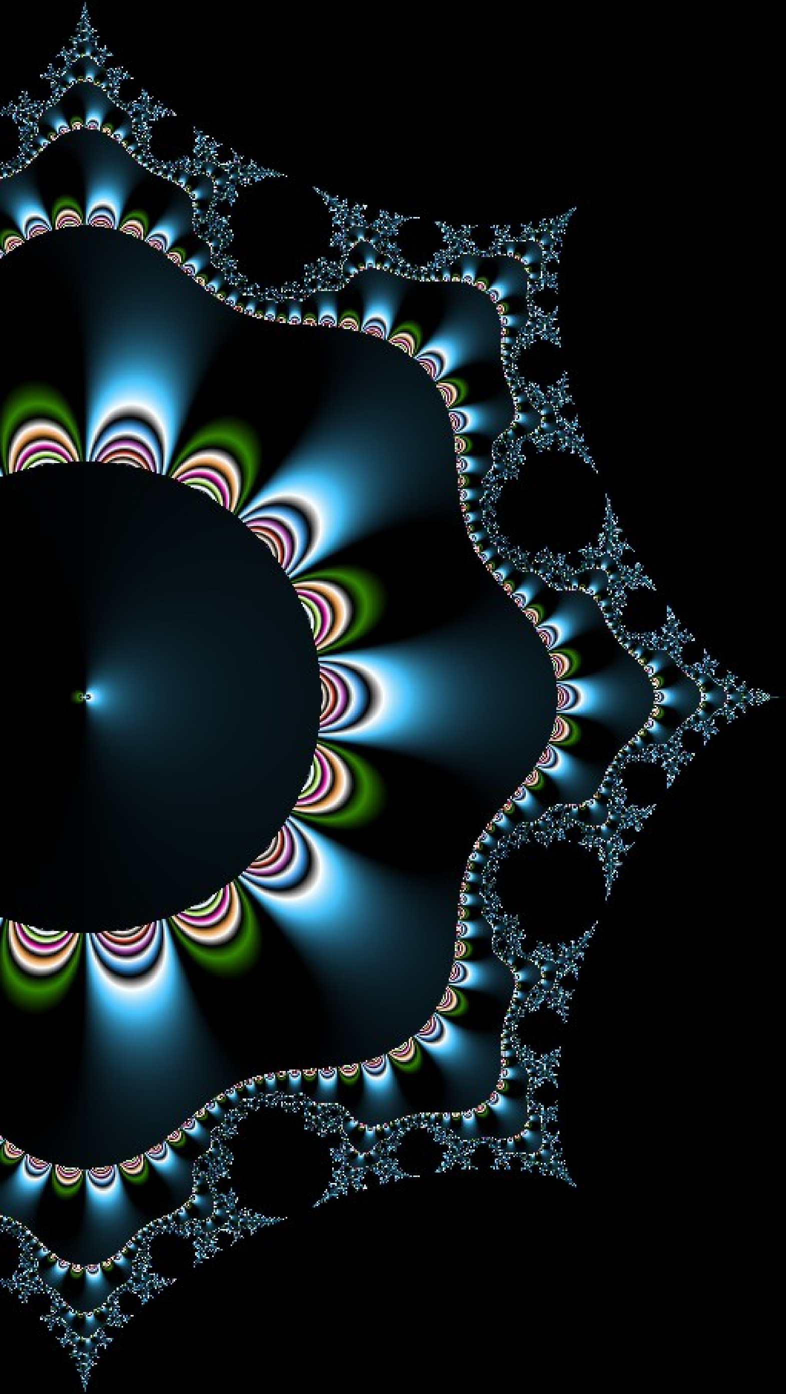


07 – Injections
XSS

Thibaut HENIN

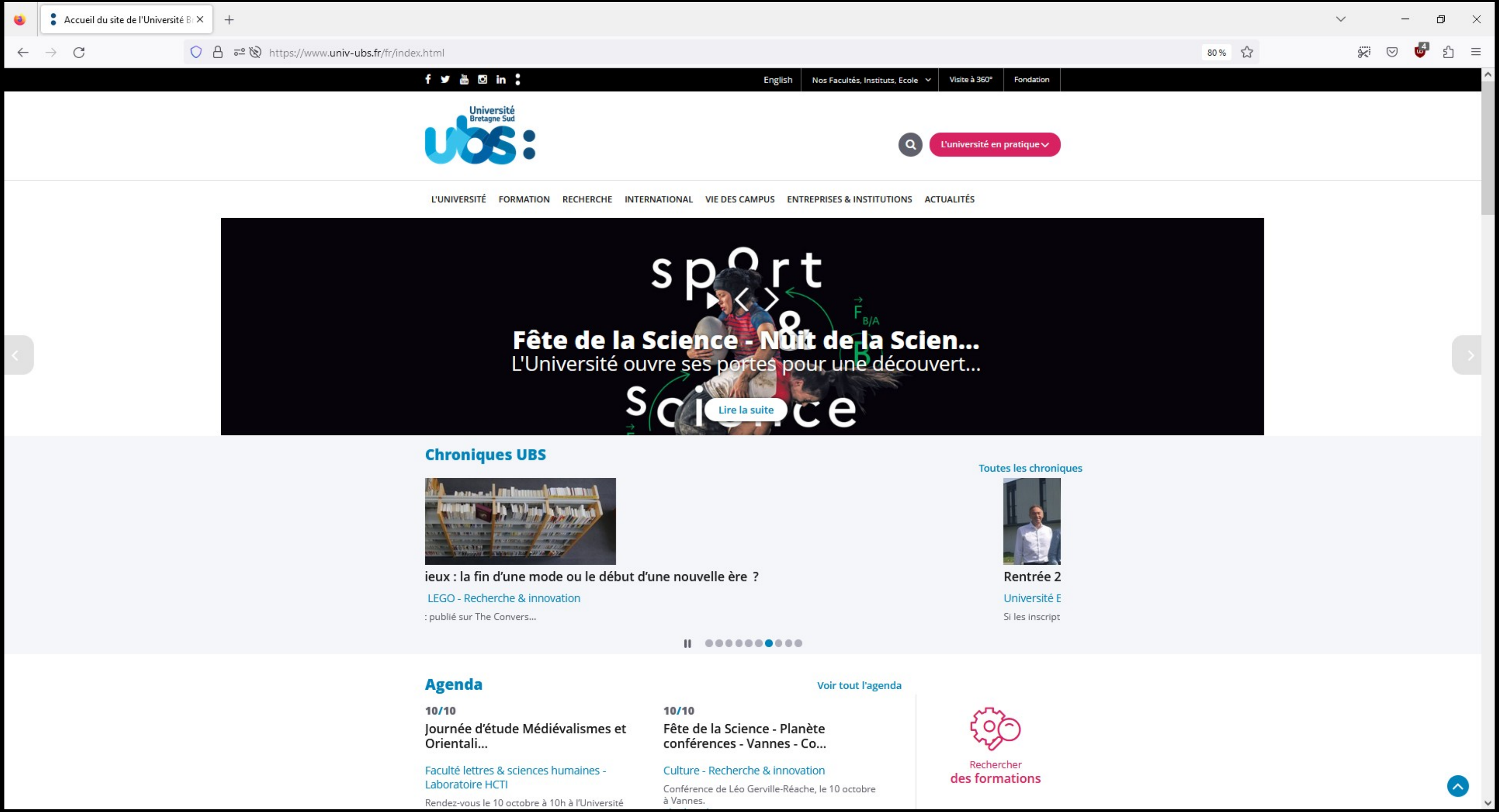
www.arsouyes.org

Web technologies

HTML, Javascript, …

World Wide Web…

Were the magic happens



Fête de la Science - Nuit de la Scien…

L'Université ouvre ses portes pour une découvert…

Chroniques UBS

ieux : la fin d'une mode ou le début d'une nouvelle ère ?

LEGO - Recherche & innovation

: publié sur The Convers…

Rentrée 2

Université E

Si les inscript

Agenda

10/10

Journée d'étude Médiévalismes et Orientali…

Faculté lettres & sciences humaines - Laboratoire HCTI

Rendez-vous le 10 octobre à 10h à l'Université

10/10

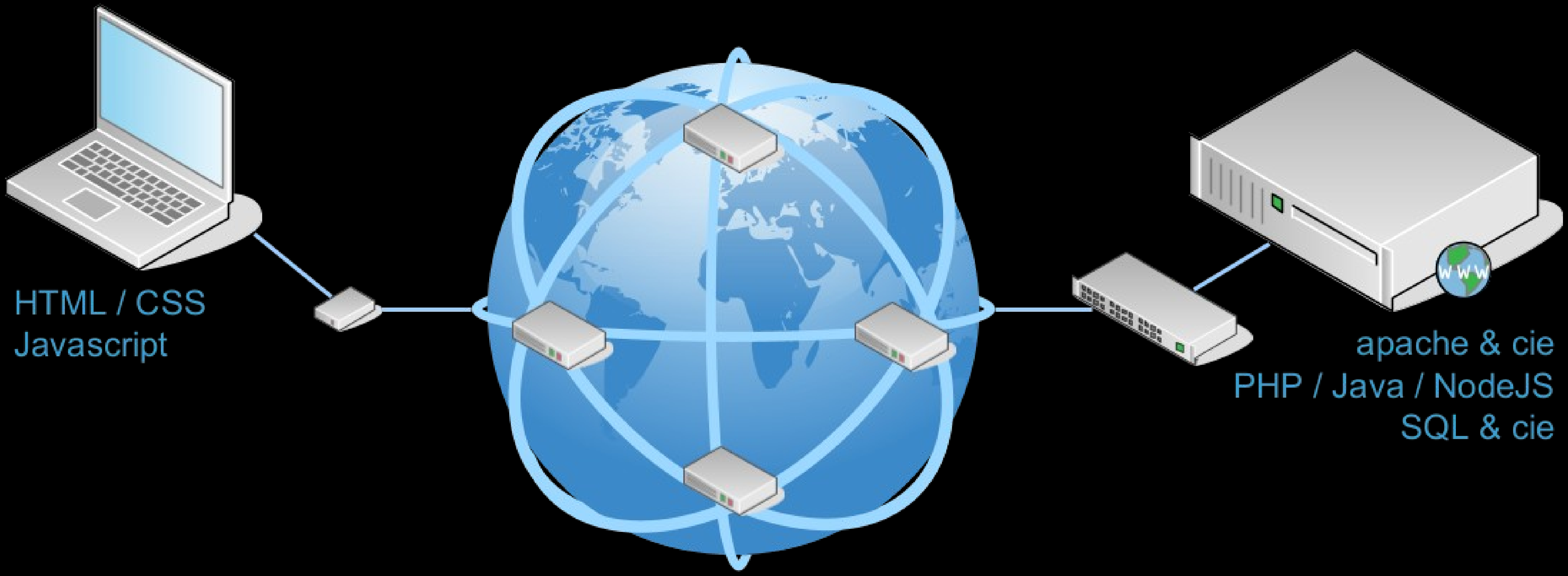
Fête de la Science - Planète conférences - Vannes - Co…

Culture - Recherche & innovation

Conférence de Léo Gerville-Réache, le 10 octobre à Vannes.

