Burp Suite Tutorial – Repeater and Comparer Tools

March 19, 2010 | Written by Security Ninja | Application Security | 7 Comments

Hi everyone,

I was very happy to see that a lot of people liked the Burp Suite Tutorial (Intruder Tool) blog post last week. I plan to publish more tutorials for the Burp Suite and this week I will be covering the Repeater and Comparer tools.

What are the Repeater and Comparer tools?

The Burp Suite is made up of multiple tools and today we will be taking a look at the Repeater and Comparer tools (descriptions take from the Port Swigger website):

Repeater: Burp Repeater is a tool for manually modifying and reissuing individual HTTP requests, and analysing their responses. It is best used in conjunction with the other Burp Suite tools. For example, you can send a request to Repeater from the target site map, from the Burp Proxy browsing history, or from the results of a Burp Intruder attack, and manually adjust the request to fine-tune an attack or probe for vulnerabilities.

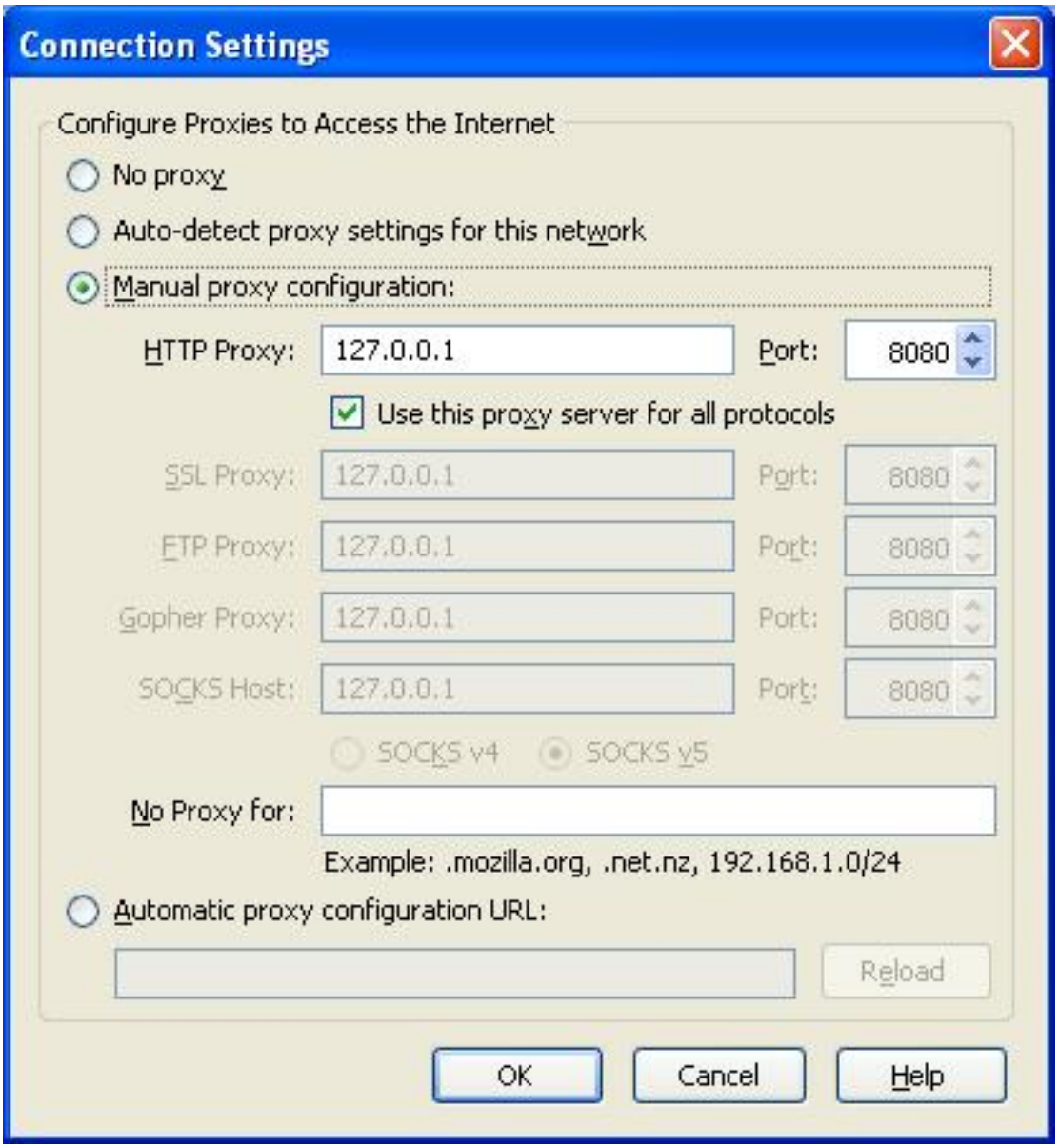
Comparer: Burp Comparer is a simple tool for performing a comparison (a visual "diff") between any two items of data. In the context of attacking a web application, this requirement will typically arise when you want to quickly identify the differences between two application responses (for example, between two responses received in the course of a Burp Intruder attack, or between responses to a failed login using valid and invalid usernames), or between two application requests (for example, to identify the different request parameters that give rise to different behaviour).

I explained how to use the Intruder tool in the last blog post and I will show you how to use the Repeater and Comparer tools this week. We will run the same Intruder tests as last week and show you how you can use the Repeater and Comparer tools to perform additional testing based on the intruder test results. The first part of the tutorial is very similar to the Intruder tool because we will use the results from the intruder test to demonstrate the Repeater and Comparer tools.

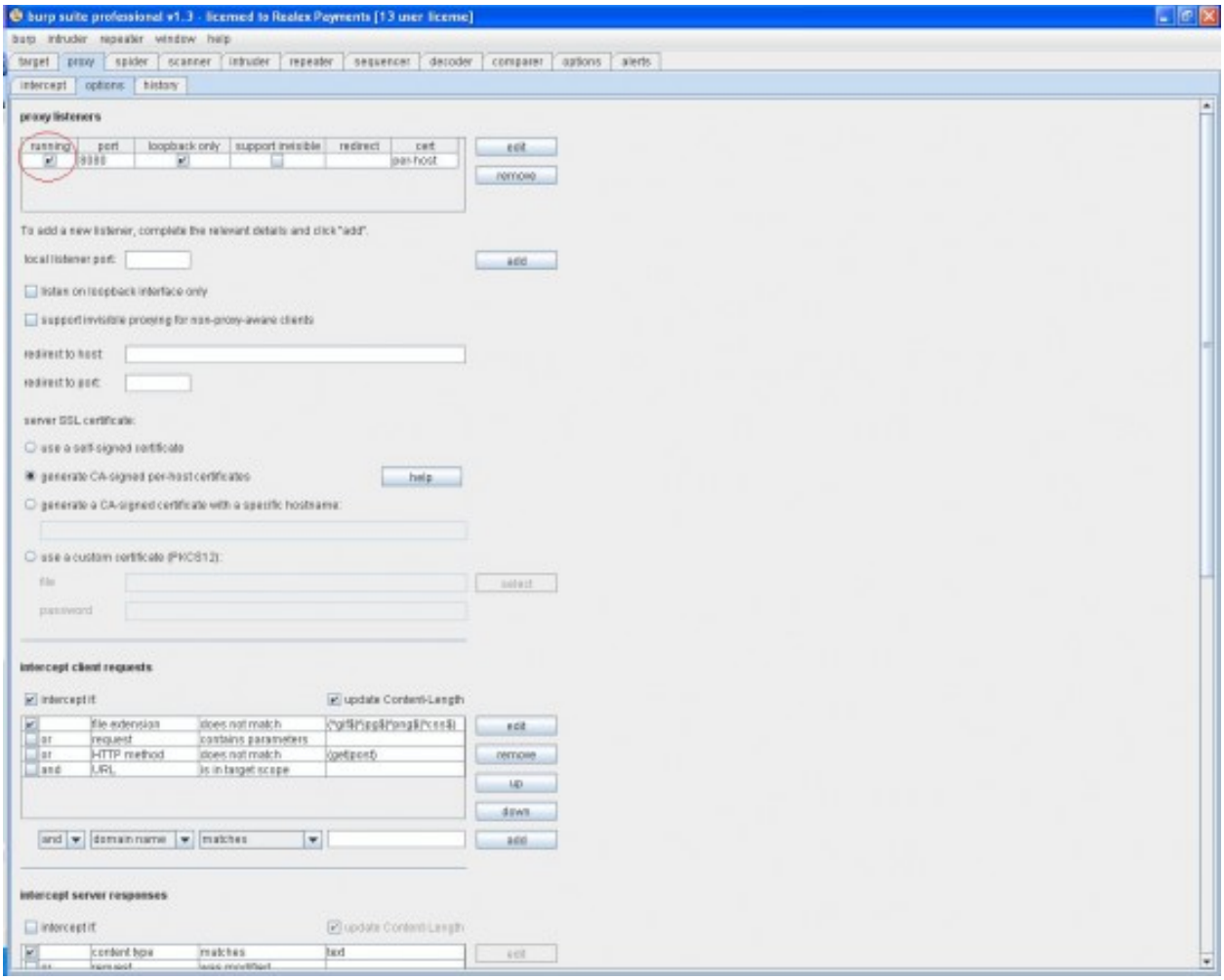
Enabling the Burp Suite Proxy

To begin using the Burp Suite to test our example web application we need configure our web browser to use the Burp Suite as a proxy. The Burp Suite proxy will use port 8080 by default but you can change this if you want to.

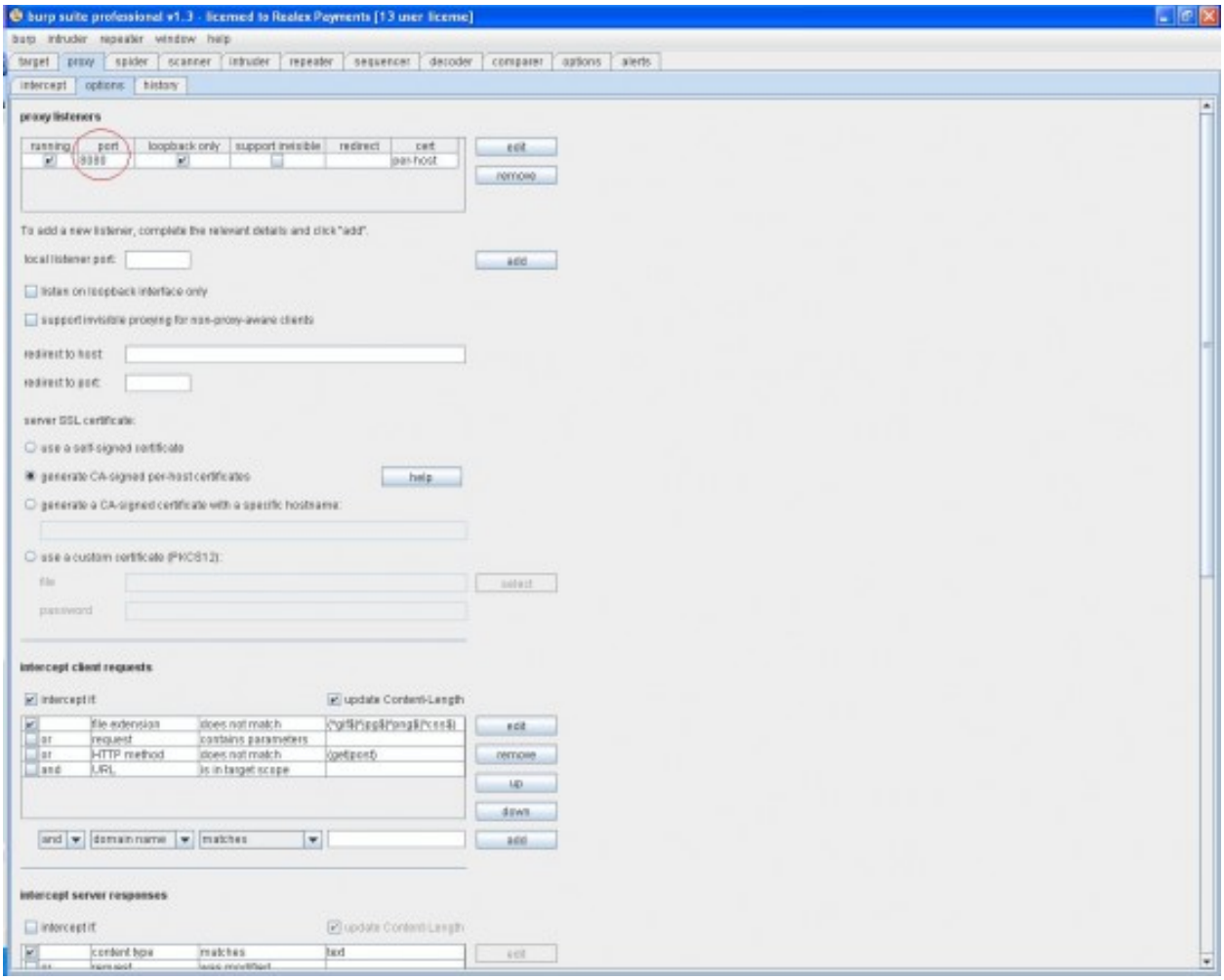
You can see in the image below that I have configured Firefox to use the Burp Suite proxy for all traffic.



