



# WATOBO

Web Application Toolbox

## TUTORIAL

by Fancy

Email: [fancy@corelan.be](mailto:fancy@corelan.be)  
Web: <http://www.corelan.be:8800>

### Table of contents

1. Introduction .....	2
2. Installation .....	4
2.1 Installation under Windows .....	4
2.2 Installation under BackTrack .....	5
3. Using WATOBO .....	7
3.1 Start WATOBO: .....	7
3.2 Passive checks .....	9
3.3 Active checks .....	10
3.3 Session management .....	18
3.4 Manual requests .....	24
Here you can change what you like in the request (e.g. id=' ') and send it away .....	24
See the comparison of the REQUEST .....	25
3.5 More functions .....	27
3.6 Fuzzing .....	31
- Enumerate Usernames - .....	31
- Fuzzing multiple values - .....	35
- Generating complex values - .....	40
4. Conclusion .....	45
5. References .....	45

# 1. Introduction

WATOBO [1] is intended to enable security professionals to perform highly efficient (semi-automated) web application security audits. I am convinced that the semi-automated approach is the best way to perform an accurate audit and to identify most of the vulnerabilities.

WATOBO has no attack capabilities and is provided for legal vulnerability audit purposes only. It works like a local proxy, similar to WebScarab, Paros or BurpSuite. Additionally, WATOBO supports passive and active checks. Passive checks are more like filter functions. They are used to collect useful information, e.g. email or IP addresses. Passive checks will be performed during normal browsing activities. No additional requests are sent to the (web) application.

Active checks instead will produce a high number of requests (depending on the check module) because they do the automatic part of vulnerability identification, e.g. during a scan.

The most important advantages of WATOBO are:

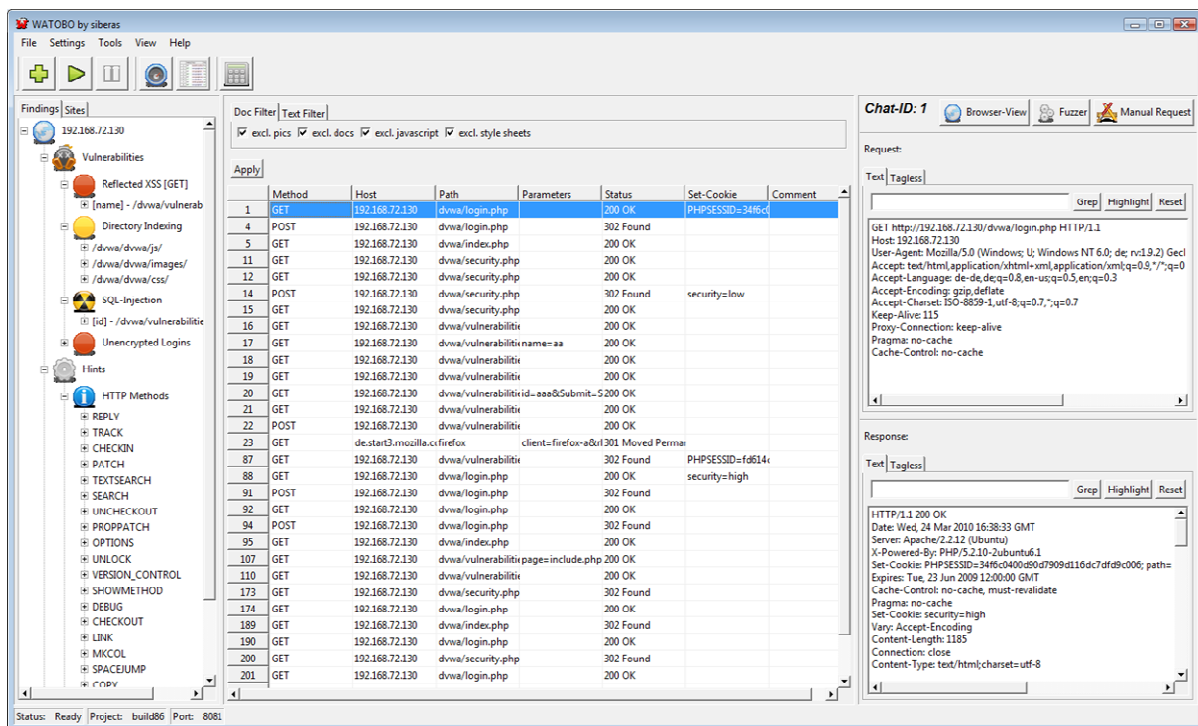
- WATOBO has Session Management capabilities! You can define login scripts as well as logout signatures. So you don't have to login manually each time you get logged out
- WATOBO can perform vulnerability checks out of the box
- WATOBO supports Inline De-/Encoding, so you don't have to copy strings to a transcoder and back again. Just do it inside the request/response window with a simple mouse click.
- WATOBO has smart filter functions, so you can find and navigate to the most interesting parts of the application easily.
- WATOBO is written in (FX)Ruby and enables you to define your own checks
- WATOBO is free software (licensed under the GNU General Public License Version 2)

## Summarizing the functions of WATOBO:

- Supports session management.
- Detects logout and automatically takes a re-login.
- Supports filter functions
- Inline-Encoder/Decoder
- Includes [vulnerability scanner](#)
- Quick-scan for targeted scanning a URL
- Full-scan to scan a whole session

- Manual request editor with special functions
- Session information is updated
- Login can be done automatically
- Transcoder
- URL, Base64, MD5, SHA-1
- Interceptor
- Fuzzer
- Free, Stable and [Open source!](#)
- Script code easy to understand
- Easy to extend / adapt
- In real-world scenarios tested and developed
- Speed / usability
- Active and Passive checks
- Runs under Windows, Linux, BackTrack.

### Screenshot:



The screenshot displays the WAYOBO Web Application Toolbox interface. On the left, a tree view shows various vulnerability categories such as Reflected XSS (GET), Directory Indexing, SQL-Injection, and Unencrypted Logins. The main panel shows a list of detected vulnerabilities with columns for Method, Host, Path, Parameters, Status, Set-Cookie, and Comment. A detailed view of a request and response is shown on the right, including headers like User-Agent, Accept, and Set-Cookie, and the response body showing an HTTP 200 OK status and various headers.

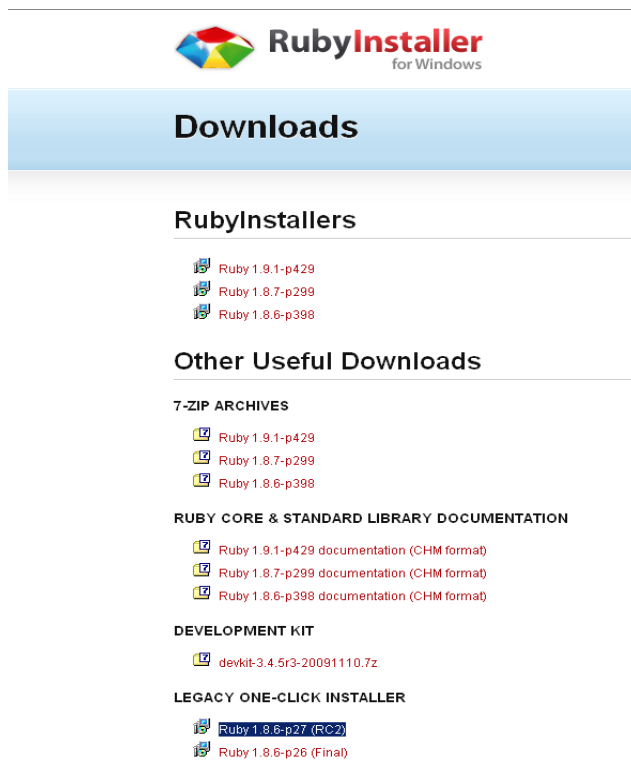
Method	Host	Path	Parameters	Status	Set-Cookie	Comment
1	192.168.72.130	dvwa/login.php		200 OK	PHPSESSID=34f6cd	
4	192.168.72.130	dvwa/login.php		302 Found		
5	192.168.72.130	dvwa/index.php		200 OK		
11	192.168.72.130	dvwa/security.php		200 OK		
12	192.168.72.130	dvwa/security.php		200 OK		
14	192.168.72.130	dvwa/security.php		302 Found	security=low	
15	192.168.72.130	dvwa/security.php		200 OK		
16	192.168.72.130	dvwa/vulnerabilit		200 OK		
17	192.168.72.130	dvwa/vulnerabilit	name=aa	200 OK		
18	192.168.72.130	dvwa/vulnerabilit		200 OK		
19	192.168.72.130	dvwa/vulnerabilit		200 OK		
20	192.168.72.130	dvwa/vulnerabilit	id=aaa&Submit=5	200 OK		
21	192.168.72.130	dvwa/vulnerabilit		200 OK		
22	192.168.72.130	dvwa/vulnerabilit		200 OK		
23	des.start3.mozilla.cu.firefox	client=firefox-a&bl	301 Moved Permanently			
87	192.168.72.130	dvwa/vulnerabilit		302 Found	PHPSESSID=fd614	
88	192.168.72.130	dvwa/login.php		200 OK	security=high	
91	192.168.72.130	dvwa/login.php		302 Found		
92	192.168.72.130	dvwa/login.php		200 OK		
94	192.168.72.130	dvwa/login.php		302 Found		
95	192.168.72.130	dvwa/index.php		200 OK		
107	192.168.72.130	dvwa/vulnerabilit	pages=include.php	200 OK		
110	192.168.72.130	dvwa/vulnerabilit		200 OK		
173	192.168.72.130	dvwa/security.php		302 Found		
174	192.168.72.130	dvwa/login.php		200 OK		
189	192.168.72.130	dvwa/index.php		302 Found		
190	192.168.72.130	dvwa/login.php		200 OK		
200	192.168.72.130	dvwa/security.php		302 Found		
201	192.168.72.130	dvwa/login.php		200 OK		

## 2. Installation

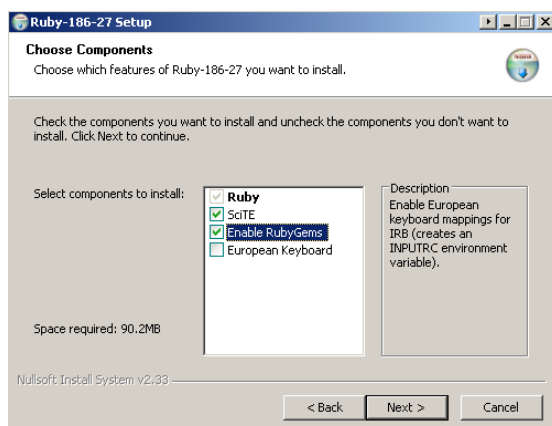
### 2.1 Installation under Windows

#### 1. Install Ruby:

Download the Ruby One-click Installer (<http://rubyinstaller.org/downloads/>) and install ruby on your computer:



The screenshot shows the RubyInstaller for Windows website. At the top, there is a logo for RubyInstaller for Windows. Below the logo, the word "Downloads" is prominently displayed. Underneath, there are sections for "RubyInstallers" and "Other Useful Downloads". The "RubyInstallers" section lists three versions: Ruby 1.9.1-p429, Ruby 1.8.7-p299, and Ruby 1.8.6-p398. The "Other Useful Downloads" section is divided into several categories: "7-ZIP ARCHIVES" (listing the same three Ruby versions), "RUBY CORE & STANDARD LIBRARY DOCUMENTATION" (listing documentation for the same three versions in CHM format), "DEVELOPMENT KIT" (listing devkit-3.4.5r3-20091110.7z), and "LEGACY ONE-CLICK INSTALLER" (listing Ruby 1.8.6-p27 (RC2) and Ruby 1.8.6-p26 (Final)).



The screenshot shows the "Ruby-186-27 Setup" window. The title bar reads "Ruby-186-27 Setup". The main window has a "Choose Components" section with the instruction "Choose which features of Ruby-186-27 you want to install." Below this, there is a list of components to install, each with a checkbox and a description. The components are: Ruby (checked), SciTE (checked), Enable RubyGems (checked), and European Keyboard (unchecked). The description for "Enable RubyGems" is "Enable European keyboard mappings for IRB (creates an INPTRC environment variable)". At the bottom of the window, there is a "Space required: 90.2MB" label and a "Nullsoft Install System v2.33" label. The window also has "< Back", "Next >", and "Cancel" buttons.

## 2. Get WATOBO:

Download WATOBO (<http://sourceforge.net/projects/watobo/>) and extract the WATOBO sources to a place/directory of your choice

### ***2.2 Installation under BackTrack***

#### **1. Update your backtrack installation (this step is optional but always recommended)**

```
apt-get update  
apt-get upgrade
```

#### **2.) Install fxruby**

Execute the following commands:

```
gem uninstall rubygems-update
```

(ignore message "Unknown gem rubygems-update >= 0")

```
gem install rubygems-update -v 1.3.4  
/var/lib/gems/1.8/bin/update_rubygems  
gem install hoe  
gem install fxruby
```

#### **3. Install JSSH Firefox Extension**

Follow the instructions of the firewatir projekt:

<http://wiki.openqa.org/display/WTR/FireWatir+Installation>

Click on "Install" and then on "Allow":

Firefox prevented this site (wiki.openqa.org) from asking you to install software on your computer.

Allow

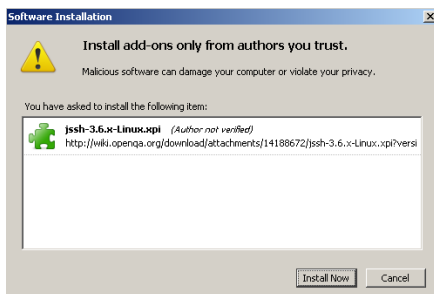
## 2) Install the JSSH Firefox Extension

The correct extension depends on your platform and your version of Firefox.

### Production FireWatir

Platform	Firefox 2.0	Firefox 3.0	Firefox 3.5	Firefox 3.6
Windows	<a href="#">install</a>	<a href="#">install</a>	<a href="#">install</a>	<a href="#">install</a>
Mac	<a href="#">install</a>	<a href="#">install</a>	<a href="#">install</a>	<a href="#">install</a>
Linux	<a href="#">install</a>	<a href="#">install</a>	<a href="#">install</a>	<a href="#">install</a>

Click on “Install now” in the following dialog box:



Then restart firefox.

## 4. Get WATOBO:

Download WATOBO (<http://sourceforge.net/projects/watobo/>) and extract the WATOBO sources to a place/directory of your choice

## 3. Using WATOBO

### 3.1 Start WATOBO:

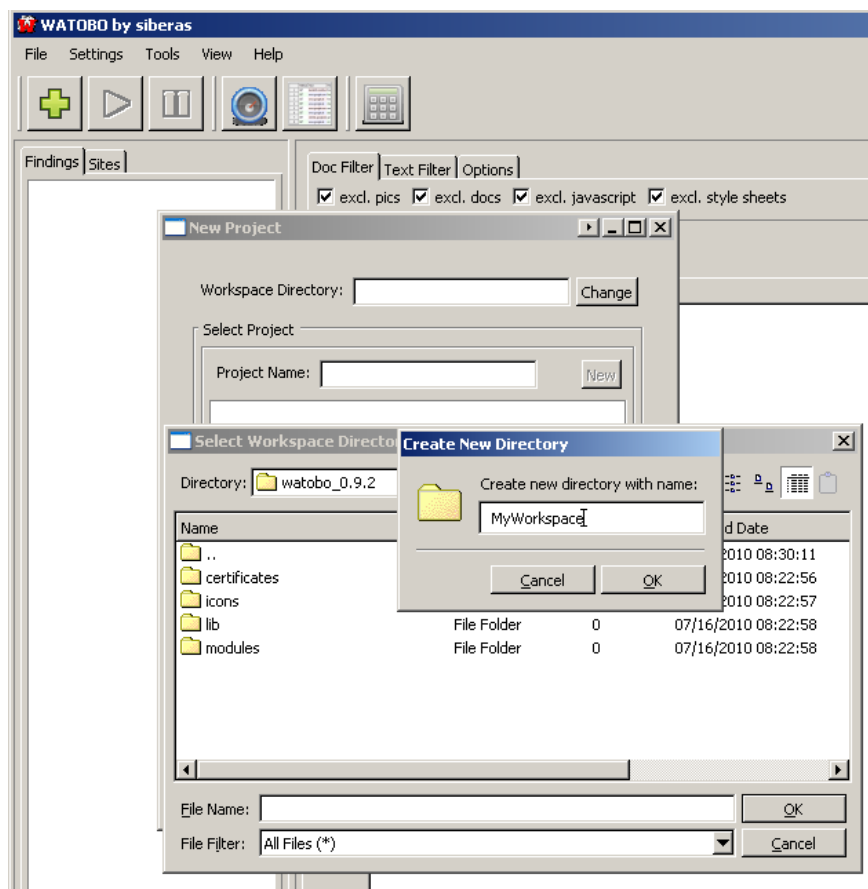
→ cd into the WATOBO directory and then issue the following command:

```
ruby start_watobo.rb
```

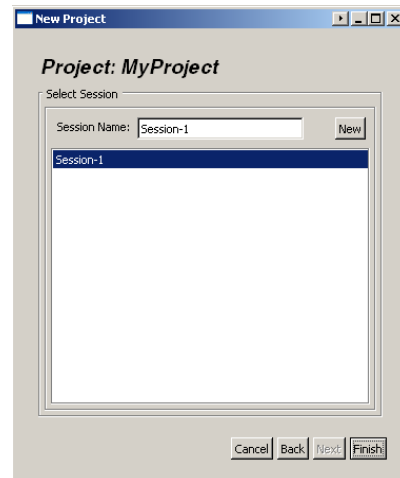
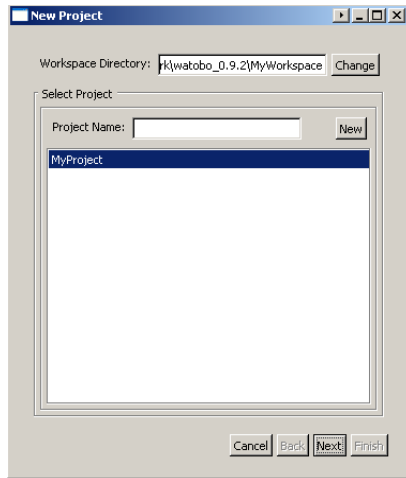
→ Click on the green



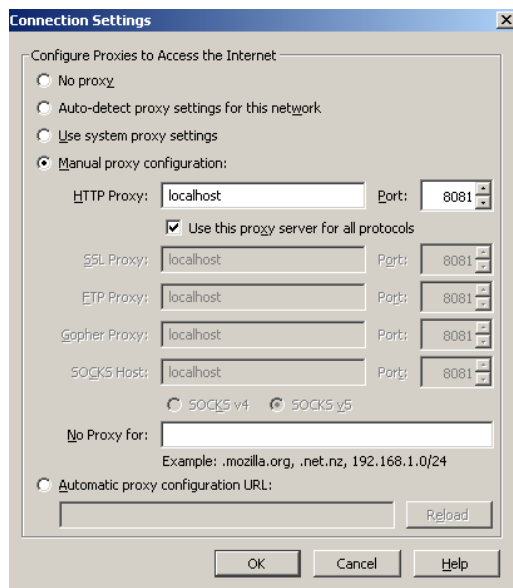
and create/select your workspace directory:



→ enter project name and session name:



→ change the proxy-settings of your preferred browser, e.g. firefox:



**Now you are ready to go!**



## 3.2 Passive checks

→ visit the target application (all the parts you want to audit):

Example: Mutillidae [1]:



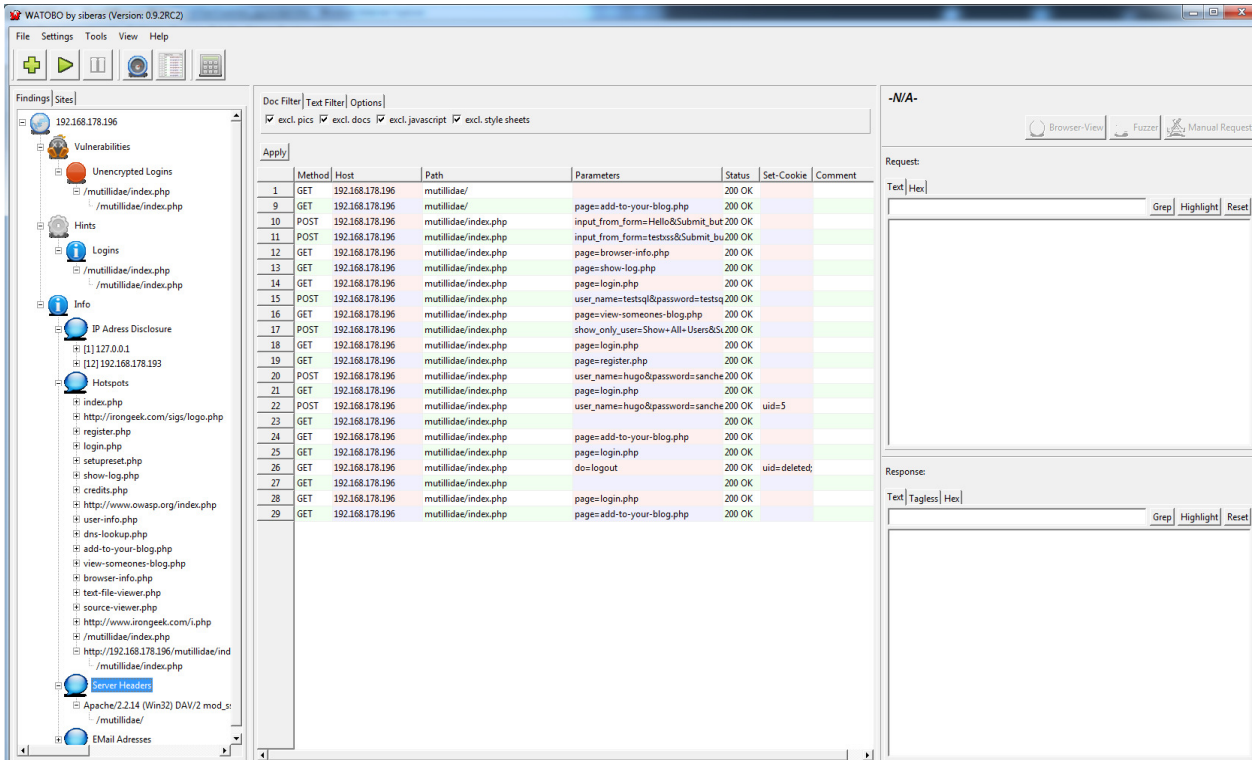
The screenshot shows the Mutillidae application interface. At the top, there is a green header with a red and black beetle logo on the left, the title "Mutillidae: Hack, Learn, Secure" in the center, and "Not logged in" on the right. Below the header, the version "Version 1.5" is displayed. A left sidebar contains navigation links: "Core Controls" (Home, Register, Login, Logout, Toggle hints, Setup/reset the DB, Show log, Credits), "OWASP Top 10 2010", "A1 - Injection (SQL and Command)" (Login, User info, DNS Lookup), and "A2 - Cross Site Scripting (XSS)". The main content area is titled "Login" and contains the text "If you do not have an account, [Register](#)". Below this is the instruction "Enter your username and password:" followed by a "Name:" label and a text input field containing "testsql". A "Password:" label is followed by a password input field with five dots. A "Submit" button is located below the password field.



The screenshot shows the Mutillidae application interface for adding a blog entry. The header is identical to the previous screenshot. The left sidebar is also identical. The main content area is titled "Add to your blog" and contains the text "Welcome to your blog, leave an entry. Login, or you will be listed as 'anonymous':". Below this is a large text input field containing "testxss". A "Submit" button is located below the input field. Underneath the button, the text "Entries:" is followed by a list of three entries:

- anonymous:(2010-07-20 18:16:43)  
testxss
- anonymous:(2010-07-20 18:16:37)  
Hello
- anonymous:(2009-03-01 22:27:11)  
An anonymous blog? Huh?

→ when finished with browsing switch back to WATOBO and look at the first results of the passive checks:



Apply	Method	Host	Path	Parameters	Status	Set-Cookie	Comment
1	GET	192.168.178.196	mutillidae/		200 OK		
9	GET	192.168.178.196	mutillidae/	pages-add-to-your-blog.php	200 OK		
10	POST	192.168.178.196	mutillidae/index.php	input_from_form=Hello&Submit_but	200 OK		
11	POST	192.168.178.196	mutillidae/index.php	input_from_form=testss&Submit_bu	200 OK		
12	GET	192.168.178.196	mutillidae/index.php	pages-browser-info.php	200 OK		
13	GET	192.168.178.196	mutillidae/index.php	pages-show-log.php	200 OK		
14	GET	192.168.178.196	mutillidae/index.php	pages-login.php	200 OK		
15	POST	192.168.178.196	mutillidae/index.php	user_name=testsql&password=testsq	200 OK		
16	GET	192.168.178.196	mutillidae/index.php	page-view-someones-blog.php	200 OK		
17	POST	192.168.178.196	mutillidae/index.php	show_only_user=Show+All+Users&S	200 OK		
18	GET	192.168.178.196	mutillidae/index.php	pages-login.php	200 OK		
19	GET	192.168.178.196	mutillidae/index.php	page-register.php	200 OK		
20	POST	192.168.178.196	mutillidae/index.php	user_name=hugo&password=sanche	200 OK		
21	GET	192.168.178.196	mutillidae/index.php	page-login.php	200 OK		
22	POST	192.168.178.196	mutillidae/index.php	user_name=hugo&password=sanche	200 OK	uid=5	
23	GET	192.168.178.196	mutillidae/index.php		200 OK		
24	GET	192.168.178.196	mutillidae/index.php	pages-add-to-your-blog.php	200 OK		
25	GET	192.168.178.196	mutillidae/index.php	pages-login.php	200 OK		
26	GET	192.168.178.196	mutillidae/index.php	do=logout	200 OK	uid=deleted	
27	GET	192.168.178.196	mutillidae/index.php		200 OK		
28	GET	192.168.178.196	mutillidae/index.php	pages-login.php	200 OK		
29	GET	192.168.178.196	mutillidae/index.php	pages-add-to-your-blog.php	200 OK		

### 3.3 Active checks

A full scan will perform an automated vulnerability analysis of all recorded chats (except the excluded ones).

First you have to exclude the chats from scanning which:

- **may harm our system/application**
- **may lock our login**
- **will logout our sessions**
- **we don't want to analyze**

→ because we don't want to lock our account we exclude the login chat (15):

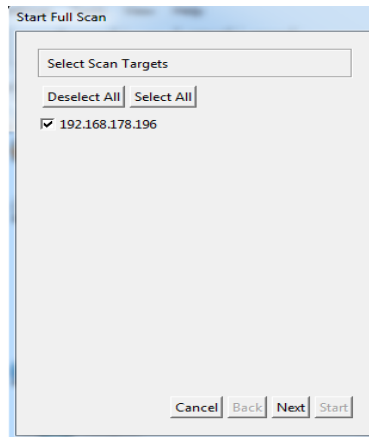
Method	Host	Path	Parameters	Status	Set-Cookie	Comment
1	GET	192.168.178.196	mutillidae/index.php	200	OK	
2	GET	192.168.178.196	mutillidae/index.php	pages-login.php	200	OK
3	POST	192.168.178.196	mutillidae/index.php	user_name=test&password=200	OK	
4	GET	192.168.178.196	mutillidae/index.php	pages-user-info.php	200	OK
5	POST	192.168.178.196	mutillidae/index.php	view_user_name=test&password=200	OK	
6	GET	192.168.178.196	mutillidae/index.php	pages-dns-lookup.php	200	OK
7	POST	192.168.178.196	mutillidae/index.php	target_host=test&submit=200	OK	
8	POST	192.168.178.196	mutillidae/index.php	target_host=sd-%26%26-%26	OK	
9	POST	192.168.178.196	mutillidae/index.php	pages-add-to-your-blog.php	200	OK
10	GET	192.168.178.196	mutillidae/index.php	input_from_form=test&submit=200	OK	
11	POST	192.168.178.196	mutillidae/index.php	pages-view-someones-blog	200	OK
12	GET	192.168.178.196	mutillidae/index.php	pages-view-someones-blog	200	OK
13	POST	192.168.178.196	mutillidae/index.php	show_only_users>Show+All+200	OK	
14	GET	192.168.178.196	mutillidae/index.php	pages-login.php	200	OK
15	POST	192.168.178.196	mutillidae/index.php	user_name=john&password=200	OK	uid=3
16	GET	192.168.178.196	mutillidae/index.php	mutillidae	200	OK
17	GET	192.168.178.196	mutillidae/index.php	mutillidae	200	OK
18	POST	192.168.178.196	mutillidae/index.php	input_from_form=test&submit=200	OK	
19	GET	192.168.178.196	mutillidae/index.php	pages-view-someones-blog	200	OK
20	POST	192.168.178.196	mutillidae/index.php	show_only_users=admin&submit=200	OK	
21	POST	192.168.178.196	mutillidae/index.php	show_only_users>Show+All+200	OK	
22	GET	192.168.178.196	mutillidae/index.php	pages-browser-info.php	200	OK
23	GET	192.168.178.196	mutillidae/index.php	pages-show-log.php	200	OK
24	GET	192.168.178.196	mutillidae/index.php	pages-text-file-viewer.php	200	OK
25	POST	192.168.178.196	mutillidae/index.php	text_file_name=http%3A%2F%2Fwww.example.com	200	OK
26	GET	192.168.178.196	mutillidae/index.php	pages-source-viewer.php	200	OK
27	GET	192.168.178.196	mutillidae/index.php	pages-source-viewer.php	200	OK
28	GET	192.168.178.196	mutillidae/index.php	pages-source-viewer.php	200	OK
29	GET	192.168.178.196	mutillidae/index.php	pages-show-log.php	200	OK
30	GET	192.168.178.196	mutillidae/index.php	pages-credits.php	200	OK
31	GET	192.168.178.196	mutillidae/index.php	pages-credits.php	200	OK
32	GET	192.168.178.196	mutillidae/index.php	pages-credits.php	200	OK
33	GET	192.168.178.196	mutillidae/index.php	pages-credits.php	200	OK
34	GET	192.168.178.196	mutillidae/index.php	do=logout	200	OK uid=deleted;
35	GET	192.168.178.196	mutillidae/index.php	mutillidae	200	OK
36	GET	192.168.178.196	mutillidae/index.php	mutillidae	200	OK

and exclude the logout chat (34):

→ Start scan:



→ select target(s):



Start Full Scan

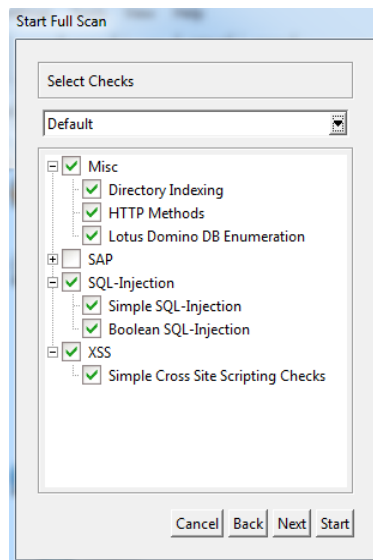
Select Scan Targets

Deselect All | Select All

192.168.178.196

Cancel | Back | Next | Start

→ select checks:



Start Full Scan

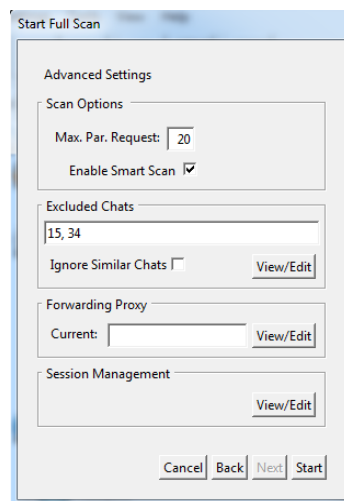
Select Checks

Default

- Misc
  - Directory Indexing
  - HTTP Methods
  - Lotus Domino DB Enumeration
- SAP
- SQL-Injection
  - Simple SQL-Injection
  - Boolean SQL-Injection
- XSS
  - Simple Cross Site Scripting Checks

Cancel | Back | Next | Start

→ verify excluded Chats:



Start Full Scan

Advanced Settings

Scan Options

Max. Par. Request: 20

Enable Smart Scan

Excluded Chats

15, 34

Ignore Similar Chats  View/Edit

Forwarding Proxy

Current: View/Edit

Session Management

View/Edit

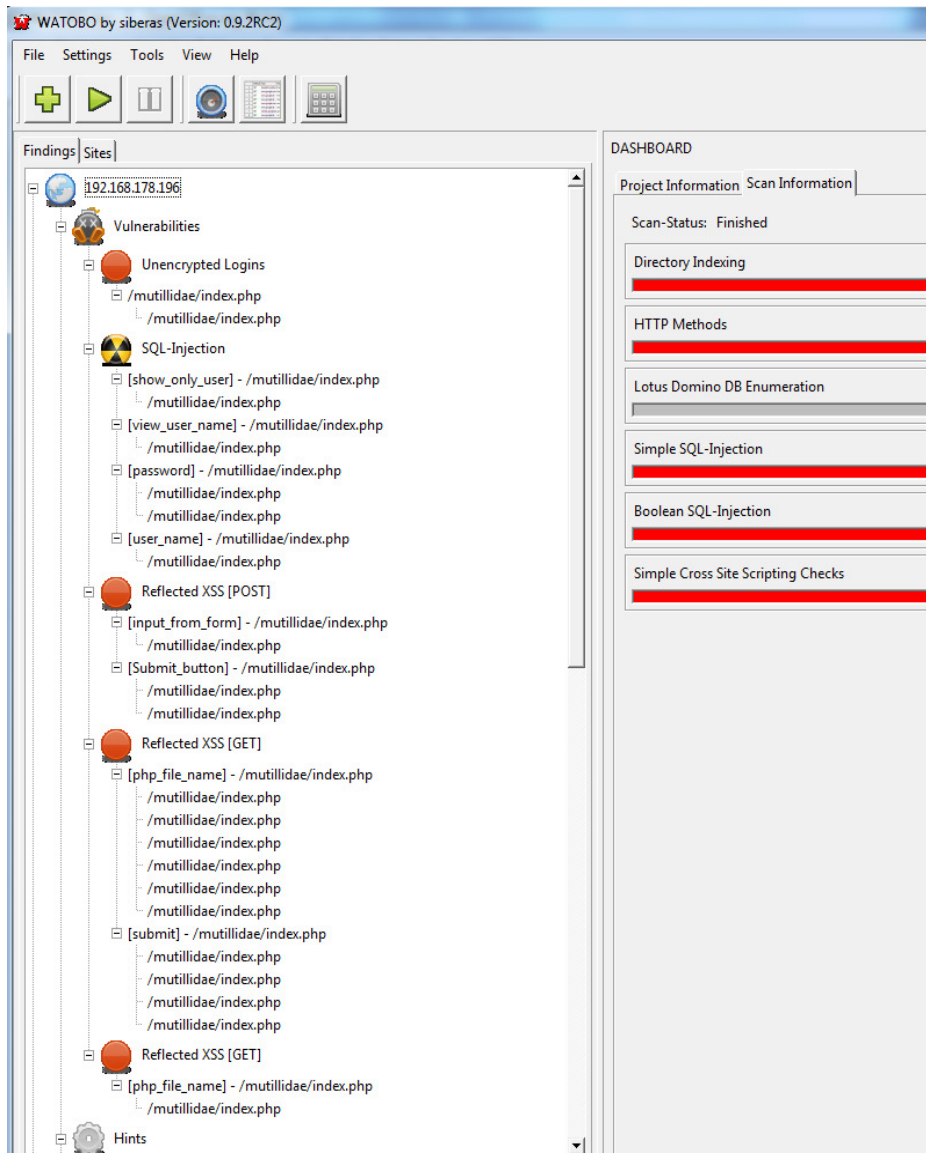
Cancel | Back | Next | Start

→ press **Start** → the findings are updated immediately:

You can watch the scan progress with the dashboard:



Findings:



The screenshot displays the WATOBO (Web Application Toolbox) interface. The main window is titled "WATOBO by siberas (Version: 0.9.2RC2)" and features a menu bar with "File", "Settings", "Tools", "View", and "Help". Below the menu is a toolbar with icons for adding, running, pausing, and refreshing scans. The interface is split into two main sections: "Findings" and "Sites".

