Angst vor dem Datengau? Tipps und Tricks für einen besseren Schlaf

Thomas Nau, Kommunikations- und Informationszentrum (kiz) der Universität Ulm

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

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

Wikipedia definiert den Begriff „Ransomware“ wie folgt *(siehe „https://de.wikipedia.org/wiki/Ransomware“)*: „Ransomware (von englisch „ransom“ für „Lösegeld“), auch Erpressungs-Trojaner, Erpressungs-Software, Krypto-Trojaner oder Verschlüsselungs-Trojaner, sind Schadprogramme, mit deren Hilfe ein Eindringling den Zugriff des Computer-Inhabers auf Daten, deren Nutzung oder auf das ganze Computer-System 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 1980er-Jahren, erlangte aber erst in den letzten Jahren traurige Berühmtheit als Massen-Phänomen.

Sieht man von gezielten Angriffen gegen Personen oder Unternehmen ab, denen oft spezielle Angriffsmethoden zugrunde liegen, so werden die meisten Betroffenen eher zufällig Opfer. Die Einfallstore sind mannigfaltig; E-Mail-Anhänge und Drive-by-Downloads, ausgelöst von aktiven Inhalten und mit automatischer 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 *(siehe „https://www.heise.de/security/meldung/Alles-was-wir-bisher-ueber-den-Petya-NotPetya-Ausbruch-wissen-3757607.html?artikelseite=all")* möglich ebenso wie eine Kombination mehrerer Verbreitungsmethoden.

Klar ist, dass häufig propagierte Schutzmaßnahmen wie 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 zugrunde legt. Daraus lässt sich schließen, dass neben Schutz- und Gegenmaßnahmen vor allem auch organisatorische Maßnahmen dringend angeraten sind. Essenziell sind im allgemeinen Bedrohungsszenario zumindest folgende Punkte:

- Erkennung eines Angriffs
- Existenz einer unabhängigen, schreibgeschützten Datenkopie, im Idealfall inklusive Versionierung
- Kenntnis, Dokumentation und Beschränkung von Zugriffsrechten
- Organisatorische Abwägung und Entscheidung: „ease of use" vs. Sicherheit

Die Dauer, für die schreibgeschützte Kopien der essenziellen Datenbestände vorzuhalten sind, wird maßgeblich durch die Zeit bestimmt, die erforderlich ist, um einen Angriff im schlechtesten Fall erkennen zu können. Diese Erkennung gestaltet sich im Allgemeinen als schwierig. Zwar existieren mittlerweile Lösungen, die das Problem auf Basis von Verhaltens-Analysen adressieren, jedoch ist das Grundproblem der Erkennung dem der Spam-Erkennung ähnlich. Während „brute force"-Angriffe klare Muster liefern und oft sogar der Nutzer direkt zur Kasse gebeten wird, sind subtile Angriffe nur äußerst aufwendig erkennbar; etwa wenn nur wenige Dateien pro Zeit-Intervall verschlüsselt sind und diese darüber hinaus spezifisch nach Typ und Alter ausgewählt werden.

Es darf nicht unerwähnt bleiben, dass die Kenntnis, Dokumentation und Beschränkung von Zugriffsrechten oft stiefmütterlich behandelt werden. Als Beispiel seien Netz-Laufwerke genannt, die oft ganzen Abteilungen als für alle beschreibbare Datenablage dienen und damit auch ein entsprechend höheres Schadenspotenzial mit sich bringen.

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

Listing 1

```
INSTANCE=nfs
HOUR=${INSTANCE}-hourly
DAY=${INSTANCE}-daily

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

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

Listing 2

```
svcadm refresh time-slider:default
svcadm enable  time-slider:default
svcadm enable  auto-snapshot:${HOUR}
svcadm enable  auto-snapshot:${DAY}
```

Listing 3

