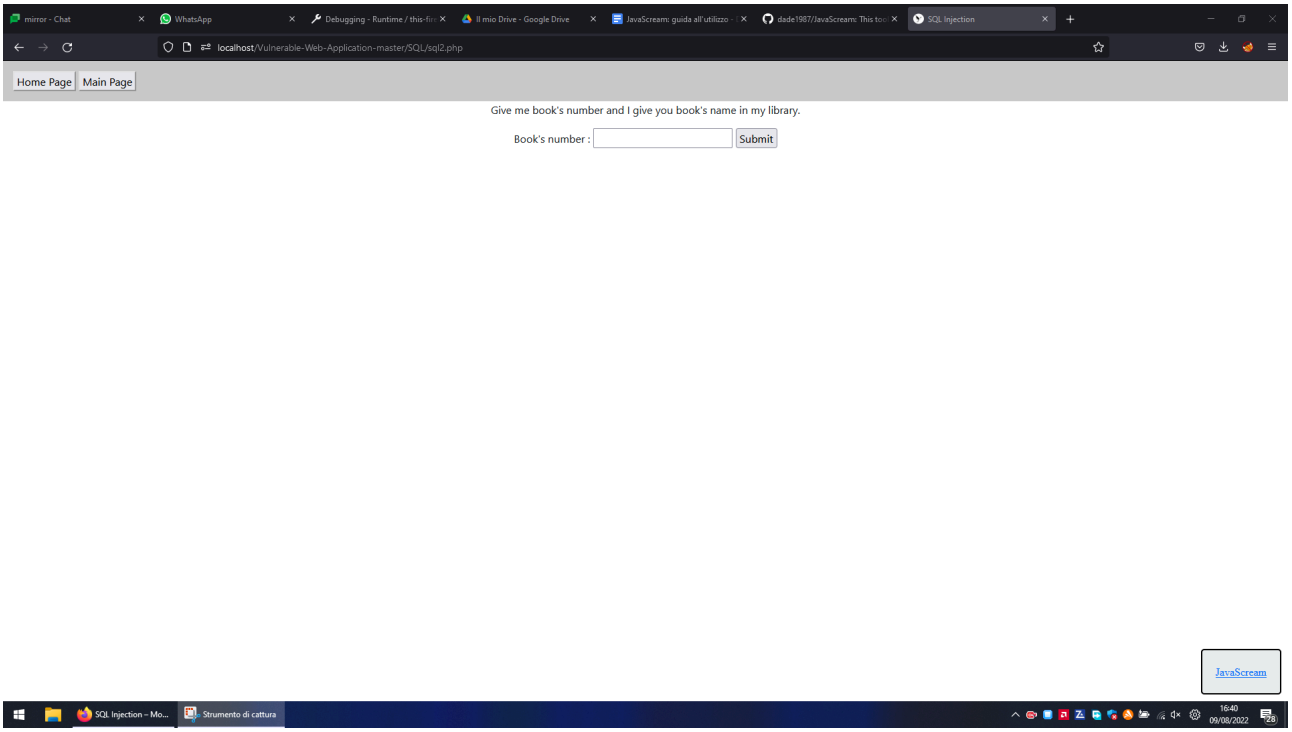
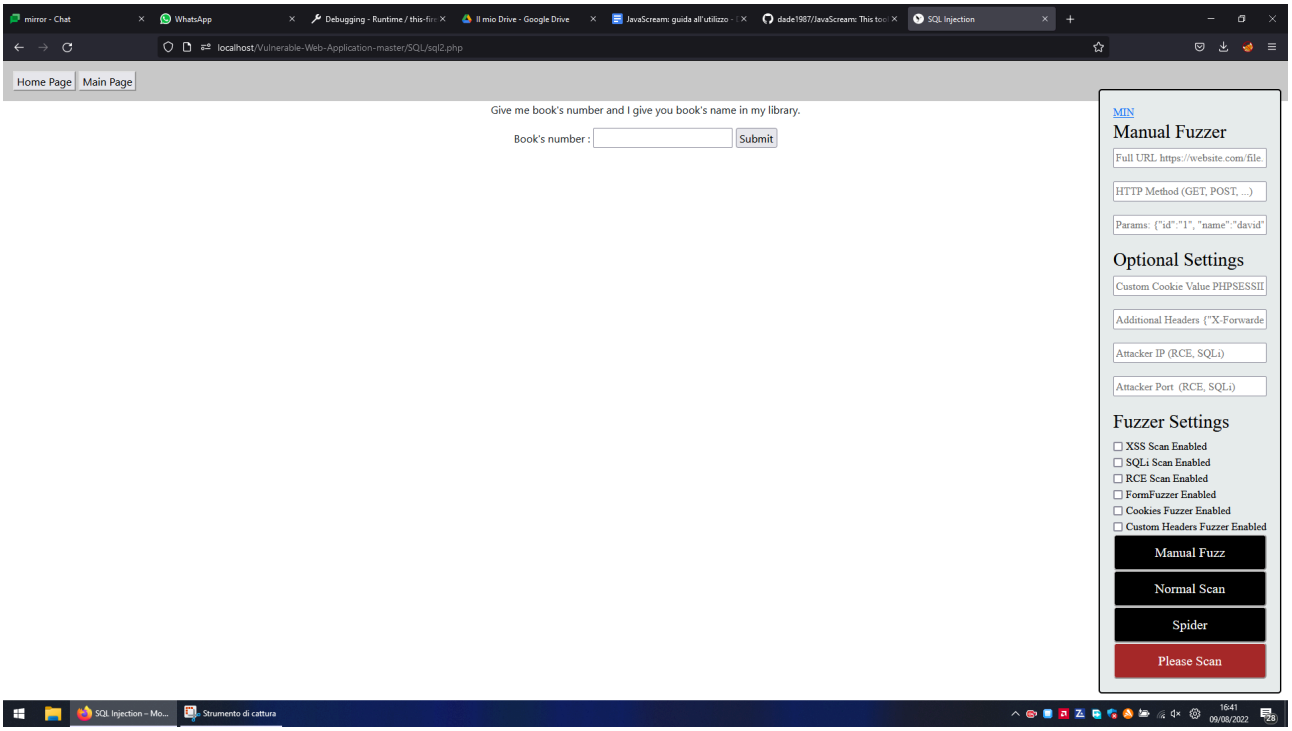
INSTALLATION

- Clone the repository with ‘git clone https://github.com/dade1987/JavaScream.git’
- Run “Mozilla Firefox”
- Type ‘about:debugging’ in the URL bar
- On the left menu click ‘This Firefox’
- Under ‘Temporary Extensions’ click ‘Load Temporary Add-On’’
- From the cloned folder select the file ‘manifest.json’

Now you’ll see a rectangle on the bottom right side of the page.



Click ‘JavaScream’ inside the new rectangele. You’ll show you all the tool’s functionalities.
Click ‘MIN’ in the top side of the rectangle to re-minimize the window.

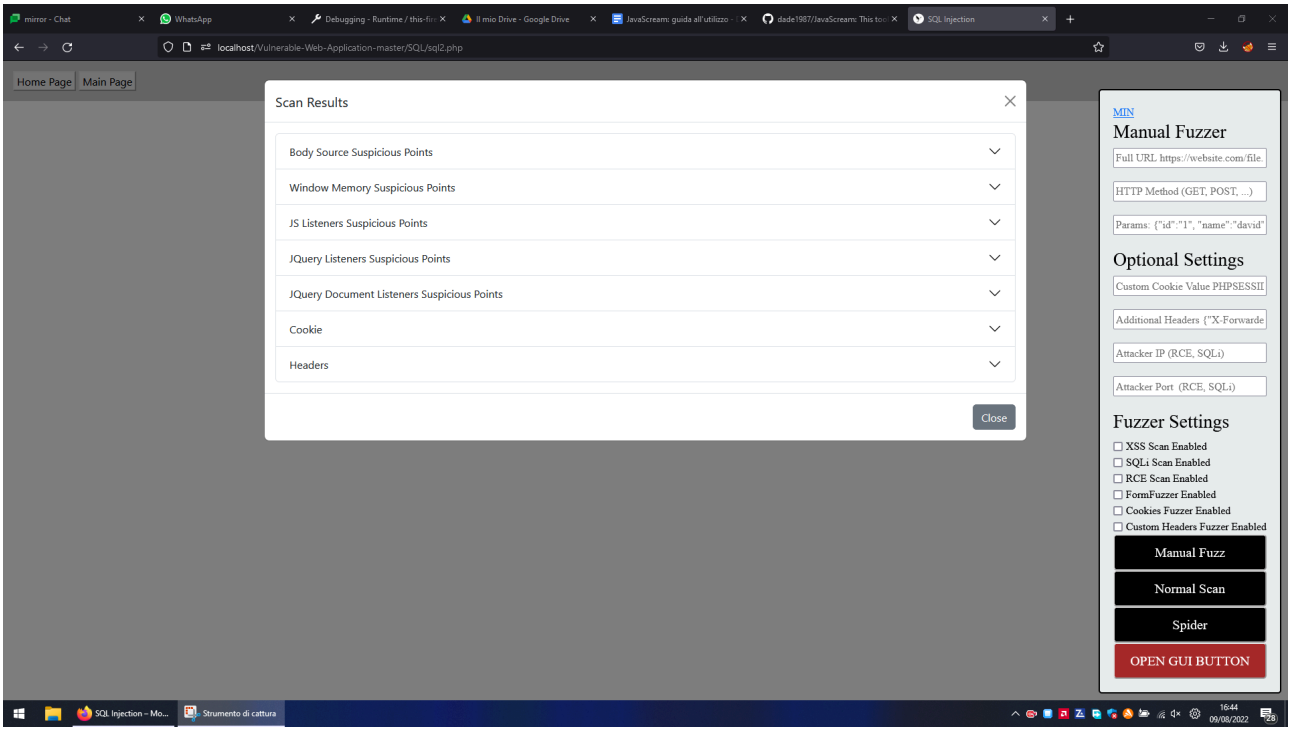


Ok, let's start with the explanations.