Network protocols

How the magic happens



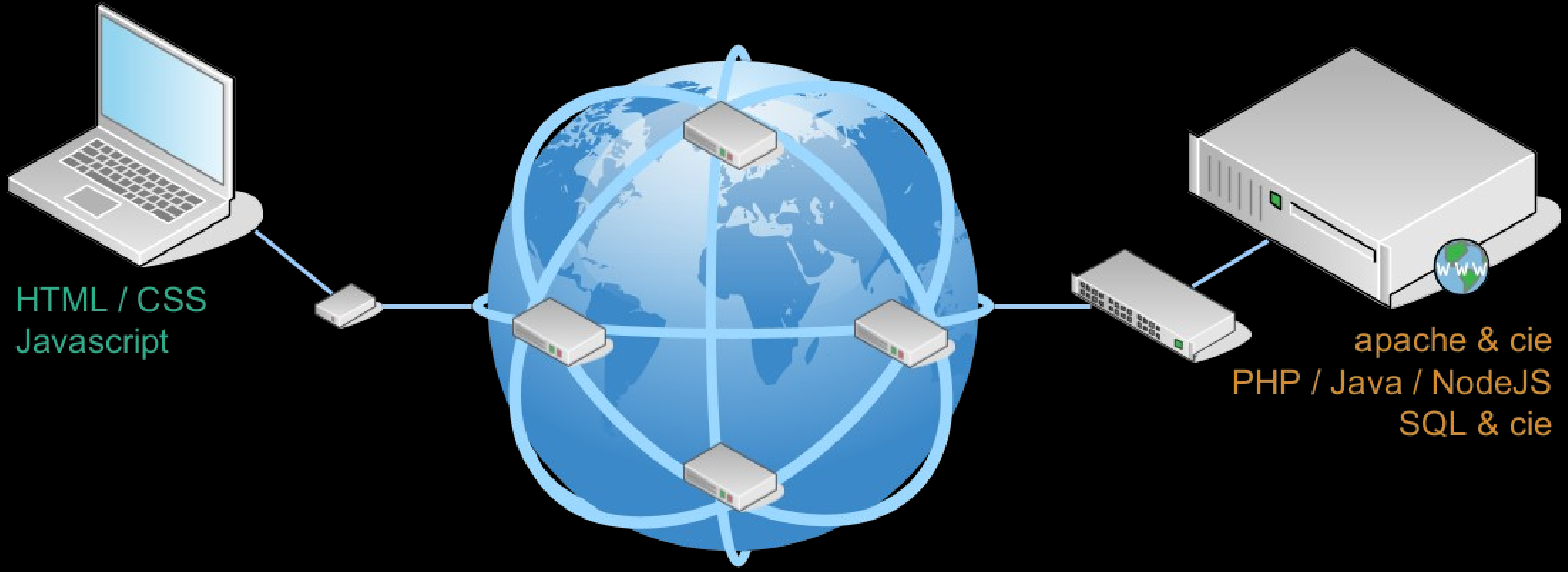
HTML / CSS
Javascript

IP / TCP UDP / DNS
TLS / HTTP

apache & cie
PHP / Java / NodeJS
SQL & cie

Network protocols

How the magic happens



HTML / CSS
Javascript

IP / TCP UDP / DNS
TLS / HTTP

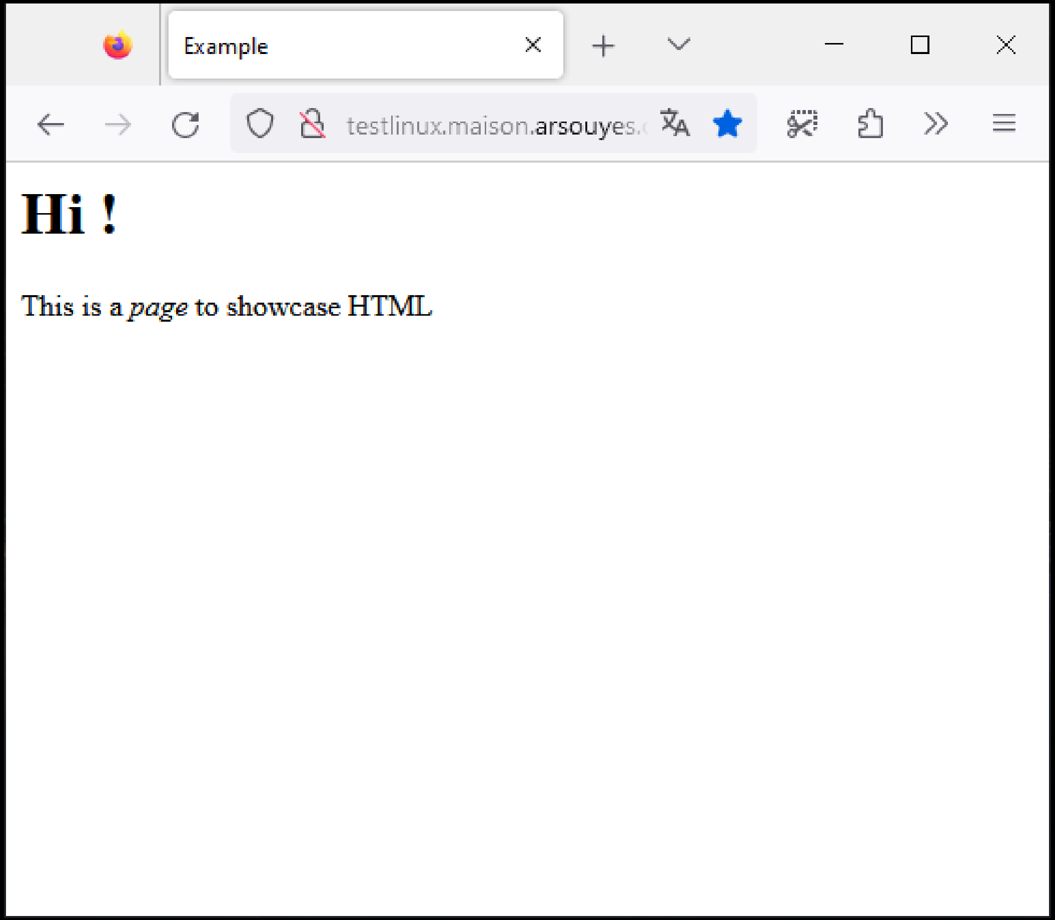
apache & cie
PHP / Java / NodeJS
SQL & cie

HTML / CSS / Javascript

Fastest introduction ever

HTML Basis

```
<html lang="en">
  <head>
    <title>Example</title>
  </head>

  <body>

    <h1>Hi !</h1>

    <p>This is a
      <em>page</em>
      to showcase HTML</p>

  </body>
</html>
```

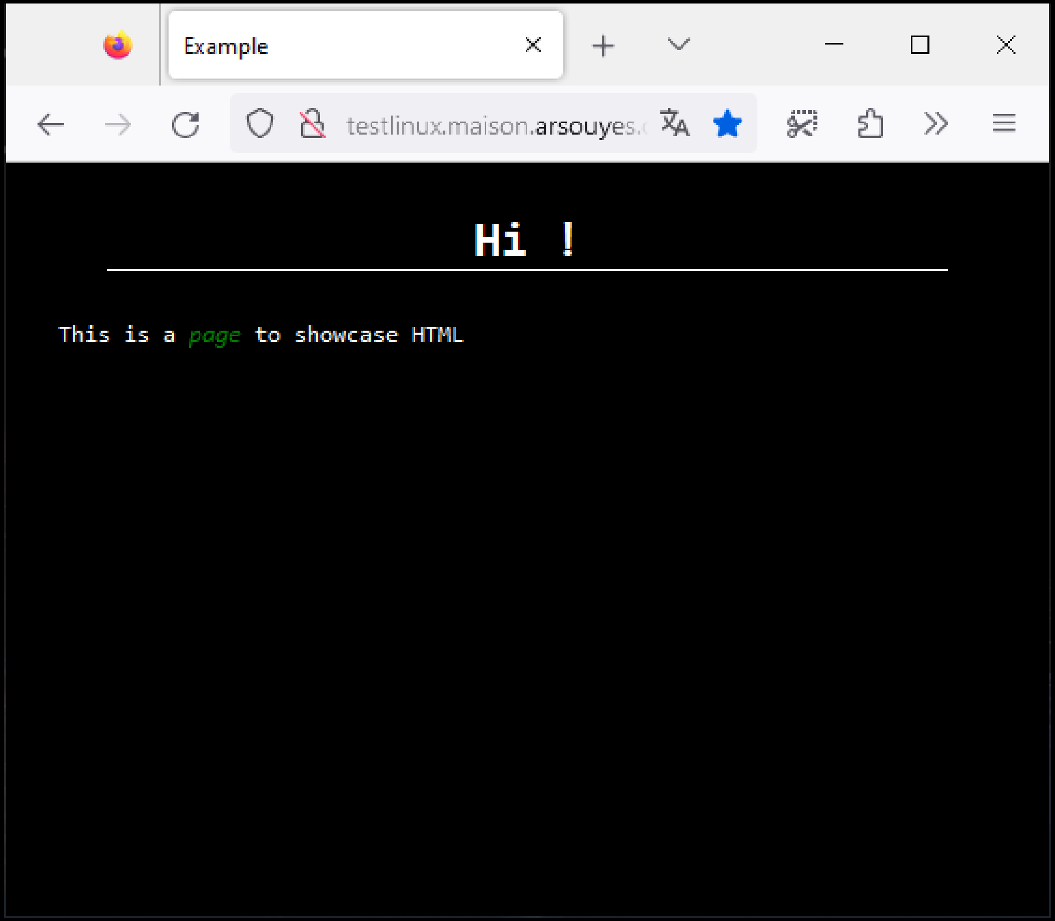


Hi !

This is a *page* to showcase HTML

CSS Basis

```
body {
        background-color: black ;
        color: white ;
        font-family: monospace ;
        margin: 0 auto 0 auto ;
        width: 90% ;
}
h1 {
        margin: 1em ;
        text-align: center ;
        border-bottom: solid 1px ;
}
em {
        color: green ;
}
```

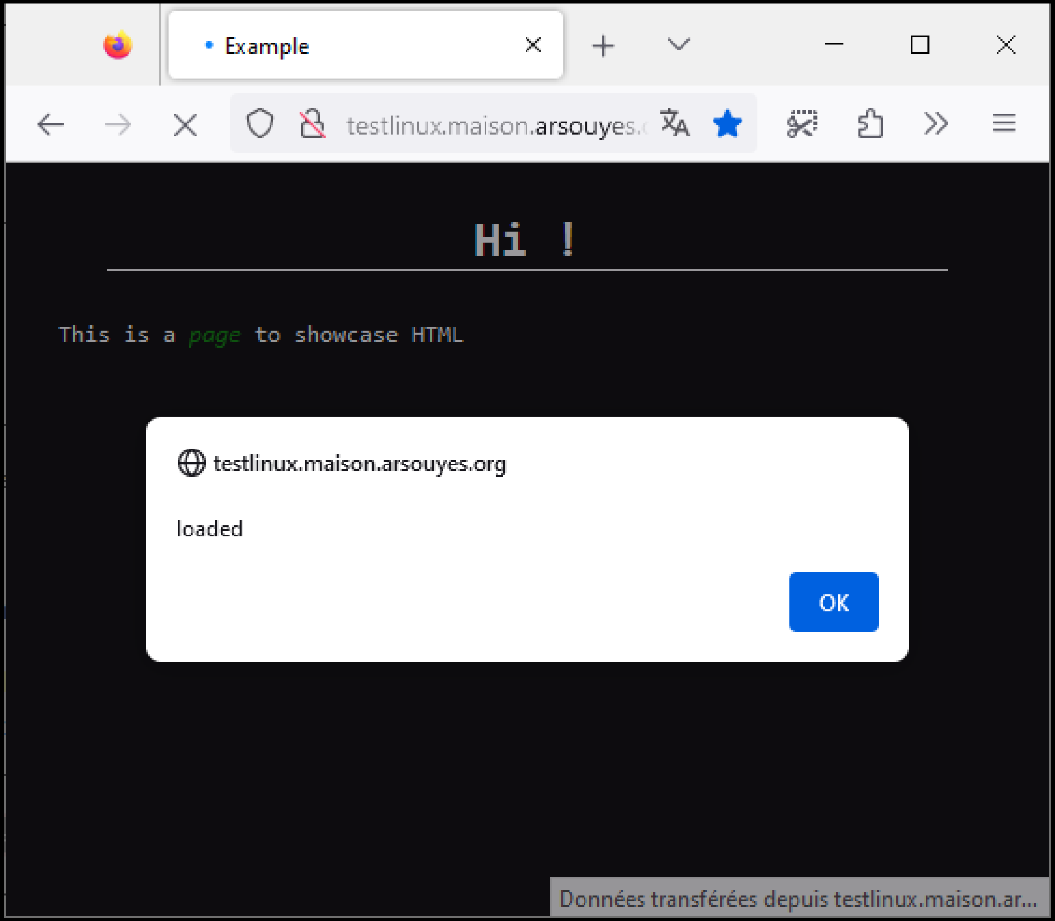


Hi !

This is a *page* to showcase HTML

JS Basis

```
window.onload =
    function() {
        alert("loaded") ;
    } ;
```



Example
testlinux.maison.arsouyes.

Hi !

This is a *page* to showcase HTML

testlinux.maison.arsouyes.org

loaded

OK
Données transférées depuis testlinux.maison.ar...

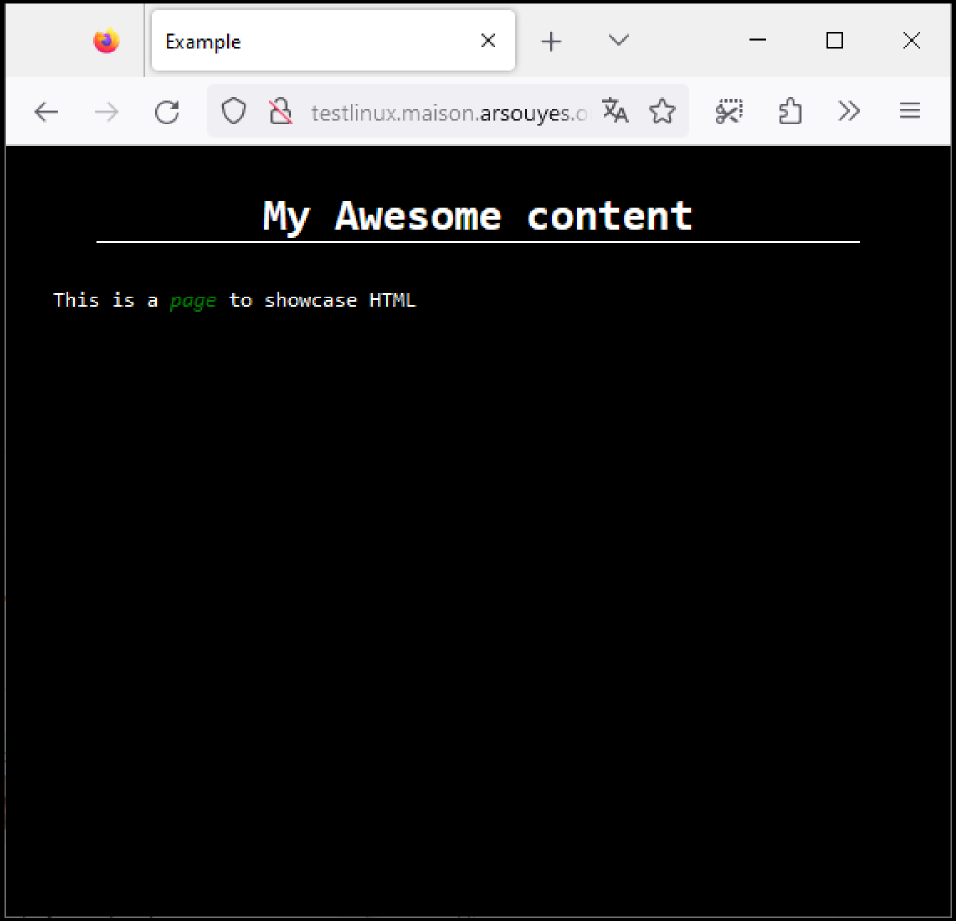
Ajax Basis

Main.js

```
window.onload = function() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4
     && this.status == 200) {
      document.getElementsByTagName("h1")
[0].innerHTML
          = this.responseText;
    }
  };
  xhttp.open("GET", "Test.txt", true);
  xhttp.send();
}
```

Test.txt

```
My Awesome content
```



My Awesome content

This is a *page* to showcase HTML

XSS Reflected

Cross Site Scripting

Example of Vulnerable Application

```
<html lang="en">
  <head>
    <title>Example</title>
    <link href="style.css" rel="stylesheet" />
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>Hi !</h1>
    <p>Hello <em>
             <?php echo $_GET["user"] ?? "you" ; ?>
             </em>!</p>
  </body>
</html>
```

Example of Vulnerable Application

```
<html lang="en">
  <head>
    <title>Example</title>
       <link href="style.css" rel="stylesheet" />
       <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>Hi !</h1>
    <p>Hello <em>
             <?php echo $_GET["user"] ?? "you" ; ?>
             </em>!</p>
  </body>
</html>
```

http://example.com/

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?>
    </em>!</p>
```

http://example.com/

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo null ?? "you" ; ?>
    </em>!</p>
```

http://example.com/

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo null ?? "you" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "you" ; ?>
    </em>!</p>
```

http://example.com/

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo null ?? "you" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "you" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    You
</em>!</p>
```

