Studienprogramm Sachbearbeiter:in Digitale Forensik
Praktikum Digitale Forensik Grundlagen
Dozenten: Tim Wetterau | Leander Hossfeld
wetterau@hs-mittweida.de | hossfeld@hs-mittweida.de
Stand: 29.03.2023

HOCHSCHULE MITTWEIDA
University of Applied Sciences

Digitale Forensik

Teil 3 - Forensische Praxis

In diesem Praktikum sollen Sie einmal exemplarisch eine forensische Anwendung im Bereich von Linux ausprobieren. Nachdem Sie in den vorherigen Praktika schon einige Grundlagen in Linux kennengelernt haben, sollten Sie nun so weit vertraut sein, dass Sie sich einen Großteil der Befehle selbst erarbeiten können. Wir gehen neben der Vorlesung auch kurz auf die Theorie der Verfahren ein, um Ihnen eine kleine Wiederholung und ggf. auch Erweiterung des bisherigen Wissens zu geben.

Vorbereitung

Wir gehen davon aus, dass die virtuelle Maschine mit Linux Mint auf ihrem System in VirtualBox oder einer anderen Virtualisierungsumgebung läuft. Des Weiteren müssen Sie zur Durchführung dieses Praktikums einen virtuellen Datenträger herunterladen, der das benötigte Material enthält, welches es zu untersuchen gilt.

Ab hier sind auch die fortgeschrittenen Nutzer dazu angehalten, das Praktikum durchzuführen und somit einen Einstieg in das forensische Vorgehen vermittelt zu bekommen.

Forensische Datensicherung

Jede digital-forensische Untersuchung ist zu Beginn durch die forensische Datensicherung geprägt. Dieser Schritt ist in der Regel mit der Erstellung eines korrekten forensischen Abbildes, auch Image genannt, verbunden. Dabei ist dieser Schritt nicht unproblematisch, da jede Nachlässigkeit zu Manipulationsverdacht führen und nicht mehr rückgängig gemacht werden kann. Dies würde im schlimmsten Falle bedeuten, dass die forensische Datensicherung nicht mehr gerichtsverwertbar ist. Der Eigenschaft **Integrität einer Datensicherung** ist somit eine essenzielle Rolle in der digital-forensischen Untersuchung zuzuschreiben. Wie die Integrität durch beispielsweise Hashwerte sichergestellt werden kann, erfahren Sie im Kapitel **„Hashing“**[1].

In der IT-Forensik unterscheiden wir grundsätzlich zwischen zwei Untersuchungsformen in Bezug auf Datensicherungen: a.) Post-mortem-Analyse und b.) Live-Forensik. Die **Post-mortem-Analyse**, auch Offline-Forensik genannt, kommt zum Einsatz, wenn ein Vorfall nachträglich aufgeklärt wird. In der Regel geschieht durch die Untersuchung von Datenträgerabbildern auf persistente, also nichtflüchtige Spuren. Die Gewinnung und Analyse von gelöschten, umbenannten sowie anderweitig verschlüsselten und versteckten Dateien aus Massenspeichern sind dabei die wichtigsten Untersuchungsinhalte. Um die Inhalte für die Analyse zu gewinnen, wird bei der Post-mortem-Analyse eine **physikalische Datensicherung** durchgeführt. In der Regel wird dafür jeder einzelne Sektor des Datenträgers gelesen und gesichert. Dies betrifft somit auch Bereiche, die vom Betriebssystem als „frei“ angesehen werden. In Bezug auf die Integrität erhält der Forensiker somit eine 1:1 Kopie des Datenträgers, an welchem er diverse Untersuchungsschritte ausführen kann, ohne das Original zu verändern [1].

