Angst vor dem Datengau?

Tipps und Tricks für einen besseren Schlaf

Thomas Nau
Universität Ulm – kiz
Ulm

Schlüsselworte
Backup, ZFS, Ransomware, Datenmanagement

Einleitung
Ransomware, auch als Verschlüsselungs-Trojaner bekannt, ist heute in aller Munde, wenn über die Sicherheit bzw. Integrität von Daten diskutiert wird. Längst ist dieses Problem aber nicht mehr auf private Systeme beschränkt, sondern es werden auch immer wieder Fälle in Behörden, Unternehmen usw. weltweit publik, wobei die Dunkelziffer als hoch einzuschätzen ist.

Mindestens ebenso problematisch wie diese offensichtliche und reale Bedrohung sind jedoch Defizite im eigenen Datenmanagement. Diese Defizite sind dabei oft fehlender Anpassung an die schnell fortschreitende technische Entwicklung geschuldet, aber auch der enormen Zunahme von Datenbeständen. Darüber hinaus halten sich manche Mythen seit Jahren, auch im professionellen Umfeld, und bremsen damit die interne Anpassung von Prozessen.

Der Autor verdeutlicht an Hand von Beispielen welche mächtigen Werkzeuge mit ZFS und Solaris heute zur Verfügung stehen, und wie sie uns geholfen haben, den zweimaligen Befall zentraler Systeme mit Ransomware unbeschadet zu überstehen. Daneben wird er beleuchten an welchen Stellen, nach seiner Erfahrung, vorhandene Sicherungs- und vor allem auch Wiederherstellungsstrategien zu überdenken sind und welche Vorteile sich hier aus dem Einsatz neuer Techniken ergeben. Zu guter Letzt wird er einigen Mythen auf den Grund gehen bzw. ihnen die Grundlage entziehen.

Hintergrund des Autors
Der Autor ist seit mehr als 30 Jahren im IT Umfeld tätig und leitet derzeit die „Abteilung Infrastruktur“ des Kommunikations- und Informationszentrum (kiz) an der Universität Ulm. Das kiz, dessen stellvertretender Leiter der Autor ebenfalls ist, trägt unter anderem die Gesamtverantwortung für die universitäre IT-Infrastruktur. Diese umfasst auch die Telefonie, sowie die Versorgung der Wissenschaftler und Studenten sowohl mit elektronischen als auch mit Print-Medien.
Die Kernaufgaben der Abteilung Infrastruktur umfassen hierbei insbesondere Planung, Weiterentwicklung und den Betrieb der Netzwerke, sowie aller zentralen Server. Desweiteren ist die Abteilung sehr stark in Projekte[1] des Landes Baden-Württemberg eingebunden und erbringt in diesem Zusammenhang auch Dienstleistungen für weitere Hochschulen des Landes basierend auf dem leistungsfähigen Landeshochschulnetzes BelWü[2].

1 bwCloud: Standortübergreifende Servervirtualisierung
bw100G: Forschung und innovative Dienste für ein flexibles 100G-Netz in Baden- Württemberg
bwHPC, bwHPC-C5: http://www.bwhpc-c5.de
2 http://www.belwue.de

Ransomware
Wikipedia definiert den Begriff wie folgt:

> *Ransomware (von englisch ransom für „Lösegeld"), auch Erpressungstrojaner, Erpressungssoftware, Kryptotrojaner oder Verschlüsselungstrojaner, sind Schadprogramme, mit deren Hilfe ein Eindringling den Zugriff des Computerinhabers auf Daten, deren Nutzung oder auf das ganze Computersystem verhindern kann. Dabei werden private Daten auf dem fremden Computer verschlüsselt oder der Zugriff auf sie verhindert, um für die Entschlüsselung oder Freigabe ein Lösegeld zu fordern.*

