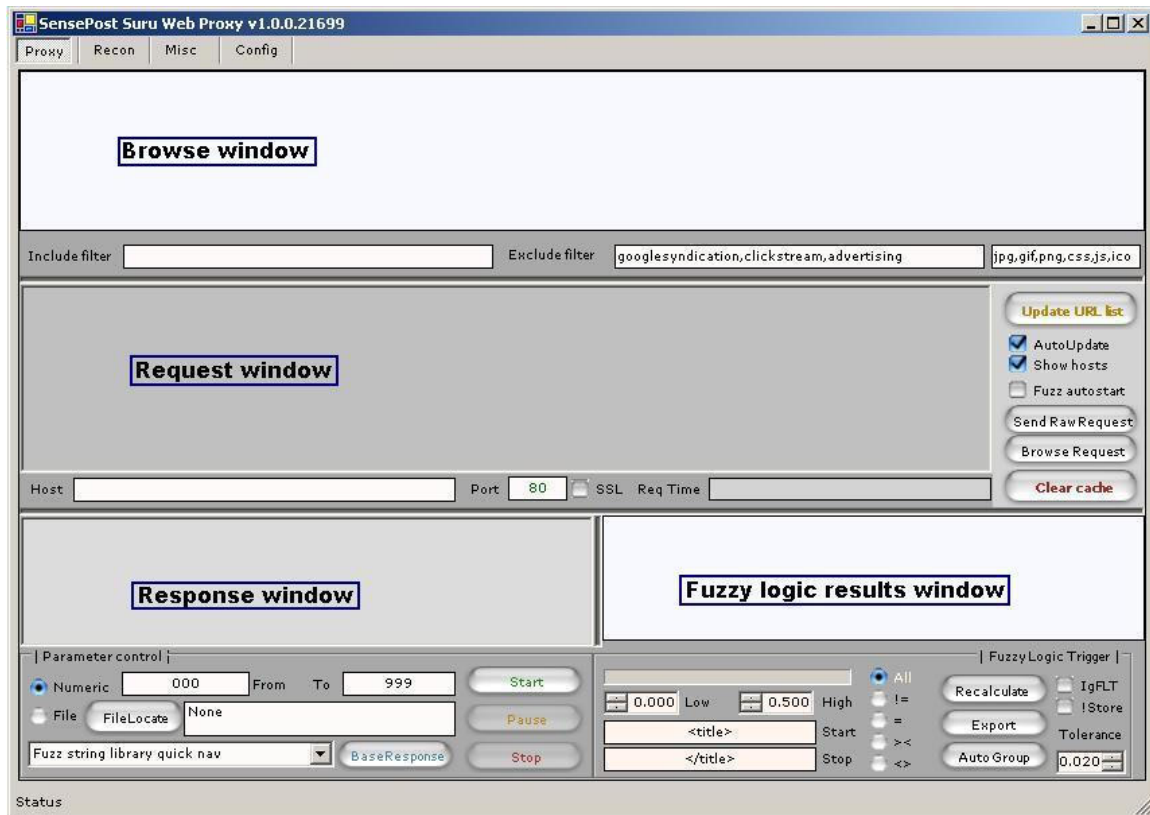
5. In the HTTP Proxy field enter in 127.0.0.1; In the Port field enter in 2002 (the default port used by Suru. See the Config section on how to change to an alternative port number).

3 The Interface

3.1 Overview

The interface is arranged in a single window, the main functions are split up into tabs; Proxy, Recon, Misc, Config. The rest of this section will explain the functionality of each component.

3.2 Proxy



3.2.1 Windows

Browse window

The browse window displays an interactive list of requests you have submitted from your browser or through Suru.

Request window

This window displays the raw request sent to the web server. This window is an editable text box and allows the user to edit any part of the HTTP request.

Response window

This is the response received from the server per request sent to the server.

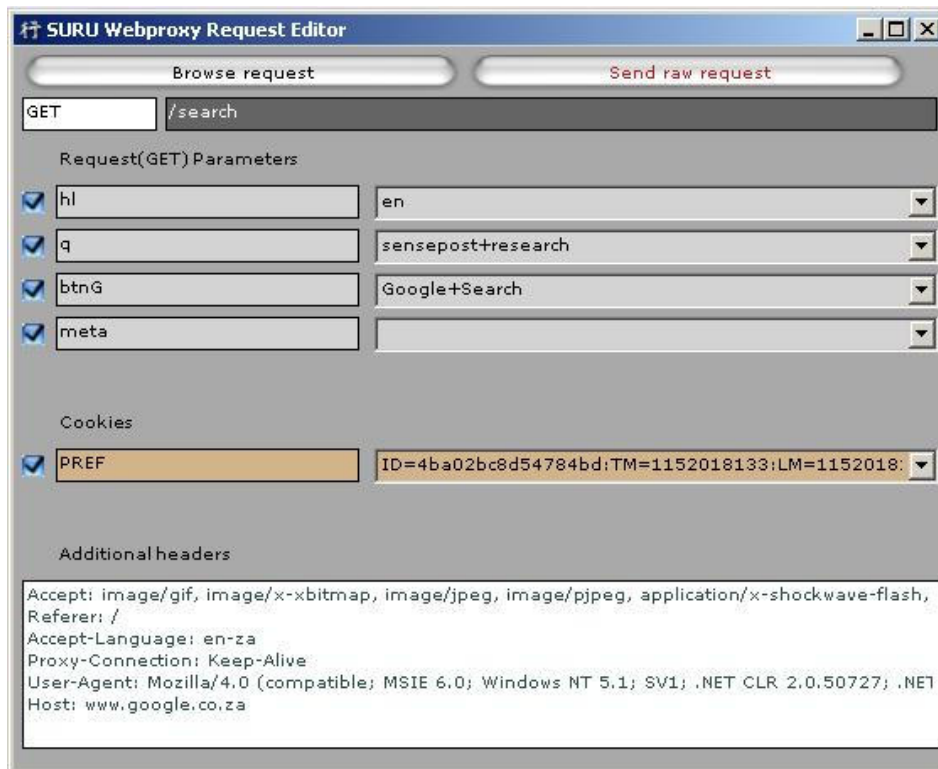
Note: For normal browsing only the header is displayed. For fuzzing sessions the entire fuzzed response will be displayed.

Fuzzy logic results window

Results from a fuzzing session are displayed in this window. Results displayed in this window can be filtered by modifying the filters in the Fuzzy Logic Trigger section below the window.

Request edit window

When double-clicking on an entry in the browse window, the following window is displayed:

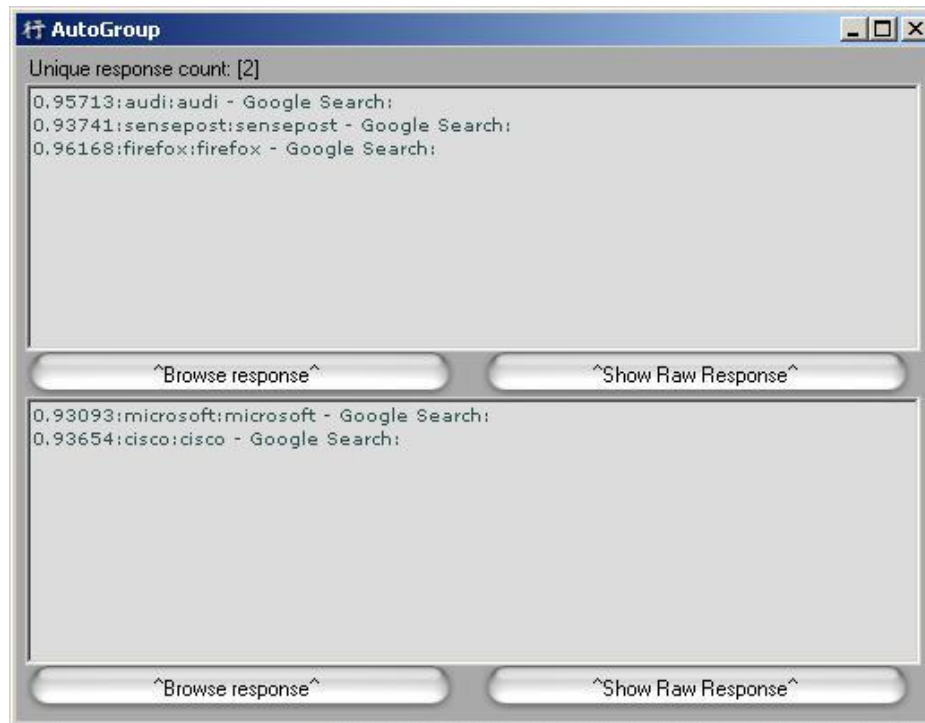


The screenshot shows the 'SURU Webproxy Request Editor' window. It has a title bar with standard window controls. Below the title bar are two buttons: 'Browse request' and 'Send raw request'. The main area is divided into sections. The first section shows the request method 'GET' and the URL '/search'. Below this is the 'Request(GET) Parameters' section, which contains four rows of parameters: 'hl' with value 'en', 'q' with value 'sensepost+research', 'btnG' with value 'Google+Search', and 'meta' with an empty dropdown. Each parameter has a checked checkbox to its left. Below the parameters is the 'Cookies' section, which contains one row: 'PREF' with value 'ID=4ba02bc8d54784bd;TM=1152018133;LM=1152018', also with a checked checkbox. The bottom section is 'Additional headers', which contains a text area with the following headers: 'Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, Referer: /', 'Accept-Language: en-za', 'Proxy-Connection: Keep-Alive', 'User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; .NET', and 'Host: www.google.co.za'.

This window is used to edit the request parameters in a user friendly manner. Any entries in the Request edit window can be modified. This window can also be used to mark a parameter fuzz point by selecting the drop down list associated with each parameter. This window is explained in greater detail in Section 4.4.2 Editing from the Request Edit window.

Auto Group window

Results from a fuzzing session can be easily displayed in this window, which will pop-up after the Auto Group button is selected (In the Fuzzy Logic Trigger section). This window displays the results from a fuzz session in a more optimised manner. It also gives the user the ability to browse the requests for each of the results.



3.2.2 Filters

Results in the browse window can be filtered to suite the user. For example, a user can choose not to display requests for images in the browse window, etc.

| | | | | |
|----------------|----------------------|----------------|--|---|
| Include filter | <input type="text"/> | Exclude filter | <input type="text" value="googlesyndication,clickstream,advertising"/> | <input type="text" value="jpg,gif,png,css,js,ico"/> |
|----------------|----------------------|----------------|--|---|

Include

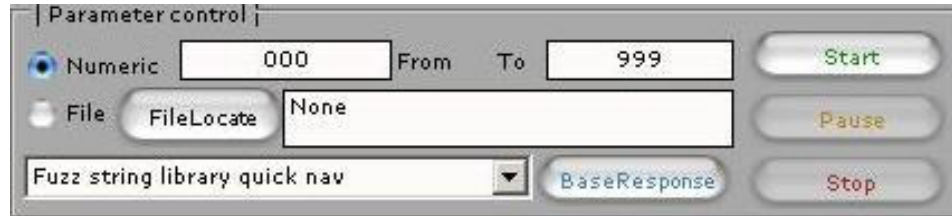
The include field will ensure that any pattern in the URI is matched to the include pattern field. Fields are separated by a comma (,). Be sure not to include spaces unless actually required.

Exclude

The exclude field as two options; a pattern field and a file type field. Entering in either patterns or file types will cause the requests matching the patterns to be excluded in the browse window.