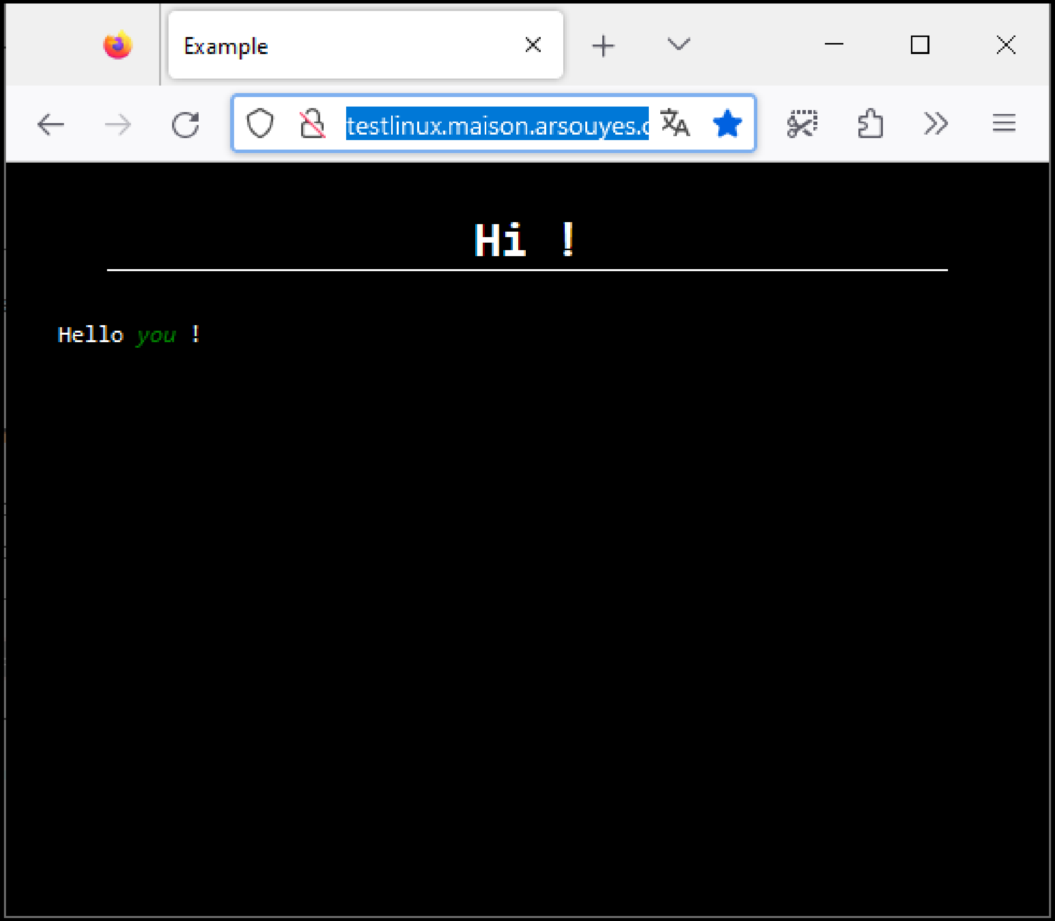
http://example.com/

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo null ?? "you" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "you" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    You
</em>!</p>
```



Hi !

Hello *you* !

http://example.com/?user=tbowan

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

http://example.com/?user=tbowan

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "tbowan" ?? "you" ; ?>
    </em>!</p>
```

http://example.com/?user=tbowan

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "tbowan" ?? "you" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "tbowan" ; ?>
    </em>!</p>
```

http://example.com/?user=tbowan

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "tbowan" ?? "you" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "tbowan" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    tbowan
    </em>!</p>
```

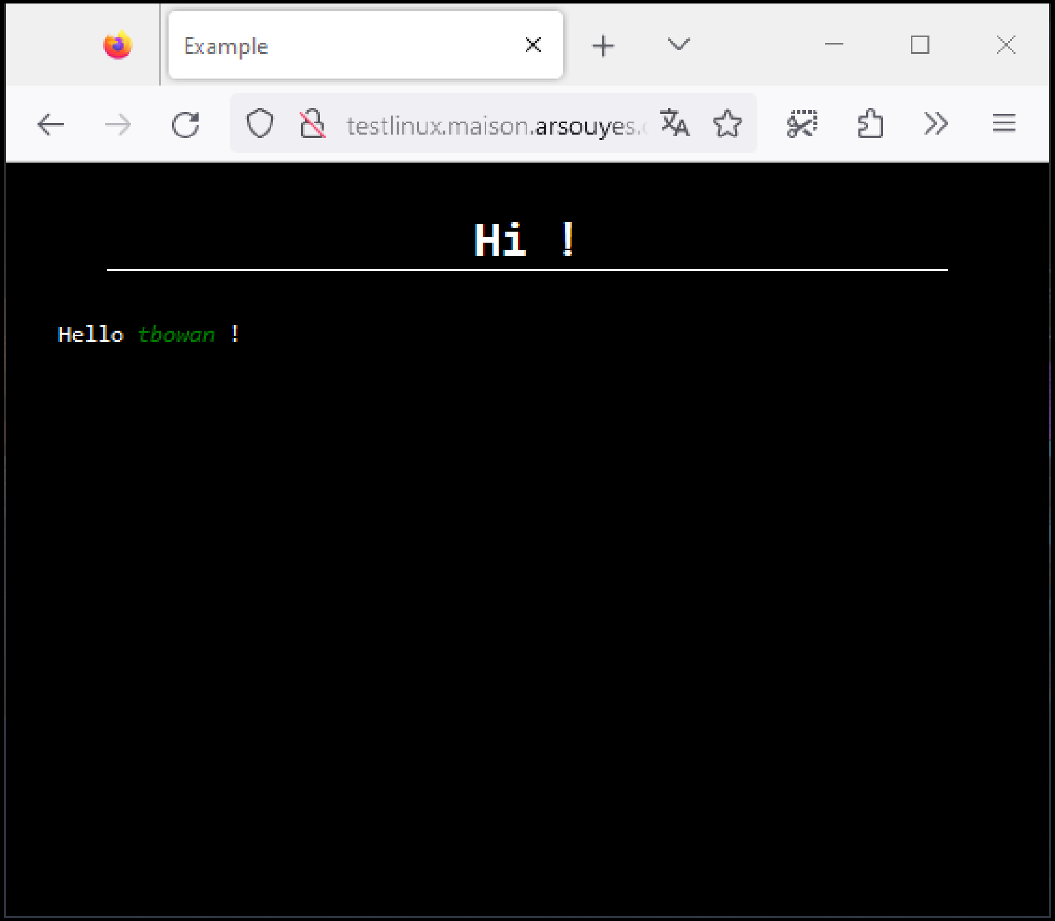
http://example.com/?user=tbowan

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "tbowan" ?? "you" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "tbowan" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    tbowan
    </em>!</p>
```



Hi !

Hello *tbowan* !

http://example.com/?user=<h1>tbowan</h1>

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?>
    </em>!</p>
```

http://example.com/?user=<h1>tbowan</h1>

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "<h1>tbowan</h1>" ??
"you";?>
    </em>!</p>
```

http://example.com/?user=<h1>tbowan</h1>

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "<h1>tbowan</h1>" ??
"you";?>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "<h1>tbowan</h1>" ; ?>
    </em>!</p>
```

http://example.com/?user=<h1>tbowan</h1>

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "<h1>tbowan</h1>" ?? "you";?>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "<h1>tbowan</h1>" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    <h1>tbowan</h1>
    </em>!</p>
```

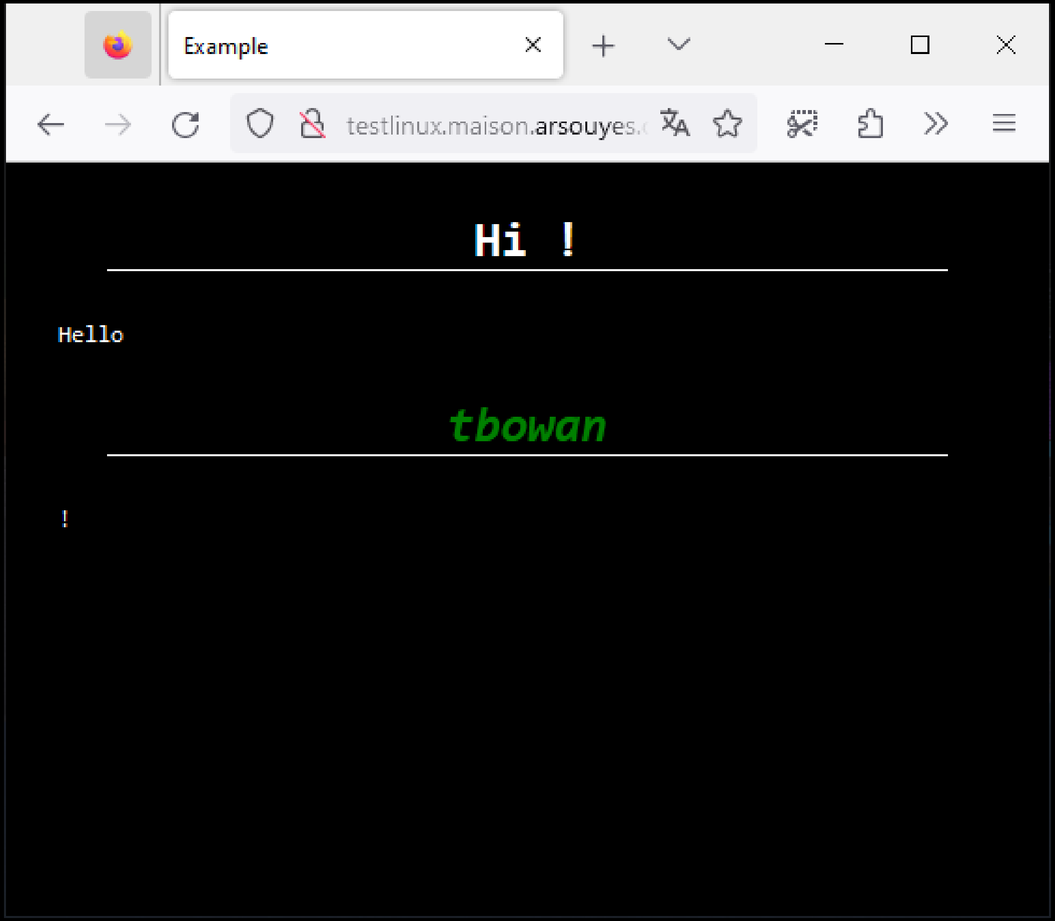
http://example.com/?user=<h1>tbowan</h1>

```
<p>Hello <em>
    <?php echo $_GET["user"] ?? "you" ; ?
>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "<h1>tbowan</h1>" ??
"you";?>
    </em>!</p>
```

```
<p>Hello <em>
    <?php echo "<h1>tbowan</h1>" ; ?>
    </em>!</p>
```

```
<p>Hello <em>
    <h1>tbowan</h1>
    </em>!</p>
```



Hi !

Hello

tbowan

!

Better injection :

```
http://example.com/?user=
    tbowan</em>!</p>
    <h1>Congratulation</h1>
    <p>Click
        <a href="evil.com">HERE</a>
        to win your price
    <em>
```

Better injection :

```
http://example.com/?user=
    tbowan</em>!</p>
    <h1>Congratulation</h1>
    <p>Click
        <a href="evil.com">HERE</a>
        to win your price
    <em>
```

```
<p>Hello <em>
    <?php echo "tbowan</em>…
pricece<em>" ;?>
    </em>!</p>
```

Better injection :

```
http://example.com/?user=
    tbowan</em>!</p>
    <h1>Congratulation</h1>
    <p>Click
        <a href="evil.com">HERE</a>
        to win your price
    <em>
```

```
<p>Hello <em>
    <?php echo "tbowan</em>…
pricece<em>" ;?>
    </em>!</p>
```

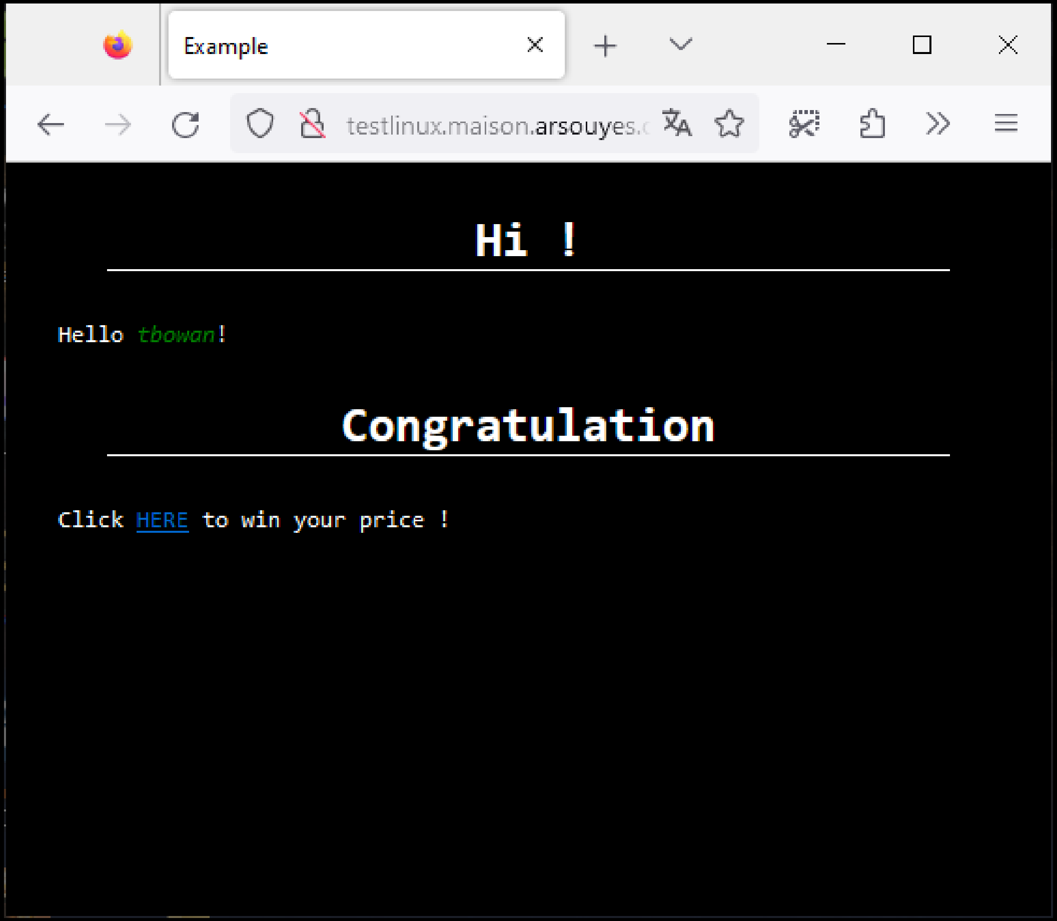
```
<p>Hello <em>
    tbowan</em>!</p>
    <h1>Congratulation</h1>
    <p>Click
        <a href="evil.com">HERE</a>
        to win your price
    <em>
    </em>!</p>
```

Better injection :