When you open the Burp Suite go to the proxy tool and check that the proxy is running by clicking on the options tab:

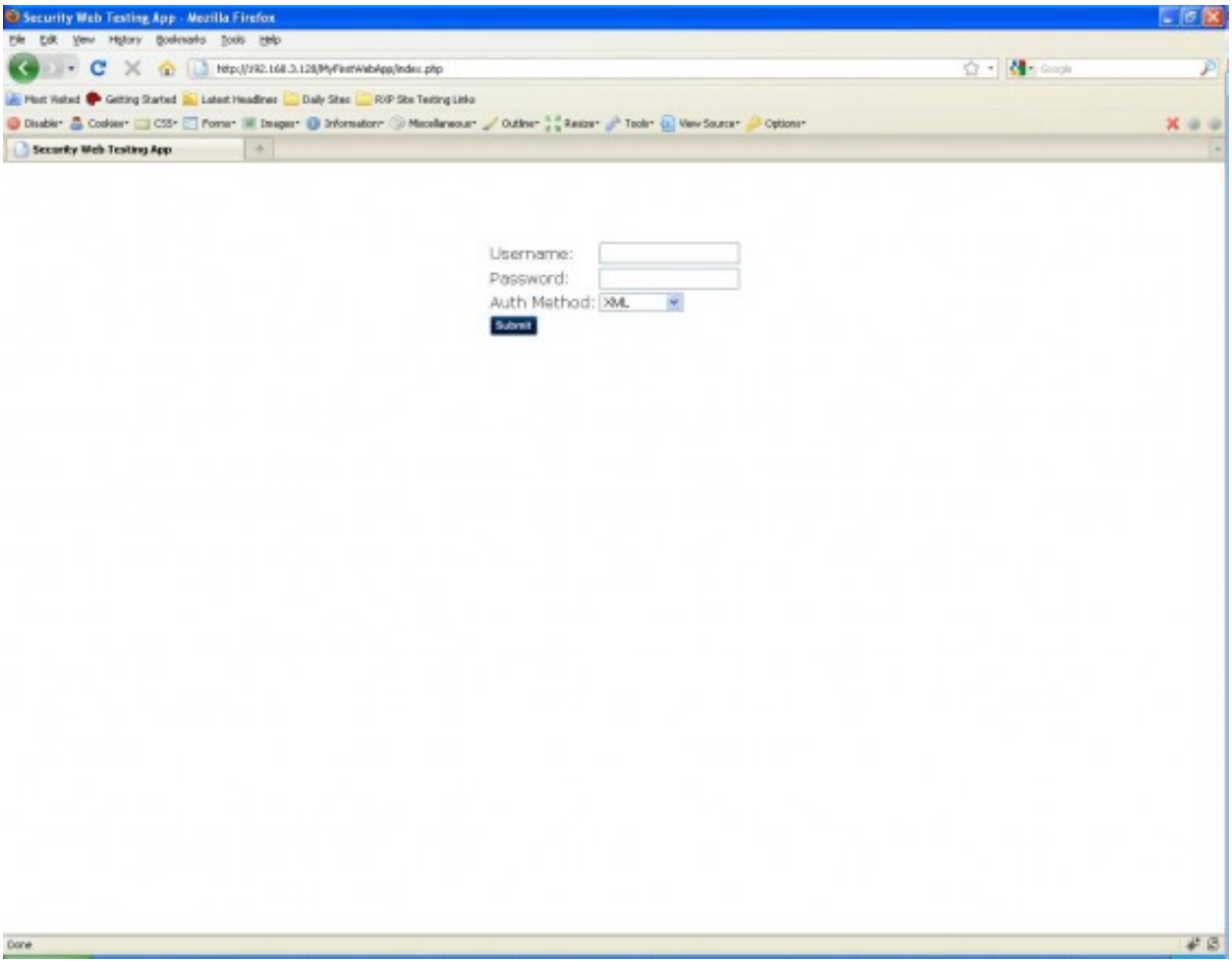


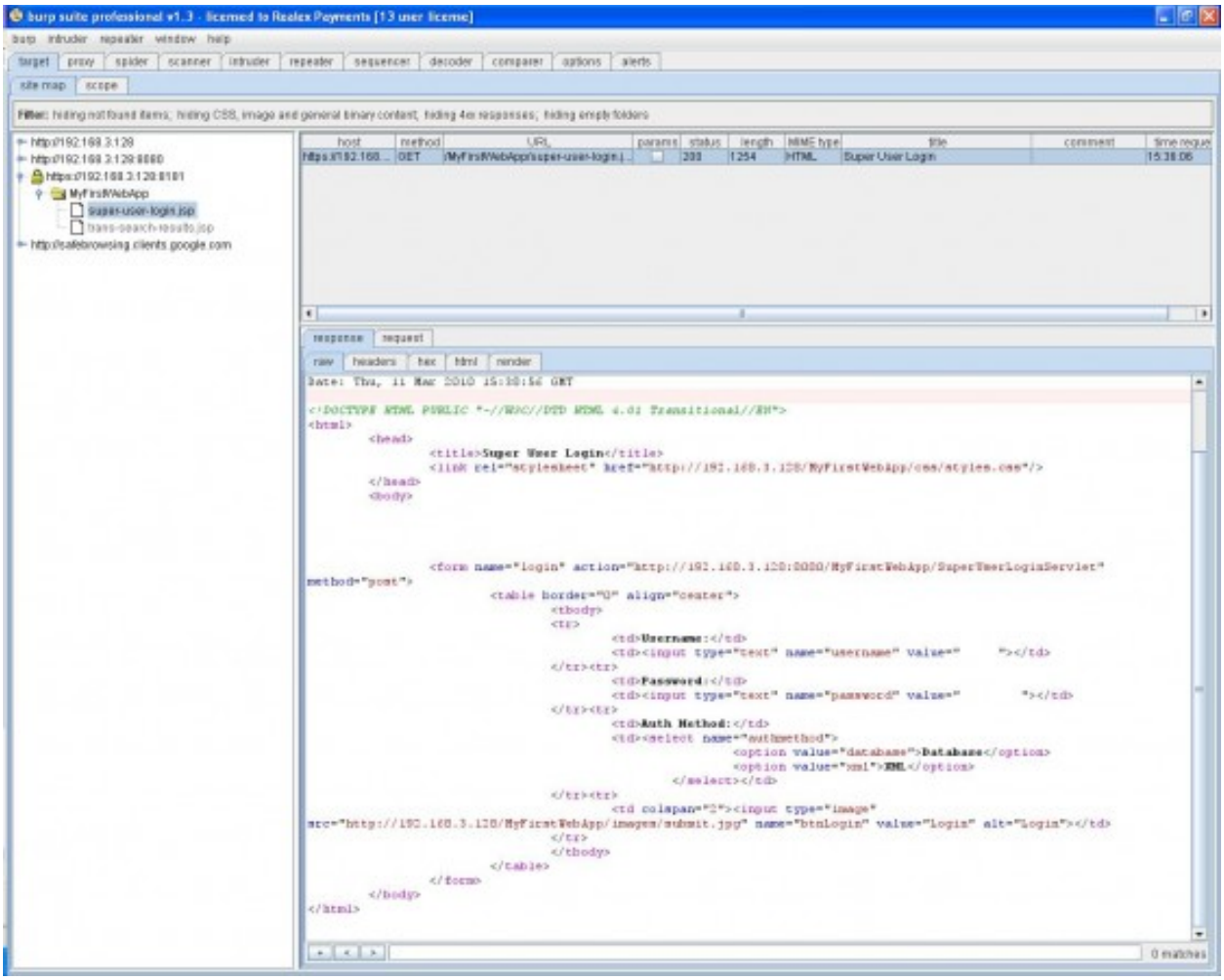
You can see that the proxy is using the default port:



The proxy is now running and ready to use. You can see that the proxy options tab has quite a few items that we can configure to meet our testing needs. A lot of these items are out side of the scope of this tutorial.

The Burp Suite will now begin logging the requests and responses that pass through the proxy. We have browsed to the logon page of our example application and the Burp Suite proxy has captured the request and response:

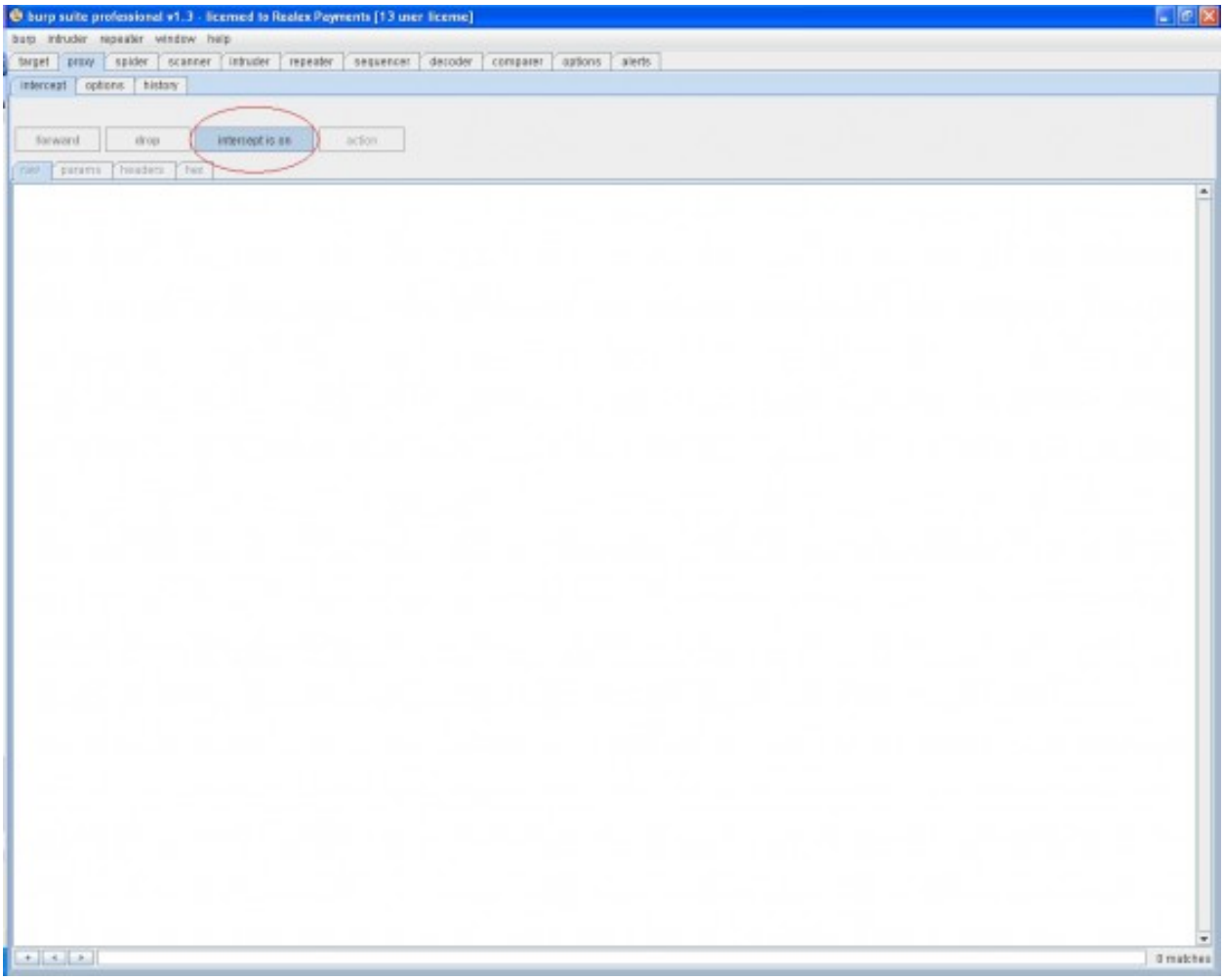

Security Web Testing App - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Getting Started Latest Headlines Daily Sites RVP Site Testing Links
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Security Web Testing App
Username:
Password:
Auth Method: XML
Submit
Done



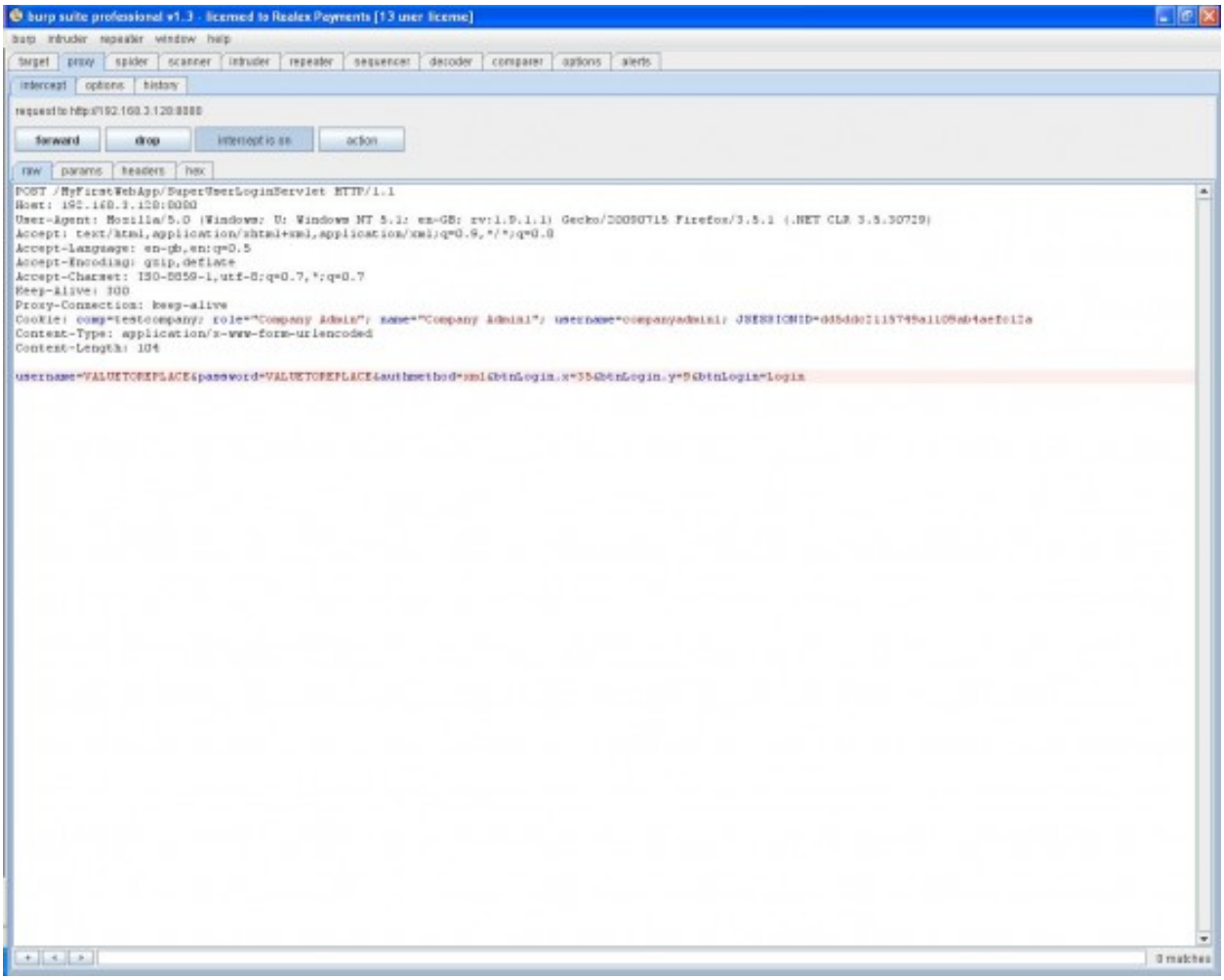
The logon page is very simple, we have two input points that we need to test for input validation weaknesses.