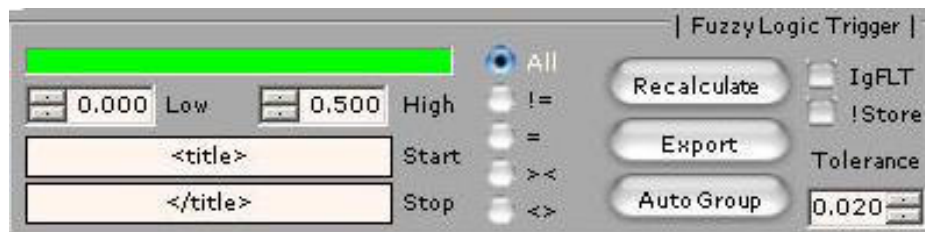
3.2.3 Parameter control

The parameter control box is used for fuzzing. The user has the ability to fuzz various parameters including numeric fields, imported word lists and even fuzz strings for testing input sanitisation issues in web applications.



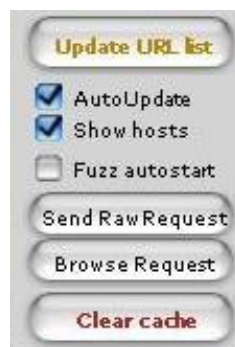
3.2.4 Fuzzy logic trigger

The Fuzzy logic trigger box is used to fine-tune the results received during a fuzzing run. The user has multiple options that range from content extraction to brute force fine-tuning. The results from the fuzzing run can be mined or filtered for false positives, positives or false negative findings. This is explained in greater detail in later sections.



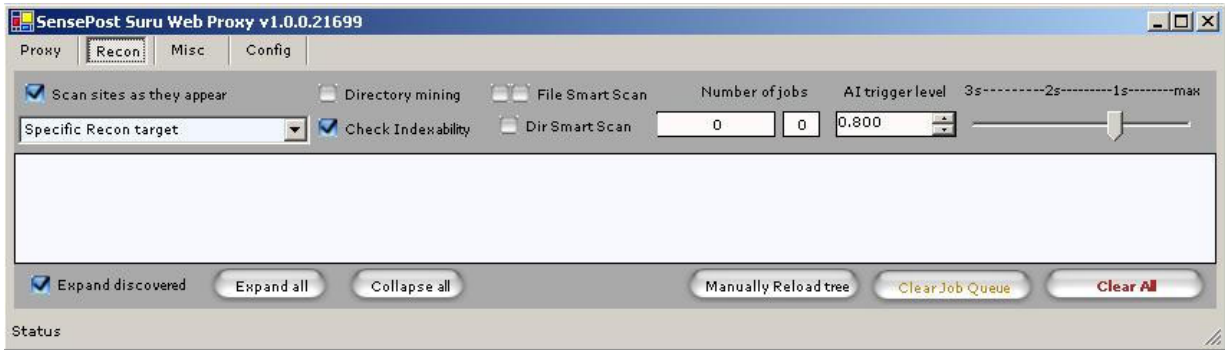
3.2.5 Request functions

The actions on the right hand side of the Proxy window are used to update the browser list, send requests, clear the proxy cache and enable auto fuzzing. These functions are explained in greater detail later in this manual.



3.3 Recon

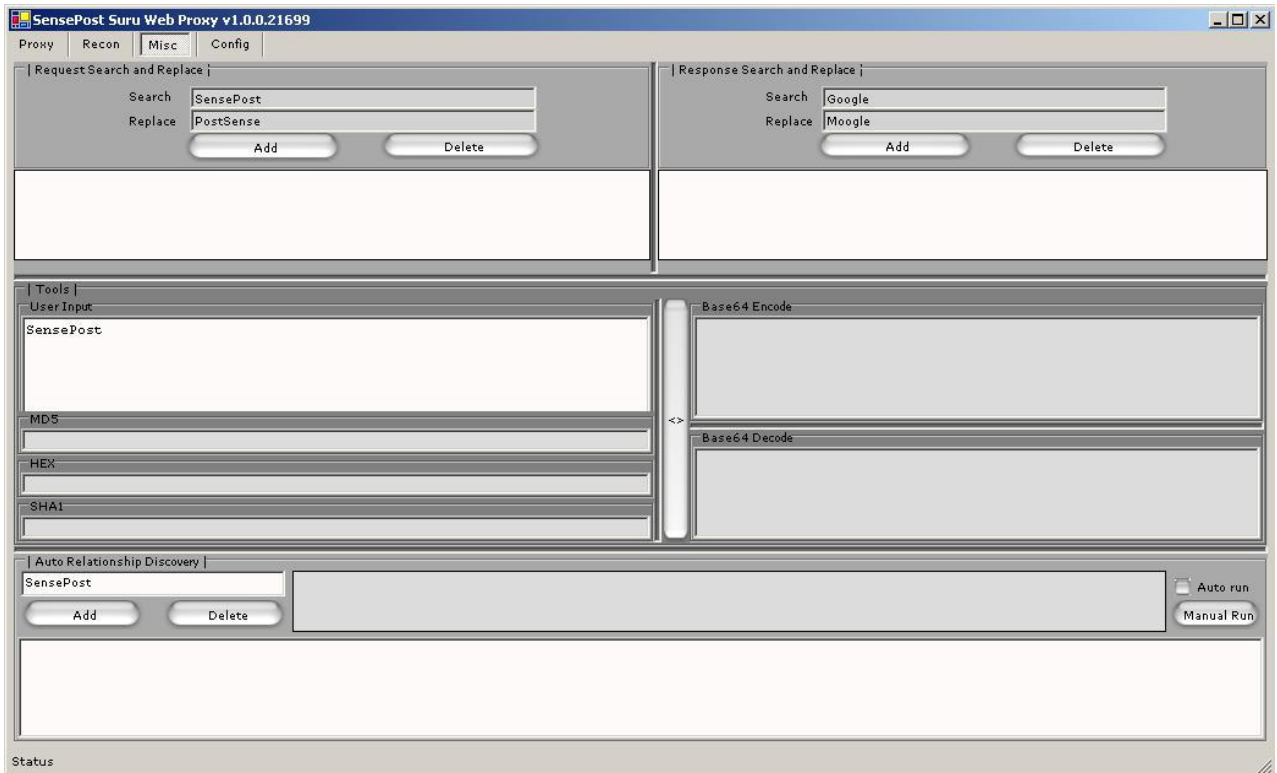
Functions in the recon tab allow the user to conduct directory and file mining. After the user has browsed sites, the main window will display a tree view of the pages. The user can now use the results from the browsed sites to conduct 'black box' directory and file mining, this would typically be used to locate hidden files or even admin back ends.



3.4 Misc

The Misc tab contains additional tools that can be of value when conducting web application tests. The user has the ability to search and replace any fields in a response or request, i.e. modifying request or response headers to include specific data such as altering a request to contain a different user agent in its HTTP header. Also included are conversions from text to different types of encoding and hashes.

The Auto Relationship Discovery box is an added function for determining relationships in sessions that would normally require human intervention. E.g. a site uses a SHA1 hash of the username as a session cookie field. This feature will automatically compare 'browsed/mined' data with parameters in the request.

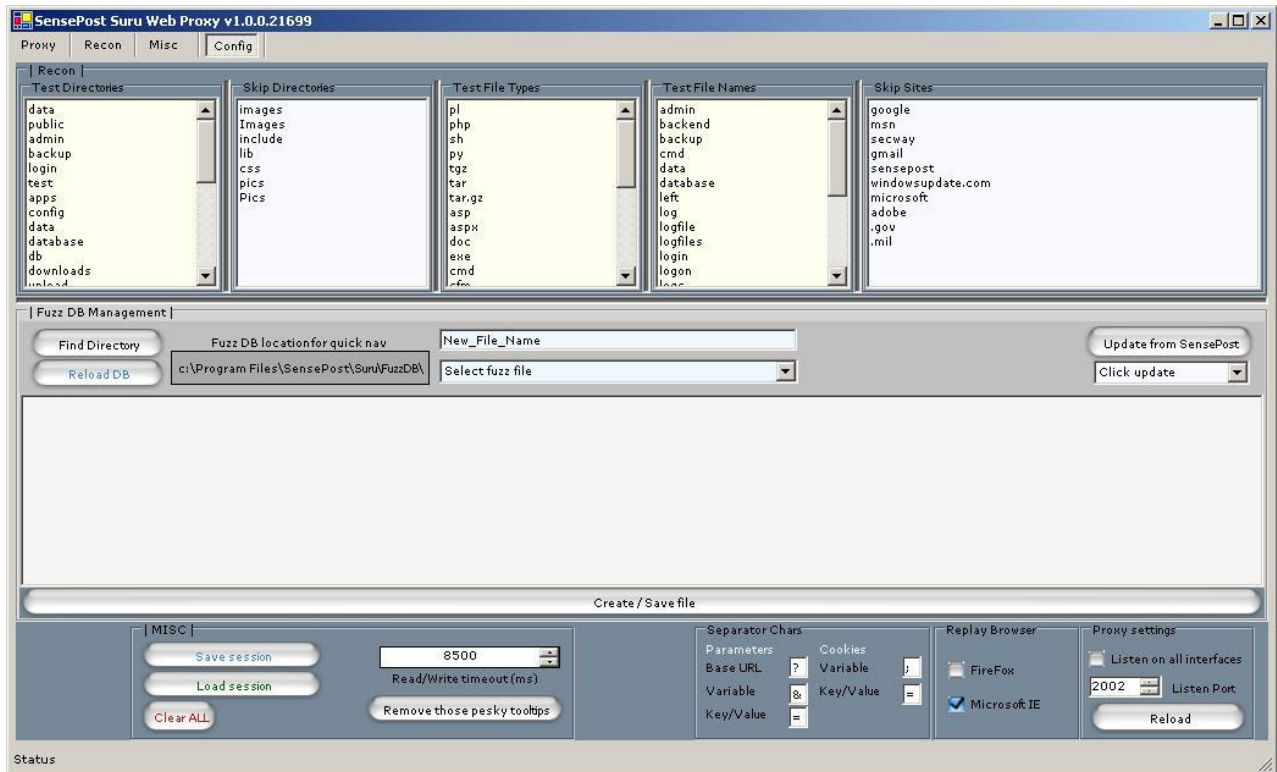


3.5 Config

The Config tab proxy configuration options. The Recon box lists the fields used for the directory and file mining. Updated lists can be updated from SensePost or custom entries can be manually entered to the respective window. The Fuzz DB Management box is used to load and configure the fuzz scripts used in fuzzing sessions. Here you can load generic scripts (e.g. a dictionary list) or build your own custom fuzz files.

