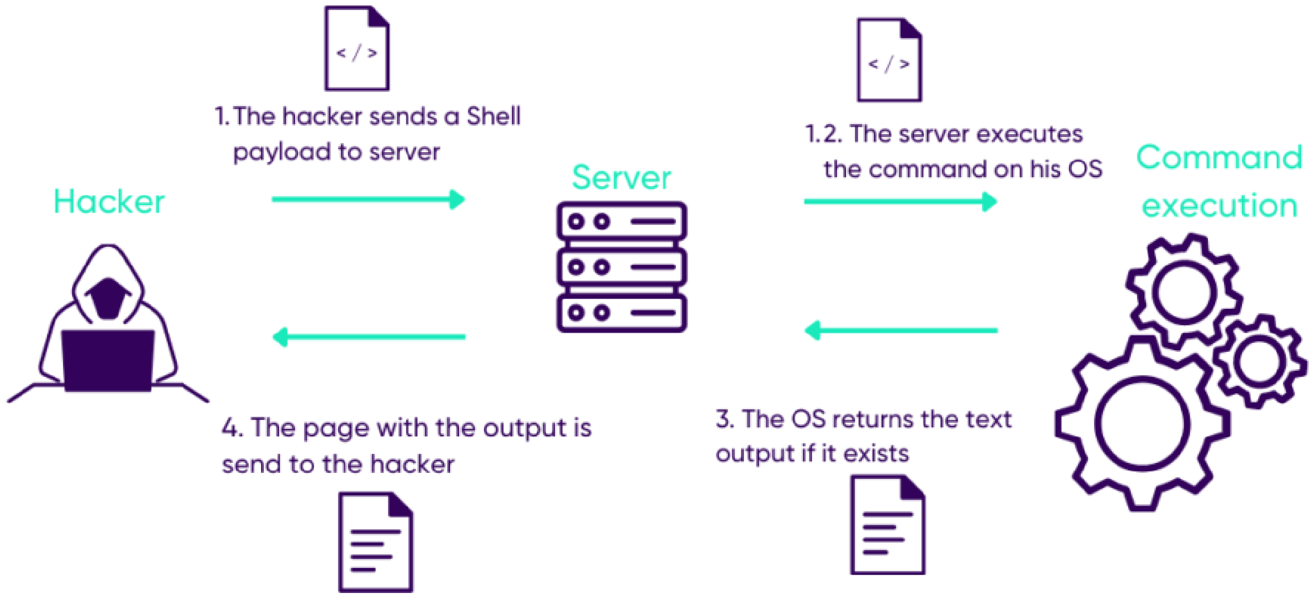
POST

How to effectively protect against script injection attacks?

What is a script injection?

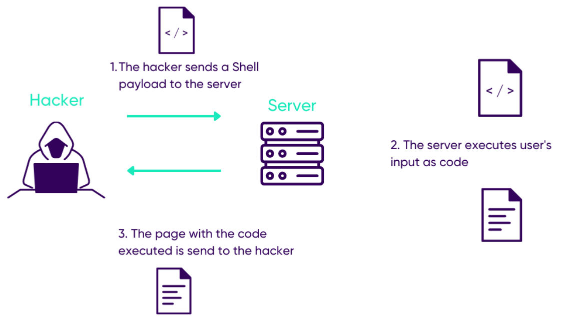
A script language injection or script injection is a flaw mostly present in the web.
A script language injection is when a programming language uses user input to execute code without filtering it.
This vulnerability is similar to OS command injection, but differs in the way it is executed.

Let’s look at a command injection scheme:



1. The hacker sends a Shell payload to server
1.2. The server executes the command on his OS
Hacker
Server
Command execution
4. The page with the output is send to the hacker
3. The OS returns the text output if it exists

In this example, the server will execute a command on the operating system that hosts it, so it is commands specific to the different shells that can be injected.
Let’s look at the script injection:


1. The hacker sends a Shell payload to the server
Hacker
Server
2. The server executes user's input as code
3. The page with the code executed is send to the hacker

Here, the server itself executes a command specific to the language it uses.
An attacker must then use language-specific functions.
However, most languages used on a server can interact with the operating system, so the risks are similar to OS command injection
However, where command injection is only located on the server side, a script injection can also be performed on the client side if the javascript used allows it.

