Der sichere und effiziente Umgang mit großen und größten Datenmengen gehört heute sicherlich zu den herausragenden Aufgaben eines jeden IT-Infrastruktur-Dienstleisters – egal, ob betriebsintern oder für den freien Markt. Dabei stehen besonders die Datenintegrität, die technischen und organisatorischen Maßnahmen, die diese gewährleisten sollen, und natürlich der Sicherheitsaspekt beim Zugriff auf die gespeicherten Daten im Vordergrund.

ZFS-Verschlüsselung und andere Neuigkeiten in Solaris 11

Thomas Nau, Universität Ulm – kiz

ZFS hat mit seinem Erscheinen in Solaris 10 im Bereich der Datenintegrität nicht nur neue Maßstäbe gesetzt, sondern sich in den vergangenen Jahren zu einem Maßstab für Filesysteme schlechthin entwickelt. Alle anderen seither entstandenen und auch zukünftigen Entwicklungen müssen sich daran messen lassen. Die Dynamik der ZFS-Weiterentwicklung ist jedoch ungebrochen. Mit Solaris 11 stehen neben Performance-Verbesserungen auch neue Fähigkeiten zur Verfügung. So besteht jetzt die Möglichkeit, ZFS-Filesysteme und Volumes durch ZFS-bereitgestellte Block-Devices transparent zu verschlüsseln und diese Datenbereiche dann beispielsweise für virtualisierte Systeme in Zonen zu nutzen. Ebenfalls in Solaris 11 adressiert und vereinfacht wurde die Datenmigration von UFS hin zu ZFS.

Grundlegende ZFS-Design-Kriterien

Für den Betrieb einer universitären, zentralen Infrastruktur sind File- und Betriebssysteme notwendig, die weitreichende Vorkehrungen zum Schutz der Datenintegrität sowie der Datensicherheit bieten und damit vor Datenverlust schützen. Bei der Entwicklung von ZFS wurden von Beginn an insbesondere auch die designbedingten Schwachstellen herkömmlicher Filesysteme korrigiert:

- Gültige Daten werden niemals überschrieben (copy-on-write, COW).
- Starke Prüfsummen wie „fletcher4“ oder „sha256“ ermöglichen es, Fehler auf dem gesamten Datenpfad zu erkennen und diese, bei entsprechender redundanter Auslegung der Platten-Systeme, auch zu korrigieren. Hervorzuheben ist hierbei, dass die Prüfsummen getrennt von den eigentlichen Daten im sogenannten „Pointer-Bereich“ abgelegt sind. Aufgrund dieser Anordnung lassen sich die Auswirkungen durch den Ausfall einzelner Plattensektoren reduzieren.
- Die besonders wichtigen Metadaten des Systems, also diejenigen, die die Strukturen von Filesystemen und Pools beschreiben, werden mehrfach und von der übergeordneten Redundanz unabhängig in sogenannten „Ditto-Blocks“ gespeichert. Auf Wunsch lässt sich dieses Prinzip auch auf die Nutzerdaten erweitern, indem die entsprechenden Properties für das ZFS-Filesystem gesetzt werden.

ZFS bietet darüber hinaus eine Vielzahl betrieblicher Vorteile, etwa Snapshots und Slones. Diese kommen vermehrt als Ergänzung beziehungsweise als Ersatz üblicher Backup-Lösungen zum Einsatz. Nur so sind mit vergleichsweise geringem Aufwand Filesysteme mit mehreren 10 Millionen Dateien ein oder mehrmals pro Tag aus Sicht der Anwendung zeitlich konsistent zu sichern.

Regelmäßiges „scrubbing“, also das Überprüfen aller Prüfsummen, einschließlich gegebenenfalls notwendiger Korrekturen schützt vor unliebsamen Überraschungen, da Probleme bereits sehr frühzeitig erkannt werden können. Das nachfolgende Beispiel verdeutlicht die Erkennung von Fehlern (siehe Listing 1).

