Studienprogramm Sachbearbeiter:in Digitale Forensik
Praktikum Betriebssysteme
Dozenten: Tim Wetterau
wetterau@hs-mittweida.de
Stand: 26.06.2023

HOCHSCHULE MITTWEIDA
University of Applied Sciences

Betriebssysteme

Praktikum 3 - RAID II und Scripting

Dieses Praktikum knüpft an das vorherige Praktikum mit der Sicherung von RAID-System an. In diesem Praktikum schauen wir uns an, wie wir Daten von einem RAID-System sichern können und wie wir diese händisch anschauen können. Danach wird das Skripting in der Bash thematisiert anhand eines kleinen Skripts zum Backup von Verzeichnissen. Dieses binden wir beispielhaft in der Crontab ein zur Automatisierung.

Inhalte des Praktikums:

- Sichern von RAID-Systemen
- Erstellen von Archiven mittel **`tar`**
- Erstellen eines Backup Skripts

Sichern von RAID-Systemen

RAID-Systeme, beispielsweise in Form von NAS-Systemen, sind eine großartige Sache. Für die Anwender. Nicht für Forensiker. Oftmals ergeben sich aus der Sicherung von NAS-Systemen verschiedene Probleme, je nach deren Größe und Betriebszustand.

Überlegen Sie sich an dieser Stelle mögliche Probleme, welche beim Sichern von RAID-Systemen auftreten können.

- **Sehr große Festplatten**
  - **Problem Speicherplatz**
  - **Problem Zeit**
- **Unbekannte RAID-Konfiguration (RAID-Level und Reihenfolge der Platten)**
- **RAID möglicherweise nicht mehr funktionsfähig**
  - **defekte Festplatte im Verbund**
  - **beschädigter RAID-Header/Controller**
- **produzieren je nach Größe viele Daten, die ausgewertet werden müssen.**
  - **Sind die gesicherten Daten letztlich von forensischem Wert?**

Je nach dem, in welchem Zustand sich das RAID befindet, muss entschieden werden, wie der Verbund zu sichern ist. Möglich ist die Sicherung eines RAIDs im Ganzen, also unter Nutzung des funktionsfähigen RAID-Controllers. Wenn nicht anders möglich, muss ein RAID manchmal auch Festplatte für Festplatte gesichert werden und danach wieder zusammengesetzt werden. Wir wollen uns in diesem Teil exemplarisch am vorhandenen RAID anschauen, wie solch eine Sicherung aussehen kann.

Können wir das RAID im Ganzen sichern, ist das Vorgehen gleich zu dem, wie das Sichern einer einzelnen Festplatte, welches wir im Praktikum Digitale Forensik kennengelernt haben. Dazu hängen wir unser RAID einfach aus dem System aus und nutzen **`dd`** zum Erstellen eines Abbildes. *Nutzen Sie die Unterlagen aus dem Praktikum und sichern Sie den zusammengesetzten RAID-Verbund unter* **`~/images/md0.raw`**.

```
$ dd if=<Quelldatei> of=<Zieldatei>
```

Wir wollen an der Stelle versuchen die Daten aus dem Verbund auf den einzelnen Datenträgern wiederzufinden. Um die Zusammensetzung zu prüfen, laden wir eine große Datei herunter, die sich über mehrere Blöcke des Verbundes erstrecken. *Machen Sie zuerst den Datenträger für alle Nutzer schreibbar. Dann laden wir uns den Text der*

Bibel mit dem folgenden Befehl auf das RAID herunter, entpacken das Archiv anschließend und verschieben die Datei auf das RAID:

```
$ sudo chmod -cR 777 /mnt/raid0
$ wget http://truth.info/bigfiles/bible.txt.zip
$ unzip bible.txt.zip
$ cat bible.txt >> /mnt/raid0/bible.txt
```

Schauen wir uns den Inhalt der Bibel mal an. Dafür brauchen wir einen Hexeditor. Beispielsweise können wir **hexedit** verwenden. *Installieren Sie* **hexedit** *und starten sie es danach mit dem Aufruf von* **~/images/md0.raw**. *Wechseln Sie mittels der Tabulator-Taste in den ASCII-Modus und drücken Sie „***Strg + S***" und geben Sie anschließend „Genesis 1:1". Merken Sie sich den Offset des Anfangs des Bibeltextes. „***Strg + S***" ermöglicht es uns nach Hex oder ASCII-Werten zu suchen.*

```
$ sudo apt install hexedit
$ sudo hexedit /dev/md0
$ TAB
$ Strg + S
$ Genesis 1:1
```

