



Crowbar:
New generation web application brute force attack tool



Tables of contents

Introduction.....	3
Challenges with brute force attacks	3
Crowbar – a fresh approach	3
Other applications of crowbar.....	13
Conclusion	13

Introduction

In some rare cases we are left with only one option – brute force. We have written numerous PERL scripts to perform some type of brute force attack against web applications. When do you use brute force? Hereby some examples:

1. Administrative backend to web sites usually use weak passwords...and have no lockout. While it might mildly amusing to try a few passwords its gets boring after about the 3rd try.
2. Some application has been found to respond different when the username is correct but the password/PIN is incorrect. Noting the difference in response could lead to user enumeration.
3. Believe it or not – some web application still does not enforce lockout...or worse let the client keep record of how many tries has been made.
4. Applications that do enforce lockout open themselves up to denial of service. Hit each user with 3 attempts and lock the user out.
5. Access to a site is controlled by an account number and a PIN. While there's lockout on 3 incorrect attempts against the PIN one might consider cycling through account numbers while keeping the PIN static.

Challenges with brute force attacks

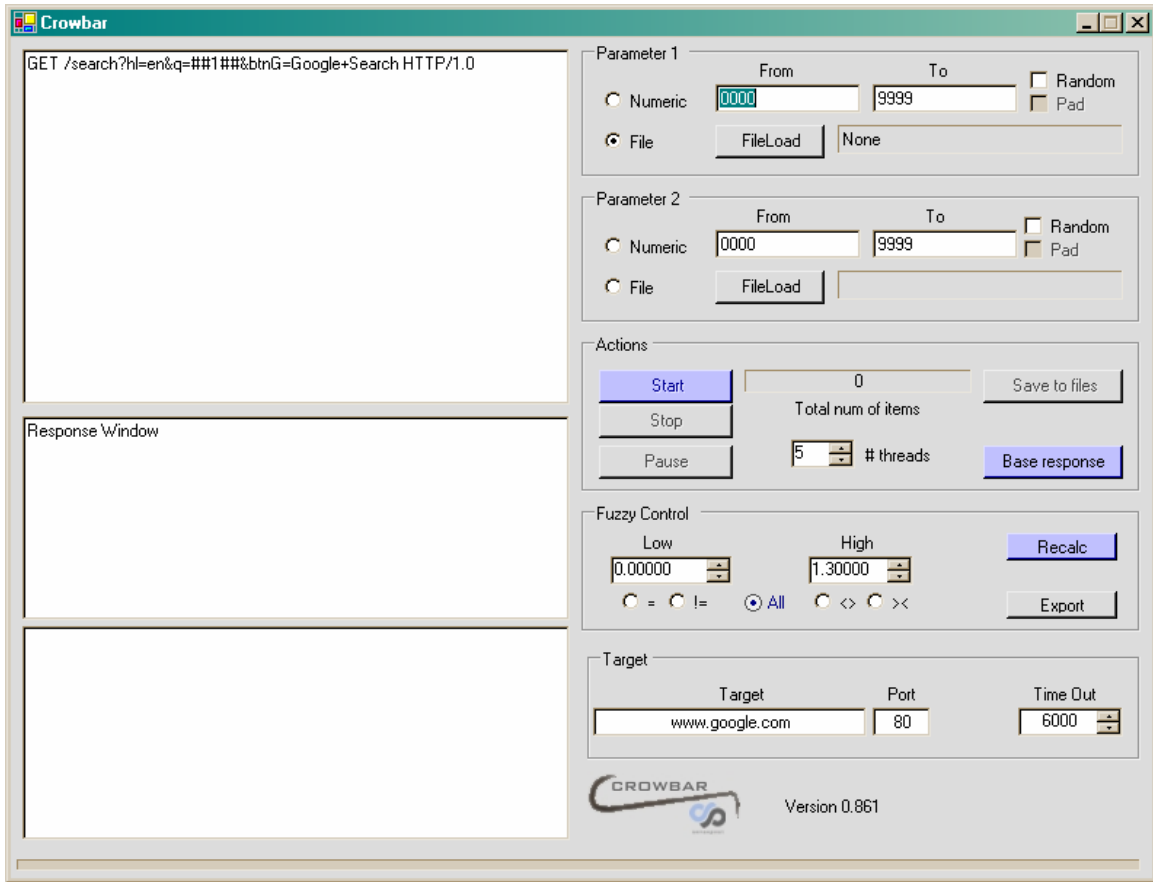
The problem with creating brute force attacks is that each application is different. Some apps require you to brute the state session contained in the cookie – some pass credentials as parameters. In most cases you also don't know what a positive response looks like. Also – it's really easy to miss very subtle changes in the response – minute changes in the response (sometimes in the HTML code and not even visible) could lead to some form of enumeration.