Verschlüsselung mit ZFS

Mit der Freigabe von Solaris 11 haben Anwender nun zusätzlich die Möglichkeit, ZFS-Filesysteme und -Volumes mit sicheren Algorithmen wie AES-256 zu verschlüsseln. Voreingestellt ist „AES-128-CCM“. Sofern die eingesetzte Hardware Beschleuniger für kryptographische Operationen bietet, etwa Oracle T4- und T5-Chips oder auch die Intel AES-NI-Erweiterung, werden diese automatisch vom Solaris-Crypto-Framework erkannt und genutzt. Eine Verschlüsselung des Root-Filesystems ist derzeit ausschließlich für nicht globale Zonen möglich. Hierzu später mehr.

Zwei Punkte sind vor dem Einsatz der Verschlüsselung zu beachten. Zum einen kann sie nur für neu anzulegende Filesysteme und Volumes aktiviert werden und zum anderen ist auch eine spätere Deaktivierung nicht möglich. Man bindet sich also für die Lebensdauer der Filesysteme. Alle notwendigen Befehle sind in das „zfs“-Kommandozeilen-Tool integriert. Der Einsatz von Verschlüsselung ist auch bei bereits existierenden Pools möglich, solange diese mindestens die Versionsnummer 30 tragen oder ein dahingehender Upgrade möglich ist (siehe Listing 2). Die aktuellen, auf den ehemaligen OpenSolaris-Quellen basierenden ZFS-Versionen in Illumos und FreeBSD bieten die Möglichkeit zur Verschlüsselung nicht. Die zugehörigen ZFS-Properties lassen sich wie gewohnt auslesen (siehe Listing 3).

Der einmal gewählte Verschlüsselungsalgorithmus ist nicht änderbar. Alle zugehörigen ZFS-Properties wer-

```
# zpool scrub testpool
# sleep 15 ; zpool status -v testpool

  pool: testpool
 state: ONLINE
status: One or more devices has experienced an
        unrecoverable error. An attempt was made to correct
        the error.  Applications are unaffected.
action: Determine if the device needs to be replaced, and
        clear the errors using 'zpool online' or replace the
        device with 'zpool replace'.
        see: http://www.sun.com/msg/ZFS-8000-9P
 scrub: scrub in progress, 6.21% done, 0h4m to go
config:
        NAME          STATE   READ WRITE CKSUM
        testpool      ONLINE     0     0     0
          mirror-0    ONLINE     0     0     0
            c4t0d0s0  ONLINE     0     0     0
            c4t1d0s0  ONLINE     0     0    58  228.5 repaired
```

Listing 1

```
# zfs create -o encryption=aes-256-ccm pool/thomas
Enter passphrase for 'pool/thomas':
Must be at least 8 characters.
Enter passphrase for 'pool/thomas':
Enter again:
```

Listing 2

den vererbt, also alle nachgeordneten Filesysteme, insbesondere auch Snapshots und Clones, sind ebenfalls zwingend verschlüsselt.

Die bei ZFS zum Einsatz kommende Schlüsselverwaltung ist zweistufig. Nach außen sichtbar ist der sogenannte „Wrapping Key". Er wird dem System beispielsweise aus einer „passphrase" generiert. Alternativ kann der Schlüssel auch als Byte-Folge oder Hexadezimal-String bereitgestellt werden. Der Wrapping Key dient ausschließlich dazu, den eigentlichen, vom Kernel verwendeten Schlüssel, den „Encryption Key", zu schützen. Beim Setzen einer neuen „passphrase" verschlüsselt das System nur den vorhandenen Encryption Key neu. Der Datenbestand bleibt davon unberührt, die Daten werden also nicht komplett neu kodiert, sondern bleiben mit den jeweiligen alten Encryption Keys verschlüsselt. Dies gilt übrigens auch für die Änderung des Encryption Key durch den Solaris-Kern. Hier wird lediglich eine ausreichend lange Byte-Folge vom System zufällig ausgewählt. Hier nutzt das System mittels Solaris Crypto Framework gegebenenfalls vorhandene Zufallszahlen-Generatoren. Das Kommando „# zfs key -K pool/thomas" übernimmt diese Aufgabe. Die Änderung spiegelt sich direkt in den Properties wieder (siehe Listing 4).

