


Feature: support for Bcrypt password hashes #225

 Merged

ikedas merged 4 commits into [sympa-community:sympa-6.2](#) from [mpkut:feature_bcrypt_hash](#) on 12 Mar 2018


Conversation 14

Commits 4

Checks 0

Files changed 6

+191 -22

 mpkut commented on 7 Mar 2018 • edited

Contributor

...

Some Sympa sites may find it desirable to use other password hash functions than MD5. At our site we have been looking at Bcrypt, which has useful properties like supporting a random salt and scaling in expense as computing power becomes cheaper over time. Both features make it much more costly to crack a site's user passwords if they are revealed in some way.

Since Sympa has already modularized password hashing in `Sympa::User::password_fingerprint`, we found that the changes required for basic Bcrypt support are relatively small in number:

- add new `sympa.conf` options to select and control Bcrypt hashes:
 - `password_hash`: `md5` or `bcrypt` (default `md5`)
 - `bcrypt_cost`: integer value used to determine the "cost" parameter of the Bcrypt algorithm (default `12`)
- update `Sympa::User::password_fingerprint` in `Auth.pm`:
 - select hash functions from an array of callbacks based on `password_hash` setting
 - add `'salt'` parameter (ignored for `md5`, used as Bcrypt settings for `bcrypt`)
- update all calls to `Sympa::User::password_fingerprint` to supply the user's current password hash
- update `upgrade_sympa_password.pl` to recognize both MD5 and Bcrypt hashes
- increase the width of the `password_user` column from 40 to 64 columns in order to store Bcrypt hashes without truncation


Some additional notes:

- This implementation uses perl-Crypt-Eksblowfish, which is available in EPEL for CentOS/RHEL 7, which matches our site's chosen Linux distribution.
- The implementation only addresses authentication for Sympa users found in the `user_table`. Other Sympa features that use MD5 continue to use that hash exclusively.


We understand that it is not a trivial thing to suggest changes to the Sympa authentication infrastructure, and we definitely have some uncertainty about some of the choices here:

- are there better option names than `password_hash` etc?
- does making the `password_user` field wider present an issue?
- is perl-Crypt-Eksblowfish available on enough platforms to be a good choice?
- the extra argument to `password_fingerprint` seems awkward

If it would be more appropriate to pose this as an issue rather than a pull request, please let me know. Either way, we would like to improve this code to the point where it is acceptable inclusion in Sympa, so suggestions for improvement would be very welcome.

 Draft support for bcrypt password hashes

✓ b3182f1


 xavierba commented on 7 Mar 2018

Contributor

...

hiwix, the latest perl-Crypt-Eksblowfish (0.009, dated 27 April 2011) is old enough to be available available for RHEL/CentOS 6 and 7 from EPEL and for all supported Fedora releases. Debian 8 and 9 have Libcrypt-eksblowfish-perl 0.009, while the soon to be end-of-life Debian 7 has 0.008.


Thus I guess availability of the perl module will not be an issue for any distribution. And it shouldn't be a big deal to package it for any distribution shall it be missing.

 xavierba commented on 7 Mar 2018

Contributor

...

Quickly reading through the commit, I think it might be missing the part to change the database schema on upgrade.

 mpkut commented on 8 Mar 2018


Contributor

Author

...

Thanks for the review! Good news that it looks like perl-Crypt-Eksblowfish module may be usable.

As for the password column, I looked for a place to update the column in Upgrade.pm and didn't find an obvious place. I observed though that just setting the width to 64 in DatabaseDescription.pm led to Sympa automatically altering the column in the database when I restarted Sympa. It looks like Sympa::DatabaseManager::probe_db calls Sympa::DatabaseDriver::update_field as part of the probe sequence, which then issues the database-specific query to update the column.

 ikedas commented on 8 Mar 2018

Member

...

Great enhancement @mpkut. May I add this PR to 6.2.26 milestone?

- does making the `password_user` field wider present an issue?

I think no.

- is perl-Crypt-Eksblowfish available on enough platforms to be a good choice?

We should use more widely adopted module to avoid difference between implementations^[1]. Wandering CPAN, I felt this module looks popular enough: [cbl](#)


- the extra argument to `password_fingerprint` seems awkward

Salt is generated by default in `$fingerprint_hashes(bcrypt)` subroutine. I guess extra argument of `password_fingerprint()` is unnecessary.