PASSIVE PAGE ANALYSIS

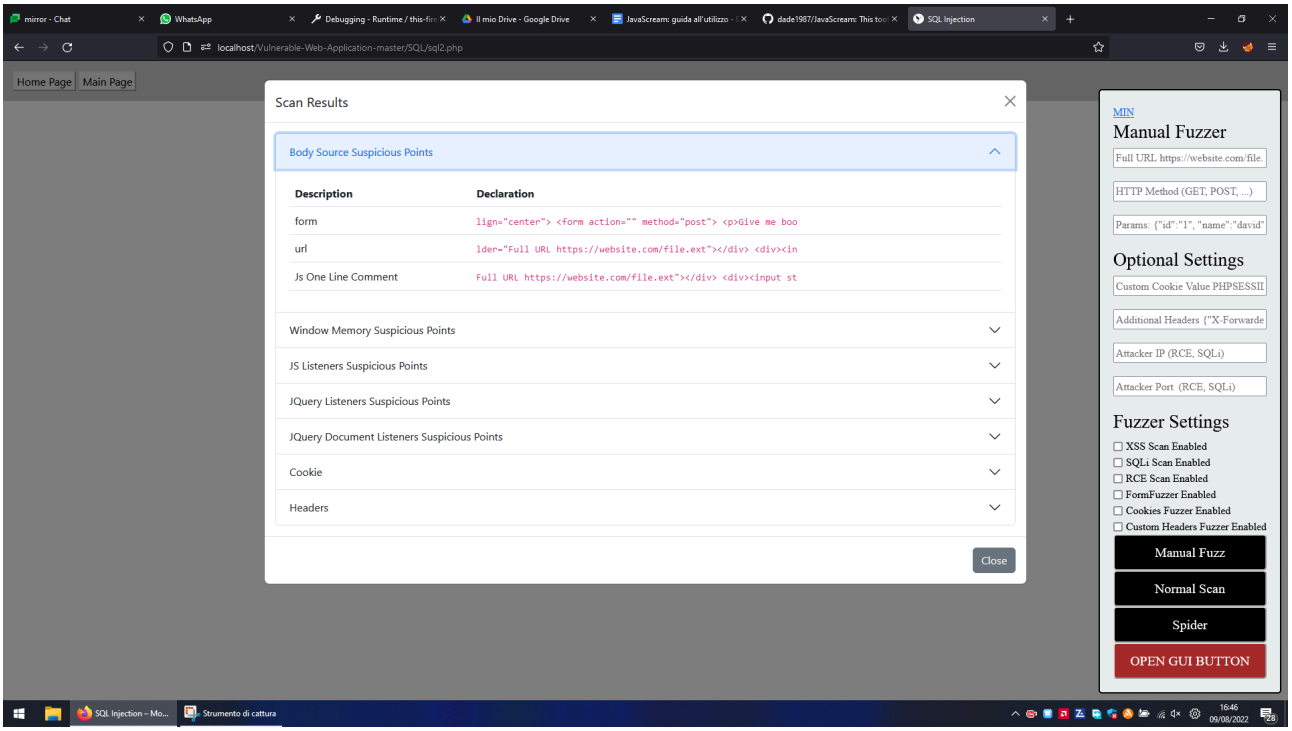
To make a passive page analysis:

- don't select any checkbox and click "Normal Scan"
- the button "Please Scan" will become "Open Gui"
- click "Open GUI" and you'll see the following window:

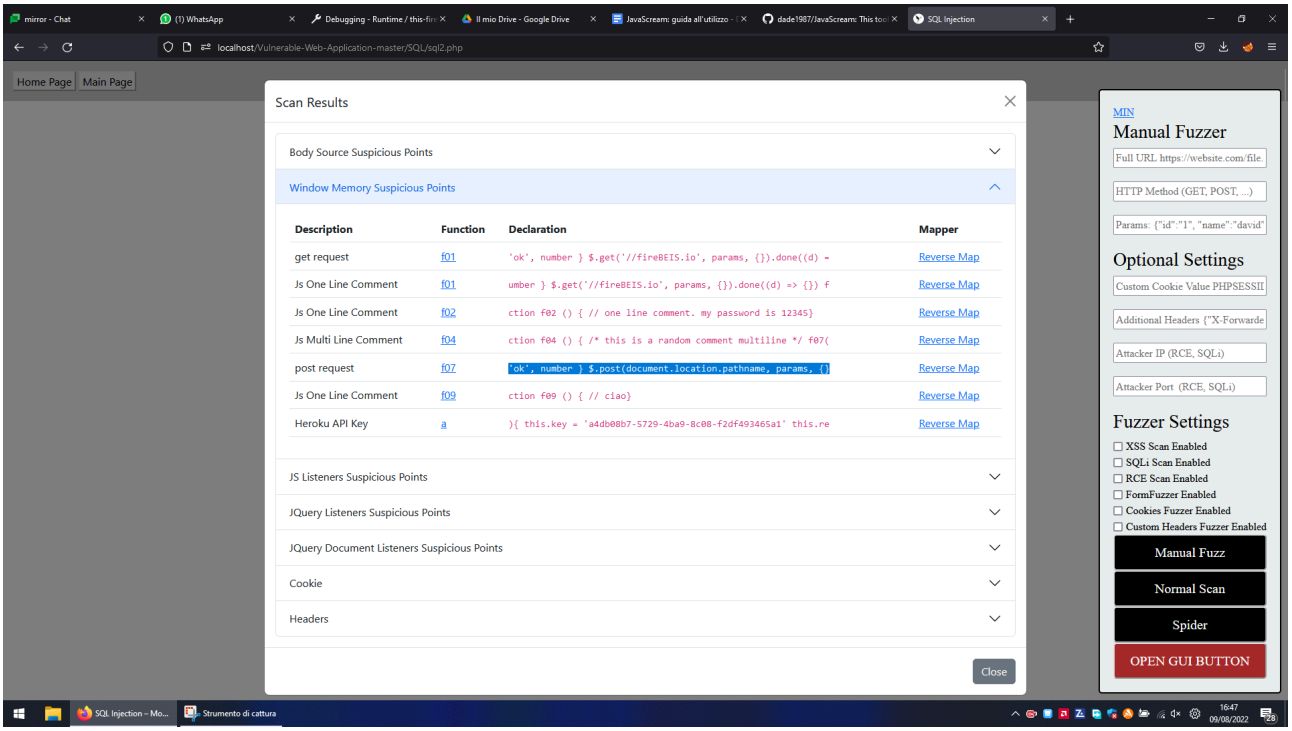


Actually we are seeing:

- the most significant code in the page body

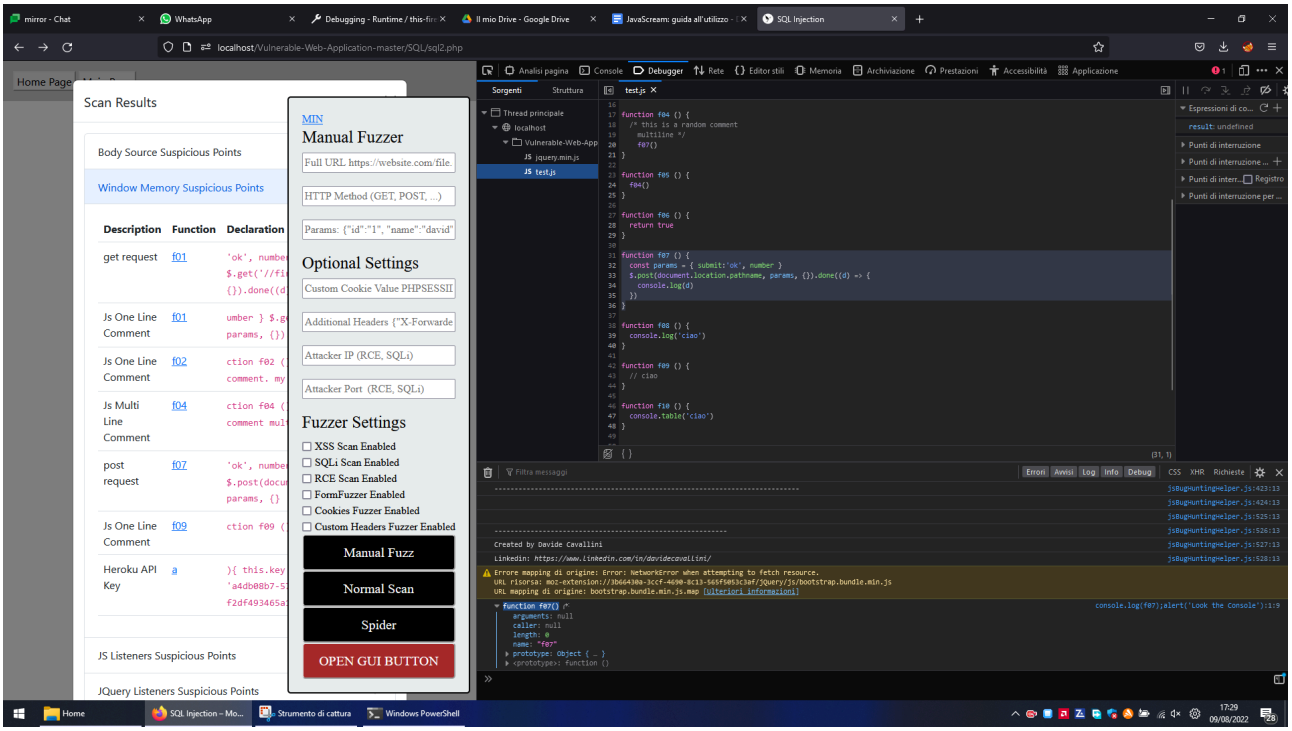


- the most interesting javascript loaded inside the current page

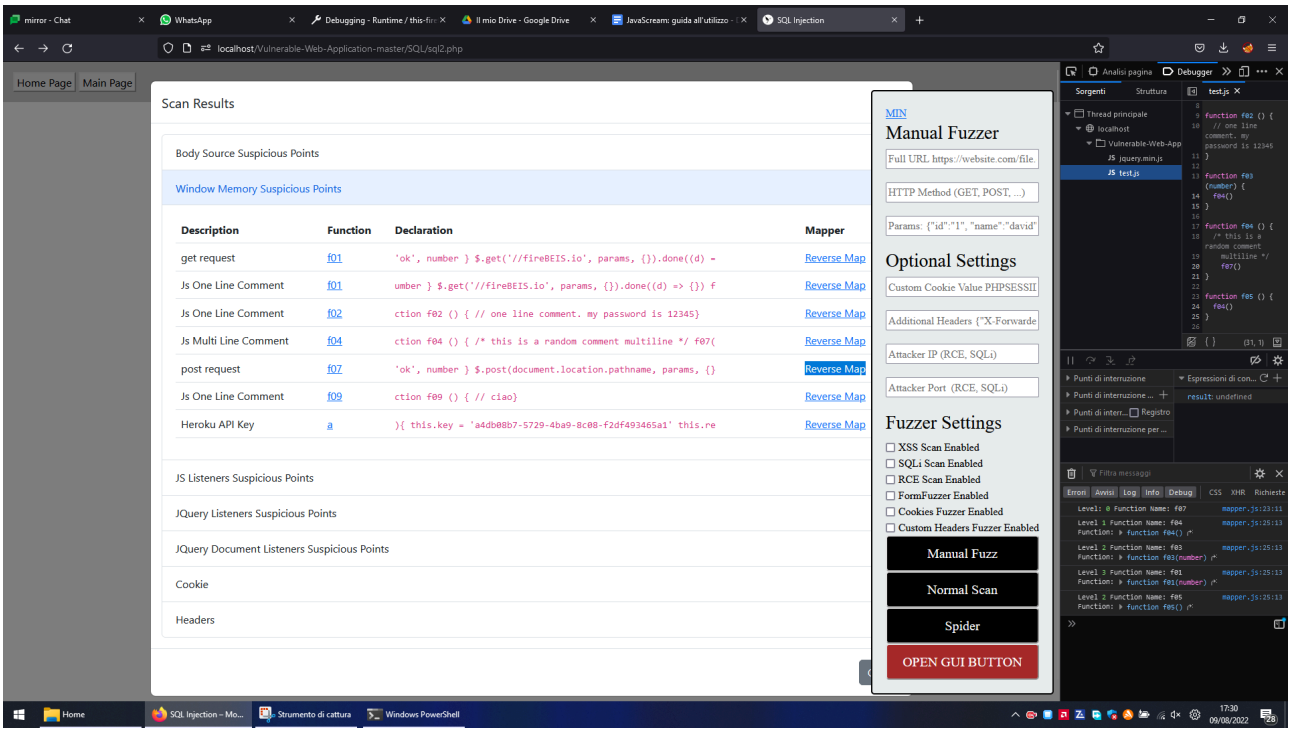


We can also open a little pathentesis.

If you click on the function name, opening the Developer Tools with the F12 key, you'll see the function name with a little arrow on its right. If you click that small arrow then the debugger will show you the function definition without time losses:

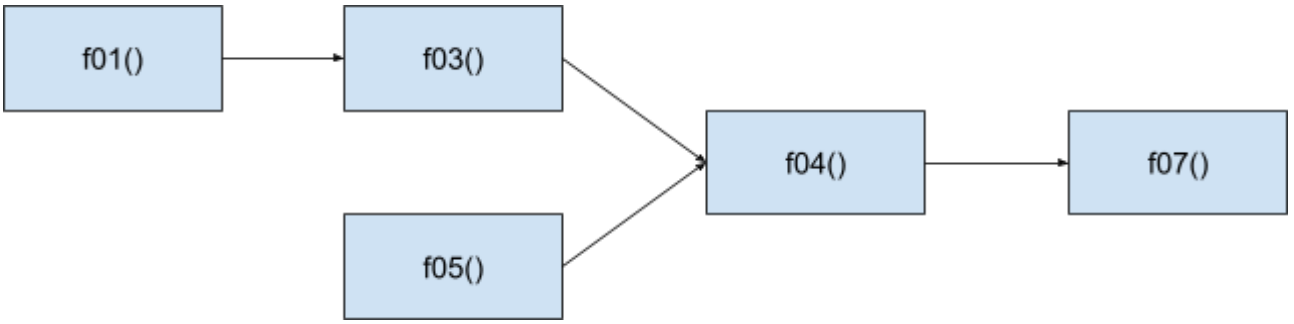


If you click the link “Reverse Map” you can see which functions go into the called method, and you’ll see the Entry Point of the called method itself:

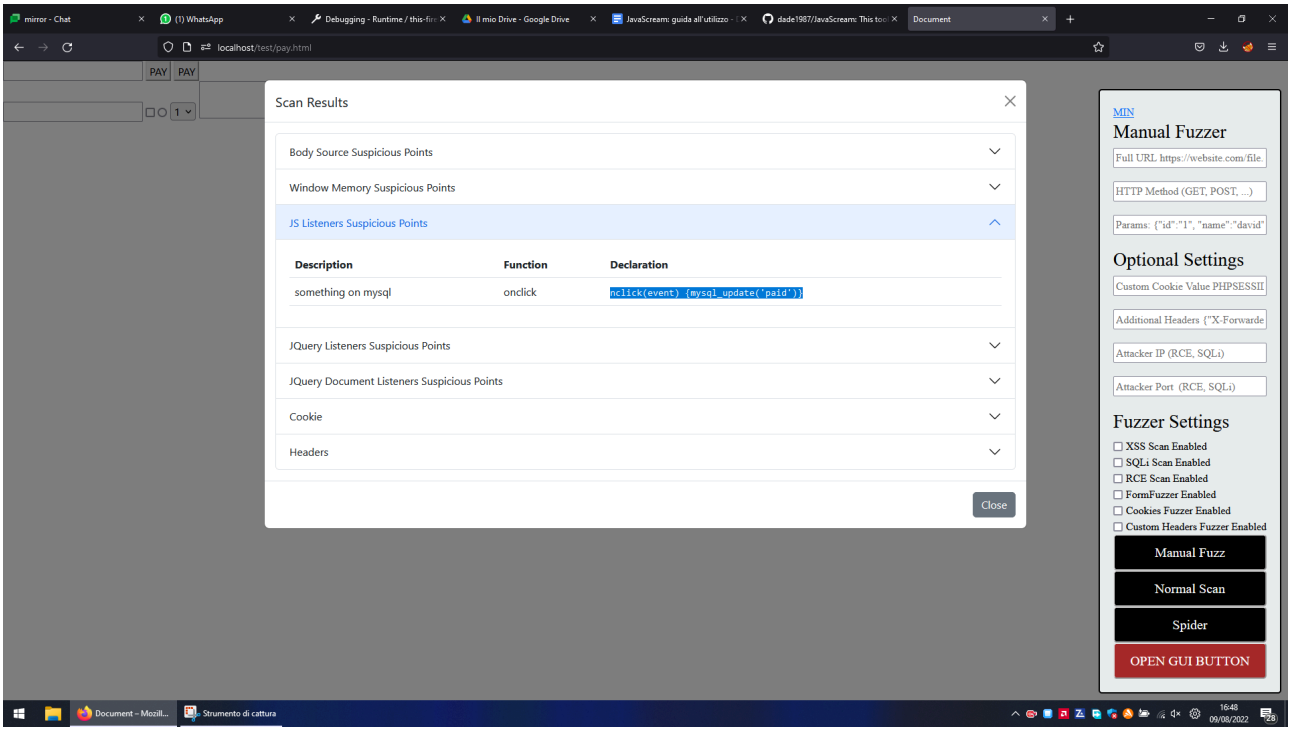


In this case f07() is in depth 0 because it’s just called. f07() starts from f04() at depth 1. f04() can starts from both f03() or f05 at depth 2. f03() starts from f01() at depth 3.

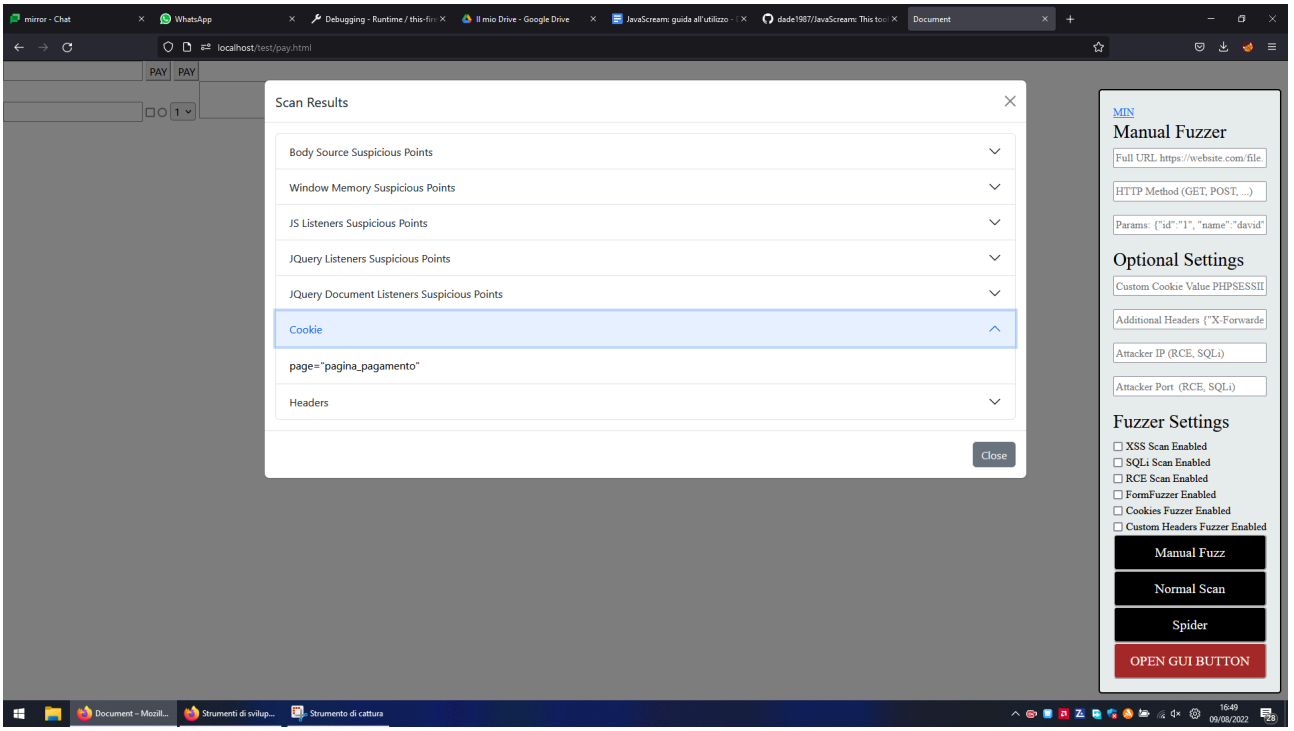
We can represent the call stack using this scheme:



- here you can see the JS, jQuery and document listeners, containing suspect code



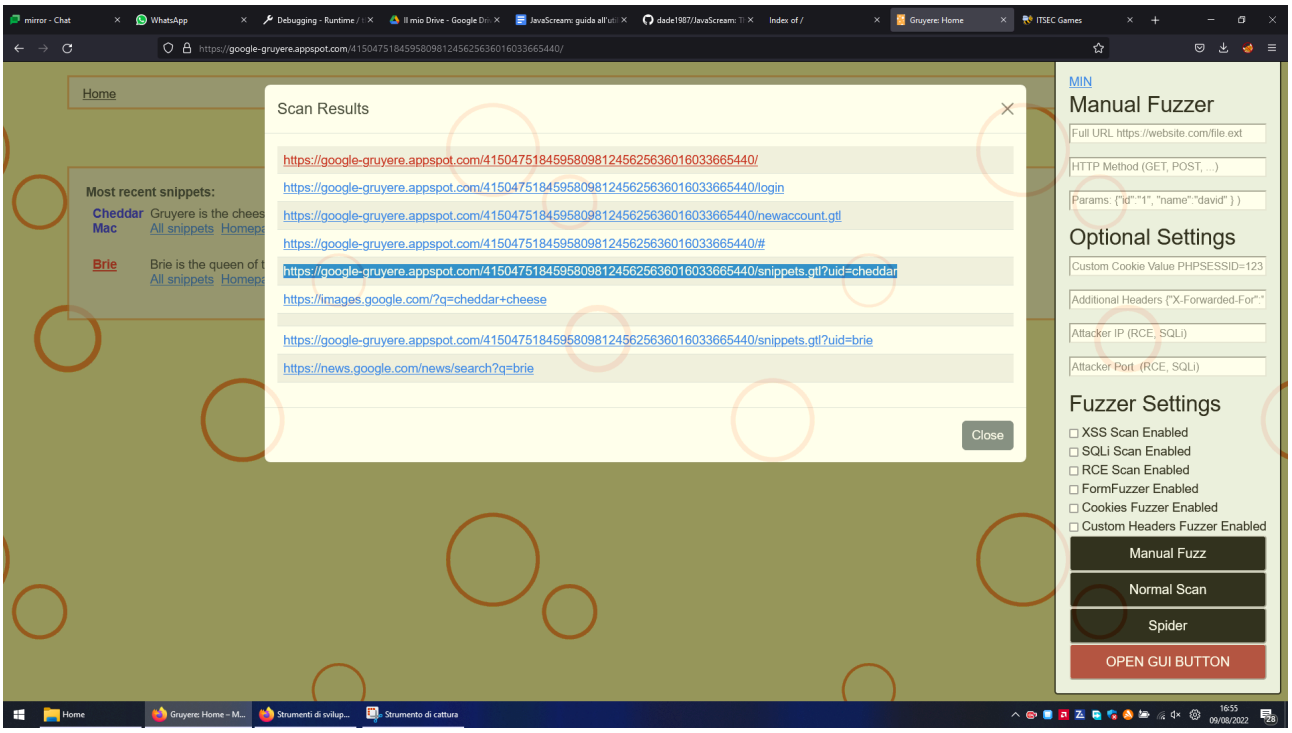
- here you have page's haders and cookies



If you want to see what resources are recursively called from the current page, we have also another scanner: the “Spider”

In this example we are using a vulnerable website expressly studied for pentesting.

In the results we can see some links. Someone contain some GET params, too.

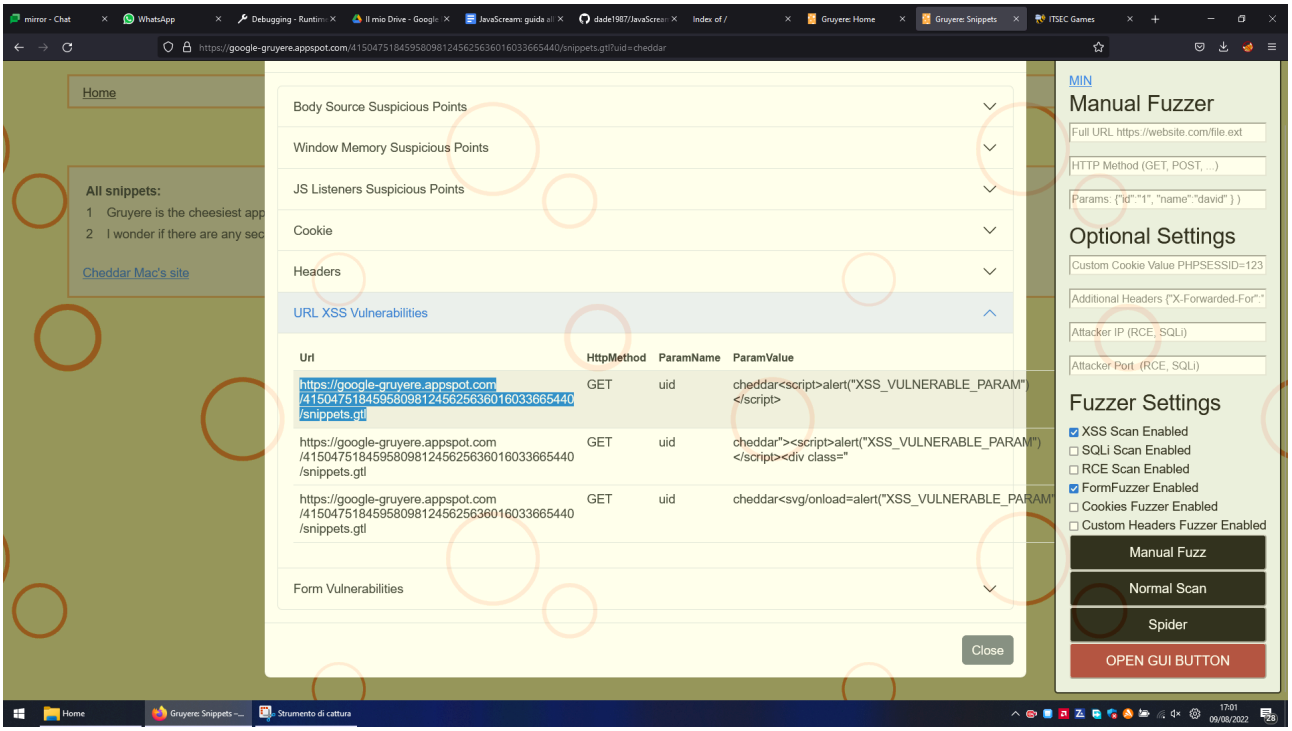


ACTIVE PAGE ANALYSIS

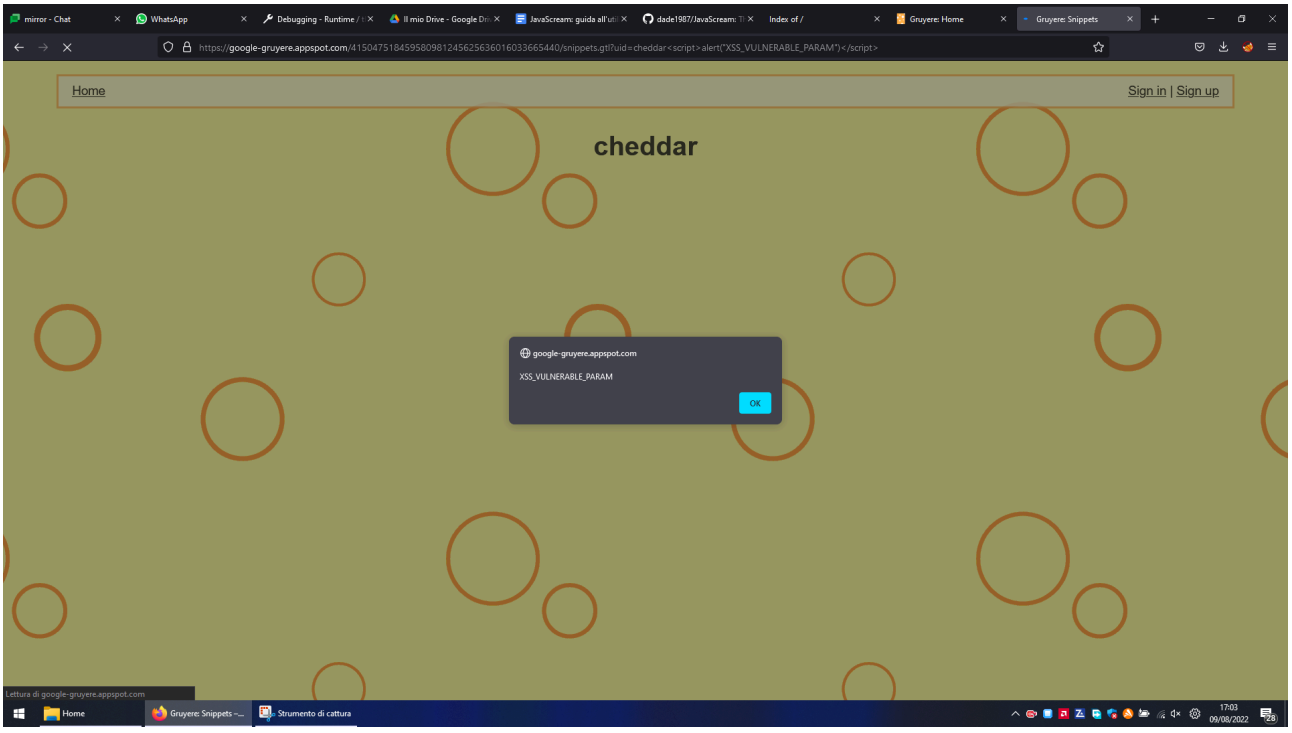
We have two types of Active Scanners:
“Manual Fuzzer” and “Normal Scan with checked boxes”