Impact

Script injections have the impacts of 3 vulnerabilities:

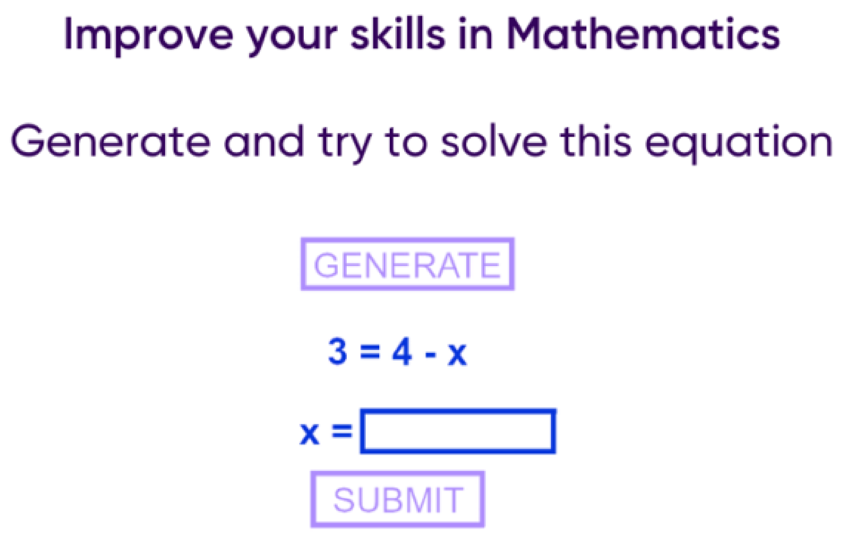
OS Command Injection

Script injections allow for command injections with functions such as system() that execute a command on the OS See article on command injections. SQL injection If the victim server uses a database, then it is possible to perform SQL injections and have the same risks as the latter. (See article on SQL injections). Cross Site Scripting (XSS) When a script injection vulnerability is located on the client side, the application then becomes vulnerable to reflected XSS {reflected XSS}, the consequences can be multiple. See article on script injections.

How can a hacker inject code?

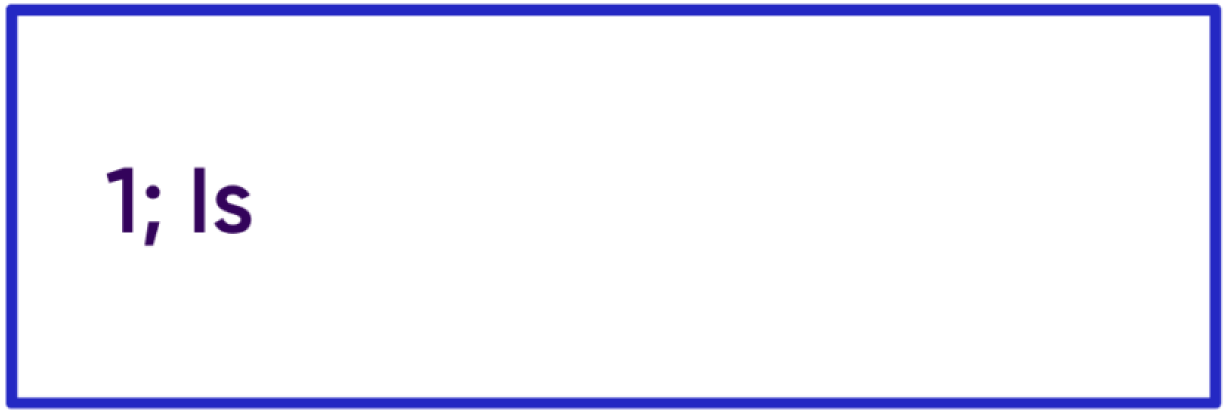
1. A vulnerable client side and server side

To illustrate a script injection attack, I will use the example of a web application containing 2 vulnerabilities: one located on the client side and one located on the server side. It is advisable to have followed the articles on command injections and Cross Site Scripting in order to understand what a command injection is and how a thoughtful XSS works. The vulnerable site allows you to practice equations, a “generate” button displays a random equation and a form to complete it.