Gehen Sie kurz durch den Text. Sie müssten feststellen, dass Sie diesen am Stück lesen können. Das Ablegen der Daten obliegt, wie erwähnt, dem RAID-Controller. Dadurch ist es möglich, dass dieser auch Daten verteilt ablegt und diese nicht am Stück gelesen werden können. Zur Sicherung der einzelnen Festplatten müssen wir zuerst das RAID aushängen und anhalten. *Hängen Sie zuerst das RAID aus dem Mountpoint (***/mnt/raid0***) aus und nutzen Sie dann die folgende Syntax, um das RAID anzuhalten und überprüfen Sie den Erfolg ihres Befehls:*

```
$ sudo mdadm --stop <RAID-Verbund>
$ sudo mdadm --stop /dev/md0
```

Anschließend liegen die Festplatten wieder einzeln vor. Auch hier nutzen wir **dd**, um die Festplatten zu sichern. Wir könnten der Integrität wegen auch **ewfacquire** nutzen und anschließend diese in ein RAW-Format umwandeln. An der Stelle konzentrieren wir uns aber auf die Sicherung per **dd**. *Nutzen Sie* **dd***, um die drei Festplatten einzeln als Image zu sichern*

```
$ mkdir ~/images
$ dd if=/dev/sdc of=~/images/raid-disk1.raw
$ dd if=/dev/sdd of=~/images/raid-disk2.raw
$ dd if=/dev/sde of=~/images/raid-disk3.raw
```

Nachdem wir die Festplatten gesichert haben, können wir auch den RAID-Verbund auch schon wieder zusammensetzen. Auch dazu bietet sich **mdadm** an. Dazu können wir die Image-Dateien als Loopdevice dem System zur Verfügung stellen. Wir nutzen **losetup** mit folgender Syntax. Es sei ein Beispiel für das Image der RAID-Disk 1 gegeben:

```
$ sudo losetup /dev/loop[X] <Imagedatei>
$ sudo losetup /dev/loop1 ~/raid-disk1.raw
```

Stellen Sie ein Image nach dem anderen als Loopdevice zur Verfügung und überprüfen Sie zwischendurch immer wieder mittels ***lsblk****. Was stellen Sie nach dem Erstellen des letzten Loopdevices fest? Warum ist das so? Machen Sie die Informationen, welche diese Operation ermöglichen in der Ausgabe von* ***mdadm --examine*** *aus.*

```
$ sudo losetup /dev/loop1 ~/images/raid-disk1.raw
```

```
$ sudo losetup /dev/loop2 ~/images/raid-disk2.raw
$ sudo losetup /dev/loop3 ~/images/raid-disk3.raw
$ mdadm --examine raid-disk[1,2,3].raw
$ Magic
$ Array UUID
$ Raid Level
$ Raid Devices
$ Device Role
```

Wir haben gesehen, dass **mdadm** auch in der Lage ist, automatisch ohne unser Zutun ein RAID anhand der im Superblock vorhandenen Informationen zusammenzusetzen kann. Wären diese Superblocks nicht verfügbar, muss der Forensiker selbst versuchen die Daten wieder zusammenzusetzen. X-Ways bietet unter anderen die Möglichkeit ein RAID je nach dessen Level wieder zusammensetzen. Das bietet sich eben an, wenn die Datenträger eines NAS einzeln gesichert werden müssen, was ohnehin forensisch immer sauberer ist, und wieder zusammengesetzt werden sollen. Mittels X-Ways kann das zusammengesetzte Abbild als ein Image behandelt werden.

Erstellen von TAR-Archiven

Das Kommando **tar** stammt aus der Urzeit der Linux Systeme, in welcher zum Archivieren Bandlaufwerke verwendet wurden. Das kann auch in dem ausgeschriebenen Namen Tape Archiver abgeleitet werden. Nach wie vor ist das Tool aber sehr beliebt, da dessen Funktionsumfang groß ist und dessen Handhabung einfach. Im Gegensatz zum dem unter Windows verwendeten ZIP-Format, kann ein **tar**-Archiv auch die Nutzerrechte von Nutzern an einer Datei mitspeichern und beim Entpacken wiederherstellen.