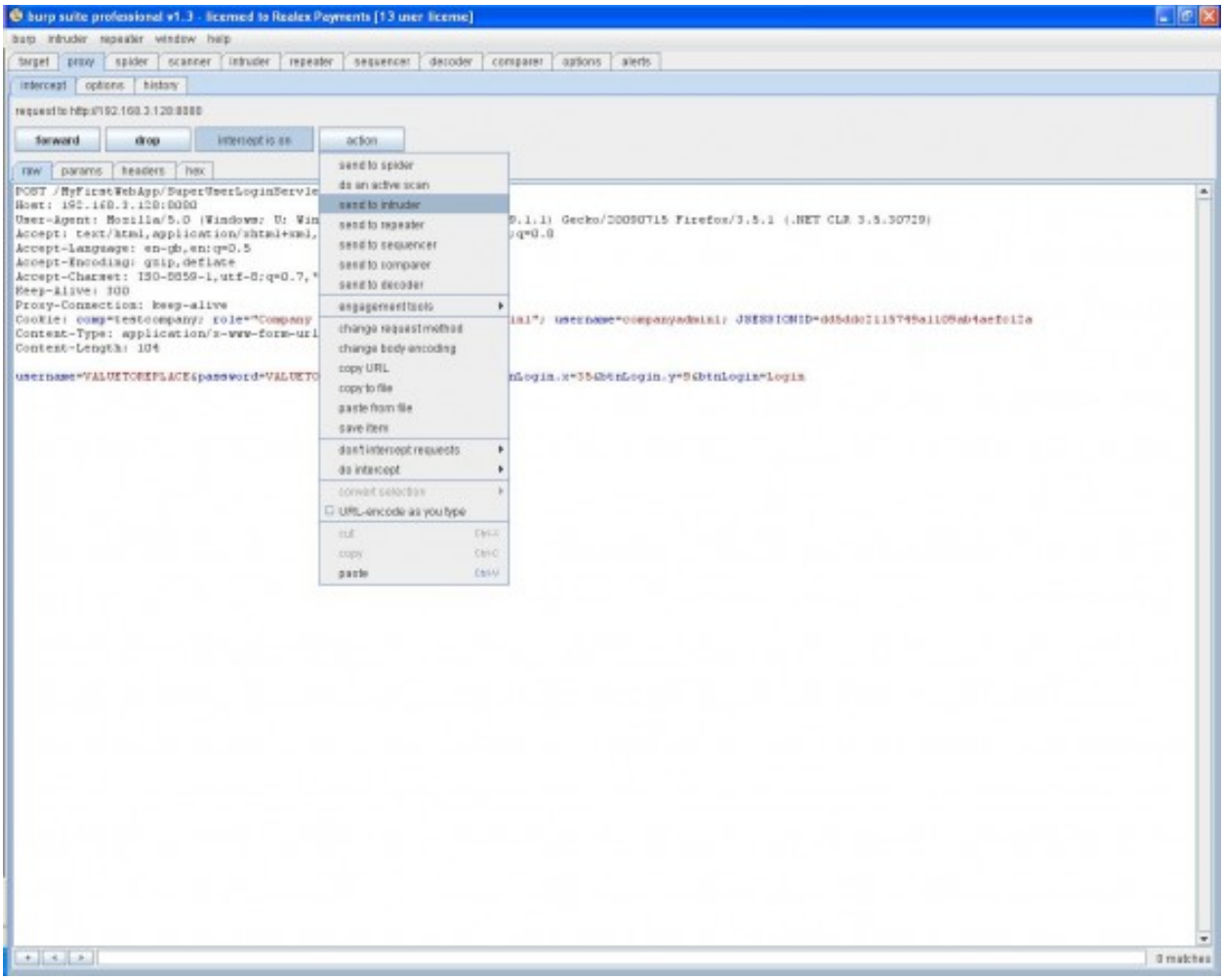
We need to capture a logon request and replace the username and password values with our test inputs.

To do this we must ensure that the Burp Suite proxy is configured to intercept our requests:

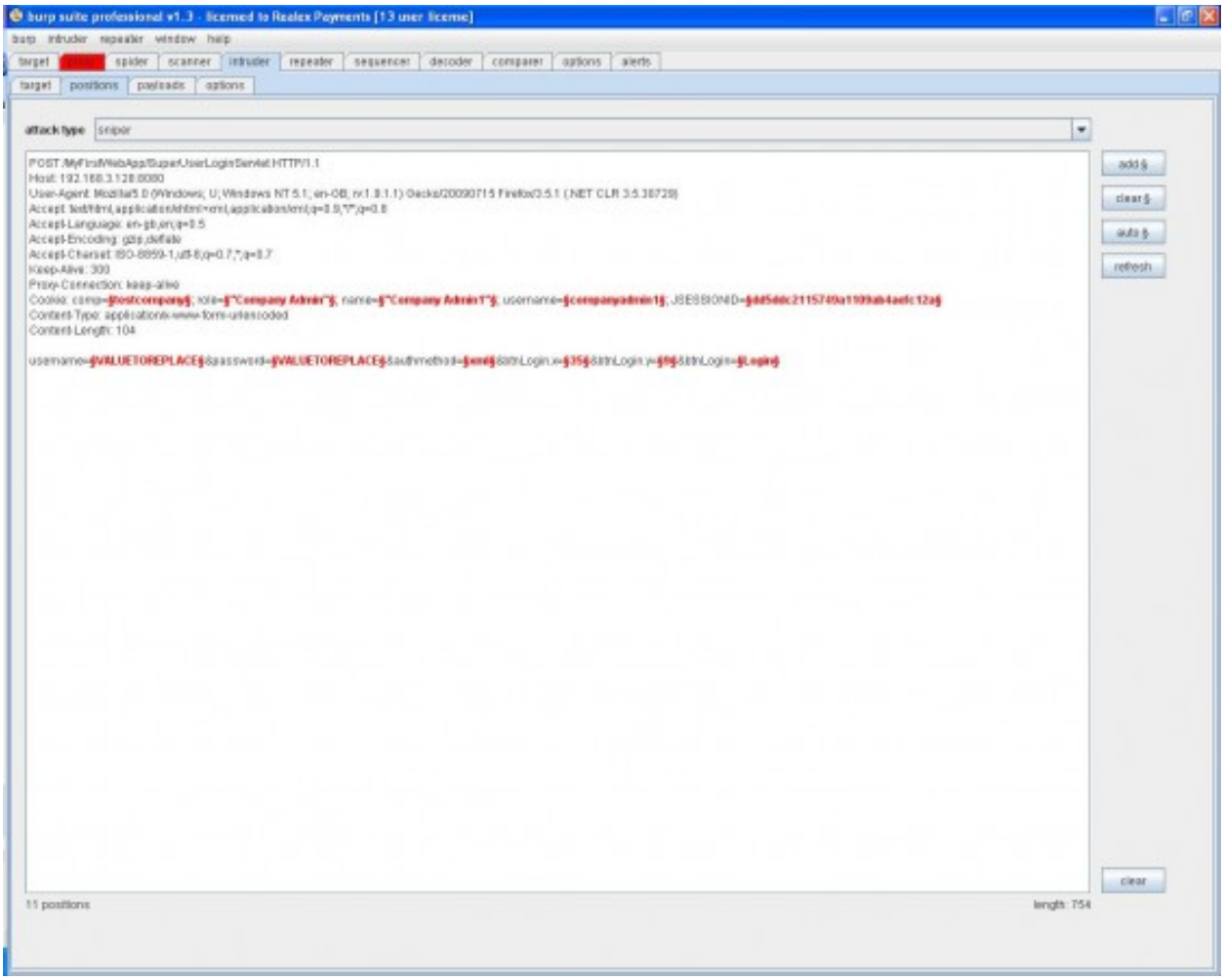


With the intercept enabled we will submit the logon form and send it to the intruder as you can see below:


burp suite professional v1.3 - licensed to Realex Payments [13 user license]
burp intruder repeater window help
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
intercept options history
request to http://192.168.3.120:8888
forward drop intercept is on action
raw params headers hex
POST /MyFirstWebApp/SuperUserLoginServlet HTTP/1.1
Host: 192.168.3.120:8080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.1.1) Gecko/20090715 Firefox/3.5.1 (.NET CLR 3.5.30729)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Cookie: comp=testcompany; role="Company Admin"; name="Company Admin"; username=companyadmin; JSESSIONID=dd5ddc1115795a1109ab4aefc12a
Content-Type: application/x-www-form-urlencoded
Content-Length: 104
username=VALUETOREPLACE&password=VALUETOREPLACE&authmethod=xml&btnLogin.x=35&btnLogin.y=5&btnLogin=Login
0 matches



The Burp Suite will send our request to the intruder tool so we can begin our testing. You can see the request in the intruder tool below:

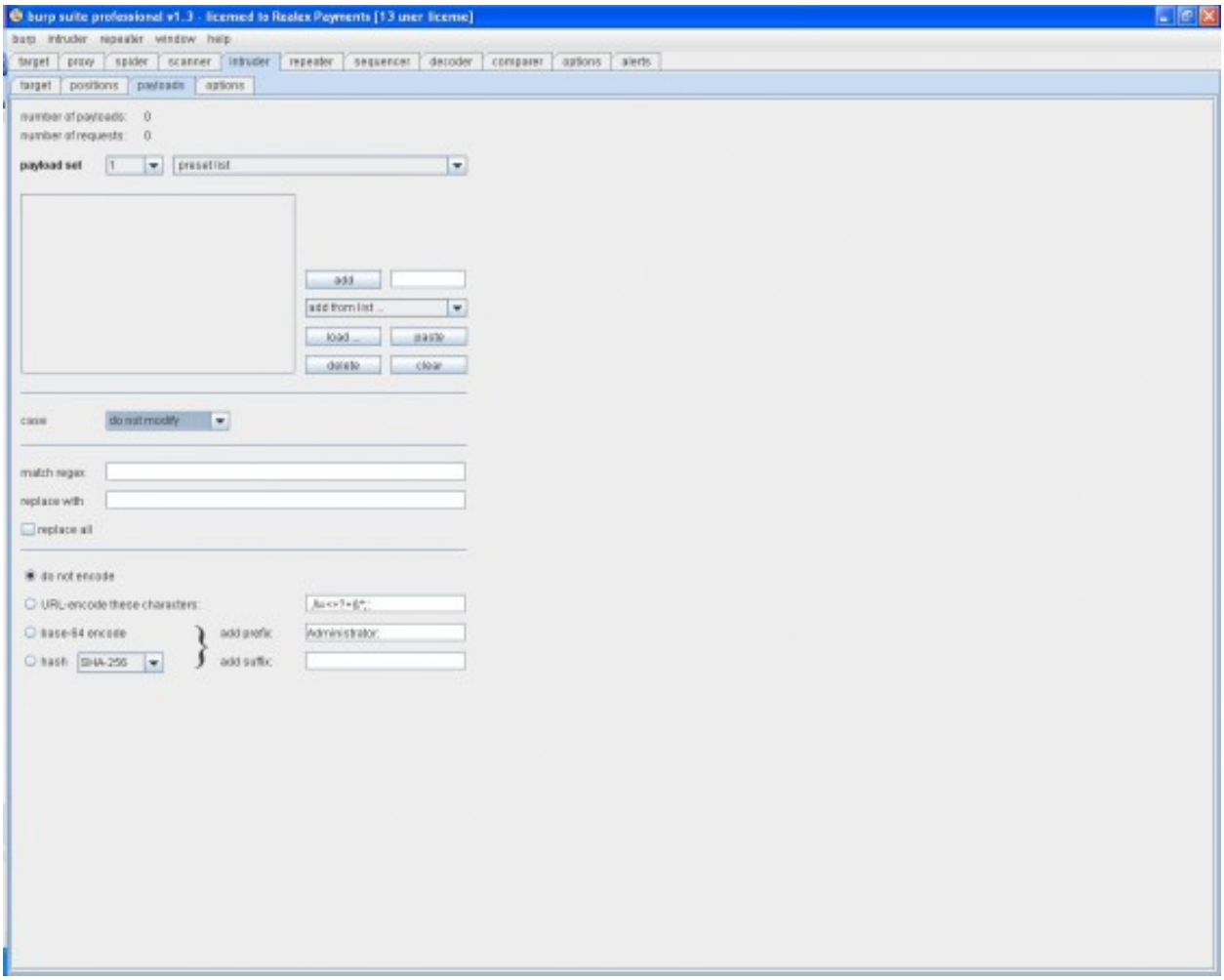


The tool has automatically created payload positions for us. The payload positions are defined using the § character, the intruder will replace the value between two § characters with one of our test inputs.