Improve your skills in Mathematics
Generate and try to solve this equation
GENERATE
3 = 4 - x
x =
SUBMIT

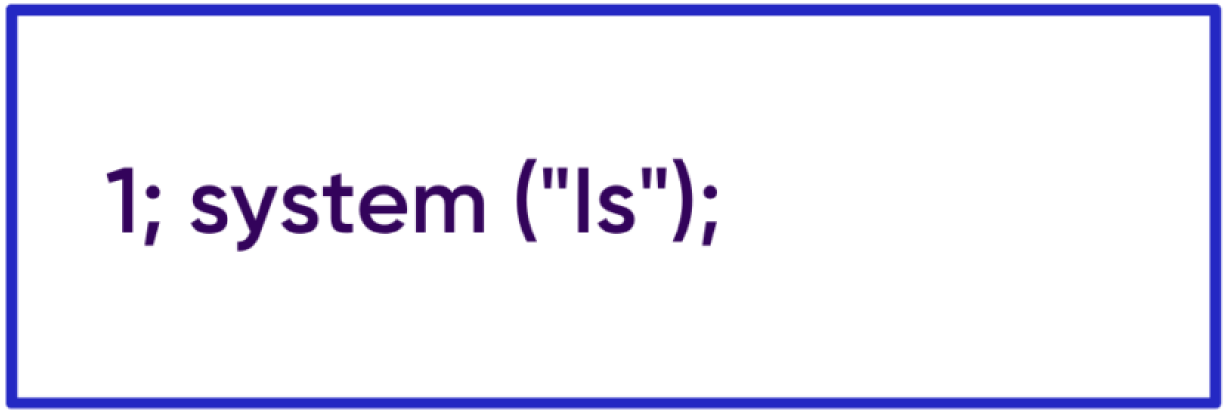
2. Exploiting server-side script injection

On the form, Loris will attempt to enter a Shell command.



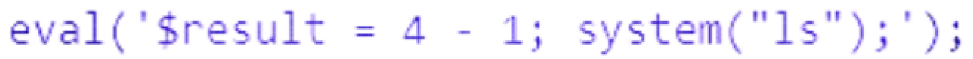
```
1; ls
```

He ends up with a PHP error message.
He then understands that his input is being processed by the eval function, a function for executing PHP code in a PHP program.
In order to be able to execute Shell commands, he will then transform his input into :



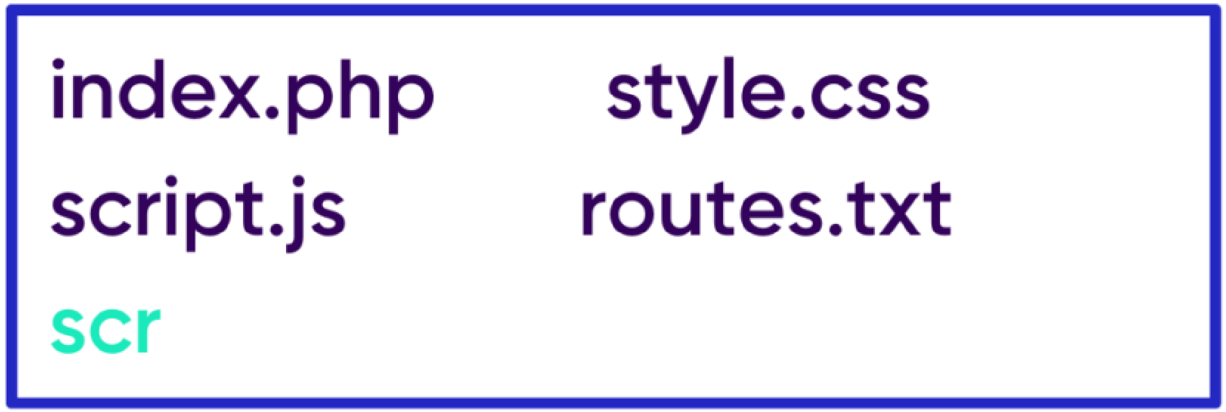
```
1; system ("ls");
```

The system function will allow any Shell command to be executed, so the malicious code executed by the server will look like this:



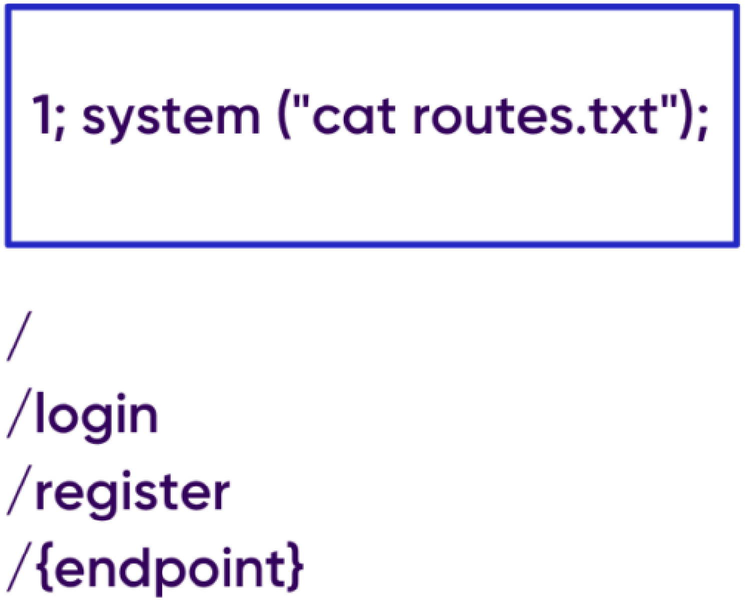
```
eval('$result = 4 - 1; system("ls");');
```

When doing an ls, Loris sees a file called “routes.txt”,



index.php style.css
script.js routes.txt
scr

Using cat on it, Loris ends up with all the routes available on the website, including “/{endpoint_name}” which is not visible on the web page.

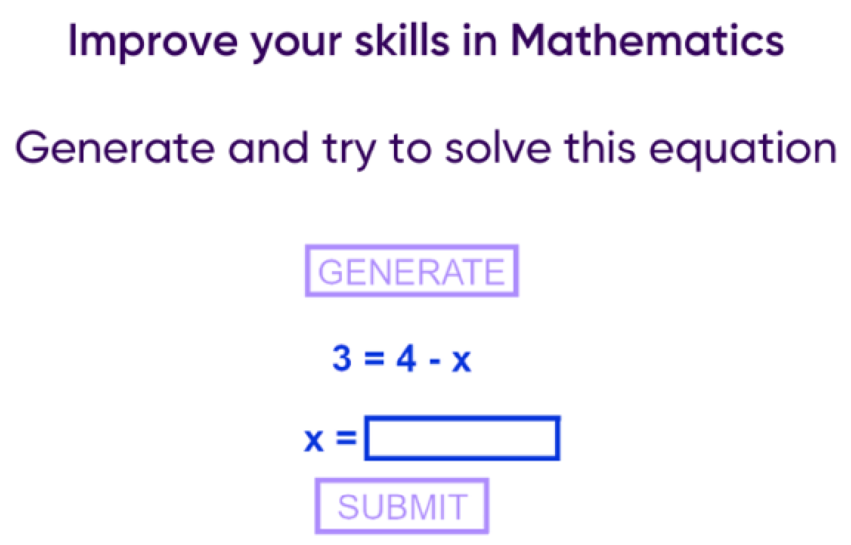


```
1; system ("cat routes.txt");
```

```
/
/login
/register
/{endpoint}
```

3. Exploiting client-side script injection

Going to it, Loris then finds a classic page:



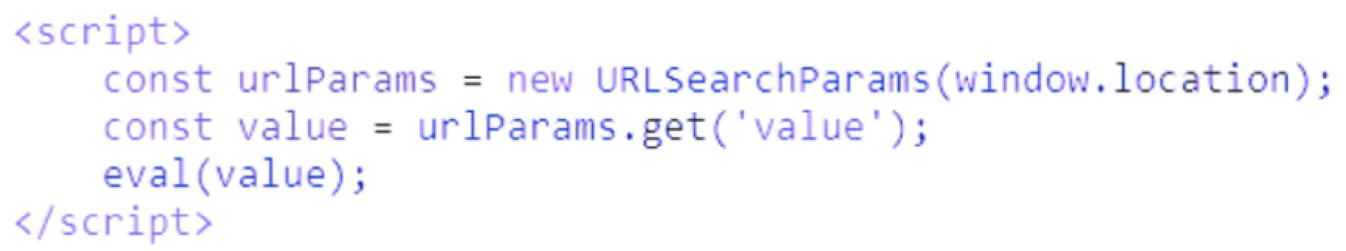
Improve your skills in Mathematics

Generate and try to solve this equation

$3 = 4 - x$

$x =$

Looking in the HTML code, he notices a section <script> calling the eval function, working similarly to the eval function in PHP but with Javascript code. This function seems to use a parameter of the url.



```
<script>
    const urlParams = new URLSearchParams(window.location);
    const value = urlParams.get('value');
    eval(value);
</script>
```