Let’s start from Normal Scan: it makes **undifferentiated scanning of the whole page.**

- Let’s visit the url “https://google-gruyere.appspot.com/415047518459580981245625636016033665440/snippets.gtl?uid=cheddar”
- Let’s click “XSS Scan Enabled” and “Form Fuzzer”
- Let’s run “Normal Scan”
- Let’s click “Open GUI”
- Clicking the section “URL XSS Vulnerabilities” this window will appear:



- let's see immediately the url "https://google-gruyere.appspot.com/4150475184595809812456256360160336654 40/snippets.gtl" contains a XSS vulnerability on the param "uid"
- Let's try to open the UR copying the first value associated with the paramName, and we'll obtain:



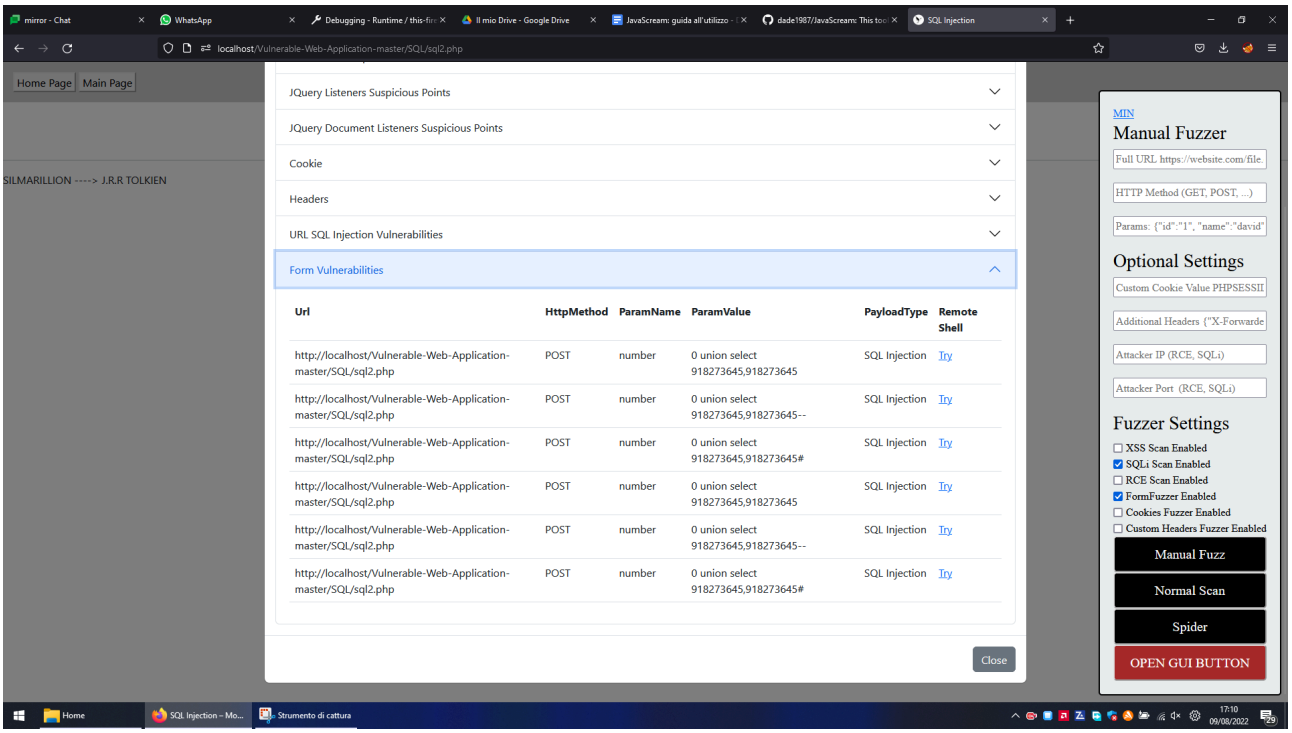
Ok, the site is really vulnerable! The page doesn't contain forms, then the form fuzzer didn't run. Then the section "Form Vulnerabilies" was empty.

Let's move on to something more crunchy… here we have a simple form. If we write an “id” we’ll see a book name associated with it, and an author, too. There aren’t URL (or GET) params in this page.

Let’s see if the database is well protected…

Check the box “SQLiScan Enabled” and “Form Fuzzer”, while we are working with a “form”.

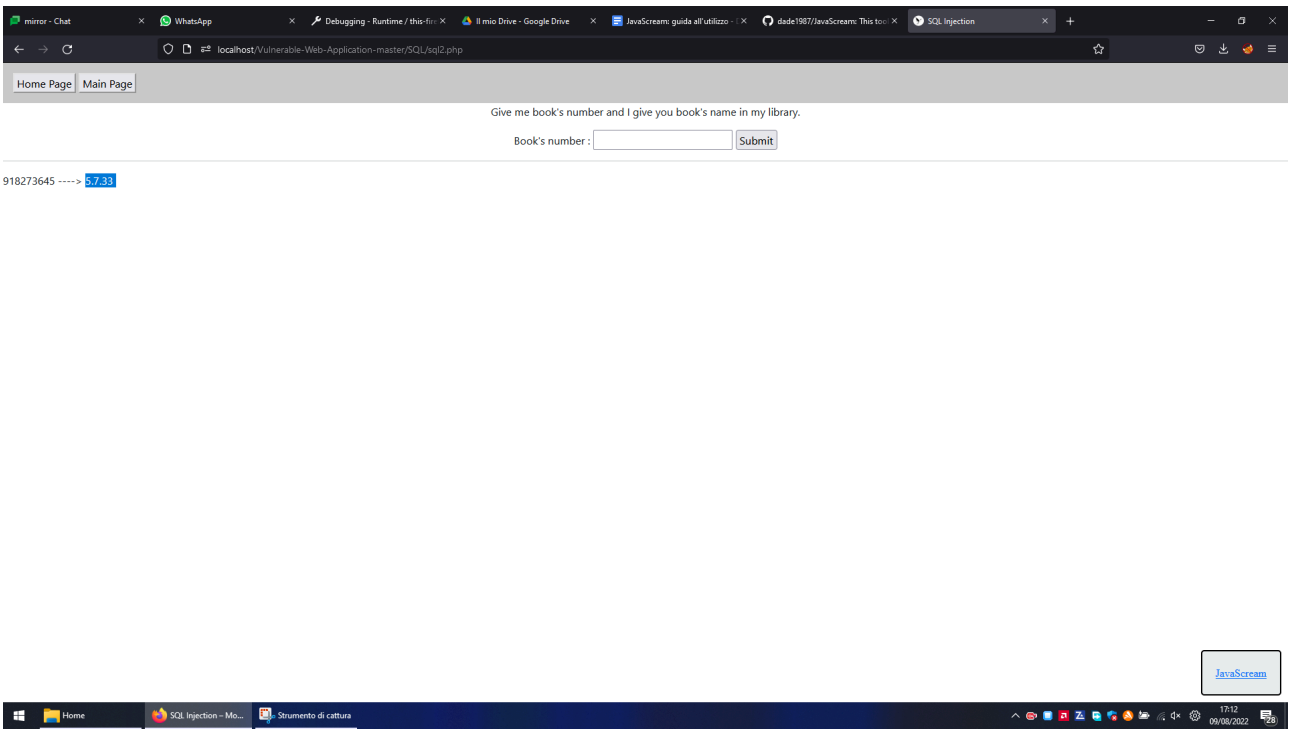
Now “URL SQL Injection Vulnerabiliteis” will be obiviously empty, but look at the “Form Vulnerabilities” section:



The “form action script” is vulnerable to SQL injection!

Try to copy the payload inside the “form”, and replace the last union select value with version().

We’ll see the MySql version!

