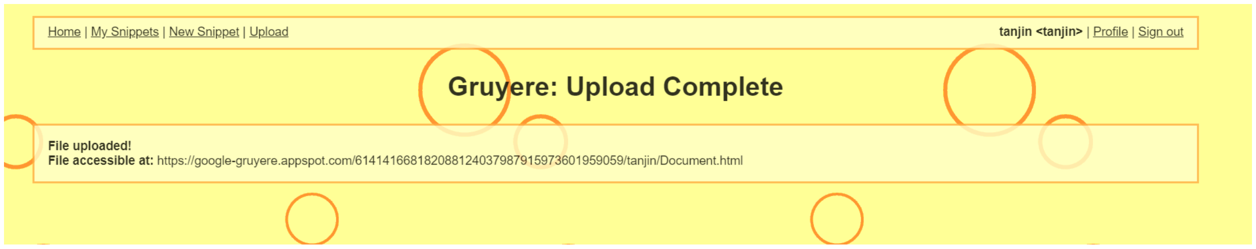
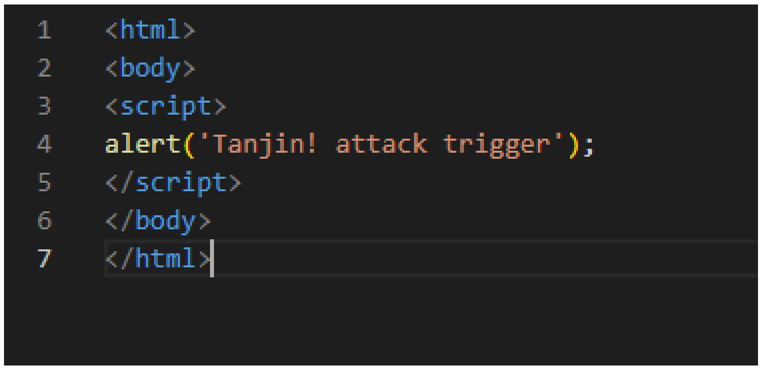
Challenge 1
File Upload XSS

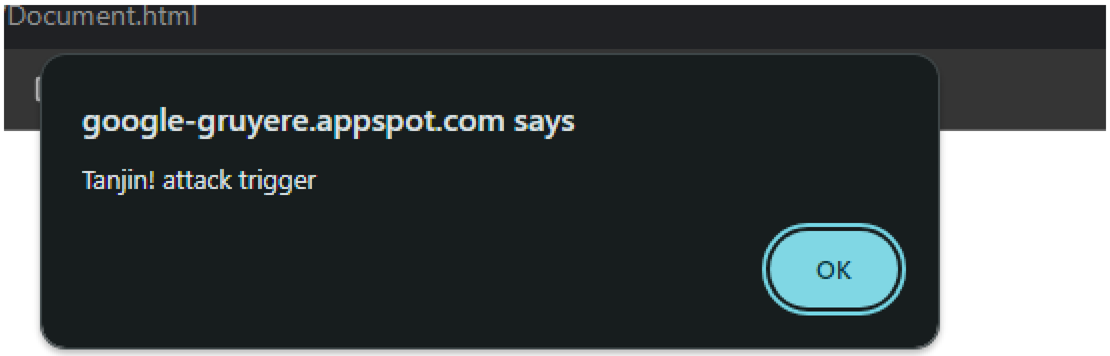

Home | My Snippets | New Snippet | Upload
tanjin <tanjin> | Profile | Sign out

Gruyere: Upload Complete

File uploaded!
File accessible at: https://google-gruyere.appspot.com/614141668182088124037987915973601959059/tanjin/Document.html



```
<html>
<body>
<script>
alert('Tanjin! attack trigger');
</script>
</body>
</html>
```