Selecting a payload

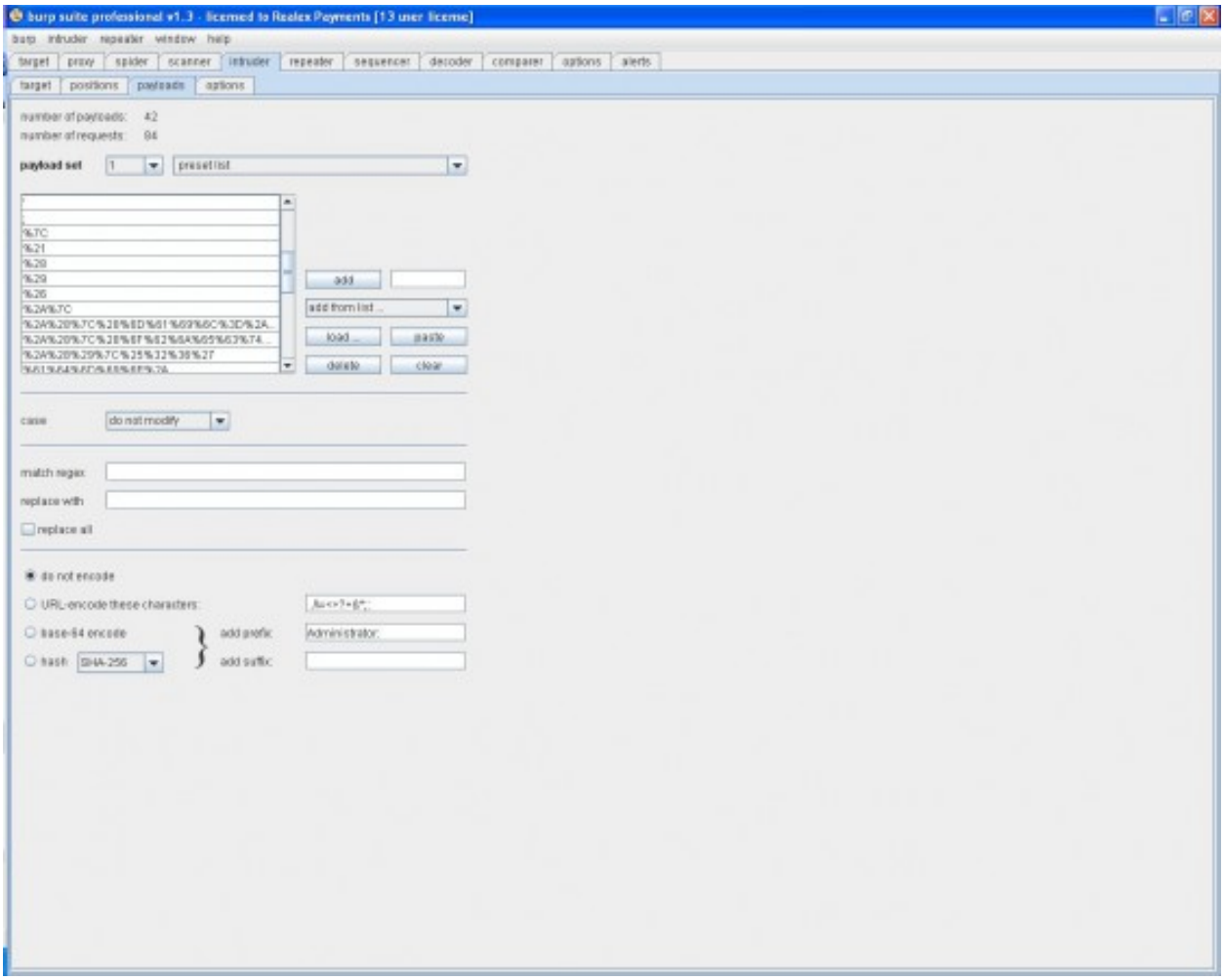
So far we have enabled the Burp Suite proxy, captured a request, sent it to the intruder tool and marked our payload positions. We now need to tell the intruder tool what values to insert into the positions.

To define our payloads we need to click on the payloads tab within the intruder tool:



We are going to use the sniper attack type in this example so we only have one payload set. The dropdown menu next to the payload set number allows you perform many different types of testing/data manipulation.

We are going to use a predefined list of inputs for our testing:



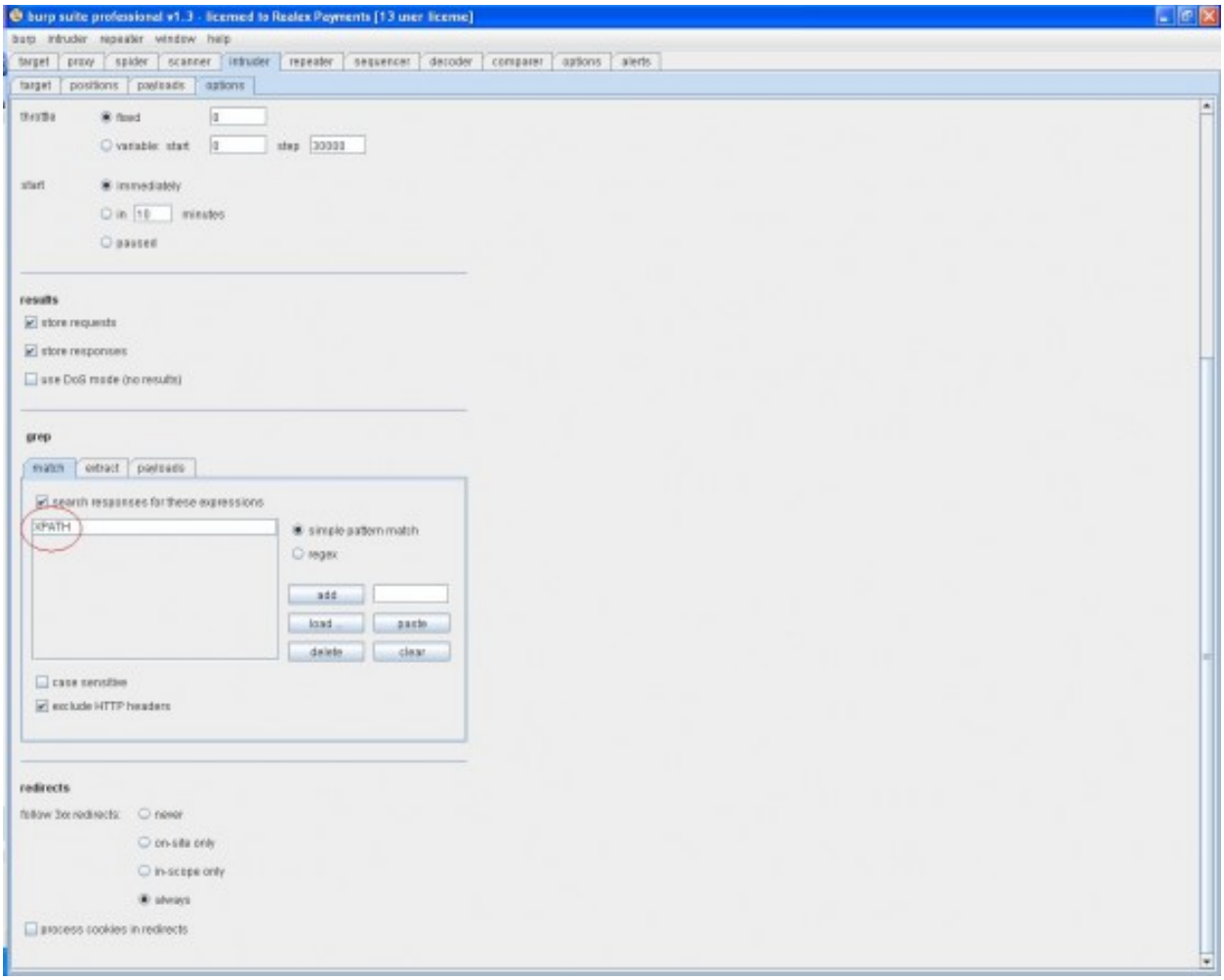
Configuring intruder options

The options tab in the intruder tool allows you to configure additional test parameters including a grep function, enabling a DOS mode and deciding how you want the Burp Suite to handle 3xx redirects.

For this example we are going to the grep function to perform a simple pattern match. The Burp Suite will provide you with a default list of words to match against but we are going to remove these and add our own for this test.

The logon form that we are going to test uses an XML file containing usernames and passwords to authenticate our users. The username and password values we provide will be entered into an XPATH query. If the application fails to securely validate these values an XPATH Injection vulnerability could exist.

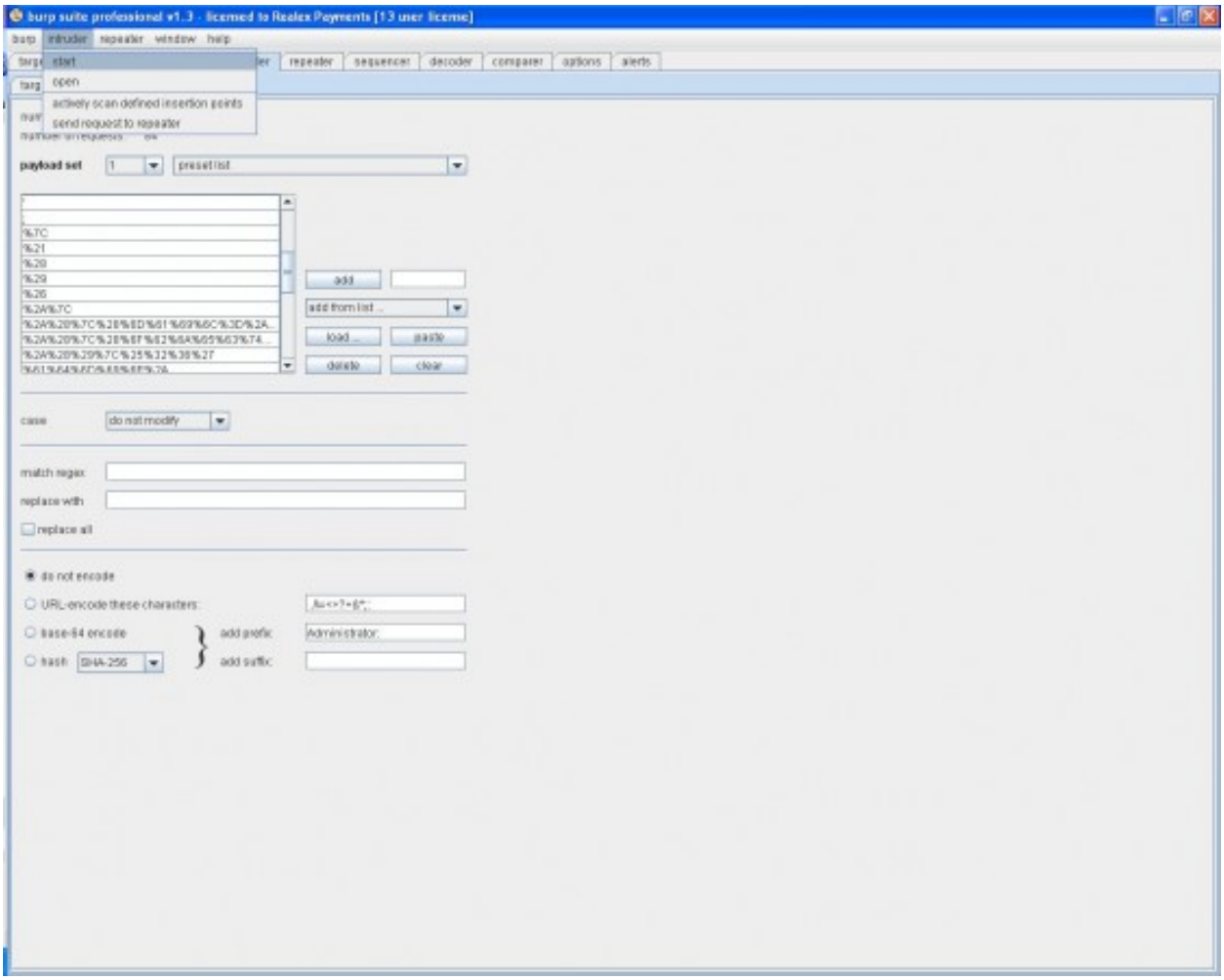
We want the intruder tool perform a pattern match for us using the word “XPATH”. This should allow us to easily identify any XPATH errors caused by our test inputs:



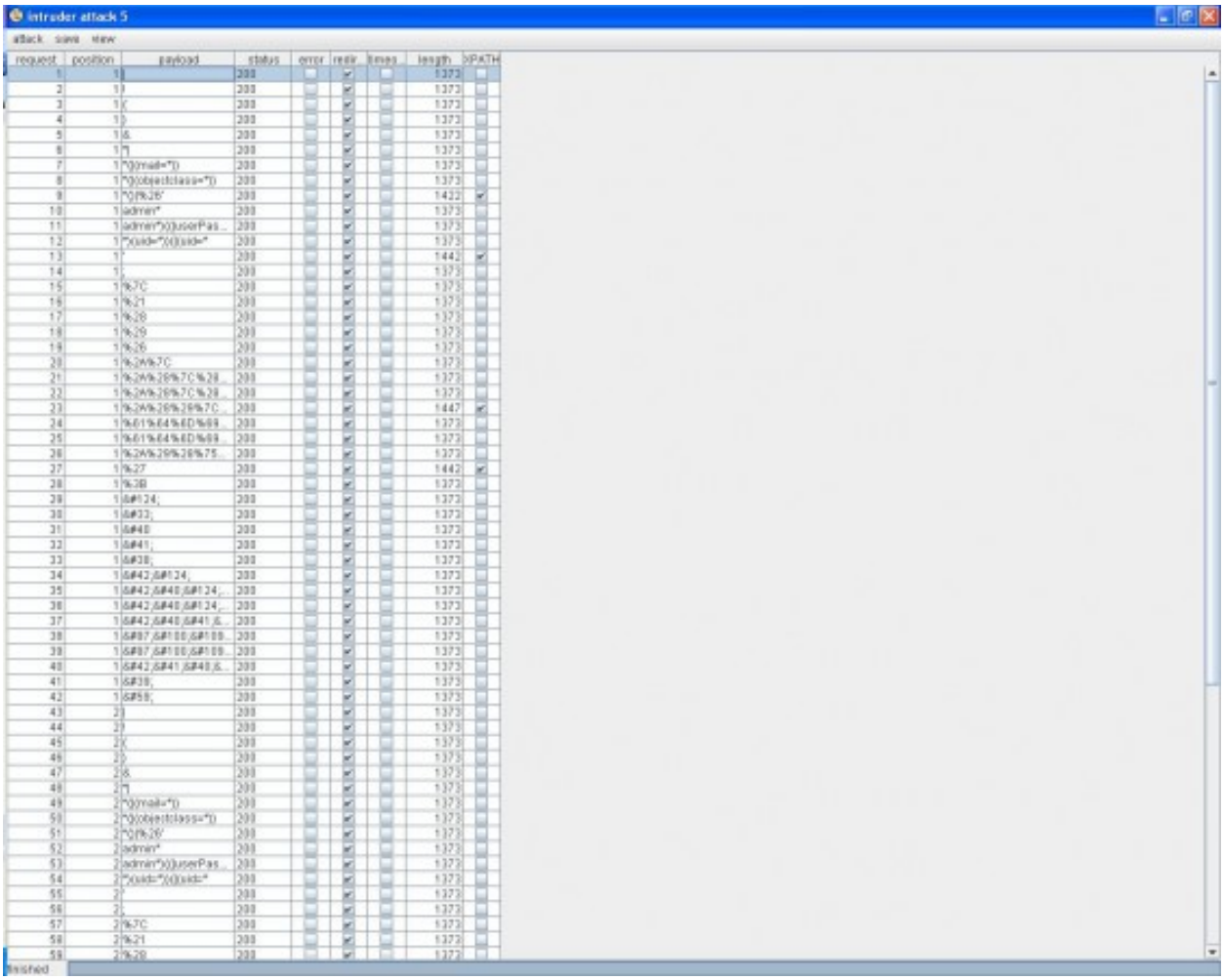
Executing our tests

The only thing left for us to do now is to execute our tests.