```
http://example.com/?user=
    tbowan</em>!</p>
    <h1>Congratulation</h1>
    <p>Click
        <a href="evil.com">HERE</a>
        to win your price
    <em>
```

```
<p>Hello <em>
    <?php echo "tbowan</em>…
pricece<em>" ;?>
    </em>!</p>
```

```
<p>Hello <em>
    tbowan</em>!</p>
    <h1>Congratulation</h1>
    <p>Click
        <a href="evil.com">HERE</a>
        to win your price
    <em>
    </em>!</p>
```



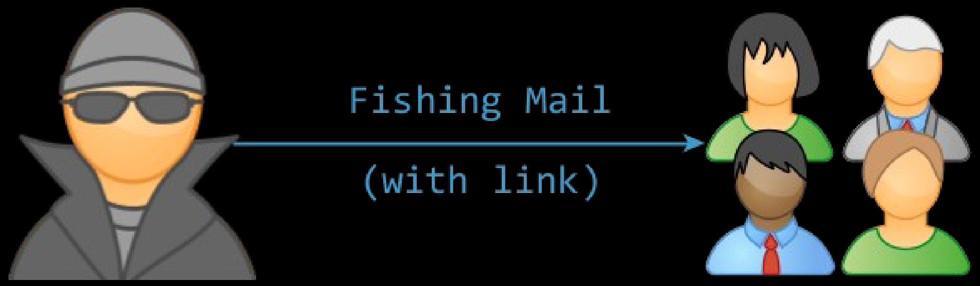
Hi !

Hello *tbowan*!

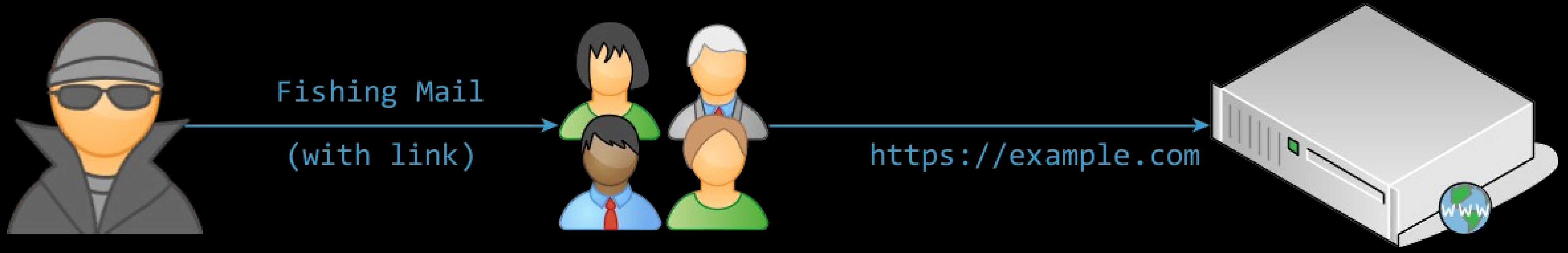
Congratulation

Click HERE to win your price !

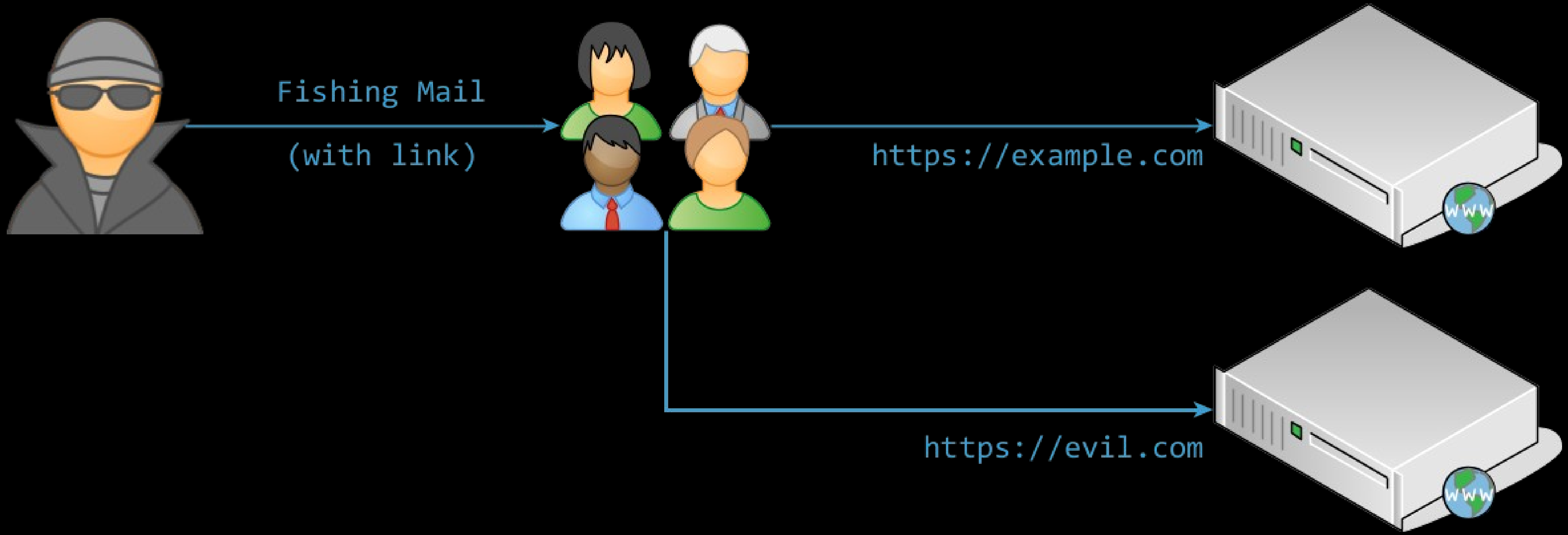
Principle



Principle



Principle



JS Injection

a.k.a. XSS – Cross Site Scripting

```
http://example.com/?user=
    <script>
        alert("Virus Detected");
    </script>
```

JS Injection

a.k.a. XSS – Cross Site Scripting

```
http://example.com/?user=
    <script>
        alert("Virus Detected");
    </script>
```

```
<p>Hello <em>
    <?php echo "<script>…</script>" ;?>
    </em>!</p>
```

JS Injection

a.k.a. XSS – Cross Site Scripting

```
http://example.com/?user=
    <script>
        alert("Virus Detected");
    </script>
```

```
<p>Hello <em>
    <?php echo "<script>…</script>" ;?>
    </em>!</p>
```

```
<p>Hello <em>
    <script>
        alert("Virus Detected");
    </script>
</em>!</p>
```

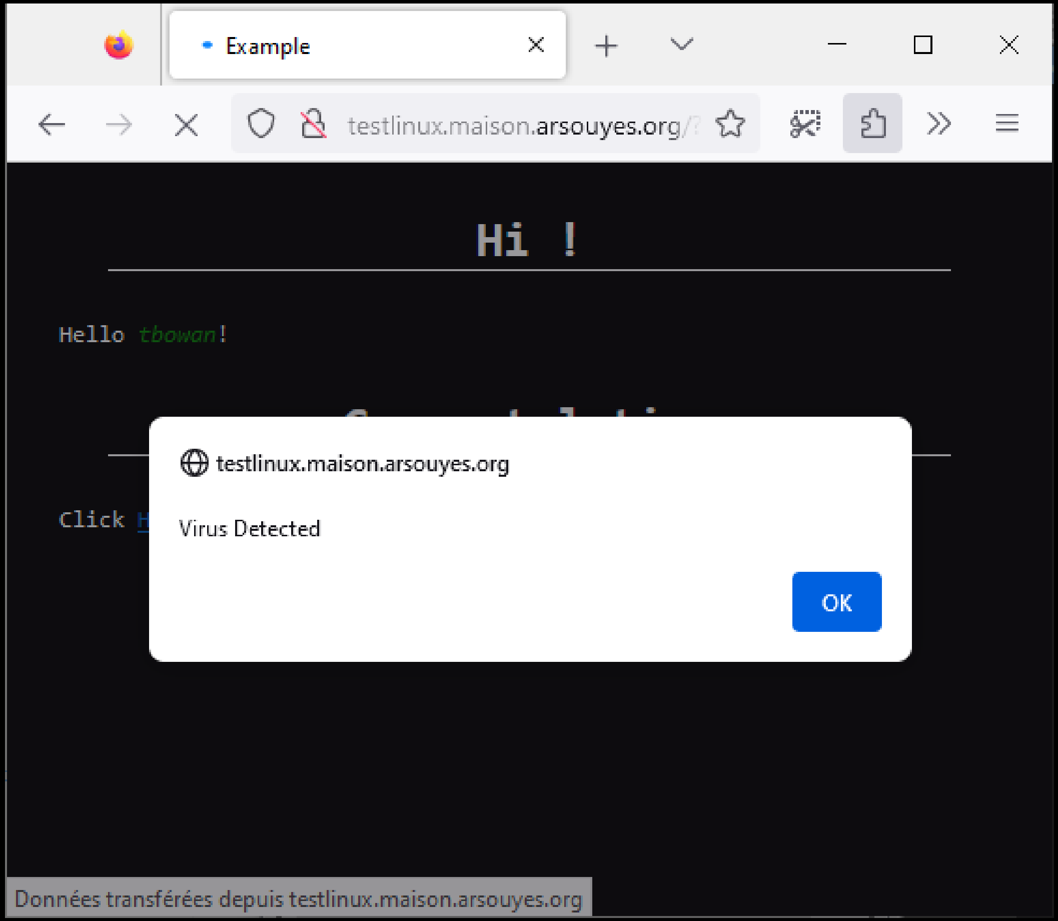
JS Injection

a.k.a. XSS – Cross Site Scripting

```
http://example.com/?user=
    <script>
        alert("Virus Detected");
    </script>
```

```
<p>Hello <em>
    <?php echo "<script>…</script>" ;?>
    </em>!</p>
```

```
<p>Hello <em>
    <script>
        alert("Virus Detected");
    </script>
</em>!</p>
```



testlinux.maison.arsouyes.org

Virus Detected

Données transférées depuis testlinux.maison.arsouyes.org

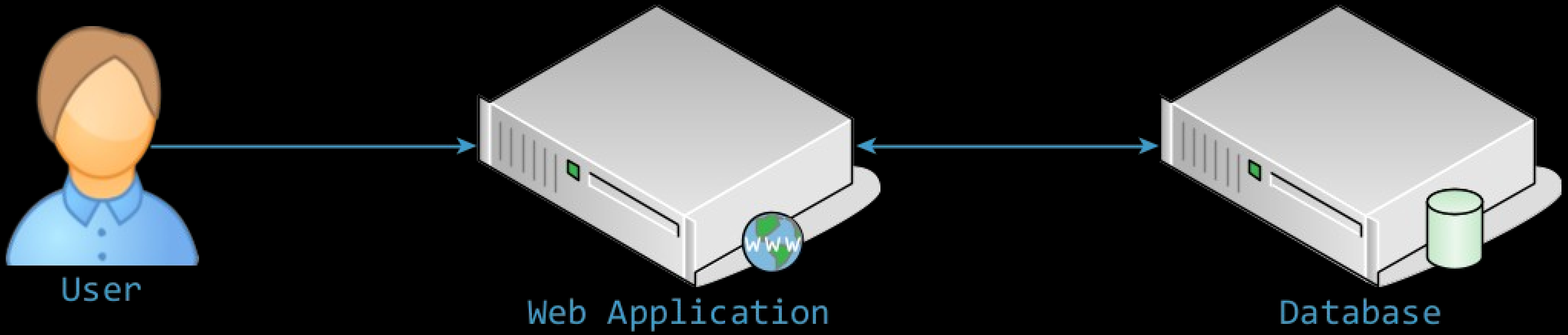
Problem :

Need to send a link to every victim

XSS - Stored

Cross Site Scripting

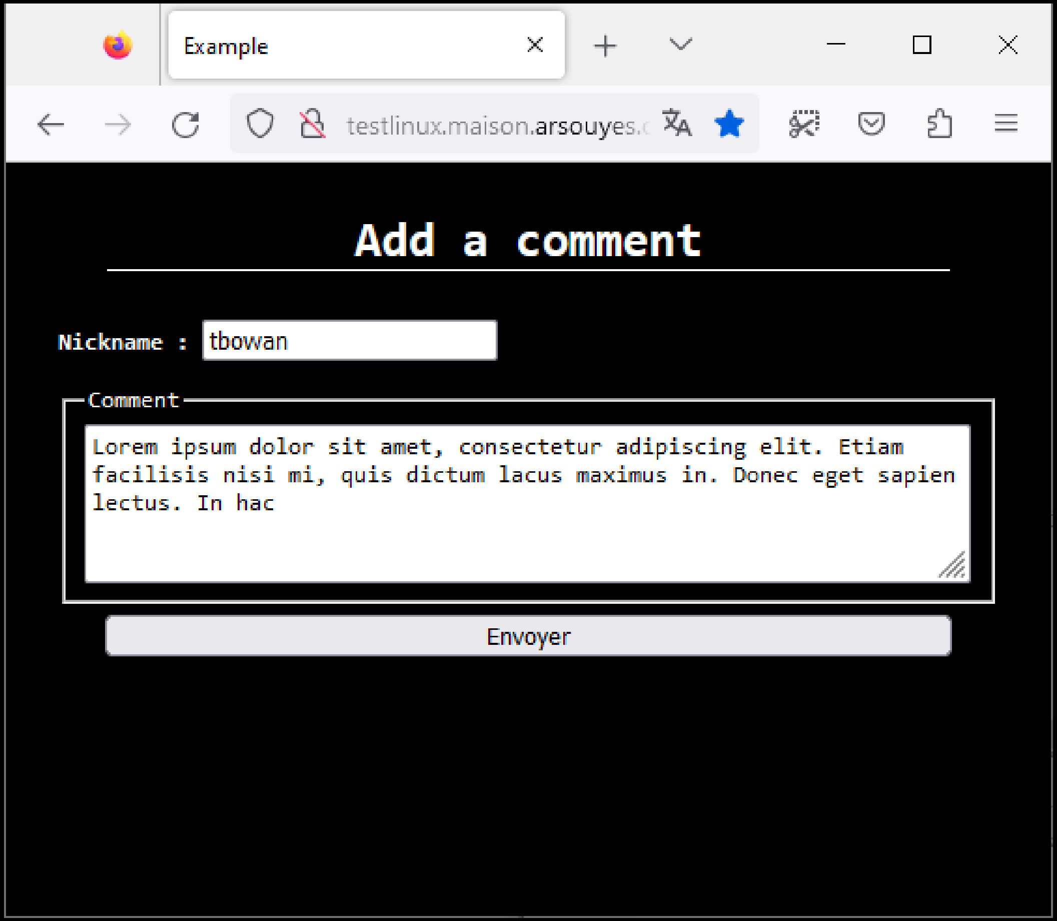
Persistant applications



Example of data creation

i.e. adding a blog comment