Other configurations include:

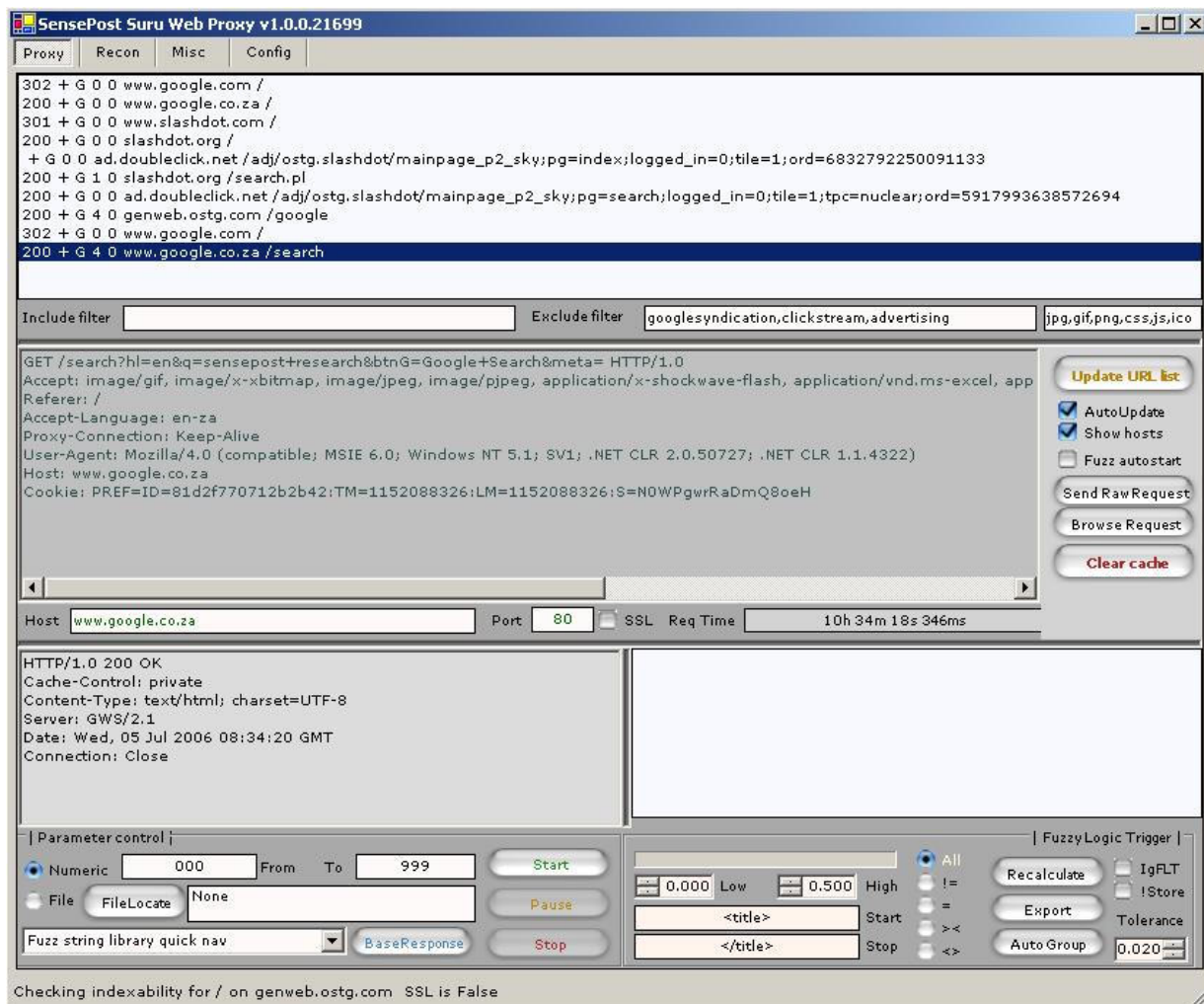
- Misc – where you can save or load sessions, read/write timeouts, toggle on/off tooltips.
- Separator chars – If you are browsing a non-standard web application this is where you can modify the key-value pair separator used in the request parameters and cookies.
- Replay browser – choose which browser you wish to replay your request in.
- Proxy settings – Set Suru to listen on a different port or enable it to listen on other interfaces on your host.



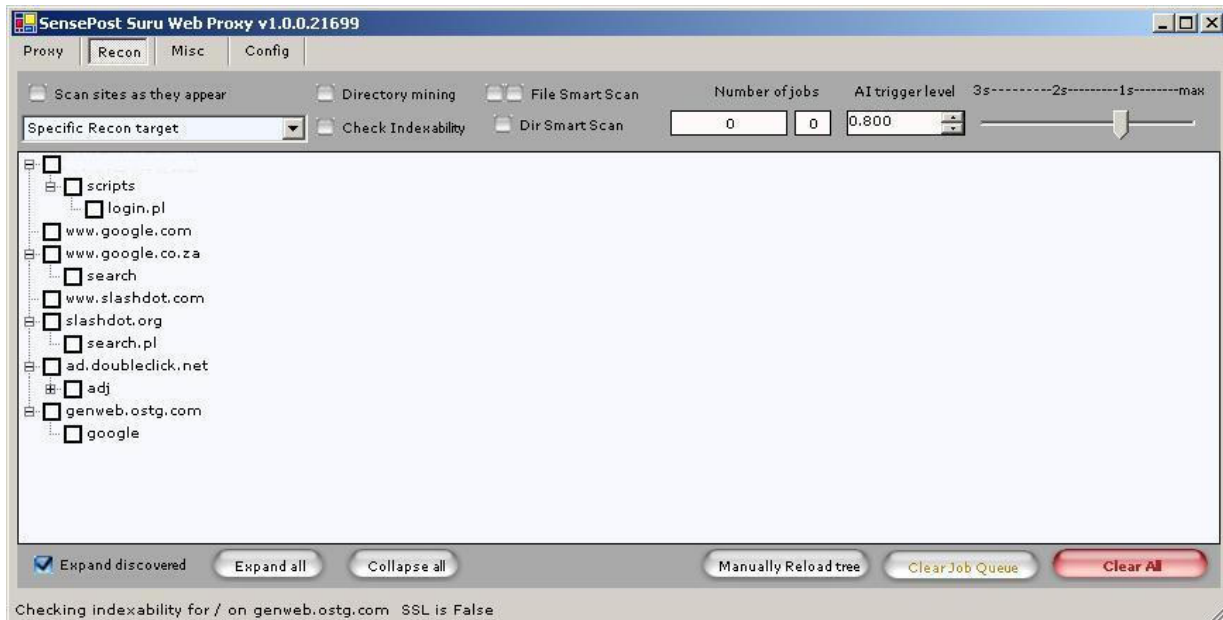
4 Browsing

4.1 Normal browsing

1. Start up your browser and begin surfing.
2. The requests from the browser will be displayed in the Browse window under the Proxy tab.
3. Selecting a result from the Browse window will result in the request being displayed in the Request window and the respective response from the server displayed in the response window.
4. The Request window is an editable text box, which allows the user to modify any of the parameters for a selected HTTP request.



- If you select the Recon tab, a tree view of the browsed sites will be displayed in the main recon window.



4.1.1 What the entry in the Browse window means

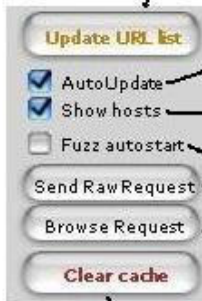
Results displayed in the Browse window will be similar to the following –

200=+ G 4 0 www.google.com /search

- 200 is the HTTP status code extracted from the HTTP header in the response received.
- = indicates whether you have manually edited the response. This helps to identify which requests have been modified.
- + indicates the request was received from a browser.
- G indicates that a GET request was submitted. Other possible values are:
 - P for a POST
 - X for XML (web services)
 - MP for Multipart
- 4 is the number of GET parameters sent to the server.
- 0 is the number POST parameters sent to the server.
- www.google.com is the host the request was sent to. Unchecking the "Show Hosts" checkbox under the Proxy tab, will remove the host from the browse field.
- /search, is the URL path and page for the requested page.

4.2 Functions in the proxy tab

**Updates the Browse window with the latest requests.
Only required if "AutoUpdate" is not checked**



Toggles the automatic updating of the Browse window. By default this is enabled. When editing a request in the Request window or in the Request Edit window, automatic updating will be disabled

By default this is checked. In the Browse window, the host is listed in the browse field. If this is unchecked the host will not be displayed. (i.e. Used when only testing a single site, because showing the host is not necessary)

When checked, the application will automatically begin fuzzing using the selected Fuzz file, after the "FUZZCTRL" marker is inserted in the request.

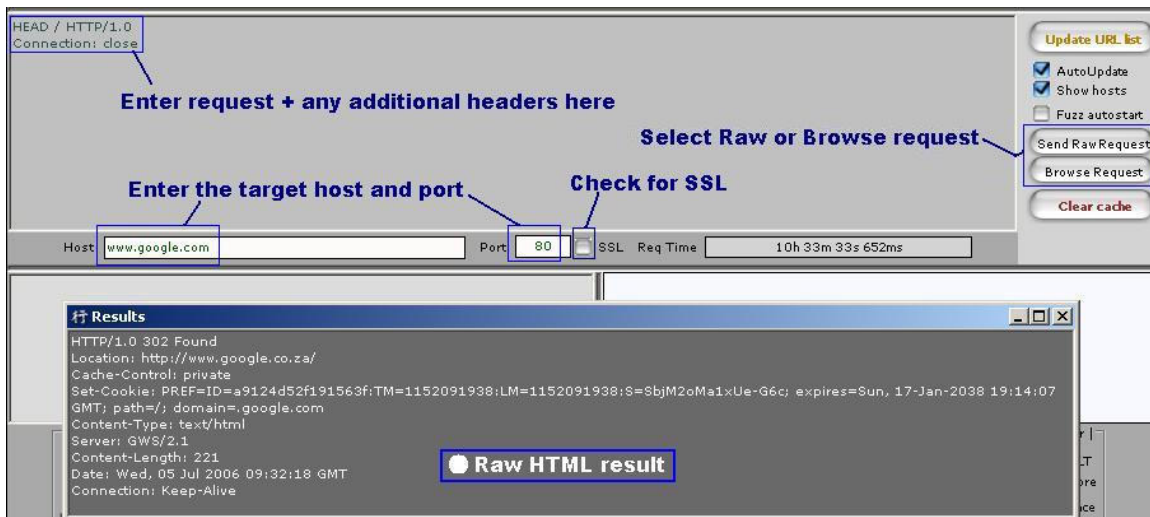
Sends a request and renders the response in raw HTML format

Sends a request and renders the result in the replay browser

Clears all entries in the Browse window. WARNING! It does not ask you if you are sure or not. The cache will disappear after this button is clicked

4.3 Sending requests through Suru

1. Complete the GET/POST request in the Request window including any additional headers.
2. Enter the target host under the Host field.
3. Select the port of the target server. And check the SSL checkbox if the site is using SSL.
4. Select either "Send Raw Request" for a raw HTML response or "Browse Request" to render the results in a browser.



4.4 Editing requests

4.4.1 From the Response window

1. Select a request from the Browse window.
2. The Response and Request window will update with the respective entry.
3. Any detail in the Request window can be modified or deleted.
4. After modifying, select either "Send Raw Request" or "Browse Request"
5. Note that once the user has modified an entry in the Request window the "AutoUpdate" checkbox will automatically be disabled. This is done to prevent confusion when modifying multiple requests.
6. After submitting the request the user must either select "Update URL list" or check "AutoUpdate" to refresh the results in the Browse window.



