

Mini MySQLat0r 0.3

User Manual



Table of Contents

'01-- Description.....3

'02-- Installation.....4

'03-- Usage.....5

 '03 AND 1-- Crawler Module.....5

 '03 AND 2-- Tester Module.....5

 '03 AND 3-- Exploiter Module.....7

'01-- Description

Mini MySquat0r is an application written to help with the discovery and exploitation of SQL injection vulnerabilities in web sites using MySQL. It consists of three different processes that consist of :

1. Crawler : to discover all pages and their respective parameters on a website
2. Tester : to test all the parameters for SQL injection vulnerabilities
3. Exploiter : to exploit the vulnerabilities found by the tester.

Mini MySquat0r is written in java which makes it portable to any platform having a java environment such as Windows, Linux and others. With the help of a simple graphical user interface, the discovery and exploitation of SQL injection vulnerabilities is greatly facilitated.

'02-- Installation

The only requirement in order for Mini MySQLat0r to function is that the JAVA runtime environment must be installed. It can be found at :

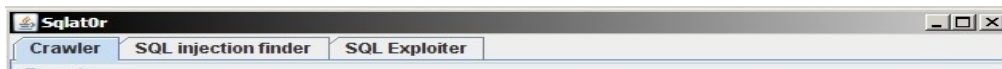
<http://java.sun.com/javase/downloads/index.jsp>

To run the application one can then simply double-click the MiniMySQLat0r_0_1.jar file or from the command line type :

```
java -jar MiniMySQLat0r_0_1.jar
```

'03-- Usage

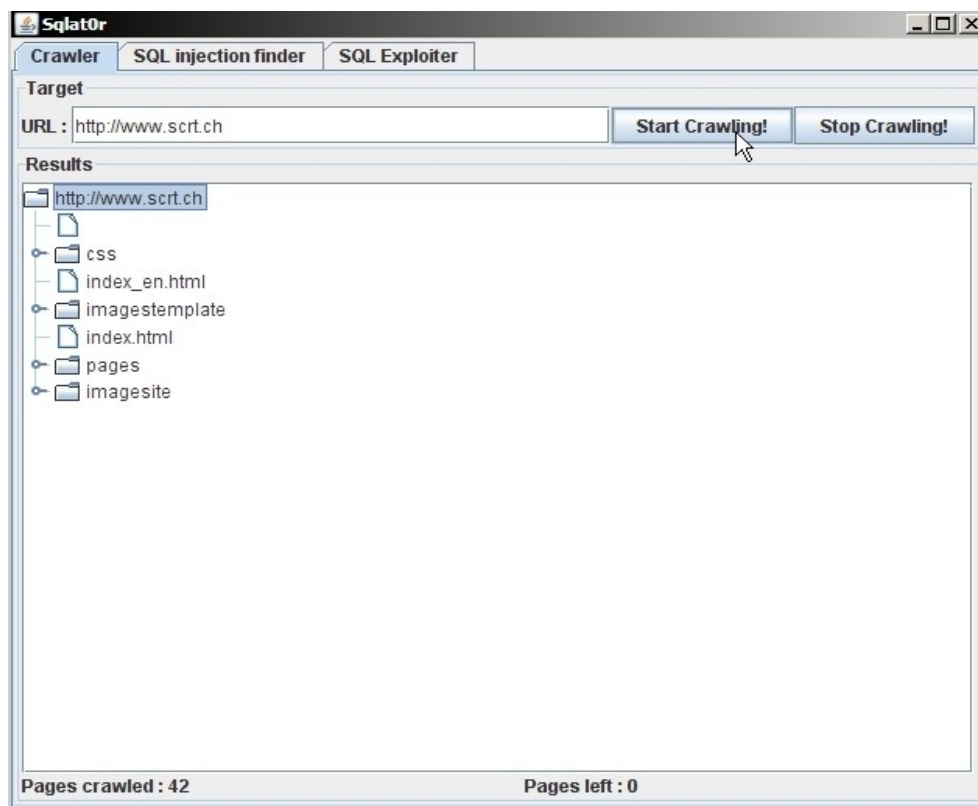
Using Mini MySQLat0r is very simple. The three different modules are available as tabs at the top of the application.



Most of the time a user will start from the Crawler module and then go on to the Testing module and finally the Exploiter module as information from each module can help in using the next.