Other points I noticed:

- Use of `2` (no null padding) is discouraged and supporting only `$2a$` seems sufficient. Is there any supposed use case of `2`?
- Crypt::Eksblowfish::Bcrypt would be added as optional module to Sympa::ModDef which is used by `sympa_wizard.pl` and `Task::Sympa`.

^[1] Some implementations mistakenly use only 56 octets (448 bits) at beginning of input. Crypt-Eksblowfish's bcrypt uses 72 octets as original algorithm does.

 Add Crypt::Eksblowfish to ModDef.pm for use by sympa_wizard.pl and Ta...

✓ 696298a

 mpkut commented on 9 Mar 2018 • edited

Contributor

Author

...

Great enhancement @mpkut. May I add this PR to 6.2.26 milestone?

Thank you for looking this over! If you think the code is close enough to being ready, that would be quite agreeable. :-)

Salt is generated by default in `$fingerprint_hashes(bcrypt)` subroutine. I guess extra argument of `password_fingerprint()` is unnecessary.

To back up a bit, Bcrypt uses the settings in the existing hash to generate an identical fingerprint from the same password. That means we need to pass the value of `$user->{ "password" }` to the Bcrypt hash callback when we are attempting to check a password.


It seemed least invasive to add a second `salt` parameter to `password_fingerprint` and let each hash function use or ignore it as needed. But some places in the calling code we don't have `$user->{ "password" }` ready at hand. For those I elected to supply the value `'$'`. That is at least explicit rather than just ignoring the fact that some algorithms may require a salt value. But as I said it feels a little awkward and hard to read. Perhaps `'undef'` instead of `'$'` would be a little clearer?

Use of `2` (no null padding) is discouraged and supporting only `$2a$` seems sufficient. Is there any supposed use case of `2`?

For the purposes of Sympa authentication it seems like either could be used as long as the usage was consistent. Given the discouragement of `2`, it doesn't seem like there is much of a case for supporting it as well.

Crypt-Eksblowfish::Bcrypt would be added as optional module to Sympa::ModDef which is used by `sympa_wizard.pl` and `Task::Sympa`.

Got it. Added to the PR branch. Thank you!

 ikedas commented on 9 Mar 2018 • edited

Member

...

Salt is generated by default in `$fingerprint_hashes(bcrypt)` subroutine. I guess extra argument of `password_fingerprint()` is unnecessary.

To back up a bit, Bcrypt uses the settings in the existing hash to generate an identical fingerprint from the same password. That means we need to pass the value of `$user->{ "password" }` to the Bcrypt hash callback when we are attempting to check a password.

It seemed least invasive to add a second `salt` parameter to `password_fingerprint` and let each hash function use or ignore it as needed. But some places in the calling code we don't have `$user->{ "password" }` ready at hand. For those I elected to supply the value `'$'`. That is at least explicit rather than just ignoring the fact that some algorithms may require a salt value. But as I said it feels a little awkward and hard to read. Perhaps `'undef'` instead of `'$'` would be a little clearer?


I got it. I missed another call of `password_fingerprint()`. I prefer to `undef` than `'$'`, or simply omitting optional argument.


Use of `2` (no null padding) is discouraged and supporting only `$2a$` seems sufficient. Is there any supposed use case of `2`?

For the purposes of Sympa authentication it seems like either could be used as long as the usage was consistent. Given the discouragement of `2`, it doesn't seem like there is much of a case for supporting it as well.


I see. Sympa itself will generate only `$2a$`, but importing from other systems can contain `2`.

On `password_hash`: How about gradually upgrading password? Idea is: password verification on both `md5` and `bcrypt` is supported, but when user updates their password, only `bcrypt` is used.

 ikedas added the enhancement label on 9 Mar 2018

 Use "undef" instead of empty string when calling Sympa::User::password...

✓ 3affa32

 mpkut commented on 10 Mar 2018

Contributor

Author


...

Ok, great. I have updated the calls to `password_fingerprint` to use `"undef"` where we don't have the user password string.


I had thought about gradual password upgrades as you suggest. The code that checks the user fingerprint during login already calls `update_global_user` and is in possession of the cleartext password, so that's really the only place for a password hash update on the fly.

A question: would this behavior be controlled by a separate config setting, or would it be implicit upon changing the `password_hash` setting? I could see some sites wanting to immediately invalidate old MD5 hashes after an algorithm change, while others might want to have a gradual transition.