From the figure below you will see the new entry has an = sign after the HTTP code to identify that the request was modified.

```
200 + G 3 0 www.google.co.za /search
200=+ G 3 0 www.google.co.za /search
```

4.4.2 From the Request edit window

1. Double-click an entry in the Browse window to open up the Request Edit window.
2. Entries in the Request Edit window can now be modified.
3. After modifying the request there is an option to submit via a raw request or to render the response in the replay browser.

| | |
|------------------|--|
| GET/POST field | The user has the option of changing the HTTP action parameter (i.e. to use POST, PROFIND or even a PUT instead of a GET action). |
| Check boxes: | The checkboxes next to each parameter and be unchecked to exclude the parameter from the request. |
| Parameter name: | A text field that can be modified to user requirements. |
| Parameter value: | Is a List box that allows the user to add a custom value or mark the parameter as a fuzz |

| | |
|--------------------|---|
| | point, which includes predefined fuzz files from the FUZZ DB or a custom fuzz file. |
| Additional headers | By default Suru will extract and display the session header, which can be modified in the additional header section (E.g. the Host: field, if we are testing for virtually hosted sites). |

4.5 Replays

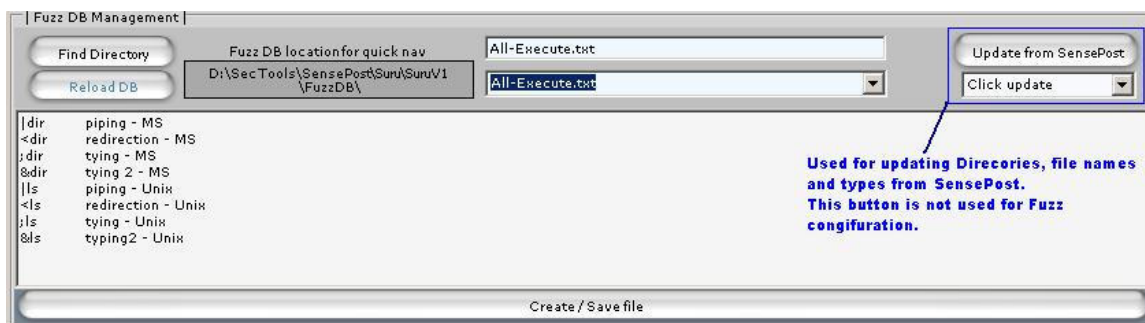
To replay requests simply select the entry from the Browse window and select "Send Raw Request" or "Browse Request".

5 Fuzzing

5.1 Fuzz Scripts

5.1.1 Location

The default location of the Fuzz DB, which contains fuzz files, is in *<installed location>\SensePost\Suru\Fuzz DB*.



- You can verify/relocate the Fuzz DB by selecting the Config tab.
- The "Fuzz DB Management" section will display the Fuzz DB location, if incorrect. Click on "Find Directory" to locate your Fuzz DB.
- Selecting the drop down list to the right of the "Fuzz DB location" will list the fuzz files in the Fuzz DB.
- Selecting one of the files will display the contents of the file in the window below.
- The field above the drop-down list displays the name of the selected file.
- The button "Reload DB" is used to update additional files added to the Fuzz DB folder.
- Fuzz files can be created or modified in the main window. After modification/creation of a fuzz file, click on "Create / Save file" to save the file to the Fuzz DB.

5.1.2 Format

Alternatively a user can create custom fuzz scripts and store it in the Fuzz DB folder. Below describes the construction of the fuzz script.

- Comments are marked with a double hash (##).
- First part of the fuzz parameter is the fuzz string.

- The second part of the fuzz parameter is a comment.
- The fuzz string and the fuzz comment are separated by a TAB.

| Fuzz file structure | |
|---------------------|--|
| ## | Comment begins with TWO hashes |
| ## | NO blank lines |
| ## | Comment fuzzstring - tab separator (any amount of) |
| ## | |
| ## | SQL injection section-----> |
| admin'-- | SQL 101 - test for admin with comment |
| admin' | SQL 101 - test for admin without comments |
| 'test | SQL injection test |
| 'test-- | SQL injection test with comments |

5.2 Fuzzing How-to

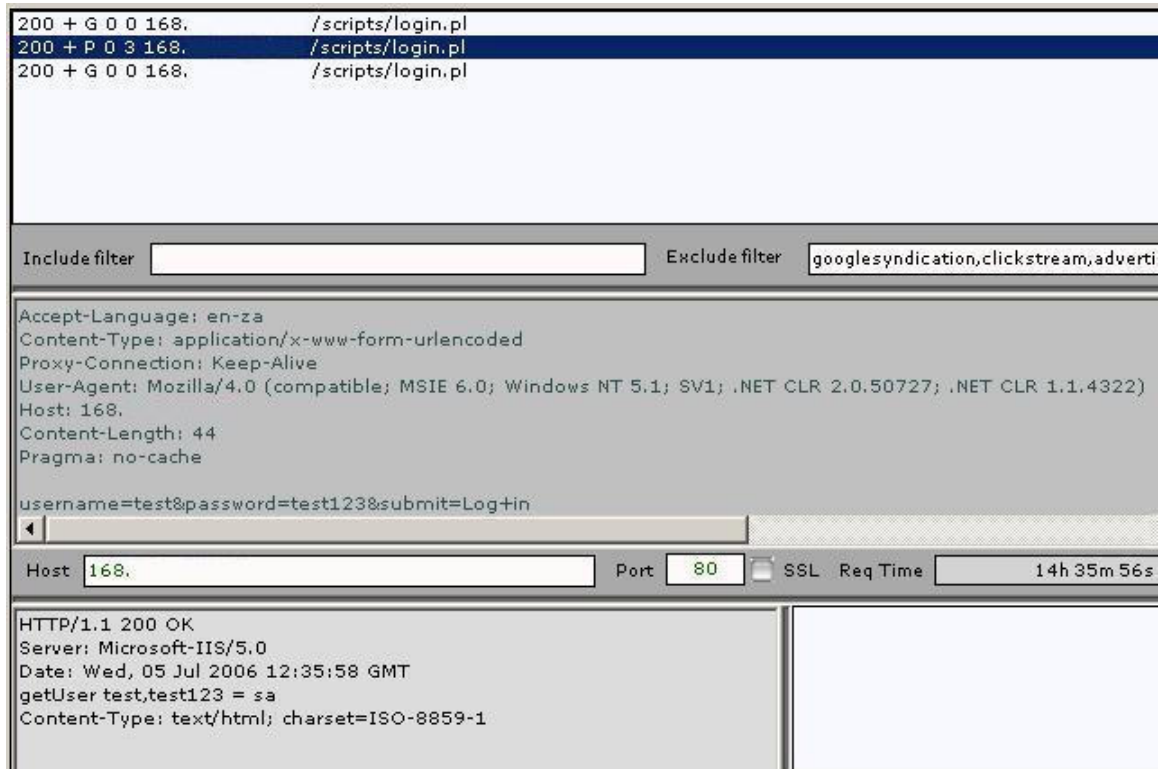
5.2.1 Using fuzz scripts

The example below will describe how to use Suru for fuzzing or brute forcing web requests.

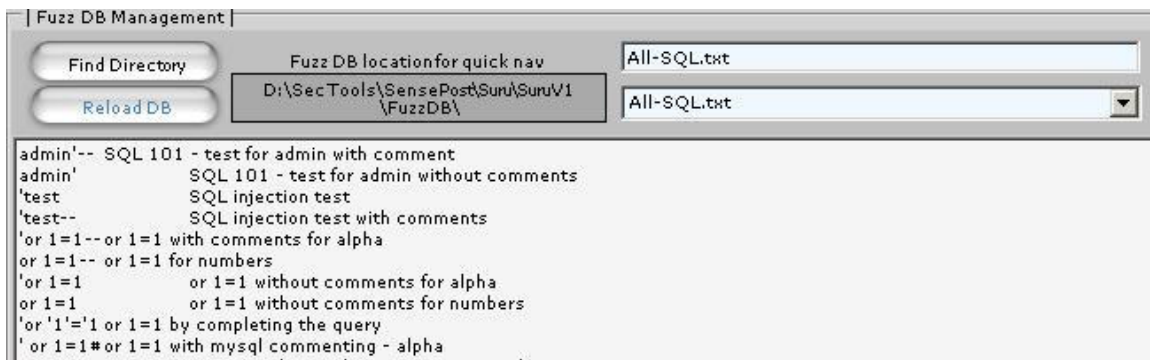
- Browse to the target site. And submit a request to the server.



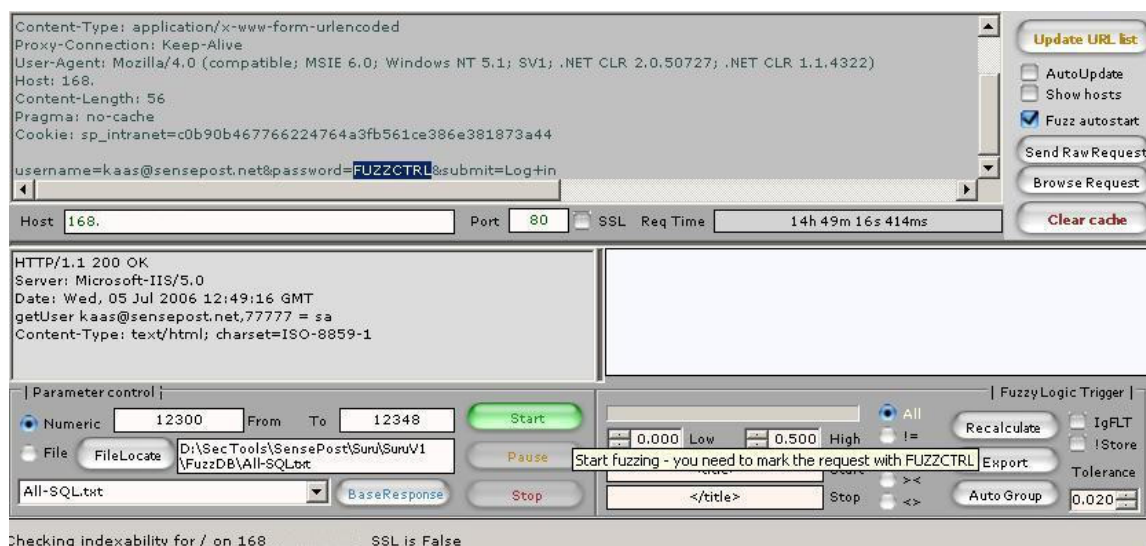
- The request will pass through Suru and show in the Browse window.



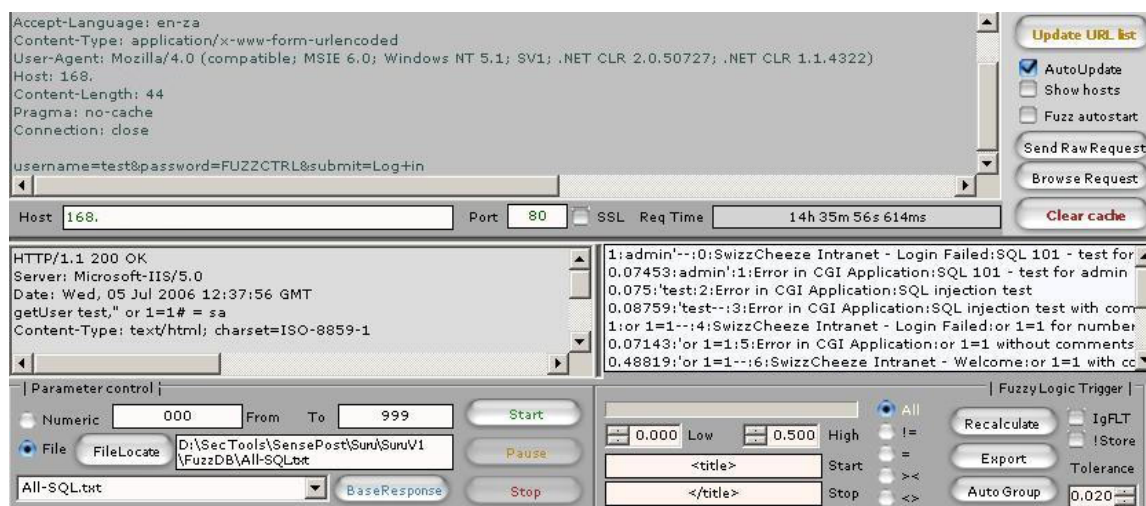
- Select the Config tab to build your Fuzz file (or alternatively you can use the pre-generated fuzz files and you can skip the Fuzz DB Management section).



