EXT: Salted user password hashes

Extension Key: saltedpasswords
Language: en
Keywords: forAdmins, forDeveloper, forIntermediates
Copyright 2000-2009, TYPO3 Core Team, <marcus#exp2010@t3sec.info>

This document is published under the Open Content License
available from http://www.opencontent.org/opl.shtml

The content of this document is related to TYPO3
- a GNU/GPL CMS/Framework available from www.typo3.org

Table of Contents

Introduction

What does it do?

This extension allows to store a user's password in form of a salted hash. Together with this possibility it brings also a necessary authentication service to use such salted password hashes.

Screenshots

```
mysql> SELECT uid,username,password FROM fe_users ORDER BY uid DESC LIMIT 1;
+-----+----------+------------------------------------+
| uid | username | password                           |
+-----+----------+------------------------------------+
|  24 | magkes   | $1$Lg6X68Yn$45.oZbodkpCNJkH65Pqz60 |
+-----+----------+------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Image 1: Example salted user password hash of a FE user record

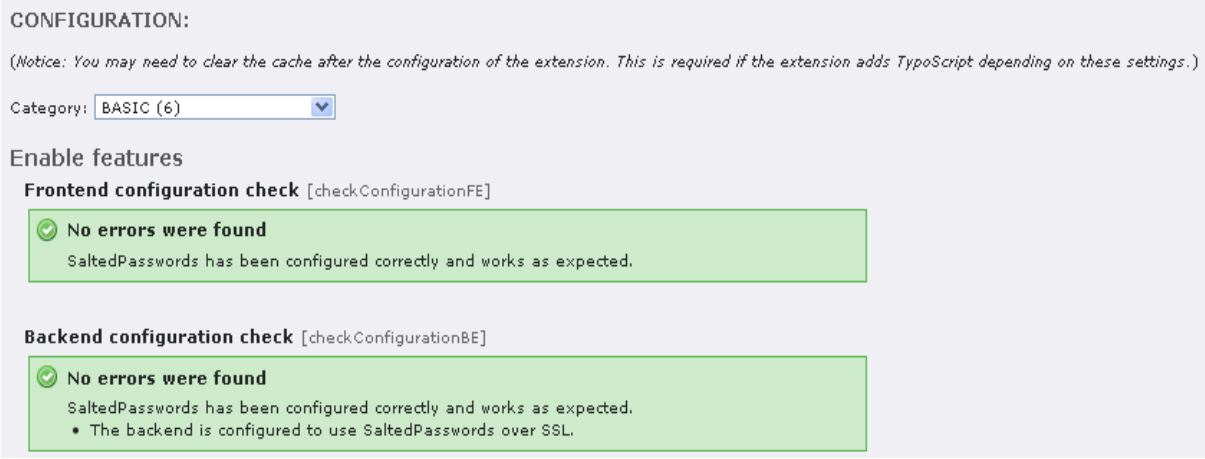

Image 2: Extension configuration check

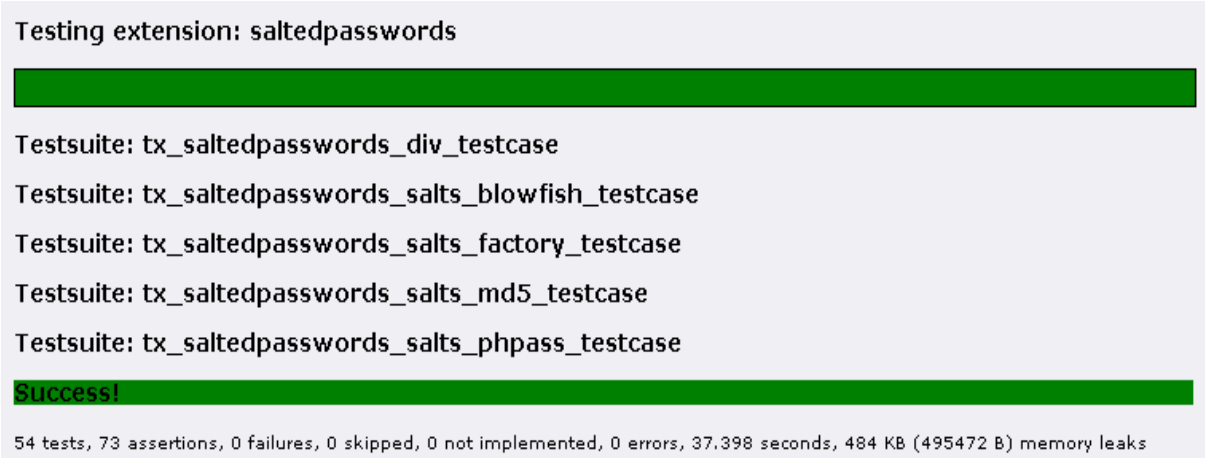

Image 3: Passed Unit Tests of extension code

Overview

Why you should use salted user password hashes?

By using this extension, you get rid of plain-text passwords or MD5 password hashes for user records in TYPO3. MD5 hashes are no longer safe to use for passwords. Using rainbow tables is widely spread these days and retrieving an according valid plain-text password is just a matter of time. With salted hashes, an attacker needs to create separate rainbow tables for each salt. The salt itself is different for each stored password hash. So retrieving plain-text passwords for all user records in a TYPO3 installation is quite expensive in terms of complexity.

You still are advised to use well-chosen passwords. Avoid wordlist entries; use arbitrary complex non-wordlist passwords/passphrases.

Supported types of hashing method

The extension provides several types of hashing method:

- **Portable PHP password hashing**
  This method allows to exchange salted hashes with other CMS like Drupal or Wordpress as they support phpass too. This is the format of previously generated salted user passwords by extension t3sec_saltedpw. This method is derived from a third-party library. Portable PHP password hashing method is available in any environment, TYPO3 4.3 will run with. It's the **default and recommended setting**.
- **MD5 salted hashing**
  This method allows to use Salted user password hashes for other server daemon authentications (mailserver,etc.) too. Use this setting if you need to authenticate other services against TYPO3 user records. This method uses PHP standard capabilities.
- **Blowfish salted hashing**
  This method provides increased security in comparison to MD5 salted hashing. Use this setting if you have higher requirements on password security. This requires a PHP > 5.3.0, PHP 5.X.X with suhosin patch applied or PHP compiled with a recent glibc. You might want to execute the Unit Tests brought together with this extension; if tests in blowfish testsuite fail, your server installation most probably does not support blowfish. Once you've chosen blowfish hashing, you need to make sure blowfish is available on the server you might move to in future. Otherwise, users won't be able to login any longer.

Suggested server environment

Due to the nature of salted user password hashes, the server needs to have a plain-text password to check against stored salted user password hashes of a database user record during authentication. This requires a transfer of the plain-text password from a user's browser to the TYPO3 server.