Many software writers have attempted to write a generic web application brute forcer. Most of these pieces of software try to make it as easy as possible to perform the brute force attack. Dumbing software down usually causes it to be less generic – you are almost certain that the software won't work for your “non-standard” application.

Crowbar – a fresh approach

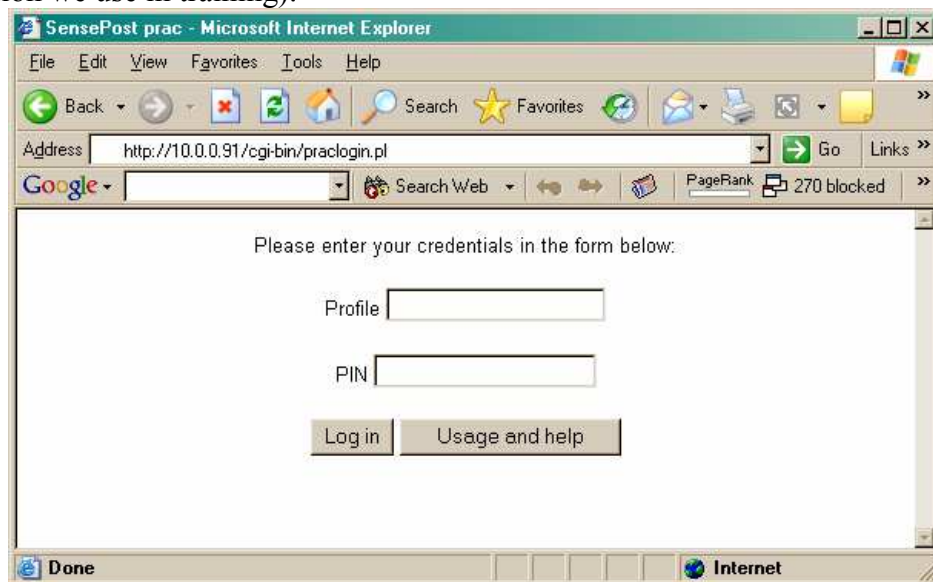
Crowbar gives you the opportunity to be in total control of what is submitted to the web server. Crowbar does not attempt to identify a positive response (e.g. when you hit a correct username/password combination). It rather asks you to give it a “baseline” – it then compares the content of the response with the content of the baseline. The content compare process is really basic but works really well – it is the same content compare algorithm that we use in Wikto. Crowbar then gives you the opportunity to define conditions that represents a positive.

Let's see how it works...when crowbar fires up it looks like this:



To identify the actual request that goes to the server we use any type of inline proxy (Paros, Spike, @stake web proxy, etc.) From here we basically copy and paste the request straight into crowbar.

Let us for now assume that we are trying to brute force a very basic application (this is an application we use in training):

