ZFS –
Verschlüsselung und andere Neuigkeiten in Solaris 11

Thomas Nau
Universität Ulm - kiz
Ulm

Schlüsselworte:
ZFS, Verschlüsselung, Security, Solaris Container, Zonen

Einleitung:
Der sichere und effiziente Umgang mit großen und größten Datenmengen gehört heute sicherlich zu den herausragenden Aufgaben jedes IT-Infrastruktur-Dienstleisters. Dabei stehen besonders die Datenintegrität, die technischen und organisatorischen Maßnahmen, die diese gewährleisten sollen und der Sicherheitsaspekt beim Zugriff auf diese Daten im Vordergrund.
ZFS hat mit seinem Erscheinen in Solaris 10 im Bereich der Datenintegrität nicht nur neue Maßstäbe gesetzt, sondern sich in den vergangenen Jahren zu "dem" Maßstab für Filesysteme schlechthin entwickelt, an dem sich andere Implementierungen messen lassen müssen. Die Dynamik der ZFS Weiterentwicklung ist jedoch ungebrochen. Mit Solaris 11 stehen, neben vielen Performance-verbesserungen unter der Haube, auch wieder neue Features zur Verfügung. Es besteht jetzt die Möglichkeit ZFS Filesysteme und Volumes transparent zu verschlüsseln und diese Datenbereiche dann z.B. für virtualisierte Systeme in Zonen zu nutzen. Auch im Bereich der Datenmigration hat Solaris 11 Neuerungen zu bieten.

Hintergrund des Autors:
Das Kommunikations- und Informationszentrum (kiz) der Universität Ulm trägt unter anderem die Gesamtverantwortung für deren IT-Infrastruktur, inklusive Telefonie, sowie deren Versorgung mit sowohl elektronischen als auch mit Print-Medien. Die Kernaufgaben der Abteilung Infrastruktur, deren Leiter der Autor ist, umfassen hierbei insbesondere Planung, Weiterentwicklung und den Betrieb der Netzwerke, sowie aller zentralen Server. Zu diesen zählen neben Backup- und HPC-Systemen insbesondere auch die auf HA-Clustern basierenden Mail-, LDAP-, Portal-, Datenbank- und File-Server der Universität Ulm.

ZFS Grundlagen aus 10.000m Höhe:
Für den Betrieb einer derartigen zentralen Infrastruktur sind File- und Betriebssysteme notwendig, die weitreichende Vorkehrungen zum Schutz der Datenintegrität, der Datensicherheit und damit vor Datenverlust bieten. Bei der Entwicklung von ZFS wurden insbesondere auch die designbedingten Schwachstellen herkömmlicher Filesysteme korrigiert.

- "Starke" Prüfsummen wie *fletcher4* oder *sha256* ermöglichen es, Fehler auf dem gesamten Datenpfad zu erkennen und diese, bei entsprechender redundanter Auslegung der Platten-Systeme, zu korrigieren. Die Prüfsummen werden getrennt von den Daten im sogenannten Pointer-Bereich abgelegt.
- Gültige Daten werden niemals überschrieben (copy on write, COW).
- Besonders wichtige Metadaten werden mehrfach, und von der übergeordneten Redundanz unabhängig, in sogenannten ditto-blocks abgespeichert.

ZFS bietet darüber hinaus eine Vielzahl betrieblicher Vorteile, etwa snapshots und clones. Diese kommen vermehrt als Ergänzung bzw. als Ersatz üblicher Backup-Lösungen zum Einsatz. Nur so sind, mit vergleichsweise geringem Aufwand, Filesystemen mit mehreren 10 Millionen Dateien, ein oder mehrmals pro Tag zu sichern.
Regelmäßiges scrubbing, also das Überprüfen aller Prüfsummen einschließlich ggf. notwendiger Korrekturen, schützt vor unliebsamen Überraschungen, da Probleme bereits sehr frühzeitig erkannt werden können. Das nachfolgende Beispiel verdeutlicht die Erkennung von Fehlern:

Verschlüsselung von ZFS Filesystemen und Volumes:
Mit der Freigabe von Solaris 11 haben Anwender nun zusätzlich die Möglichkeit ZFS Filesysteme und Volumes mit "starken" Algorithmen, etwa AES-256, zu verschlüsseln. Voreingestellt ist AES-128-CCM. Sofern die Hardware Beschleuniger für kryptographische Operationen bietet, wie etwa Oracles T4-Chip, werden diese automatisch vom Solaris crypto framework genutzt. Das root-Filesystem lässt sich derzeit ausschließlich für Zonen verschlüsseln. Dazu später mehr.
Zwei Punkte sind im Vorfeld zu beachten. Die Verschlüsselung kann nur für neu anzulegende Filesysteme und Volumes aktiviert werden. Auch ist eine spätere Deaktivierung nicht möglich, man bindet sich also für's Leben. Alle notwendigen Befehle sind in das *zfs* Kommandozeilentool integriert. Die Verschlüsselung lässt sich auch bei älteren, bereits existierenden, Pools verwenden so lange diese mindestens der Version #30 entsprechen oder upgegraded werden.
Die aktuellen, auf den OpenSolaris Quellen basierenden, ZFS-Versionen in Illumos und FreeBSD bieten die Möglichkeit zur Verschlüsselung nicht.

```
obi-wan# 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:
```

Die zugehörigen ZFS properties:

```
obi-wan# 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
```

Der einmal gewählte Verschlüsselungsalgorithmus ist nicht änderbar. Alle zugehörigen ZFS properties werden vererbt, d.h. alle nachgeordneten Filesysteme, ebenso snapshots und clones, sind ebenfalls zwingend verschlüsselt.

Die ZFS Schlüsselverwaltung ist zweistufig. Der sogenannte *wrapping-key* ist derjenige der nach außen "sichtbar" ist d.h. dem System z.B. indirekt über die *passphrase* bereitgestellt werden muss bzw. aus ihr generiert wird. Alternativ kann der Schlüssel auch als Bytefolge oder Hex-String bereitgestellt werden. Dieser *wrapping-key* dient ausschließlich dazu, den eigentlichen *encryption key* zu schützen. Bei einer Änderung des *wrapping-keys* wird vom System nur der *encryption key* neu verschlüsselt. Der zu diesem Zeitpunkt vorhandene Datenbestand bleibt davon unberührt. Dies gilt auch für die Änderung bzw. eigentlich die zufällige Auswahl eines neuen *encryption keys* durch den Solaris Kernel. Das folgende Kommando übernimmt diese Aufgabe:

```
obi-wan# zfs key -K pool/thomas

obi-wan# 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
```

Die zum Zeitpunkt der Änderung vorhandenen Daten bleiben mit den jeweiligen alten *encryption keys* verschlüsselt. Der neue Schlüssel kommt erst für die ab dem Änderungszeitpunkt geschriebenen, neuen Blöcke zum Einsatz. Änderungen des *encryption key* sollten eher selten und im Einklang mit den lokal vorgegebenen Sicherheitsempfehlungen vorgenommen werden. NIST (National Institute of Standards and Technology) empfiehlt, derartige Schlüssel alle 2 Jahre zu wechseln.

Gänzlich anders verhält es sich mit den zu jedem verschlüsselten ZFS Volume bzw. Filesystem gehörenden individuellen Schlüsseln bzw. *passphrases*. Diese können nach Belieben geändert werden. Um sie vom Benutzer abzufragen, oder auszulesen, stehen mehrere Mechanismen und Formate zur Verfügung. Dies sind derzeit folgende Mechanismen:

- prompt — Abfrage wenn das Filesystem erzeugt oder gemountet wird
- file:///filename — Schlüssel wird aus Datei (USB-Stick) gelesen
- pkcs11 — Schlüssel wird aus einem PKCS#11 token gelesen
- https://location — Schlüssel ist auf einem Web-Server abgelegt

Als Format für die Schlüssel bestehen folgende Möglichkeiten:

- raw — Bytefolge
- hex — Hexadezimal-String
- passphrase — Passwort aus dem der Schlüssel generiert wird

Achtung: das Aushängen eines verschlüsselten Filesystems sperrt die Schlüssel nicht. Dies bedeutet, dass ein Administrator das Filesystem trotzdem wieder mounten kann und zwar ohne, dass der Schlüssel benötigt wird. Um dies zu verhindern, muss der encryption key des Filesystems oder Volumes explizit aus dem Kernel entfernt werden: `zfs key -u filesystem`

Die folgenden Beispiele verdeutlichen den einfachen Umgang.