tar funktioniert so, dass es alle angegebenen Dateien sequenziell in ein Archiv verpackt. Es können auch neue Dateien zu einem Archiv hinzugefügt werden, was es ermöglicht, Dateien mit dem gleichen Namen in ein Archiv zu speichern. Sie unterscheiden sich dann anhand der Änderungszeit bzw. anhand der Dateiversion. Der Aufruf des Tools funktioniert über das Kommando:

```
$ tar [Options] [<Datei> <Archiv> <Verzeichnis>]
```

Dabei kann ein Archiv über folgende Kommandos gepackt, angesehen und wieder extrahiert werden:

```
$ tar -cvf <Archiv> <Datei1> [<Datei2> … ]     # Anlegen eines neuen Archivs
$ tar -tvf <Archiv>                            # Anzeigen des Inhaltes des Archivs
$ tar -xvf <Archiv>                            # Extrahieren des Archivs
```

Mit dem letzten Befehl können außerdem Daten hinzugefügt werden:

```
$ tar -uf <Archiv> <neue Dateien>
```

Mittels **tar** können die Daten nicht nur zusammen in ein Archiv gepackt werden. Das Tool ermöglicht es auch eine Komprimierung per **gzip2** anzuwenden, womit die Größe des Archivs erheblich verringert werden kann. Durch den folgenden Befehl kann eine Komprimierung nach dem Anlegen des Archivs stattfinden:

```
$ tar -czvf <Archiv> <Datei>
```

Sie haben im vorherigen Abschnitt die Abbilder des RAIDs gesichert. Die Archivierung der gesicherten Asservate und deren Kopien stellt auch Forensiker teils vor Probleme aufgrund von fehlendem Speicherplatz. Daher bietet es sich an die Daten zu packen und zusätzlich zu komprimieren.

*Nutzen Sie das Tool **tar**, um aus den drei erstellten Abbildern ein Archiv mit dem Namen images.zip zu erstellen. Komprimieren Sie das Archiv mittels **gzip2**!*

```
$ tar -czvf Images.zip ~/raid-disk[123].raw
$ tar -czvf Images.zip ~/raid-disk1.raw
$ tar -czvf Images.zip ~/raid-disk2.raw
$ tar -czvf Images.zip ~/raid-disk3.raw
```

Schauen Sie sich den Erfolg ihres Befehls im Dateisystemen an und lassen Sie sich außerdem den Inhalt des erstellten Tar-Archivs anzeigen. Was stellen Sie fest? Wie sieht die Ausgabe aus? Kommt das Ihnen bekannt vor?

```
$ ls -l
$ tar -tvf images.zip
$ gleiche Ausgabe wie bei ls -l → sehen auch die Berechtigungen
```

Legen Sie eine neue Textdatei an mit einem Inhalt ihrer Wahl und fügen Sie diese dem Archiv hinzu. Überprüfen Sie den Erfolg erneut. Hat alles funktioniert, wie erwartet?

```
$ cat testtext > test.txt
$ tar -uf images.zip test.txt
$ nein, funktioniert nicht, da komprimierten Archiven keine Daten hinzugefügt werden könnnen
```

Erstellen Sie anschließend ein neues Verzeichnis und kopieren Sie das Archiv in dieses hinein. Extrahieren Sie das Archiv dort und überprüfen Sie Ihren Erfolg.

```
$ mkdir <neues Verzeichnis>
$ cp images.zip <neues Verzeichnis>
$ cd <neues Verzeichnis>
$ tar -xvf images.tar
$ ls -l
```

Erstellen eines Backupskripts

Der im vorherigen Kapitel gelernte Umgang mit dem Tool **tar** kann sehr sinnvoll sein bei der Automatisierung von Backups. Ein Beispiel dafür ist z.B. ein Verzeichnis für Daten einer Serveranwendung, welche ständigen Änderungen unterliegen und möglichst unbeschadet sein sollen. Ziel ist es, die Daten aus diesem Ordner so zu sichern, dass täglich ein Backup des gesamten Ordners erstellt wird. Damit soll erreicht werden, dass bei einer möglichen Beschädigung der Daten der Datenbestand des Tages davor wiederhergestellt werden kann. Sinnvoll sind Backups immer nur auf Datenträger oder Orten, an denen die Originaldaten nicht liegen. Am besten sollten Backups auch einmal komplett offline gelagert werden, um zumindest ältere Datenbestände zu schützen. Zum Thema Backups können Sie sich gern auf dieser Webseite belesen:

$ https://www.it-experte-augsburg.de/effektive-backups-dank-der-3-2-1-1-0-regel/

Wir wollen den Gedanken des Backups eines Verzeichnisses in Form eines kleinen Skripts auf unserer VM umsetzen. Im Sinne der Sicherheit sollten auch wir dies auf einem anderen Datenträger vornehmen als unserer Hauptfestplatte. Dazu haben wir aber ein RAID-System angelegt, welches wir als Backup-Location nutzen wollen.