The "Findings" section shows a tree view of vulnerabilities for the target IP 192.168.178.196. The vulnerabilities are categorized as follows:

- Vulnerabilities**
  - Unencrypted Logins**
    - /mutillidae/index.php
    - /mutillidae/index.php
  - SQL-Injection**
    - [show\_only\_user] - /mutillidae/index.php
    - /mutillidae/index.php
    - [view\_user\_name] - /mutillidae/index.php
    - /mutillidae/index.php
    - [password] - /mutillidae/index.php
    - /mutillidae/index.php
    - /mutillidae/index.php
    - [user\_name] - /mutillidae/index.php
    - /mutillidae/index.php
  - Reflected XSS [POST]**
    - [input\_from\_form] - /mutillidae/index.php
    - /mutillidae/index.php
    - [Submit\_button] - /mutillidae/index.php
    - /mutillidae/index.php
    - /mutillidae/index.php
  - Reflected XSS [GET]**
    - [php\_file\_name] - /mutillidae/index.php
    - /mutillidae/index.php
    - /mutillidae/index.php
    - /mutillidae/index.php
    - /mutillidae/index.php
    - /mutillidae/index.php
    - /mutillidae/index.php
    - [submit] - /mutillidae/index.php
    - /mutillidae/index.php
    - /mutillidae/index.php
    - /mutillidae/index.php
    - /mutillidae/index.php
  - Reflected XSS [GET]**
    - [php\_file\_name] - /mutillidae/index.php
    - /mutillidae/index.php
- Hints**

The "Sites" section is currently empty. On the right side, there is a "DASHBOARD" with two tabs: "Project Information" and "Scan Information". The "Scan Information" tab shows the scan status as "Finished". Below this, several progress bars are displayed for different scan modules, all of which are 100% complete (indicated by red bars):

- Directory Indexing
- HTTP Methods
- Lotus Domino DB Enumeration
- Simple SQL-Injection
- Boolean SQL-Injection
- Simple Cross Site Scripting Checks

## Chat of SQL-injection finding:

### Finding "SQL-Injection" [Chat-ID: 3]

Browser-View Fuzzer Manual Request

Request:

Text Hex

Grep Highlight Reset

```
POST http://192.168.178.196/mutillidae/index.php?page=login.php HTTP/1.1
Host: 192.168.178.196
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.6) Gecko/20100625 Firefox/3.6.6
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Referer: http://192.168.178.196/mutillidae/index.php?page=login.php
Cookie: uid=5
Content-Type: application/x-www-form-urlencoded
Content-Length: 52
Connection: Close
Proxy-Connection: Close
Accept-Encoding: None

user_name=&password=testsq&Submit_button=Submit
```

Response:

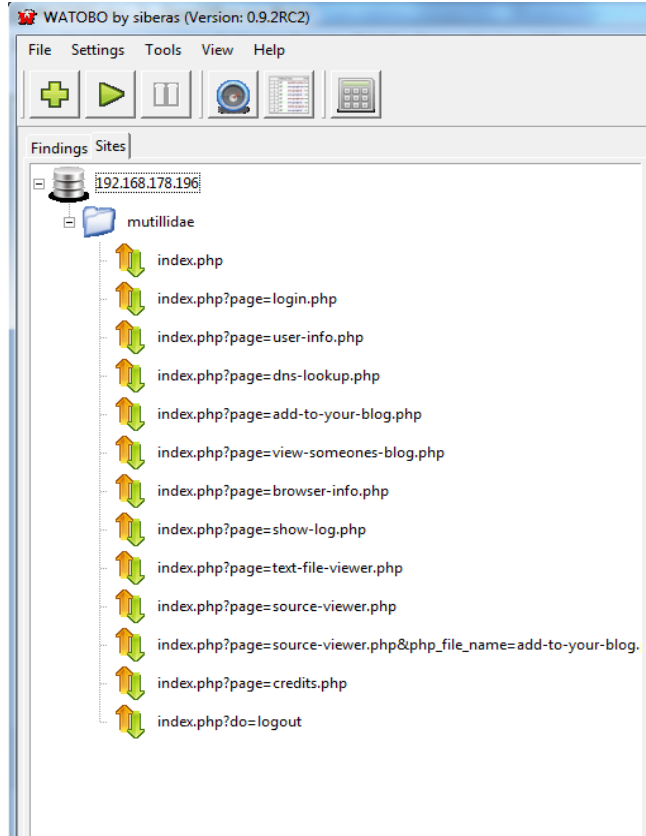
Text Tagless Hex

Grep Highlight Reset

```
090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1

ntax; check the manual that corresponds to your MySQL server version for the right syntax to use near
```

Sites:



Chats:

Doc Filter		Text Filter		Options		
Clear				<input type="checkbox"/> Full Request	<input type="checkbox"/> Full Response	
Apply						
	Method	Host	Path	Parameters	Status	Se
1	GET	192.168.178.196	mutillidae/index.php		200 OK	
2	GET	192.168.178.196	mutillidae/index.php	page=login.php	200 OK	
3	POST	192.168.178.196	mutillidae/index.php	user_name=testsql&passwo	200 OK	
4	GET	192.168.178.196	mutillidae/index.php	page=user-info.php	200 OK	
5	POST	192.168.178.196	mutillidae/index.php	view_user_name=testsql&pi	200 OK	
6	GET	192.168.178.196	mutillidae/index.php	page=dns-lookup.php	200 OK	
7	POST	192.168.178.196	mutillidae/index.php	target_host=testsql&Submit	200 OK	
8	POST	192.168.178.196	mutillidae/index.php	target_host=sd+%26%26+ls	200 OK	
9	POST	192.168.178.196	mutillidae/index.php	target_host=sd+%26+dir&S	200 OK	
10	GET	192.168.178.196	mutillidae/index.php	page=add-to-your-blog.ph	200 OK	
11	POST	192.168.178.196	mutillidae/index.php	input_from_form=testxss&S	200 OK	
12	GET	192.168.178.196	mutillidae/index.php	page=view-someones-blog.	200 OK	
13	POST	192.168.178.196	mutillidae/index.php	show_only_user=Show+All+	200 OK	
14	GET	192.168.178.196	mutillidae/index.php	page=login.php	200 OK	
15	POST	192.168.178.196	mutillidae/index.php	user_name=john&password	200 OK	ui
16	GET	192.168.178.196	mutillidae/index.php		200 OK	
17	GET	192.168.178.196	mutillidae/index.php	page=add-to-your-blog.ph	200 OK	
18	POST	192.168.178.196	mutillidae/index.php	input_from_form=testxss&S	200 OK	
19	GET	192.168.178.196	mutillidae/index.php	page=view-someones-blog.	200 OK	
20	POST	192.168.178.196	mutillidae/index.php	show_only_user=admin&Su	200 OK	
21	POST	192.168.178.196	mutillidae/index.php	show_only_user=Show+All+	200 OK	
22	GET	192.168.178.196	mutillidae/index.php	page=browser-info.php	200 OK	
23	GET	192.168.178.196	mutillidae/index.php	page=show-log.php	200 OK	
24	GET	192.168.178.196	mutillidae/index.php	page=text-file-viewer.php	200 OK	
25	POST	192.168.178.196	mutillidae/index.php	text_file_name=http%3A%2f	200 OK	
26	GET	192.168.178.196	mutillidae/index.php	page=source-viewer.php	200 OK	
27	GET	192.168.178.196	mutillidae/index.php	page=source-viewer.php&p	200 OK	
28	GET	192.168.178.196	mutillidae/index.php		200 OK	
29	GET	192.168.178.196	mutillidae/index.php	page=show-log.php	200 OK	
30	GET	192.168.178.196	mutillidae/index.php	page=credits.php	200 OK	
31	GET	192.168.178.196	mutillidae/index.php		200 OK	
32	GET	192.168.178.196	mutillidae/index.php		200 OK	
33	GET	192.168.178.196	mutillidae/index.php		200 OK	
34	GET	192.168.178.196	mutillidae/index.php	do=logout	200 OK	ui
35	GET	192.168.178.196	mutillidae/index.php		200 OK	
36	GET	192.168.178.196	mutillidae/index.php		200 OK	



Single chat (1):

**Chat-ID: 1**

[Browser-View](#) [Fuzzer](#) [Manual Request](#)

Request:

Text | Hex

Grep Highlight Reset

```
GET http://192.168.178.196/mutillidae/index.php HTTP/1.1
Host: 192.168.178.196
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.6) Gecko/20100625 Firefox/3.6.6
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Proxy-Connection: keep-alive
Cookie: uid=5
```

Response:

Text | Tagless | Hex

Grep Highlight Reset

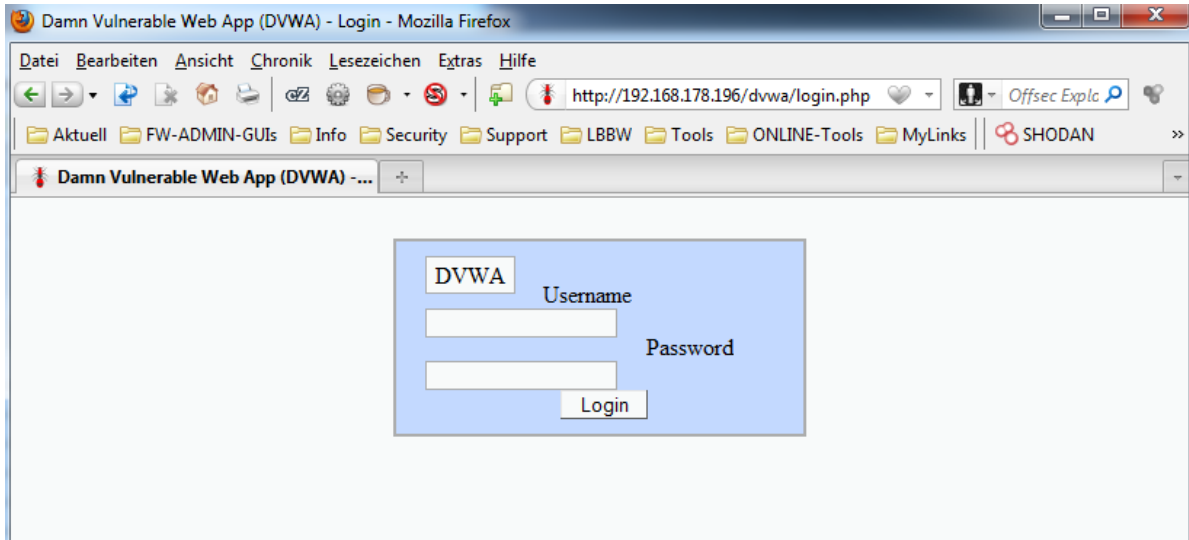
```
HTTP/1.1 200 OK
Date: Tue, 20 Jul 2010 17:07:21 GMT
Server: Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8i mod_autoindex_color PHP/5.3
X-Powered-By: PHP/5.3.1
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html

270b
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/1999/04/01-dtd.html">
<html>
<head>
<meta content="text/html; charset=us-ascii" http-equiv="content-type">
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
</head>
<body>
<table border="0" width="100%" cellspacing="0" cellpadding="0">
<tr><td bgcolor="#88ff88" align="center" colspan="2">
```

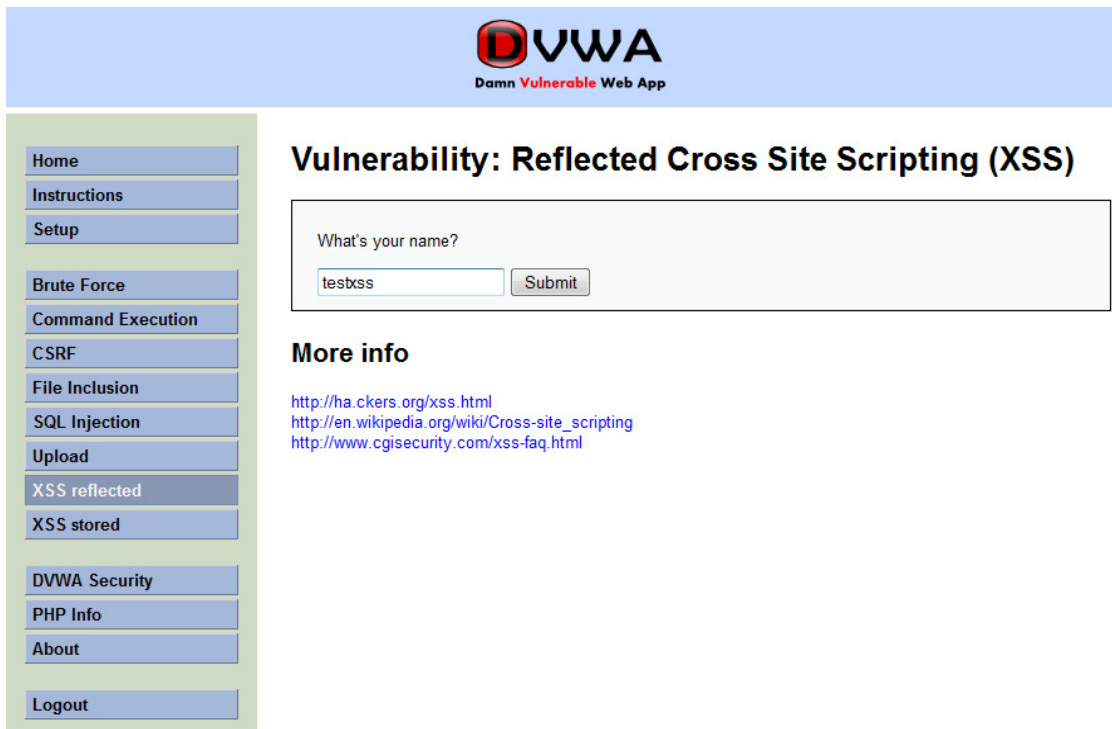
### 3.3 Session management

To demonstrate session management we need an application where you have to login first, like DVWA.

**Example: Damn Vulnerable Web App [3]:**



First we login with *admin/password* and browse the application (passive checks):



*Note: Logout from the application after browsing since we want to test session management.*

First we need to identify all chats (request/response pair) which are responsible for the login process and add them to the login script (add the chat where the cookie is set and where the login credentials are posted - "**302 found**")

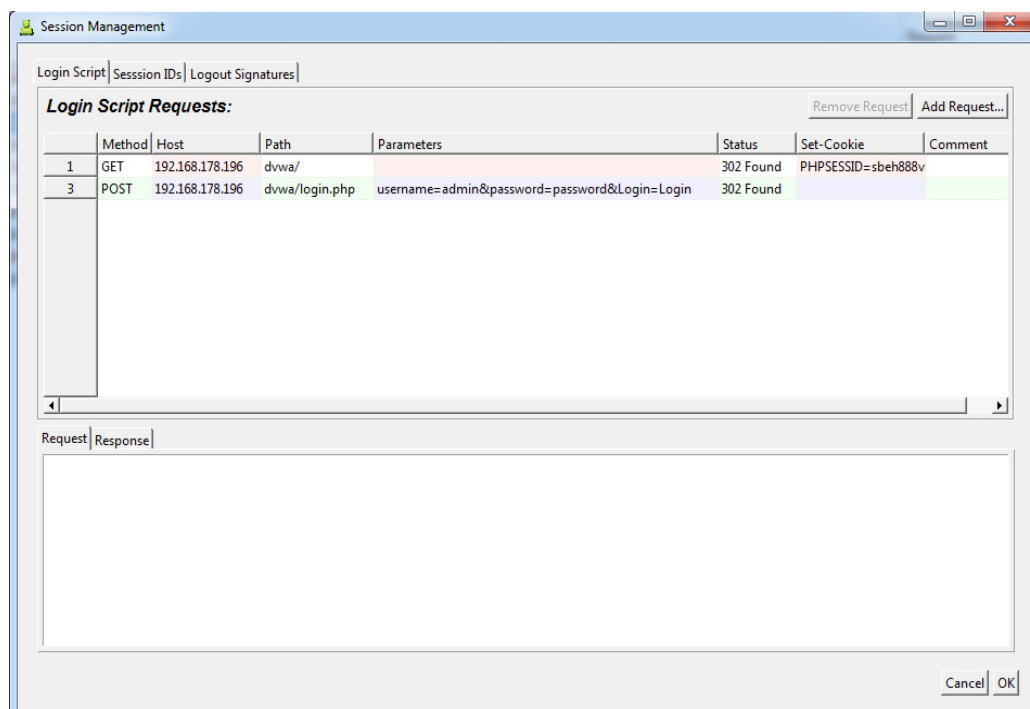
In our example it is Chat 1 where the session cookie is set:

	Method	Host	Path	Parameters	Status	Set-Cookie
1	GET	192.168.178.196	dvwa/		302 Found	PHPSESSID=sbeh888vm5j75a920r0s4bt
2	GET	192.168.178.196	dvwa/login.php		200 OK	
3	POST	192.168.178.196	dvwa/login.php	username=admin&password=password	302 Found	
4	GET	192.168.178.196	dvwa/index.php		200 OK	
5	GET	192.168.178.196	dvwa/vulnerabilities/xss_r/		200 OK	
6	GET	192.168.178.196	dvwa/vulnerabilities/xss_r/	name=testxss	200 OK	

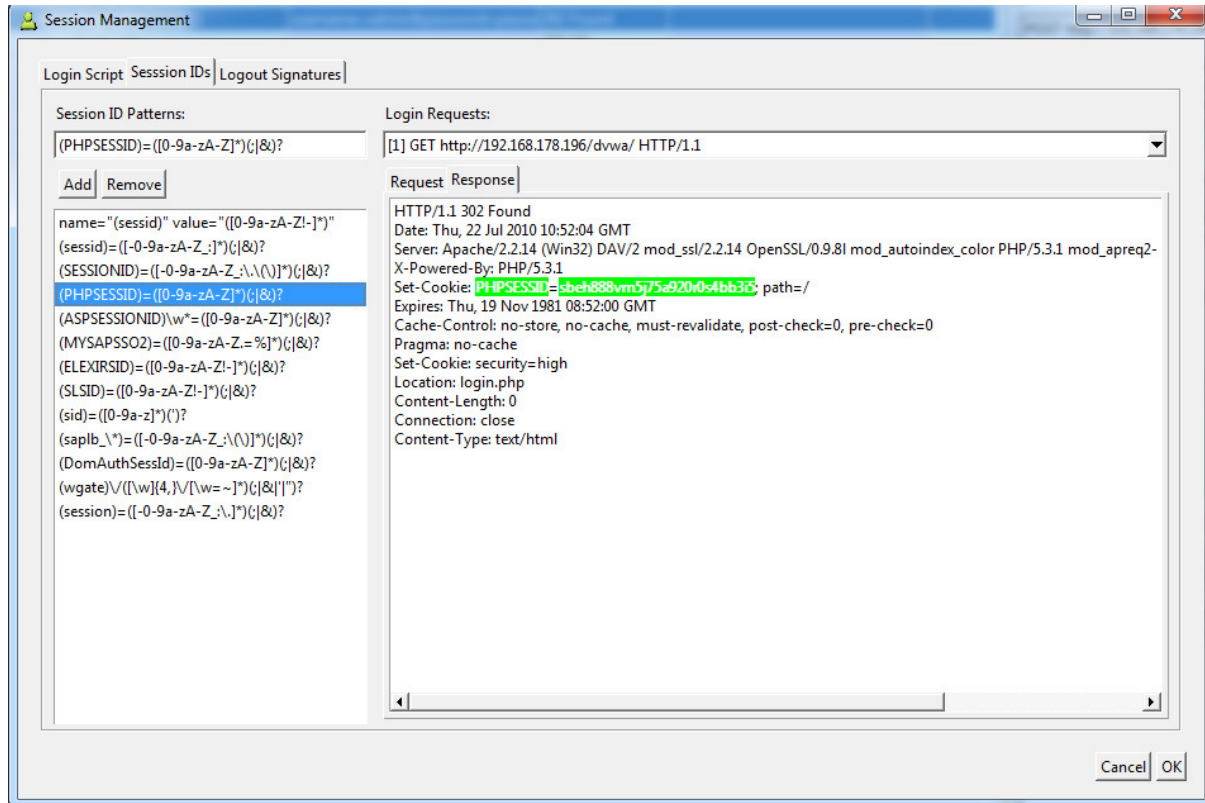
as well as chat 3 where the login credentials are posted:

	Method	Host	Path	Parameters	Status	Set-Cookie
1	GET	192.168.178.196	dvwa/		302 Found	PHPSESSID=sbeh888vm5j75a920r0s4bt
2	GET	192.168.178.196	dvwa/login.php		200 OK	
3	POST	192.168.178.196	dvwa/login.php	username=admin&password=password	302 Found	
4	GET	192.168.178.196	dvwa/index.php		200 OK	
5	GET	192.168.178.196	dvwa/vulnerabilities/xss_r/		200 OK	
6	GET	192.168.178.196	dvwa/vulnerabilities/xss_r/	name=testxss	200 OK	
7	GET	192.168.178.196	dvwa/vulnerabilities/xss_s/		200 OK	
8	POST	192.168.178.196	dvwa/vulnerabilities/xss_s/	txtName=testxss&mbxMessage=testxss	200 OK	
9	GET	192.168.178.196	dvwa/vulnerabilities/sqli/		200 OK	
10	GET	192.168.178.196	dvwa/vulnerabilities/sqli/	id=testsql&Submit=Submit	200 OK	

To validate the session settings open the Session Management Menu (Settings → Session management):



Open the “Session Ids” tab, then open the “Response” tab to see where the session information has been set:

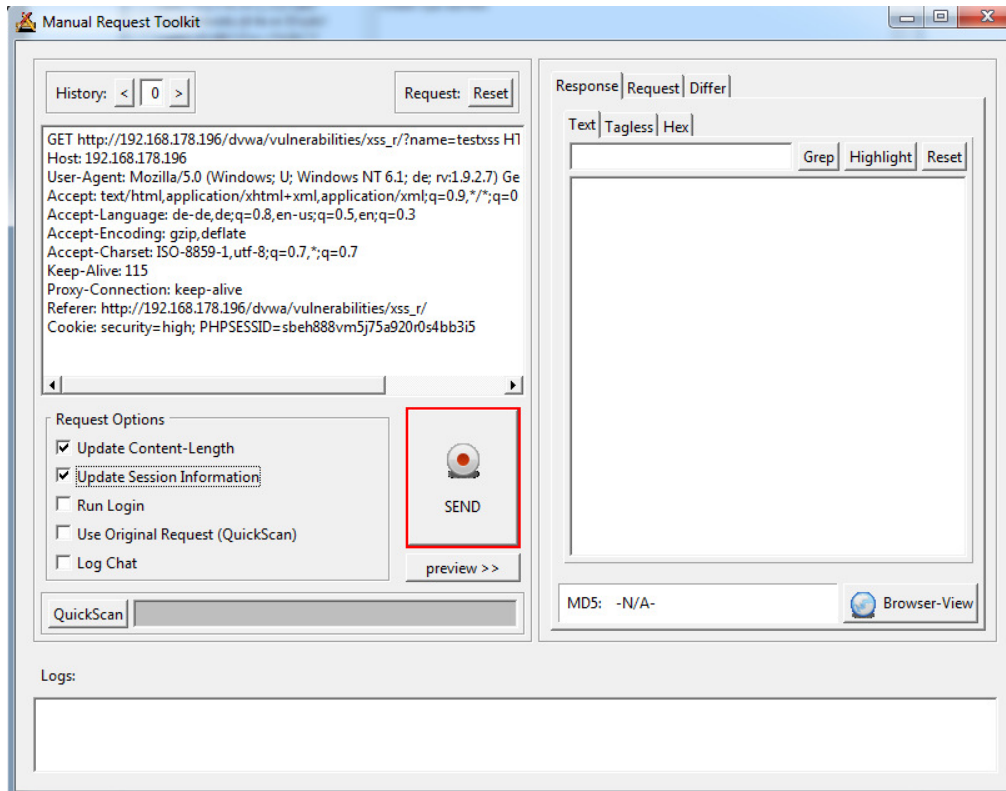


After we finished verifying our session management settings let's see if it really works:

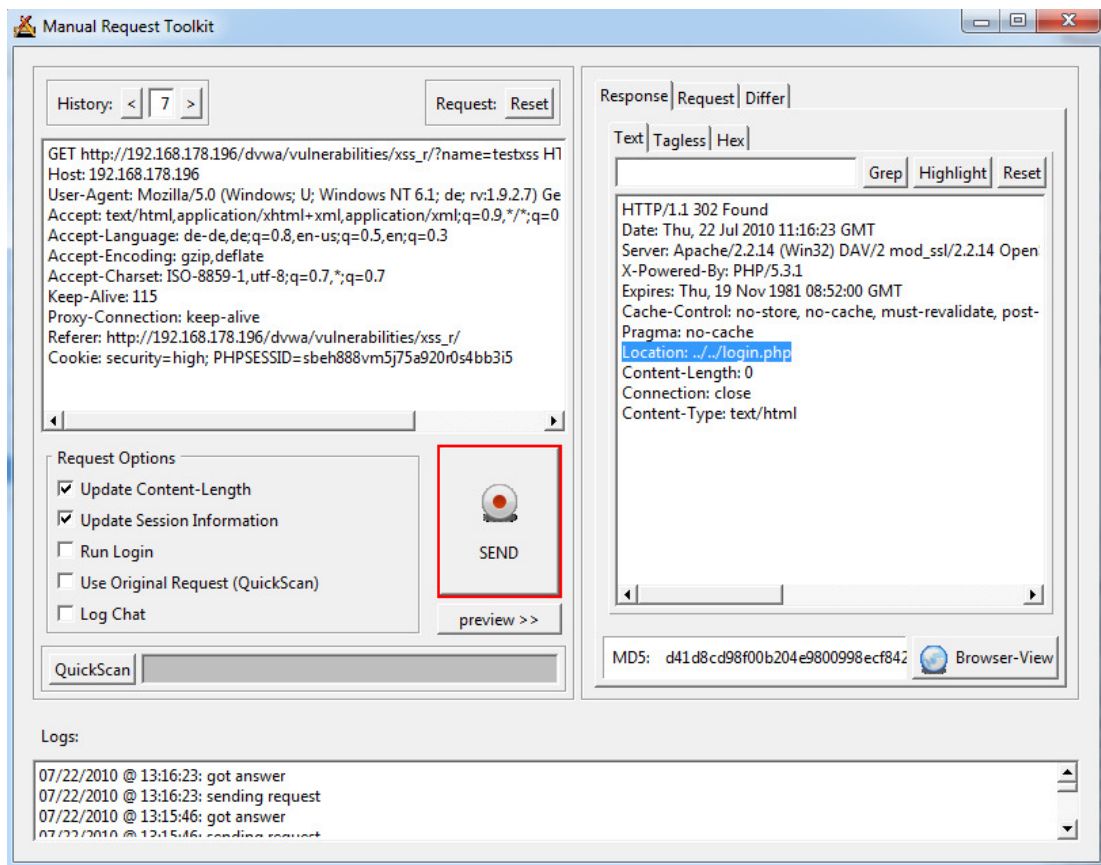
→ chose a chat and open the Manual Request Editor:

	Method	Host	Path	Parameters	Status	Set-Cookie
1	GET	192.168.178.196	dvwa/		302 Found	PHPSESSID=sbeh888vm5j75a920r0s4bt
2	GET	192.168.178.196	dvwa/login.php		200 OK	
3	POST	192.168.178.196	dvwa/login.php	username=admin&password=password	302 Found	
4	GET	192.168.178.196	dvwa/index.php		200 OK	
5	GET	192.168.178.196	dvwa/vulnerabilities/xss_r/		200 OK	
6	GET	192.168.178.196	dvwa/vulnerabilities/xss_r/	name=testxss	200 OK	
7	GET	192.168.178.196	dvwa/vulnerabilities/xss_s/		200 OK	
8	POST	192.168.178.196	dvwa/vulnerabilities/xss_s/	txtName=testxss&mbxMessage=testxss	200 OK	
9	GET	192.168.178.196	dvwa/vulnerabilities/sqli/		200 OK	
10	GET	192.168.178.196	dvwa/vulnerabilities/sqli/	id=testsql&Submit=Submit	200 OK	

### "Update Session Information" enabled:

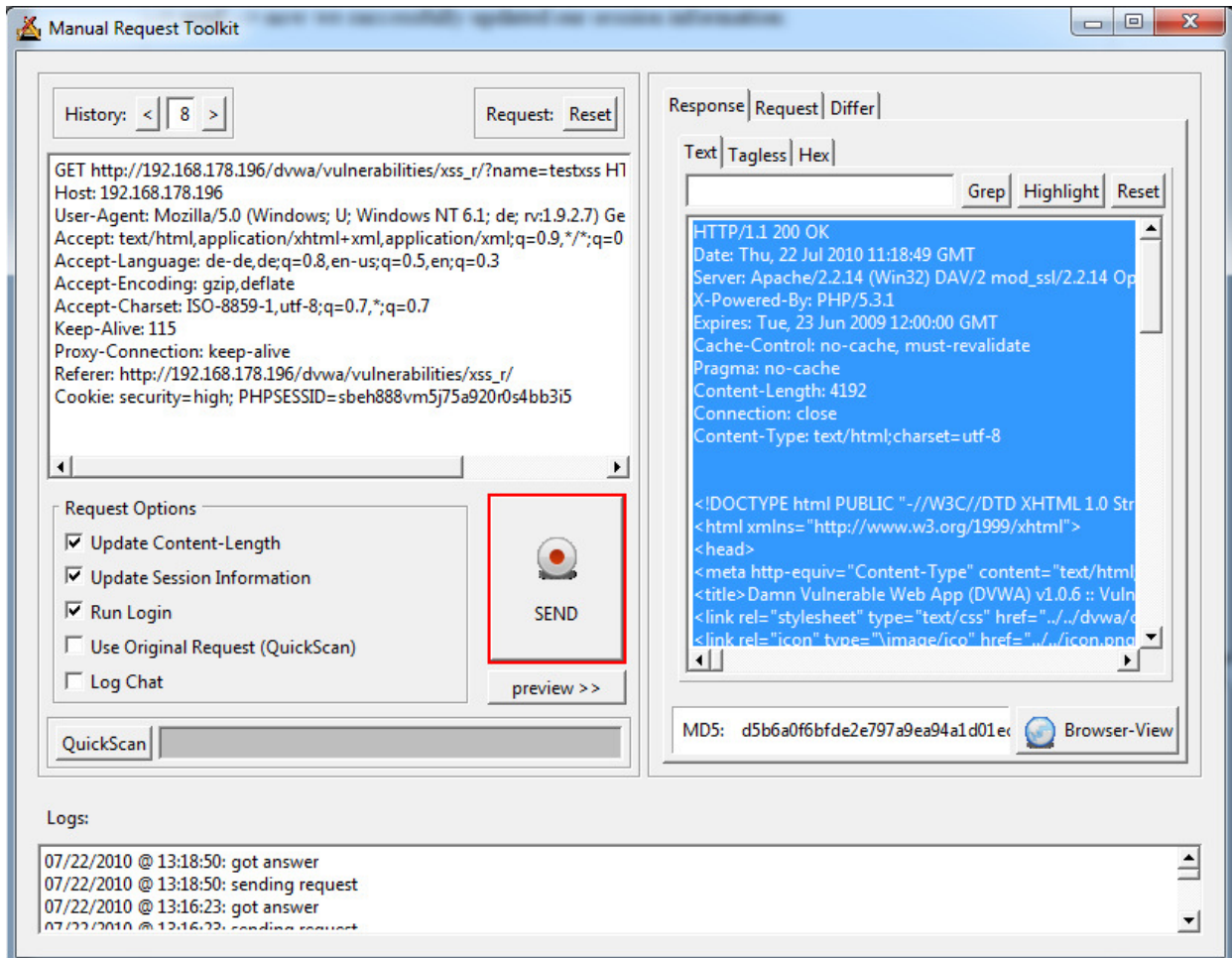


→ **Send** → we are redirected to the login page:



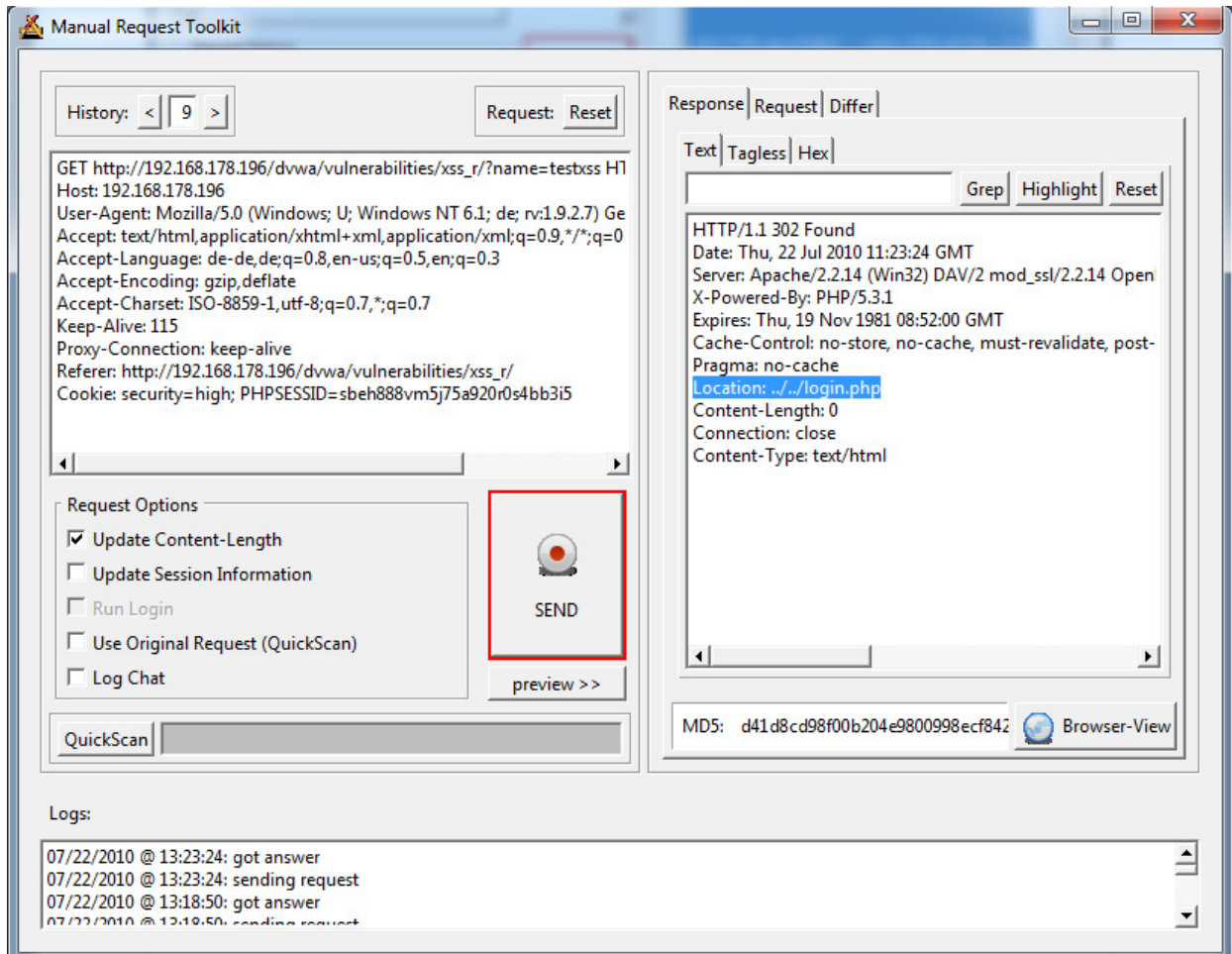
Now check "**Run Login**" which runs the login script (chats 1 + 3) to get valid session information

→ **Send** → now we successfully updated our session information:



Once you got a valid session information you can disable "**Run Login**" because the session information is still remembered

Disabling "**Update Session Information**" will redirect you to the login page again:



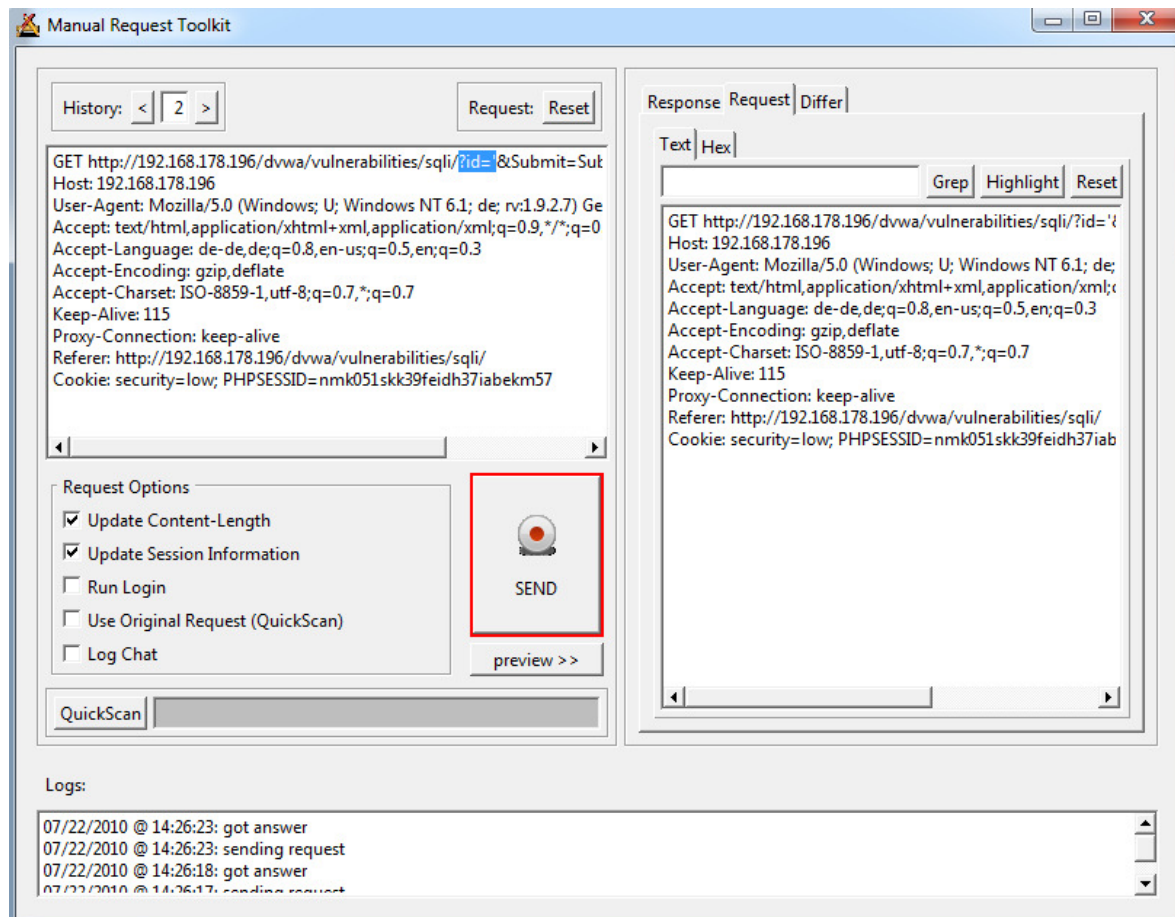
Now you can also try an active scan but do not forget to exclude the login- and logout chats from scanning.