```
<h1>Add a comment</h1>
<form method="post"
      action="addComment.php">

<input type="hidden“
       name="article“
       value="123" />

<p><strong>Nickname :</strong>
   <input type="text" name="author" />
   </p>

<fieldset><legend>Comment</legend>
   <textarea name="content"></textarea>
   </fieldset>

<input type="submit" />
</form>
```



Add a comment

Nickname : tbowan

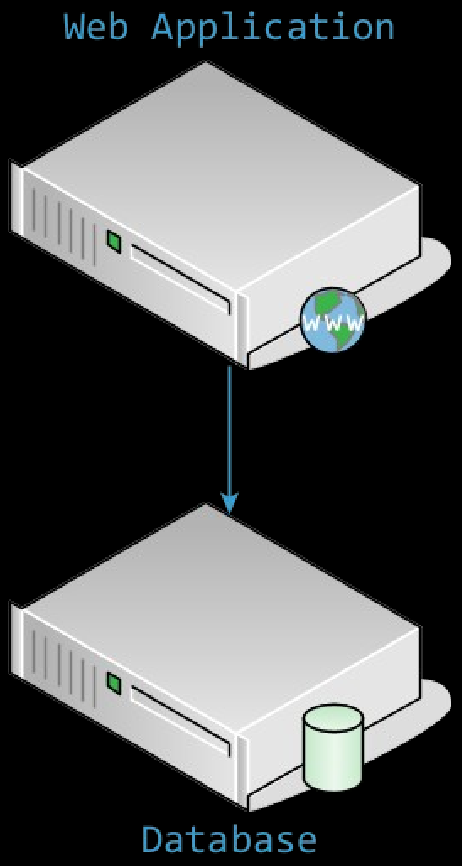
Comment

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam facilisis nisi mi, quis dictum lacus maximus in. Donec eget sapien lectus. In hac

Example of data creation

i.e. adding a blog comment

```
<?php // Add Comment

$cmd = $pdo->prepare(""
       . "insert into comment"
       . " (article, author, content)"
       . " values"
       . "
(:article, :author, :content)"
       ) ;

$cmd->exec([
       "article" => $_POST["article"],
       "author"  => $_POST["author"],
       "content" => $_POST["content"]
       ]) ;
```



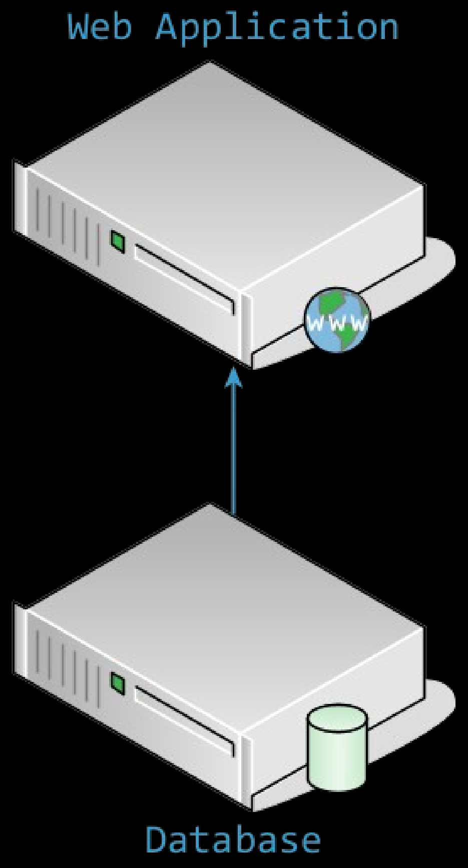
Example of database

Article	Author	content
…	…	…
123	Tbowan	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam facilisis nisi mi, quis dictum lacus maximus in. Donec eget sapien lectus. In hac
…	…	…

Example of data retrieval

i.e. displaying a blog comment

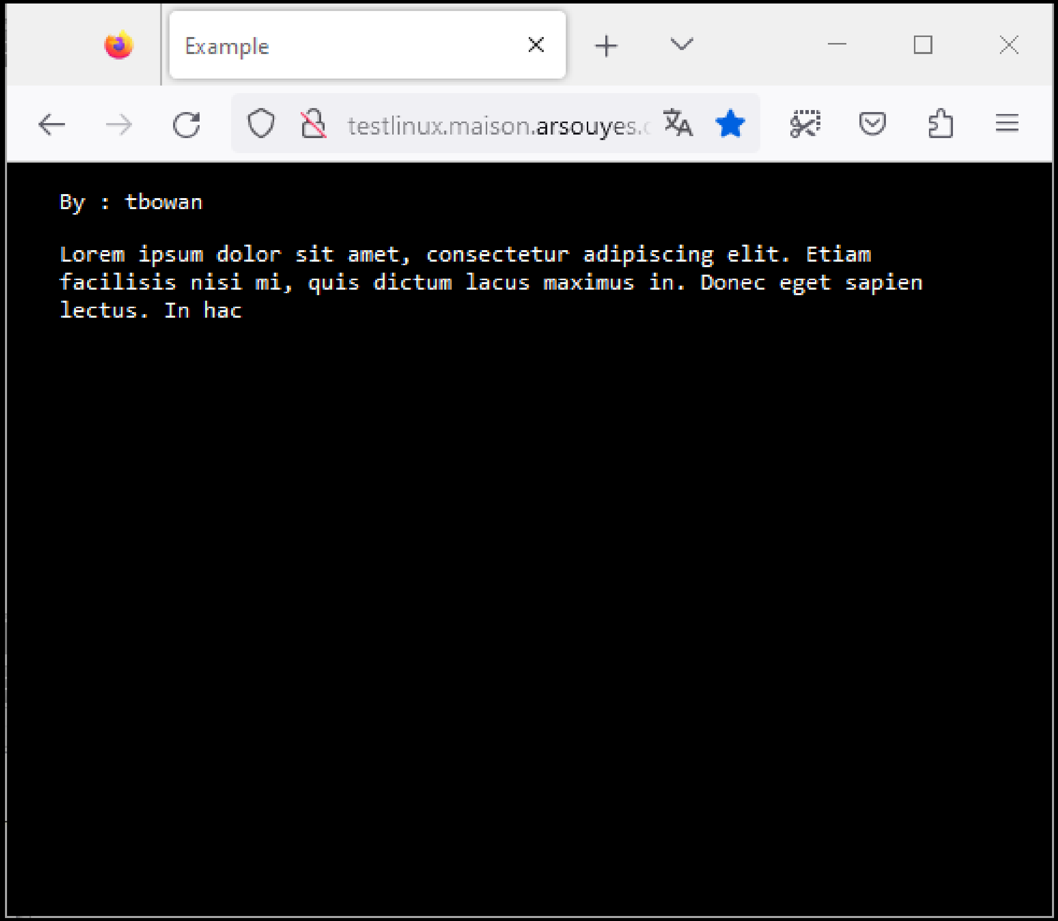
```php
<?php // show post

$cmd = $pdo->prepare(""
       . "select * from comment"
       . " where article = :article"
       ) ;

$st = $cmd->exec(["article" => $_GET["id"] ]) ;

foreach ($st as $row) {
       echo '<div class="comment">' ;
       echo '<p>By : ' . $row["author"] . '</p>'
;
       echo $row["content"] ;
       echo '</div>' ;
}
```



Example of data retrieval

i.e. displaying a blog comment

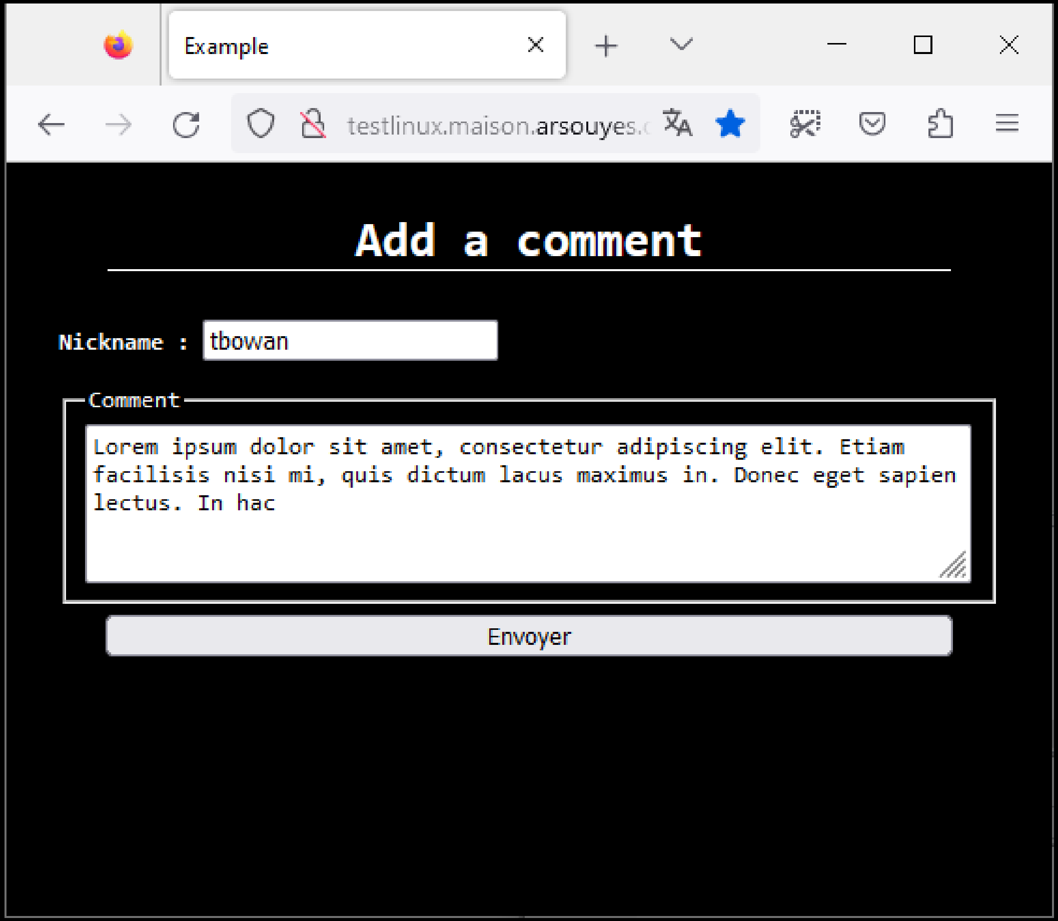
```
       <div class="comment">
       <p>By : tbowan</p>
       Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Etiam
facilisis nisi mi, quis dictum lacus
maximus in. Donec eget sapien
lectus.
In hac
       </div>
```