Die **Live-Forensik**, auch Online-Forensik genannt, umfasst hingegen, ableitbar aus ihrer Namensgebung, Untersuchungsmethodiken, die man bereits während der Laufzeit des zu sichernden Gerätes vornimmt. Im Vordergrund stehen hier vor allem nicht persistente, also flüchtige Daten, die gewonnen und anschließend ausgewertet werden können. Flüchtige Daten können beispielsweise Informationen über bestehende

[1] https://monami.hs-mittweida.de/frontdoor/deliver/index/docId/13529/file/BA_52212_Leander-Hossfeld_geschwaerzt.pdf (Zugegriffen am: 27.03.2023)

Netzwerkverbindungen, gestartete Prozesse sowie Speicherinhalte aus dem Arbeitsspeicher sein. Um flüchtige Daten zu gewinnen, wird in der Live-Forensik eine **logische Datensicherung** durchgeführt. Die Datensicherung geschieht hier im Gegensatz zur physikalischen immer unter Zuhilfenahme des aktiven Dateisystem, also direkt aus dem laufenden System. Für den IT-Forensiker sind vor allem jene Daten interessant, welche nur während einer offenen Sitzung erhoben werden können, da sie andernfalls nicht zugänglich wären. Zu solchen Daten gehören beispielsweise: Cloud Speicher, Messenger Inhalte, Inhalte von Datenbanken, offene verschlüsselte Bereiche sowie Online E-Mail-Postfächer [1].

Für beide Untersuchungsformen existieren eine Reihe an forensischen Sicherungsstationen und Tools. Einige Beispiele sind:

- Forensic Falcon-NEO (https://www.logicube.com/shop/forensic-falcon-neo/)
- FTK-Imager (https://www.exterro.com/ftk-imager)
- ewfaquire (https://github.com/libyal/libewf)
- PALADIN (https://sumuri.com/software/paladin/)

Hashing

Wie bereits erwähnt, muss zwangsläufig die Integrität der sichergestellten Daten im Sinne der *Chain of Custody* erfüllt werden. Eine Möglichkeit, die sich in der IT-Forensik etabliert hat, nachträgliche Manipulationen an digitalen Beweismitteln zu erkennen, ist die zu sichernden Daten mittels eines Hashwertes inhaltlich überprüfbar zu machen (***Hashing***). Im Datensicherungsprozess spielen die dafür verwendeten **kryptografischen Hashfunktionen** eine zentrale Rolle. Als Ergebnis der Berechnung erhält man einen **Hashwert**, der mit einer Art digitalen Fingerabdruck für eine Datenquelle vergleichbar ist[2].

Technisch betrachtet, handelt es sich um mathematische Funktionen, welche eine Zeichenfolge nahezu beliebiger Länge *L* entgegennehmen und folglich daraus einen Ausgabewert fester Länge, den Hashwert *H(L), berechnet*[2]. Der erzeugte Wert muss dabei zwangsläufig folgende Eigenschaften erfüllen:

1. Es sollte nicht effizient möglich sein, von *H(L)* auf den ursprünglichen Wert, also die Zeichenfolge *L* zu schließen. Man spricht bei dieser Eigenschaft von der sogenannten **Einwegeigenschaft**;
2. Es darf nicht möglich sein, dass zwei unterschiedliche Eingabewerte, den gleichen Hashwert nach Anwendung der Funktion ergeben. Kommt es dazu, spricht man dabei von einer **Kollision**. Bei kryptografischen Hashfunktionen darf es folglich also nicht möglich sein, solch eine Kollision zu finden. Diese Eigenschaft nennt man **Kollisionsresistenz**[2].

Die Eigenschaften stellen somit sicher, dass in einer Hashfunktion nicht einfach eine Zeichenfolge durch eine zweite ersetzt werden kann, der zufällig der identische Hashwert zuordenbar ist[2].
In der forensischen Praxis sind aktuell *MD5*, *SHA1* und *SHA2* als Hashingverfahren etabliert. Bereits im Jahr 2013 wurde ein Verfahren entwickelt, welches in wenigen Sekunden eine *MD5*-Kollision erzeugen kann. Im Gegensatz dazu, ist es bei *SHA1* bislang nur theoretisch mit großem Aufwand möglich gewesen, eine Kollision zu erzeugen. Bei ***SHA2*** (*Secure Hash Algorithm 2*) ist bisher keine Kollision bekannt, weswegen wir dieses sichere Hashingverfahren auch im Praktikum anwenden[2].

