Encrypted Local & iSCSI Storage with ZFS

OpenSolaris ZFS Crypto

ZFS Elevator Pitch

"To create a reliable storage system from inherently unreliable components"

- Data Integrity
  - Historically considered "too expensive"
  - Turns out, no it isn't
  - Real world evidence shows silent corruption a reality
  - Alternative is unacceptable

ZFS Elevator Pitch

- Ease of Use
  - Combined filesystem and volume management
  - Underlying storage managed as Pools which simply admin
  - Two commands: zpool & zfs
    - zpool: manage storage pool (aka volume management)
    - zfs: manage filesystems

ZFS Terminology

- Reminder if you did the ZFS workshop earlier in the week:
- XXX

Back to the Elevator Pitch

- Lets add just one word:

"To create a secured reliable storage system from inherently unreliable components"

High Level Requirements

- Support software only solution
  - Including single disk laptop use case
- SPARC, Intel, AMD64
  - Anything that OpenSolaris runs on and that ZFS has already been ported to
- Support keys & cryptographic operations in hardware
- Local key management:
  - HSM, TPM, smart card, passphrase
- Remote/Centralised key management

High Level Requirements

- Don't break Copy-On-Write semantics
- Integrate with existing ZFS admin model
  - CLI & GUI
- Support existing ZFS pools
- Delegation of key management to users & virtualized & Multi Level (MLS) environments
  - ability to create encrypted datasets
  - Including separation of key use vs key change

ZFS Encryption

- Set encryption policy at the ZFS data set
  - Most systems have only one or two pools but many (10s, 100s, 1000s,) datasets
  - AES-128 and AES-256 only initially but designed to be extensible (through minor code changes).
- Encrypted iSCSI targets via ZVOLs
- Encrypted datasets CAN be shared using NAS:
  - NFSv2,v3,v4 & CIFS (SMB)
  - No key management for NAS clients

Key Management

- Three types of key scope: set per dataset (and inherited)
  - Pool – user/admin manages one key for all encrypted datasets
  - Dataset - user/admin manages a per dataset key
  - Pool + Dataset
    - Both pool & dataset key required
- Keys as passphrase, in file, or HSM/Smartcard

Key Management

- Data set encryption property set at create time
  - Actual encryption key is randomly generated
    - wrapped by user/admin provided key (pool / dataset)
  - Avoids encrypt later problem
  - Avoids old clear text due to COW
  - Encryption cannot be enabled or changed later for existing dataset
  - Forces use of SHA256 for checksum

Key Change

- Key change supported
  - Doesn't actually re-encrypt data
  - Changes wrapping key.
- Key Change is online
  - Datasets must be mounted – or at least key available
  - Datasets stay mounted/shared during key change

What is encrypted ?

Yes

- All “application” data
- POSIX layer data
  - Permissions, owner etc
- Directory structure
- All ZVOL data
- All the above in a snapshot
- All the above in a clone

No

- Pool metadata
  - Disks, mount time, raid config, etc.

Deployment Issues

- Data set names
- Data set properties

Current Restrictions

- Initially can't boot from encrypted dataset
  - /var/tmp could be a separate file system
  - /tmp is backed by swap
- No support initially for encrypted crash dump devices
  - But Swap on an encrypted ZVOL is supported

ZFS Encryption Support Availability

- OpenSolaris project
- Targeting delivery to OpenSolaris in 2008
  - http://opensolaris.org/os/project/zfs-crypto/