Derartige Software existiert bereits seit den 80er Jahren des letzten Jahrhunderts, erlangte aber erst in den letzten Jahren traurige Berühmtheit als Massenphänomen.
Sieht man von gezielten Angriffen gegen Personen oder Unternehmen ab, diesen liegen oft spezielle Angriffsmethoden zu Grunde, so werden die meisten Betroffenen eher zufällig Opfer. Die Einfallstore sind dabei mannigfaltig. E-Mail attachments und Drive-by-Downloads, ausgelöst von aktiven Inhalten und für den Nutzer transparente Weiterleitung auf entsprechende Web-Seiten, sind nur zwei mögliche Wege. Andererseits ist auch eine wurmartige Verbreitung über unbekannte oder zero-day Lücken wie im Fall von *WannaCry* möglich[3], ebenso wie eine Kombination mehrerer Verbreitungsmethoden.

Klar ist, dass die häufig propagierten Schutzmaßnahmen, etwa der Einsatz aktueller Virenscanner und Betriebssysteme, allenfalls das Ausmaß begrenzen. Diese Aussage wird durch die rasante Verbreitung von *WannaCry* und *Petya* auch im Unternehmensbereich gestützt, zumindest sofern man dort einen professionellen IT-Betrieb zu Grunde legt.

Daraus lässt sich schließen, dass neben Schutzmaßnahmen insbesondere Gegenmaßnahmen dringend angeraten sind. Essentiell sind im allgemeinen Bedrohungsszenario zumindest folgende Punkte:

1. Erkennung eines Angriffs
2. Existenz einer unabhängigen schreibgeschützten Datenkopie, ggf. auch Versionierung
3. Kenntnis der Zugriffsrechte
4. Organisatorische Entscheidungen: „ease of use" versus Sicherheit

Das Zeitfenster für die Erkennung definiert hierbei gleichzeitig die Randbedingung für die Dauer, für die schreibgeschützte Kopien der wichtigen Daten vorzuhalten sind.
Oft vernachlässigt wird die in Punkt 3 genannte Tatsache, dass die Zugriffsrechte einzelner Personen durchaus weit über die persönlichen Daten hinausgehen können. Als Beispiel seien Netzlaufwerke genannt, die ganzen Teams oder Abteilungen als Datenablage dienen.

Die Erkennung gestaltet sich im Allgemeinen als schwierig. Zwar existieren mittlerweile Lösungen von Sicherheitsfirmen, die das Problem auf Basis von Verhaltensanalysen adressieren, jedoch ist das Grundproblem der Erkennung dem der SPAM-Erkennung ähnlich. Während brute-force klare Muster liefert und oft sogar der Nutzer direkt „zur Kasse gebeten wird" sind subtile Angriffe nur äußerst aufwändig erkennbar. Etwa wenn nur wenige Dateien pro Zeitintervall verschlüsselt werden und diese darüber hinaus nach Typ und Alter ausgewählt werden.

3 https://www.heise.de/security/meldung/Alles-was-wir-bisher-ueber-den-Petya-NotPetya-Ausbruch-wissen-3757607.html?artikelseite=all

Hilfsmittel in Solaris
Solaris bietet bereits seit Jahren die Hilfsmittel an, die die Datenintegrität und Datensicherheit in hohem Maß unterstützen. Darren Moffat schreibt darüber 2008 in seinem Blog *Making files on ZFS Immutable (even by root!)*[4]. Grundlage ist hierbei ZFS. Durch dessen Verfügbarkeit in weiteren Betriebssystemen, sehr früh schon in FreeBSD nun zunehmend auch in Linux, gilt das nachfolgend gesagte weitestgehend auch für diese.
Diese Mittel können im Falle der Ransomware-Bedrohung fileserverseitig dazu genutzt werden, die dort gespeicherten Daten zu schützen bzw. den Verlust drastisch zu minimieren. Generell lassen sich die skizzierten Lösungen auch für Datenbank-, Web- oder Mailserver einsetzen jedoch sollten Endanwender dort eigentlich keinen direkten Zugriff auf die Filesysteme erhalten, also die Schnittstelle der sich heutige Ransomware bedient, sollte hier nicht zur Verfügung stehen.
Hinweis: um die Lesbarkeit zu verbessern wurden Ausgaben im Folgenden teilweise gekürzt oder umgebrochen.