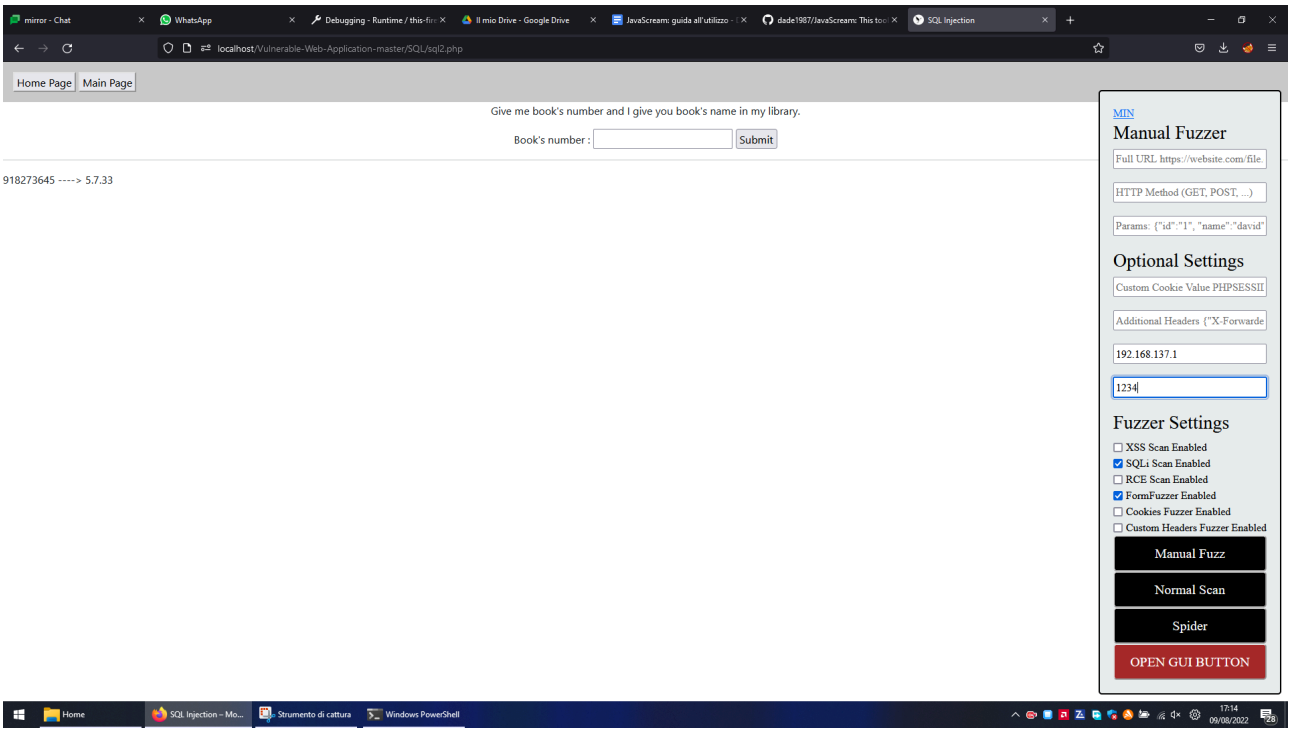


The website is certainly vulnerable to SQL Injection, a very dangerous vulnerability.

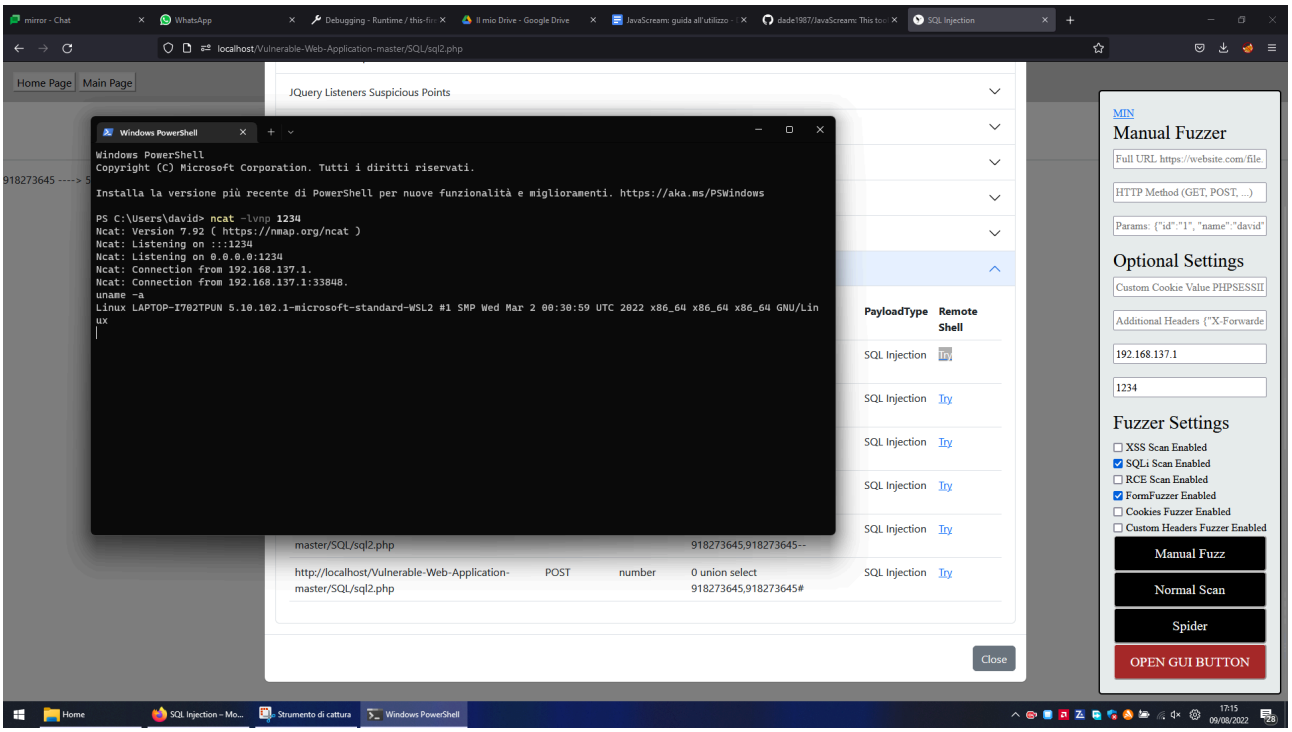
Now let's try another thing. Sometimes we can obtain a direct access to the server bash, only exploiting the SQL database.

Let's open "netcat -lvnp 1234" (or ncat in windows).

Let's set the attacker IP e attacker Port (your ip and a random port - in this case 1234)



Let’s click “Try”

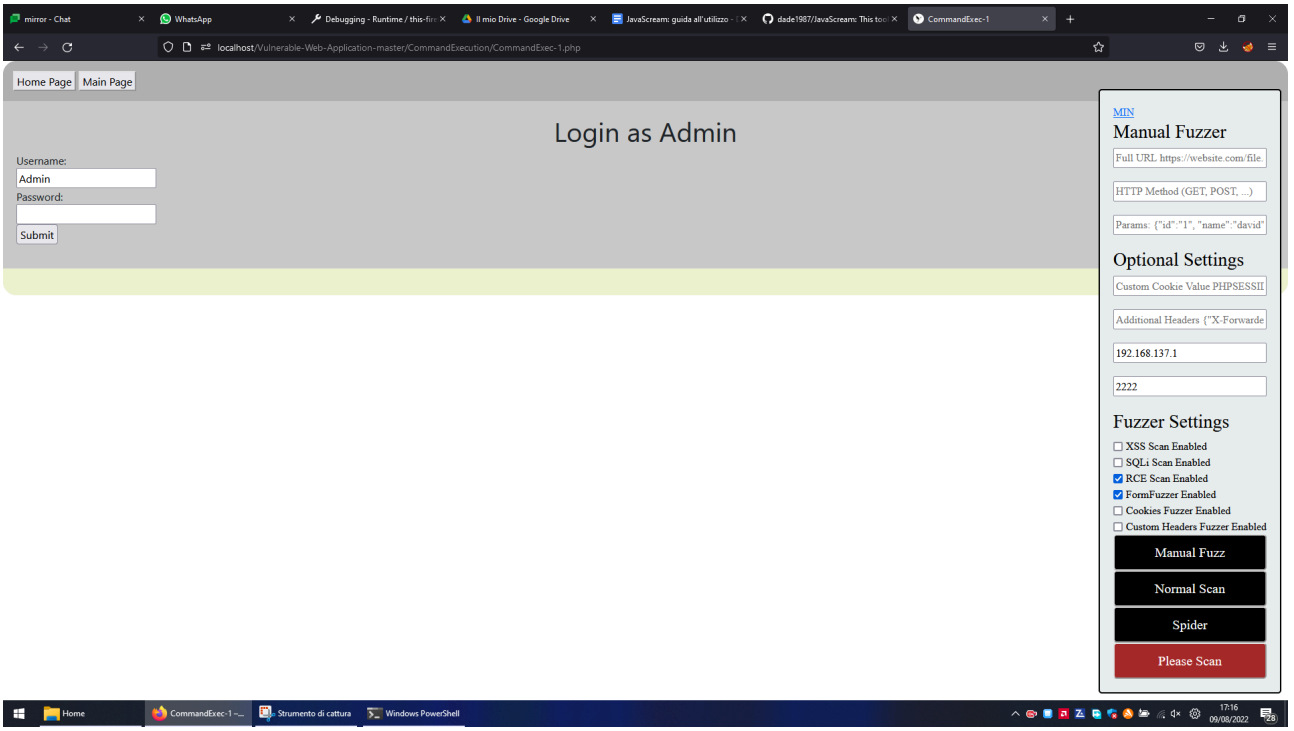


Et voilà. We have obtaind the full server access, only by exploiting MySql. Now we can make all operations inside the server, or we could also try to escalate privileges.

