Redmine - Feature #24520

Use more secure hashing algorigthm

2016-12-02 14:27 - mohammad hasbini

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Accounts / authentication	**Estimated time:**	0.00 hour
Target version:			
Resolution:			

Description

Introduction

Currently the hashing algorithm used is: SHA1 [0].
I suggest to use a more secure (computationally expensive) algorithm to store the password. Some alternative algorithms to use:

- bcrypt with reasonable iteration count.
- scrypt.

Drawbacks

The only drawback I can think of is the migration of the database to use the new algorithm. I'm thinking about using this approach to fix this issue:

Let's call the new secure hashing algorithm: H.
- The salt will be kept in the database.
- Foreach user in the database, **replace** the hashed password: SHA1($salt.$plain_password) with H(SHA1($salt.$plain_password).
- The algorithm H(SHA1($salt.$plain_password) will be used from now when creating a new users/resetting a new password ...

Why is SHA1 insecure ?

When I say *insecure* I'm not talking about the collision ratio. I'm referencing that it's easy (fast) to compute.
Example: Using hashcat[1] v3.10 with GPU: `R9 290X (+10Mhz) - AMDGPU-pro 16.40`[2], It's able to compute:
- 4,102,360,845 sha1 hash per second.
- 94,960 scrypt hash per second.
- 12,070 bcrypt hash per second (cost of 10 iirc).

Thoughts ?

[0] https://github.com/redmine/redmine/blob/master/app/models/user.rb#L840
[1] https://hashcat.net/
[2] https://docs.google.com/spreadsheets/d/1B1S_t1Z0KsqByH3pNkYUM-RCFMu860nlfSsYEqOoqco/edit#gid=1591672380

Related issues:

Related to Redmine - Feature #36056: Update password hash function	**New**

History

#1 - 2022-02-22 16:38 - Vincent Robert

- Related to Feature #36056: Update password hash function added