To start the intruder tool you need to click “intruder” on the top menu and then click on “start”:



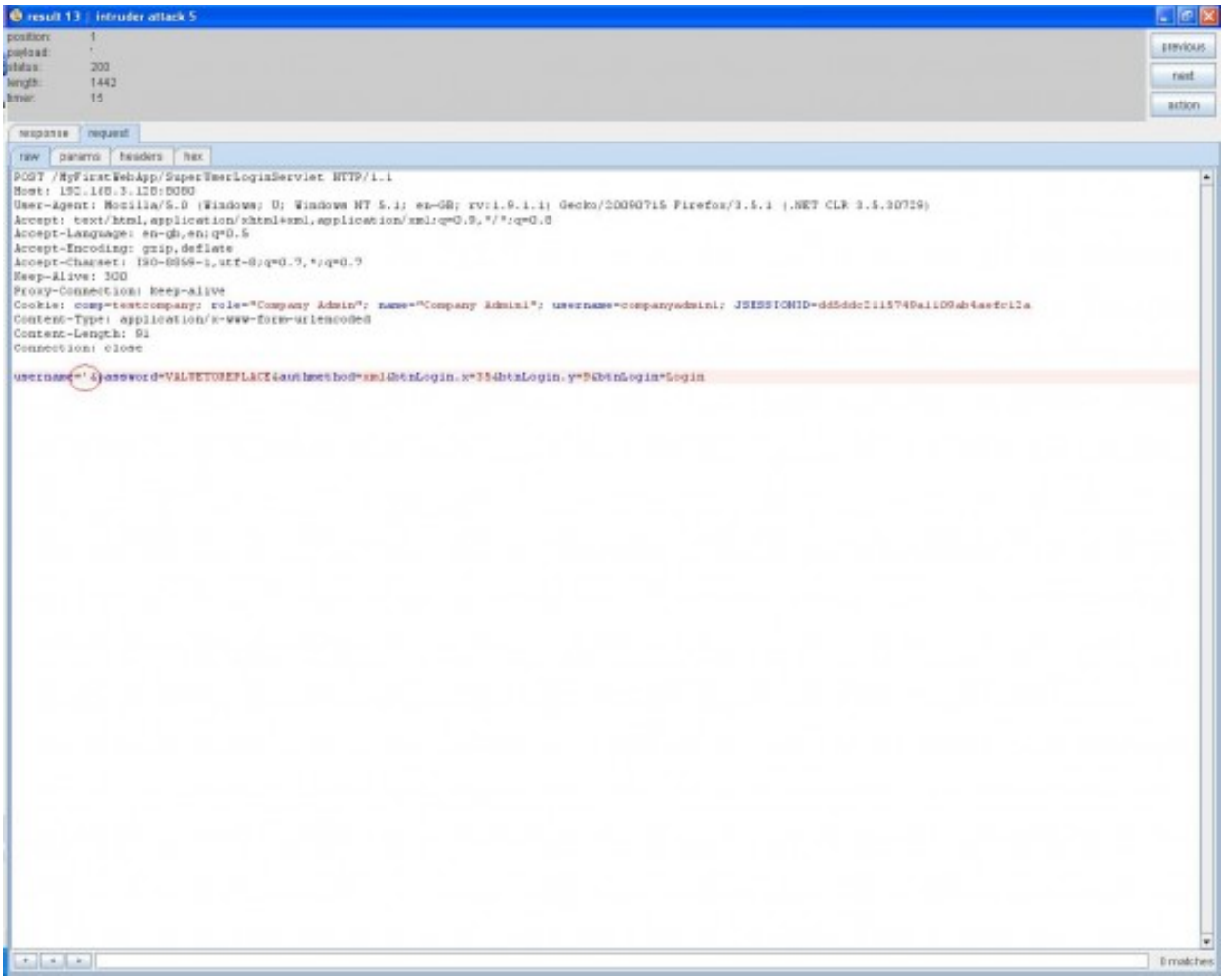
A separate window will be opened which will show you each test, the payload used, the status code, length and in our case the tests which match our XPATH pattern match word:



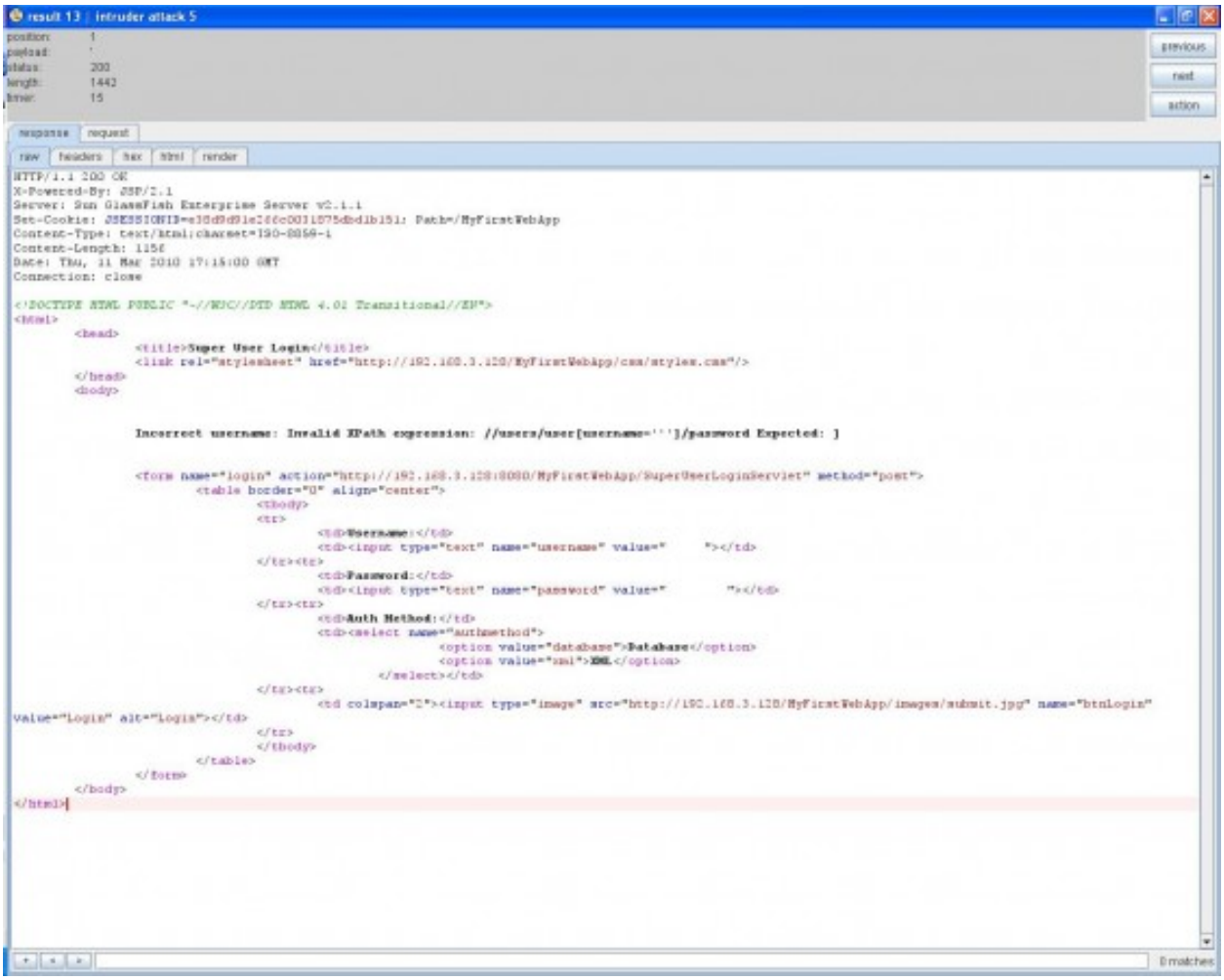
You can see that some of the tests matched our pattern match word by looking at the tick boxes in the XPATH column.

To review the request and the response for each test you can double click any of the requests shown in the attack window.

We are going to review request 13 which entered a single quote (‘) into the username field:



This single quote caused an XPATH error to be thrown by the application:

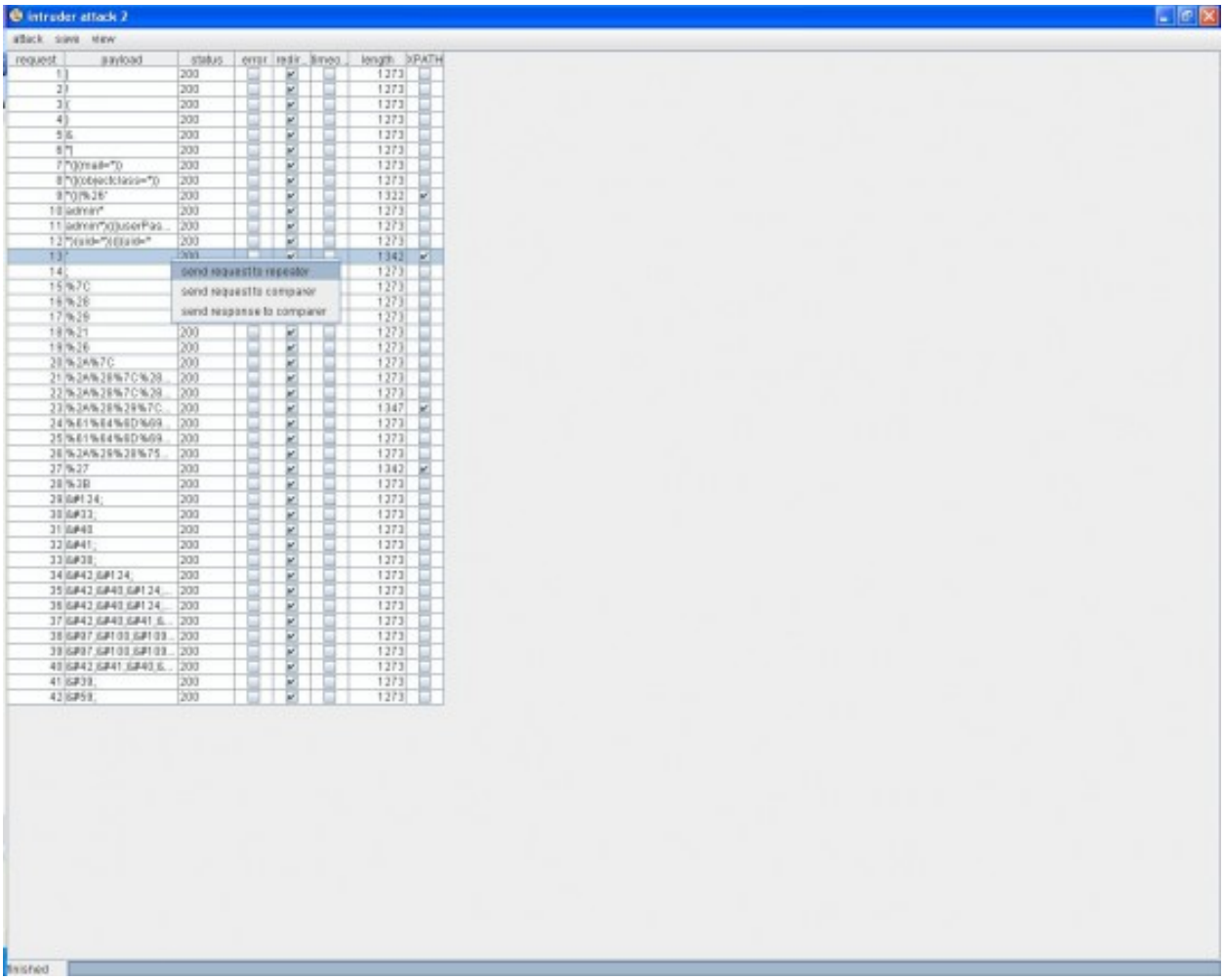


So as you can see we have used the Burp Suite intruder tool to identify a potential XPATH injection vulnerability.