Um unter Linux beispielsweise einen *SHA2*-Hashwert (***SHA-256***) für einen beliebigen Datenträger zu bestimmen führt man den Befehl *sha256sum* auf den gewünschten Datenträger oder der Datei aus:

```
$ echo "Forensik" > forensik.txt
$ sha256sum forensik.txt
  4764caf513f3192777cb0bb06bafe731410706526cb5424f206a0297886443c2 forensik.txt
```

[2] D. Pawlaszczyk, „Digitaler Tatort, Sicherung und Verfolgung digitaler Spuren“ in Forensik in der digitalen Welt: Moderne Methoden der forensischen Fallarbeit in der digitalen und digitalisierten realen Welt, D. Labudde und M. Spranger, Hg., Berlin, [Heidelberg]: Springer Spektrum, 2017, S. 113–166.

Als Ausgabe erhält man den *SHA-256*-Hash für die Datei „forensik.txt“. Um Manipulationen auszuschließen, kann der Hash der Originaldatei mit der gesicherten verglichen werden. Dies sieht unter Zuhilfenahme der Kommandozeile beispielsweise wie folgt aus:

```
$ sha256sum --check <Hash>
  forensik.txt: OK
```

Das „**OK**“ signalisiert uns, dass die Hashes der Hashdatei *hash* und der Originaldatei übereinstimmen und somit keine Manipulationen durchgeführt wurden. Die Integrität ist somit sichergestellt.

Praktische Untersuchung des virtuellen Datenträgers

Nachdem wir nun die theoretischen Grundlagen besprochen haben, können wir das gelernte praktisch anwenden. Viele forensische Tools funktionieren ausschließlich auf Windows, wie XWays. Im Feldeinsatz können daher Live-Distributionen, wie Paladin Edge sinnvoll werden. Mit diesen können Sie schnell Daten sichern, und sich über die gesicherten Daten einen Überblick verschaffen.

Wir wollen das Szenario einer sichergestellten SD-Karte aus einer Kamera hernehmen. Den Link zur gesicherten SD-Karte haben wir Ihnen auf der Kursseiten ebenfalls zur Verfügung gestellt. Dies ist der gleiche, über den Sie auch schon die virtuelle Maschine heruntergeladen haben. Laden Sie die Datei herunter und speichern Sie diese optimaler Weise direkt in dem Verzeichnis, in dem Sie auch die virtuelle Maschine gespeichert haben.

In VirtualBox können Sie nun die heruntergeladene SD-Karte einbinden. Dazu müssen Sie die virtuelle Maschine allerdings beendet haben. Gehen Sie dann im Startbildschirm von VirtualBox auf die virtuelle Maschine und im oberen Bereich auf „Ändern“.