### 3.4 Manual requests

→ double-click the desired chat:

	Method	Host	Path	Parameters	Status	Set-Cookie	Comment
1	GET	192.168.178.196	dvwa/login.php		200 OK		
2	POST	192.168.178.196	dvwa/login.php	username=admin&password=	302 Found		
3	GET	192.168.178.196	dvwa/index.php		200 OK		
4	GET	192.168.178.196	dvwa/vulnerabilities/sqli/		200 OK		
5	GET	192.168.178.196	dvwa/vulnerabilities/sqli/	id=testsqli&Submit=Submit	200 OK		
6	GET	192.168.178.196	dvwa/vulnerabilities/xss_s/		200 OK		
7	GET	192.168.178.196	dvwa/security.php		200 OK		
8	POST	192.168.178.196	dvwa/security.php	security=low&seclev_submit=	302 Found	security=low	
9	GET	192.168.178.196	dvwa/security.php		200 OK		
10	GET	192.168.178.196	dvwa/vulnerabilities/xss_s/		200 OK		
11	POST	192.168.178.196	dvwa/vulnerabilities/xss_s/	txtName=hugo&mtxMessage=	200 OK		
12	POST	192.168.178.196	dvwa/vulnerabilities/xss_s/	txtName=hugo&mtxMessage=	200 OK		
13	GET	192.168.178.196	dvwa/vulnerabilities/fi/	page=include.php	200 OK		
14	GET	192.168.178.196	dvwa/		200 OK		

Here you can change what you like in the request (e.g. **id='**) and send it away.



The screenshot shows the Manual Request Toolkit interface. The main window displays a request history with the following details:

```

GET http://192.168.178.196/dvwa/vulnerabilities/sqli/?id='&Submit=Submit
Host: 192.168.178.196
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.7) Ge
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Proxy-Connection: keep-alive
Referer: http://192.168.178.196/dvwa/vulnerabilities/sqli/
Cookie: security=low; PHPSESSID=nmk051skk39feidh37iabekm57
  
```

Below the request text, there are "Request Options" with checkboxes for "Update Content-Length", "Update Session Information", "Run Login", "Use Original Request (QuickScan)", and "Log Chat". A prominent "SEND" button is highlighted with a red box. A "QuickScan" button is also visible.

The right-hand pane shows the response details:

```

GET http://192.168.178.196/dvwa/vulnerabilities/sqli/?id='
Host: 192.168.178.196
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de;
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.3
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Proxy-Connection: keep-alive
Referer: http://192.168.178.196/dvwa/vulnerabilities/sqli/
Cookie: security=low; PHPSESSID=nmk051skk39feidh37iabekm57
  
```

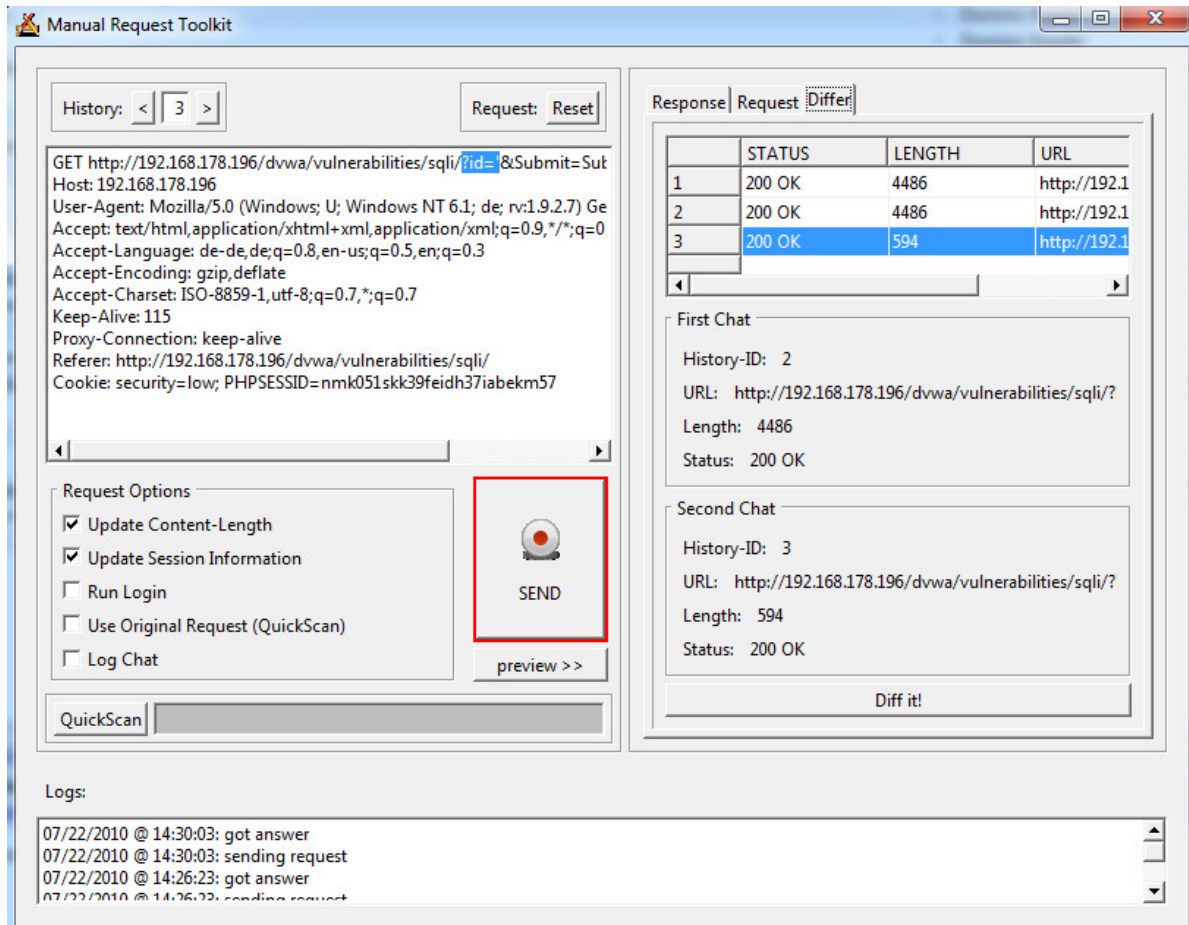
At the bottom, a "Logs" section shows the following entries:

```

07/22/2010 @ 14:26:23: got answer
07/22/2010 @ 14:26:23: sending request
07/22/2010 @ 14:26:18: got answer
07/22/2010 @ 14:26:17: sending request
  
```



The differ function is totally awesome - you can compare 2 chats from the same type:



Choose the 2 chats which you want to be compared, then click on **“Diff it!”**

See the comparison of the REQUEST

Chat Differ

Response		Request	
Type	Pos	Count	
+	10	1	
-	10	1	

**Original**

```
GET http://192.168.178.196/dvwa/vulnerabilities/sqli/?id='&Submit=Submit
Host: 192.168.178.196
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.7) Gecko
713 Firefox/3.6.7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Referer: http://192.168.178.196/dvwa/vulnerabilities/sqli/
Cookie: security=low; PHPSESSID=nmk051skt39feidh37iabekm57
Connection: Close
Proxy-Connection: Close
Accept-Encoding: None
```

**New**

```
GET http://192.168.178.196/dvwa/vulnerabilities/sqli/?id='&Submit=Submit
Host: 192.168.178.196
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.7) Gecko
713 Firefox/3.6.7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Referer: http://192.168.178.196/dvwa/vulnerabilities/sqli/
Cookie: security=high; PHPSESSID=nmk051skt39feidh37iabekm57
Connection: Close
Proxy-Connection: Close
Accept-Encoding: None
```

and the RESPONSE of the chats:

Chat Differ

Response		Request	
Type	Pos	Count	
+	1	1	
-	1	1	
-	5	2	
+	5	2	
+	8	1	
-	8	1	
+	10	1	
-	10	1	
+	12	83	
-	12	2	

**Original**

```
HTTP/1.1 200 OK
Date: Thu, 22 Jul 2010 12:42:01 GMT
Server: Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8i mod_a
color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.1
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-ch
Pragma: no-cache
Content-Length: 160
Connection: close
Content-Type: text/html

<pre> You have an error in your SQL syntax; check the manual that corresp
your MySQL server version for the right syntax to use near '''' at line 1</pre>
```

**New**

```
HTTP/1.1 200 OK
Date: Thu, 22 Jul 2010 12:42:12 GMT
Server: Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8i mod
color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.1
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Content-Length: 4097
Connection: close
Content-Type: text/html; charset=utf-8

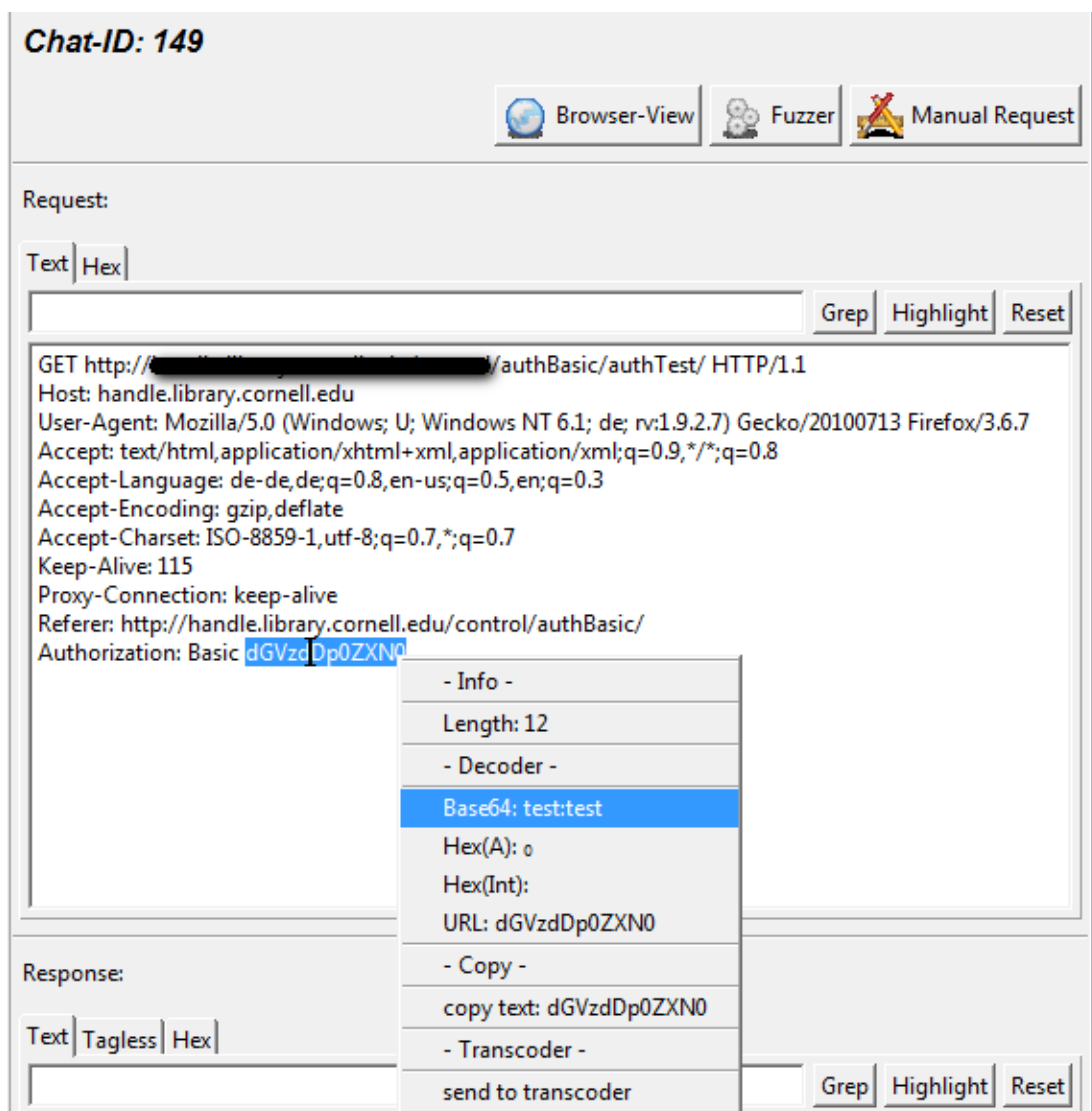
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
html/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
<title>Damn Vulnerable Web App (DVWA) v1.0.6 - Vulnerability: SQL Inje
tle>
<link rel="stylesheet" type="text/css" href="http://www.w3.org/1999/xhtml">
<link rel="icon" type="image/ico" href="http://www.w3.org/1999/xhtml">
<script type="text/javascript" src="http://www.w3.org/1999/xhtml">
</head>
<body>
<div id="container">
<div id="header">

<div id="main_menu">
<div id="main_menu_padded">
<ul>
<li onclick="window.location='./././' class=""><a href='./././'>Ho
</li>
<li onclick="window.location='././instructions.php' class=""><a href='././instructions.php">Instructions</a>
</li>
<li onclick="window.location='././setup.php' class=""><a href='././setup.php">Setup</a>
</li>
</ul>
</div>
```

### 3.5 More functions

#### Inline De-/Encoding:

If you for example have a HTTP basic authentication you can decode the base64 encoded string immediately with WAYOBO. Just select the string, right click your mouse and you can see immediately the credentials *test/test*.

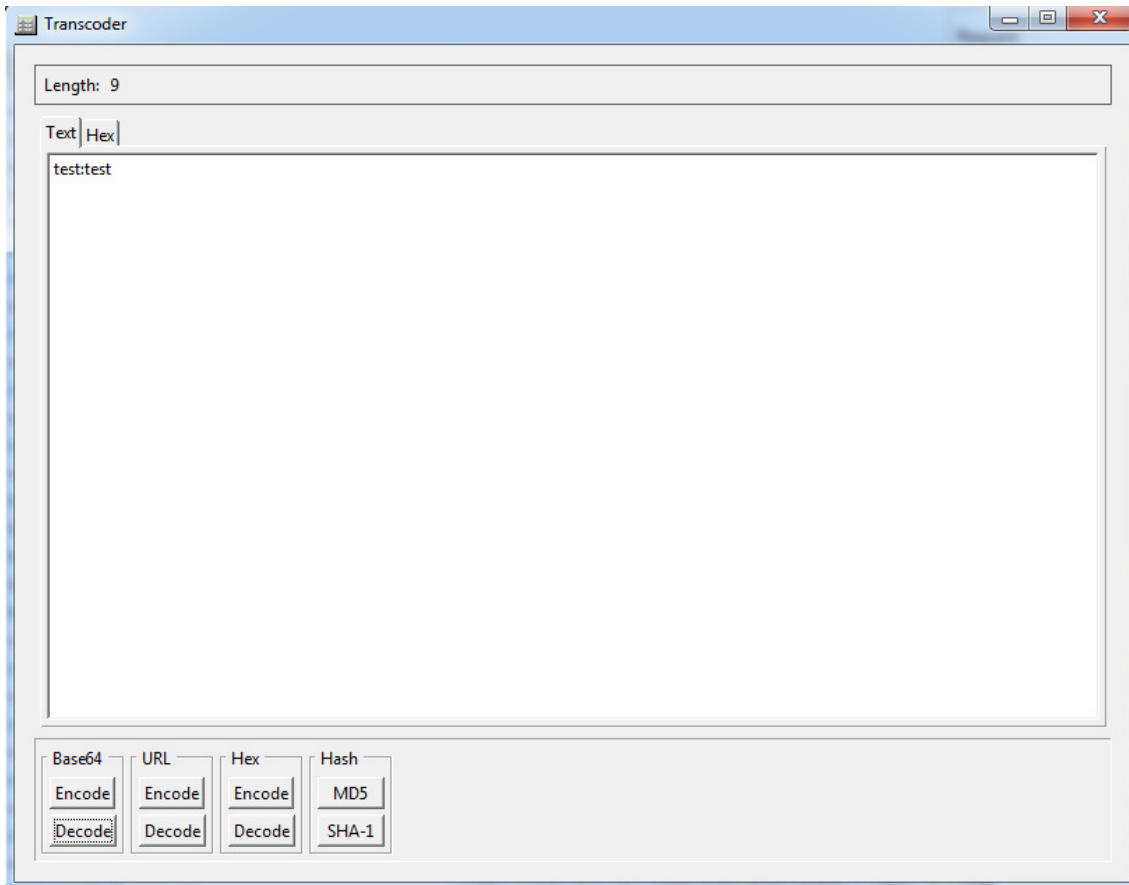


The screenshot shows the WAYOBO web application interface. At the top, there are three buttons: "Browser-View", "Fuzzer", and "Manual Request". Below these is a "Request:" section with "Text" and "Hex" tabs. A search bar with "Grep", "Highlight", and "Reset" buttons is present. The main content area displays an HTTP request:

```
GET http://[redacted]/authBasic/authTest/ HTTP/1.1
Host: handle.library.cornell.edu
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.7) Gecko/20100713 Firefox/3.6.7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Proxy-Connection: keep-alive
Referer: http://handle.library.cornell.edu/control/authBasic/
Authorization: Basic dGVzdDp0ZXN0
```

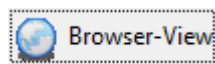
The string "dGVzdDp0ZXN0" in the Authorization header is selected. A context menu is open over it, showing options: "- Info -", "Length: 12", "- Decoder -", "Base64: test:test" (highlighted), "Hex(A): 0", "Hex(Int):", "URL: dGVzdDp0ZXN0", "- Copy -", "copy text: dGVzdDp0ZXN0", "- Transcoder -", and "send to transcoder". Below the request is a "Response:" section with "Text", "Tagless", and "Hex" tabs, and another search bar with "Grep", "Highlight", and "Reset" buttons.