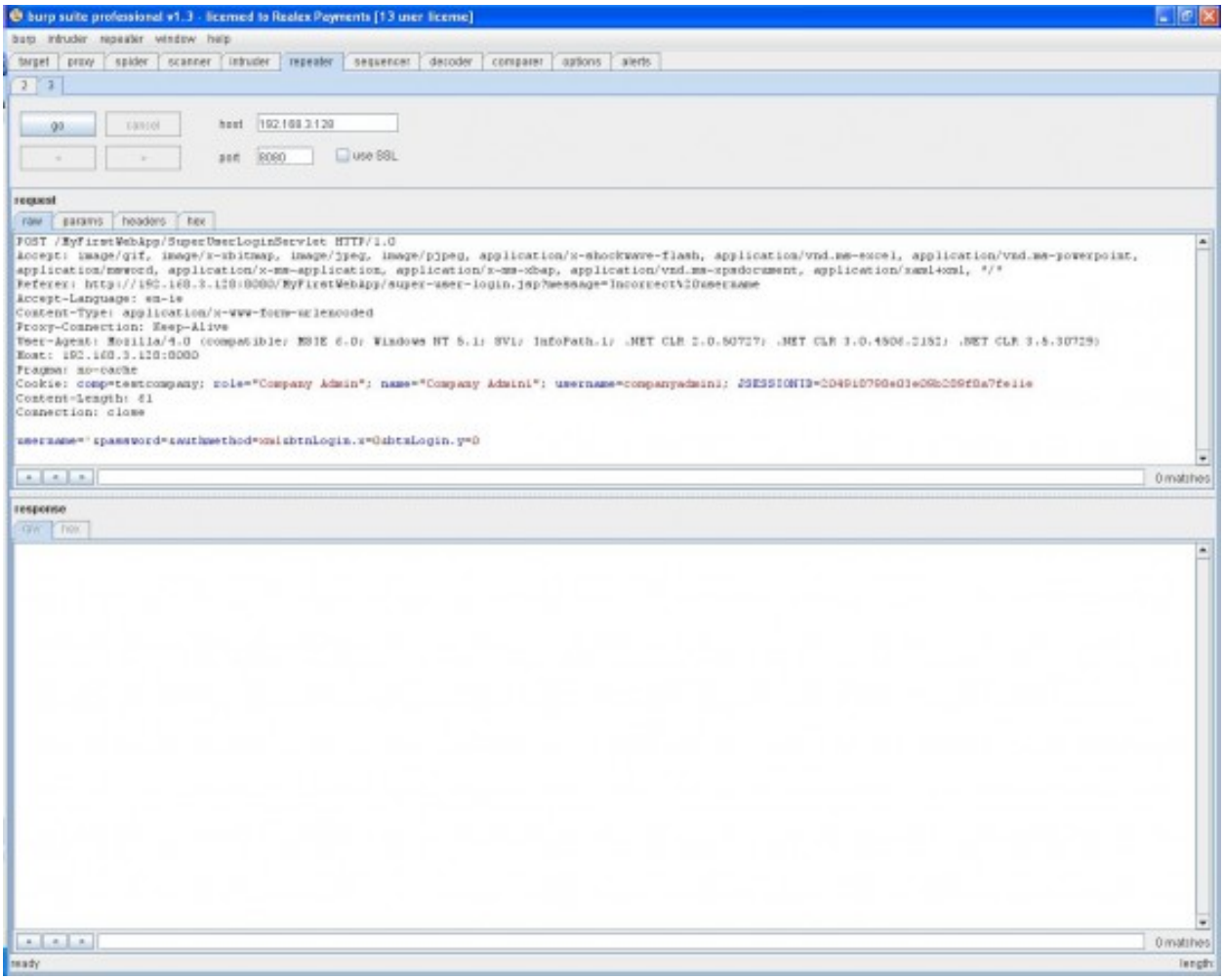
Fine tuning the attack with the Repeater Tool

The Repeater tool will allow us to manually modify and submit requests and view the response within the Burp Suite. We can use the Repeater tool to manually test any potential vulnerabilities that have been highlighted by the Intruder tool. This approach allows us to submit a large amount of requests (84 in this example) automatically with the Intruder tool and fine tune any "interesting" requests (4 in this example) with the Repeater. The Intruder tool has highlighted 4 responses which include our pattern matching keyword "XPATH".

We will right click request number 13 (the same request as we looked at above) and send this to the Repeater tool:

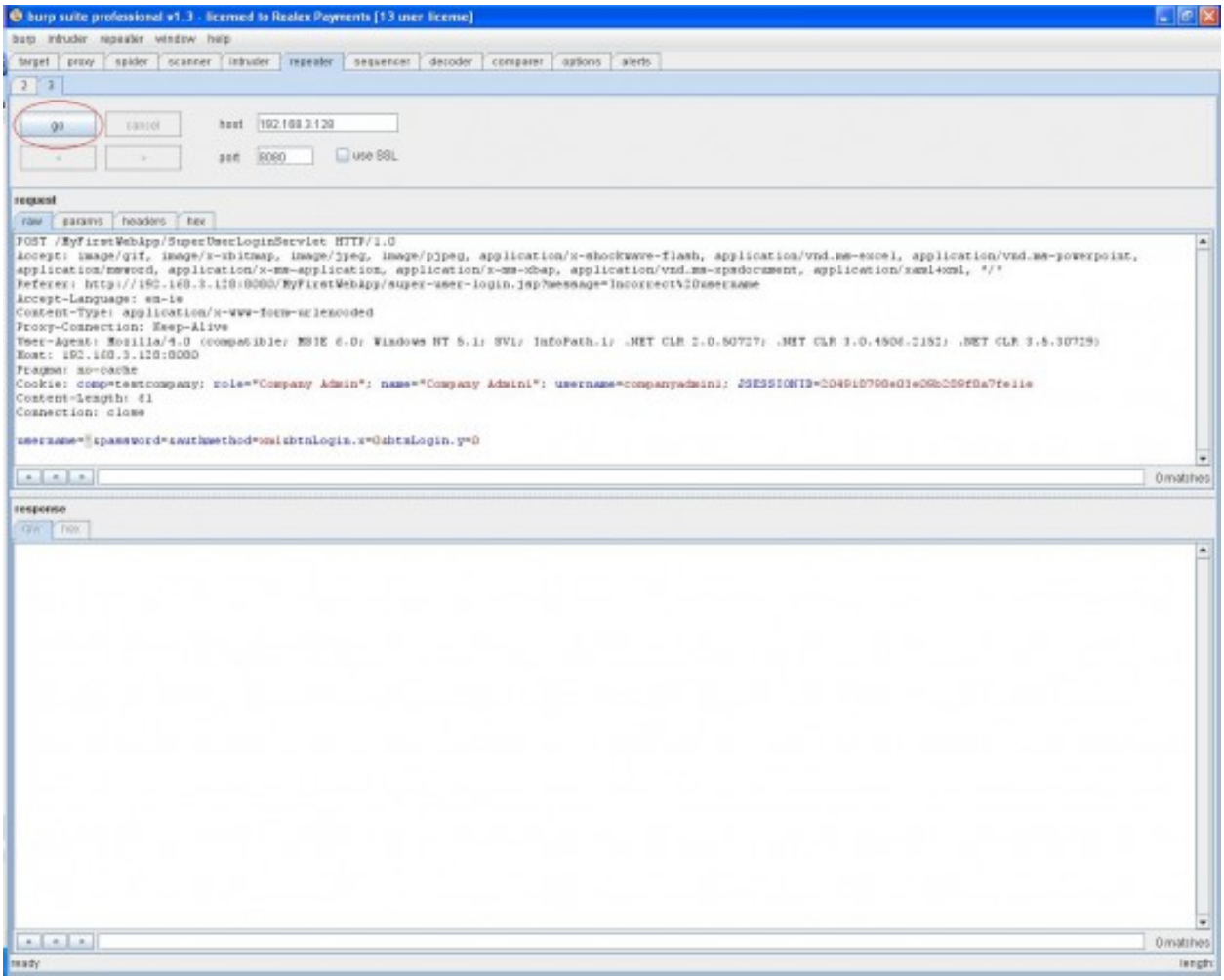


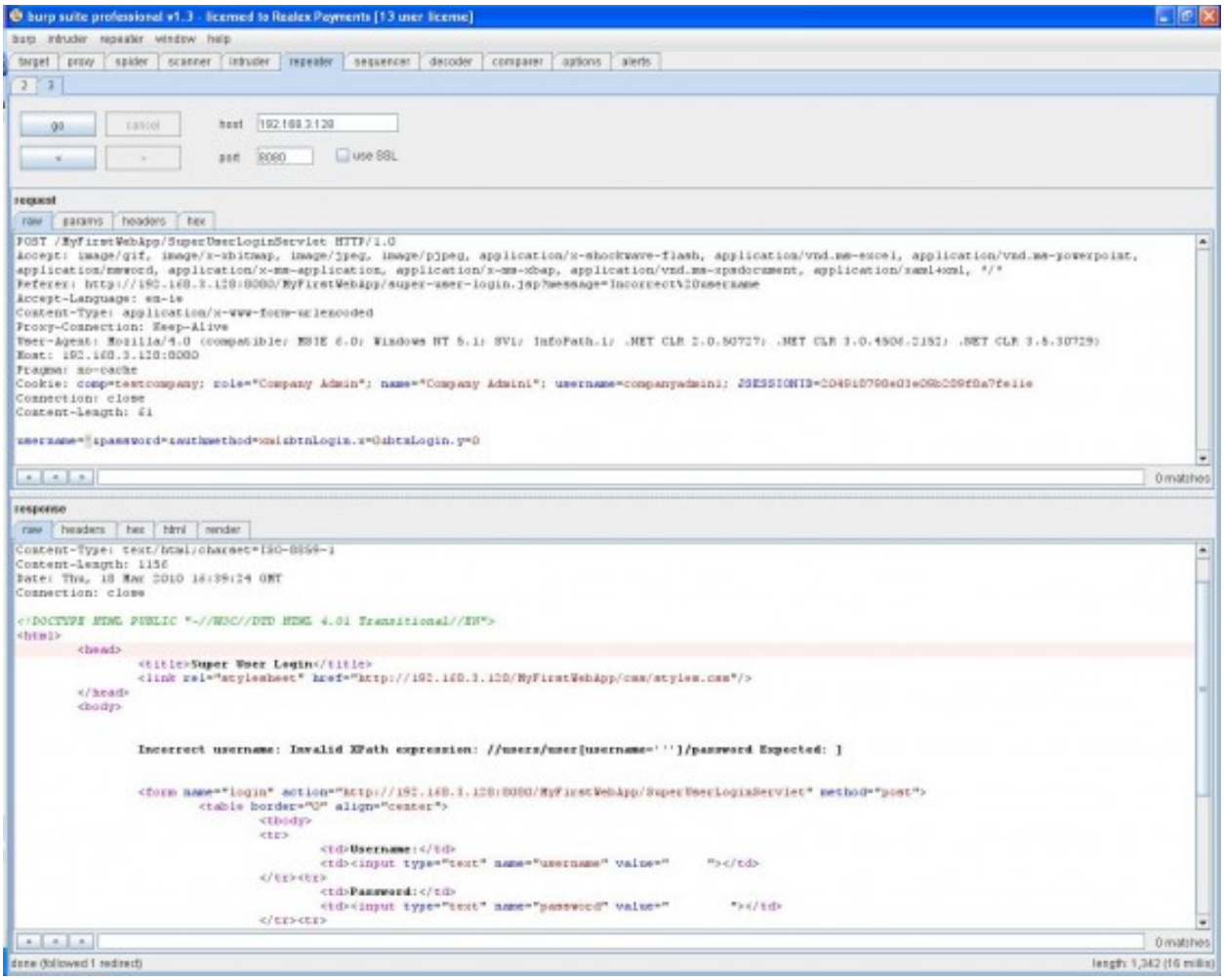
The Burp Suite will send request number 13 to the Repeater tool as shown below:



We have seen in the Intruder tool that entering a single quote (‘) into the username field gives us an XPATH error. This could potentially be exploited by a malicious user to bypass our authentication check.

In the Repeater tool you modify the request however you want and click on the “go” button. The response will be shown in the bottom pane of the Repeater tool:


burp suite professional v1.3 - licensed to Realex Payments [13 user license]
burp intruder repeater window help
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
go
cancel
host 192.168.3.120
port 8080
use SSL
request
raw params headers hex
POST /MyFirstWebApp/SuperUserLoginServlet HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-ms-application, application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xaml+xml, */*
Referer: http://192.168.3.120:8080/MyFirstWebApp/super-user-login.jsp?message=Incorrect%20username
Accept-Language: en-ie
Content-Type: application/x-www-form-urlencoded
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; InfoPath.1; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Host: 192.168.3.120:8080
Pragma: no-cache
Cookie: comp=testcompany; role="Company Admin"; name="Company Admin"; username=companyadmin1; JSESSIONID=CD491D790E01E08BC59F8A7FE11E
Content-Length: 61
Connection: close
username=&password=&authmethod=xmlbtnLogin.x=0&btnLogin.y=0
0 matches
response
raw hex
0 matches
ready
length



We want to use the Repeater tool to fine tune our attack against the username field so we can replace the username and password value with different inputs and submit these individual requests.

The error message we received in the response shown above was:

Incorrect username: Invalid XPath expression: //users/user[username='']/password Expected:]

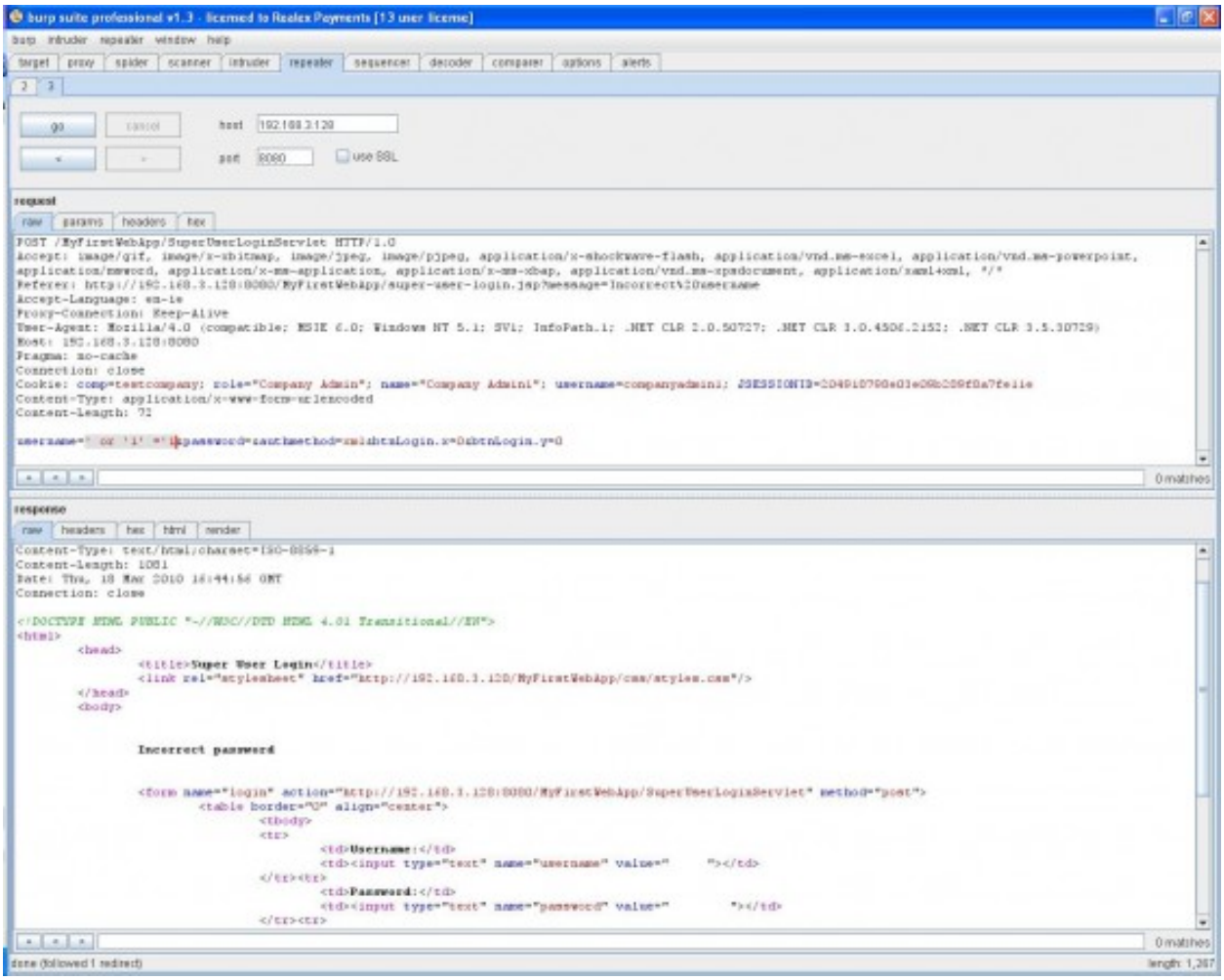
You can see that our single quote is breaking the XPATH query. We need to use these error messages to our advantage. If we continue to modify the username value we should eventually be able to exploit this vulnerability thanks to the lack of input validation and the informative error messages.

