2 Improper integrity protection of server-side encryption keys

Share:

TIMELINE

vahe submitted a report to Nextcloud. Nov 8th (3 years ago)

The public keys used for the server-side encryption are not integrity-protected. These can easily replaced by anyone who has access to the data-at-rest data (even when the per-user-keys are enabled, as described in https://nextcloud.com/security/threat-model/). This holds true for all key types - from the master key, the per-user-keys as well as for the (optional) recovery key.

Attack scenarios may look like this:

- A recovery key is set by the admin and users enabled the recovery feature (or it was mandatorily set by adding the corresponding configuration to the `oc_preferences` table for all users). As it is unlikely that the recovery key is used very often, a person that has access to the data directory (even if at rest) is able to replace the public recovery key with a newly generated one. Each newly added file and each modified file will also be encrypted for the newly generated recovery key.
- A per-user-key encryption scheme is introduced and organizational shared folders are set up. To better handle access management all organizational shared folders are owned by the admin that also handles access management requests. As it is unlikely that the admin account will be used to access the individual files, a person that has access to the data directory (even if at rest) is able to replace the public key of the admin account with a newly generated one. Each newly added file and each modified file that is put into any of the organizational shared folders will also be encrypted for the newly generated admin account key.

The mentioned attack scenarios may also be executed by an external storage provider if the folder that is used as the data directory is stored at this external storage provider. Administrative access to the actual Nextcloud server is **not** necessary to mount this attack.

Impact

After mounting the attack the person would be able to read and modify all newly created files as well as files that have been modified since the attack was launched.

Preventing this attack could look as follows:

- The fingerprints of public and private key files that have been generated by the application could be stored in the database.
- Whenever public or private key files are read from disk the fingerprint of that file is checked against the value stored in the database.

An alternative approach for **preventing** this attack could look as follows:

- For each public and private key file that has been generated by the application a MAC could be calculated.
- The key for the MAC could be derived from the instance id, the instance secret and the relative file name of the corresponding key file.
- The MAC could be stored directly in the corresponding key file.
- Whenever a public or private key file is read from disk the MAC of that file could be calculated and compared with the MAC stored in the file.

1 attachment:
F629444: nextcloud_poc2.mp4

nc-bot: posted a comment. Nov 8th (3 years ago)

Thanks a lot for reporting this potential issue back to us!

Our security team will take a look at this issue as soon as possible. We will reply to your report within 72 hours, usually much faster. For obvious reasons we'd like to ask you to not disclose this issue to any other party.

nickvergessen Nextcloud staff posted a comment. Nov 11th (3 years ago)

Administrative access to the actual Nextcloud server is not necessary to mount this attack.

First of all, having access to the file system where Nextcloud runs is actually "having administrative access", as you can simply add a user which is part of the admin group.

> The mentioned attack scenarios may also be executed by an external storage provider if the folder that is used as the data directory is stored at this external storage provider.

Our idea for this problem is to allow to change the location of the key storage.
Check https://docs.nextcloud.com/server/17/admin_manual/configuration_files/encryption_configuration.html#occ-encryption-commands

> **Code** 96 Bytes — Wrap lines Copy Download

```
occ encryption:show-key-storage-root
Current key storage root:  default storage location (data/)
```

> Move keys to a different folder, either locally or on a different server. The folder must already exist, be owned by root and your HTTP group, and be restricted to root and your HTTP group. Further the folder needs to be located somewhere in your Nextcloud data folder, either physically, or as a mount. This example is for Ubuntu Linux. Note that the new folder is relative to your occ directory:

> **Code** 227 Bytes — Wrap lines Copy Download

```
cd /your/nextcloud/data
mkdir keys
chown -R root:www-data keys
chmod -R 0770 keys
occ encryption:change-key-storage-root keys
Start to move keys:
    4 [============================]
Key storage root successfully changed to keys
```

yahe posted a comment. Nov 15th (3 years ago)

Hi, I know had time to look at this again. Your documentation says:

> Further the folder needs to be located somewhere in your Nextcloud data folder, either physically, or as a mount.

So your idea for this problem wouldn't work in cases where the whole data directory is stored at an external party as - according to the documentation - the keys have to reside in the data directory.

rullzer posted a comment. Nov 25th (3 years ago)

I like the idea of storing the fingerprints in the database and comparing them.
I'll see what I can do. But it might take a while as the 18 wrap up is consuming a lot of time.

rullzer changed the status to Triaged. Nov 25th (3 years ago)

yahe posted a comment. Jan 21st (3 years ago)

Hi, I guess that now that Nextcloud 18 has been published there will be the time to look into the issues of the server-side encryption? My plan is to to submit a talk about the Nextcloud server-side encryption to the upcoming Gulaschprogrammiernacht (May 21st to May 24th). This should be enough time to fix the issues.

yahe posted a comment. May 27th (3 years ago)

Hello, this issue hasn't seen any update for 4 months. We approached the end of May without a fix. Do you still intend to work on this problem?

rullzer posted a comment. May 29th (3 years ago)

Hi,

Yes this is still on our list but right now we don't have a good way to solve it.