You can also send the selected string to the **transcoder** which can do several de-/encodings:



### **Browser-View:**

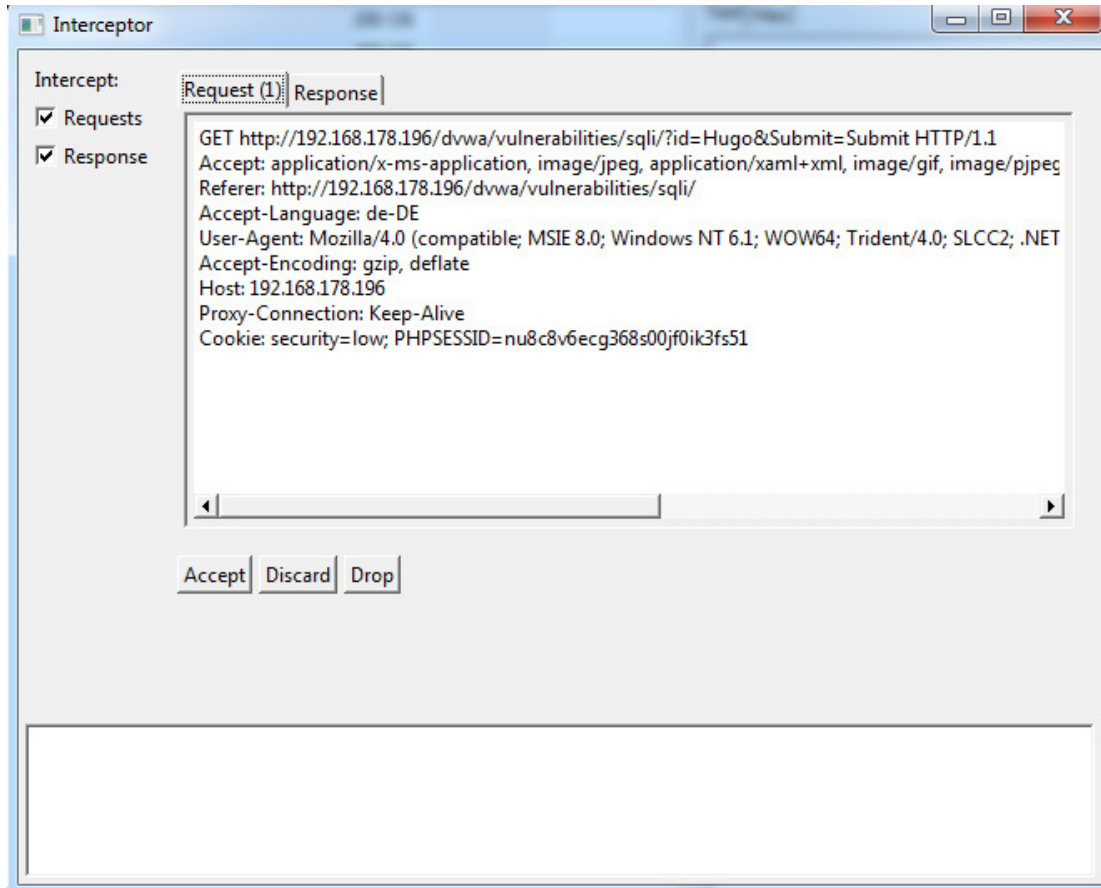
A nice feature is when you click on



you can see the response in your browser (on windows only IE supported )

### **Interceptor:**

Of course, WATOBO has an interceptor too:



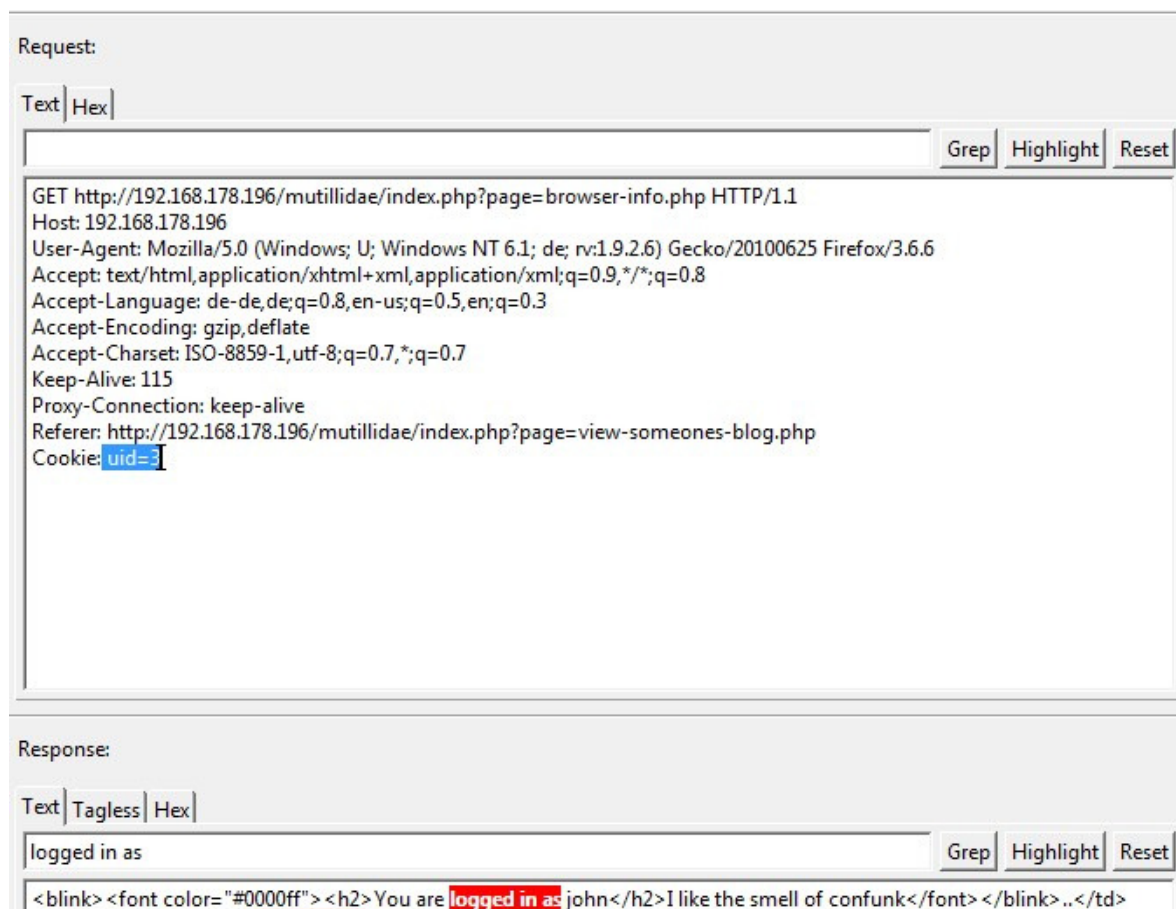


## 3.6 Fuzzing

### - Enumerate Usernames -

Here we use the fuzzer for collecting usernames from the mutillidae web application [2]. First examine the response for the username with uid=3.

Here we have a corresponding username („logged in as john“):



The screenshot displays a web application fuzzer interface with two main sections: Request and Response.

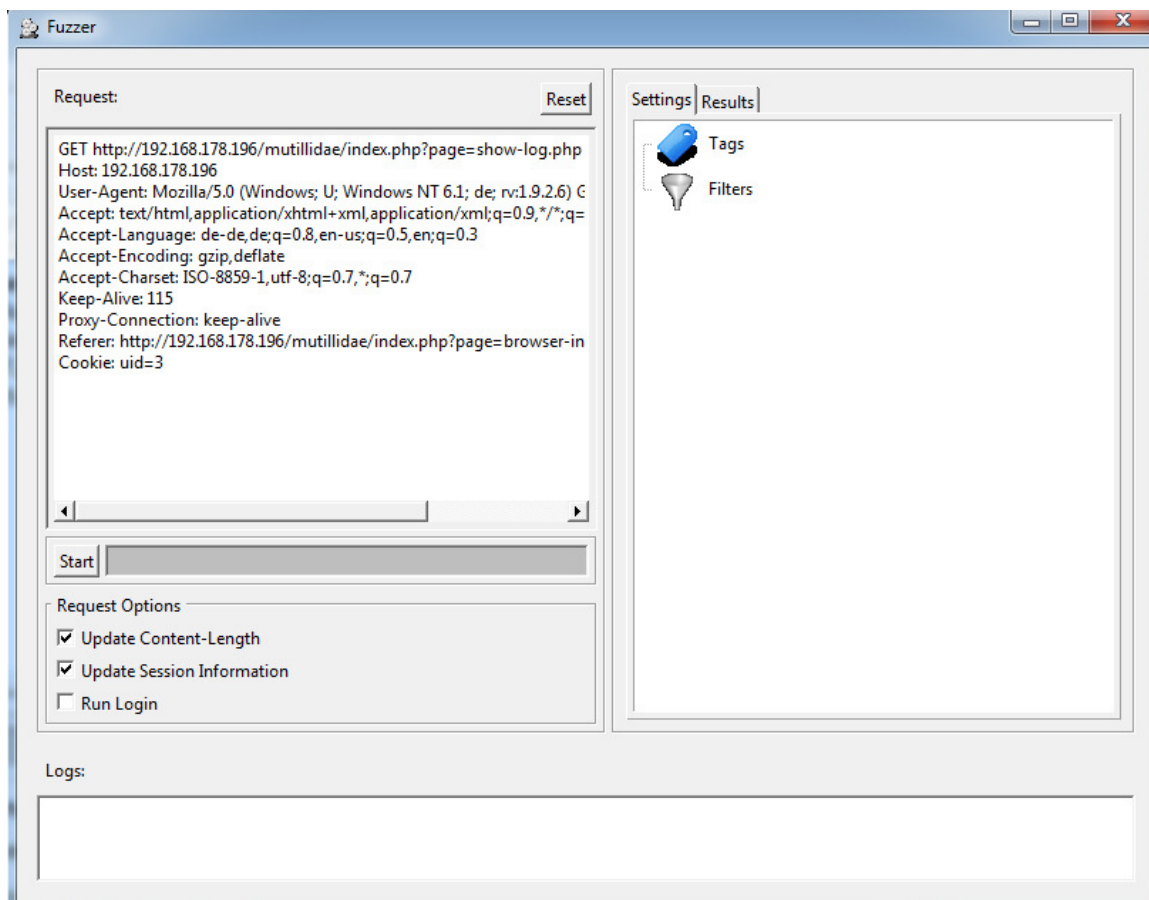
**Request:** The interface shows a text input field with the following content: `GET http://192.168.178.196/mutillidae/index.php?page=browser-info.php HTTP/1.1`. Below the input field are buttons for `Grep`, `Highlight`, and `Reset`. The request body contains the following headers and cookies:

```
Host: 192.168.178.196
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.6) Gecko/20100625 Firefox/3.6.6
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Proxy-Connection: keep-alive
Referer: http://192.168.178.196/mutillidae/index.php?page=view-someones-blog.php
Cookie: uid=3
```

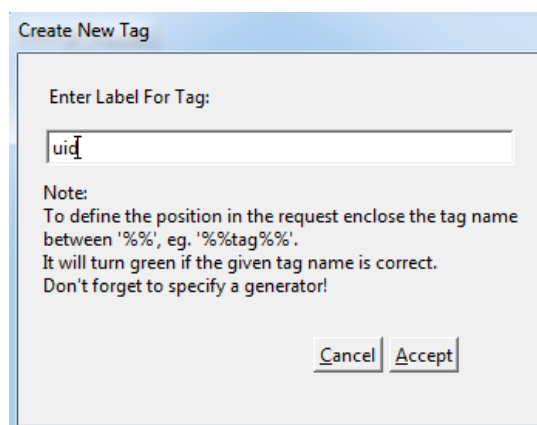
**Response:** The interface shows a text input field with the following content: `logged in as`. Below the input field are buttons for `Grep`, `Highlight`, and `Reset`. The response body contains the following HTML snippet:

```
<blink><font color="#0000ff"><h2>You are logged in as john</h2>I like the smell of confunk</font></blink>..</td>
```

Now open the fuzzer:

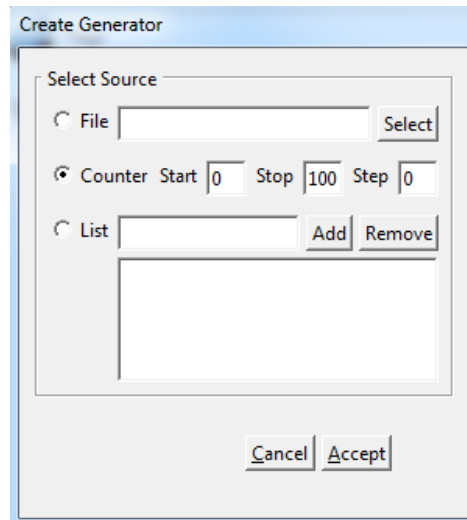


First we have to define a tag by which we can define the position of a generated value in the request later. Double-click on **Tags** and enter a tag name:

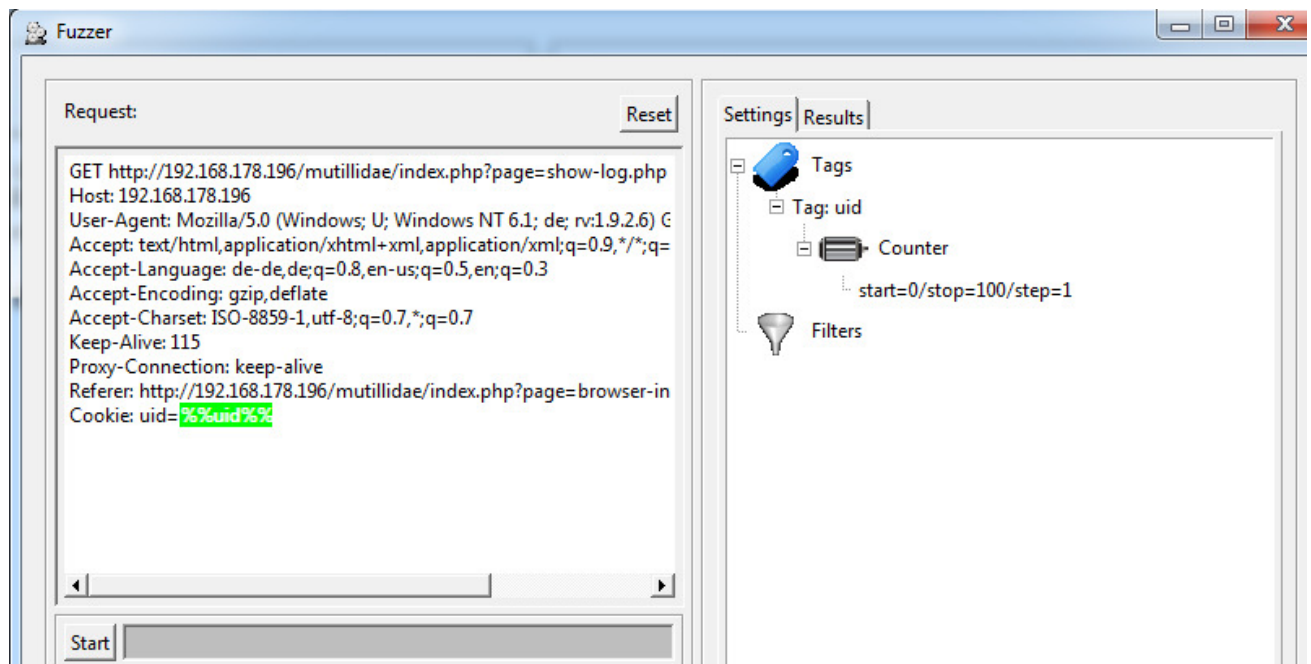


Next we have to define a generator which will produce the values we need. Double-click on "Tag: uid", select "**Counter**" and choose start=0, stop = 100, step = 0 (=1). This results in the values 0,1,2,3,...100.





To define the position of our values inside the request simply enter the tag name enclosed between “%%”. That means replace **uid=3** with **uid=%%uid%%** :

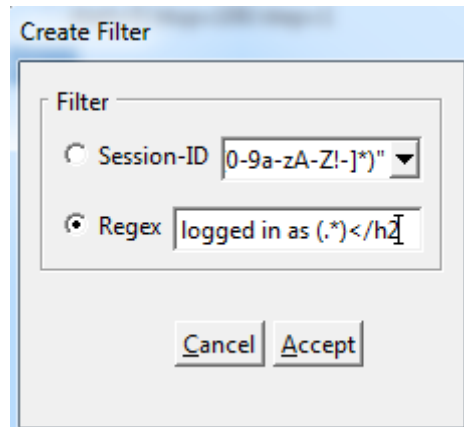


To extract the usernames we also have to define a filter. Double-click on **Filters**. So let's define a regex. In this case the regex

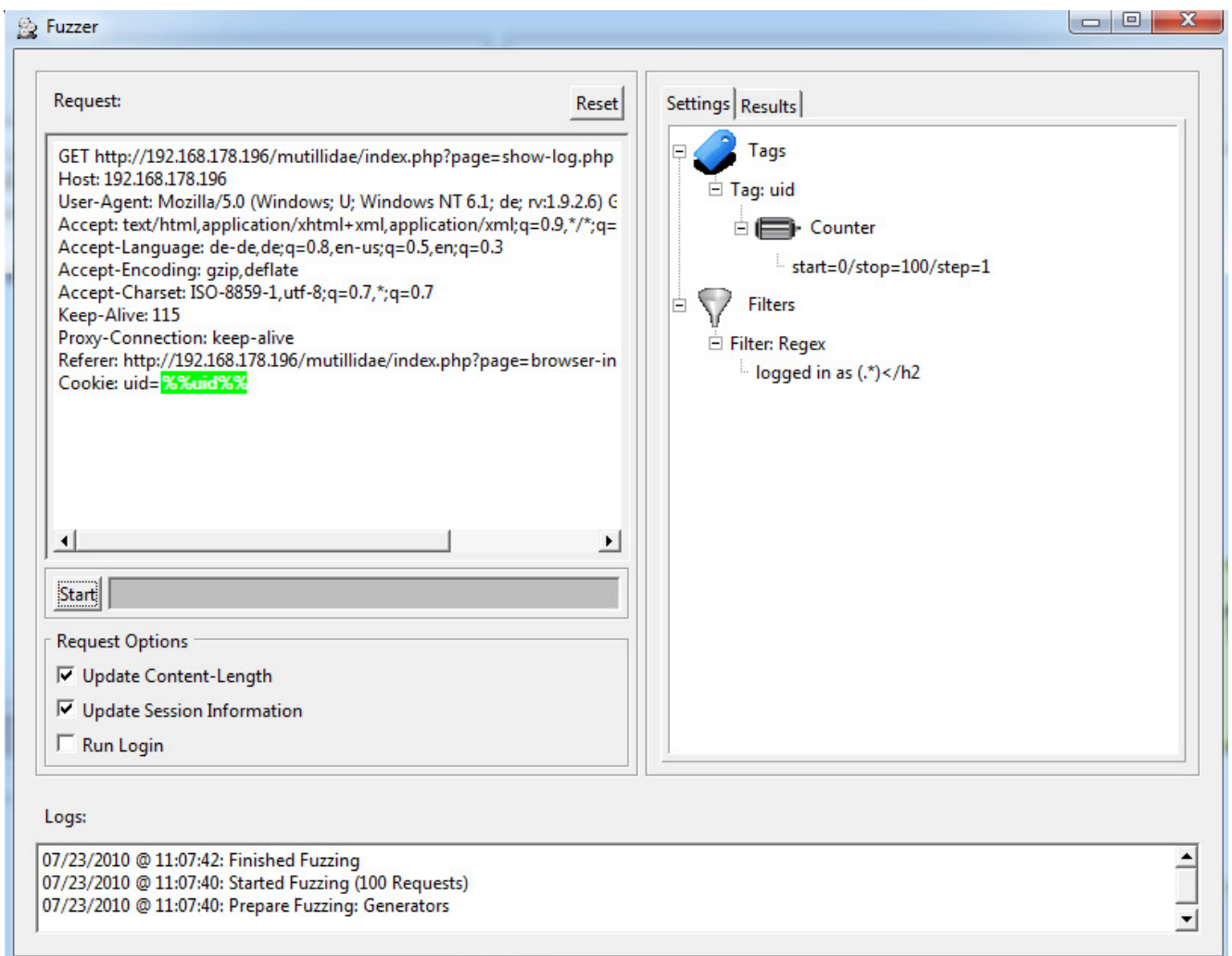
**logged in as (.\*)</h2'**

is just fine.

