TYPO3.Flow - Bug #37001

Catch Exception from inactivityTimeout

2012-05-09 14:41 - Carsten Bleicker

Status:	Resolved	**Start date:**	2012-05-09
Priority:	Must have	**Due date:**	
Assignee:	Karsten Dambekalns	**% Done:**	100%
Category:	Security	**Estimated time:**	0.00 hour
Target version:			
PHP Version:	5.3	**Complexity:**	
Has patch:	No		

Description

i played around a little bit with the session lifetime.
setting inactivityTimeout: 30
on inactivity for about 30 seconds flow3 throws the exception:
"#1258721059: The security context contained no tokens which could be authenticated"
Should this be catched and force a redirect to the default configured auth provider?

Related issues:

Related to TYPO3.Flow - Feature #39423: Custom Error Renderers	**Resolved**	**2012-12-25**
Related to TYPO3.Flow - Feature #29907: Redirect to /login instead of raising...	**Resolved**	**2011-09-16**
Has duplicate TYPO3.Flow - Feature #37243: Authentication after long time of ...	**Closed**	**2012-05-17**
Has duplicate TYPO3.Flow - Bug #40563: When session times out, Exception occu...	**Closed**	**2012-09-03**

Associated revisions

Revision 8d1c1372 - 2012-09-04 21:57 - Andreas Förthner

[BUGFIX] Fix request injection in security context

As inject* methods are excluded from the session
lazy loading proxy mechanism, the injectRequest
method of the security context didn't work correctly
in requests with a session timeout. Renaming the
method to setRequest solves the problem.

Change-Id: Iaf12a23097433697e86897f4517caea1ef2b0b7d
Related: #37001

Revision 1b409cc3 - 2012-09-05 16:57 - Robert Lemke

[BUGFIX] Prevent data from destroyed session from being loaded

This fixes a problem with session data which is being unserialized
even though its session is about to be destroyed. Because the
time of the last activity was stored in the session and needs to be
retrieved in order to determine if the session is still valid, also all
other session data is being unserialized by PHP. This results in
session scope objects registering themselves at the Object Manager
due to their code in __wakeup().

Now the session time out is not stored in the session anymore but
in its own cookie. Therefore the session data is only unserialized
if the session is still valid.

Change-Id: If7d004c7922c4c14e3713eae2f34d36a14d55b84
Resolves: #37001
Releases: 1.1, 1.2

Revision 86755da9 - 2012-09-10 11:48 - Andreas Förthner

[!!!][BUGFIX] Fix request injection in security context

As inject* methods are excluded from the session
lazy loading proxy mechanism, the injectRequest
method of the security context didn't work correctly

in requests with a session timeout. Renaming the
method to setRequest solves the problem.

In case you implemented your own RequestHandler, this is
a breaking change, and you need to call setRequest() now.

Change-Id: Iaf12a23097433697e86897f4517caea1ef2b0b7d
Related: #37001
Releases: 1.1, 1.2

Revision a4c094a7 - 2012-09-10 11:48 - Robert Lemke

[BUGFIX] Prevent data from destroyed session from being loaded

This fixes a problem with session data which is being unserialized
even though its session is about to be destroyed. Because the
time of the last activity was stored in the session and needs to be
retrieved in order to determine if the session is still valid, also all
other session data is being unserialized by PHP. This results in
session scope objects registering themselves at the Object Manager
due to their code in __wakeup().

Now the session time out is not stored in the session anymore but
in its own cookie. Therefore the session data is only unserialized
if the session is still valid.

Change-Id: If7d004c7922c4c14e3713eae2f34d36a14d55b84
Resolves: #37001
Releases: 1.1, 1.2

History

#1 - 2012-06-06 20:58 - Martin Brüggemann

- Assignee changed from Robert Lemke to Andreas Förthner

- Priority changed from Should have to Must have

- Target version set to 1.1 RC1

+1

#2 - 2012-06-25 22:13 - Karsten Dambekalns

- Target version changed from 1.1 RC1 to 1.1

#3 - 2012-06-26 11:29 - Karsten Dambekalns

- Assignee deleted (Andreas Förthner)

#4 - 2012-07-10 10:02 - Karsten Dambekalns

- Status changed from New to Needs Feedback

- Assignee set to Karsten Dambekalns

I just tried this, and at least in the context of Phoenix I get no exception when the session times out. Is this still an issue for you?

#5 - 2012-07-10 10:41 - Carsten Bleicker

Karsten Dambekalns wrote:

> I just tried this, and at least in the context of Phoenix I get no exception when the session times out. Is this still an issue for you?

I don't know. I am out of development with FLOW3 atm because of daily business, sorry.

#6 - 2012-07-10 15:42 - Karsten Dambekalns

- Target version deleted (1.1)

#7 - 2012-07-10 17:05 - Martin Brüggemann

This is still an issue for me with the default session timeout setting. Even in production context FLOW3 throws an exception, after a session has timed out. The only workaround that works is setting the inactivityTimeout to 0:

```
TYPO3:
  FLOW3:
    session:
      inactivityTimeout: 0
```