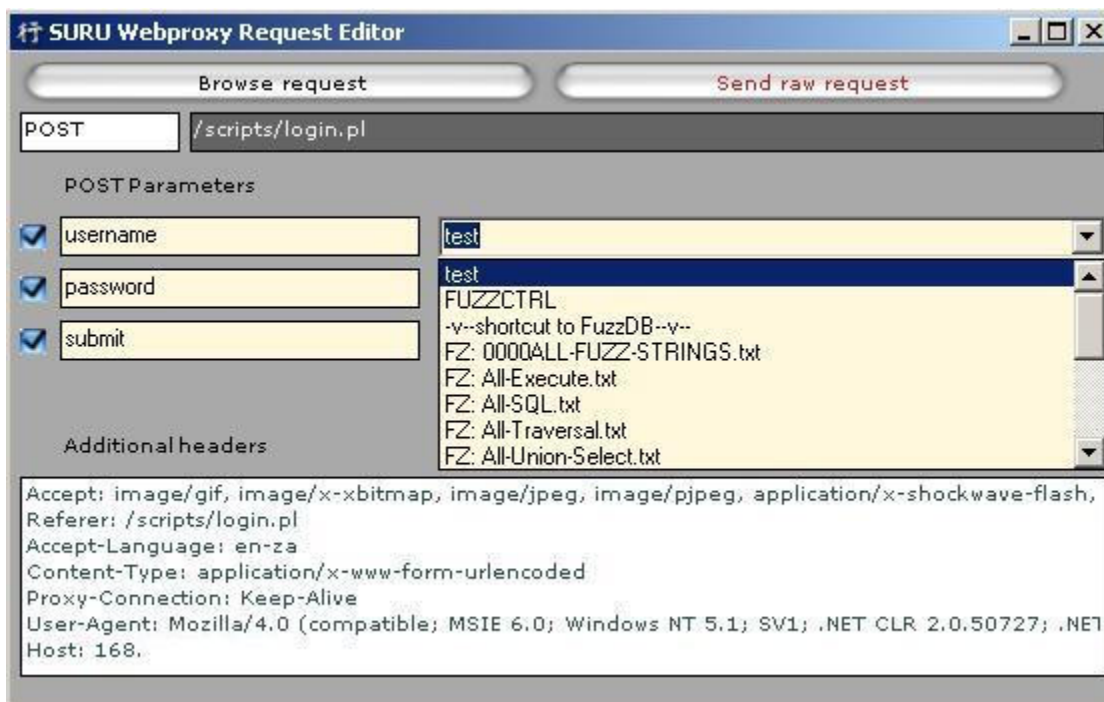
- Select the request required for fuzzing from the Browse window. The request will show up in the Request window.
- Mark the fuzz point with the fuzz marker "FUZZCTRL".
- In the Parameter control section select the fuzz parameters. At this point there are two parameter options which can be selected:
 - A numeric fuzz
 - A fuzz file (either from the Fuzz DB or from an alternative location – e.g. a dictionary list).



- Click "Start".
- The results of the fuzz session will be displayed in the Fuzzy Logic Results window.



- An alternative method to conduct fuzzing is to double-click on a request in the Browse window, which will open the Request Edit window.
- From the drop-down list next to each parameter, the Fuzz control point or a fuzz file can be set.
- Once the "Browse request" or "Send raw request" is clicked Suru will begin fuzzing. The results of which will be displayed in the Fuzzy Logic Results window under the Proxy tab.



5.2.2 About the Base Response

In order to conduct Fuzz logic or content extraction runs, Suru needs to compare the results to a base response. For example if we need to brute force an application we need to compare a failed attempt with a successful attempt to gauge successes or failures.

Suru takes the content received from the base response and puts it through a content algorithm which generates a base number. The content received is also passed through the same algorithm. Comparison between the base number and the fuzzed result can now be achieved (i.e. like compared with like). By setting the AI limits the user can tune the AI triggers to display fuzzed results that are the same as the base response, differing slightly or are totally different from the base response.

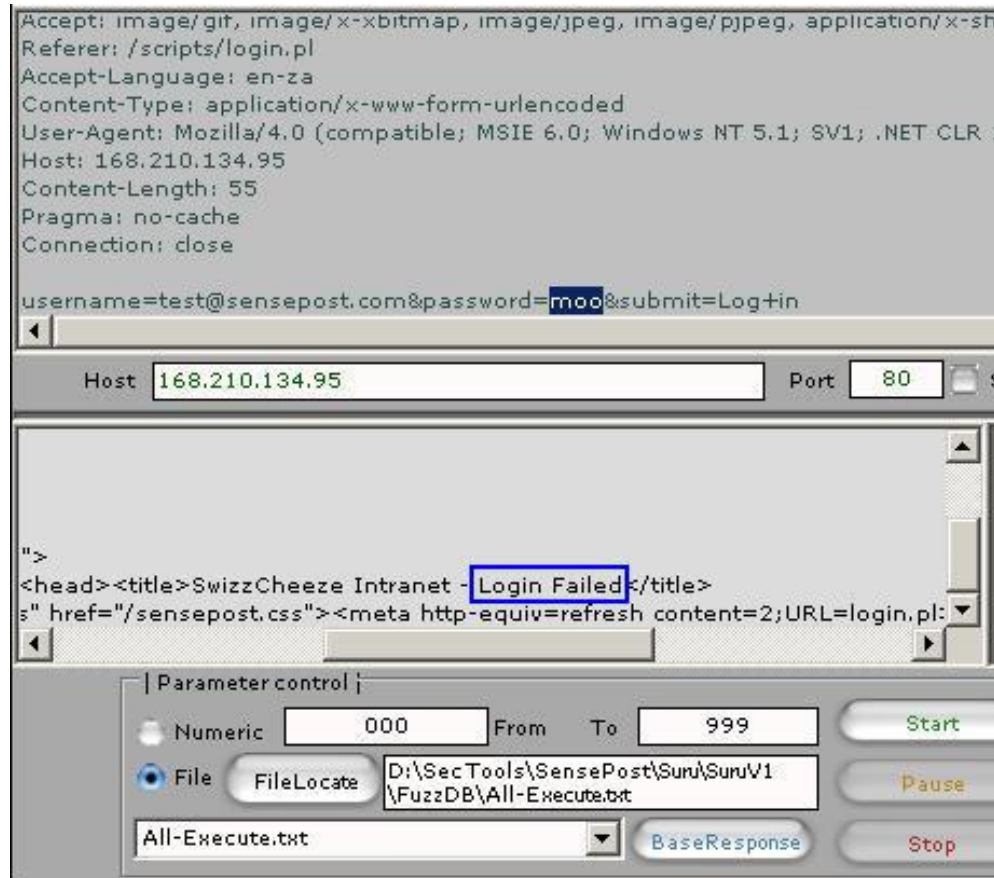
By default Suru submits the first request with the "FUZZCTRL" marker as the base request, if the Start button is clicked without pre-generating a base response. The user may want to compare a different base response, for instance if the web application is returning different error messages for different requests. So the response received from the application may not result in desired results.

To generate a specific base response which can be compared to fuzz responses simply generate a request in the Request window and hit the "Base Response" button. This will then become the base response from which the fuzz results will be compared. Thereafter the "FUZZCTRL" market can be added.

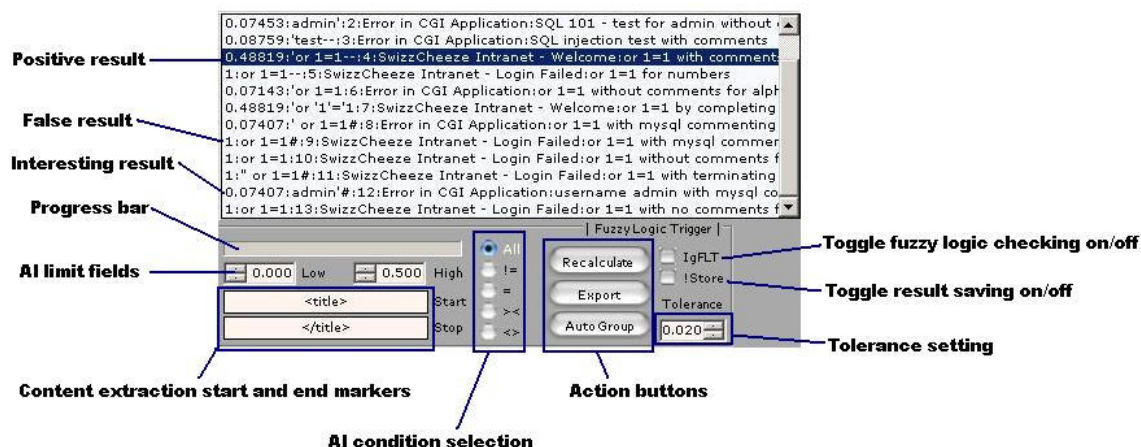
5.2.3 Using AI

AI is used to fine tune the results from a fuzzing session.

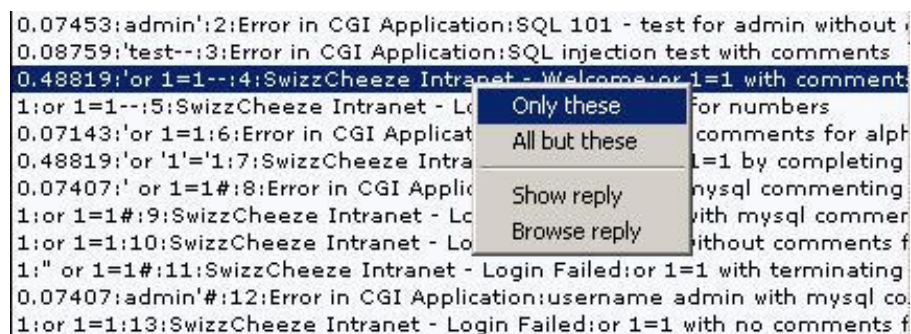
- We input *moo* into the password field (which is an invalid password) and hit "Base Response". The response received now becomes our base response from which the fuzz requests can be compared.



- Select our Fuzz file, mark the "FUZZCTRL" point (which is the password value) and click "Start".
- From the image below we can see the results trickling in.