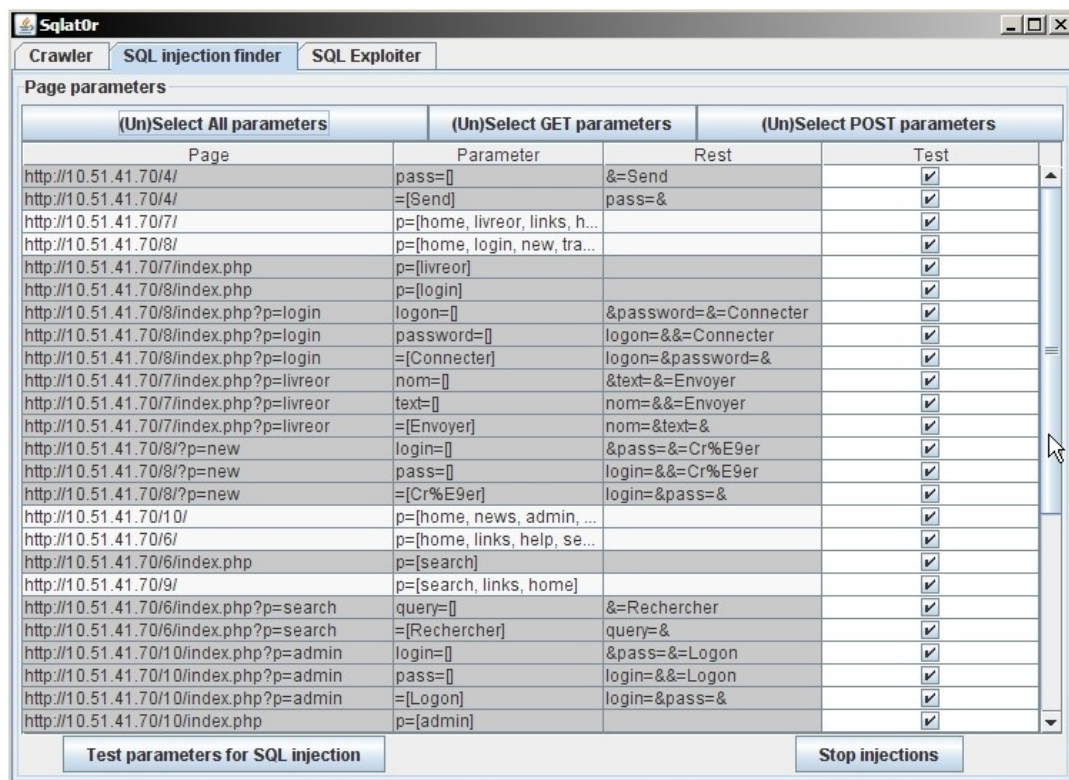
'03 AND 1-- Crawler Module

The crawler module as its name suggests is used to crawl a website, or part of a website. The user must simply input the target URL in the designated area and then click on « Start Crawling ». The result should look like following image.