In terms of code, it seems useful to add a function to `User.pm` to detect the type of hash used to generate an encrypted password string. That can be used in various other places to either preserve the existing hash type (say in `password_fingerprint`) or decide to upgrade it (in `Auth.pm`).

 Refactor to support multiple hashes at once and gradual upgrading of ...

✓ d7ca3ba

 mpkut commented on 10 Mar 2018 • edited

Contributor


Author

...

Here is a draft implementation of the gradual update idea.

- `ConfDef.pm`: new `sympa.conf` setting `password_hash_update` (default 1 = yes) to control whether password hashes are updated during login
- `User.pm`: new function `hash_type` to return 'md5', 'bcrypt' etc on inspection of a password hash
- `User.pm`: update `password_fingerprint` to use the same hash type as the input `salt` parameter, if provided
- `User.pm`: new function `update_password_hash` to check and update the hash. Controlled by current values of `password_hash` and `password_hash_update`. Hashes are only updated if the detected hash type differs from the current system setting.
- `Auth.pm`: call `Sympa::User::update_password_hash` immediately after confirming that the user has provided a correct password.

With these changes, the code allows me to log in with the same password string even as I toggle the setting `password_hash` between `md5` and `bcrypt`, thereby supporting gradual transitions to a different hash.

 ikedas commented on 10 Mar 2018

Member


...

@mpkut, thanks for quick response.

A question: would this behavior be controlled by a separate config setting, or would it be implicit upon changing the `password_hash` setting? I could see some sites wanting to immediately invalidate old MD5 hashes after an algorithm change, while others might want to have a gradual transition.

I supposed that if administrator decided switching to `bcrypt` at all, they may clear `password_user` field at once (in this case all users have to reset their password), and `password_hash` parameter is almost no use.

I'll review your code briefly, and if no critical problem found, I'll merge it in Monday morning UTC (in early morning CT), then it will be included in the next beta on 13th March.

 mpkut commented on 10 Mar 2018 • edited

Contributor

Author

...


I supposed that If administrator decided switching to `bcrypt` at all, they may clear `password_user` field at once (in this case all users have to reset their password), and `password_hash` parameter is almost no use.

Thank you @ikedas. I'm entirely in favor creating good default behavior and avoiding useless config file parameters. If old password hashes should **always** be transparently converted to the current hash algorithm, then yes, there is no need for an `update_password_hash` setting. Easy enough to remove.

One idea for a slightly more useful config file parameter might be a combined "accept and update old hashes" flag.

- If `@`, only the current hash algorithm is valid. All users must reset their passwords immediately. Perhaps `@` should be the default in the name of greater security.
- Sites that want a gradual transition to the new hash can set the value to `1`; in that case old hashes are accepted and updated and allow the gradual transition you suggested.
- At a later point in time, the service admin can set the parameter back to `@`, completing the transition and forcing password resets for any remaining users with old hashes.

Either way it seems like the current implementation is a bit off base. I will hold off on making any updates to this PR pending your input.

 ikedas commented on 12 Mar 2018 • edited


Member

...

@mpkut, thanks for suggestion.


For now I'll merge recent code and close this PR. We would be better to examine about migration scenario further.


Thanks again!

 ikedas merged commit c311ab5 into sympa-community:sympa-6.2 on 12 Mar 2018

View details

1 check passed

 ikedas added this to the 6.2.26 milestone on 12 Mar 2018


 mpkut commented on 13 Mar 2018


Contributor

Author

...

Thank you @ikedas! Yes, it will take some more thought about various migration scenarios to determine the best semantics and defaults. We really appreciate the consideration and acceptance of this contribution!


 mpkut deleted the mpkut:feature_bcrypt_hash branch on 14 Mar 2018

 ikedas commented on 19 Mar 2018

Member

...

Built-in authentication page was added to documentation site. Additions / corrections of contents are welcome.


 mpkut commented on 19 Mar 2018

Contributor

Author


...

Thanks, will review. Per my comments above, I will need to correct the behavior so old hashes are *only* honored if "password_hash_update" is true, which is not the case with the current code. Will submit a separate fix PR ASAP.

 ikedas mentioned this pull request on 20 Mar 2018

Password encryption: Dropping Crypt::CipherSaber #87

Closed

 © 2020 GitHub, Inc.

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

Sign up for free to join this conversation on GitHub. Already have an account? [Sign in to comment](#)