Zutat #1: ZFS snapshots
Unabhängig von lokalen Backup-Strategien bietet ZFS mit seiner snapshot Funktionalität sehr einfach, schnell und ressourcenschonend die Möglichkeit, die obige Forderung #2 nach schreibgeschützten Kopien von Daten zu bedienen. ZFS nutzt hierzu seine Basistechnologie *copy-on-write* aus. Die einer Änderung als Basis dienenden Blöcke werden dabei nicht in die Freiliste überführt, sondern bleiben den jeweiligen snapshots zugeordnet.
Die Solaris SMF Services

svc:/application/time-slider:default
*svc:/system/filesystem/zfs/auto-snapshot:**

übernehmen das Management hinsichtlich Erzeugung und Löschung der snapshots nach vordefinierten Regeln. Wir raten dazu nicht die default Instanzen zu verwenden, sondern für jede Aufgabe eigene Instanzen zu erzeugen. Dies ist einfach wie folgt etwa für einen NFS Server möglich. Exemplarisch definieren wir zwei neue Services, die stündliche und tägliche snapshots erzeugen. Letztere stehen für 28 Tage zur Verfügung, die stündlichen decken jeweils die letzten 24 Stunden ab.

```
INSTANCE=nfs
I_HOUR=${INSTANCE}-hourly
I_DAY=${INSTANCE}-daily

svccfg -s svc:/system/filesystem/zfs/auto-snapshot add ${I_HOUR}
svccfg -s auto-snapshot:${I_HOUR} addpg zfs application
svccfg -s auto-snapshot:${I_HOUR} setprop zfs/interval = hours
svccfg -s auto-snapshot:${I_HOUR} setprop zfs/keep = 23
svccfg -s auto-snapshot:${I_HOUR} setprop zfs/period = astring: 1
svccfg -s auto-snapshot:${I_HOUR} addpg general framework
svccfg -s auto-snapshot:${I_HOUR} setprop general/complete = astring: \"\"
svcadm refresh  svc:/system/filesystem/zfs/auto-snapshot:${I_HOUR}

svccfg -s svc:/system/filesystem/zfs/auto-snapshot add ${I_DAY}
svccfg -s auto-snapshot:${I_DAY} addpg zfs application
svccfg -s auto-snapshot:${I_DAY} setprop zfs/interval = days
svccfg -s auto-snapshot:${I_DAY} setprop zfs/period = astring: 1
svccfg -s auto-snapshot:${I_DAY} setprop zfs/keep = 28
svccfg -s auto-snapshot:${I_DAY} addpg general framework
```

4 https://blogs.oracle.com/darren/making-files-on-zfs-immutable-even-by-root

```
svccfg -s auto-snapshot:${I_DAY} setprop general/complete = astring: \"\"
svcadm refresh svc:/system/filesystem/zfs/auto-snapshot:${I_DAY}
```

Analog lassen sich auch wöchentliche snapshots generieren oder, bei Bedarf, auch welche im Abstand einiger Minuten.

Zu debugging Zwecken ist anfangs die Aktivierung von debug-Meldungen hilfreich. Diese finden sich in */var/svc/log/application-time-slider:default.log*

```
svccfg -s time-slider setprop daemon/verbose = true
```

jedoch natürlich erst nachdem die SMF services aktiviert wurden:

```
svcadm refresh svc:/application/time-slider:default
svcadm enable  svc:/application/time-slider:default
svcadm enable  svc:/system/filesystem/zfs/auto-snapshot:${I_HOUR}
svcadm enable  svc:/system/filesystem/zfs/auto-snapshot:${I_DAY}
```