```
By : tbowan

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam
facilisis nisi mi, quis dictum lacus maximus in. Donec eget sapien
lectus. In hac
```

Example of interraction

i.e. adding a blog comment

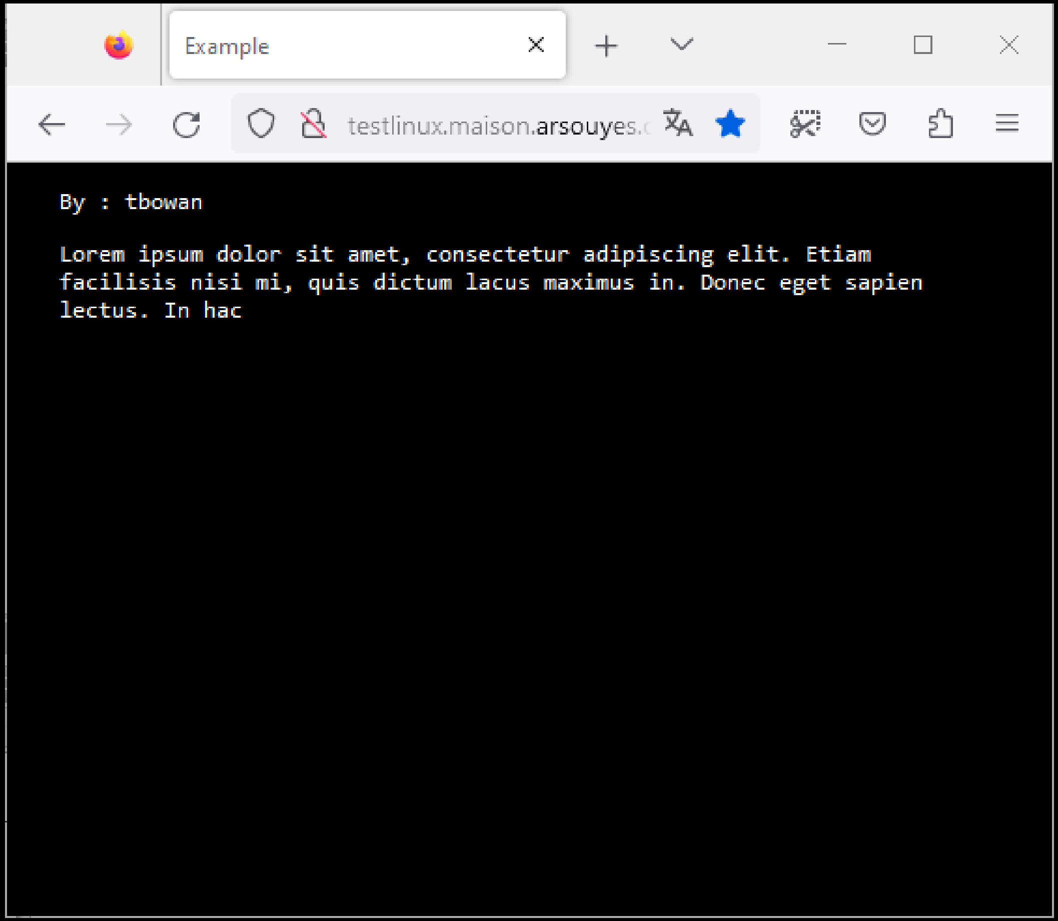


Add a comment

Nickname : tbowan

Comment

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam facilisis nisi mi, quis dictum lacus maximus in. Donec eget sapien lectus. In hac

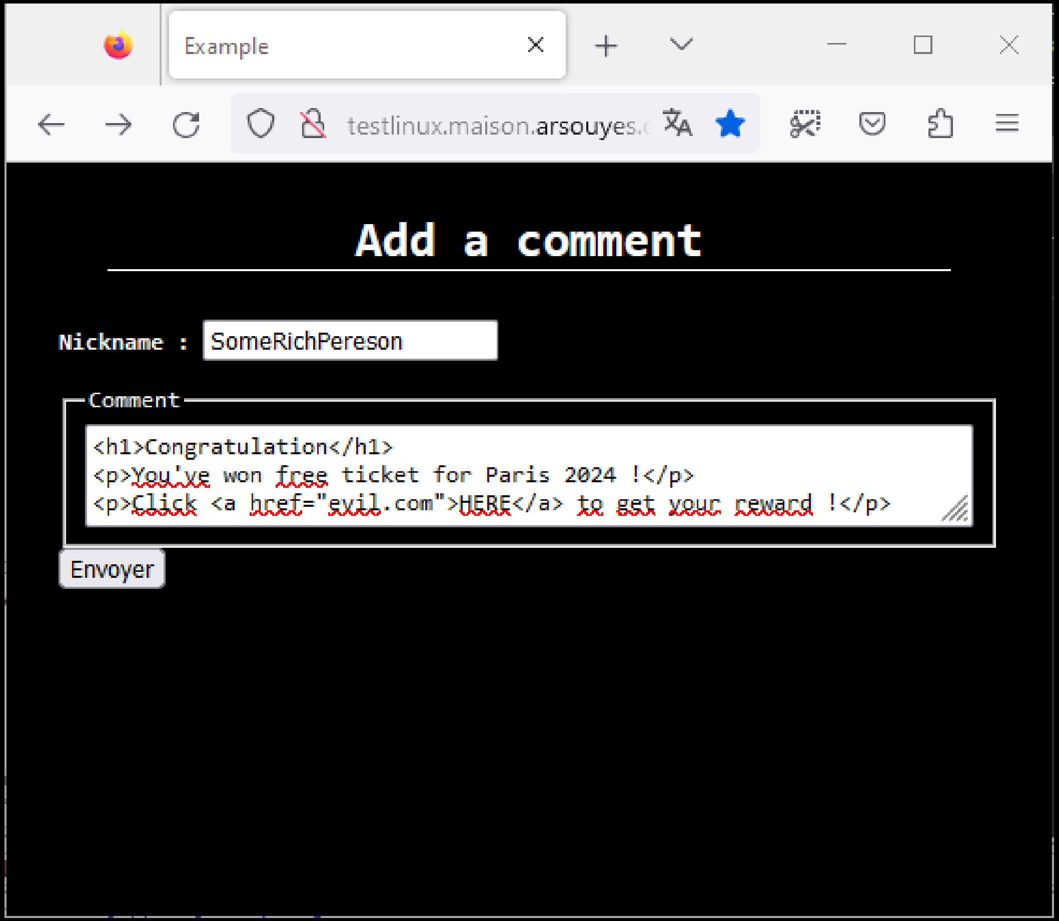


By : tbowan

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam facilisis nisi mi, quis dictum lacus maximus in. Donec eget sapien lectus. In hac

Example of persistant injection

i.e. adding some fishing HTML

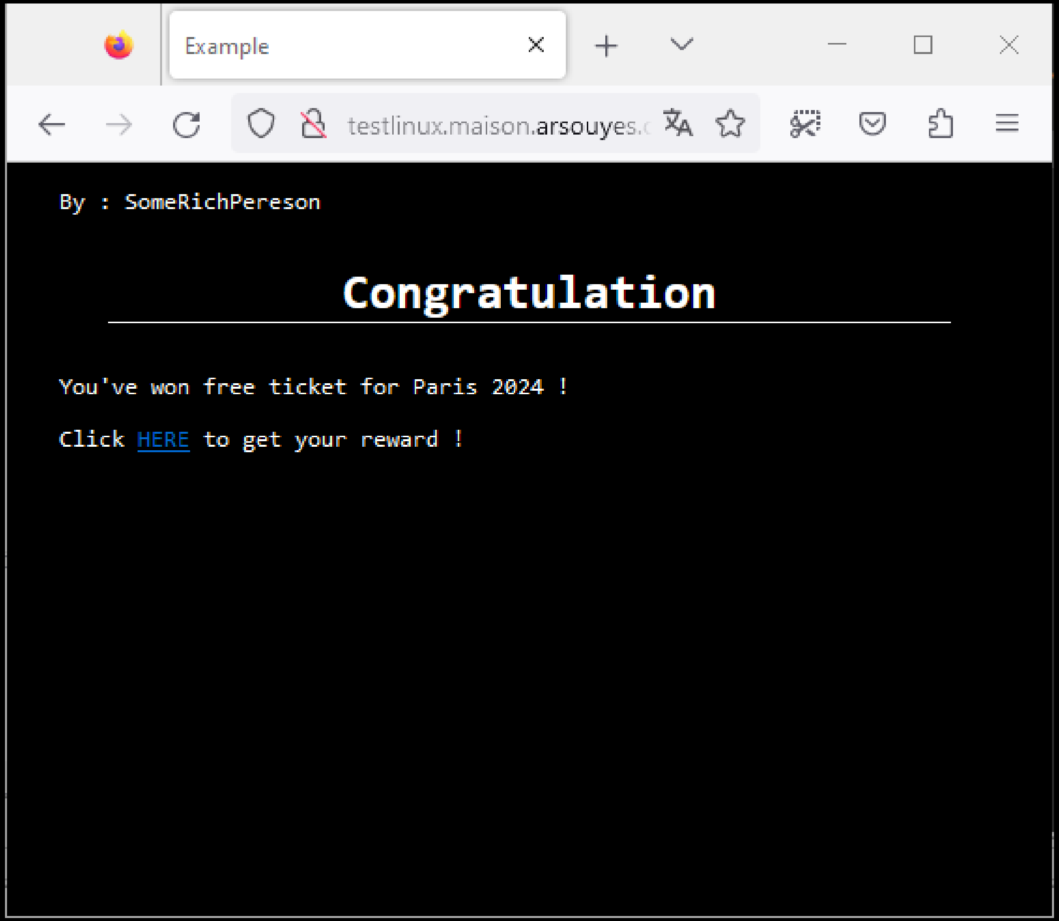


Add a comment

Nickname : SomeRichPereson

Comment