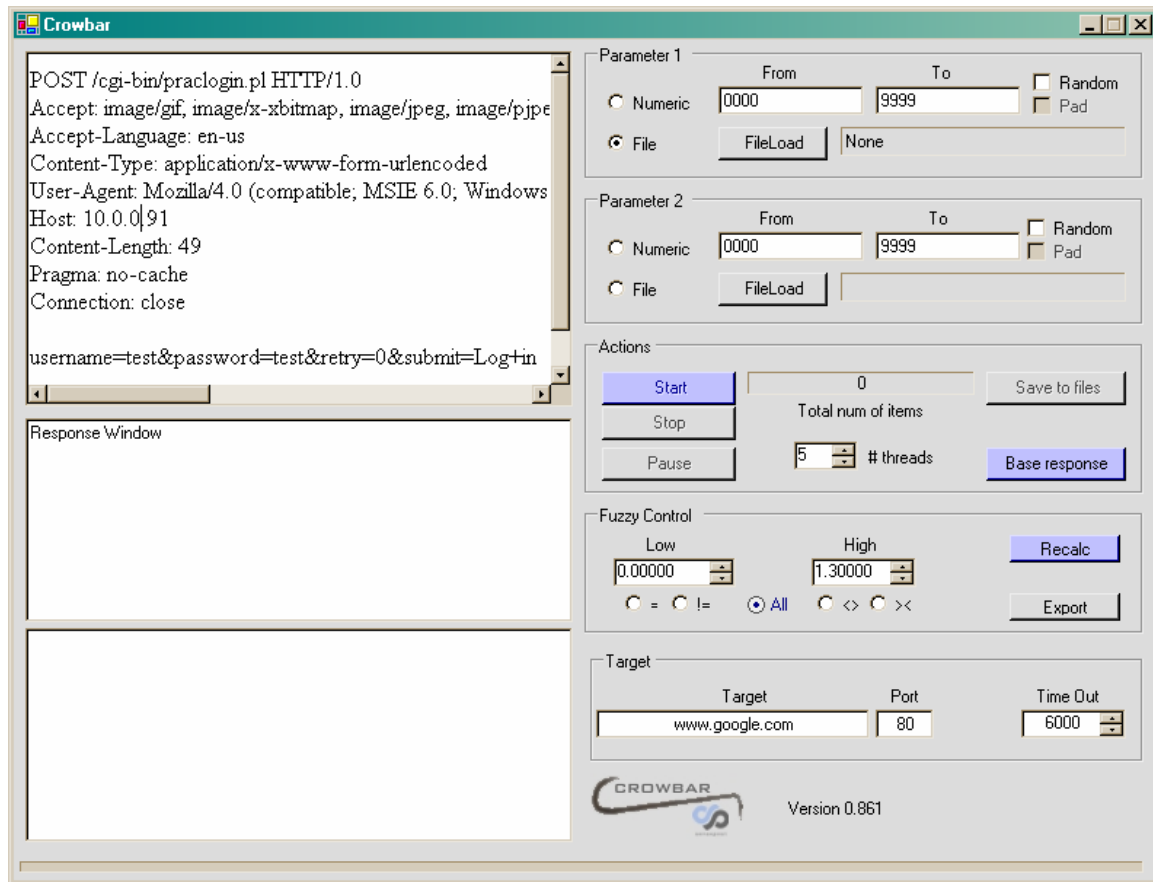


After trying to log in with test/test we see that the request looks like this:

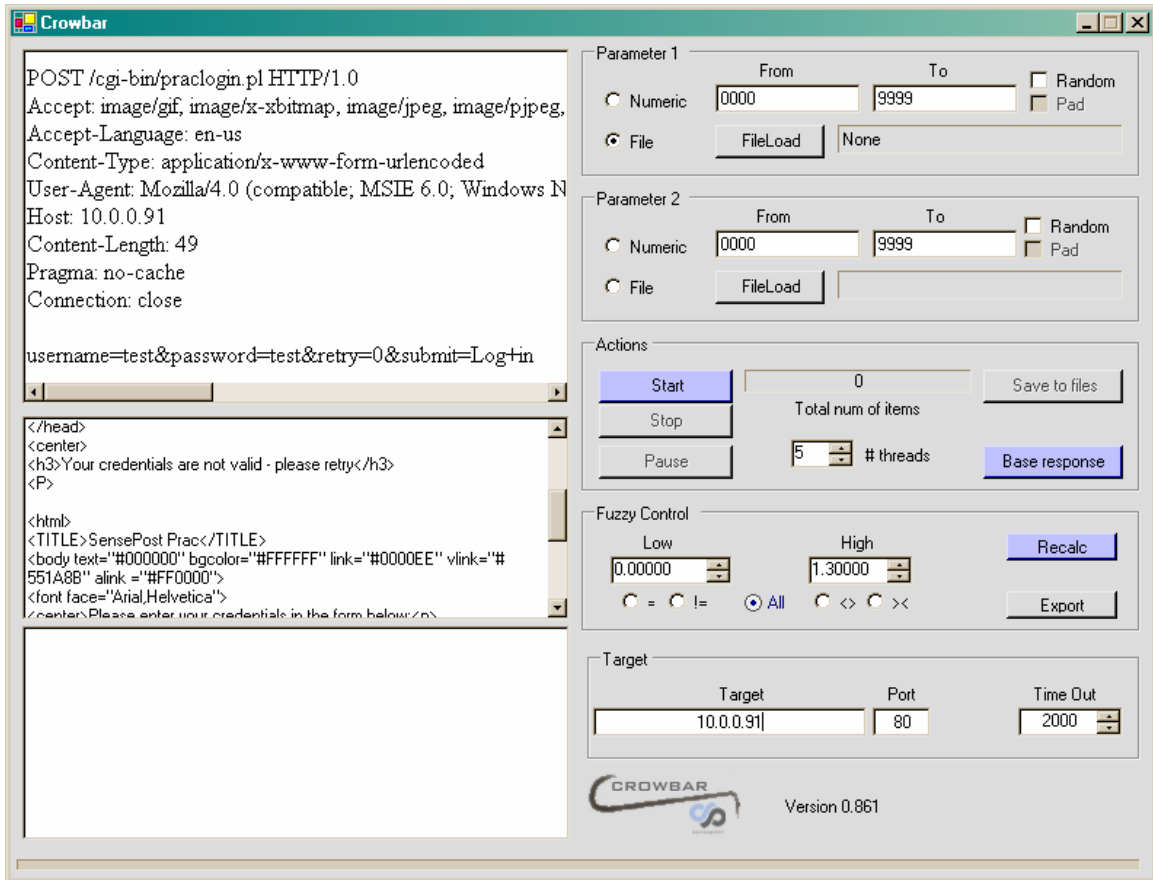
```
POST /cgi-bin/praclogin.pl HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-shockwave-flash, */*
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.0.3705; .NET CLR 1.1.4322)
Host: 10.0.0.91
Content-Length: 49
Pragma: no-cache
Connection: close
```

```
username=test&password=test&retry=0&submit=Log+in
```