The informative error messages will help us exploit this flaw because we can see that the XPATH query looks something like this:

//users/user[username=' “username supplied by user” ']/password

It should now be obvious why our single quote broke the query. We can manually modify the username value using the Repeater tool until we find an input that doesn't break the query and allows us to authenticate without knowing a valid username.

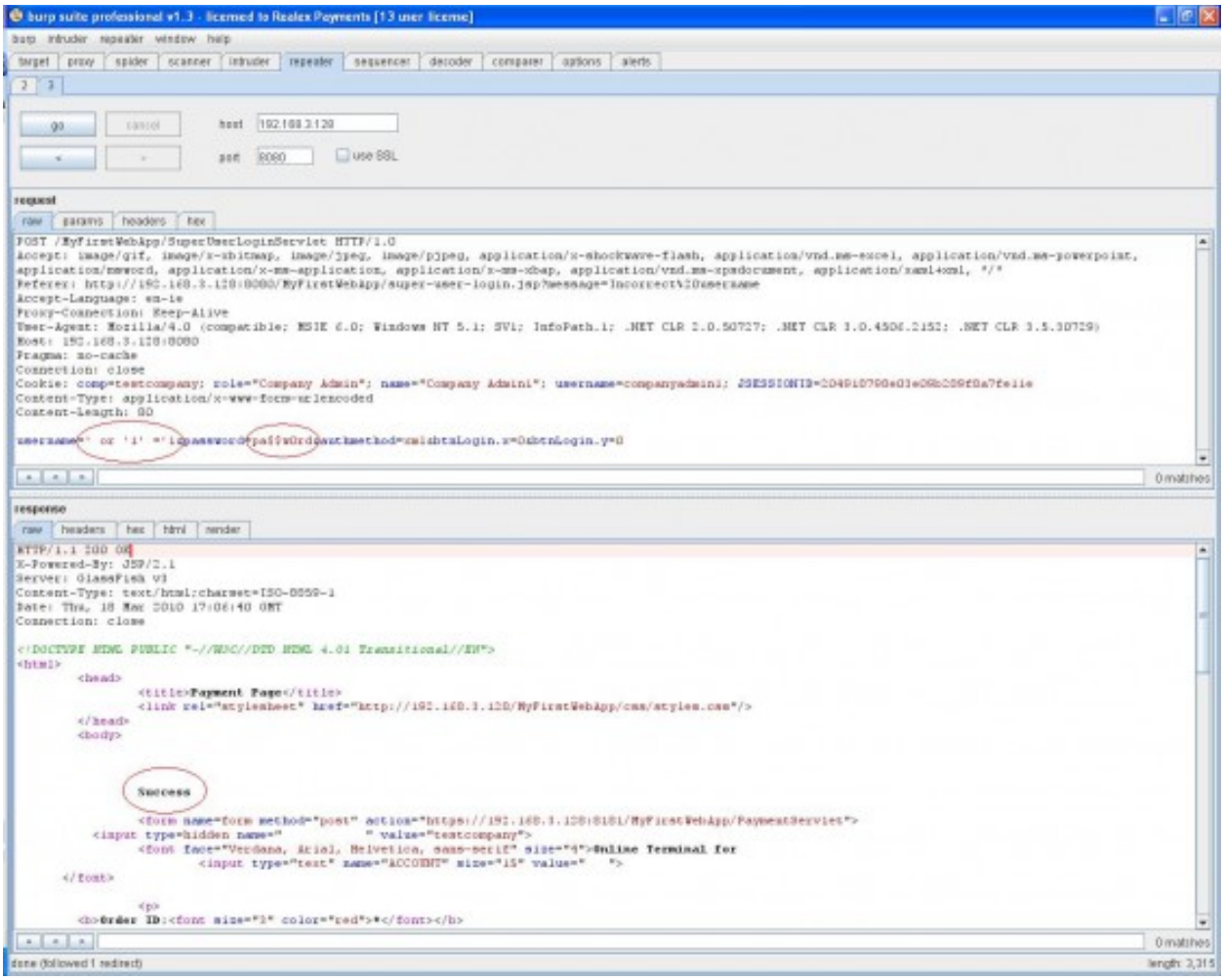
In the image below you can see that we have replaced the single quote with a different input which allows us to move closer to exploiting this flaw:



When we use the Repeater to submit a request where the username value is **' or '1' ='1** we get a different error. The error tells us the password (blank in the request) we submitted was incorrect. The XPATH query will now look something like this:

//users/user[username=' **' or '1' ='1** ']/password

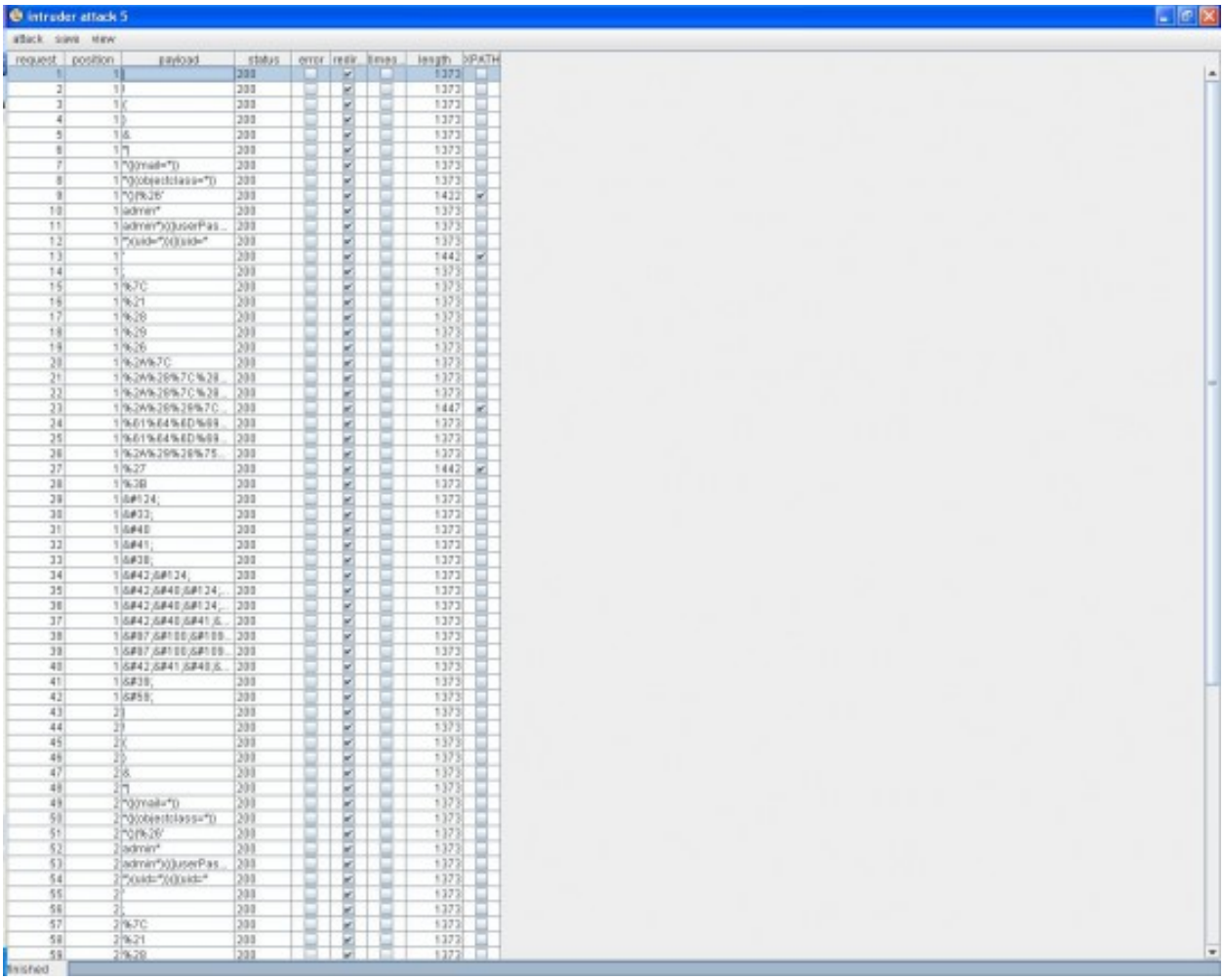
This means that the username value we submitted is "OK" and we now need to guess a password value. We can brute force this value using the Intruder tool as we saw in the Intruder tool tutorial. I would like you to assume that I have brute forced this value using the Intruder tool and I have entered it into the Repeater tool below:



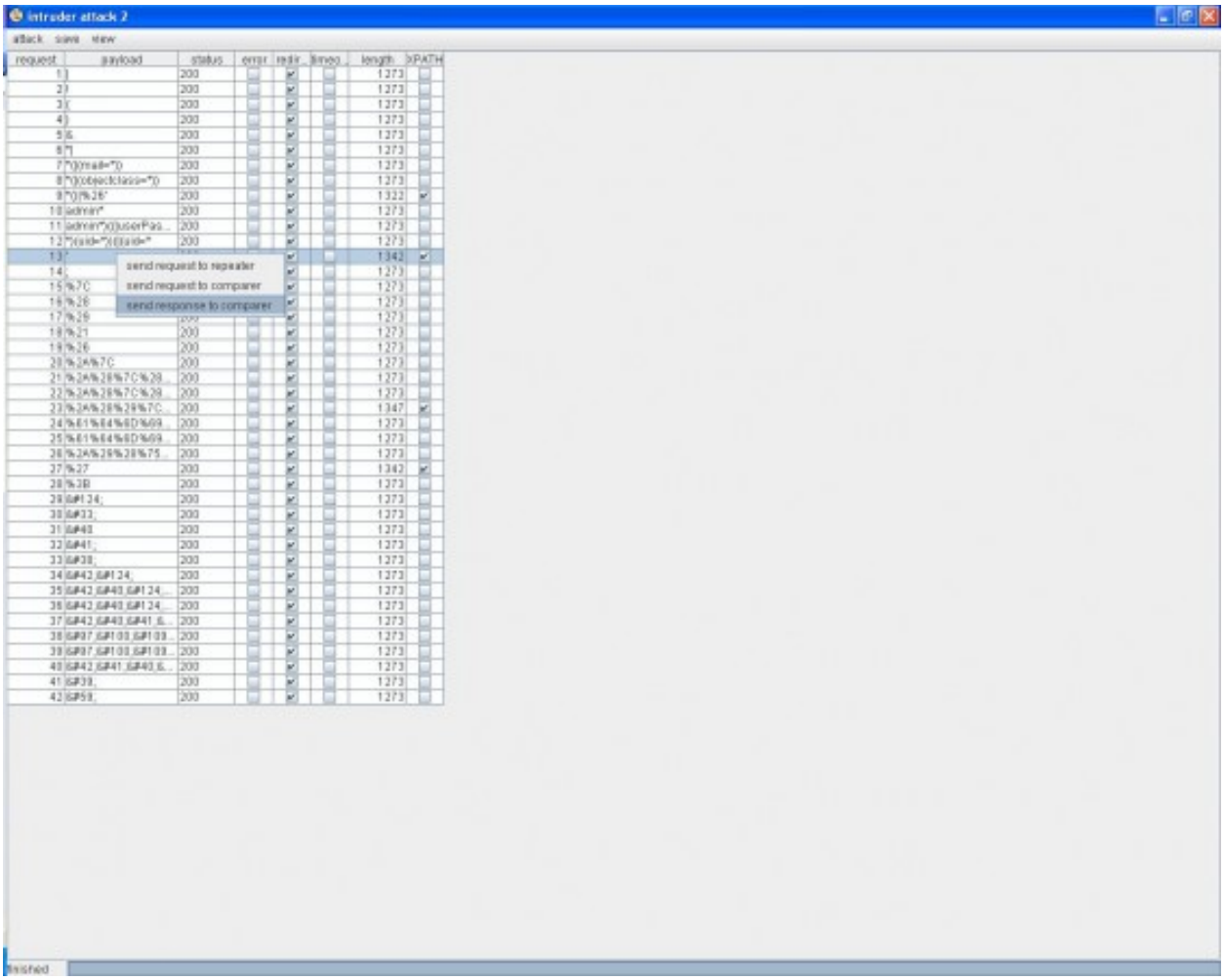
By passing in the **‘ or ’1′ =’1** value as the username and our brute forced password value we have successfully logged into the Payment Page of our application.