Die dadurch entstehenden snapshots lassen sich auch auf Grund ihres Namens einfach von anderen unterscheiden.

```
pool1/home/kiz@zfs-auto-snap_nfs-hourly-2017-09-24-16h09
pool1/home/kiz@zfs-auto-snap_nfs-daily-2017-09-24-16h58
pool1/home/kiz@zfs-auto-snap_nfs-hourly-2017-09-24-17h09
pool1/home/kiz@zfs-auto-snap_nfs-hourly-2017-09-24-18h09
```

Bleibt die Frage wie die snapshots gegen typische Ransomware Probleme helfen. Dazu muss man sich vor Augen führen welche Fileoperationen im Zuge der Verschlüsselung auf die Dateien angewendet werden. Vereinfacht werden folgende Schritte pro Datei durchlaufen:

1. Öffnen der Quelldatei *S*
2. Anlegen und Öffnen einer Zieldatei *T* deren Namen aus dem verschlüsselten Dateinamen von *S* besteht. Hier ist im Allgemeinen ein encoding notwendig um spezielle Zeichen wie „:“ zu eliminieren.
3. Verschlüsselung der Daten aus *S* nach *T*
4. Löschen der Quelldatei *S*

```
# ls -l
total 49
-rw-r--r--   1 root     root       15205 Sep 24 19:38 etc.txt
-rw-r--r--   1 root     root        8694 Sep 24 19:37 var.tmp.txt

# for s in *.txt; do
    t=$(echo $s | encrypt -a aes -k /tmp/MyKey | base64 -w 0 | tr '/' '@')
    encrypt -a aes -k /tmp/MyKey -i "$s" -o "$t"
    rm "$s"
done

# ls -l
total 50
-rw-r--r--   1 root     root        8744 Sep 24 19:40 AAAAAQAAA+h07s8rfI...
-rw-r--r--   1 root     root       15256 Sep 24 19:40 AAAAAQAAA+jUjV4IEU…
```

Mit Hilfe des Kommandos *zfs diff* ist man nun in der Lage sich Änderungen des aktuellen datasets bezüglich eines oder mehrerer snapshots anzeigen zu lassen. Im Falle unseres Beispiels wurde vor dem „Angriff“ ein snapshot mit Namen *snap* angelegt.

```
# zfs diff -o user -o oldname -o name rpool/test@snap
M     root  -     /test/
-     root  -     /test/var.tmp.txt
-     root  -     /test/etc.txt
+     root  -     /test/AAAAAQAAA+jUjV4IEU...
+     root  -     /test/AAAAAQAAA+h07s8rfI…
```

Erkennbar ist die Änderung des Verzeichnisses */test* (M) sowie die Löschung (-) und Erzeugung (+) zweier Dateien des Besitzers *root*. Mit diesen Informationen, sowie dem bei Bedarf zusätzlich einblendbaren Zeitstempel, lassen sich die bertroffenen Dateien sehr gut eingrenzen. Anschliessend können die Daten aus dem für jede Datei jeweils neuesten „guten“ snapshot zurück kopiert werden. Je nach zeitlicher Granularität der vorhandenen snapshots und dem Startzeitpunkt des Angriffs sind dessen Auswirkungen erheblich reduziert oder gar vernachlässigbar.

Bedenken hinsichtlich des durch die snapshots benötigten zusätzlichen Plattenplatzes lassen sich durch die beiden folgenden Beispiele leicht zerstreuen.
Im ersten Fall handelt es sich um einen Mailserver. Die letzten 24 Stunden werden mit stündlichen, die letzten 3 Monate mit täglichen, snapshots abgedeckt.

```
# zfs list -o space -r data2/mail
NAME         AVAIL   USED  USEDSNAP  USEDDS  USEDREFRESERV  USEDCHILD
data2/mail  50.0T  10.9T     5.35T   5.57T              0          0
```

Die snapshots schlagen hier mit rund 50% des gesamten Bedarfs zu Buche. Es ist jedoch zu bedenken, dass gelöschte Mails, auch SPAM, erst nach 90 Tagen wirklich aus dem System verschwinden.