```
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
```

Listing 4

Hilfe in Form von Solaris-Bordmitteln

Solaris bietet bereits seit Jahren Hilfsmittel an, um die Integrität und Sicherheit von Daten in hohem Maß zu unterstützen. Darren Moffat schreibt darüber bereits im Jahr 2008 in seinem Blog *(siehe „https://blogs.oracle.com/darren/making-files-on-zfs-immutable-even-by-root")* den Artikel „Making files on ZFS Immutable (even by root!)". Grundlage ist dabei ZFS. Durch dessen Verfügbarkeit in weiteren Betriebssystemen, sehr früh schon in FreeBSD und nun stark zunehmend auch in Linux, gilt das nachfolgend Gesagte weitestgehend auch für den Einsatz dieser Betriebssysteme.

Der Einsatz dieser Techniken auf dem Fileserver kann im Falle der Bedrohung durch Ransomware dazu genutzt werden, die dort gespeicherten Daten zu schützen beziehungsweise den Verlust drastisch zu

minimieren. Generell lassen sich die skizzierten Lösungen auch für Datenbank-, Web- oder Mail-Server einsetzen. Endanwender sollten dort jedoch keinen direkten Zugriff auf die Filesysteme erhalten. Damit steht dann die Schnittstelle, die die heutige Ransomware verwendet, nicht zur Verfügung.

```
# 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+h07s…
-rw-r--r--   1 root  root 15256 Sep 24 19:40 AAAAAQAAA+jUjV4I…
```

Listing 5

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

Listing 6

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

Listing 7

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

Listing 8

```
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}
```

Listing 9

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

Listing 10

Solaris Hilfsmittel #1: ZFS-Snapshots

Hinweis: Um die Lesbarkeit zu verbessern, wurde die Ausgabe von Programmen oder deren Quellcode im Folgenden teilweise gekürzt oder umformatiert.

Ergänzend zu lokalen Backup-Strategien bietet ZFS – in der Zwischenzeit auch einige andere Filesysteme – mit seiner Snapshot-Funktionalität eine sehr einfache, Ressourcen-schonende und schnelle Möglichkeit, die genannte Forderung nach schreibgeschützten Kopien von Daten zu bedienen. ZFS nutzt hierzu seine Basis-Technologie „copy-on-write" aus. Die einer Änderung als Basis dienenden Blöcke werden dabei nicht in die Freiliste des Filesystems überführt, sondern bleiben den jeweiligen Snapshots, jetzt nicht mehr veränderbar, zugeordnet. Die Solaris-Service-Management-Facility-Services *(SMF, siehe „http://www.oracle.com/technetwork/articles/servers-storage-admin/intro-smf-basics-s11-1729181.html")* übernehmen die notwendige Steuerung von Erzeugung und Löschung der Snapshots nach vordefinierten Regeln *(siehe Listing 1)*.

In der täglichen Praxis hat es sich bewährt, nicht die Default-Instanzen zu verwenden, sondern für jede Aufgabe eigene Instanzen zu erzeugen. Dies gestaltet sich einfach, wie *Listing 2* zeigt. Die beiden exemplarisch definierten, neuen Services erzeugen stündliche und tägliche Snapshots. Letztere stehen für 28 Tage zur Verfügung, während die stündlichen jeweils die letzten 24 Stunden abdecken.

Nach diesem Rezept lassen sich auch grob- oder feingranularere Snapshots erzeugen. Anfangs sind Debug-Meldungen hilfreich. Sie werden über das Setzen einer SMF-Property eingeschaltet „svccfg -s time-slider setprop daemon/verbose = true" und finden sich dann in der Datei „/var/svc/log/application-time-slider:default.log". Jetzt müssen die Dienste nur noch aktiviert werden *(siehe Listing 3)*. Die dadurch entstehenden Snapshots lassen sich auch aufgrund ihres Namens einfach von anderen, gegebenenfalls manuell erzeugten, unterscheiden *(siehe Listing 4)*.

Um zu verstehen, wie Snapshots gegen typische Ransomware-Probleme helfen, muss man sich vor Augen führen, welche File-Operationen im Zuge der Verschlüsselung auf die Dateien angewendet werden. Vereinfacht werden folgende Schritte pro Datei durchlaufen:

- Öffnen der Quelldatei „S“
- 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
- Verschlüsselung der Daten aus „S“ nach „T“
- Löschen der Quelldatei „S“

Die Bash-Kommandos in *Listing 5* illustrieren dies. Mithilfe des Kommandos „zfs diff“ lassen sich Änderungen des aktuellen Datasets bezüglich eines Snapshots anzeigen. Für das Beispiel in *Listing 6* wurde vor dem Angriff ein Snapshot mit dem Namen „original“ angelegt.

Erkennbar sind 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 betroffenen Dateien sehr gut eingrenzen. Anschließend können die Originaldaten aus dem für jede Datei jeweils neuesten geeigneten Snapshot zurückkopiert werden. Je nach zeitlicher Granularität der Snapshots sowie dem Startzeitpunkt des Angriffs sind dessen Auswirkungen erheblich reduziert oder gar vernachlässigbar.

Bedenken hinsichtlich des Plattenplatzes, der für die Snapshots zusätzlich bereitzustellen ist, lassen sich durch die beiden folgenden Beispiele leicht zerstreuen. Im ersten Fall handelt es sich um einen Mailserver. Die letzten 24 Stunden sind mit stündlichen, die vergangenen drei Monate mit täglichen Snapshots abgedeckt *(siehe Listing 7)*.

Die Snapshots schlagen hier mit rund 50 Prozent des gesamten Bedarfs zu Buche. Dabei ist aber zu bedenken, dass gelöschte E-Mails, darunter auch SPAM, erst nach 90 Tagen wirklich aus dem System verschwinden. Ein Solaris CIFS Server, der rund 200 Arbeitsplätze im administrativen Bereich bedient, ist 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 *(siehe Listing 8)*. In diesem Fall beträgt der zusätzliche Bedarf rund 20 Prozent, ein geringer Aufschlag für den gewonnenen zusätzlichen Schutz.