You obviously want to send the password over an encrypted channel. According possibilities are the usage of either SSL with your web server or TYPO3 system extension rsaauth.

Installation

Installing the extension

Install the extension in the Extension Manager!

As TYPO3 by default uses a superchallenged method for authentication but Salted user password hashes requires plain-text passwords on server side (outlined in previous chapter), you need to change that. So please open the **TYPO3 install tool**. Depending on which TYPO3 mode (FE or BE) you like to use salted user password hashes for, please change the according "**loginSecurityLevel**" configuration variable! A valid setting for working salted user password hashes are

- "**rsa**" (requires installed system extension rsaauth; sends password over an encrypted channel) or
- "**normal**" (send passwords in plain-text; works for TYPO3 frontend and backend; please use SSL to send password over an encrypted channel)

Now return back to the Extension manager and again open the extension configuration for system extension "Salted user password hashes"(saltedpasswords)!

Depending on which TYPO3 mode (FE or BE) you like to use salted user password hashes for, activate the according **checkbox** "**Enable FE**" or "**Enable BE**"!

Additionally, you might want to change the hashing method. See previous chapter for an explanation on when to use which method.

When done, please click the update button!

Checking the setup

The extension brings a smart configuration check with it. Success/warning/error messages will report the status of the configuration. In case of problems, it will show up advices which steps are wrong or missing.

A successful configuration will show up like following:

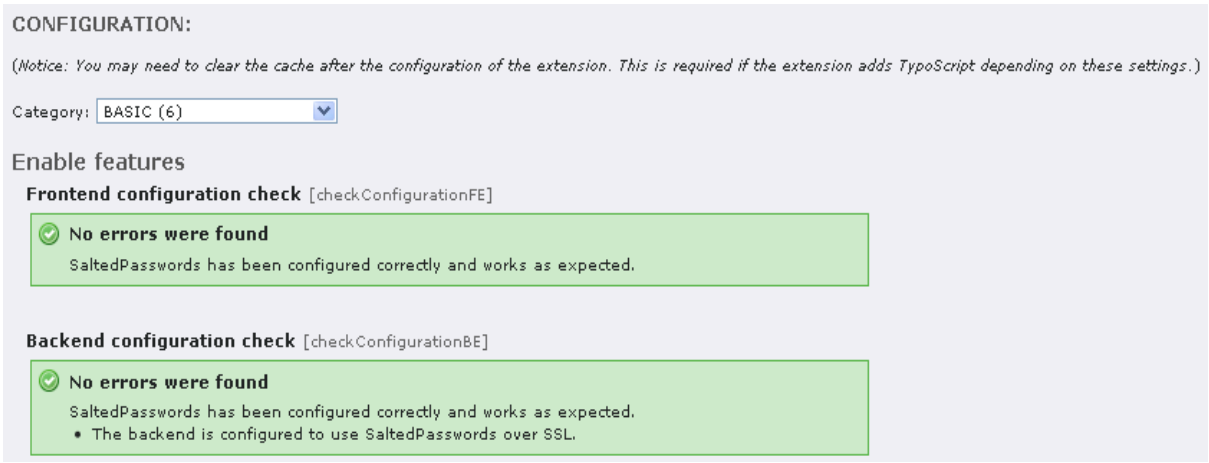

Image 4: Extension configuration check

Re-use of existing passwords when upgrading to TYPO3 Core 4.3

If you have previously used extension t3sec_saltedpw, you don't have to worry at all. The extension will gracefully accept these passwords.

If you have password records in form of plain-text passwords or MD5, the extension will use them too and update the records to salted user password hashes as long as you deactivate "**Exclusive Usage**" and activate "**Update user passwords**" (default settings, see extension configuration chapter in this manual for details)

Compatibility of other extensions with Salted user password hashes

All TYPO3 Core system extensions are compatible with Salted user password hashes. This means that authentication works (see installation chapter for requirements), login works (using system extension felogin) and creating/editing user records in the TYPO3 backend is working too.

Unfortunately, using Salted user password hashed together with other TER extensions might not work. E.g. widely used extensions sr_feuser_register or mm_forum are not capable of dealing with Salted user password hashes at the time of writing this manual. The Developer's guide chapter in this manual provides example implementations. We hope that this manual helps to spread the usage of Salted user password hashes.

If your favourite extension still does not support Salted user password hashes, please file a feature request for these projects and refer to this manual. We're sure that acceptance will increase in near future.

Advanced configuration

All extension configuration settings are divided into three categories: basic, advanced frontend, advanced backend
Please use the category drop-down box in the extension configuration to switch between these categories!

Basic configuration