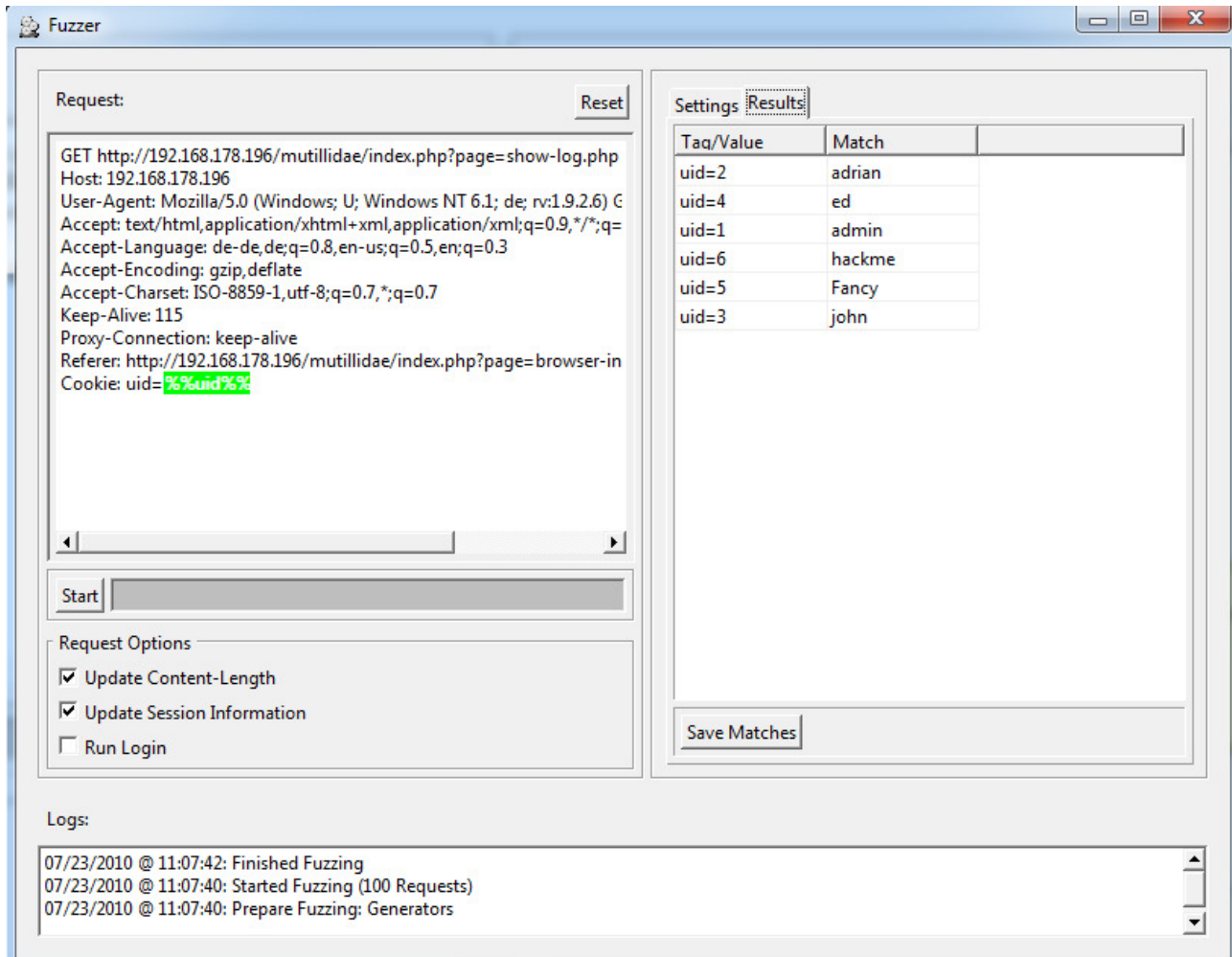
Note: the match value must be enclosed between brackets.



Let's go → click on **Start**



Click on the tab **Results**:



→ we found the users adrian, ed, admin, hackme, Fancy and john.

## - Fuzzing multiple values -

Here we want to enumerate a valid combination of filename + extension. In detail we want to test combinations of 3 filenames (index, test and xxx) and 3 extensions (mp3, wav and php).

First we define a tag and a list generator for the filenames we want to test.

Create a tag for the filenames:

Create New Tag

Enter Label For Tag:

AAA

Note:  
To define the position in the request enclose the tag name between '%%', eg. '%tag%'.  
It will turn green if the given tag name is correct.  
Don't forget to specify a generator!

Cancel Accept

Create the generator:

Create Generator

Select Source

File  Select

Counter Start  Stop  Step

List  Add Remove

xxx  
test  
index

Cancel Accept

Create a tag for the extensions:

Create New Tag

Enter Label For Tag:

BBE

Note:  
To define the position in the request enclose the tag name between '%%', eg. '%tag%'.  
It will turn green if the given tag name is correct.  
Don't forget to specify a generator!

Cancel Accept

Create the generator:

The 'Create Generator' dialog box has a title bar 'Create Generator'. It contains a 'Select Source' section with three radio buttons: 'File', 'Counter', and 'List'. The 'List' radio button is selected. Next to 'List' is a text input field and two buttons: 'Add' and 'Remove'. Below this is a list box containing the items 'wav', 'php', and 'mp3'. At the bottom of the dialog are 'Cancel' and 'Accept' buttons.

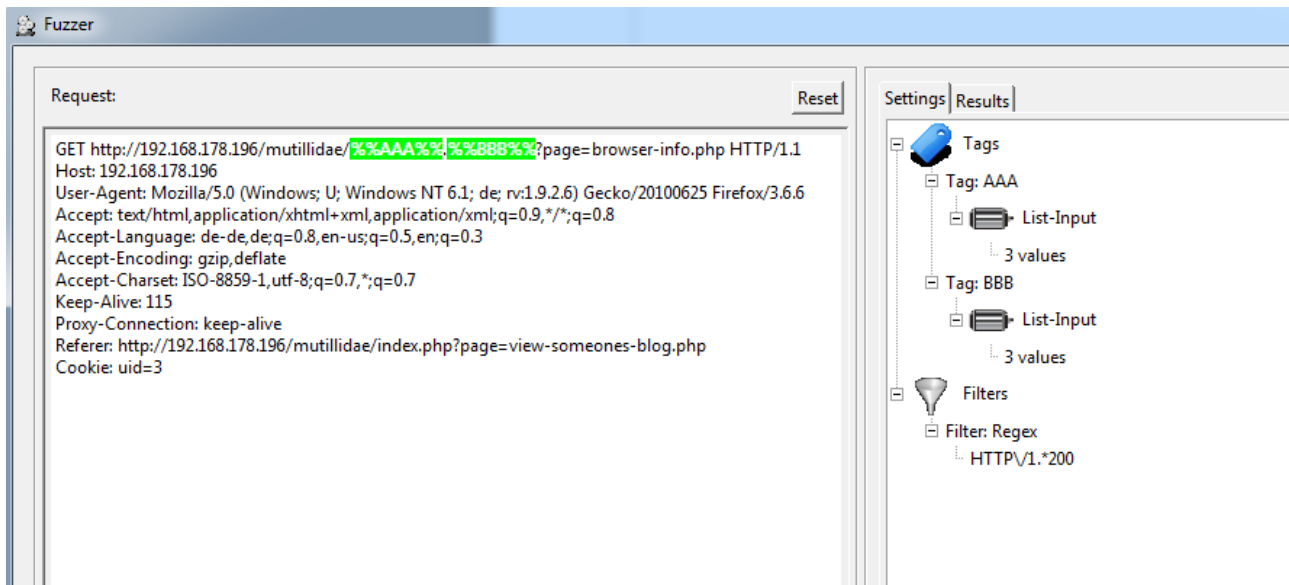
Because we only want to know about valid combinations we define a filter for all

*'HTTP/1.1 200 OK'*

responses:

The 'Create Filter' dialog box has a title bar 'Create Filter'. It contains a 'Filter' section with two radio buttons: 'Session-ID' and 'Regex'. The 'Regex' radio button is selected. Next to 'Regex' is a text input field containing the text 'HTTPV1.\*200'. At the bottom of the dialog are 'Cancel' and 'Accept' buttons.

Next we place our tags:

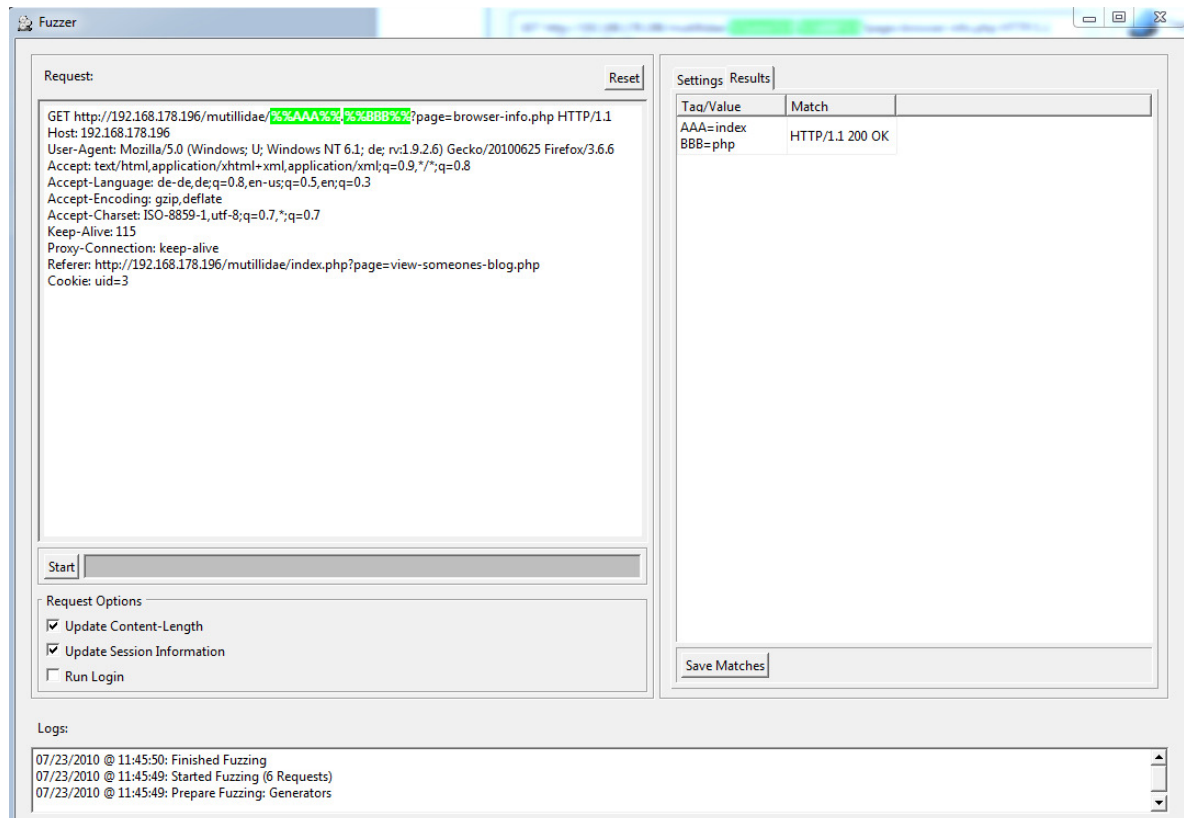


Note, we have:

**filename = % %AAA% %**

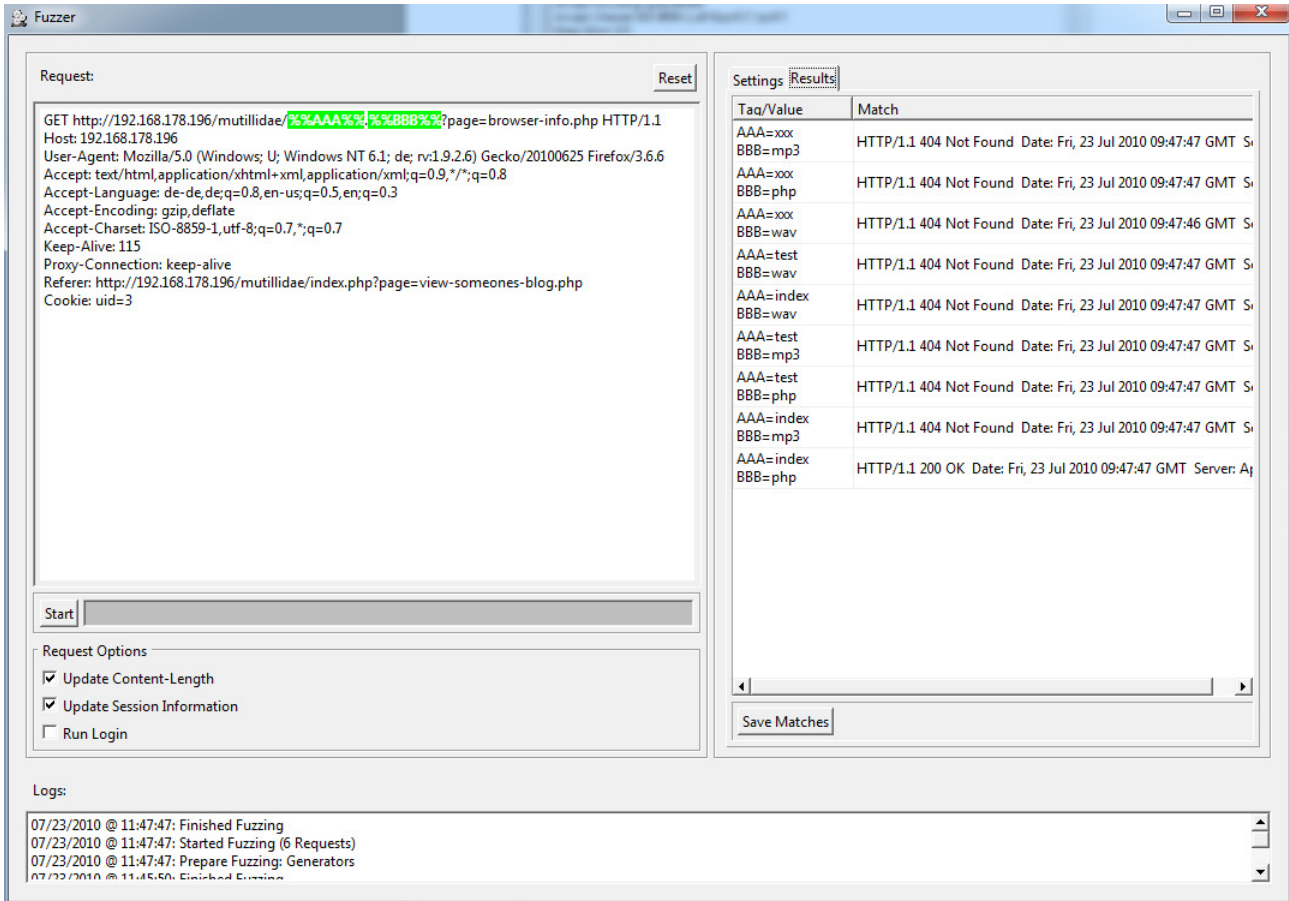
**extension = % %BBB% %**

→ Start fuzzing:



→ we have only one single match.

If you want to see all combinations of values simply remove the filter:



The screenshot shows the Fuzzer application interface. The 'Request' tab is active, displaying the following details:

```
Request:
GET http://192.168.178.196/mutillidae/%AAA%/%BBB%?page=browser-info.php HTTP/1.1
Host: 192.168.178.196
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.6) Gecko/20100625 Firefox/3.6.6
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Proxy-Connection: keep-alive
Referer: http://192.168.178.196/mutillidae/index.php?page=view-someones-blog.php
Cookie: uid=3
```

Below the request, there is a 'Start' button and 'Request Options' with the following checked items:

- Update Content-Length
- Update Session Information
- Run Login

The 'Results' tab is also visible, showing a table of matches:

Taq/Value	Match
AAA=xxx BBB=mp3	HTTP/1.1 404 Not Found Date: Fri, 23 Jul 2010 09:47:47 GMT S
AAA=xxx BBB=php	HTTP/1.1 404 Not Found Date: Fri, 23 Jul 2010 09:47:47 GMT S
AAA=xxx BBB=wav	HTTP/1.1 404 Not Found Date: Fri, 23 Jul 2010 09:47:46 GMT S
AAA=test BBB=wav	HTTP/1.1 404 Not Found Date: Fri, 23 Jul 2010 09:47:47 GMT S
AAA=index BBB=wav	HTTP/1.1 404 Not Found Date: Fri, 23 Jul 2010 09:47:47 GMT S
AAA=test BBB=mp3	HTTP/1.1 404 Not Found Date: Fri, 23 Jul 2010 09:47:47 GMT S
AAA=test BBB=php	HTTP/1.1 404 Not Found Date: Fri, 23 Jul 2010 09:47:47 GMT S
AAA=index BBB=mp3	HTTP/1.1 404 Not Found Date: Fri, 23 Jul 2010 09:47:47 GMT S
AAA=index BBB=php	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 09:47:47 GMT Server: A

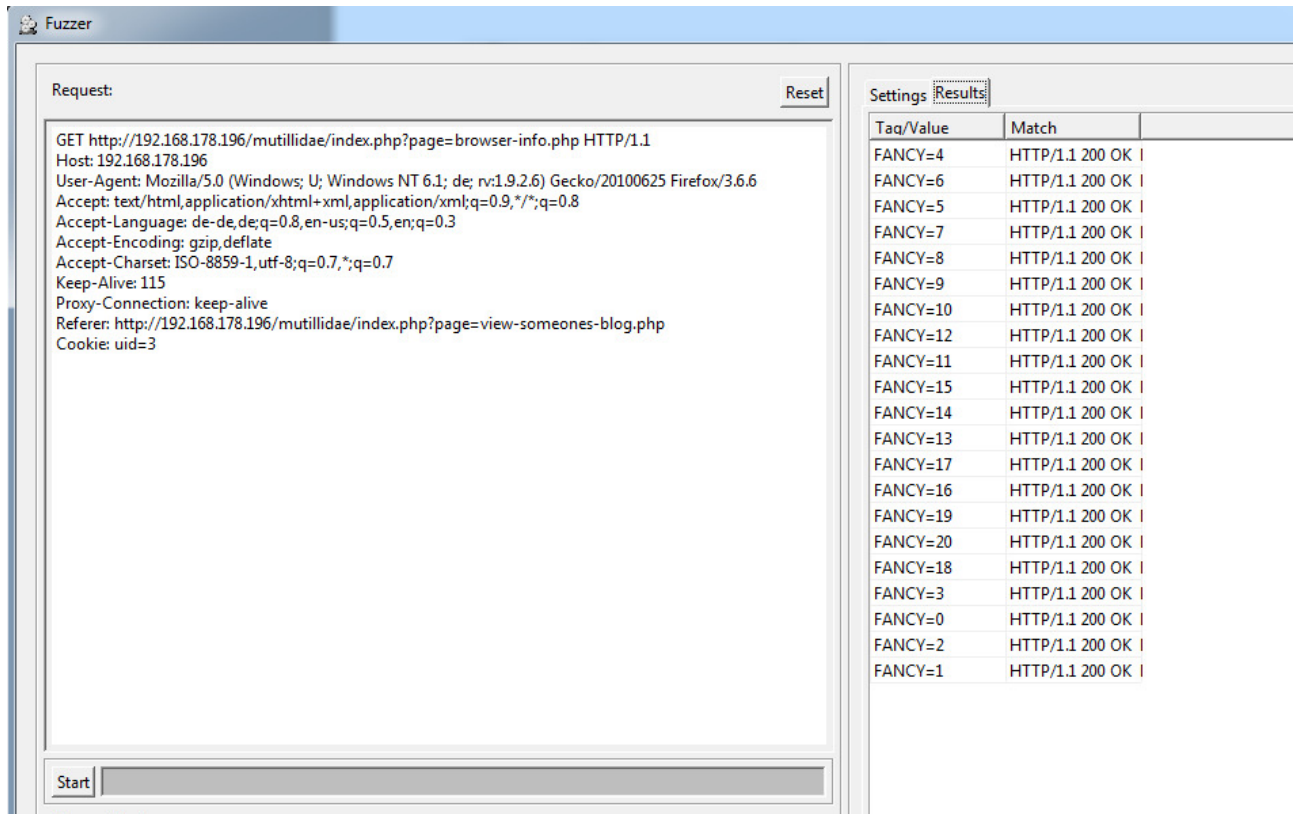
At the bottom, the 'Logs' section shows the following entries:

```
07/23/2010 @ 11:47:47: Finished Fuzzing
07/23/2010 @ 11:47:47: Started Fuzzing (6 Requests)
07/23/2010 @ 11:47:47: Prepare Fuzzing: Generators
07/23/2010 @ 11:45:56: Finished Fuzzing
```

## - Generating complex values -

Here we only generate more complex values without really fuzzing the web application so we neither place a tag at the request nor we need to define a filter.