Cheers,
--Roeland

rullzer posted a comment. Aug 11th (2 years ago)

Hi,

So I'm back form my holiday and am continuing on https://github.com/nextcloud/server/pull/21529

Since keys can be moved for files etc. Linking the data in the database is tricky. However in this new approach the keys are encrypted with the system secret. And hence the attacker can't modify them.

Would you have the possibility to check out that PR?

Cheers,
--Roeland

yahe posted a comment. Aug 11th (2 years ago)

I think that the solution goes in the right direction, but I see a certain flow: If I understand it correctly, it is still possible to replace the public key of one user with the public key of another user. So the given solution improves on the currently existing problem but does not completely fix it.

Unfortunately, you currently have no possibility to have integrity-protected metadata for public key files. If you switched the process around to: `encrypt(json_encode($data))` you could have integrity-protected metadata. Then, a possibility to fix this problem completely would be to not only introduce the `$data["key"]` value but also a `$data["user"]` value containing the name of the associated user. This could be checked on usage to make sure that the key actually belongs to the correct user. Thanks to the EtM encryption taking place at the end, the key as well as the metadata would be integrity-protected.

rullzer posted a comment. Aug 11th (2 years ago)

Hi,

Thanks for the quick analysis.
Right turning it around would indeed make more sense. Let me do that.

I need to think about your comment. I think you are right and we can indeed store the user key with the uid.
In any case with this in place the attack scopes limits significantly in the sense that the attacker now must have an account on the system and access to the external storage. However lets fix it properly now that we are touching the code in any case.

I need a break now since it is way to hot here. I'll get back to you tomorrow.

Cheers,
--Roeland

rullzer posted a comment. Aug 12th (2 years ago)

Hi,

I updated the PR. Now it also stores the UID in the userkeys (and null in the systemkeys). This is then all encrypted.
It needs some cleanup but I think this now goes in the right direction.

Cheers,
--Roeland

yahe posted a comment. Aug 12th (2 years ago)

Hi, I had a short look at the current version in the PR. IMHO this is coming together quite nicely.

I am, however, a bit confused with the additional encodings going on. Of course, you may have certain reasons to double- and tripple-encode data, but I just wanted to mention what I see. I stumbled over this kind of (in my opinion) unnecessary file increases before with the 35% size-increase of the encrypted content files.:

- What I don't understand is why you Base64-encode the encrypted content again before storing it in a file here only to Base64-decode it here when you read the file back in. I really don't see the benefit of this additional Base64-encoding. `file_get_contents()` and `file_put_contents()` are perfectly able to handle binary data without precautionary encoding.

But, as I said, this is more of an observation rather than a security issue.

ullzer posted a comment. Aug 12th (2 years ago)

Hi,

No you are of course right. It is a left over I think from the first step where I encrypted only the key field (so that the json encode would not puke).

This can go indeed.
I'll tackle that in a bit.

Cheers,
--Roeland

ullzer posted a comment. Aug 13th (2 years ago)

Hi,

Ok we went over the code here again and some cleanup was done.
Now we are in the process of doing lot of tests but all seems good. I'm positive this can land in thenext release.

Cheers,
--Roeland

yahe posted a comment. Aug 13th (2 years ago)

Awesome to hear that. :)

ullzer posted a comment. Aug 24th (2 years ago)

Hi,

Ok this got merged into 20.
It is already in the beta1.

Cheers,
--Roeland

yahe posted a comment. Oct 5th (2 years ago)

Hi, I've seen that you have announced Nextcloud 20 that contains the fix for this issue. Will there be a security advisory for this issue?

nickvergessen Nextcloud staff closed the report and changed the status to **Resolved**. Oct 6th (2 years ago)

Thanks a lot for your report again. This has been resolved in our latest maintenance releases and we're working on the advisories at the moment.

Please let us know how you'd like to be credited in our official advisory. We require the following information:

- Name / Pseudonym
- Email address (optional)
- Website (optional)
- Company (optional)

nickvergessen Nextcloud staff posted a comment. Oct 6th (2 years ago)

Disclosure planned for 1st November

https://nextcloud.com/security/advisory/?id=NC-SA-2020-041
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-8159

nickvergessen Nextcloud staff posted a comment. Oct 6th (2 years ago)

@yahe do you still not want bounties?

yahe posted a comment. Oct 6th (2 years ago)

As said, merch would be welcome. :)

yahe posted a comment. Oct 6th (2 years ago)

Concerning the crediting in the advisory:

Name: Kevin "Kenny" Niehage
E-Mail: kenny@syseleven.de
Website: https://www.syseleven.de/
Company: SysEleven GmbH

yahe posted a comment. Oct 7th (2 years ago)

CVE-2020-8159 is already taken. Maybe that's just a typo.

nickvergessen Nextcloud staff posted a comment. Oct 7th (2 years ago)

Right, it's https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-8259

yahe posted a comment. Nov 5th (2 years ago)

ickvergessen Nextcloud staff agreed to disclose this report. Nov 13th (2 years ago)

SA published, CVE should autopublish when this is disclosed.

This report has been disclosed. Nov 13th (2 years ago)

extcloud has decided that this report is not eligible for a bounty. Jan 20th (2 years ago)

Setting to "ineligible" as you didn't want a bounty, but this closes the tickets cycle in the H1 workflow.