# TYPO3.Flow - Feature # 4960

| | | | |
|---|---|---|---|
| **Status:** | Resolved | **Priority:** | Must have |
| **Author:** | Sebastian Kurfuerst | **Category:** | Security |
| **Created:** | 2009-10-09 | **Assigned To:** | Sebastian Kurfuerst |
| **Updated:** | 2010-10-20 | **Due date:** | |
| **PHP Version:** | | | |
| **Has patch:** | | | |
| **Complexity:** | | | |
| **Subject:** | There should be a Request hash check when objects are modified | | |

**Description**

## Scenario:

The developer programmed a form to edit a "Customer", and he can select one role from a dropdown list.
This field will have the following name: customer[role], and its expected value will be a UUID.

Now, if an attacker replaces this form field by some fields which are named like:
customer[role][__identity] = ...
customer[role][isAdministrator] = TRUE
... he is able to completely change related domain objects which is like SQL-Injections "built right into the framework". Nice, eh? :-)

## How to solve this issue?

The general idea: If an object is *modified*, make sure that the appropriate form fields have been rendered on the last request. Else, it might be an attack.

## Solution

We build a list of all form fields which can be submitted. This list is signed/hashed (with a private key), and the list and the hash of the list are sent in a parameter called __hmac to the server as well.
If the parameter __hmac is set, it verifies the signature and checks that all submitted fields are also inside the form field list. If this is the case, the parameter "hmacVerified" inside the Request object is set to TRUE.

Remember we only wanted to enable this feature when data was *modified*? That's why we modified the Argument to remember the *origin* of the data (which could be directly from the client, from the persistence layer but unmodified, from the persistence layer and modified, and a new object).
If any argument has been mapped with the cases "persistence layer and modified" and "new object", we make sure the HMAC was validated correctly. If the HMAC was not validated correctly, an exception is thrown.

## How to disable it

For building public APIs, a new annotation @dontverifyrequesthash has been introduced. If you annotate your action with this annotation, no HMAC error will be thrown at all, but you have to care about such security issues yourself.

**Related issues:**

| | | |
|---|---|---|
| related to TYPO3.Flow - Feature # 2817: Provide safeguard for preventing mult... | **Needs Feedback** | **2009-03-10** |
| related to TYPO3.Flow - Major Feature # 5659: Implement content security | **Resolved** | **2009-12-07** |

**Associated revisions**

**Revision 762b8bd6 - 2009-10-09 14:59 - Sebastian Kurfuerst**

[!!!][+FEATURE] FLOW3 (Security): Added a HMAC generator and checker to prevent unauthorized access on objects where no edit fields were generated for. It is mandatory in case objects are modified on the server side. See the issue for a more in-depth explanation. This feature does NOT break backwards-compatibility as long as you use only Fluid for form-generation. In case of custom fields, it WILL break backwards compatibility, and you might need the @dontverifyrequesthash annotation. Resolves #4960.
[+FEATURE] Fluid (ViewHelpers): Added a request hash to all form fields. It is mandatory in case objects are modified on the server side. Relates to

#4960.

**Revision 2fe35f37 - 2009-10-09 15:18 - Sebastian Kurfuerst**

[BUGFIX] FLOW3 (Security): Bugfix to automatic request hashing in context with CLI. Relates to #4960.

**Revision 044fe9a1 - 2009-10-09 15:20 - Sebastian Kurfuerst**

[BUGFIX] FLOW3 (MVC): Bugfix to automatic request hashing in context with CLI. Relates to #4960.

**Revision a81826bd - 2009-10-13 09:17 - Sebastian Kurfuerst**

[+BUGFIX] FLOW3 (Security): Fixed two issues with Request Hashing. Changed hash implementation from normal SHA1 to a real HMAC. Thanks to Markus Krause for pointing this out. Relates to #4960.

**Revision 027a4016 - 2009-12-07 19:09 - Robert Lemke**

[~TASK] FLOW3 (AOP): Removed the "Resource" sub package from the blacklisted sub packages because it now contains a class (Resource) which needs to be persistable.
[-FEATURE][!!!] FLOW3 (MVC): For now removed the request hash feature (HMAC) because it mocks a level of security for incoming data which it doesn't provide. The current mechanism effectively puts control over content security into Fluid templates and it doesn't belong there. Although there might be a need for a request hash, the content security must be implemented by other means. Relates to #4960 and relates to #5659.
[+FEATURE] FLOW3 (MVC): Implemented support for file uploads. Uploading files is cooperation between the Web Request Builder, the Property Mapper and the Resource sub package. The solution included in this commit provides handling of incoming files (including nested arguments) and transparent conversion into Resource objects. Resources (files) are only stored once, no matter how often they are uploaded or what original filename they carried. Still missing: view helper, documentation and automatic purging of unused resource files. Addresses #342.
[~API][!!!] FLOW3 (Property): Renamed the property mapper class to "PropertyMapper" (was just "Mapper" before). Relates to #5658
[+FEATURE] FLOW3 (Property): The Property Mapper now supports a mechanism called Object Converters. These convertes enable the mapper to convert strings, arrays or numbers to certain objects, for example a unix time stamp to a DateTime object. Resolves #5660.
[+FEATURE] FLOW3 (Reflection): Implemented the methods "isPropertySettable" and "isPropertyGettable" for the ObjectAccess class.
[~TASK] FLOW3 (Resource): Renamed the StreamWrapper class to StreamWrapperAdapter
[+FEATURE] FLOW3 (Resource): Implemented a ResourceObjectConverter which is capable of converting arrays or strings to Resource objects.
[+FEATURE] FLOW3 (Utility): Implemented a setValueByPath() method for the Array utilities class.
[+FEATURE] FLOW3 (Utility): Added support for the _FILES super global to the Environment class. The array of information about uploaded files can be obtained in a much cleaner way than PHP provides it by the new getUploadedFiles() method.

## History

**#1 - 2009-10-09 15:00 - Sebastian Kurfuerst**

*- Status changed from Accepted to Resolved*

*- % Done changed from 0 to 100*

Applied in changeset r3309.