```
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 privi-
leged
      {appendonly,nounlink}

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

Listing 11

```
root# ppriv -f +D -e chmod S+vappendonly,vnounlink logfile
root# 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}
```

Listing 12

Tools wie ZnapZend *(siehe „http://www.znapzend.org“)* ersetzen nicht nur die von Solaris bereitgestellte Time-Slider-Funktionalität, sondern erweitern diese zum Teil erheblich. Unter anderem erlaubt es ZnapZend, die Snapshots mithilfe der ZFS-Mechanismen „send/receive“ auf andere Systeme zu übertragen. Unterschiedliche Retention-Zeiten für die Snapshots pro System sind ebenfalls unterstützt.

Solaris Hilfsmittel #2: ZFS-Datei-Attribute

Vorab ein Hinweis für alle, die die Beispiele nachvollziehen möchten. Die mit Solaris ausgelieferten GNU-Utilities unterstützen die zusätzlichen Funktionalitäten nicht. Daher muss „/usr/bin“ zwingend im Pfad vor „/usr/gnu/bin“ stehen. Außerdem sind gegebenenfalls Shell-Aliasse für Kommandos zu prüfen, falls das Ergebnis nicht dem Erwarteten entspricht; Analoges gilt für MANPATH.

ZFS-Datei-Attribute sind im Falle von Ransomware weniger effektiv, da sie im Allgemeinen auch die tägliche Arbeit zu sehr einschränken. In speziellen Fällen, etwa bei überwiegend statischen Dateien, sind sie jedoch extrem hilfreich. So lässt sich mit ihrer Hilfe die Manipulation von Web-Server-Konfigurationen, PHP-Installationen sowie der zugehörigen Log-Dateien zuverlässig unterbinden, sofern diese Dienste nicht in Immutable-Zones *(siehe „https://docs.oracle.com/cd/E53394_01/html/E54762/glglv.html“)* laufen können. Einige der Attribute sind auch in der Lage, „root“ vom Zugriff auszuschließen, zumindest ohne vorherigen Eingriff. Daher ist ein Test der Gesamt-Funktionalität nach Aktivierung dieser Schutzmaßnahmen dringend angeraten. Um mögliche Verschleierungen zu verhindern, sollten Log-Dateien im Allgemeinen nur im Sinne eines Anhängens neuer Daten veränderbar sein *(siehe Listing 9)*. Für den geplanten Einsatzzweck werden die Attribute „appendonly“ und „nounlink“ gesetzt, die dann gemeinsam sowohl das Löschen als auch das Überschreiben der Datei verhindern *(siehe Listing 10)*.

Offensichtlich erlaubt Solaris es einem normalen Account nicht, diese Attribute zu verändern. Warum? Um dies herauszufinden, kommen die Debugging-Möglichkeiten des Least-Priviledge-Systems *(siehe „https://docs.oracle.com/cd/E53394_01/html/E54830/prbac-2.html#scrolltoc“)* zum Einsatz *(siehe Listing 11)*.