Ein Solaris CIFS Server, dieser bedient rund 200 Arbeitsplätze im Administrativen Bereich, dient als ein weiteres Beispiel. Die nahezu identische Konfiguration deckt hierbei nur den letzten Monat mit täglichen snapshots ab, wird jedoch durch ein herkömmliches Backup System ergänzt.

```
# zfs list -o space -r smb/cifs
NAME         AVAIL   USED  USEDSNAP  USEDDS  USEDREFRESERV  USEDCHILD
smb/cifs    27.2T  6.25T     1019G   5.26T              0          0
```

In diesem Fall beträgt der zusätzliche Bedarf rund 20%. Ein geringer Aufschlag für den gewonnenen zusätzlichen Schutz.

Tools wie *ZnapZend*[5] ersetzen nicht nur die von Solaris bereit gestellte time-slider Funktionalität, sondern erweitern diese auch erheblich. Unter anderem erlaubt es das Tool die snapshots mit mittels der ZFS Mechanismen *send/receive* auf ein remote System zu übertragen. Unterschiedliche retention Zeiten für die snapshots auf Quell- und Zielsystem werden ebenfalls unterstützt.

Zutat #2: ZFS Datei Attribute

Vorab ein Hinweis für alle, die das Beispiel nachvollziehen möchten. Die mit Solaris ausgelieferten GNU utilities unterstützen die zusätzlichen Funktionalitäten **nicht**. Daher muss */usr/bin* zwingend im

5 http://www.znapzend.org/

Pfad vor */usr/gnu/bin* stehen. Außerdem sind ggf. *shell-aliases* für Kommandos zu prüfen falls das Ergebnis nicht dem Erwarteten entspricht. Analoges gilt für *MANPATH*.

ZFS File Attribute sind im Falle von Ransomware weniger, effektiv da sie im Allgemeinen auch die tägliche Arbeit zu sehr einschränken. In speziellen Fällen, bei quasi statischen Dateien, sind sie jedoch extrem hilfreich. So lässt sich mit ihrer Hilfe etwa die Manipulation von Web-Server Konfigurationen oder PHP Installationen sowie auch zugehöriger log-Dateien zuverlässig unterbinden sofern diese Dienste nicht in *immutable-zones* laufen können. Einige der Attribute sind auch in der Lage, root vom Zugriff, zumindest ohne vorherigen Eingriff, auszuschließen. Daher ist ein Test der Gesamtfunktionalität nach Aktivierung entsprechender Schutzmaßnahmen dringend angeraten.

Log-Dateien sollten im Allgemeinen nur im Sinne eines *append* veränderbar sein, um mögliche Verschleierungen zu verhindern. Der Ausgangszustand:

```
nau@alderaan:/test> ls -l logfile
-rw-r-----   1 nau      kizinfra    1808 Sep 25 10:45 logfile

nau@alderaan:/test> ls -/v logfile
-rw-r-----   1 nau      kizinfra    1808 Sep 25 10:45 logfile
               {archive,nohidden,noreadonly,nosystem,noappendonly,
                nonodump,noimmutable,av_modified,noav_quarantined,
                nonounlink,nooffline,nosparse,nosensitive}
```

Im genannten Umfeld scheint es sinnvoll die Attribute *appendonly* und *nounlink* zu setzen, die dann gemeinsam sowohl das Löschen als auch das Überschreiben der Datei verhindern.

```
nau@alderaan:/test>  chmod S+vappendonly,vnounlink logfile
chmod: ERROR: cannot set the following attributes on logfile: not
privileged
        {appendonly,nounlink}
```