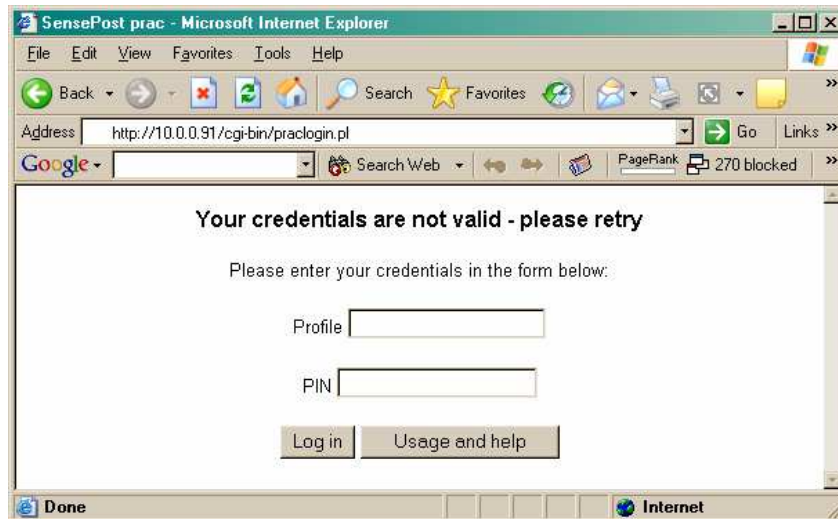
We take this very request and paste it into crowbar:



We set the IP address and port and hit “Base response” to obtain a base response:

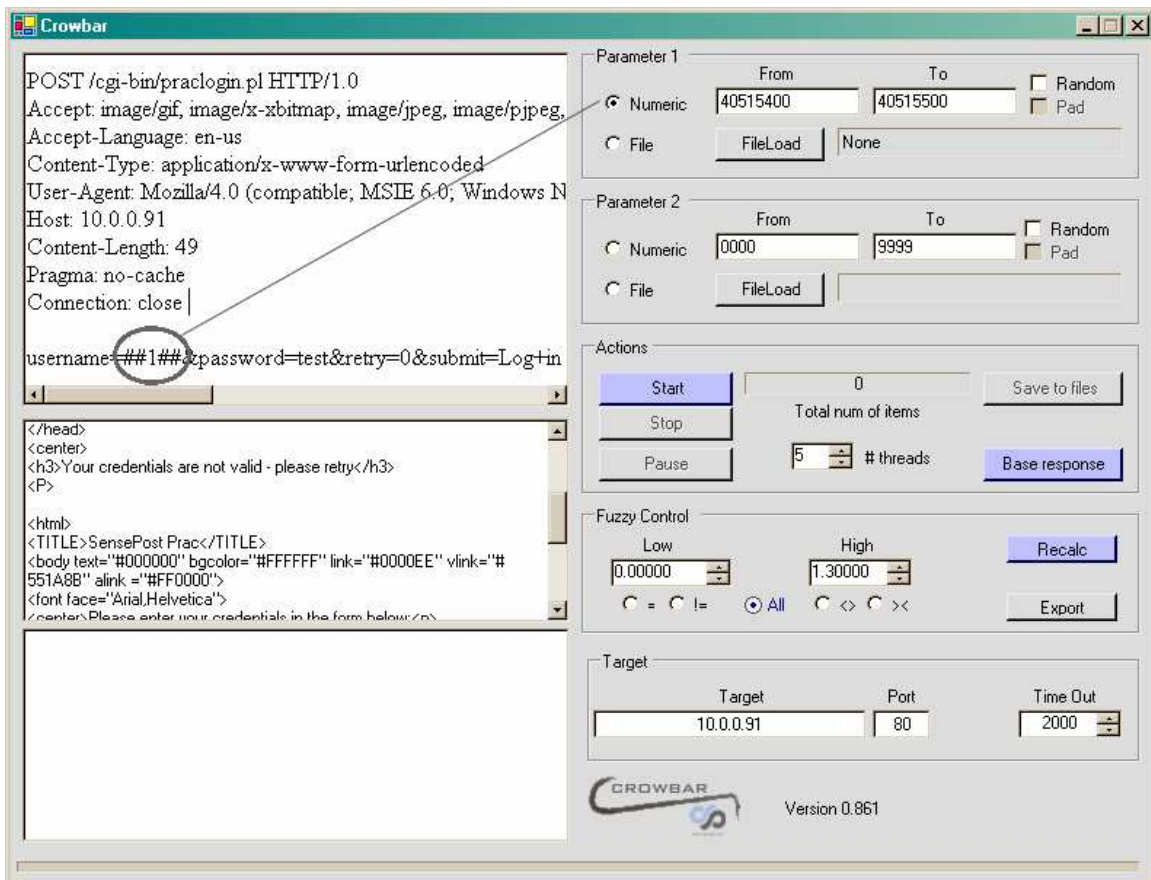


Note that the response corresponds to the page displayed when the credentials are incorrect:

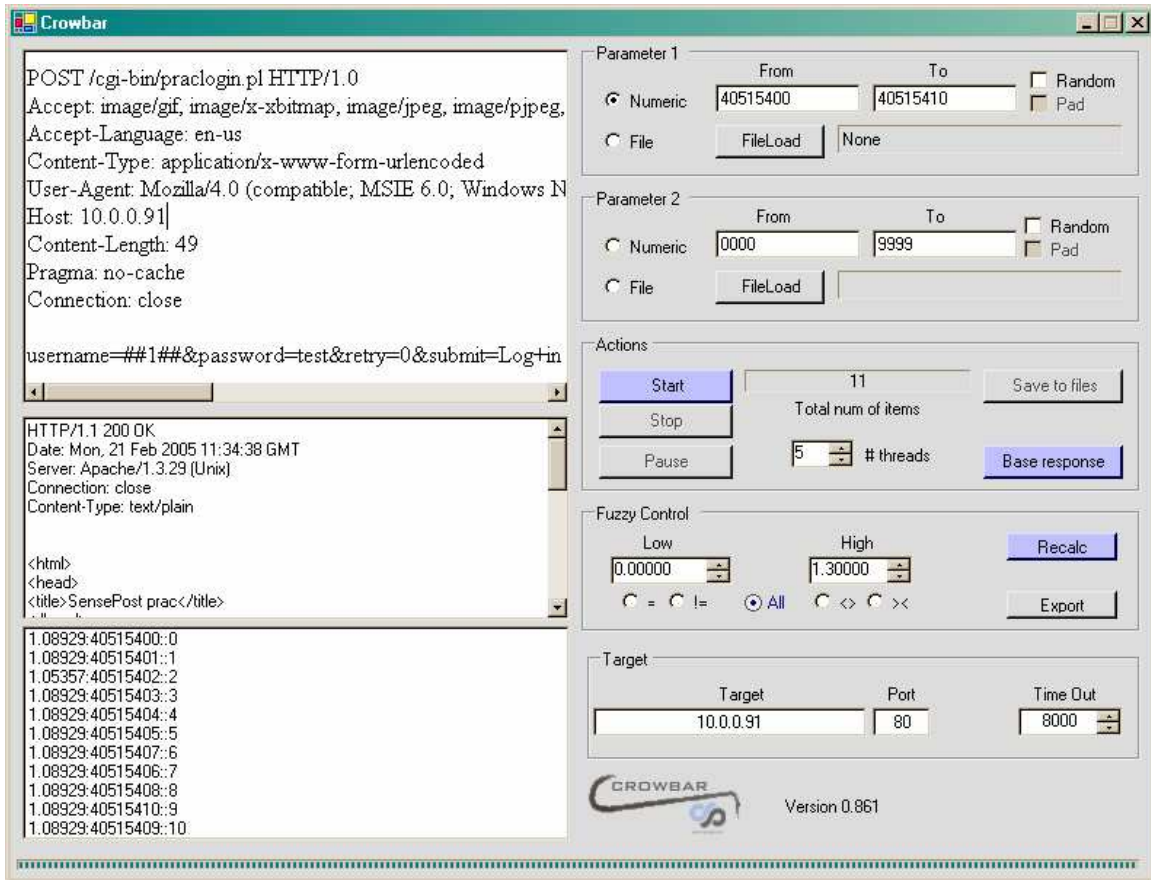


Now that we have the base response we can start the brute force attack. We need to determine which part of the request we want to “attack”. In this case we’ll start with the profile number. To connect this parameter to an action we mark it with a special string. To connect it to parameter 1 we enter ##1## in the request – it could be anywhere – in the

request, in a parameter or in the cookie. Here we connect it to the profile parameter. We set the parameter to numeric and let it range from 40515400 to 40515500 (the initial number comes from a help file on the server):