```
# Enable FE (boolean)
FE.enabled = 1
```
Enables usage of salted user password records for the TYPO3 frontend

```
# Hashing method for the frontend (list)
FE.saltedPWHashingMethod = tx_saltedpasswords_salts_phpass (Portable PHP password hashing)
```
Defines hashing method to use for TYPO3 frontend.

```
# Enable BE (boolean)
BE.enabled = 1
```
Enables usage of salted user password records for the TYPO3 backend

```
# Hashing method for the backend (list)
BE.saltedPWHashingMethod = tx_saltedpasswords_salts_phpass (Portable PHP password hashing)
```
Defines hashing method to use for TYPO3 backend.

Advanced frontend configuration

```
# Force salted passwords (boolean)
FE.forceSalted = 0
```
Enforces usage of salted user password hashes only. Any other type of stored password will result in a failed authentication.

```
# Exclusive FE usage (boolean)
FE.onlyAuthService = 0
```
If enabled and authentication fails, no further authentication service will be tried.

```
# Update FE user passwords (boolean)
FE.updatePasswd = 1
```
Uses existing FE user passwords but automatically convert them to the salted hash format during authentication (will not work if forceSalted is enabled).

Advanced backend configuration

```
# Force salted passwords (boolean)
BE.forceSalted = 0
```
Enforces usage of salted user password hashes only. Any other type of stored password will result in a failed authentication.

```
# Exclusive BE usage (boolean)
BE.onlyAuthService = 0
```
If enabled and authentication fails, no further authentication service will be tried.