Offensichtlich erlaubt Solaris es einem normalen Account nicht diese Attribute zu verändern. Warum? Um das herauszufinden kommen die debugging Möglichkeiten des least-priviledge System zum Einsatz:

```
nau@alderaan:/test> ppriv -f +D -e chmod S+vappendonly,vnounlink logfile
chmod[1289]: missing privilege "file_flag_set" (euid = 3201, syscall =
             "write") for "/test/logfile" at zfs_setattr+0xc90
chmod: ERROR: cannot set the following attributes on logfile: not
privileged
        {appendonly,nounlink}

nau@alderaan:/test> ppriv -lv file_flag_set
file_flag_set
        Allows a process to set immutable, nounlink or appendonly
        file attributes.
```

Das Problem kann auf zwei Arten gelöst werden. Zum einen könnten dem Account die notwendigen zusätzlichen Privilegien zugewiesen werden. Da dies aber mit einer Schwächung des Schutzes einher geht ist es sinnvoller die Attribute durch einen anderen, privilegierten, Account passend zu setzen.

```
root@alderaan:/test# ppriv -f +D -e chmod S+vappendonly,vnounlink logfile
```

```
root@alderaan:/test# ls -/v logfile
-rw-r-----   1 nau      kizinfra    1808 Sep 25 10:45 logfile
               {archive,nohidden,noreadonly,nosystem,appendonly,
                nonodump,noimmutable,av_modified,noav_quarantined,
                nounlink,nooffline,nosparse,nosensitive}
```

Die Wirkung ist die erwünschte:

```
nau@alderaan:/test> > logfile
bash: logfile: Not owner

nau@alderaan:/test> rm logfile
rm: logfile not removed: Not owner

nau@alderaan:/test> mv logfile tmp123456
mv: cannot rename logfile to tmp123456: Not owner

nau@alderaan:/test> echo A_NEW_LINE_APPENDED >> logfile
```

selbst mit root Privilegien:

```
root@alderaan:/test> rm logfile
rm: remove logfile (yes/no)? y
rm: logfile not removed: Not owner
```

Im Falle von Software-Installationen und insbesondere von Konfigurationsdateien ist das Setzen des *immutable* Attributes hilfreich. Auch hier sind entsprechende Privilegien notwendig.

```
root@alderaan:/test> chmod S+vimmutable httpd.conf

root@alderaan:/test> > httpd.conf
-bash: httpd.conf: Not owner

root@alderaan:/test> rm httpd.conf
rm: httpd.conf: override protection 644 (yes/no)? y
rm: httpd.conf not removed: Not owner
```

Um Anpassungen, im Beispiel an der Konfiguration des Apache Webservers, vornehmen zu können muss die Datei also zuerst „entsperrt“ werden.

```
root@alderaan:/test> chmod S-vimmutable httpd.conf

root@alderaan:/test> vi httpd.conf

root@alderaan:/test> chmod S+vimmutable httpd.conf
```

Datenmanagement

Im Rahmen des Datenmanagements gibt es einige Legenden, die über die letzten Jahrzehnte mehr oder weniger gepflegt wurden, ohne sie jemals auf den Prüfstand aktueller Technologien zu stellen.
Nach unseren Erfahrung sind im Datenbank-Umfeld viel häufiger Sicherungs- und auch Recovery-Szenarien implementiert und dokumentiert als etwa im Bereich von File-, Web- oder Mailservern.

Regelmäßige Tests und die Überprüfung, ob die Anforderungen an Ausfallzeiten bzw. deren Folgen tragbar sind, finden jedoch in nur wenigen Fällen statt.
Das bringt uns insbesondere hinsichtlich von Backupstrategien zu einigen Mythen, die sicherlich nicht nur im Hochschulumfeld anzutreffen sind.