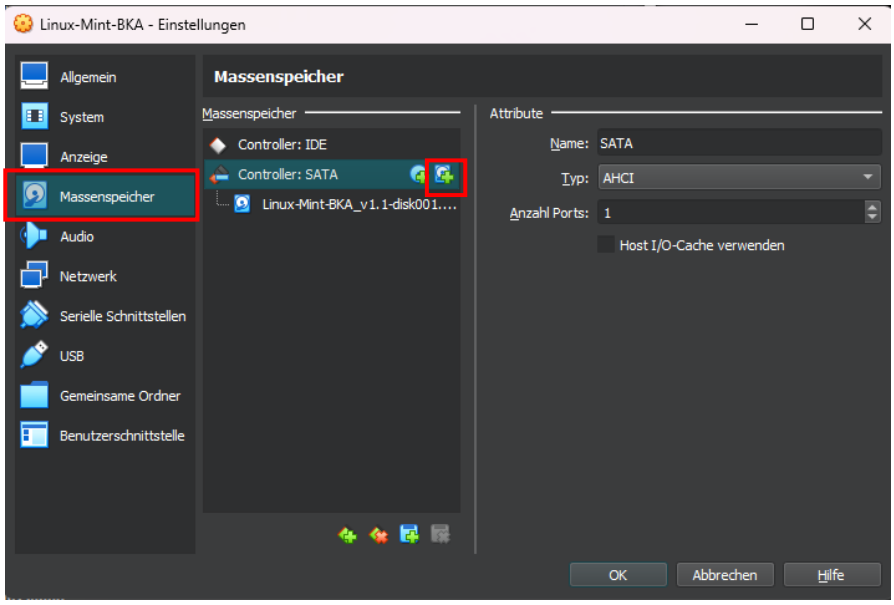

Abbildung 1: Öffnen der Massenspeichereinstellungen

Klicken Sie nun oben links auf das Festplattensymbol „Hinzufügen“ und navigieren Sie in dem sich öffnenden Dateiexplorerfenster zu dem Ordner, in dem Sie die virtuelle Festplatte (.vdi) gespeichert haben. Wählen Sie diese mit dem Button „Öffnen“ aus.

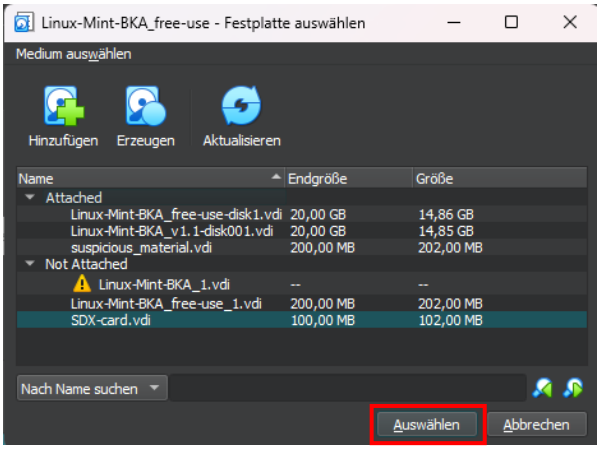

Abbildung 2: Hinzufügen der Festplatte zur virtuellen Maschine

Wählen Sie die SDX-Card.vdi in der Liste aus und klicken Sie im unteren Bereich auf „Auswählen“. Danach sehen Sie in den Einstellungen für Ihre virtuelle Maschine dem SATA-Controller die gerade ausgewählte Festplatte zugeordnet. Sie können nun das Fenster durch einen Klick auf „OK“ schließen und die VM wieder starten.

In Linux wieder angekommen, melden wir uns an und öffnen ein Terminal. Mit dem folgenden Befehl können wir uns die am System verfügbaren Blockspeichergeräte anzeigen lassen:

```
$ lsblk    # list blockdevices
```

Sie bekommen nun eine grobe Übersicht über die am System verfügbaren Speichermedien. Sie sollten eine ähnliche Ansicht bekommen:

```
praktikus@praktikus-VB:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   20G  0 disk
├─sda1   8:1    0    1M  0 part
├─sda2   8:2    0  513M  0 part /boot/efi
└─sda3   8:3    0 19,5G  0 part /
sdb      8:16   0  200M  0 disk
└─sdb1   8:17   0  199M  0 part
```

Abbildung 3: Übersicht der Speichermedien am System

Die Informationen sind wie folgt zu interpretieren. Am Anfang sehen Sie die Systembezeichnung des Mediums. **`sdx`** steht für SCSI/SATA getriebene Festplatten (Blockdevices). Der folgende Buchstabe (**`a,b`**) ist dann immer abhängig von der Reihenfolge der Anmeldung der Datenträger am System. Die folgende Zahl steht für die Bezeichnung der Partition auf der jeweiligen Festplatte. Weiterhin ist für uns die Größe des Mediums wichtig. Diese steht hier unter **`SIZE`** mit der entsprechenden Einheit. **`RM`** sagt uns, ob es ein entfernbares Wechselmedium ist. Wenn 1, dann ist das so, andernfalls nicht. Weiterhin sehen wir unter **`RO`**, ob das Medium nur lesbar (read only) ist. Wenn 1, dann ist es nur lesbar, andernfalls auch schreibbar. **`TYPE`** sagt uns, wie bereits erklärt, ob es sich um eine Platte oder eine Partition handelt. Am Ende sehen wir uns **`MOUNTPOINTS`** an. Diese Angabe sagt uns in welchem Verzeichnis der entsprechende Datenträger eingehängt ist. Beispielsweise können wir über den Pfad **`/boot/efi`** auf die zweite Partition der ersten Festplatte zugreifen.

