TYPO3 Core - Bug #80888

GeneralUtility::removeXSS() doesn't respect base64 encoded links

2017-04-19 09:54 - Alex Kellner

Status:	Rejected	**Start date:**	2017-04-19
Priority:	Must have	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
TYPO3 Version:	8	**Complexity:**	
PHP Version:		**Is Regression:**	
Tags:		**Sprint Focus:**	

Description

In some projects we're using GeneralUtility::removeXSS() for user variables.
It turned out, that this is a failure. Base64 encoded links are not disarmed.

Example test.php in TYPO3 Webroot:

```
<?php
require __DIR__ . '/vendor/autoload.php';

$string = 'data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K';
echo '<a href="' . $string . '">1</a>';
echo \TYPO3\CMS\Core\Utility\GeneralUtility::removeXSS('<a href="' . $string . '">2</a>');
```

Related issues:

Related to TYPO3 Core - Task #76164: Deprecate RemoveXSS	**Closed**	**2016-05-12**

History

#1 - 2017-04-21 22:08 - Georg Ringer

- Related to Task #76164: Deprecate RemoveXSS added

#2 - 2017-04-21 22:10 - Georg Ringer

- Status changed from New to Needs Feedback

I am in favor of closing this issue. Since 8.2 GeneralUtility::removeXSS has been marked as deprecated and has been removed with 9. see https://docs.typo3.org/typo3cms/extensions/core/8-dev/Changelog/8.2/Deprecation-76164-DeprecateRemoveXSS.html for details

is that ok for you?

#3 - 2017-04-22 21:33 - Alex Kellner

What does that mean? Even if there is a removeXSS() function (ok, it will be removed in future), it is not secure?
In my eyes, you should offer a secure function or remove it now if it's unsecure.

Beside that: Are there any known methods to allow HTML in fronten from user input without XSS-problems?

#4 - 2017-04-28 15:28 - Alexander Opitz

The removeXSS() function works by blacklisting cases that are known as XSS issue, but it can't know what is all around it and what will be newly implemented by the browsers.
As it works with blacklisting it is not 100% secure and won't be.

So you should use a way, which works with Markdown, Wikimedia or other syntax or use a whitelisting function where you explicitly say what is allowed (like good old RTE in backend).

A solution for you may be to not allow "data:" protocol in the RemoveXSS class. Try following:

- Open RemoveXSS class file (typo3/sysext/core/Resources/PHP/RemoveXSS.php)
- Go to line 105 "$protocolKeywords ="
- change to $protocolKeywords = ['javascript', 'vbscript', 'expression', 'data'];

- Try if this helps for your issue, if ok, I may do patches.

BTW: Do you really want, that someone can link to everywhere?

#5 - 2017-05-03 10:52 - Alex Kellner

- Assignee set to Alexander Opitz

I understand the problem with blacklisting in a progressive environment.

I will try to not use removeXSS() by default in future (but I'm not sure where we used it in the past).

I added

```
$protocolKeywords = ['javascript', 'vbscript', 'expression', 'data'];
```

but this doesn't help as it turned out in a small test.

#6 - 2017-05-03 14:02 - Alexander Opitz

- Assignee deleted (Alexander Opitz)

#7 - 2017-08-01 12:20 - Jigal van Hemert

- File removexssdatauri.patch added

- Status changed from Needs Feedback to Rejected

I found this issue in the past and back then made a patch against 6.2. We decided not to include it because a blacklist approach will always have issues and provides a false sense of security. RemoveXSS would be deprecated (which is now the case). It would be possible to include a whitelist based tool, but using an RTE (possibly with limited functionality) would be a nicer solution anyway.

Setting the issue to rejected because deprecated functionality should not be used any more and it should not be expected that deprecated functions get new functionality. If it's absolutely needed the code in the patch can be used or better, use a tool like HTML Purifier to clean the HTML.

#8 - 2018-01-10 11:01 - Alex Kellner

- Priority changed from Should have to Must have

The same problem is still located in TYPO3 6, 7 and 8 in ContentObjectRenderer.php in removeBadHTML() function

Every TYPO3 instance in the wild that is using TypoScript with removeBadHTML is vulnerable

```
# page.1 is save
page.1 = TEXT
page.1.value = <span onclick="alert('xss')">click</span>
page.1.removeBadHTML = 1

# page.2 is unsave
page.2 = TEXT
page.2.value = <a href="data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K">click</a>
page.2.removeBadHTML = 1
```

That is exactly the same problem like in GeneralUtilty::removeXSS()

In my eyes that must be fixed on both places as long as TYPO3 offers both API-methods!

#9 - 2018-01-11 11:46 - Helmut Hummel

Alex Kellner wrote:

> In my eyes that must be fixed on both places as long as TYPO3 offers both API-methods!

The concept (blacklist) of both is broken as mentioned here already. This cannot be fixed and therefore we removed removeXSS and removeBad in TYPO3 9
Therefore using any of those (removeBadHTML or removeXSS) and **not** encoding properly instead, would leave the code vulnerable.

Just because at some point such API was introduced, does not mean it is safe to use it today. Don't get me wrong, this should not be generalized, but in this case we learned (the hard way)
That this mitigation just not suffice and proper encoding is mandatory.

Files

removexssdatauri.patch	6.79 KB	2017-08-01	Jigal van Hemert