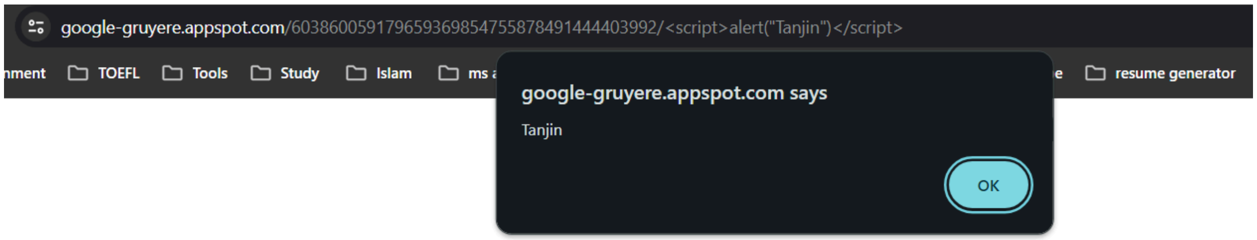


google-gruyere.appspot.com says

Tanjin! attack trigger

This challenge involves uploading a file with a malicious script embedded in it. The goal is to exploit any vulnerability in the file upload functionality that allows the uploaded script to be executed when the file is accessed or opened. I have uploaded a malicious file and the url triggered the attack.

Challenge 2
Reflected XSS

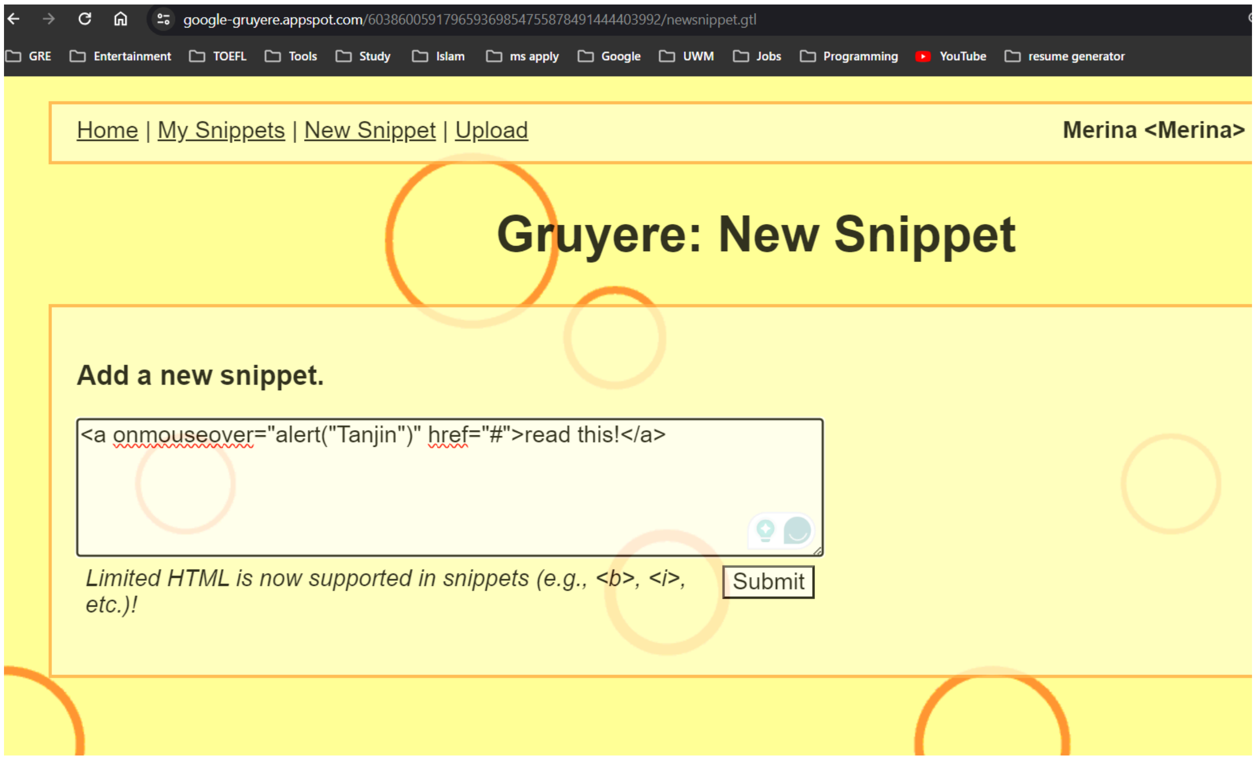

google-gruyere.appspot.com/603860059179659369854755878491444403992/<script>alert("Tanjin")</script>

google-gruyere.appspot.com says

Tanjin

In this challenge, I find a reflected XSS vulnerability by injecting scripted code into a URL parameter that gets reflected in the application's response. This can bypass certain browser protections against XSS attacks.