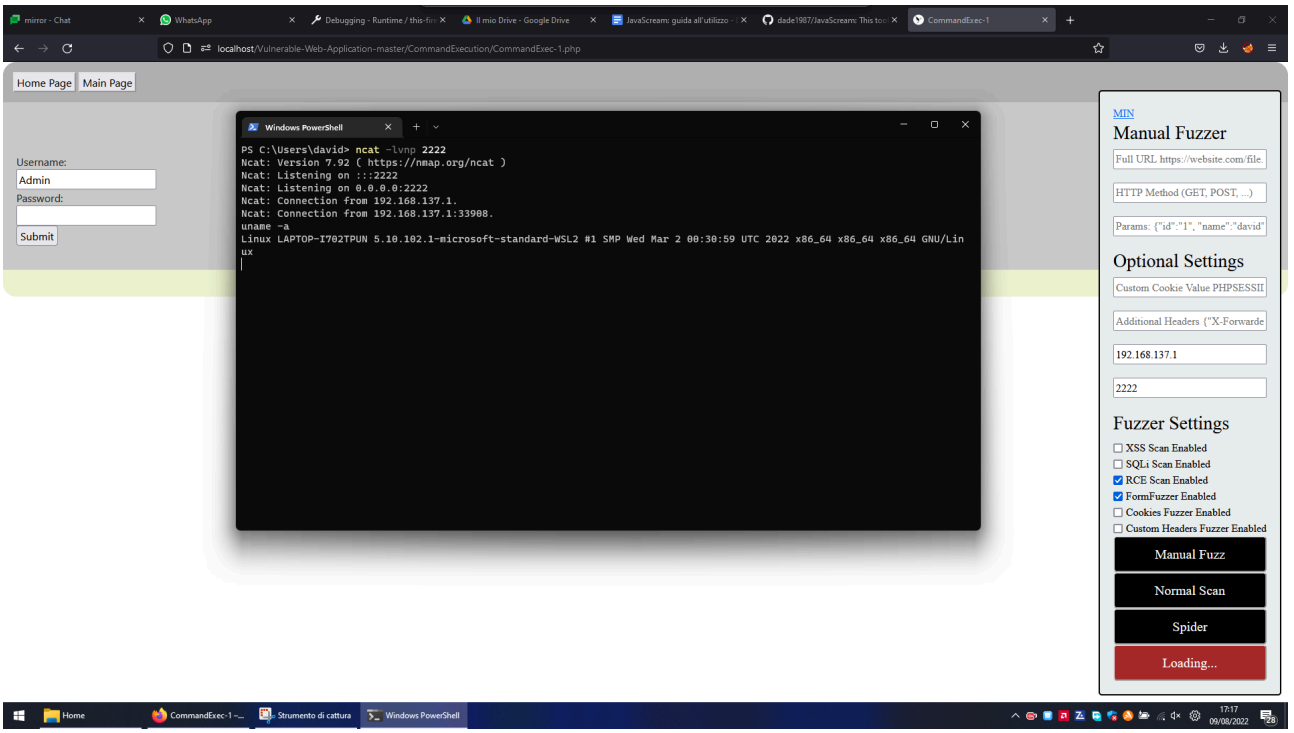
Now let’s try to look for Remote Code Execution vulnerabilities inside another page.

Let’s open netcat -lvnp 2222 and set Attacker Ip and Port (in this case the port will be 2222).

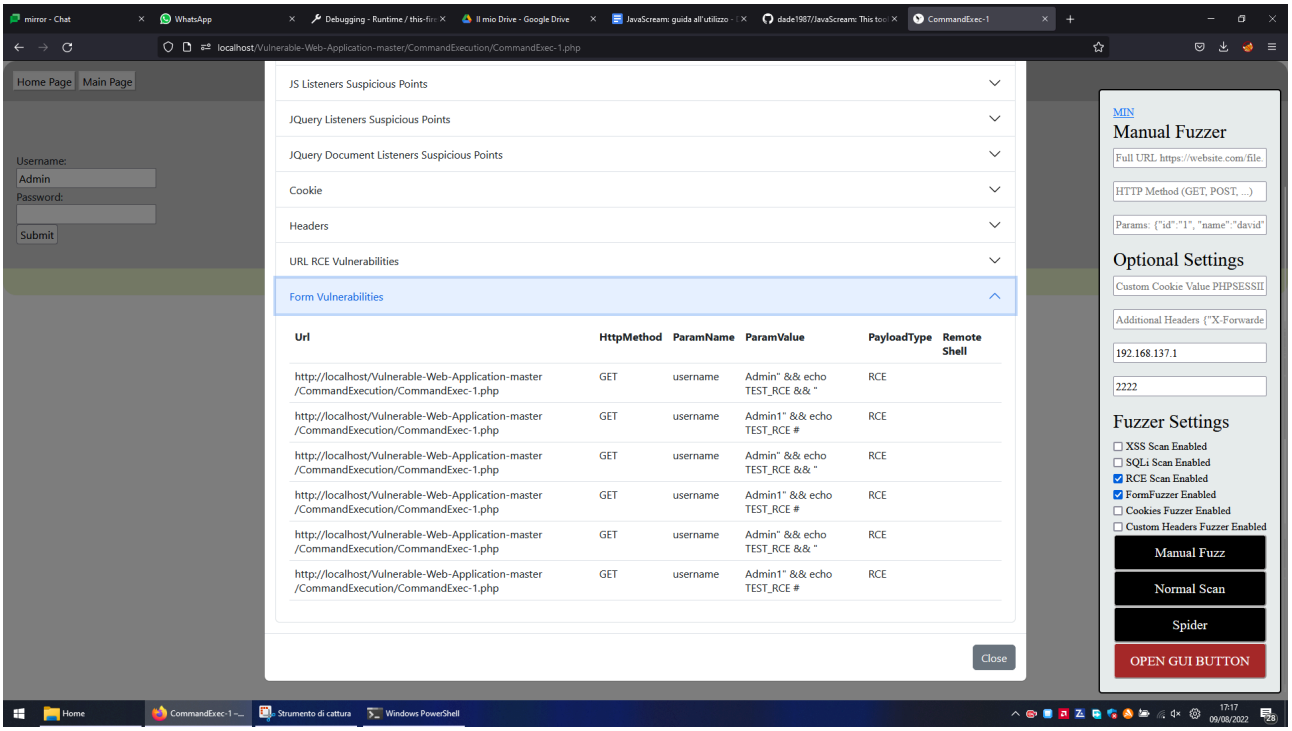
Let’s check the “RCE Scan Enabled” and “FormFuzzer Enabled” boxes.



Let's run "Normal Scan"

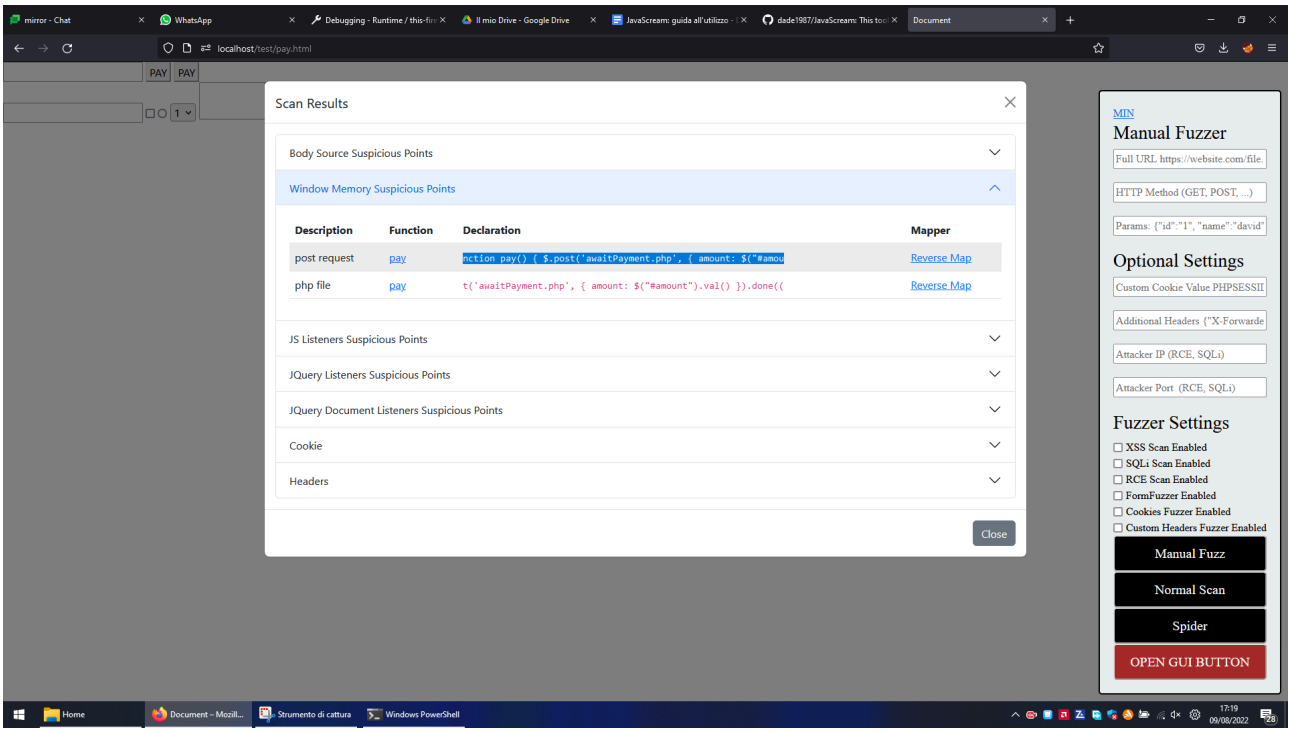


We are inside the server again. If we didn't set the attacker Ip and Port, we obtained all the vulnerable params inside the forms, into the section "Form Vulnerabilities", exactly like this:



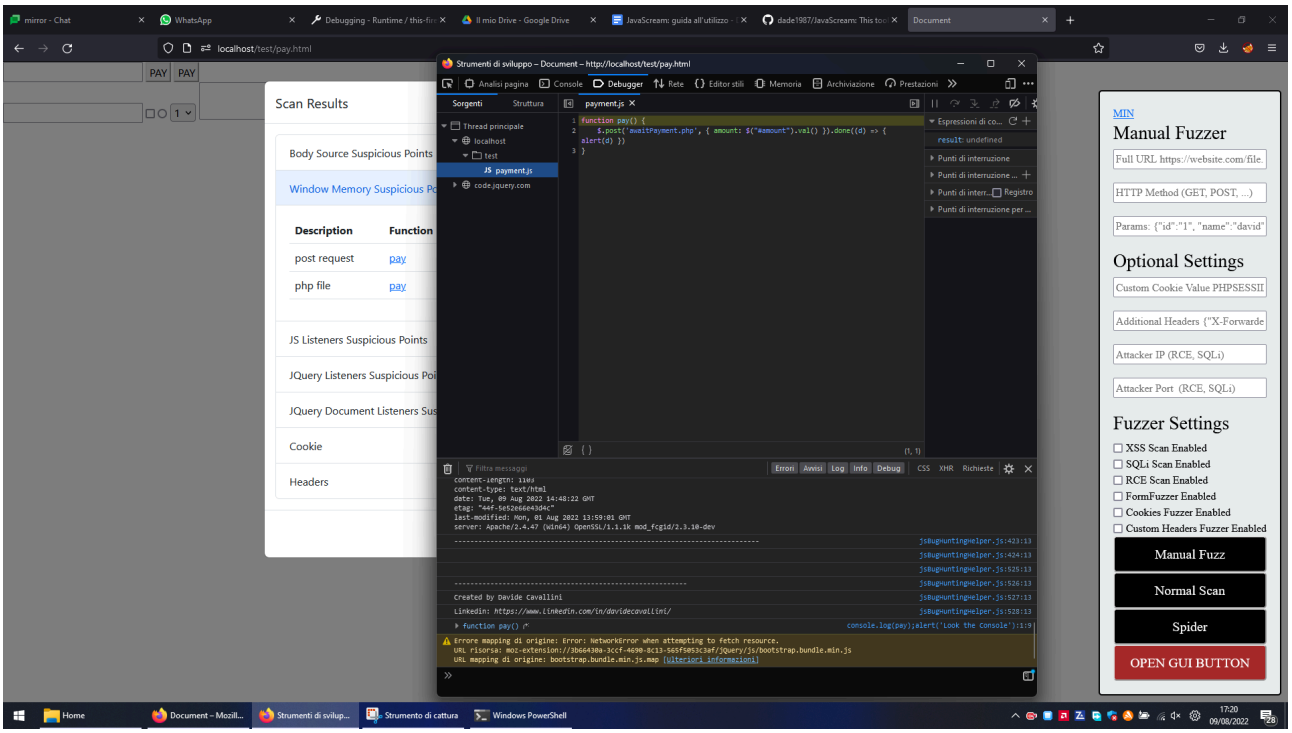
Ok. Let's make another example:

in the suspect elements inside the "Window Memory Suspicious Points" we found this particular method:



"nction pay() { $.post('awaitPayment.php', { amount: $("#amou"

Let's click on "pay" and we see this method definition:



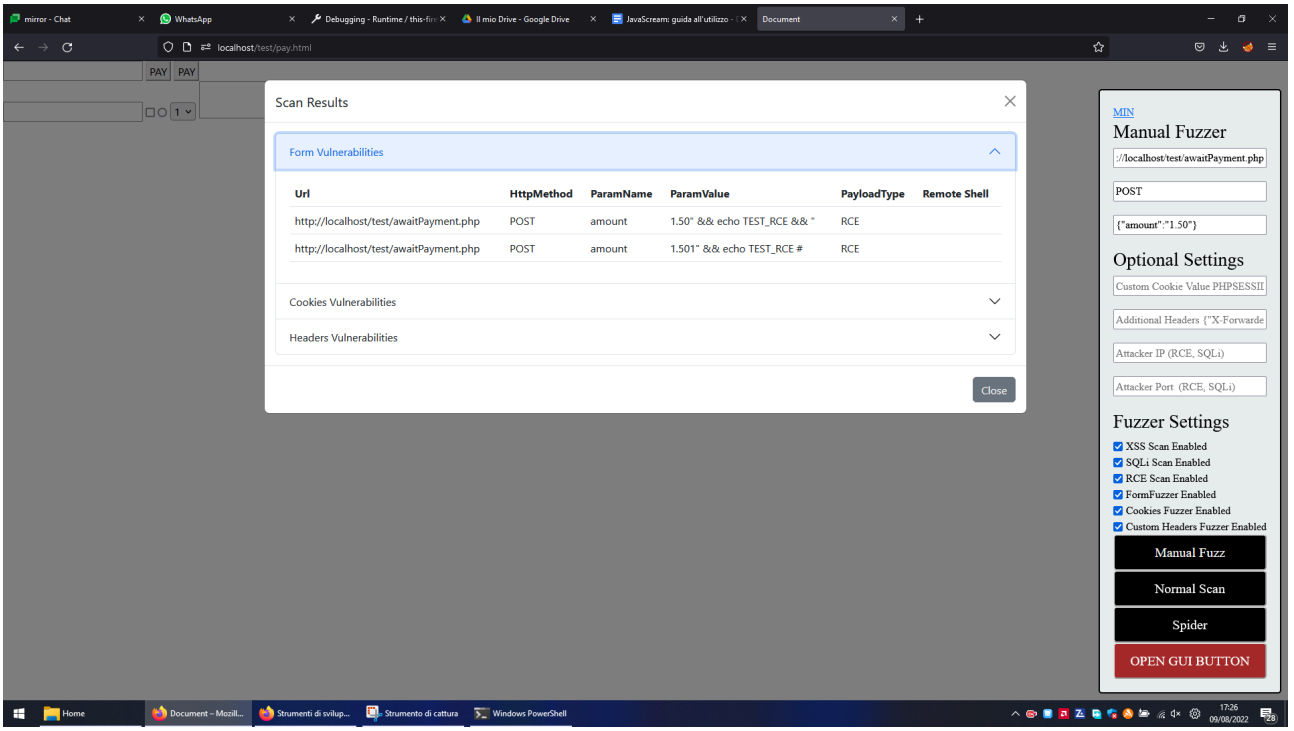

```
function pay() {
        $.post('awaitPayment.php', { amount: $("#amount").val() }).done((d) => { alert(d) })
}
```

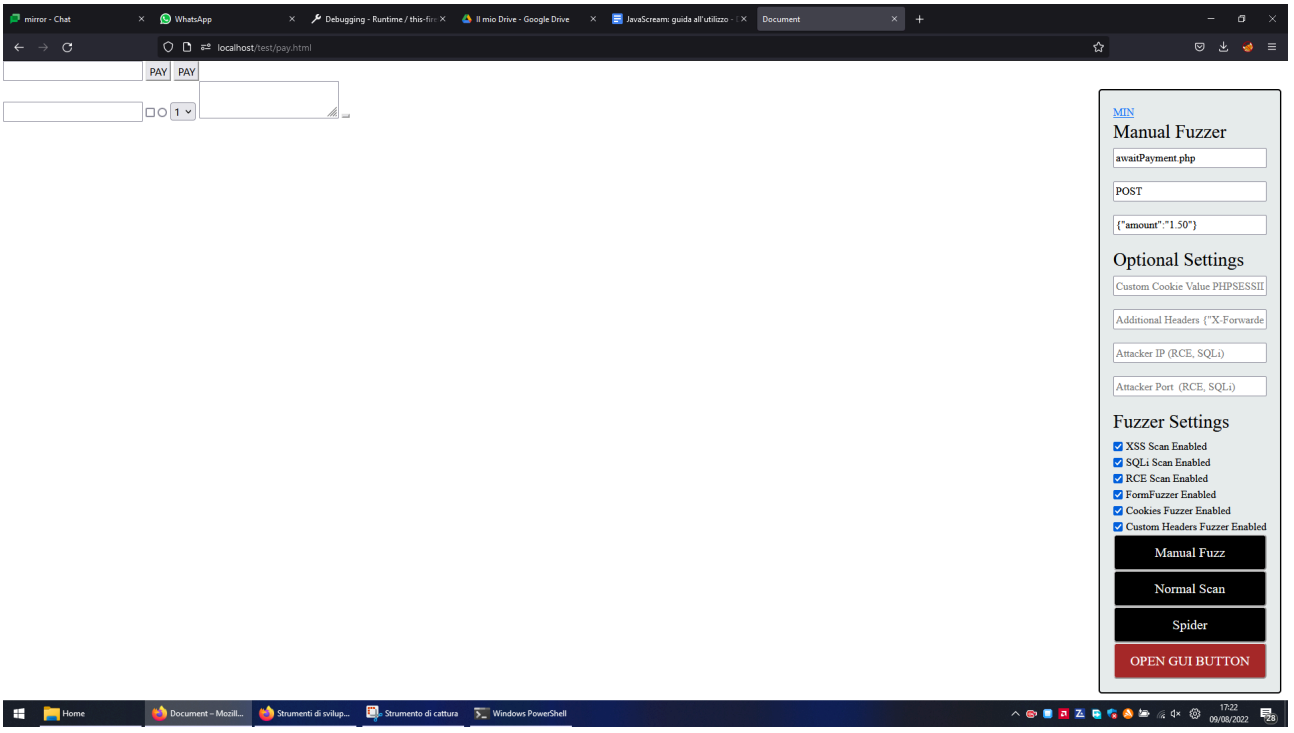
How can we effortlessly test it?

- Let's write "http://localhost/test/awaitPayment.php" inside the "**Full** URL" input
- Let's set the "HTTP Method" to post, because it's "$.post()"
- Let's set the "amount" param into the "Params" section, like a JS Object: Example: {"amount":"1.50"}
- Let's check all the checkboxes
- Now let's click "Manual Fuzzer"

Because we checked all the checkboxes, the tool will try to inject all the payloads inside the page params, into the cookies and eventually also additional headers.

Let's open the GUI and we'll get the following result:





In this case the page is vulnerable to RCE, then we could take the full server access again.

NOTES

The software is studied to be easily implemented

If you want to collaborate on our Open Source project, write to Davide Cavallini, via a message to the Linkedin profile https://www.linkedin.com/in/davidecavallini/

Good fun!