Wechsel des Schlüssels:

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

Schlüssel in den kernel laden:

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

Wechsel der Ablage des Schlüssels in eine Datei. Diese sollte z.B. auf einem USB Memory Stick gespeichert sein.

```
obi-wan# echo "MySecret" > /root/KEY
obi-wan# chmod 400 /root/KEY
obi-wan# zfs key -u pool/thomas
obi-wan# zfs set keysource=passphrase,file:///root/KEY pool/thomas
obi-wan# zfs key -l pool/thomas
obi-wan# ls /pool/thomas/
```

```
obi-wan# zfs key -l pool/thomas
obi-wan# ls /pool/thomas/
File
```

Die Verschlüsselung von Homedirectories mit jeweils eigenen ZFS Filesystemen wird in Solaris 11 durch PAM (pluggable authentication module) unterstützt. Dabei leitet sich der *wrapping key* aus dem Passwort des Nutzers ab. Interessierte finden Details hierzu in Darren Moffats Blog:
https://blogs.oracle.com/darren/entry/user_user_home_directory_encryption

Auch *immutable zones* sind neu in Solaris 11. Sie bieten eine noch weiter reichende Sicherheit, indem das zugehörige root-Filesystem vor Manipulationen innerhalb der Zone geschützt wird. Dieser Schutz kann nach Wahl strikt ausfallen, d.h. keinerlei Ausnahmen zulassen, oder auch flexibel gestaltet sein. Im letzteren Fall können beispielsweise Dateien im Homediretory von *root* oder in */etc* und */var* verändert werden. Auch eine Kombination mit verschlüsselten Filesystemen ist leicht zu bewerkstelligen wie Darren Moffat in einem weiteren seiner Blog Beiträge verdeutlicht.
https://blogs.oracle.com/darren/entry/immutable_zones_on_encrypted_zfs

Das Zusammenspiel der Verschlüsselung mit Deduplizierung und Komprimierung stellt sich beim Schreiben von Daten wie folgt dar:

1. komprimieren der Daten
2. verschlüsseln der komprimierten Daten
3. Bildung der Prüfsumme
4. Deduplizierung sofern aktiviert

Da die zur Verschlüsselung der Datenblöcke verwendeten Schlüssel in unterschiedlichen Volumes und Filesystemen, auch bei gleicher *passphrase*, im Allgemeinen unterschiedlich sind, unterscheiden sich identische Eingangsdaten nach dem zweiten Schritt der Kette. Dies kann erheblichen Einfluss auf die zu erwartende Deduplizierungsrate des gesamten Pools haben.

Migration von Filesystemen:
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 ist für Anwender transparent. Eine Ausnahme bilden nur die gelegentlich auftretenden Verzögerungen beim Datenzugriff sofern eine Datei noch nicht migriert wurde. Die zu erfüllenden Voraussetzungen sind einfach:

- Das Quellfilesystem muss read-only gemountet sein und sollte auch von anderen Maschinen nicht verändert werden.
- Das Zielfilesystem muss leer sein.

Mit diesem Mechanismus lassen sich z.B. bestehende Homedirectories transparent und quasi im laufenden Betrieb in verschlüsselte ZFS Filesysteme migrieren da eine Aktivierung für bereits benutzte Filesysteme nicht möglich ist. Lediglich die notwendige Änderung des Homedirectories in */etc/passwd* hat ggf. eine Unterbrechung zur Konsequenz

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

```
obi-wan# zfs set shadow=file:///home/tn rpool/home/tn_enc
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
drwx--x--x   2 nau      kizinfra       2 Sep 13 13:04 tmp
```

Dank:
Mein besonderer Dank für die großartige Unterstützung geht an Cindy Swearingen, Darren Moffat und Jan Friedel von Oracle sowie meinen Kollegen Dr. Harald Däubler.

Kontaktadresse:
Thomas Nau
Universität Ulm -kiz
Albert Einstein Allee 11
D-89081 Ulm

Telefon:	+49 (0) 731 50-22464
Fax:	+49 (0) 731 50-12-22464
E-Mail	Thomas.Nau@uni-ulm.de
Internet:	http://www.uni-ulm.de/einrichtungen/kiz