Das Problem lässt sich auf zwei Arten lösen. Zum einen könnten dem Account die notwendigen zusätzlichen Privilegien zugewiesen werden, was aber mit einer Schwächung des Schutzes einhergeht. Daher ist es sinnvoller, die Attribute durch einen anderen – privilegierten – Account passend zu setzen *(siehe Listing 12)*. Die Wirkung ist die erwünschte, selbst mit Root-Rechten *(siehe Listing 13)*.

Im Falle von Software-Installationen und insbesondere von Konfigurationsdateien, hier beispielsweise die eines Apache-Webservers, ist das Setzen des „immutable"-Attributs hilfreich. Auch dazu sind passende Privilegien notwendig *(siehe Listing 14)*. Um Anpassungen an der Konfiguration vornehmen zu können, muss die Datei also zuerst entsperrt sein *(siehe Listing 15)*.

Daten-Management

Neben technischen Aspekten sind auch die organisatorischen Belange des Daten-Managements von größter Bedeutung. In diesem Umfeld existieren jedoch auch einige Legenden, die über die letzten Jahrzehnte mehr oder weniger gepflegt wurden, ohne sie jemals auf den Prüfstand aktueller Technologien zu stellen.

Nach Erfahrung des Autors sind im Datenbank-Umfeld viel häufiger Sicherungs- und auch Recovery-Szenarien implementiert und dokumentiert, als dies im Bereich von File-, Web- oder Mail-Servern der Fall ist. Regelmäßige realitätsnahe Tests und damit die Überprüfung, ob die Anforderungen an Ausfallzeiten beziehungsweise deren Folgen tragbar sind, finden jedoch in nur wenigen Fällen statt. Das bringt uns insbesondere hinsichtlich eingesetzter Backupstrategien zu einigen Mythen, die sicherlich nicht nur im Hochschulumfeld anzutreffen sind:

Mythos #1: „Nutzer kennen ihre Daten sowie die sich daraus ergebenden Anforderungen und folgen einem Daten-Management-Konzept." System-Verwaltern ist hier meist kein Vorwurf zu machen, vielmehr scheitert es oft an fehlenden organisatorischen Vorgaben und Entscheidungen. Zu den zu beantwortenden Fragen gehören unter anderem:

- *Welche Daten sind heiß/wichtig und wie oft wird diese Frage neu beantwortet?*
  Setzen von Prioritäten bei Wiederherstellung
- *Liegen Daten strukturiert/chaotisch vor (Spotlight-Problem)?*
  Einfache Auswahl und Zeitgewinn bei Wiederherstellung
- *Wie lange kann eine Gruppe/Einrichtung ohne Daten produktiv arbeiten?*
  Entscheidet über Technologie

Mythos #2: „Die Netzwerk-Bandbreite ist der begrenzende Faktor, Clients brauchen 10 GE!" Mag dies in Einzelfällen zutreffen – vor allem bei mobilen Endgeräten –, so beschränkt im täglichen Betrieb im Allgemeinen die Suche im zugrunde liegenden File- und Speicher-System die Geschwindigkeit des Backups. Nach Erfahrungen des Autors liegen die durchschnittlichen Bandbreiten-Anforderungen inkrementeller Backups deutlich unter dem, was ein 1-Gb/s-Interface zu liefern vermag. Ein wichtiger Rückschluss daraus ist, dass auch im Falle eines Full-Restore in den allermeisten Fällen das Speicher- und File-System der wesentliche zeitbestimmende Faktor ist.

Mythos #3: „Ein Archiv ist doch nur Backup mit langen Lagerzeiten." Neben einem grundlegend anderen Verständnis von Bibliothekaren hinsichtlich der Eigenschaften eines Archivs existieren auch im IT-Umfeld klare Abgrenzungen zu Backups: Sind Archivdaten mit Metadaten angereichert, vereinfacht das die Suche erheblich, und werden Archivdaten vom Quellsystem gelöscht, sind sie damit nach einiger Zeit auch im Backup gelöscht. Kurz gesagt wird oft von „Archivdaten" gesprochen, aber in Wirklichkeit sind „kalte Daten" gemeint *(siehe Mythos #1)*.