Now we are ready to begin the process – we specified the number of threads to use – let’s leave it at 5 and hit “Start”. Crowbar will now replay the request, working out the Content-Length for every request. It will search and replace the `##1##` marker to a range of numbers from 40515400 to 40515410 and compare the reply from the server to the baseline:

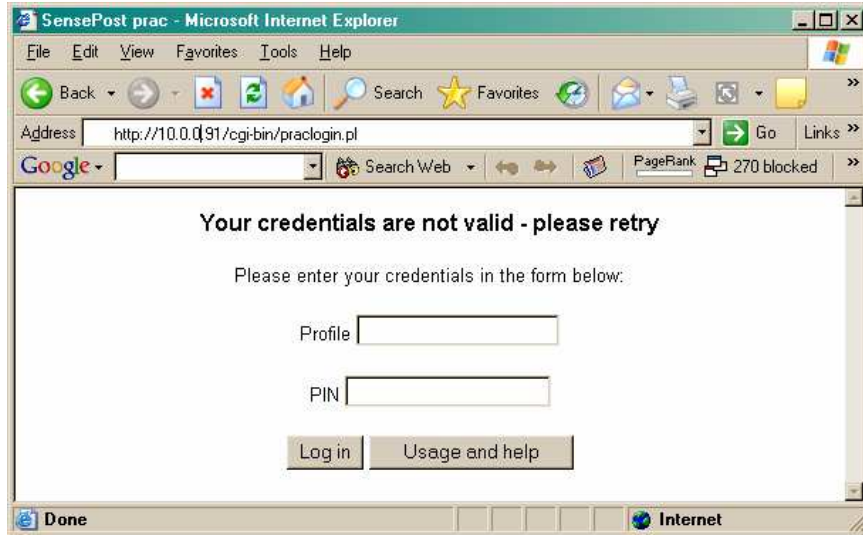


In the listbox note the following items:

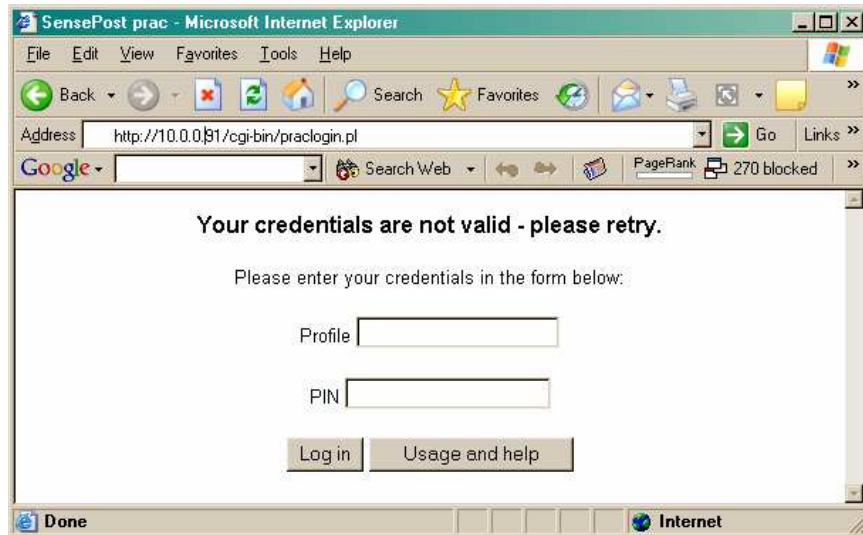
Content-compare index:parameter 1: parameter 2: iteration

We see that most content compare indexes have a number of 1.08929 – with the exception of 40515402 which is 1.05357. This means that the content returned when parameter 1 (username) was set to 40515402 was different to the other responses. Let’s investigate. We enter 40515400 with password “test” and 40515402 with the same password. See if you can spot the difference:

40515400:

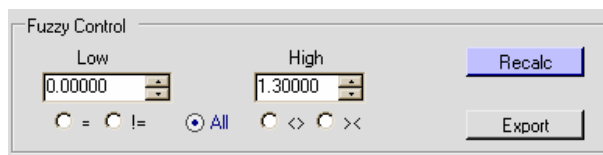


40515402:



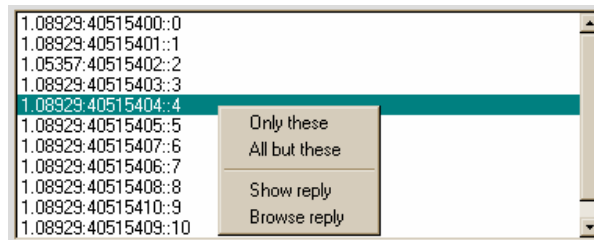
Note the period (.) at the end of the error message! Crowbar tells us that there is a slight difference in response.

We can now repeat the process with a larger range of profile numbers...but let us rather look at how we can easily isolate results. The following window is where you can set filters for results:



The control box contains a low and high watermark. Selecting the “<>” radio button and clicking “Recalc” 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 “Recalc” button will update the list.

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



“Only these” will automatically populate the low watermark with the selected value, set the equal radio button and click on “recalc”. “All but these” will do the inverse.

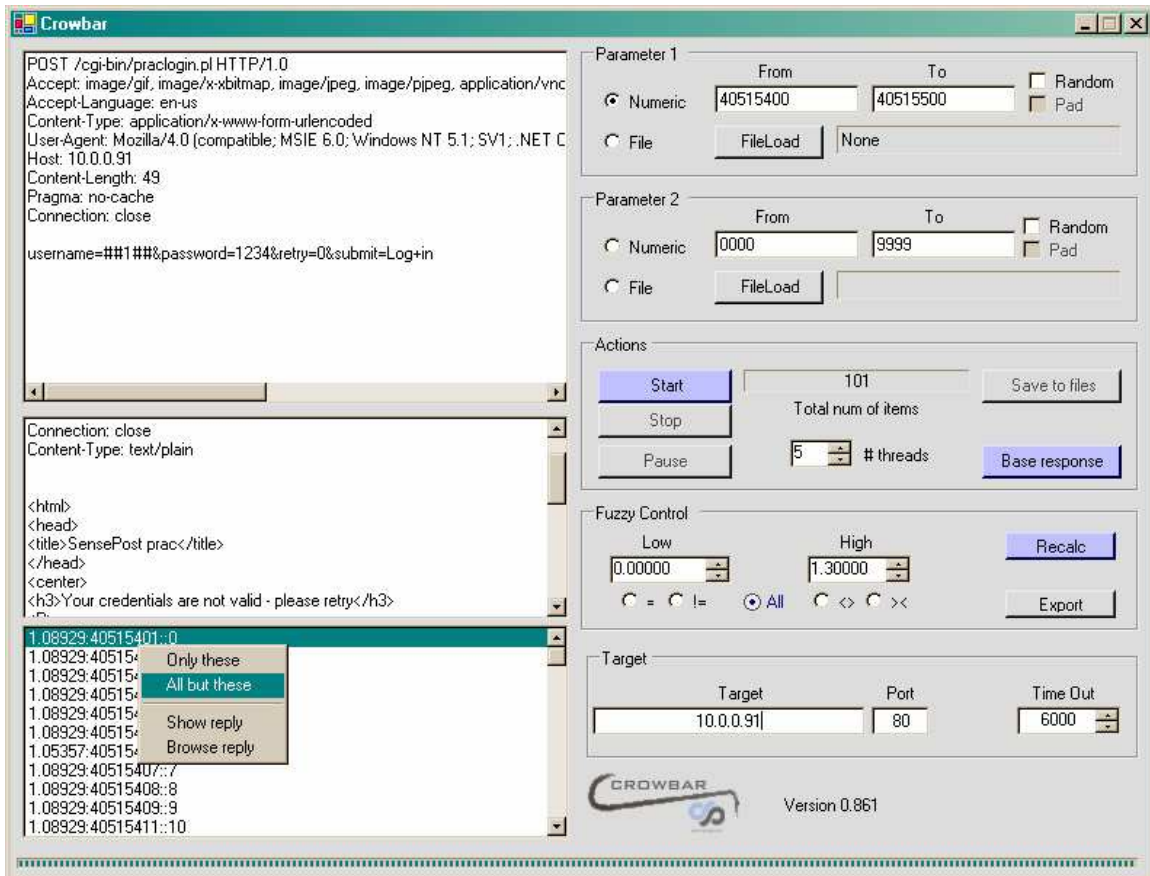
You can also look at the response in HTML format by clicking on “show reply”:



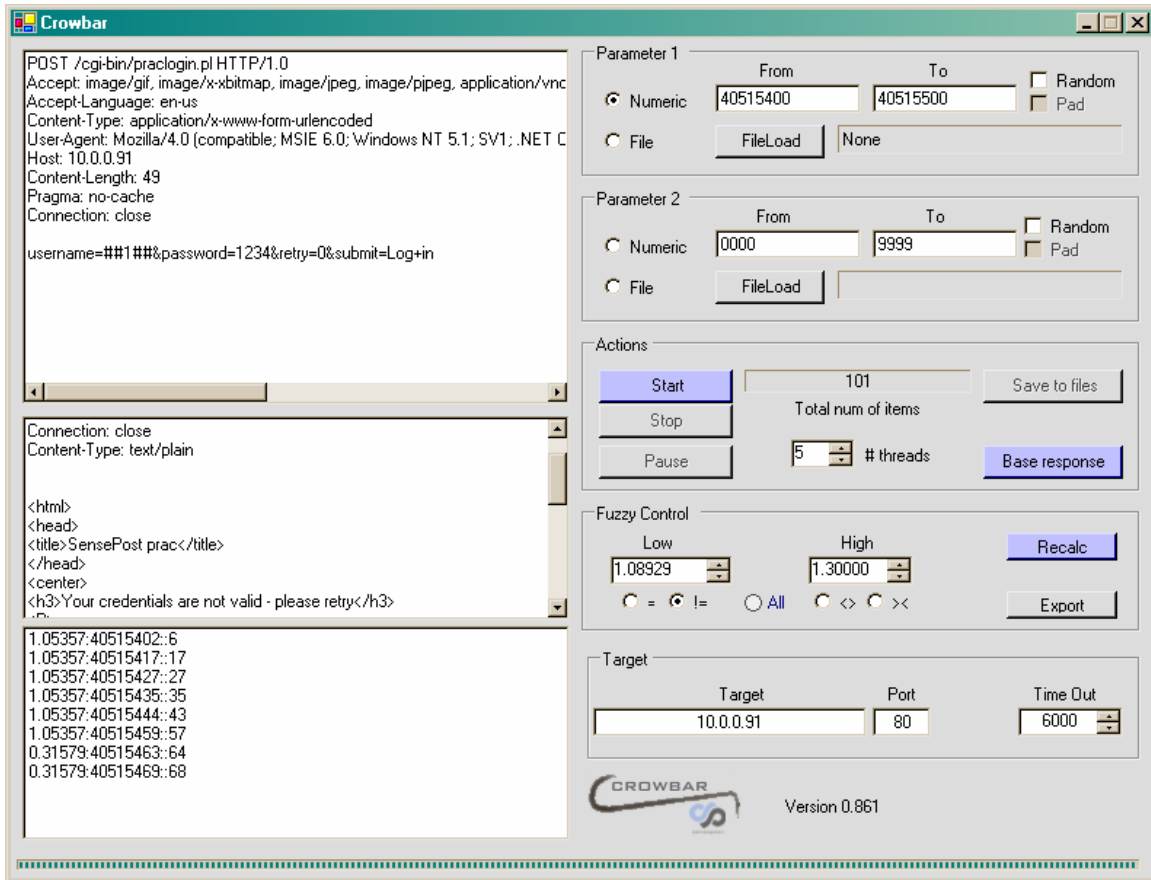
The “surf reply” shows the same page, but in a browser – it creates the file c:\crowbar-temp.html and surfs to the page.

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 text box – the format of the output is really simple – parameter1:parameter2.

Let us look at another use of crowbar with the same web application. We set the PIN to 1234 and look at the range again. As in the previous run we find that the most common response has an index of 1.08929. We now set the filter to show all responses that is not 1.08929:



This results in the following:



As in the previous run we see 1.05357 – these are valid profile numbers. But notice the 0.31579s – let’s see the response to these requests:



Ha! – valid credentials to the application. The brute forcing of a specific PIN is left for the reader as an exercise....:)

This demonstration should give you a good idea of how crowbar can be used.

Other applications of crowbar

Crowbar was intended for the brute forcing of web applications. Some people at SensePost have found some other interesting applications for the tool:

- Very primitive parameter fuzzer – load fuzzing type strings from a file and load into a parameter.
- Web services tester – do the same but in XML.
- Stress test tool

Conclusion

Crowbar does not try to be a particularly sexy tool – it's rather ugly. It's also not a point and click brute force tool – it rather gives the analyst as much as possible control of the process. There are many features in upcoming versions of Crowbar – alphanumeric brute forcing, automatic encoding to base64, md5 etc. But this is in the future...