ChiliProject - Feature # 323: Salted user passwords

Status:	Ready for review	**Priority:**	Normal
Author:	Stephan Eckardt	**Category:**	User accounts
Created:	2011-04-08	**Assignee:**	
Updated:	2011-05-26	**Due date:**	
Remote issue URL:	http://www.redmine.org/issues/7410		
Affected version:			
Description:	Hashed passwords without a salt are a security risk. Redmine has a patch for it in "r4936":https://github.com/edavis10/redmine/commit/ce84bb1a0194d98b4db99e258cc0ada6b98e19b8. I applied this patch to ChiliProject and changed their scheme for hashing passwords. In Redmine hashed passwords are saved as @SHA1(salt+SHA1(cleartext_password))@ to allow migrating the hashed passwords in one single migration. I found it a cleaner scheme to save passwords as @SHA256(salt+cleartext_password)@. Nevertheless the code works with old unsalted passwords and also with Redmine's scheme. The salting occurs when the user successfully logs in. Also the used hashing algorithm is saved in the database to allow using stronger algorithms in the future when SHA256 is not considered secure enough anymore.		

History

2011-04-08 10:26 pm - Eric Davis

I think I have some of r4936 in my upstream merge plan (#288).

2011-04-09 04:23 pm - Stephan Eckardt

Are there any plans to use SHA2 instead of SHA1 as SHA1 is so insecure?

2011-04-16 11:50 pm - Eric Davis

Stephan Eckardt wrote:

> Are there any plans to use SHA2 instead of SHA1 as SHA1 is so insecure?

I don't have any plans. Is there an easy way to upgrade to SHA2?

I've also just merged in the salted passwords from Redmine into unstable (commit:9964c43) so if you want to rebase your patches we can review the differences.

2011-04-18 01:55 pm - Eric Thomas

Stephan, rather than implementing SHA scheme, have you considering looking into bcrypt? The latest version of rails has it "built in":https://github.com/rails/rails/blob/master/activemodel/lib/active_model/secure_password.rb and the sound advice of many people nowadays is to use bcrypt over other hash algorithms. The major difference with bcrypt is that it's slow and that is exactly the kind of quality one wants to prevent brute-forcing a password.

I think this sums it up quite nicely:

<pre>

Why is bcrypt such a huge win? Think of the problem from two perspectives: the server, and the attacker.

First, the server: you get tens of thousands of logins per hour, or tens per second. Compared to the database hits and page refreshes and IO, the password check is negligable. You donâ€™t care if password tests take twice as long, or even ten times as long, because password hashes arenâ€™t in the 80/20 hot spot.

Now the attacker. This is easy. The attacker cares a lot if password tests take twice as long. If one password test takes twice as long, the total password cracking time takes twice as long.

</pre>

* http://chargen.matasano.com/chargen/2007/9/7/enough-with-the-rainbow-tables-what-you-need-to-know-about-s.html

2011-04-18 02:23 pm - Eric Thomas

Also, bcrypt does it's own salting so that should make things less complicated.

2011-05-26 11:35 am - Ammler _

just be sure to have a solution which works cross over other applications like
"hgredmine":https://bitbucket.org/redmine/hgredmine/changeset/05937604af87 (python)