Der neue Schlüssel kommt dann für alle ab dem Änderungszeitpunkt geschriebenen neuen Daten zum Einsatz. Änderungen dieses internen Schlüssels sollten eher selten und im Einklang mit den lokal vorgegebenen Sicherheitsempfehlungen vorgenommen werden. Das National Institute of Standards and Technology (NIST) empfiehlt, derartige Schlüssel alle zwei Jahre zu wechseln.

Gänzlich anders verhält es sich mit den zu jedem verschlüsselten ZFS-Volume beziehungsweise -Filesystem gehörenden individuellen Schlüsseln und „passphrases". Diese lassen sich nach Belieben ändern. Um sie vom Benutzer abzufragen oder auch aus einem Gerät auszulesen, stehen mehrere Mechanismen und Formate zur Verfügung. Dies sind derzeit:

- *Prompt*
  Interaktive Abfrage der „passphrase", wenn das Filesystem erzeugt oder „gemountet" wird
- *file://filename*
  Der Schlüssel wird aus einer Datei, etwa aus einem USB-Stick, gelesen
- *pkcs11*
  Der Schlüssel wird aus einem PKCS#11-Token ausgelesen
- *https://location*
  Der Schlüssel wird von einem Web-Server über eine HTTPS-Verbindung bereitgestellt

Dabei stehen folgende Formate zur Verfügung:

- *Raw*
  Bytefolge
- *Hex*
  Hexadezimal-String
- *Passphrase*
  Passwort, aus dem der Schlüssel generiert wird

Achtung: Beim Aushängen („umount") eines verschlüsselten Filesystems werden die Schlüssel nicht aus dem Kernel entfernt, daher kann ein Administrator das Filesystem bis zu einem „reboot" wieder „mounten" – und zwar ohne dass der Schlüssel benötigt wird. Um dies zu verhindern, muss der Encryption Key des infrage kommenden Filesystems oder Volumes mit „# zfs key -u pool/thomas" explizit aus dem Kernel entfernt werden. Die folgenden Beispiele verdeutlichen den einfachen Umgang mit den Verschlüsselungsfähigkeiten von ZFS. Wechsel des Schlüssels (siehe Listing 5), Laden des Schlüssels in den Kernel (siehe Listing 6) und Wechsel des Bereitstellungsmechanismus, etwa aus einer Datei (siehe Listing 7). Diese ist im Idealfall auf einem leicht entfernbaren

```
# zfs get \
      encryption,keychangedate,keysource,keystatus,rekeydate \
      pool/thomas
NAME         PROPERTY       VALUE                  SOURCE
pool/thomas  encryption     aes-256-ccm            local
pool/thomas  keychangedate  Sat Sep 15 18:03 2012  local
pool/thomas  keysource      passphrase,prompt      local
pool/thomas  keystatus      available              -
pool/thomas  rekeydate      Sat Sep 15 18:03 2012  local
```

Listing 3

```
# zfs get keychangedate,rekeydate pool/thomas
NAME          PROPERTY       VALUE                  SOURCE
pool/thomas   keychangedate  Sat Sep 15 18:03 2012  local
pool/thomas   rekeydate      Sat Sep 15 18:06 2012  local
```

Listing 4

```
# zfs key -c pool/thomas
Enter new passphrase for 'pool/thomas':
Enter again:
```

Listing 5

```
# zfs key -l pool/thomas
Enter passphrase for 'pool/thomas':
```

Listing 6

```
# echo "My Big Secret" > /root/KEY
# chmod 400 /root/KEY
# zfs key -u pool/thomas
# zfs set keysource=passphrase,file:///root/KEY pool/thomas
# zfs key -l pool/thomas
```

Listing 7

```
# echo "My_Passphrase" > /root/KEY_TN
# zfs create -o encryption=aes-256-ccm \
             -o keysource=passphrase,file:///root/KEY_TN \
             rpool/home/tn_enc
# ls -l /home/tn_encrypt
total 0

obi-wan# zfs set shadow=file:///home/tn rpool/home/tn_encrypt
obi-wan# ls -l /home/tn_enc
total 4
drwx------   2 nau      kizinfra       2 Sep 10 15:08 bin
drwx------   2 nau      kizinfra       2 Jul 10 08:04 doc
drwx------   2 nau      kizinfra       2 Oct 27  2002 src
```

Listing 8

Datenträger wie einem USB-Memory-Stick gespeichert.