Wenn Sie nichts anderes in den Praktika unternommen haben, sollte das RAID nach wie vor als Blockdevice an Ihrer VM vorliegen. *Sollte dem nicht so sein, binden Sie es bitte wieder so ein, dass Sie lesenden und schreibenden Zugriff auf das RAID haben.*

Erstellen Sie nun auf dem RAID ein neues Verzeichnis mit dem Namen „apache-backup“. Das ist unsere neues Backup-Ziel, in welchem wir unsere Backups anlegen wollen.

```
$ cd /mnt/raid0
$ mkdir apache-backup
```

Im Falle eines Apache Webservers ist es sinnvoll dessen Konfiguration als auch die Webseiten mit zu sichern, welche betrieben werden. Dabei sind folgende Pfade zu beachten:

- **/etc/apache2/**
- **/var/www/html/**

Aufgabe: *Schreiben ein **Bash-Skript**, welches ein **Backup der genannten Ordner** des Apache2-Servers erstellt und beide Ordner in einem **Tar-Archiv** zusammenfasst. Dieses Archiv soll zusätzlich mittels **gzip2** komprimiert werden, um Platz zu sparen. Ebenso soll jedes Archiv mit dem **aktuellen Datum** versehen werden. Nutzen Sie dazu das Format „**YYYY-MM-DD_hh-mm**“. Letztlich soll jedes Verzeichnis einen **Dateinamen nach folgender Vorgabe** erhalten: „backup_YYYY-MM-DD_hh:mm.zip“. Zur Absicherung des Programmablaufs soll vor der Ausführung des Backups **geprüft werden**, ob **beide Verzeichnisse** vorhanden sind. Wenn nicht wird das Programm mit dem **Exit-Code 1** beendet. Wenn das Zielverzeichnis nicht vorhanden ist, dann soll es entsprechend erstellt werden.*

In diesem Fall soll eine statische Lösung aber dafür auch sehr einfache Lösung betrachtet werden. Dazu werden die Pfade der Verzeichnisse hart im Quellcode definiert und später als Variable abgerufen. Zuerst müssen wir aber wie in jedem Skript die Shebang definieren. Dafür setzen wir an erste Stelle des Skripts die Zeile:

```
# #!/bin/bash
```

Wie bereits erwähnt, wollen wir für diese einfache Lösung die Pfade fest im Quellcode verankern. Dafür setzen wir entsprechende Variablen:

```
# targetdir=/mnt/raid0/apache_backup
# dir1=/etc/apache
# dir2=/var/www/html
```

Passen Sie bitte gegebenenfalls die Pfade so an, dass Sie auf ihr System passen, je nachdem, wie Sie das System in den vorherigen Praktika eingerichtet haben.

Es ist ebenso sinnvoll den Dateinamen des Zielarchivs schon als Variable zu definieren. Dafür brauchen wir den Befehl **date**. *An dieser Stelle sind Sie aber gefragt: Konstruieren Sie den Befehl mit **date** so, dass das Datum in der gewünschten Form ausgegeben wird.* Anschließend konkatenieren wir die restlichen Teile des Dateinamens in eine Variable **filename**:

```
# filename="backup_$(<date-Befehl>).zip"
```

Denken Sie daran, dass Sie den Formatierungsstring im **date**-Befehl mit einfachen Hochkommata versehen und den String für die Variable mit doppelten Hochkommata!!!

Nachdem wir nun die Vorarbeiten abgeschlossen haben, können wir uns um den eigentlichen Programmablauf kümmern. Dazu müssen zuerst, wie verlangt, prüfen, ob die beiden Verzeichnisse des Apache-Servers existieren. Dazu verwenden wir eine If-Verzweigung, in der wir prüfen, ob das Verzeichnis **dir1** und das Verzeichnis **dir2** vorhanden sind:

```
# if [[ -d $dir1 && -d $dir2 ]]
```

```
# then
#   <Anweisungsblock 1>
# else
#   <Anweisungsblock 2>
# fi
```

Achten Sie bei den Klammerausdrücken auf die korrekte Setzung der Leerzeichen, da sonst syntaktische Fehler entstehen, die von der Bash nicht interpretiert werden können. Sinngemäß muss im Anweisungsblock 1 alles ausgeführt werden, was das eigentliche Backup erstellt. Im Anweisungsblock 2, was passiert, wenn die beiden Verzeichnisse nicht vorhanden sind.