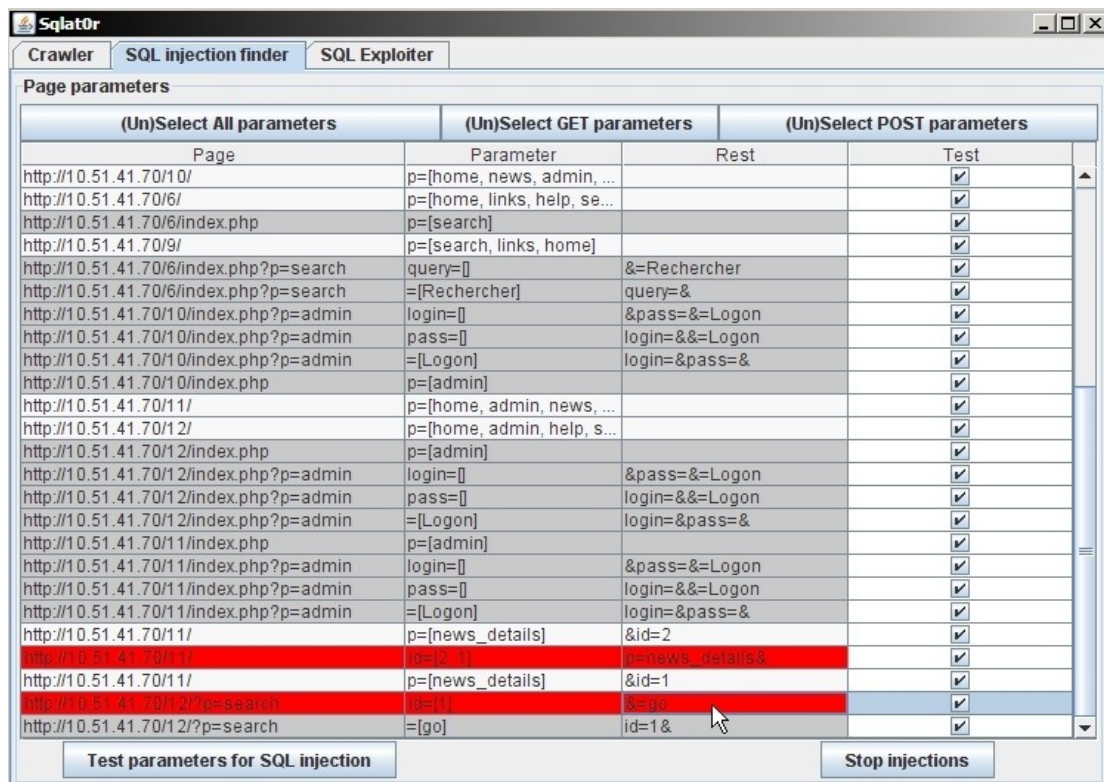
'03 AND 2-- Tester Module

Once a site has been crawled, all pages containing dynamic parameters are shown in the Tester module as seen below.



Pages in dark grey are accessed by POST request instead of GET. They are therefore usually associated to forms found on the different pages. To test a parameter for injection, the user must check the « Test » box associated to the desired parameter. The top buttons allow a user to quickly select or unselect all parameters, or only GET or POST ones.

Once clicked, the « Test parameters for SQL injection » will launch the discovery attacks to detect if a parameter is vulnerable. If it is the case, the corresponding line will be highlighted in red as shown below.



By clicking on one of the parameters, all its information is sent to the Exploiter module to make the exploitation simpler.

'03 AND 3-- Exploiter Module

The exploiter module is the part of the program that exploits an SQL injection vulnerability. If the vulnerability was found by using the Tester module, a simple click on the given line in the Tester module will set all required parameters in the Exploiter module. Otherwise all parameters must be entered manually.

The screenshot shows the 'Exploiter' module configuration window. It has several fields: 'URL' with the value 'http://10.51.41.70/11/', 'Vulnerable parameter' with 'id', 'Default value' with '2', 'Other arguments' with 'p=news_details&', 'Method' with 'GET', and 'Injection Type' with 'Numerical with comments'.

The injection type parameter corresponds to the type on injection that will be used. This depends on the type of field that is being exploited (numerical or literal) and whether the query must be ended with a comment or not. Other values are pretty straightforward.

The options panel allows the user to specify what kind of injections will be attempted against the website. « Get all database information » will attempt to gather table and column information from the database. Other options are straightforward.

If the injections are successful, a result similar to the following image should be visible.

The screenshot shows the 'Results' window. It contains text indicating the results of the initial query: 'Found 5 columns in initial query.', 'Column 2 is displayable.', 'User : formation@localhost', 'Version : 5.0.32-Debian_7etch4-log', 'Found 26 rows for table csrf.users', and 'Found 2 rows for table sqlinj.news'. Below this text is a table with 4 columns: 'Schema Name', 'Table Name', 'Columns', and 'Dump!'. The table lists four entries: 'csrf.users' with columns '[id, name, password, am...]', 'sqlinj.news' with columns '[id, title, date, time, descri...]', 'sqlinj.users' with columns '[id, login, pass]', and 'xss.livreor' with columns '[id, nom, text, date]'. Each entry has a 'Dump!' button next to it.

Schema Name	Table Name	Columns	Dump!
csrf	users	[id, name, password, am...	Dump!
sqlinj	news	[id, title, date, time, descri...	Dump!
sqlinj	users	[id, login, pass]	Dump!
xss	livreor	[id, nom, text, date]	Dump!

By clicking on « Dump! », all information in the corresponding table is retrieved and displayed. If file retrieval is successful, the content of each file is displayed in a new frame.

The screenshot shows a window titled 'Content of sqlinj.news'. It contains a table with 5 columns: 'id', 'title', 'date', 'time', and 'description'. The table has two rows of data.

id	title	date	time	description
1	Lancement sit...	2008-05-28	14:01:42	Site de news l...
2	La séc...	2008-05-28	14:08:37	Tout est s&ea...