It is therefore possible to make a browser execute javascript code thanks to a parameter in the url
Using malicious code to redirect an administrator's cookies to his server with 'document.cookie', Loris would then have access to an account with elevated privileges.
So all he has to do is build his code for a thoughtful XSS:
Then place it in the 'value' parameter of the url.
Finally, using a bit of social engineering, he just has to send an email to a site administrator to make him click on the link containing the attack to execute the javascript.

What are the protections against script injection attacks?

To protect against script injection attacks the recommendations for other injections apply:

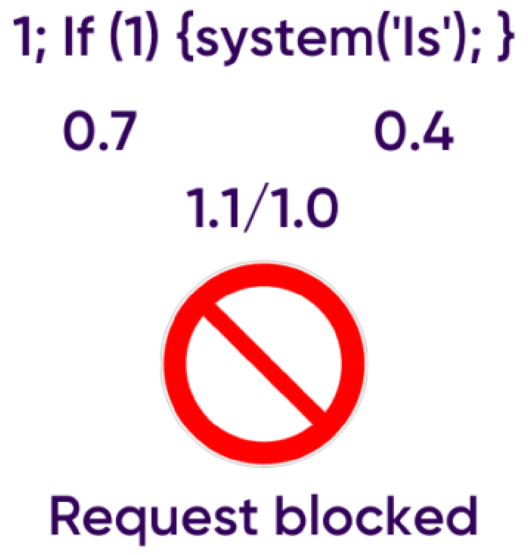
- Filter all user input

- Allow only certain values to restrict a user's scope of action to the maximum

However script injection attacks are very dangerous and involve huge consequences.
Moreover, such functions are usually easily replaceable.
It is therefore mandatory to never use a function to execute code entered for the user.

How to protect yourself from script injection attacks with UBIKA Cloud protector

UBIKA Cloud Protector allows to protect web applications thanks to its syntax engine which allows it to detect the grammar of the different languages used on the server side: PHP, Java etc.



In this example, UBIKA Cloud Protector will detect 2 malicious patterns:

« if () { } » et « system() ; »

, and assign them a score.

If the total score is above the allowed limit, the request will be intercepted and the user will be redirected to a blocking page.

REQUEST A 14-DAY FREE TRIAL

Posted 2 years ago by Camila

Digital Marketing & Channel Manager @ Ubika

in VIEW ALL POSTS BY CAMILA

Related Posts

Posts

Navigating the WAAP/WAF Landscape: Bal...

Posts

UBIKA renews its ANSSI certification

UBIKA solutions are CSPN (Certification de Sécurité de Premier Niveau) certified

Posts

How challenging is it to ensure the se...

READ MORE

b...

READ MORE

READ MORE

UBIKA

UBIKA, the new DenyAll, is a European cybersecurity provider. Its mission is to help organizations secure their digital transformation by protecting applications against cyber attacks.

Produits
- WAAP Gateway / On Pr…
- WAAP Gateway / Clou…
- Cloud Protector
- WAAP Container

Solutions
- Solutions
- By use case
- By vertical

Resources
- Posts
- Webinars
- Events

About
- About Us
- Press & Awards
- Career

Follow us

Subscribe to our newsletter

your email address, you sign up to receive our latest blog
nail and confirm that you have read our privacy policy. You
ibe at any time using the unsubscribe links at the bottom
or by contacting us.

Contact us

9 rue Jeanne Braconnier, 92366 Meudon, France
info@ubikasec.com • +33 (0)1 46 20 96 00

We value your privacy

We use cookies to enhance your browsing experience, serve personalized ads or content, and analyze our traffic. By clicking "Accept All", you consent to our use of cookies.

Customize Reject All Accept All