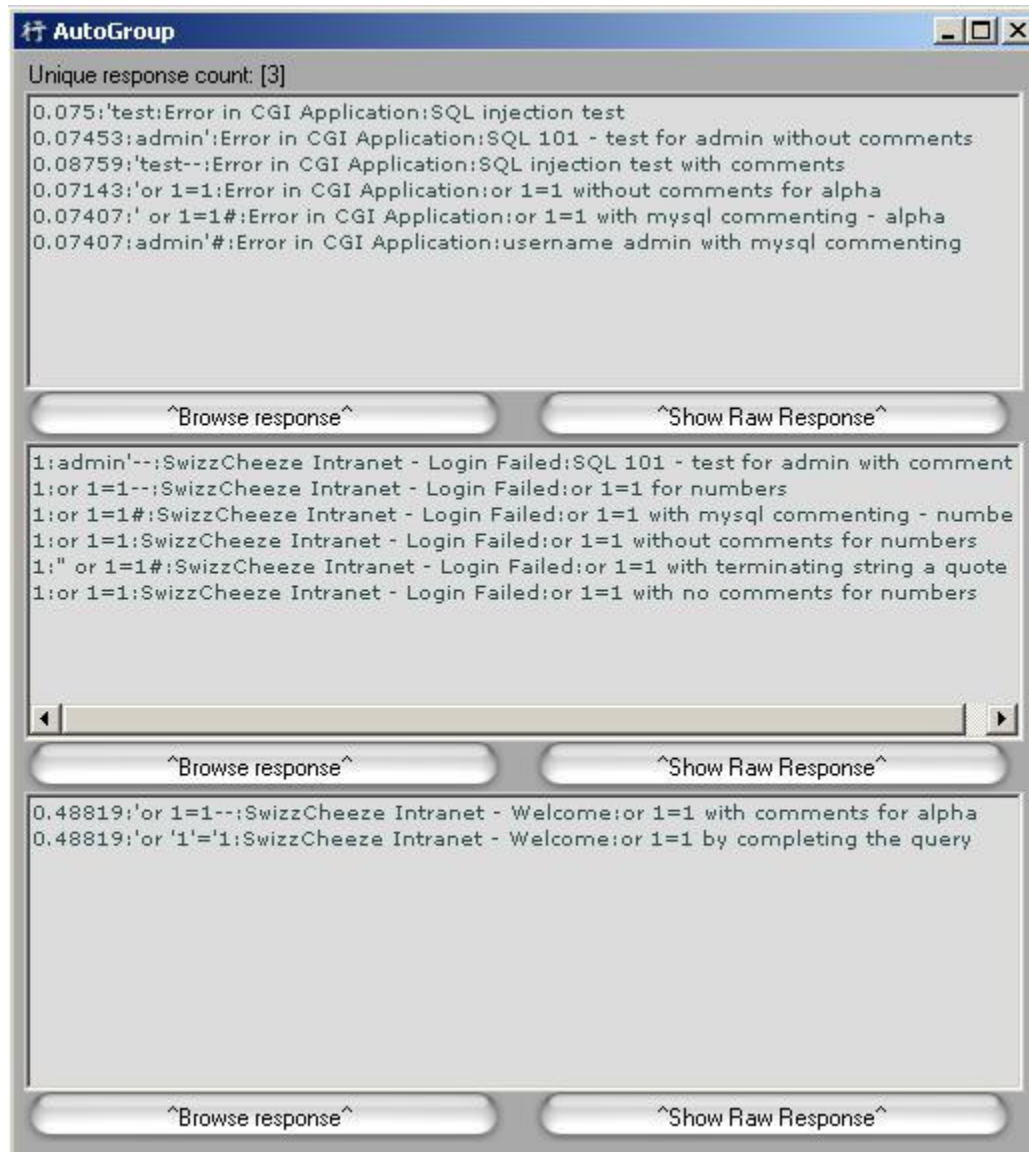
- We can either set the limit fields using the AI limit fields and conditions, but in this case we only want to view successful attempts.
- Using the right click menu, we select "Only these".



- This automatically sets the AI conditions. If the limit fields "!=" or "=" are selected Suru only uses the value in the lower limit field.



- Finally we can browse each reply or Auto Group the results by clicking "AutoGroup", refer to section 5.4 Auto Grouping and Tolerances to learn more about this.



5.3 The meaning in the Decoding the Result Field

Results in the Fuzzy Logic results window will look like this –

```
1.0432:password:10>Welcome to SwissCheeze:
```

The fields are delimited by a colon (:).

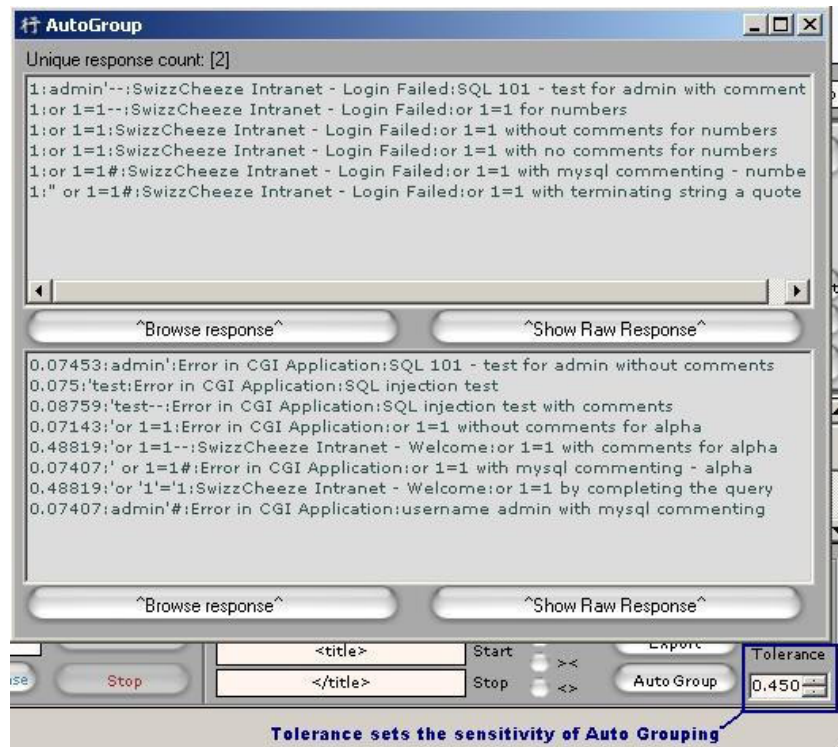
| | |
|--------------|--|
| First field | This field is the result generated by comparing the response with the base response (The base response is an arbitrary (false) request sent as the first fuzz request). This number is generated from the content received by the proxy. |
| Second field | This is the fuzz parameter used. |
| Third field | The index of the fuzz run. |
| Fourth field | Extract from the content extraction. |

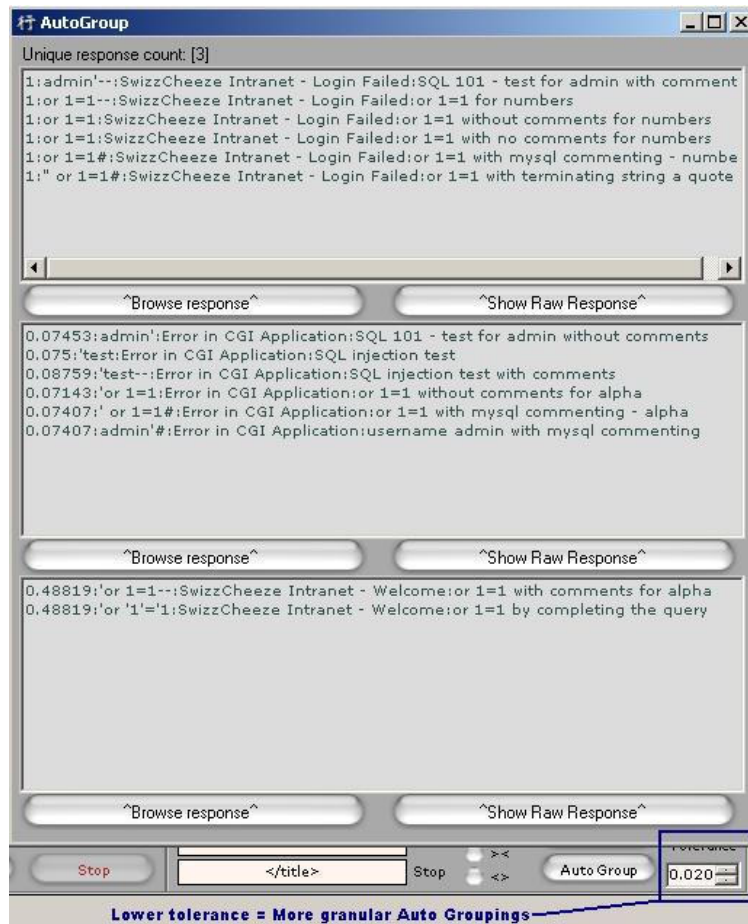
5.4 AutoGroup and Tolerance Settings

The “AutoGroup” button is used for organizing the results in an optimised manner. It enables a user to group similar results and the ability to browse the results. The tolerance setting in the main window adjusts the granularity of the groupings, the lower the tolerance the more granular the groupings (i.e. The less the response differs from the base response). The top of the AutoGroup window displays how many unique responses were discovered during a fuzz run.

AutoGrouping helps the user to quickly determine different results received from a web application. For example, when attempting a SQL injection fuzz runs, the web application may respond differently to fields submitted due to a difference in the HTTP status message returned or even a successful SQL injection login.

The figures below are examples of what can be achieved using the AutoGroup and Tolerance functions.





5.5 Recalculate

After adjusting the AI limits, the Recalculate button will update the results to match the new limits.

5.6 Export

The export button exports all the results to a text box, which can be copied and imported to a user defined file.

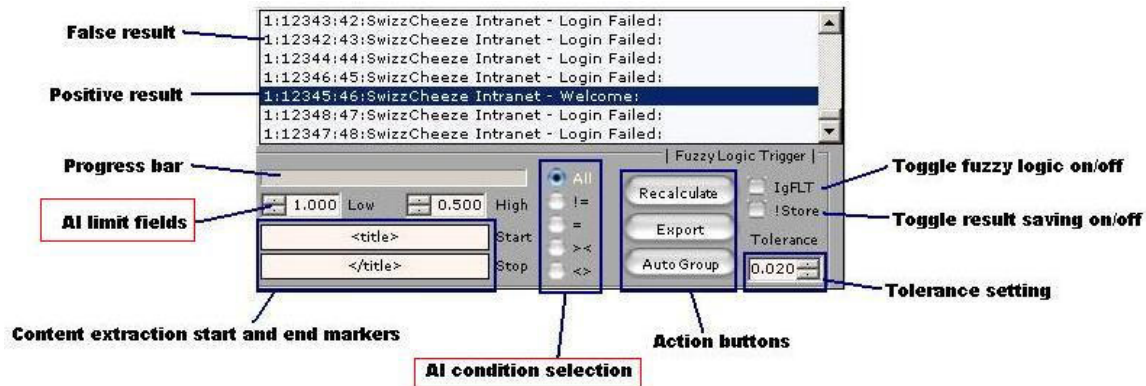
5.7 Note on Memory usage (The use of IgFLT and !Store)

Suru stores all the responses in memory for replay purposes. If a large fuzz file is used your system memory may be a potential issue.

- IgFLT is used for content extraction – This is used to speed up content extraction. With IgFLT checked, content comparison is disabled. E.g. mining data from the web application.
- !Store, if selected, will store all results in memory, otherwise the results will only be displayed in the Fuzzy Logic Results window and therefore replays will not work.

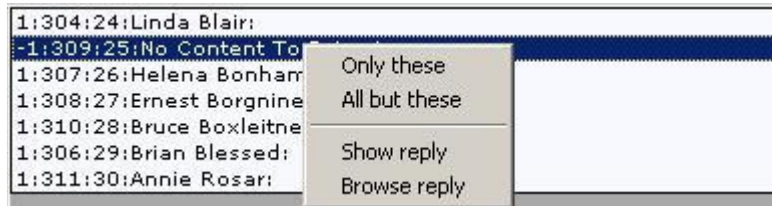
5.8 Fine tuning using AI conditions

The results of the Fuzz run can be optimised by using the low and high limits in conjunction with the AI condition statements:



The Fuzzy Logic Trigger control box contains low and high limit markers. Selecting the "><" radio button and clicking "Recalculate" will show you all the results that content compare index is between the low and high watermark. The "><" radio button corresponds with indexes outside of the range. The "=" (equals) and "!=" (not equal) buttons uses the first (low) value. Every time a new range is defined the "Recalculate" button will update the list.