Challenge 3
Stored XSS

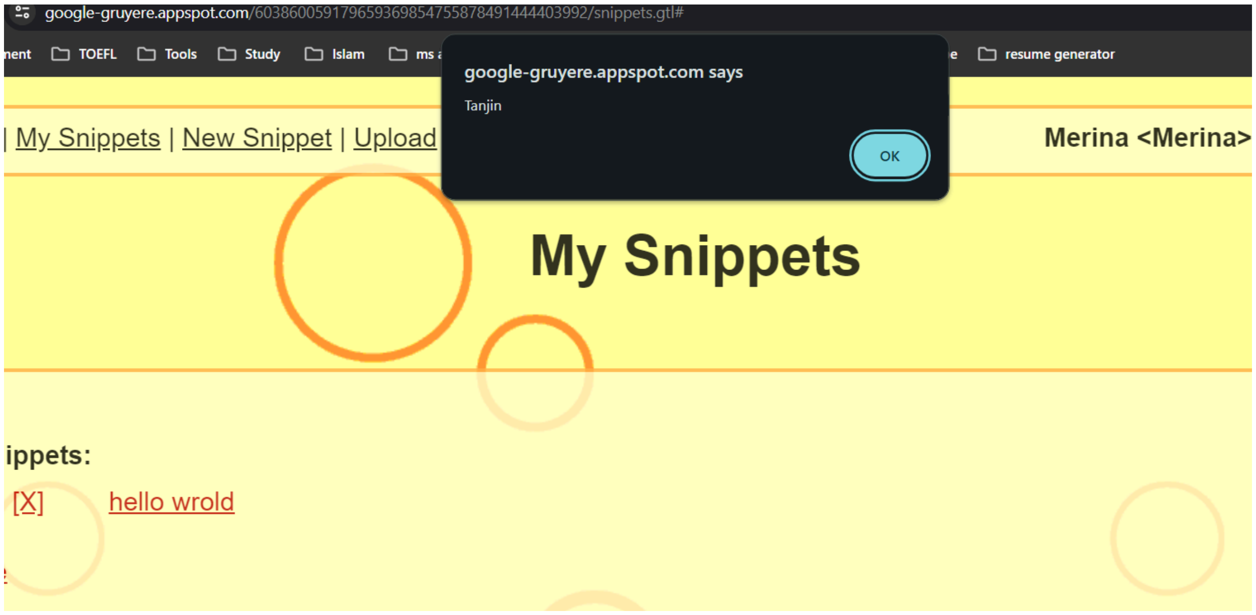

google-gruyere.appspot.com/603860059179659369854755878491444403992/newsnippet.gtl

Gruyere: New Snippet

Add a new snippet.

```
<a onmouseover="alert("Tanjin")" href="#">read this!</a>
```

Limited HTML is now supported in snippets (e.g., <b>, <i>, etc.)!