Das Automatische Einhängen ist bei Linuxsysteme standardmäßig deaktiviert, was für uns in der Forensik essenziell ist. Durch das Einhängen am System im Schreib-/Lesemodus können schon beim Einhängen Daten verändert werden, was den Datenträger manipuliert und somit die Integrität des Asservates verletzt. Die Besonderheit in forensischen Distributionen, wie PALADIN EDGE ist, dass grundsätzlich zum Systemstart keinerlei Platten eingehängt werden, außer der Live-USB-Stick auf dem PALADIN selbst liegt. Wir müssen uns also selbst um das Einhängen kümmern. Das können wir grundsätzlich per

```
$ mount [-option] /dev/<Gerät> <Verzeichnis>
```

Mit dem Befehl mounten wir (einbinden) das Dateisystem des angegebenen Gerätes in das angegebene Verzeichnis unserer Maschine. Für die forensische Sauberkeit ist es wichtig, den Datenträger nur im Lesemodus zu mounten. Das bewirken wir mit einem der zwei Befehle:

```
$ mount <Gerät> <Verzeichnis> -o ro        # Option readonly
$ mount <Gerät> <Verzeichnis> -r           # direkt readonly-Flag setzen
```

Für welchen man sich entscheidet, ist hierbei egal. *Hängen Sie die Partition der SD-Karte (wahrscheinlich* **sdb1***) in ihr System mit dem mount-Befehl im Verzeichnis* **/tmp/sd-card** *ein! Achten Sie darauf, dass diese nur im Lesemodus gemountet wird. Weiterhin muss das Verzeichnis, in welches der Datenträger eingehängt werden soll, auch existieren. Beachten Sie auch, dass das Einhängen nur mit Root-Berechtigungen erfolgen kann. Konsultieren Sie die Manpages.*

Schauen Sie sich erneut an, welche Datenträger am System verfügbar sind, und überprüfen Sie den Erfolg des Einbindens. Wechseln Sie auf den Datenträger und Legen Sie ein neues Verzeichnis an. Was stellen Sie fest?

Sichern des Datenbestandes

Einstiegsfrage in den Unterabschnitt: *Um welche Art der forensischen Untersuchung handelt es sich bei der Analyse der SD-Karte aus der Kamera? Begründen Sie Ihre Antwort.*

Bei der Untersuchung eines digitalen Datenträgers gilt immer der Grundsatz, dass nie am Originalasservat gearbeitet wird. Werden die Spuren auf dem Asservat versehentlich manipuliert, können die Änderungen teils nicht rückgängig gemacht werden und beeinflussen nebenbei die Integrität der Daten. Um das zu verhindern, werden für jede Untersuchung forensische Kopien des Datenträgers erzeugt. Wird an diesem der Datenbestand geändert oder eine Operation ausgeführt, die das Asservat unbrauchbar macht, kann eine neue Kopie angelegt werden und von vorn begonnen werden, ohne das Original zu beeinflussen. Das soll in diesem Abschnitt auch durch sie durchgeführt werden. In Praktikum 1 haben wir im Abschnitt Paketierung das Paket **ewf-tools** installiert. Dieses werden wir nun zum Einsatz bringen. Wir wollen außerdem ein weiteres kleines Standardtool betrachten.