Another (quicker) way to filter results is to use the context menu on the list box. Right -clicking on any entry in the list box will results in the follow menu:



"Only these" will automatically populate the low limit markers with the selected value, set the equal radio button and click on "Recalculate". "All but these" will do the inverse. You can also look at the response in HTML format by clicking on "Show reply", which will render the raw result in a text box:

```

Results
HTTP/1.1 200 OK
Date: Thu, 06 Jul 2006 09:30:09 GMT
Server: Apache
Cache-Control: private
Connection: close
Content-Type: text/html

<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<title>Ernest Borgnine</title>
<meta name="description" content="Ernest Borgnine - Filmography, Awards, Biography, Agent, Discussions, Photo
<meta name="keywords" content="movies,films,movie database,actors,actresses,directors,hollywood,stars,quotes
<link rel="stylesheet" type="text/css" href="http://i.imdb.com/imdb.css">
<link rel="stylesheet" type="text/css" href="http://i.imdb.com/sok.css">
<link rel="stylesheet" type="text/css" href="/images/css2/widgets.css">
<link rel="stylesheet" type="text/css" href="/images/css2/site/legacy.css">
<link rel="icon" href="http://i.imdb.com/favicon.ico">
</head>
<!-- h=imdb-online-1137.vdc.amazon.com i=2006-07-05 s=legacy(default) t='Thu Jul 6 02:30:09 2006' -->

<body bgcolor="#ffffff" text="#000000">

```

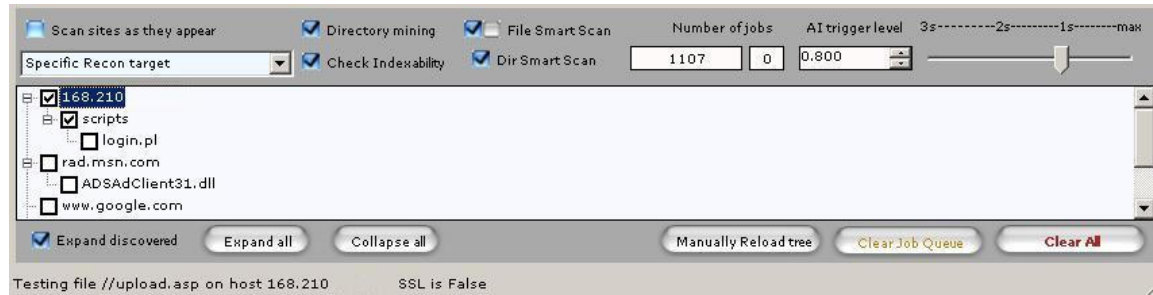
The "Browse reply" shows the same page, but renders the output in the replay browser.

At any time you might be interested in a "copy & paste"-friendly export of your results. To do this simply click on the "Export" button, this will show the filtered results in a textbox.

See Section 9 for an example on content extraction.

6 Recon

6.1 The interface

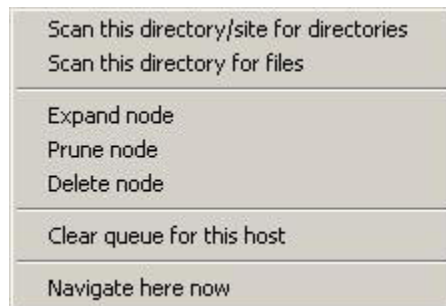


| | |
|--------------------------------------|--|
| Scan sites as they appear | Automatically begins scanning browsed sites for the scan options selected (directory mining, file scanning, indexability). |
| "Specific Recon Target" drop-down | Select the target to scan. If only a specific site must be scanned. |
| Recon Tree view | Tree view of the browsed sites. Structured hierarchically. Directories will have a checkbox associated. |
| Recon Tree view check boxes | Checkboxes used to mark which directories must be scanned for file scanning. |
| Expand discovered | Indicates whether the tree view must stay expanded and appends new findings to the tree. |
| Expand all | Expands the tree view. |
| Collapse all | Collapses the tree view to only list the target sites. |
| Directory mining | Select this if you require directory mining. Suru will mine for a common list of directories in the current directory. The list of the directories checked are located under the Config tab. |
| File smart scan (Checkbox 1 – left) | Checks for the existence of files with the same name, but different extensions across all directories. The list of file names and types are located under the Config tab. |
| File smart scan (Checkbox 2 – right) | Checks for the existence of files with the same name, but different extensions in the same directory. |
| Check Indexability | Tests if the target directories are indexable. |
| Dir Smart Scan | Used in conjunction with file and directory mining. Dynamically checks in directories already discovered. |
| Manually reload tree | After selecting the mining settings, manually reloads the tree to scan for the selected options. |
| Clear job queue | Jobs are created from the options selected (i.e. directory mining, indexability, etc.). This |

| | |
|-----------|--|
| | button clears the jobs in the job queue. |
| Clear all | Clears the entire tree. |

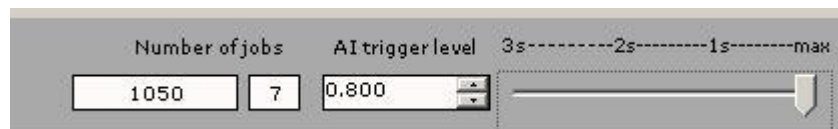
6.2 Right click menu

In the tree view there is an additional context-sensitive menu that can be activated by right-clicking on an entry in the tree view. Options can be toggled on/off for the selected entry. The user also has the option of browsing the finding in the replay browser.



6.3 Job queue

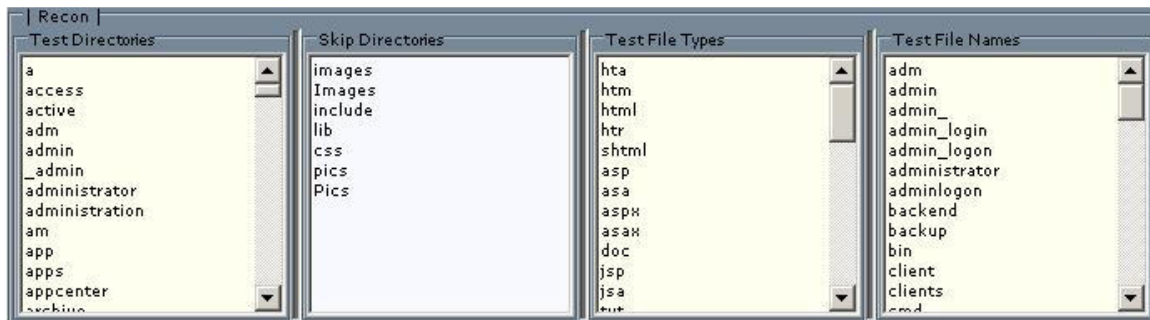
The job queue is created from the scan options. The left-hand box displays how many jobs are pending for the target. The middle box indicates the number of current jobs that are pending. By default Suru creates 6 active threads. The current number of jobs pending are the results still awaiting a response. By setting the delay on the slider on the left hand side the user can configure the delay between sending successive requests. The longer the delay the fewer requests will be sent for processing. The slider is typically used to optimize bandwidth usage, the slower your connection the higher the delay, the faster your connection the faster the requests can be submitted.



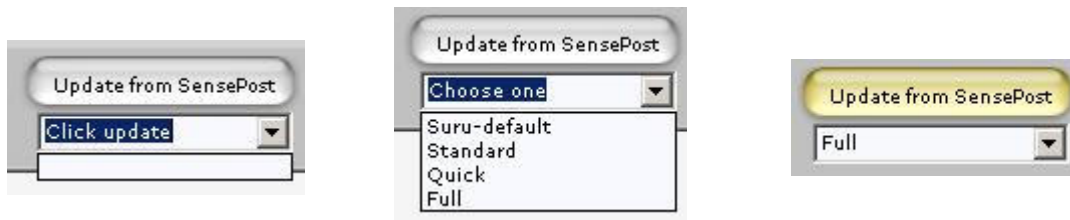
6.4 How-To

6.4.1 Updating directories and files

Recon uses lists of directories and file names and types. This can be manually updated by adding additional directories in each respective section under the Config tab.



Another method of updating the files is by downloading directly from SensePost. The button "Update from SensePost" will pull the latest lists from SensePost's site. After clicking the button, the drop-down list under the button will be populated with the various lists available. Select which list you require and click on "Update from SensePost", this will update each list with updated values.



6.4.2 Selecting a scan target

If the checkbox "Scan sites as they appear" is not selected, the user has the option of selecting a specific target to conduct scanning. From the drop-down list a target can be selected.



After the selection jobs for the respective target will be created and begin processing.



6.4.3 What do the results mean?

| | |
|-------------------------|------------------------------------|
| [Indexable] (In green) | The directory is indexable |
| file.type (In pink) | A new file/directory discovered |
| SSL file (In dark red) | A file discovered from an SSL site |
| SSL host (In dark blue) | Host browsed through SSL |

6.5 Using AI

The "AI trigger level" is used to set the sensitivity of the findings according to a base response. At the beginning of the test on a target, a base response is recorded. This is used to compare the results, i.e. the finding is the relation to how close the result differs from the base response.

E.g. At the beginning of the test Suru will send a request to the target site: /moo/blah.html, which is a non-existent page. The response is then passed through an algorithm where a number is calculated from the content received. For this example blah.html is given the value 1. The first test file is run, admin.html, let's say it exists in this case. The result will be put through the same algorithm and results in 0.3. The results are then compared numerically. 0.3 is significantly different from 1. If the AI trigger is set on 0.8, the result will show up on this tree view because anything lower than 0.8 is as a positive result.

The fuzzy logic number calculated from a response will always fall between 0 and 1, the closer the value to 1 (which will always be the base response) the more the result is similar to the base response. The lower the number the greater the response differs from the base response.

This approach can still be effective even if sites are responding with friendly error pages. By setting the AI trigger level lower will weed out any potential false positives.



Number of jobs: 1050
AI trigger level: 0.800
3s-----2s-----1s-----max

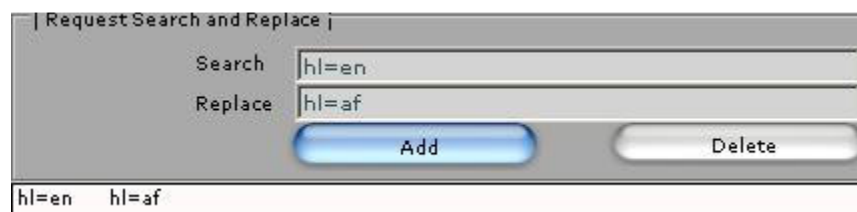
7 Misc

7.1 Search and replace

To explain this section, case studies will be used.

7.1.1 Request search and replace

- In the Request Search and Replace section we add the entry "hl=en" to be replaced by "hl=af".



Request Search and Replace

Search: hl=en
Replace: hl=af

Add Delete

hl=en hl=af

- We now browse to Google.
- From the tooltip below we see that the default search language is English (hl=en).



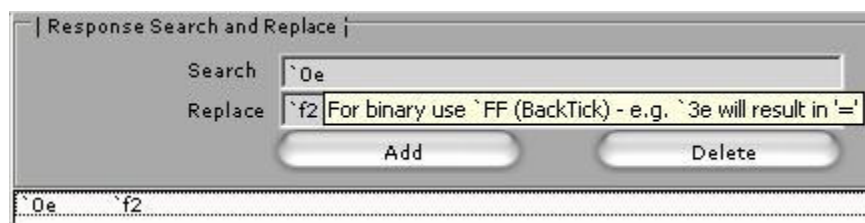
- And submit a search.
- The response received is now in Afrikaans (hl=en was changed to hl=af by Suru).



7.1.2 Response Search and Replace

In this example we are going to manipulate output of the Google picture, by replacing hex entries in the response.