Mythos #1

„Nutzer kennen ihre Daten, die sich daraus ergebenden Anforderungen und folgen einem Datenmanagement Konzept“

Mythos #2

„Die Netzwerk-Bandbreite ist der begrenzende Faktor, clients brauchen 10GE!“

Mythos #3

„Archiv ist doch nur Backup mit langen Lagerzeiten.“

Mythos #4

„Es existiert ein Plan für den Fall der Fälle.“

Mythos #5

„Es ist doch alles nur ein technisches Problem“

Diese Aussagen kritisch zu hinterfragen ist der Schlüssel und hat erhebliche Auswirkungen auf die notwendigen Leistungsmerkmale künftiger Lösungen, da sie immer spezifischer in einzelne Bereiche passen müssen.

Ein Beispiel ist die bereits diskutierte Problematik von Randomware oder anderen Hacker-Angriffen. Diese erfordern über die übliche Datensicherung hinausgehende Maßnahmen, stellen jedoch, wie das nachfolgende Beispiel zeigt keinen Einzelfall dar.

Wo ein herkömmliches Backup keinen Sinn macht

Das Mailsystem einer ebenfalls kleineren und von uns mit Dienstleistung unterstützten Universität enthält mehr als 75 Millionen Dateien. Die initiale Sicherung des Produktivsystems dauerte dabei über 2 Wochen. Grund hierfür sind die random-IO Zugriffe auf die Daten, die auf herkömmlichen SAN-Systemen liegen. Dies legt den Schluss nahe, dass eine vollständige Wiederherstellung ebenfalls zumindest einige Tage in Anspruch nehmen würde und in diesem Zeitraum eine Nutzung des Mail-Systems bestenfalls nur sehr eingeschränkt möglich wäre. Herkömmliche Sicherungskonzepte sind in diesem Fall also nicht zielführend.
Im folgenden sei kurz die Lösung skizziert, wie sie aktuell an der Universität Ulm im Einsatz ist.

- Wahl von ZFS als Filesystem
  Vorhaltung von je 24 stündlichen und 90 täglichen snapshots. Die feinere zeitliche Granularität von nur einer Stunde minimiert das Zeitfenster bzgl. der versehentlichen Löschung von Daten erheblich gegenüber einer normalen täglichen Datensicherung.

- Cyrus IMAP-Server feature „*delayed expunge*“
  Mails werden für 48 Stunden nur als gelöscht markiert und sind für den E-Mail client nicht mehr sichtbar. Im Filesystem werden sie erst im Anschluss gelöscht und sind damit in jedem Fall in den ZFS snapshots enthalten.

- Asynchrone IMAP-Replikation
  Der IMAP-Server repliziert alle Änderungen am mail-store mit eigenen Mitteln asynchron auf ein System an einem anderen Standort. Die auftretenden typischen Verzögerungen liegen im Sekundenbereich. Dies vervollständigt den Schutz gegen Totalausfall eines Systems bzw. des primären Standortes.

Ähnliche Überlegungen hinsichtlich der Dauer einer vollständigen Wiederherstellung werden von uns für alle kritischen Systeme durchgeführt. Dabei darf auch nicht außer Acht gelassen werden, dass die Verfügbarkeit der Daten nicht notwendiger Weise auch die Verfügbarkeit der Anwendung bedeutet.
Aufgrund der gemachten Erfahrungen sichern wir mit herkömmlichen Methoden nur Systeme mit maximal 15TB und 25 Millionen Dateien. Diese sind in ca. 3-5 Tagen vollständig herstellbar, jedoch ist ein Test im Einzelfall dringend angeraten.

Fazit
Solaris bietet seit Jahren geeignete Hilfsmittel, die die Datensicherheit und Datenintegrität erheblich verbessern können. Neben dem Einsatz derartiger Mittel ist jedoch ein Datenmanagementkonzept und dessen kontinuierliche Fortschreibung und Hinterfragung ein Schlüssel zum Erfolg.

Danksagung:
Mein besonderer Dank für die fortlaufende Unterstützung mit Anregungen, Ideen und Korrekturen gilt meinem Kollegen Dr. Harald Däubler sowie meiner Frau für die Spende zahlreicher Kommas und die Beseitigung sprachlicher Ungereimtheiten.

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