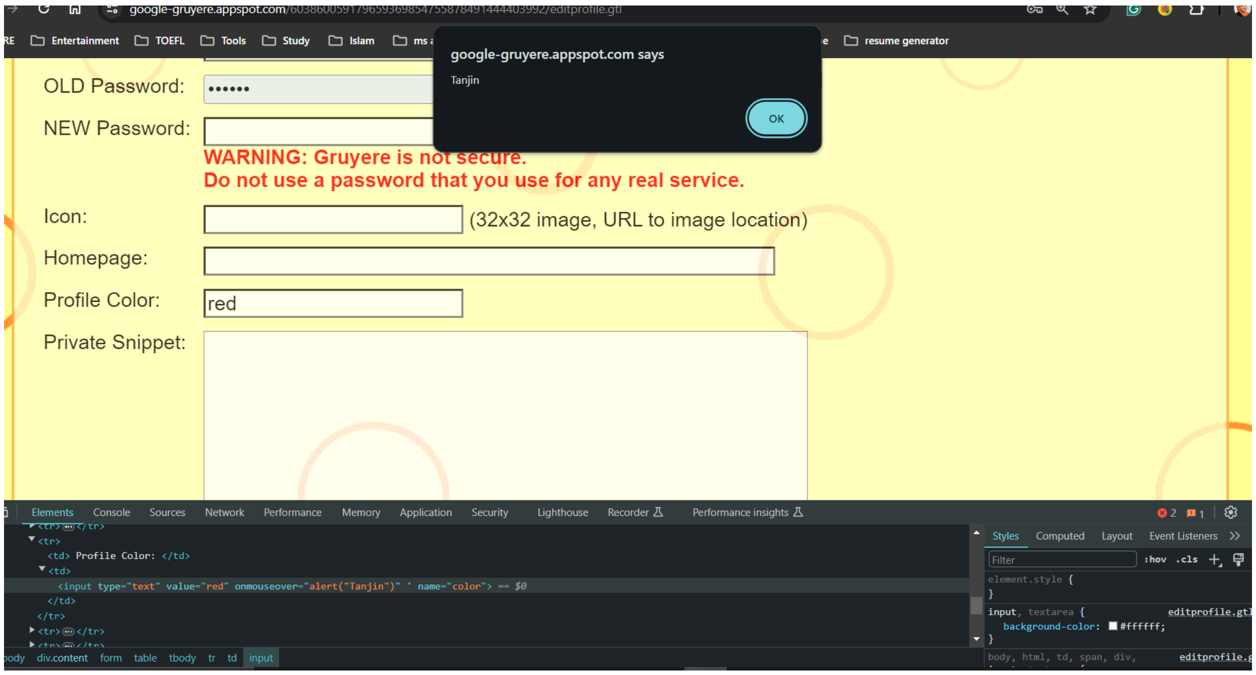

google-gruyere.appspot.com/603860059179659369854755878491444403992/snippets.gtl#

google-gruyere.appspot.com says

Tanjin

I have identified a stored XSS vulnerability where user-provided data is stored by the application and later served back to other users without proper sanitization, allowing the injected script to be executed in their browsers. the objective is to input a script into a form field that gets stored in the application's database. When the stored data is later displayed on a page without proper escaping, the script gets executed in the browsers of other users viewing the page.

Challenge 4
Stored XSS via HTML Attributes


google-gruyere.appspot.com says

Tanjin

OLD Password: ••••••

NEW Password:

WARNING: Gruyere is not secure.
Do not use a password that you use for any real service.

Icon: (32x32 image, URL to image location)

Homepage:

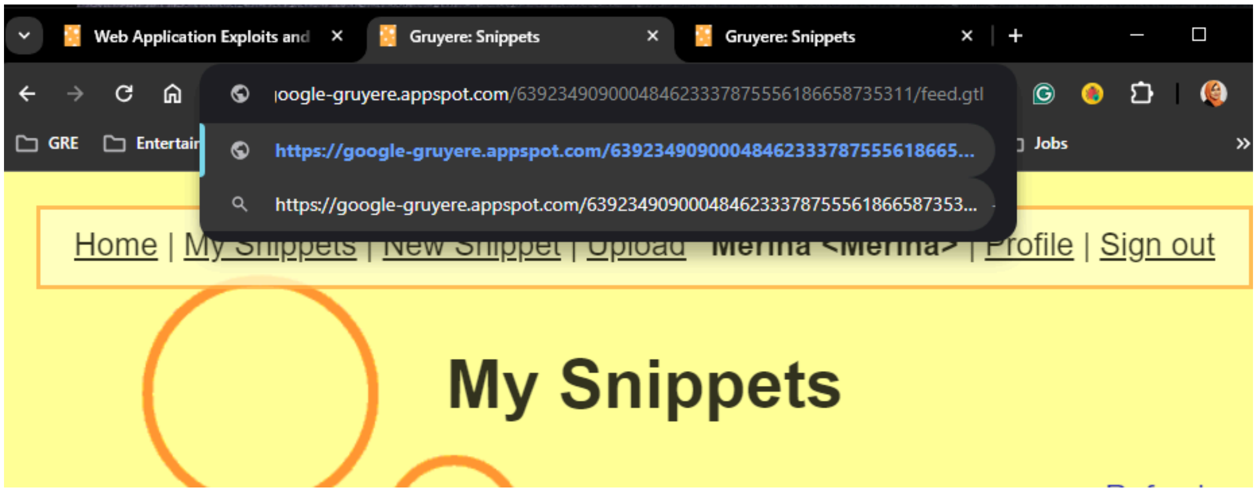
Profile Color: red

Private Snippet:

```
<tr>
  <td> Profile Color: </td>
  <td>
    <input type="text" value="red" onmouseover="alert("Tanjin")" ' name="color"> == $0
  </td>
</tr>
```

I injected a script into an HTML attribute value, particularly by setting the color value in a user profile. This lead to XSS as the application does not properly sanitize or escape the attribute value when rendering the HTML.

Challenge 5
Stored XSS via AJAX

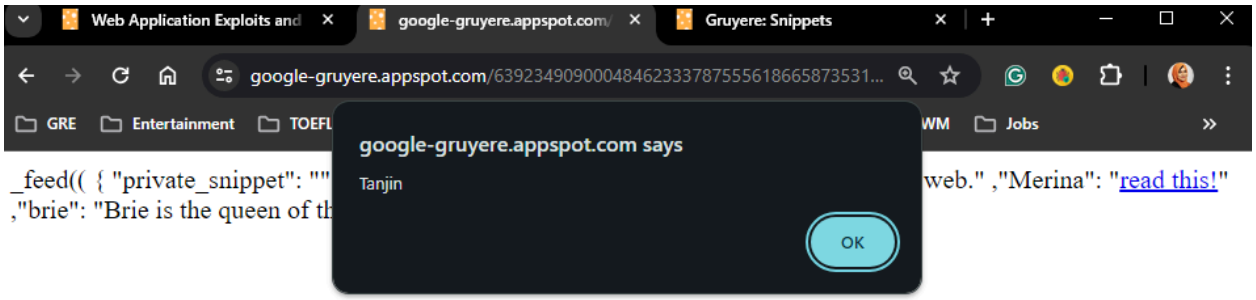

google-gruyere.appspot.com/639234909000484623337875556186658735311/feed.gtl

https://google-gruyere.appspot.com/63923490900048462333787555618665...

https://google-gruyere.appspot.com/6392349090004846233378755561866587353...

