redmineorg-copy202205 - Vote #78280

Re-use existing identical disk files for new attachments

2022/05/09 18:13 - Admin Redmine

ステータス:	Closed	**開始日:**	2022/05/09
優先度:	通常	**期日:**	
担当者:		**進捗率:**	0%
カテゴリ:	Attachments_19	**予定工数:**	0.00時間
対象バージョン:	3.4.0_119	**作業時間:**	0.00時間
Redmineorg_URL:	https://www.redmine.org/issues/25215	**status_id:**	5
category_id:	19	**tracker_id:**	3
version_id:	119	**plus1:**	0
issue_org_id:	25215	**affected_version:**	
author_id:	40856	**closed_on:**	
assigned_to_id:	1	**affected_version_id:**	
comments:	21		

説明

Uploaded files can already be associated with multiple attachment records through @Attachment#copy@. This patch adds an @after_create@ hook to change the @disk_filename@ and @disk_directory@ attributes of a newly created attachment to point to an already existing, identical (filesize and digest) diskfile if one exists.

Database locks are used to guard against the deletion of the older attachment while the reference of the new attachment is changed.

Usefulness of this feature (i.e. how much space is saved) will vary a lot depending on external circumstances of course (one large scale setup we maintain at "Planio":https://plan.io/redmine-hosting saved around 15% / 60GB). Since the possiblity to have 1:n relationships of disk files to attachments already exists it just seems logical to make use of it for new attachments as well.

journals

IMO, MD5 is too weak for this purpose and this could lead to potential vulnerabilities. The first that comes to my mind: attacker generates a malicious file and a legitimate file with the same MD5, he first uploads the malicious file then send the legitimate one to a user X who will eventually upload it => people downloading the later from user X will actually get the malicious file.
We should implement this after upgrading the digest to a safer hash function.

Please let me know what you think.

Good point. Since we're also comparing file sizes, an attacker would have to create a collision while also matching the file sizes. Still not impossible I guess. Changing the hash function would make malicious collision creation harder, but still not impossible theoretically.

Maybe adding a real byte by byte comparison for the case of matching filesize / md5 would be the better way? Uploads take a while any way so the added computation time might not weigh in too much.

What do you think about migrating to SHA-1? It is as fast as MD5 and much safer.

One problem is that migrating a existing Redmine instance may take a long time if it stores many large files.

Jens Krämer wrote:

> Good point. Since we're also comparing file sizes, an attacker would have to create a collision while also matching the file sizes. Still not impossible I guess.

MD5 collission with the same input size can be created in seconds with a standard computer. The file size comparaison does not make it safer.

> Changing the hash function would make malicious collision creation harder, but still not impossible theoretically.

Not just harder, but practically impossible. Unlike MD5, there's no known way to easily generate SHA256 collisions.

> Maybe adding a real byte by byte comparison for the case of matching filesize / md5 would be the better way? Uploads take a while any way so the added computation time might not weigh in too much.

That would be the safer option indeed. It also guarantees that the original file is not broken/missing, which is a good thing IMO before discarding the uploaded file. Replacing the MD5 with a safer hash function does make sense anyway.

Go MAEDA wrote:

> What do you think about migrating to SHA-1? It is as fast as MD5 and much safer.

SHA-1? "No":https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html. Maybe SHA256 or SHA512 instead.

> One problem is that migrating a existing Redmine instance may take a long time if it stores many large files.

Yes, this should not be done during a db migration for this reason. We can use a new hash function for new files and provide a task that updates the existing hashes.

I gave the upgrade to SHA256 a try in #25240. Independent of that I'll add the bytewise comparison to this feature here next.

Additional patch adding a @FileUtils.identical?@ check to compare actual file contents before removing the duplicate.

OK.

Patches are committed, thanks. I think it's safer to delete the duplicate file +after+ the change is committed to the DB (r16460).

A fix for this patch was submitted as #25590.

Fix committed.

---

related_issues

relates,New,19289,Exclude attachments from incoming emails based on file content or file hash
relates,Closed,23510,Reuse an exist attachment
duplicates,Closed,15257,Attachment deduplication
blocks,Closed,25240,Use SHA256 for attachment digest computation
blocks,Closed,25590,prevent deadlocks in attachment deduplication

履歴

#1 - 2022/05/10 17:06 - Admin Redmine

- カテゴリ を Attachments_19 にセット

- 対象バージョン を 3.4.0_119 にセット