Tue. 17th November, 2020

TYPO3-CORE-SA-2020-009: Cross-Site Scripting through Fluid view helper arguments

Categories: Development (/help/security-advisories/development), TYPO3 CMS (/help/security-advisories/typo3-cms)
Created by Claus Due

It has been discovered that the Fluid Engine is vulnerable to cross-site scripting.

- **Component Type:** TYPO3 CMS
- **Subcomponent:** Fluid Engine (package typo3fluid/fluid)
- **Release Date:** November 17, 2020
- **Vulnerability Type:** Cross-Site Scripting
- **Affected Versions:** 10.0.0-10.4.9, 9.0.0-9.5.22, 8.7.0-8.7.37 ELTS, 7.6.0-7.6.47 ELTS, 6.2.0-6.2.53 ELTS
- **Severity:** Medium
- **Suggested CVSS:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N/E:F/RL:O/RC:C (https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N/E:F/RL:O/RC:C&version=3.1)
- **References:** CVE-2020-26216 (https://nvd.nist.gov/vuln/detail/CVE-2020-26216), CWE-79 (https://cwe.mitre.org/data/definitions/79.html)

Problem Description

Three XSS vulnerabilities have been detected in Fluid:

1. TagBasedViewHelper allowed XSS through maliciously crafted *additionalAttributes* arrays by creating keys with attribute-closing quotes followed by HTML. When rendering such attributes, TagBuilder would not escape the keys.
2. ViewHelpers which used the *CompileWithContentArgumentAndRenderStatic* trait, and which declared *escapeOutput = false*, would receive the content argument in unescaped format.
3. Subclasses of AbstractConditionViewHelper would receive the *then* and *else* arguments in unescaped format.

Solution

Update to TYPO3 version 10.4.10, 9.5.23, 8.7.38 ELTS, 7.6.48 ELTS or 6.2.54 ELTS that fix the problem described.
Update to versions 2.0.8, 2.1.7, 2.2.4, 2.3.7, 2.4.4, 2.5.11 or 2.6.10 of the underlying standalone *typo3fluid/fluid* package.

Updated versions of this package are bundled in following TYPO3 (typo3/cms-core) releases:

- TYPO3 v9.5.23 (using typo3fluid/fluid v2.6.10)
- TYPO3 v10.4.5 (using typo3fluid/fluid v2.6.10)

The specific vulnerabilities are prevented by:

1. Explicitly escaping keys found in the *additionalAttributes* array passed to a TagBasedViewHelper before using them as attribute names.
2. Detecting "content argument" on ViewHelpers using the trait CompileWithContentArgumentAndRenderStatic and escaping it based on the state of *escapeChildren* when *escapeOutput* is toggled off. Escaping still will not occur if explicitly disabled by an enclosing ViewHelper. This homogenises escaping behavior of "content arguments" so the same strategy is used whether the "content" argument is passed as argument or child content.

Explicitly defining the *then* and *else* arguments on AbstractConditionViewHelper subclasses as escaped and applying escaping in all cases where escaping is not explicitly disabled by an enclosing ViewHelper.

Affected Cases

1. The fix for TagBasedViewHelper does not affect any valid use cases; it only prevents use of maliciously crafted attribute/value arrays passed as *additionalAttributes*.
2. Any case where a ViewHelper with a "content argument" and which defines *escapeOutput = false* is used with the content argument instead of passing variables as child node - e.g. *<v:h content="{variable}" />* instead of *<v:h>{variable}</v:h>* to intentionally circumvent escaping of any HTML in *{variable}*.
3. Any case where a condition ViewHelper is used with *then* or *else* arguments to render a variable containing HTML, excluding cases where the variable is intentionally unescaped - e.g. *<f:if condition="1" then="{variable -> f:format.raw()}" />*, and excluding any cases where a ViewHelper is used as argument value and the ViewHelper intentionally disables escaping - e.g. *<f:if condition="1" then="{f:render(section: 'MySection')}" />* does not escape the *then* argument because *f:render* disables output escaping.

Cases 2 and 3 can be mitigated to allow variables with HTML to not be escaped, by intentionally disabling escaping by chaining the variable used in the argument with *f:format.raw* as described in case 3. Note that this constitutes a potential security issue, for which the template author is solely responsible.

Example: *<f:if condition="1" then="{intentionalHtmlVariable}" />* can allow HTML in *{intentionalHtmlVariable}* by adding *-> f:format.raw()* - to become *<f:if condition="1" then="{intentionalHtmlVariable -> f:format.raw()}" />*. Variables containing HTML should only be allowed after taking great care to prevent XSS through other means, e.g. sanitising the variable before it is assigned to Fluid or only allowing such variables to come from trusted sources.

Custom ViewHelpers which use *CompileWithContentArgumentAndRenderStatic* can alternatively pass a 6th argument with value *false* to the call to *registerArgument* which registers the "content argument", which explicitly disables escaping of the argument value: *$this->registerArgument('arg', 'string', 'My argument', false, null, false);*. Note that this constitutes a potential security issue for which the ViewHelper author is solely responsible.

Credits

Thanks to Jonas Eberle and Sinan Sekerci (Dreamlab Technologies) who reported this issue and to TYPO3 core merger Claus Due who fixed the issue.

General Advice

Follow the recommendations that are given in the TYPO3 Security Guide (https://docs.typo3.org/typo3cms/CoreApiReference/Security/Index.html#security). Please subscribe to the typo3-announce (http://lists.typo3.org/cgi-bin/mailman/listinfo/typo3-announce) mailing list.

General Note

All security related code changes are tagged so that you can easily look them up in our review system (https://review.typo3.org/#/q/status:merged+project:Packages/TYPO3.CMS+topic:security,n,z).