Die Verschlüsselung von Home-Directories wird in Solaris 11 durch Pluggable Authentication Modules (PAM) unterstützt, sofern diese in jeweils eigenen ZFS-Filesystemen untergebracht sind. In diesem Fall leitet sich der Wrapping Key aus dem Passwort des Nutzers ab. Weiterführende Informationen hierzu stehen unter „https://blogs.oracle.com/darren/entry/user_user_home_directory_encryption" in Darren Moffats Blog.

Verschlüsselung versus Deduplizierung

Das Zusammenspiel zwischen ZFS-eigener Komprimierung, Verschlüsselung und Deduplizierung stellt sich beim Schreiben von Daten folgendermaßen dar:

1. Komprimierung der Daten, sofern aktiviert
2. Verschlüsselung der komprimierten Daten, sofern aktiviert
3. Bildung der Prüfsumme
4. Deduplizierung, sofern aktiviert

Da die zur Verschlüsselung der Datenblöcke eingesetzten Encryption Keys zwischen unterschiedlichen Filesystemen im Allgemeinen nicht gleich sind, unterscheiden sich auch identische Eingangsdaten nach dem zweiten Schritt der Prozesskette. Dies kann erheblichen Einfluss auf die zu erwartende Deduplizierungsrate des gesamten Pools haben.

Zonen

Ebenfalls neu in Solaris 11 sind die sogenannten „Immutable Zones". Sie bieten im Vergleich zu herkömmlichen Zonen eine noch weiter reichende Sicherheit, da das zugehörige Root-Filesystem vor Manipulationen innerhalb der Zone geschützt wird. Dieser Schutz kann nach Wahl sehr strikt ausfallen, also keinerlei Ausnahmen zulassen, oder auch flexibel gestaltet sein. Im letzteren Fall können beispielsweise Dateien im Home-Directory von „root" oder in „/etc" und „/var" verändert werden. Auch eine Kombination mit verschlüsselten Filesystemen ist leicht zu bewerkstelligen, wie Darren Moffat in einem seiner Blog-Beiträge unter https://blogs.oracle.com/darren/entry/immutable_zones_on_encrypted_zfs zusammengefasst hat.

Die Solaris 11 „Shadow Migration" gibt Administratoren ein Mittel an die Hand, um lokale, aber auch NFS-gemountete UFS- und ZFS-Filesysteme in ein neues ZFS-Filesystem zu migrieren. Dies geschieht weitestgehend transparent für die Nutzer des Systems. Eine Ausnahme bilden die gelegentlich auftretenden Verzögerungen beim Datenzugriff, sofern eine Datei noch nicht migriert wurde. Die zu erfüllenden Voraussetzungen sind einfach:

- Das Quell-Filesystem muss „read-only" gemountet sein und darf im Falle von NFS auch serverseitig nicht verändert werden
- Das Ziel-Filesystem muss leer sein

Mittels „Shadow Migration" lassen sich zum Beispiel bestehende Home-Verzeichnisse transparent und quasi im laufenden Betrieb in verschlüsselte ZFS Filesysteme migrieren, da ja eine nachträgliche Aktivierung der Verschlüsselung für bereits benutzte Filesysteme nicht möglich ist. Lediglich die notwendige Anpassung des Pfades für die Verzeichnisse, etwa in „/etc/passwd", hat gegebenenfalls eine Unterbrechung für die jeweiligen Nutzer zur Folge. Das nachfolgende Beispiel verdeutlicht die notwendigen Schritte (siehe Listing 8).

Thomas Nau
thomas.nau
@uni-ulm.de