Home | My Snippets | New Snippet | Upload Merina <Merina> | Profile | Sign out

My Snippets

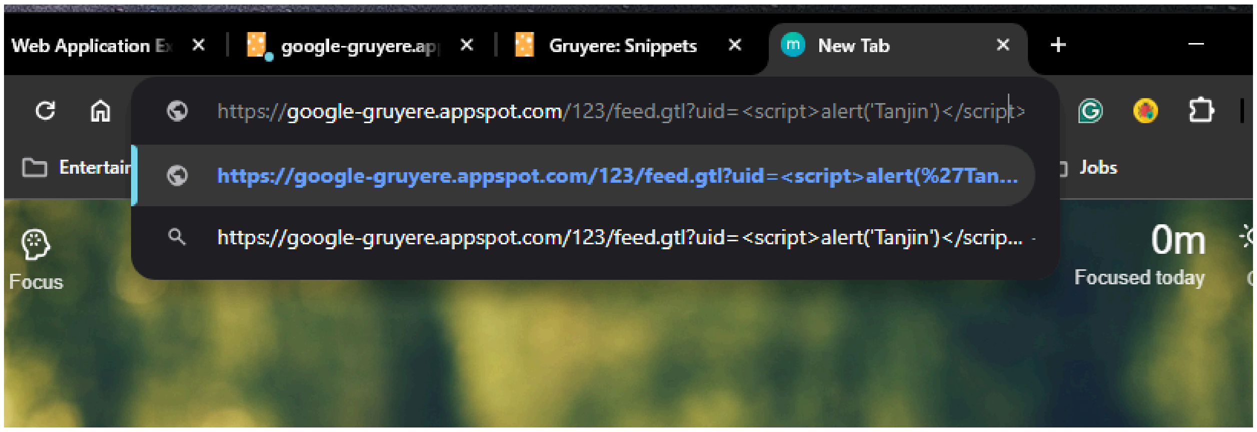

google-gruyere.appspot.com says

Tanjin

_feed(({ "private_snippet": "" ... web." ,"Merina": "read this!"
,"brie": "Brie is the queen of th

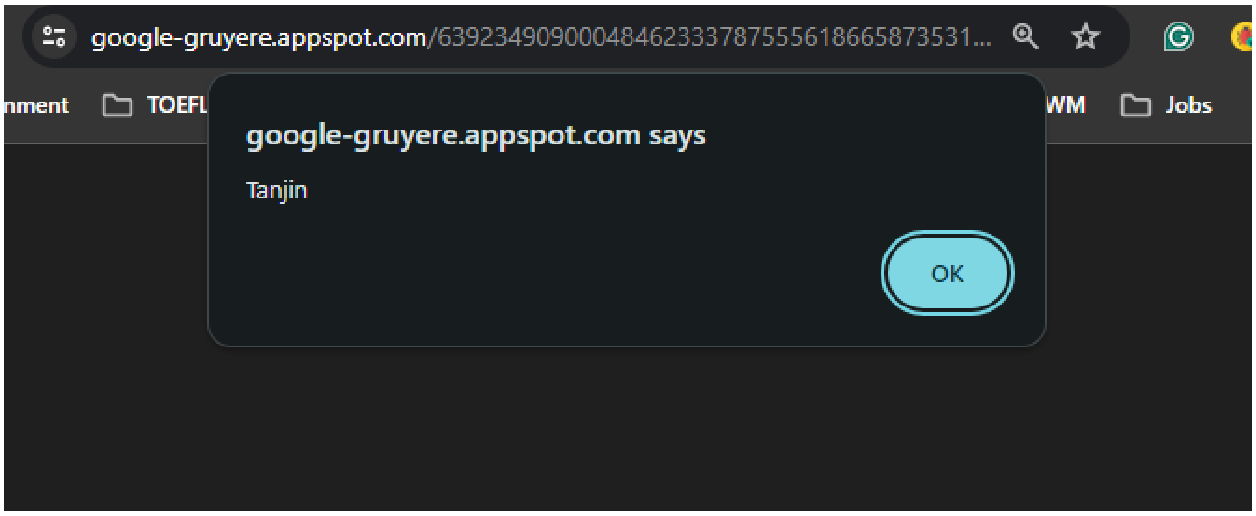
I exploited a vulnerability in Gruyere's AJAX code to trigger an XSS attack when the page's refresh link is clicked. The attack is triggered by inserting unexpected content such as ID into a snippet that gets evaluated on the client side. I submit a malicious script via an AJAX request, which gets stored in the application's database. When the response containing the stored script is rendered, it gets executed in the context of other users' browsers.

Challenge 5
Reflected XSS via AJAX


https://google-gruyere.appspot.com/123/feed.gtl?uid=<script>alert('Tanjin')</script>

https://google-gruyere.appspot.com/123/feed.gtl?uid=<script>alert(%27Tan...

https://google-gruyere.appspot.com/123/feed.gtl?uid=<script>alert('Tanjin')</scrip...


google-gruyere.appspot.com/63923490900048462333787555618665873531...
nment
TOEFL
WM
Jobs
OK

google-gruyere.appspot.com says

Tanjin

I found a URL that, when clicked, will execute a script using one of Gruyere's AJAX features. This involves exploiting a reflected XSS vulnerability within an AJAX request-response cycle. the injection and reflection of the script occur within an AJAX request and response cycle. I inject the script into an AJAX request parameter, which is then reflected back in the response without proper sanitization, leading to XSS.