Comparing responses with the Comparer tool

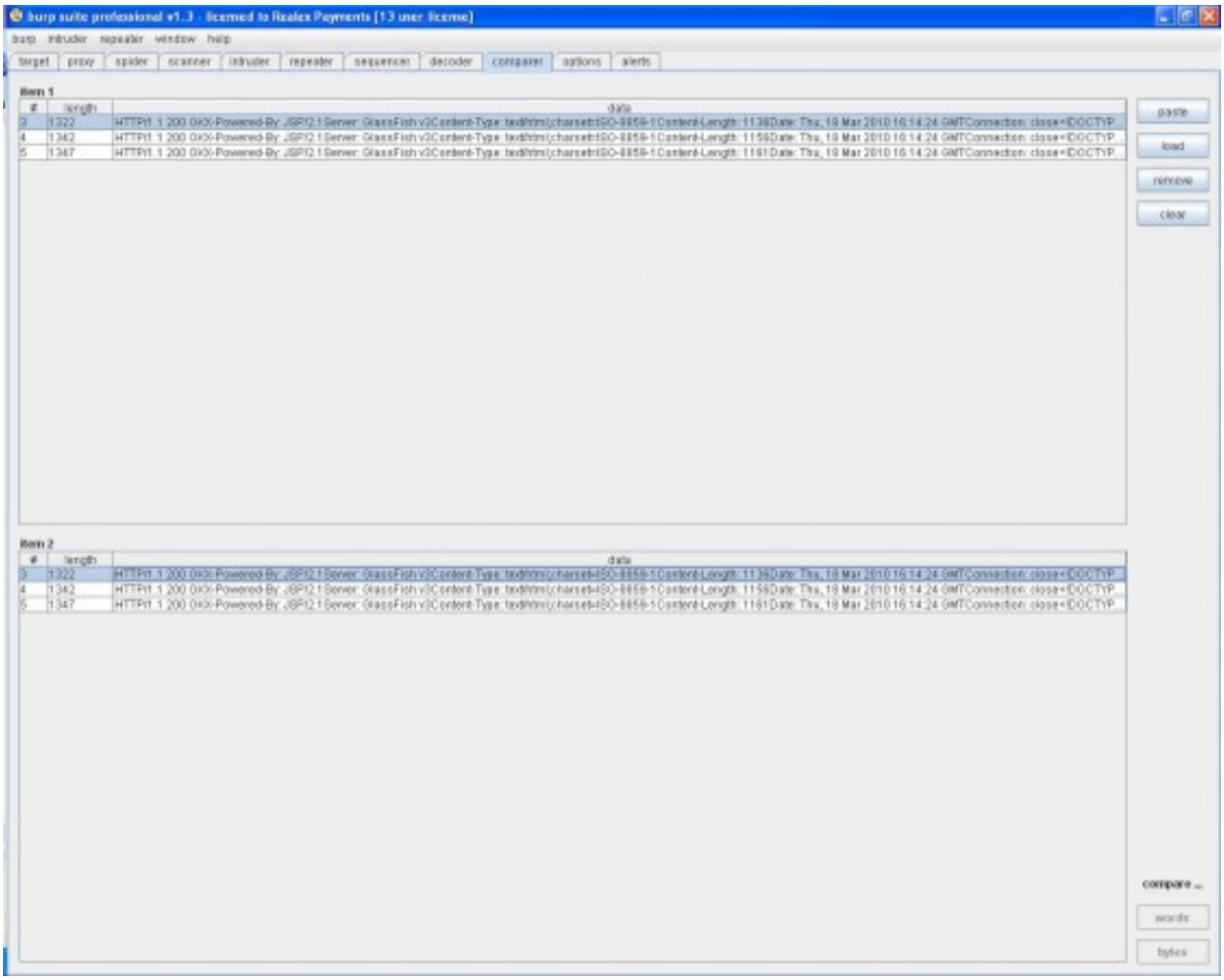
The Comparer tool allows us to compare multiple requests and responses to easily see the differences between them. In our Intruder tool test you can see that tests 9, 13 and 23 all matched our pattern matching keyword but the size of the responses were all slightly different.



We can use the Comparer tool to compare the three responses and show us the differences between them by right clicking them in Intruder attack window and selecting “send response to comparer”:



In the image below you can see that all three of the responses are now in the Comparer tool:

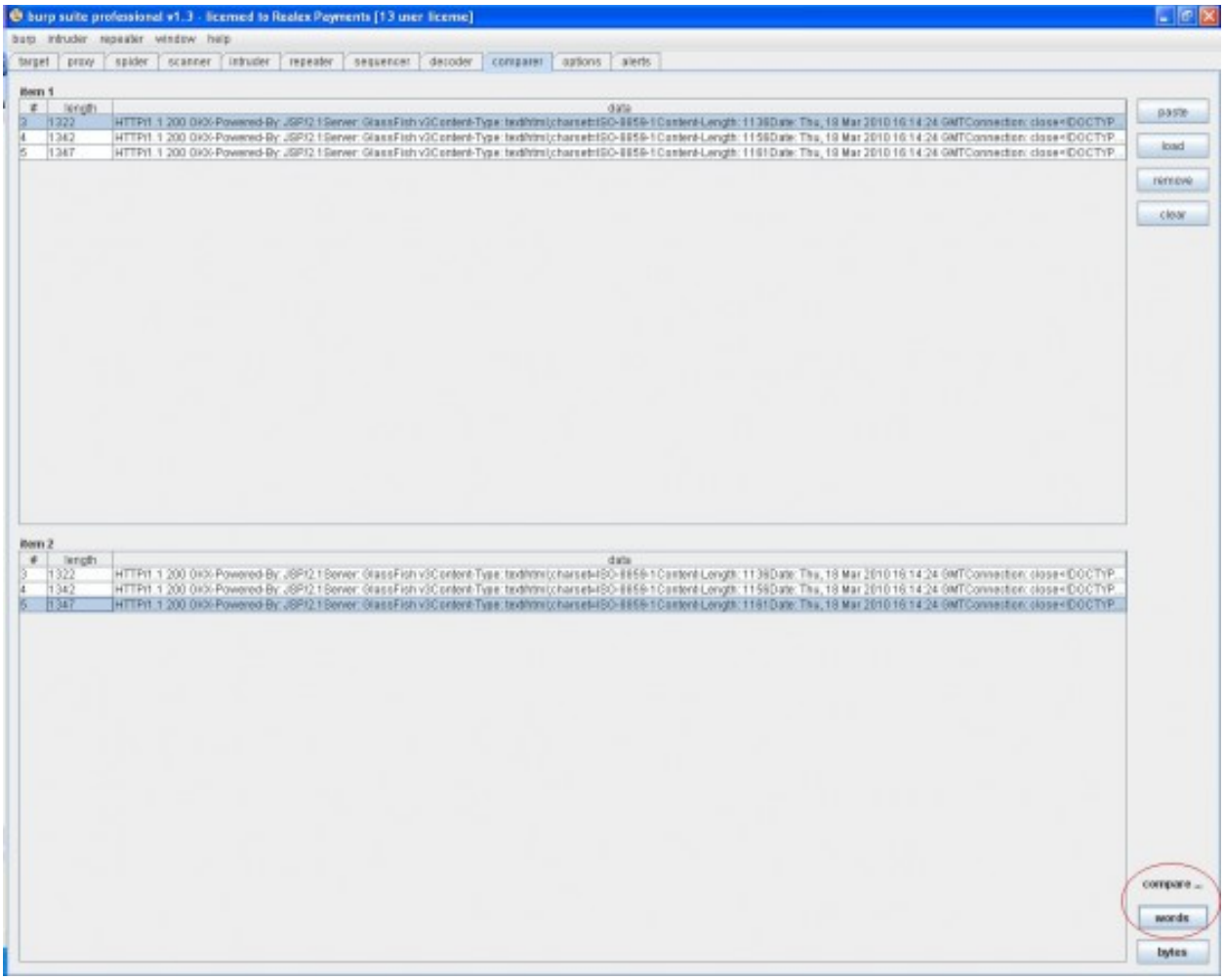


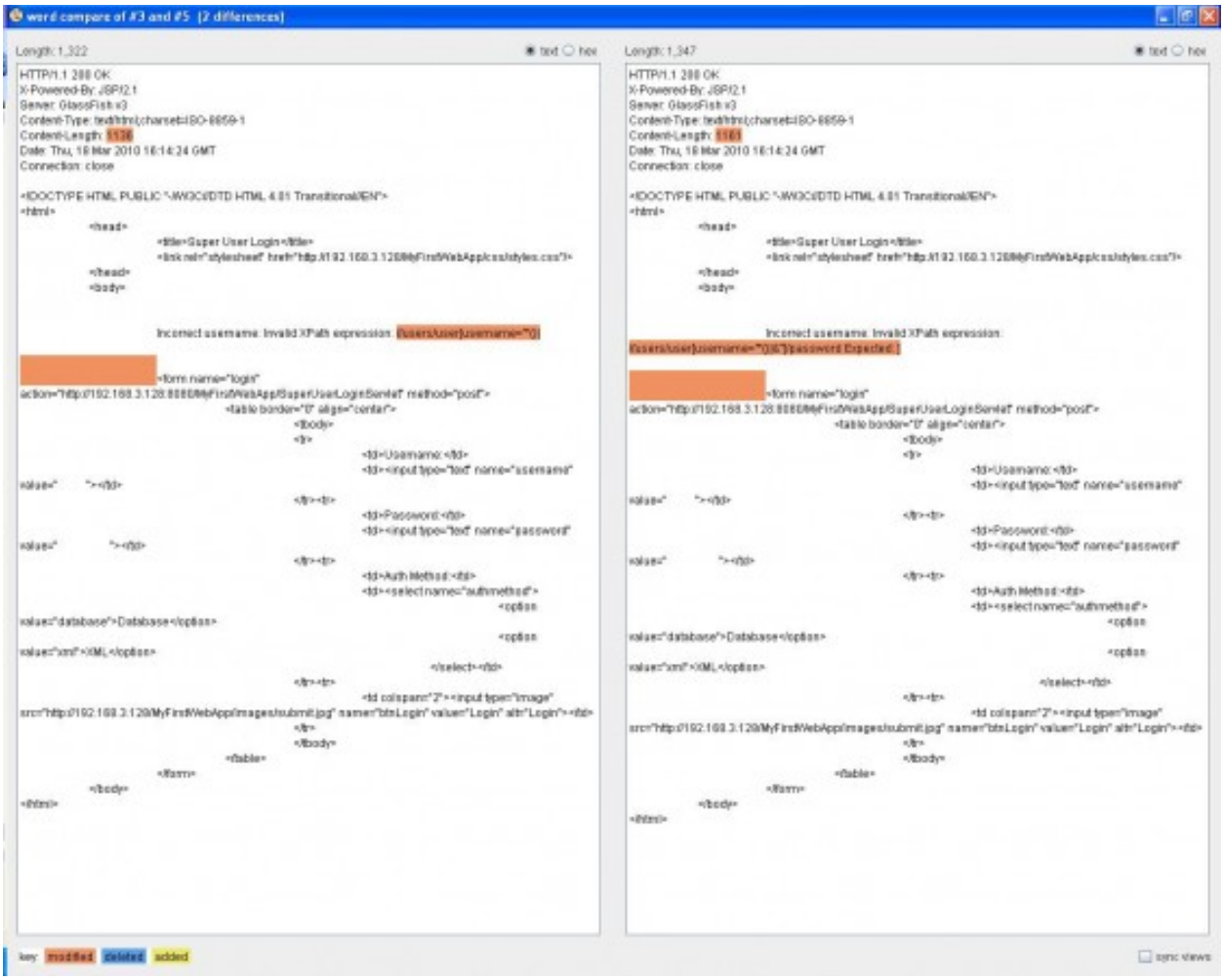
You can select two responses and click on one of the two compare types (descriptions take from the Port Swigger website):

Words: This comparison tokenises each item based on whitespace delimiters, and identifies the token-level edits required to transform the first item into the second. It is most useful when the interesting differences between the compared items exist at the "word" level, for example in HTML documents containing different content.

Bytes: This comparison identifies the byte-level edits required to transform the first item into the second. It is most useful when the interesting differences between the compared items exist at the "byte" level, for example in HTTP requests containing subtly different values in a particular parameter or cookie.

We are going to use the words comparer to compare request number 3 against request number 5:


burp suite professional v1.3 - licensed to Realex Payments [13 user license]
burp intruder repeater window help
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
item 1
length data
3 1322 HTTP/1.1 200 OKX-Powered-By: JSP/2.1Server: GlassFish v3Content-Type: text/html;charset=ISO-8859-1Content-Length: 1136Date: Thu, 18 Mar 2010 16:14:24 GMTConnection: close<!DOCTYP
4 1342 HTTP/1.1 200 OKX-Powered-By: JSP/2.1Server: GlassFish v3Content-Type: text/html;charset=ISO-8859-1Content-Length: 1156Date: Thu, 18 Mar 2010 16:14:24 GMTConnection: close<!DOCTYP
5 1347 HTTP/1.1 200 OKX-Powered-By: JSP/2.1Server: GlassFish v3Content-Type: text/html;charset=ISO-8859-1Content-Length: 1161Date: Thu, 18 Mar 2010 16:14:24 GMTConnection: close<!DOCTYP
paste
load
remove
clear
item 2
length data
3 1322 HTTP/1.1 200 OKX-Powered-By: JSP/2.1Server: GlassFish v3Content-Type: text/html;charset=ISO-8859-1Content-Length: 1136Date: Thu, 18 Mar 2010 16:14:24 GMTConnection: close<!DOCTYP
4 1342 HTTP/1.1 200 OKX-Powered-By: JSP/2.1Server: GlassFish v3Content-Type: text/html;charset=ISO-8859-1Content-Length: 1156Date: Thu, 18 Mar 2010 16:14:24 GMTConnection: close<!DOCTYP
5 1347 HTTP/1.1 200 OKX-Powered-By: JSP/2.1Server: GlassFish v3Content-Type: text/html;charset=ISO-8859-1Content-Length: 1161Date: Thu, 18 Mar 2010 16:14:24 GMTConnection: close<!DOCTYP
compare ...
words
bytes



You can see that the main difference between the two requests is the XPATH error message that is returned. The response with the smaller length does not have “password expected” in its error message.

I hope you have found this blog post useful and I’m always interested in hearing any feedback you have.

SN