Lustre Client Encryption

05/2021
sbuisson@whamcloud.com
Lustre Client Encryption

- What is encryption for Lustre and solution retained: fscrypt

- Features available with new Lustre 2.14: content encryption

- Upcoming encryption features
  - Performance optimizations
  - Name encryption
  - Compatibility with future releases
What is encryption for Lustre?

► Use case:
  • Provide special directory for each user, to safely store sensitive files

► Goals:
  • Protect files in transit between clients and servers
  • Protect files at rest

► Solution retained
  • Conform to fscrypt kernel API
    ◦ Current users are ext4, F2FS, and UBIFS
    ◦ Core principle: pages in the page cache always contain clear text data
  • Make use of fscrypt userspace tool
Lustre Client Encryption in new 2.14

► Ability to encrypt file content
  • Encrypt on write, decrypt on read

► Ability to set encryption policies on directories
  • Support new IOCTLS from fscrypt userspace tool
  • Handle encryption context atomically
Lustre Client Encryption in new 2.14

- Encryption support built by default, via embedded *llcrypt* library (via libcfs)
  - Copied from Linux v5.4 fscrypt
  - Needed to support ‘content encryption only’ mode
  - Distributions supported (client side):
    - CentOS/RHEL 8.1 and later;
    - Ubuntu 18.04 and later;
    - SLES 15 SP2 and later.

- Encryption modes supported:
  - AES-256-XTS for contents and *null* for filenames
  - AES-128-CBC for contents and *null* for filenames

- Full details in LOM Chapter 30.5 ‘Encrypting files and directories’
Lustre Client Encryption – new ioctlts for policies

- fscrypt userspace tool
  - Works with Lustre out of the box, thanks to fscrypt API support
  - Associates protectors (passphrase, raw key, pam) to policies

```
# fscrypt setup /mnt/lustre
$ fscrypt encrypt /mnt/lustre/vault
$ fscrypt lock /mnt/lustre/vault
$ fscrypt unlock /mnt/lustre/vault
$ fscrypt metadata change-passphrase
    --protector=/mnt/lustre:7626382168311a9d
$ fscrypt metadata add-protector-to-policy
    --protector=/mnt/lustre:2c75f519b9c9959d
    --policy=/mnt/lustre:16382f282d7b29ee
```
Lustre Client Encryption – new ioctlts for policies

▶ fscrypt userspace tool

```
$ fscrypt metadata add-protector-to-policy
   --protector=/mnt/lustre:2c75f519b9c9959d
   --policy=/mnt/lustre:16382f282d7b29ee
```

▶ Ability to use ‘secondary protectors’, useful for:
  • different users sharing same encrypted directory
  • access via batch scheduler, backup tool, etc.
  ◦ access without key is **impossible**, even to cipher text data!
Lustre Client Encryption – bandwidth performance

► Initial benchmarks
  - 30-35% drop in sequential write, 20-22% drop in sequential read
  - Can we do something about it?

► Testbed
  - Client
    - Skylake 48 cores, 8160 CPU @ 2.10GHz
    - 96 GB RAM
    - ConnectX-4 Infiniband adapter, EDR network
  - Storage
    - 16 x NVMe
    - 16 OSTs

► Methodology
  - IOR, file per process, sequential IO, dummy encryption mode (AES-256-XTS)
Lustre Client Encryption – performance investigations

Bandwidth performance - Write
IOR - 32 threads - sequential

IO size

MIB/s

4k 16k 256k 1M 4M 16M

no encryption nobounce voidencrypt voidencrypt full encryption
Lustre Client Encryption – performance investigations

► **Compare** nobounce and voidencrypt
  - nobounce: encryption but no bounce page allocation: 10% drop
  - voidencrypt: no encryption but bounce page allocation: 30% drop

⇒ bounce page allocation hurts

► **Possible optimization path**
  - Leverage Lustre’s enc_pool mechanism
    - Take bounce pages from this pool
    - Do not allocate bounce page for every call to encryption primitive
Lustre Client Encryption – name encryption preview

➤ LU-13717: add name encryption
  • 6 patches pushed so far, undergoing review

➤ Wire up llcrypt API in llite to encrypt/decrypt names

➤ Convert between plain text and cipher text names
  • From plain to cipher before sending request to MDT
  • From cipher to plain upon reply
  • 2 cases to support
    o Access with the key: present actual names
    o Access without the key: base64 encoding of cipher text names
Lustre Client Encryption – name encryption challenges

- ‘name’ is no longer a valid path name, not even a well-formed string
  - Binary ciphertext names just cannot be encoded (base64 or similar)
  - Hopefully, ldiskfs and ZFS backend file systems *should* be able to handle binary names
    - Client: encode binary names and send to server side
    - Server: decode names in OSD layer, just before handing over to backend FS
      - Use custom encoding, to limit overhead to strictly necessary

- LFSCK
- Metadata performances
Lustre Client Encryption – releases compatibility

► Compatibility with future versions
  • Lustre 2.14 has content encryption only
  • Future versions will add name encryption
  • But in-kernel fscrypt cannot handle \textit{null} encryption for names

► When upgrading from 2.14
  • If need to keep existing encrypted directories
    o must stick with embedded \texttt{llcrypt}
    o but urge to move encrypted dirs to new ones, to get name encryption
  • Else
    o Can directly make use of in-kernel fscrypt
Lustre Client Encryption

- Projected roadmap
  - Content encryption
  - fscrypt inclusion
  - Encryption policies support
  - Name encryption
  - Performance optimizations

  - available in 2.14
  - target 2.15
Thank you!
sbuisson@whamcloud.com