Here's an example exception:

```
Uncaught exception #1258721059 in line 160 of /MyPath/Data/Temporary/Development/Cache/Code/FLOW3_Object_Class
es/TYPO3_FLOW3_Security_Authentication_AuthenticationProviderManager.php: The security context contained no to
kens which could be authenticated.

27 TYPO3\FLOW3\Security\Authentication\AuthenticationProviderManager_Original::authenticate()
26 TYPO3\FLOW3\Security\Authentication\AuthenticationProviderManager::authenticate()
25 call_user_func_array(array, array)
24 TYPO3\FLOW3\Security\Authentication\AuthenticationProviderManager::FLOW3_Aop_Proxy_invokeJoinPoint(TYPO3\FL
OW3\Aop\JoinPoint)
23 TYPO3\FLOW3\Security\Authentication\AuthenticationProviderManager::authenticate()
22 TYPO3\FLOW3\Security\Authorization\Interceptor\PolicyEnforcement_Original::invoke()
21 TYPO3\FLOW3\Security\Aspect\PolicyEnforcementAspect_Original::enforcePolicy(TYPO3\FLOW3\Aop\JoinPoint)
20 TYPO3\FLOW3\Aop\Advice\AroundAdvice::invoke(TYPO3\FLOW3\Aop\JoinPoint)
19 TYPO3\FLOW3\Aop\Advice\AdviceChain::proceed(TYPO3\FLOW3\Aop\JoinPoint)
18 MyCompany\MyProject\Controller\Project\TicketController::__construct()
17 TYPO3\FLOW3\Object\ObjectManager::instantiateClass((MyCompany\MyProject\Controller\Project\TicketController
)MyCompany\MyProject\Controller\Project\TicketController, array)
16 TYPO3\FLOW3\Object\ObjectManager::get((MyCompany\MyProject\Controller\Project\TicketController)MyCompany\My
Project\Controller\Project\TicketController)
15 TYPO3\FLOW3\Mvc\Dispatcher_Original::resolveController(TYPO3\FLOW3\Mvc\ActionRequest)
14 TYPO3\FLOW3\Mvc\Dispatcher_Original::dispatch(TYPO3\FLOW3\Mvc\ActionRequest, TYPO3\FLOW3\Http\Response)
13 TYPO3\FLOW3\Mvc\Dispatcher::dispatch(TYPO3\FLOW3\Mvc\ActionRequest, TYPO3\FLOW3\Http\Response)
12 call_user_func_array(array, array)
11 TYPO3\FLOW3\Mvc\Dispatcher::FLOW3_Aop_Proxy_invokeJoinPoint(TYPO3\FLOW3\Aop\JoinPoint)
10 TYPO3\FLOW3\Aop\Advice\AdviceChain::proceed(TYPO3\FLOW3\Aop\JoinPoint)
9 TYPO3\FLOW3\Security\Aspect\RequestDispatchingAspect_Original::setAccessDeniedResponseHeader(TYPO3\FLOW3\Aop
\JoinPoint)
8 TYPO3\FLOW3\Aop\Advice\AroundAdvice::invoke(TYPO3\FLOW3\Aop\JoinPoint)
7 TYPO3\FLOW3\Aop\Advice\AdviceChain::proceed(TYPO3\FLOW3\Aop\JoinPoint)
6 TYPO3\FLOW3\Security\Aspect\RequestDispatchingAspect_Original::blockIllegalRequestsAndForwardToAuthenticatio
nEntryPoints(TYPO3\FLOW3\Aop\JoinPoint)
5 TYPO3\FLOW3\Aop\Advice\AroundAdvice::invoke(TYPO3\FLOW3\Aop\JoinPoint)
4 TYPO3\FLOW3\Aop\Advice\AdviceChain::proceed(TYPO3\FLOW3\Aop\JoinPoint)
3 TYPO3\FLOW3\Mvc\Dispatcher::dispatch(TYPO3\FLOW3\Mvc\ActionRequest, TYPO3\FLOW3\Http\Response)
2 TYPO3\FLOW3\Http\RequestHandler::handleRequest()
1 TYPO3\FLOW3\Core\Bootstrap::run()
```

#8 - 2012-07-31 17:49 - Bastian Waidelich

This is still a blocker.
The problem occurs in \TYPO3\FLOW3\Security\Authorization\Interceptor\PolicyEnforcement::invoke().
Adding a try/catch block here "solves" the issue for me:

```
public function invoke() {
    try {
        $this->authenticationManager->authenticate();
        $this->accessDecisionManager->decideOnJoinPoint($this->joinPoint);
    } catch (\TYPO3\FLOW3\Security\Exception $exception) {
        header('Location: http://foo/login');
        exit;
    }
}
```

Maybe #39423 could be a clean solution?

#9 - 2012-09-05 16:45 - Bastian Waidelich

- Status changed from Needs Feedback to Under Review

See https://review.typo3.org/#/c/14348/

#10 - 2012-09-05 16:58 - Gerrit Code Review

Patch set 1 for branch **master** has been pushed to the review server.
It is available at http://review.typo3.org/14379

#11 - 2012-09-05 18:22 - Gerrit Code Review

Patch set 1 for branch **FLOW3-1.1** has been pushed to the review server.
It is available at http://review.typo3.org/14383

#12 - 2012-09-05 18:35 - Robert Lemke

- Status changed from Under Review to Resolved

- % Done changed from 0 to 100

Applied in changeset 1b409cc3eeb853ee6a733e627752839f6050ce62.

#13 - 2012-09-05 18:59 - Gerrit Code Review

- Status changed from Resolved to Under Review

Patch set 2 for branch **FLOW3-1.1** has been pushed to the review server.
It is available at http://review.typo3.org/14383

#14 - 2012-09-10 11:49 - Gerrit Code Review

Patch set 3 for branch **FLOW3-1.1** has been pushed to the review server.
It is available at http://review.typo3.org/14383

#15 - 2012-09-10 12:35 - Robert Lemke

- Status changed from Under Review to Resolved

Applied in changeset a4c094a71e4df00f2128e8b14e80a03fa3e495b0.