```
# Update BE user passwords (boolean)
BE.updatePasswd = 1
```
Uses existing BE user passwords but automatically convert them to the salted hash format during authentication (will not work if forceSalted is enabled).

Developer's Guide

The Salted user password hashes extension is written in an OOP style and thus makes it very easy to extend or use it in your TYPO3 extension.

Creating a new salted user password hash from a given plain-text password

Steps to be done:

- check if salted user password hashes is enabled for the desired TYPO3 mode (frontend/backend)
- let the factory deliver an instance of the default hashing class
- create the salted user password hash

Example implementation for TYPO3 frontend:

```
$password = 'XXX'; // plain-text password
$saltedPassword = '';

if (t3lib_extMgm::isLoaded('saltedpasswords')) {
	if (tx_saltedpasswords_div::isUsageEnabled('FE')) {
		$objSalt = tx_saltedpasswords_salts_factory::getSaltingInstance(NULL);
		if (is_object($objSalt)) {
			$saltedPassword = $objSalt->getHashedPassword($password);
		}
	}
}
```

Checking a plain-text password against a salted user password hash

Steps to be done:

- check if salted user password hashes is enabled for the desired TYPO3 mode (frontend/backend)
- let the factory deliver an instance of the according hashing class
- compare plain-text password with salted user password hash

Example implementation for TYPO3 backend (here the check for enabled salted user password hashed for a specific TYPO3 mode might be obmitted):

```
$password = 'XXX';  // plain-text password
$saltedPassword = 'YYY';  // salted user password hash
$success = FALSE; // keeps status if plain-text password matches given salted user password hash

if (t3lib_extMgm::isLoaded('saltedpasswords')) {
	if (tx_saltedpasswords_div::isUsageEnabled('BE')) {
		$objSalt = tx_saltedpasswords_salts_factory::getSaltingInstance($saltedPassword);
		if (is_object($objSalt)) {
			$success = $objSalt->checkPassword($password, $saltedPassword);
		}
	}
}
```

Adding a new salting method

If you decide to add an additional salting method, you can easily make such additional method available for this extension.

Steps to be done:

- create a new salting class that implements interface tx_saltedpasswords_salts and abstract class tx_saltedpasswords_abstract_salts (see class tx_saltedpasswords_salts_md5 for an example implementation)
- register you salting method class ($GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['ext/saltedpasswords']['saltMethods']) to make it available for the salt factory (see ext_localconf.php for an example)

Autoloading

The Salted user password hashes extension uses the newly introduced TYPO3 autoloader. So, when using files from this extension in your extension or whatever, you don't need to explicitly include/require such files.

Known problems

No problems so far.

In case of bugs, please file a bug report at the TYPO3 Core bugtracker!

To-Do list

No ideas or feature requests so far.

In case of missing features, please file a feature request at the TYPO3 Core bugtracker!

ChangeLog

Please check file ChangeLog for details!

Credits

The Salted user password hashes extension derives from TYPO3 extension t3sec_saltedpw, originally developed by Marcus Krause. This extension is still available in the TYPO3 extension repository (TER) for users of TYPO3 Core 4.2.X. For a listing of contributors to the original version, please have a look at the credits chapter of t3sec_saltedpw's extension manual.

A lot of clean up and optimization was done during core integration. A tremendous effort has been made by Steffen Ritter. Furthermore, Oliver Hader (TYPO3 Core Team), Sascha Kettler and Marcus Krause (TYPO3 Security Team) made a good amount of code contribution which allowed it to integrate the extension in TYPO3 Core.

Despite a rewrite of this extension, older passwords generated by extension t3sec_saltedpw are still compatible with this system extension.

Licenses

This extension is distributed under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or any later version.

The GNU General Public License can be found at *http://www.gnu.org/copyleft/gpl.html*. A copy is found in the TYPO3 Sources at *typo3/GPL.txt*.

The modified Portable PHP password hashing framework, included in this extension, is retrieved from Drupal CMS. Drupal CMS is distributed under the terms of the GNU General Public License version 2.

The original Portable PHP password hashing framework is distributed under Public Domain.

The extension icon is retrieved from Mini Set of Mark James, which can be found at *http://www.famfamfam.com/lab/icons/mini/*. This set is distributed under Creative Commons Attribution 2.5 License. The license can be found at http://creativecommons.org/licenses/by/2.5/.

Appendix A – List of figures