First we create a tag called 'FANCY' and a simple generator which produces the values 0,1,2,.....20 and start the fuzzing process. In the Result tab we can see our values:



The screenshot shows the Fuzzer application interface. On the left, the 'Request' tab displays the following HTTP request:

```
GET http://192.168.178.196/mutillidae/index.php?page=browser-info.php HTTP/1.1
Host: 192.168.178.196
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.6) Gecko/20100625 Firefox/3.6.6
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Proxy-Connection: keep-alive
Referer: http://192.168.178.196/mutillidae/index.php?page=view-someones-blog.php
Cookie: uid=3
```

On the right, the 'Results' tab shows a table of generated values and their corresponding HTTP responses:

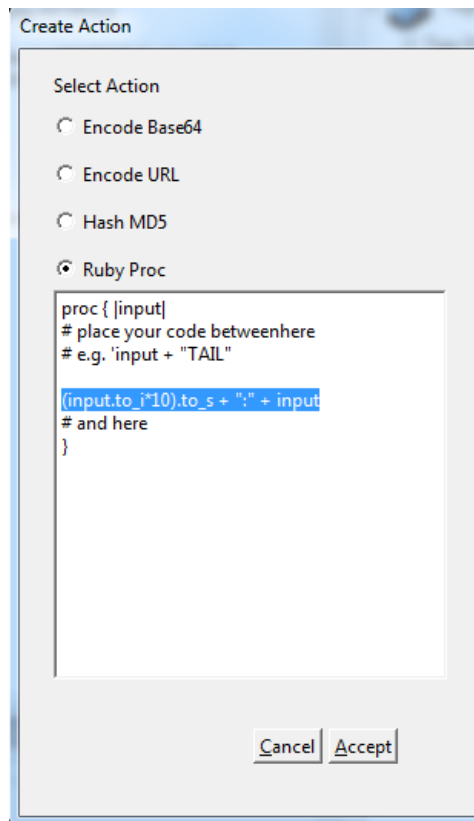
Tag/Value	Match
FANCY=4	HTTP/1.1 200 OK
FANCY=6	HTTP/1.1 200 OK
FANCY=5	HTTP/1.1 200 OK
FANCY=7	HTTP/1.1 200 OK
FANCY=8	HTTP/1.1 200 OK
FANCY=9	HTTP/1.1 200 OK
FANCY=10	HTTP/1.1 200 OK
FANCY=12	HTTP/1.1 200 OK
FANCY=11	HTTP/1.1 200 OK
FANCY=15	HTTP/1.1 200 OK
FANCY=14	HTTP/1.1 200 OK
FANCY=13	HTTP/1.1 200 OK
FANCY=17	HTTP/1.1 200 OK
FANCY=16	HTTP/1.1 200 OK
FANCY=19	HTTP/1.1 200 OK
FANCY=20	HTTP/1.1 200 OK
FANCY=18	HTTP/1.1 200 OK
FANCY=3	HTTP/1.1 200 OK
FANCY=0	HTTP/1.1 200 OK
FANCY=2	HTTP/1.1 200 OK
FANCY=1	HTTP/1.1 200 OK

Next we work on the values we get from the generator (input). We want to build values like “<input\*10>:<input>”

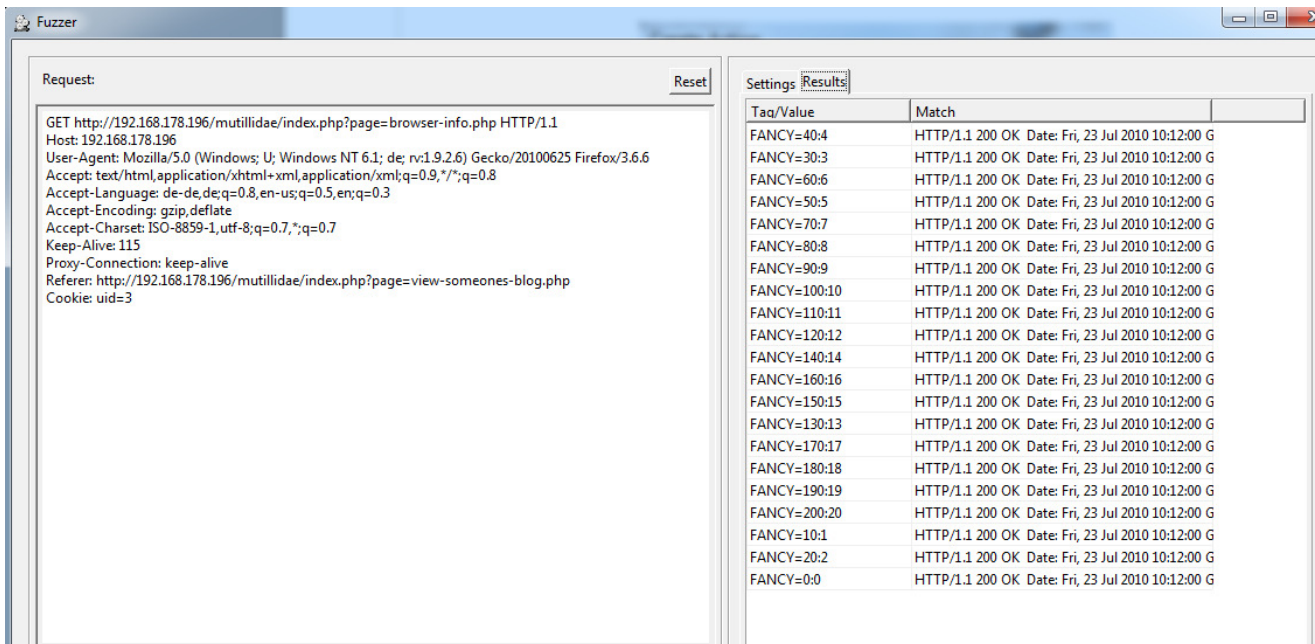
For example for *input=3* we want the resulting value “30:3”. Therefore right-click **Counter** and choose **Add Action**, select “**Ruby Proc**” and add the following line of ruby code:

**(input.to\_i\*10).to\_s + ":" + input**

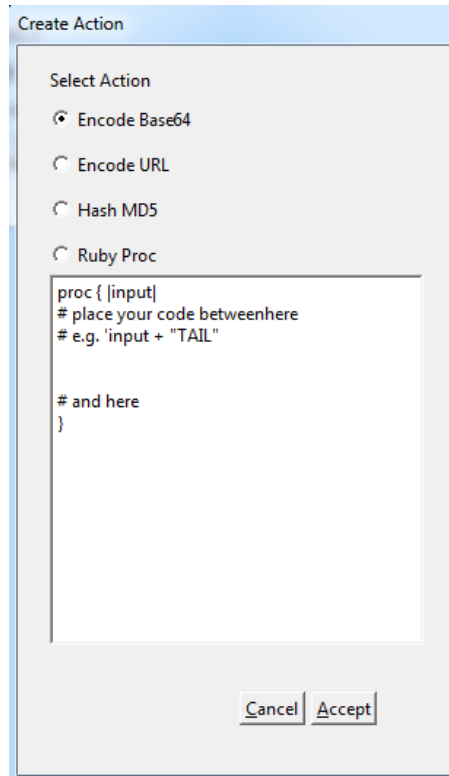




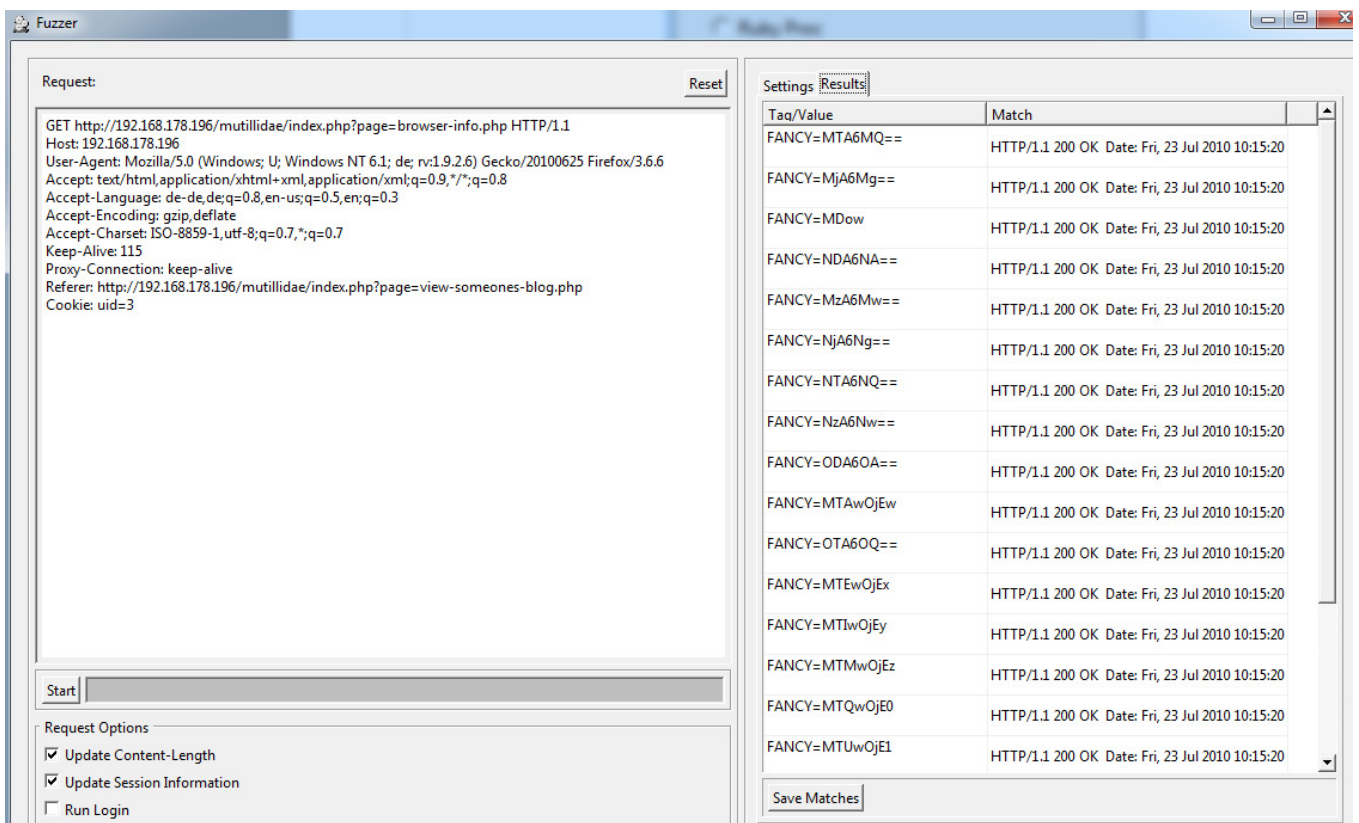
→ Start fuzzing and check the results:



In the next step we want to base64 encode this value by simply adding another action:



→ Start fuzzing and check the results:



In the final step the value should look like this:

**"WATOBO:<base64>:pwned"**

We create another action by adding the following ruby code:

**"WATOBO:" + input.strip + ":pwned"**

Create Action

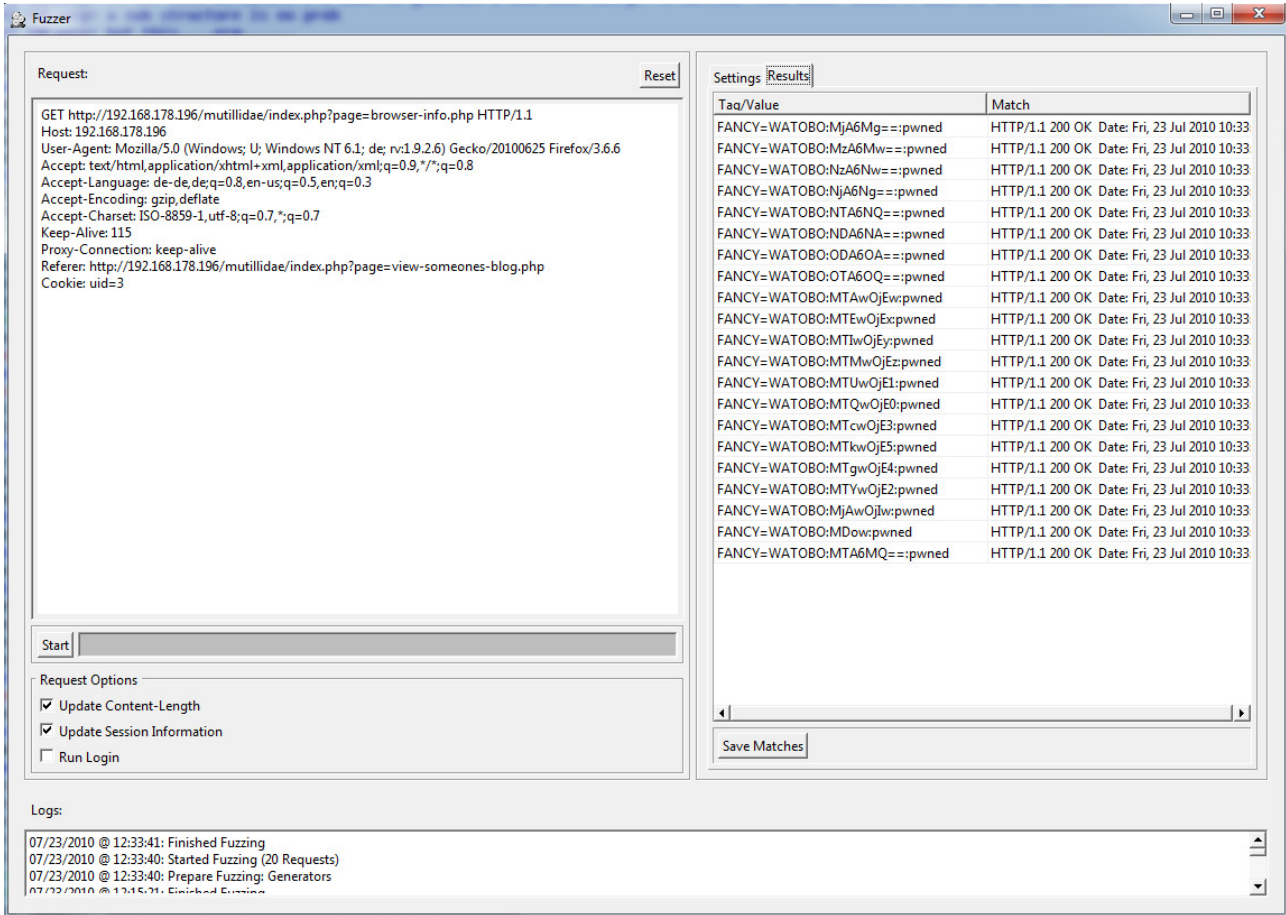
Select Action

- Encode Base64
- Encode URL
- Hash MD5
- Ruby Proc

```
proc { |input|  
# place your code betweenhere  
# e.g. 'input + "TAIL"  
"WATOBO:" + input.strip + ":pwned"  
# and here  
}
```

Cancel Accept

→ Start fuzzing and check the results:



The screenshot shows the Fuzzer application window. On the left, the 'Request' tab is active, displaying an HTTP GET request to a target URL with various headers. Below the request is a 'Start' button and 'Request Options' including 'Update Content-Length', 'Update Session Information', and 'Run Login'. At the bottom left is a 'Logs' section with a scrollable list of events.

On the right, the 'Results' tab is active, showing a table of detected matches. The table has two columns: 'Taa/Value' and 'Match'. All matches are reported as 'HTTP/1.1 200 OK' with a date of 'Fri, 23 Jul 2010 10:33'. The values are various alphanumeric strings.

Taa/Value	Match
FANCY=WATOBO:MjA6Mg==:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MzA6Mw==:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:NzA6Nw==:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:NjA6Ng==:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:NTA6NQ==:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:NDA6NA==:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:ODA6OA==:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:OTA6OQ==:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTAwOjEw:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTAwOjEx:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTAwOjEy:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTAwOjEz:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTAwOjE1:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTAwOjE0:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTcwOjE3:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTkwOjE5:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTgwOjE4:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTYwOjE2:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MjAwOjIw:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MDow:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33
FANCY=WATOBO:MTA6MQ==:pwned	HTTP/1.1 200 OK Date: Fri, 23 Jul 2010 10:33

At the bottom right of the results table is a 'Save Matches' button. The logs at the bottom left show the following entries:

```

07/23/2010 @ 12:33:41: Finished Fuzzing
07/23/2010 @ 12:33:40: Started Fuzzing (20 Requests)
07/23/2010 @ 12:33:40: Prepare Fuzzing: Generators
07/23/2010 @ 12:33:39: Finished Fuzzing
  
```

Perfect !!!!!

## 4. Conclusion

WATOBO is a really awesome tool which doesn't need an installation and can be quickly adapted to new requirements. I think the semi-automated approach of WATOBO is the best way to perform an accurate audit and to identify most of the vulnerabilities.

The session management feature is totally leet and rarely found in free tools of this genre. Most of the functions are self explanatory and easy to perform which makes WATOBO an important tool in the pentester's arsenal. Since it's written in ruby you can add your own checks.

The implemented fuzzer is very valuable in exploring a web application and finding more information and vulnerabilities.

All these great features and functions make WATOBO one of the top free web assessment tools.

## 5. References

- [1] WATOBO Homepage (by Siberas)  
[http://sourceforge.net/apps/mediawiki/watobo/index.php?title=Main\\_Page](http://sourceforge.net/apps/mediawiki/watobo/index.php?title=Main_Page)
  
- [2] Mutillidae by Irongeek  
<http://www.irongeek.com/i.php?page=security/mutillidae-deliberately-vulnerable-php-owasp-top-10>
  
- [3] Damn Vulnerable Web App  
<http://www.dvwa.co.uk/>