```
<h1>Congratulation</h1>
<p>You've won free ticket for Paris 2024 !</p>
<p>Click <a href="evil.com">HERE</a> to get your reward !</p>
```



By : SomeRichPereson

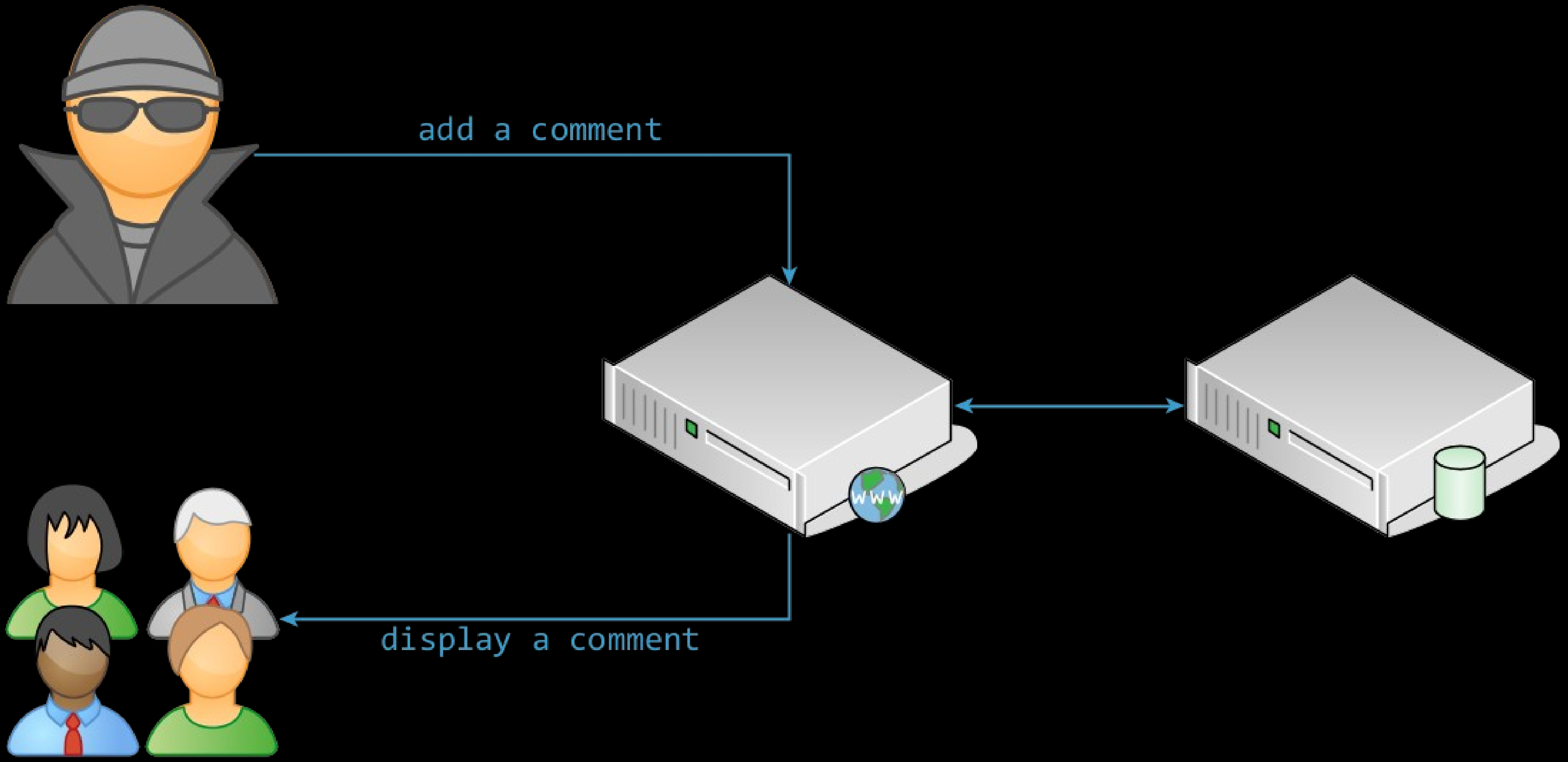
Congratulation

You've won free ticket for Paris 2024 !

Click HERE to get your reward !

Principle of stored XSS

Use vulnerable website to host your evil content



Risks

Information theft

(cookies, form data, …)

Botnet

(relay for other attacks, crypto mining, …)

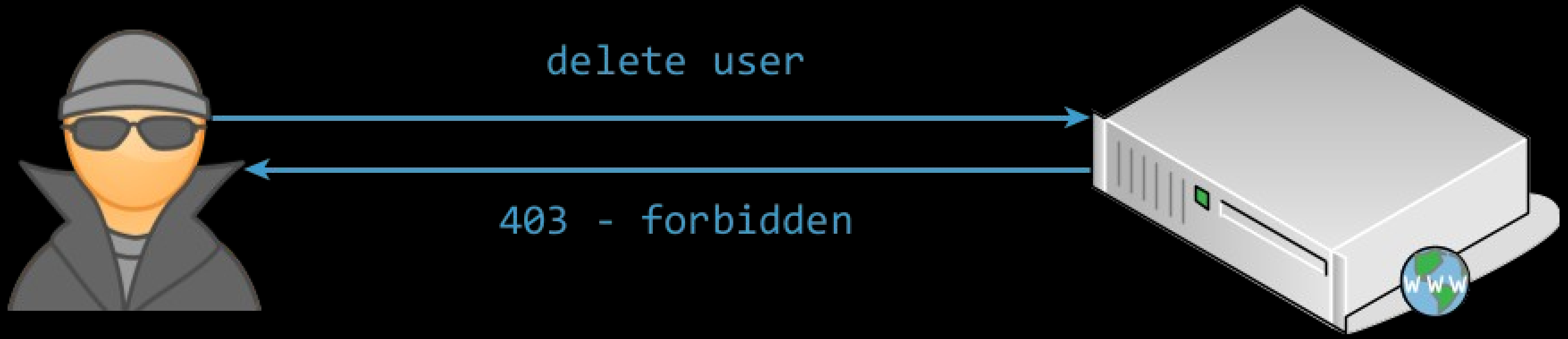
Request execution

(XSRF)

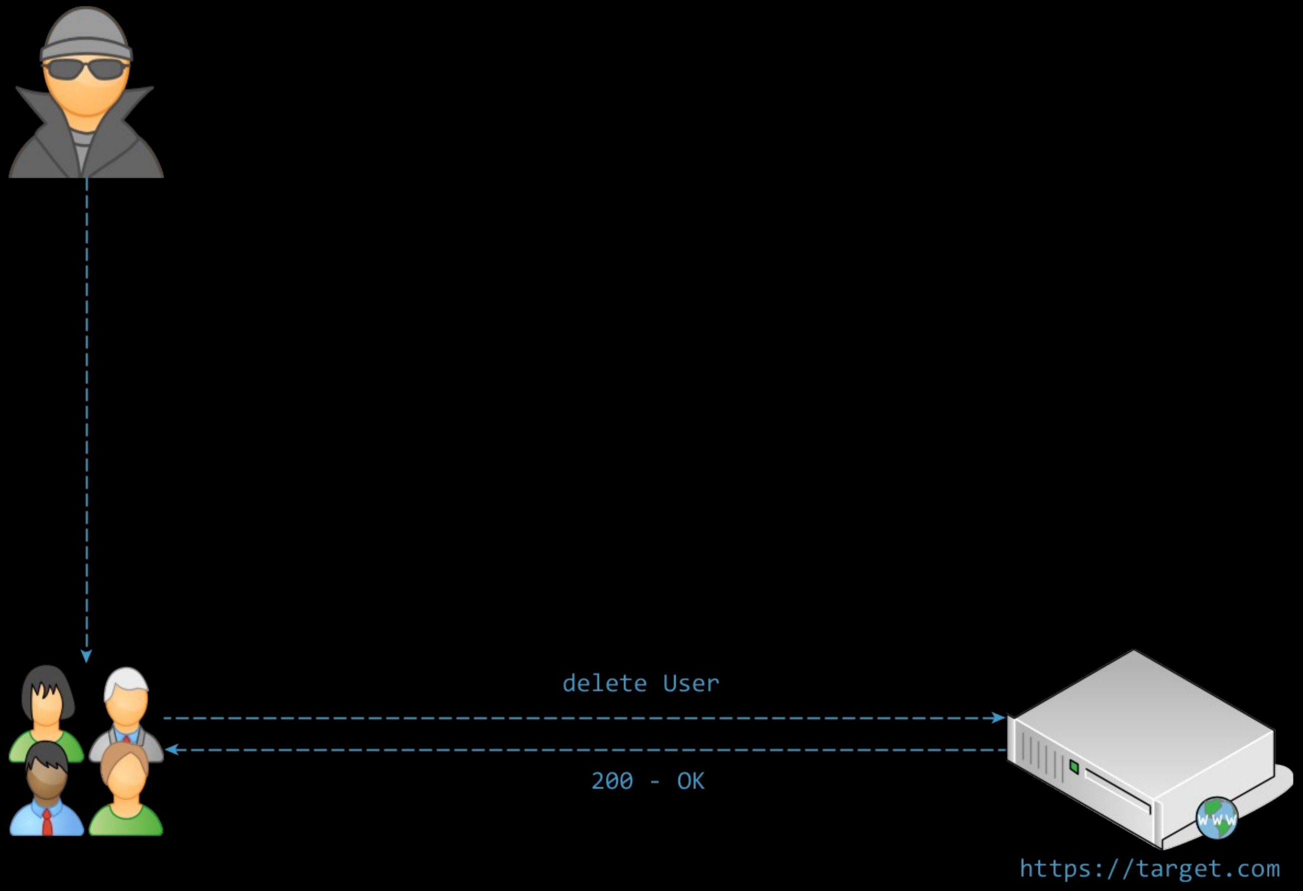
XSRF / CSRF

Cross Site Request Forgery

Principle
protected feature

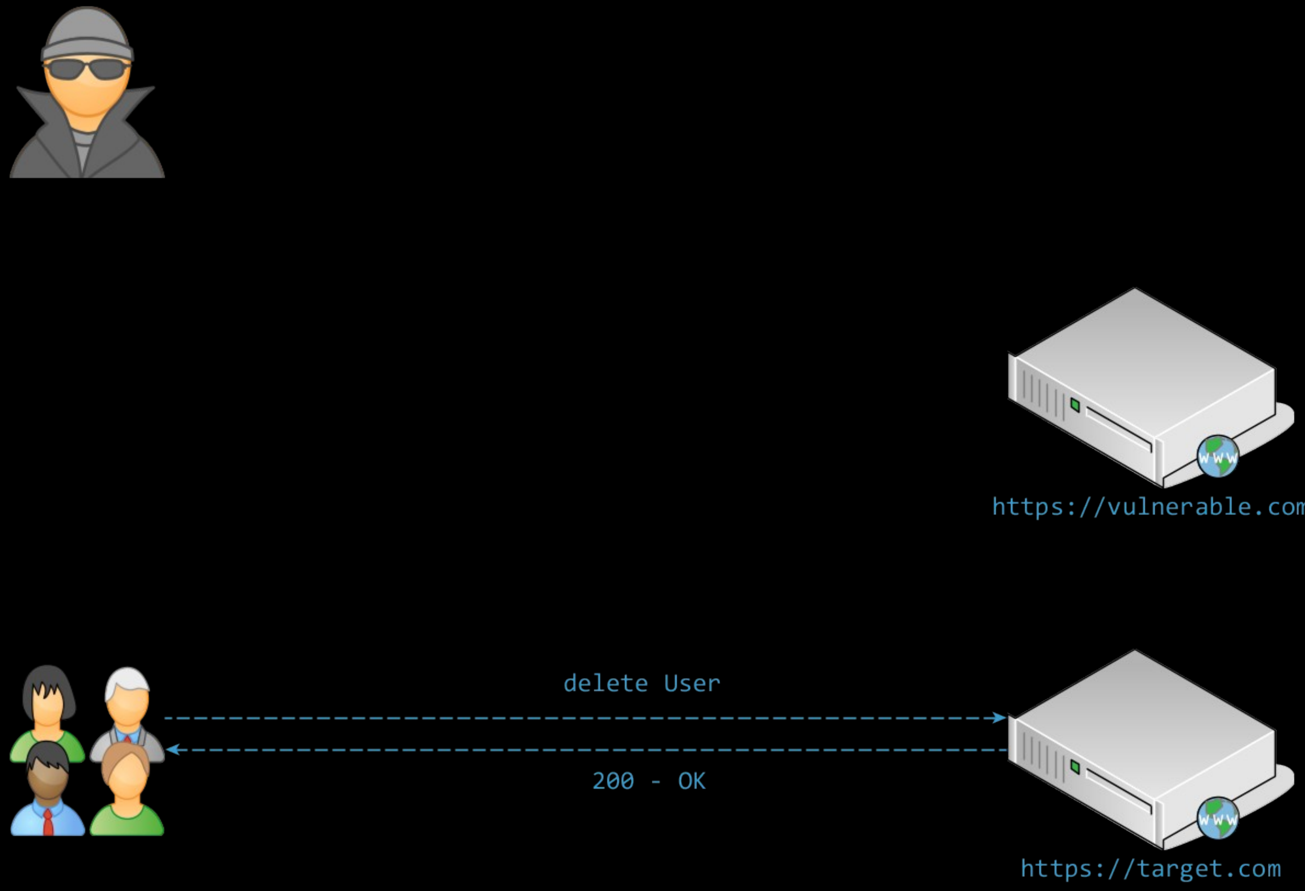


Principle
Need some admin to do the call

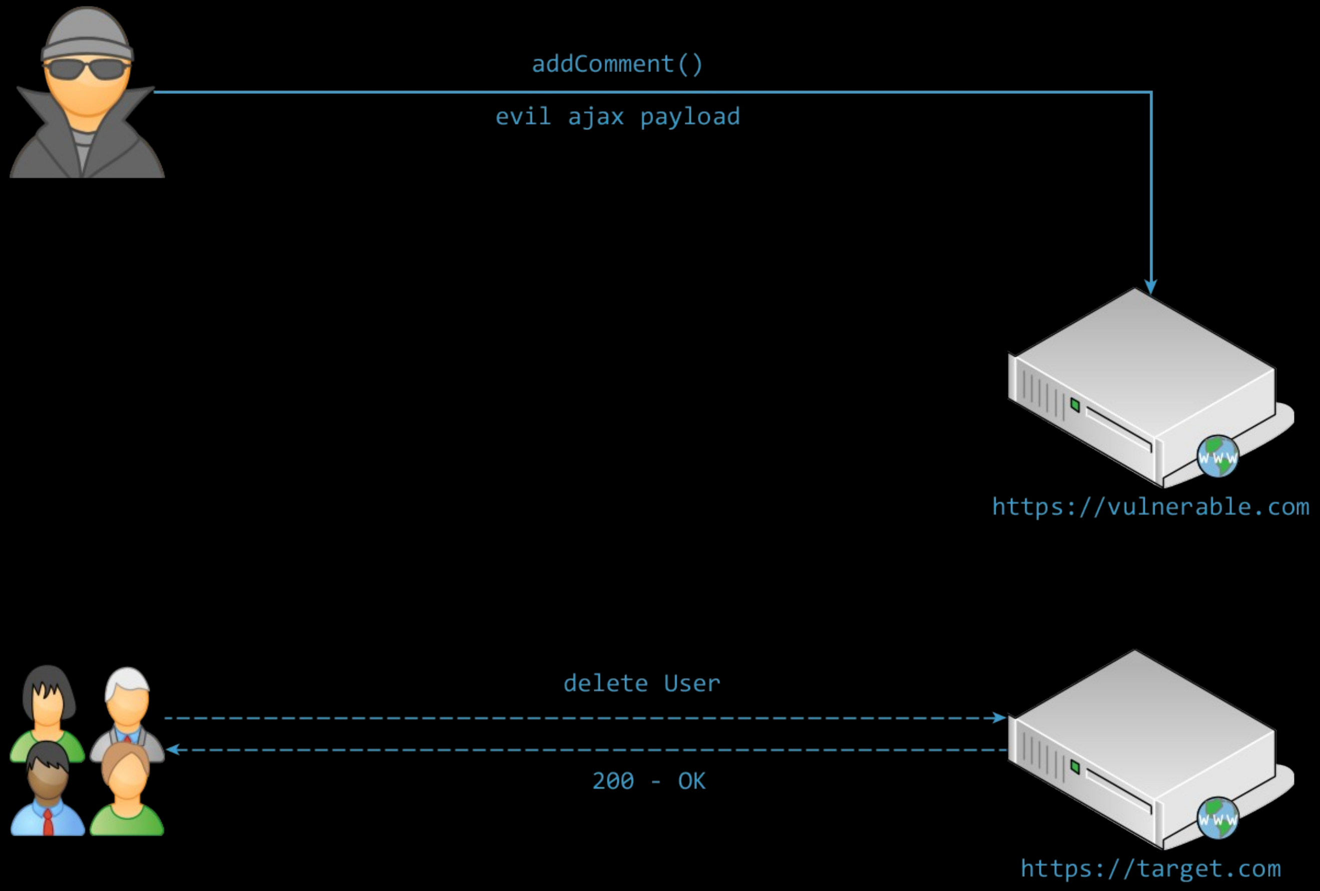


Principle

Exploit a third (vulnerable) party

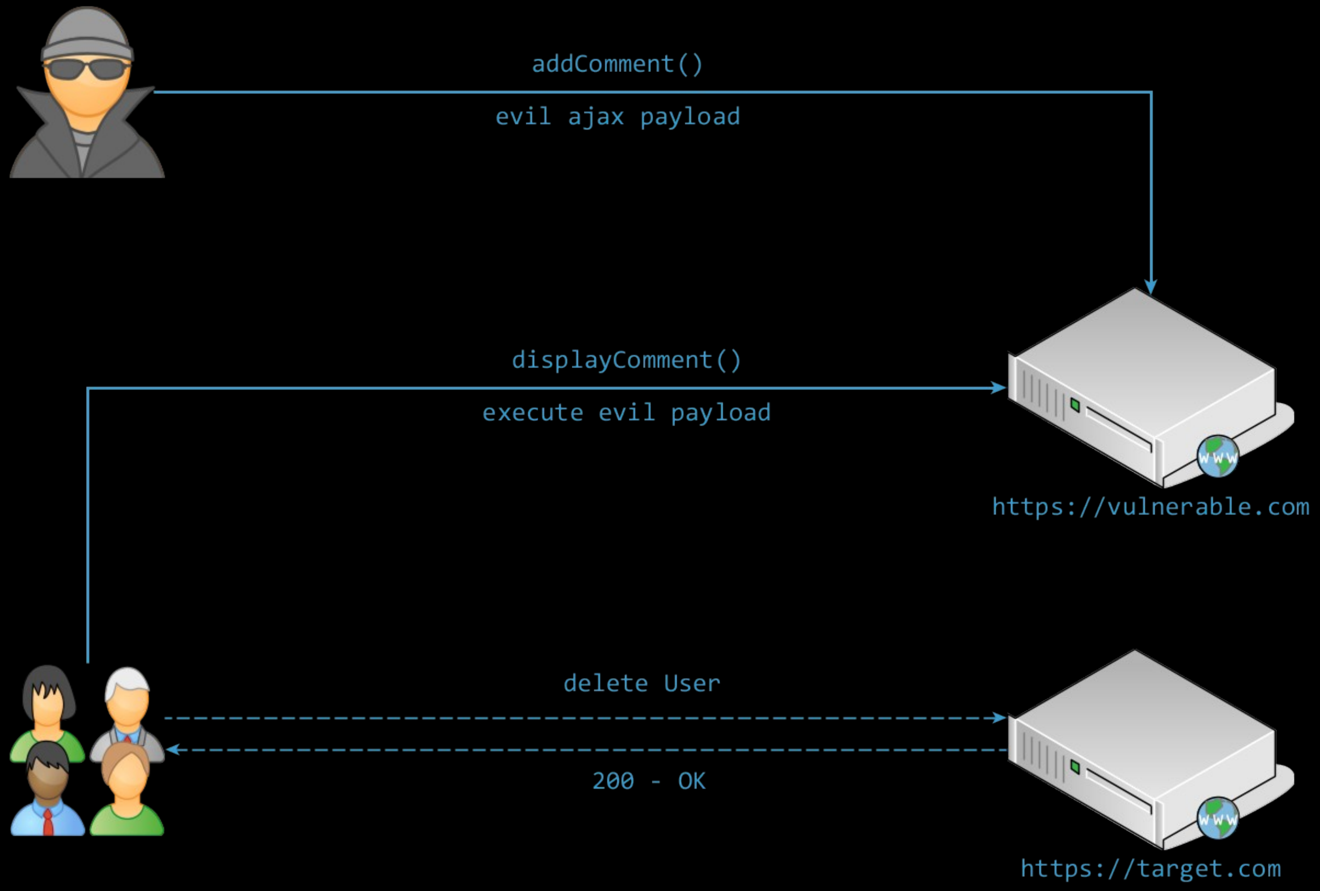


Principle
Inject Ajax that do the call

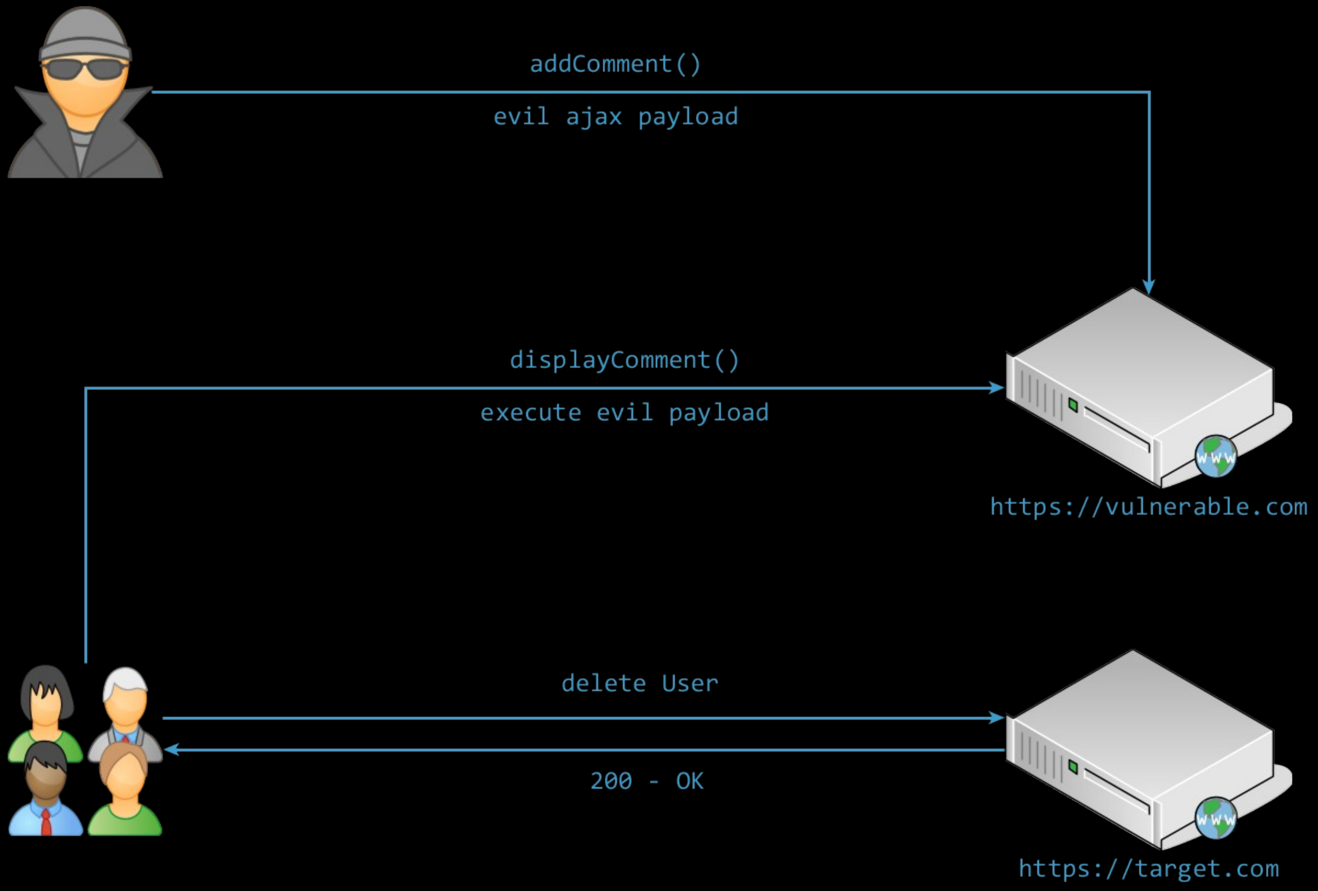


Principle

Wait some admin get the payload



Principle
Enjoy



Protections

Where to fix it ?

Complementary layers

Client Side

- Javascript

Server Side

- PHP, Java, …

Protocols

- Cookies, CORS, …

Browsers

- Chrome, firefox, …

Third parties

- Libs, …

Server side

PHP

Escape / Delete tags

`Htmlspecialchars, Htmlentities, strip_tags`

Encode attributes

`Urlencode`

Client side

Javascript

Escape / Delete tags

`Depends on frameworks`

Use html5 <template>

`textContent` *vs.* `innerHtml`

Cookies

Protocol (app + browers)

Expires

(validity duration)

Domain

(validity on domain name)

Path

(path of resources)

Secure

(transmit only if TLS)

HttpOnly

(only send to server)

SameSite

(transmit only to same site)

SOP

Same-Origin-Policy

Same Origin

Two resources share same origin if…

Same protocol

(http, https, ftp, …)

Same domain name

Same port

(80, 443, 8080, 8443, …)

Politic for other origins

Mainly for XmlHttpRequest()

Cannot access other content

But can be embeded in html

(scripts, img, video, forms, ...)

Can do requests

(GET et POST)

CORS

Cross Origin Ressource Sharing

Principle

Finer grained request to outside

New HTTP Header

Browser ask for rights

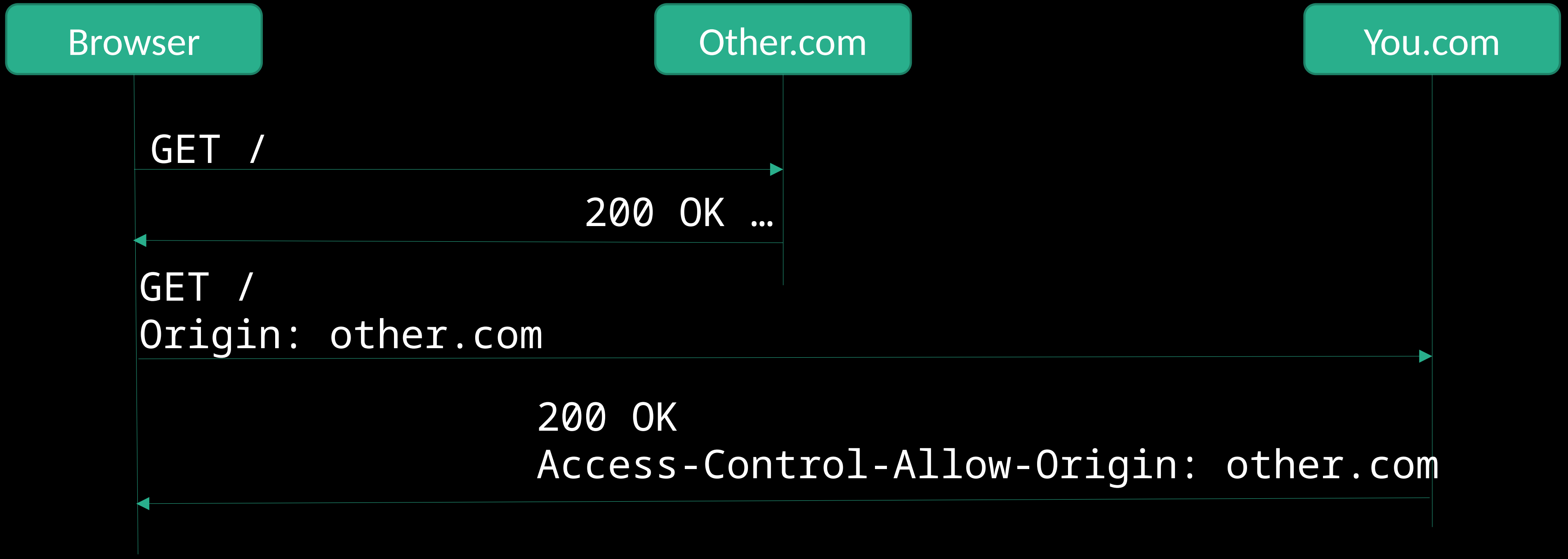
`(Origin, Access-Control-Request-Method)`

Server check/setup rights

`(Access-Control-Allow-Origin, Access-Control-Allow-Methods)`

Simple request

(GET, POST, HEAD + content type)