The search and replace function has the ability to replace hex values in addition to the traditional ascii values. To edit hex values use begin with the back tick (') character.



We submit a request for the Google homepage and a manipulated image is rendered in the browser, show in the figure below.



7.1.3 Examples of Real-Life Uses:

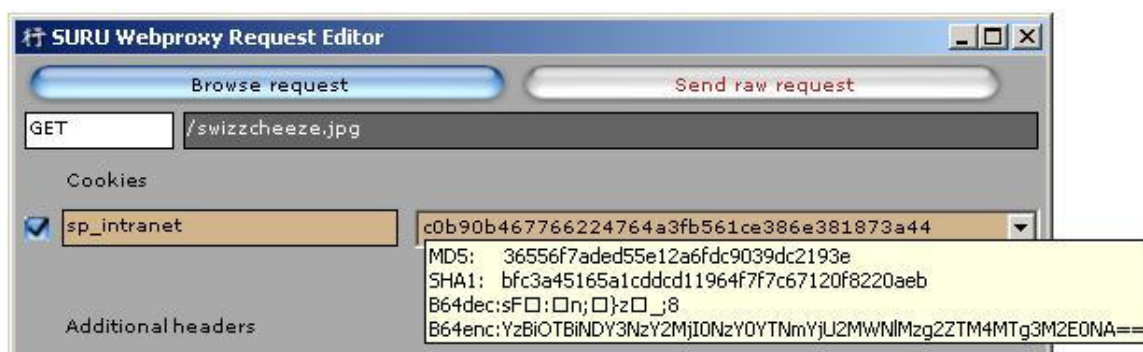
- If we want to track a specific pattern in the response received. E.g. highlighting for key words from a news article.
- Altering binary streams received in the response.
- Confusing users who are browsing through Suru.

7.2 Tools

The Tools section allows a user to convert text to common forms of encodings and hashes used in web applications.



If tooltips are enabled you can mouse over parameter values in the Request Edit window, which will automatically calculate the hashes/encodings of the value.



7.3 Auto Relationship Discovery

While browsing Suru will automatically encode and hash request parameters or parameters specified in the "Auto Relationship Discovery" section and compare the strings to content received in the response.

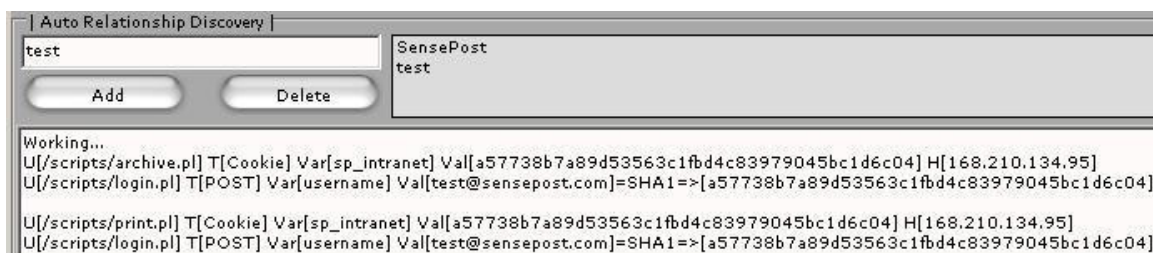
If any strings match they will show up in the results window in the "Auto Relationship Discovery" section.

For example:

- We log onto the site www.swisscheeze.com with user: test@sensepost.com and begin browsing around.

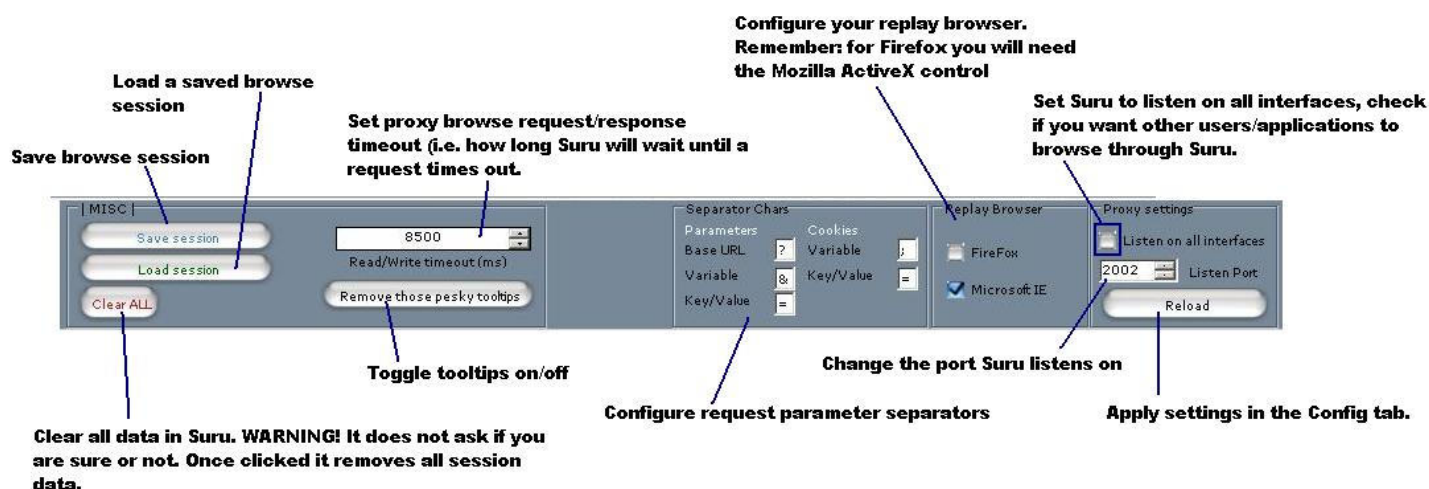


- During our browse session the Auto Relationship feature works out that the session cookie is a SHA1 hash of the username. This helps the user to save time converting and manually verifying all the parameters in a request/response.



8 Configuration

Most of the configuration features have already been explained. All that is left is to explain the sections at the bottom of the Config tab.



8.1 Saved sessions

When saving sessions in Suru two save files are stored.

1. Suru.txt – The saved requests and responses from the proxy tab
2. Suru.txt.discovered – The information discovered through recon runs.

8.2 Setting up Mozilla/Firefox as your request/replay browser

You will require the Mozilla ActiveX control. The control can be located at:
<http://umn.dl.sourceforge.net/sourceforge/wine/MozillaControl177.exe?download>

9 Content extraction example

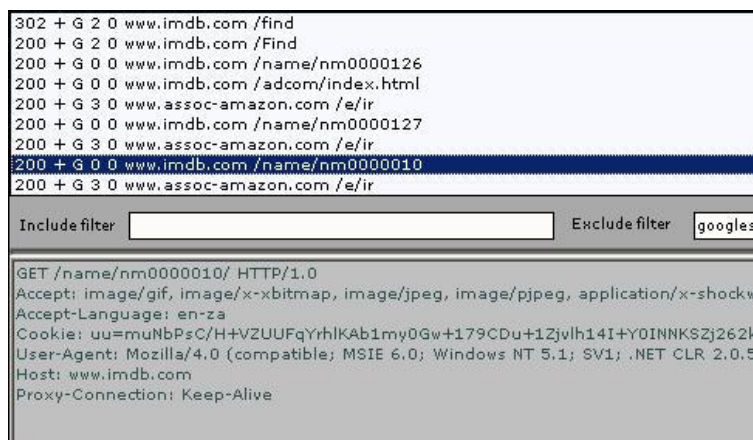
Objective: Extracting Actors names from a given actor code in www.imdb.com

Each actor is assigned a code that links to their biography. In this example we will extract the Actor associated with the code. Within the actors biography are other demographics which could also be mined, all that is required to define the extraction parameters.

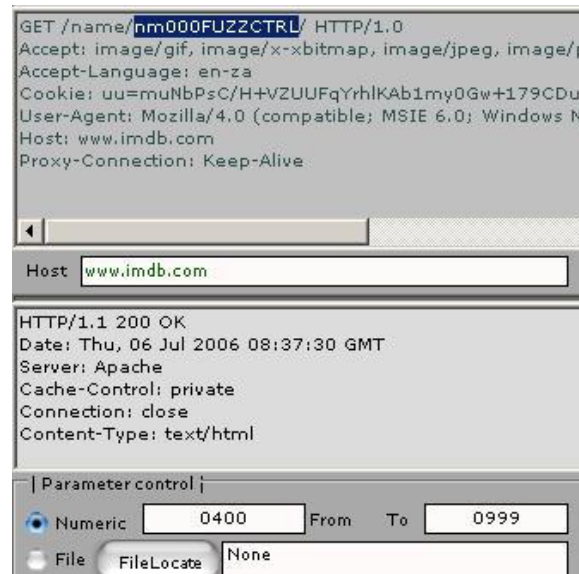
- Firstly browse to www.imdb.com and search for an actors biography in the search function.
- We then we select an actor, so that the request shows up in Suru.



- The request is highlighted in the browse window.



- Add the "FUZZCTRL" point.
- We also select the "Numeric" radio button and set the limits we require so we can iterate through numbers.



- The content extraction beginning and end points are set. We find this by viewing the HTML source and determine what parameters we want to match. In this case the conditions are:
 - `<h1><strong class="title">` for the start point &
 - `</h1></p>` for the end point

```
<table border="0" cellpadding="0" cellspacing="0" width="100%">
  <tr valign="top">
    <td width="90%">
      <br>
      <p><h1><strong class="title">James Cagney</strong></h1></p>
</td>
</tr>
</table>
```

- We can optionally check the “IgFLT” checkbox if we do not want to perform fuzzy logic checking and only wish to perform content extraction.
- Click “Start” and view the results stream in into the Fuzzy Logic Results window.



- When the run is complete, click "Export" to view the results in a text box.
- The -1 results are also interesting, they indicate that an HTTP status page was received, in this case -1 is (404 Page not found) or the timeout or retry count was exceeded.

| Exported Results |
|----------------------------------|
| 1:283:0:Brenda Bakke: |
| 1:280:1:Catherine Bach: |
| 1:282:2:Scott Bairstow: |
| 1:281:3:Scott Baio: |
| 1:284:4:Adam Baldwin: |
| -1:288:5:No Content To Extract: |
| 1:287:6:William Baldwin: |
| 1:285:7:Alec Baldwin: |
| 1:286:8:Stephen Baldwin (I): |
| 1:289:9:Ellen Barkin: |
| 1:292:10:Michelle Bauer: |
| 1:291:11:Angela Bassett: |
| 1:293:12:Sean Bean: |
| 1:290:13:John Barry (I): |
| 1:294:14:Amanda Bearse: |
| 1:295:15:Kate Beckinsale: |
| 1:296:16:Robert Beltran: |
| 1:297:17:Tom Berenger: |
| 1:299:18:Michael Biehn: |
| 1:298:19:Candice Bergen: |
| 1:301:20:Thora Birch: |
| 1:300:21:Juliette Binoche: |
| 1:302:22:Jacqueline Bisset: |
| 1:303:23:Honor Blackman: |
| 1:304:24:Linda Blair: |
| -1:309:25:No Content To Extract: |
| 1:307:26:Helena Bonham Carter: |
| 1:308:27:Ernest Borgnine: |
| 1:310:28:Bruce Boxleitner: |
| 1:306:29:Brian Blessed: |
| 1:311:30:Annie Rosar: |
| 1:313:31:Jeff Bridges (I): |
| 1:305:32:Mel Blanc: |
| 1:315:33:Louise Brooks (I): |
| 1:312:34:Amy Brenneman (I): |