Eine weitere Bedingung, welche das Skript erfüllen soll, ist die Überprüfung der Existenz des Zielverzeichnisses. Dafür nutzen wir die gleiche Syntax, wie bei der vorigen If-Verzweigung nur ohne den Else-Zweig, da dieser nicht gebraucht wird. Wenn das Verzeichnis nicht existiert, dann soll es einfach erstellt werden. Also setzen wir für **<Anweisungsblock 1>** ein:

```
# if [[ ! -d $targetdir ]]
# then
#   mkdir $targetdir
# fi
```

Anschließend bleibt an der Stelle nur noch, dass wir in das Zielverzeichnis wechseln und darin das Archiv erstellen. Dafür nutzen wir folgende Befehle nach der gerade geschriebenen If-Verzweigung:

```
# cd $targetdir
# tar -czvf $filename $dir1 $dir2
```

Es bleibt noch, dass bei der Nichtexistenz der beiden Verzeichnisse ein Fehlercode geworfen wird. Dazu schreiben an Stelle von **<Anweisungsblock 2>** eine **exit**-Anweisung und zur kosmetischen Ausgestaltung eine kleine Fehlernachricht:

```
# echo Verzeichnisse nicht vorhanden
# exit 1
```

Damit haben wir unser kleines Skript auch schon beendet. Test Sie nun das Skript auf seine Funktion und erstellen Sie über den Aufruf ihres Skripts ein Backup des Apache-Servers auf ihrem RAID-System. Denken Sie daran das Skript ausführbar zu machen.

Automatisierung mittels Crontab

Zuletzt wollen wir das Skript automatisieren, sodass wir uns um dessen regelmäßige Ausführung nicht mehr kümmern müssen. Dazu nutzen wir den zeitgesteuerten Cron, der standardmäßig auf Linux aktiv ist. um die Konfiguration des Dienstes aufzurufen, nutzen wir den Befehl:

```
$ crontab -e
```

Anschließend wird der Nutzer mit welchem Editor die Konfiguration bearbeiten möchte, wenn er die Konfiguration das erste Mal aufruft. Hier wählen wir einfach das bekannte **nano** aus. Wie sie aus der Vorlesung wissen, können wir nun Zeiten eintragen, an den der hinten stehende Befehl ausgeführt werden soll. Wir sollten hier die Eintragung so definieren, dass Sie ziemlich zeitnah ausgeführt wird, damit wir die Funktionstüchtigkeit überprüfen können. Beispieleintragungen könnten so aussehen: (Trennung durch Leerzeichen)

```
# m h dom mon dow command
30 2 * * * <command>          <command> wird jeden Tag um 2:30 Uhr ausgeführt
0 0 1 * * <command>           <command> wird jeden Monat am 1. um 0:00 Uhr ausgeführt
*/5 * * * * <command>         <command> wird alle 5 Minuten ausgeführt
0 0 * * 1 <command>           <command> wird jeden Montag um 0:00 Uhr ausgeführt
```

Dabei haben die Abkürzungen die folgende Bedeutung:

- **m** Minute
- **h** Stunde
- **dom** Tag des Monats (day of month)
- **mon** Monat
- **dow** Tag der Woche (day of week)

Das ursprüngliche Ziel war, dass das Skript, welches Sie vorhin geschrieben haben, einmal am Tag ausgeführt wird. Dazu wollen wir einen Eintrag in der Crontab anlegen. Orientieren Sie sich für den Eintrag an der aktuellen Zeit in der VM und rechnen Sie einfach 3 Minuten drauf, damit Sie genug Zeit zum Schreiben und Speichern haben.

Öffnen Sie, wenn Sie es nicht schon getan haben, die Crontab-Konfiguration auf und erstellen Sie einen Eintrag, sodass das Skript einmal am Tag ausgeführt wird. Bedenken Sie, dass die den kompletten Pfad bis zum Skript angeben müssen und dass dieses auch ausführbar ist. Überprüfen Sie den Erfolg im Zielverzeichnis des Backupskripts.

```
$ crontab -e
$ angenommene Zeit: 14:30 Uhr
$ Eintrag: 33 14 * * * /home/praktikus/apache_backup.sh
```