```
GET /
```

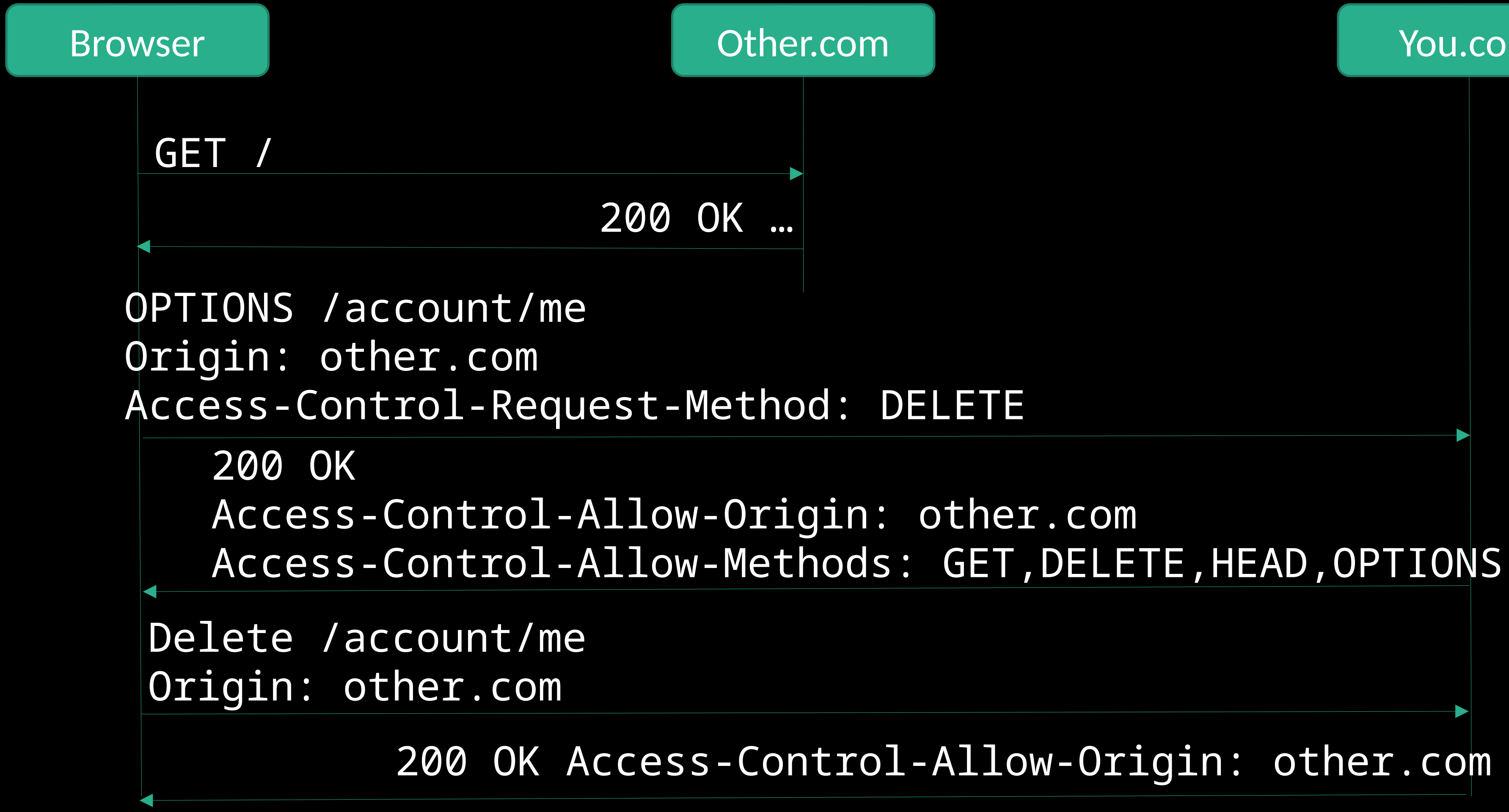
```
200 OK …
```

```
GET /
Origin: other.com
```

```
200 OK
Access-Control-Allow-Origin: other.com
```

« preflight » Request

(everything else)



```
GET /
```

```
200 OK …
```

```
OPTIONS /account/me
Origin: other.com
Access-Control-Request-Method: DELETE
```

```
200 OK
Access-Control-Allow-Origin: other.com
Access-Control-Allow-Methods: GET,DELETE,HEAD,OPTIONS
```

```
Delete /account/me
Origin: other.com
```

```
200 OK Access-Control-Allow-Origin: other.com
```

CSP

Content Security Policy

Principle : Headers

Server set the policy

HTTP header

(Content-Security-Policy)

HTML header

(meta, Content-Security-Policy)

Principle : rules

Restriction on usable origins

Type of contents

(default-src, script-src, style-src, …)

Allowed Origin

(‘self’, domaine, protocole+domaine)

Principle : reports

Error notification to an endpoint

A URL

(to get JSON report from browsers)

A mode « report only »

(To check policy before going to production)

Anti CSRF

Available techniques

CSRF Token

Server generate random value

(unique for each session)

Put on a form

`<input type=hidden>`

Checked on submit

Double submit

Idem but…

Cookie instead of session

Variants

(ciphered / signed cookie)

Re-authentification

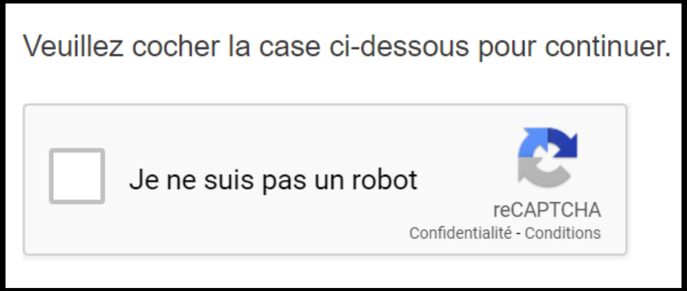
Re-ask for password

(for important requests only)

Captcha

Turing test

(painfull for humans)



Veuillez cocher la case ci-dessous pour continuer.

□ Je ne suis pas un robot

reCAPTCHA

Confidentialité - Conditions