Mythos #4: „Es besteht ein Plan für den Fall der Fälle." Die Existenz eines derartigen Planes ist zwar eine notwendige, jedoch keine hinreichende Voraussetzung. Vielmehr bedarf jeder Plan regelmäßiger und vor allem realitätsnaher Tests, die sämtliche Schritte abdecken beziehungsweise simulieren:

- Zusammenstellung eines Teams mit allem notwendigen Wissen
- System (beschaffen und) wiederherstellen
- Daten wiederherstellen
- Anwendung mit restaurierten Daten bekannt machen, also Transaction-Logs einspielen etc.

```
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 APPENDED_LINE >> logfile

root# rm logfile
rm: remove logfile (yes/no)? y
rm: logfile not removed: Not owner
```

Listing 13

```
root# chmod S+vimmutable httpd.conf

root# > httpd.conf
-bash: httpd.conf: Not owner

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

Listing 14

```
root# chmod S-vimmutable httpd.conf
root# vi httpd.conf
root# chmod S+vimmutable httpd.conf
```

Listing 15

Besonderes Augenmerk sollte unter anderem auf folgende kritische Punkte gelegt werden:

- Schlüssel-Management im Falle verschlüsselter Daten
- Priorisierung der Daten für die Wiederherstellung
- Die Zeiten, die für die Wiederherstellung einzelner Bereiche notwendig sind

Aus den aufgeworfenen Fragen und Kritikpunkten wird klar, dass im Falle von Datensicherung – aber viel mehr noch beim Daten-Management – organisatorische Entscheidungen von grundlegender Bedeutung sind, es sich also mitnichten um ein rein technisches Problem handelt.

Paradigmen und Aussagen kritisch zu hinterfragen, ist der Schlüssel zum Erfolg und hat erhebliche Auswirkungen auf die notwendigen Leistungsmerkmale künftiger Lösungen, da diese immer spezifischer in einzelne Bereiche passen müssen. Ein Beispiel ist die eingangs diskutierte Problematik von Ransomware oder anderen Angriffen. Diese erfordern spezielle und über die übliche Datensicherung hinausgehende Maßnahmen, stellen jedoch, wie das nachfolgende Beispiel zeigt, keinen Einzelfall dar.

Wo ein herkömmliches Backup keinen Sinn ergibt

E-Mail-Systeme sind heute oft genug als zentrales Kommunikationsmittel kleiner und großer Organisationen eines der kritischsten IT-Elemente, bei denen selbst kurze Ausfälle wenig Toleranz finden. Je nach technischer Implementierung kann das System hundert Millionen Dateien oder mehr enthalten, deren initiale Sicherung mit herkömmlichen Backup-Methoden mehr als zwei Wochen dauern würde. Entscheidend dafür ist nicht die Summe der Größe der Daten – es sind die Random-IO-Zugriffe auf das Filesystem. Dieses liegt häufig, wie auch in diesem Fall, auf herkömmlichen SAN-Systemen.

Das Ergebnis legt den Schluss nahe, dass eine vollständige Wiederherstellung ebenfalls wenigstens 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. Eine vollständige Migration auf SSD-basierte Speicher kommt aus wirtschaftlichen Gründen nicht in Betracht. Nachfolgend die Skizze einer Lösung, 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 bezüglich der versehentlichen Löschung von Daten erheblich gegenüber einer normalen, täglichen Datensicherung.

- *Cyrus-IMAP-Server-Feature „delayed expunge“*
  E-Mails werden für 48 Stunden nur als gelöscht markiert und sind für den E-Mail-Nutzer 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 des primären Standorts.

Überlegungen hinsichtlich der Dauer einer vollständigen Wiederherstellung erfolgen für alle kritischen Systeme. Dabei wird auch nicht außer Acht gelassen, dass die Verfügbarkeit der Daten nicht notwendiger Weise auch die Verfügbarkeit der Anwendung beziehungsweise Dienstleistung bedeutet.

Fazit

Solaris bietet seit Jahren geeignete Hilfsmittel, um die Datensicherheit und -integrität erheblich zu verbessern. Neben dem Einsatz derartiger Mittel ist jedoch ein Daten-Managementkonzept sowie dessen kontinuierliche Fortschreibung und Hinterfragung ein Schlüssel zum Erfolg.

Thomas Nau
thomas.nau@uni-ulm.de