Je nachdem, welchen Zweck wir verfolgen und Programm wir zur Auswertung verwenden, kann es sinnvoll sein einen Datenträger in andere Formate zu konvertieren. Das einfachste Format zum bitgenauen Kopieren von Datenträgern ist **RAW**, also das rohe Abbild. Dieses Format ist unkomprimiert und veränderbar. Außerdem braucht es als Image genau so viel Speicher, wie die Größe des Originaldatenträgers ist, was schnell ineffizient wird. Schauen wir uns dennoch das kleine Standardtool dazu an. Wir nutzen dazu **dd** (Disk Dump) und nutzen folgende Syntax:

```
$ dd if=<Quelldatei> of=<Ausgabedatei> [weitere Optionen]
```

weitere Optionen können z.B. die vom Datenträger verwendete Blockgröße, der Umgang mit fehlerhaften Sektoren oder das Anzeigen einer Fortschrittsleiste sein. **if** steht für Input File und **of** für output file. Um Datenträger mit **dd** zu sichern, müssen wir diese nicht mounten. Die Blockgröße (Sektorgröße) können Sie herausfinden mit dem Befehl

```
$ fdisk <Gerätepfad>
$ Befehl (m für Hilfe): p       # mit Enter bestätigen
```

*Legen Sie sich ein neues Verzeichnis „***forensic_images***“ unter ihrem Home-Verzeichnis an. An diesen Ort machen Sie nun mit* **dd** *ein Abbild der SD-Karte. Geben Sie außerdem die Blockgröße an und lassen Sie sich einen Fortschrittsbalken während der Sicherung anzeigen. Kontrollieren Sie den Erfolg Ihres Befehls im Dateisystem. Wie groß ist die gesicherte Datei?*

Der Vorteil eines RAW-Images ist, dass wir daraus jeden anderen Image-Typ generieren können. Außerdem können wir RAW-Images mittels **`mount`** behandeln, als wäre es ein physischer Datenträger. Damit bietet sich das Erstellen eines RAW-Images auch an für die Virtualisierung von gesicherten Datenträgern.

Wir wollen uns das Verfahren noch einmal mit einem richtigen forensischen Tool anschauen. Aus der Bibliothek **`ewf-tools`** nutzen wird dazu das Tool **`ewfacquire`**. Es kann Datenträger in verschiedene Dateiformate schreiben. Außerdem kann es dadurch die von den Formaten unterstützte Komprimierung nutzen und so deutlich kleinere Abbilder erzeugen. Wir nutzen dazu die Syntax:

```
$ ewfacquire [-optionen] Quelldatei
```

Die Optionen von **`ewfacquire`** bieten eine deutlich detaillierte Sicherung durch die zusätzlichen Angaben zur Chain of Custody. Darunter z.B. die Fallnummer, der Name des sichernden Forensikers und die Asservatennummer.

Fallszenario

Sie wurden als digitaler Forensiker dazu beauftragt eine Sicherung durchzuführen. In dem Fall gibt es weitere Asservate, wie Windows Desktop PCs und verschiedene Smartphones. Durch die Sicherung der anderen Asservate sind zurzeit alle Sicherungsanschlüsse an der forensischen Arbeitsstation belegt. Das einzige übrige Asservat ist eine SD-Karte aus einer Digitalkamera, die es zu sichern gilt. Sie haben weiterhin Zugriff auf eine Linux-Workstation, mit der sie eine Sicherung durchführen können. Im dem vorliegenden Fall geht es um den vorsätzlichen Missbrauch von Büroräumen in der Hochschule Mittweida durch zwei Mitarbeiter, welche dort scheinbar illegal exzessiven Kaffeekonsum vollziehen. Es müssen Beweisfotos gefunden werden, welche den Sachverhalt be- oder widerlegen. Ihnen sind folgende Falldaten bekannt:

- **Fallnummer:** 2023-03-53492
- **Asservatennummer:** Ass-05-1
- **Beschreibung:** Digitalkamera SDX-Karte
- **Sichernder Forensiker** Sie
- **Verwendete Forensiksuite:** XWays Forensics (Encase6)

Sichern Sie mit Ihrem Wissen und den vorliegenden Daten die SD-Karte der Kamera. Beziehen Sie alle Angaben mit ihren **`ewfacquire`***-Befehl ein, damit der Sicherungsprozess möglichst transparent ist. Schreiben Sie das Image unter einem Namen mit dem Muster:* ***<Fallnummer>_<Asservatennummer>_<Gerätetyp>****. Lassen Sie außerdem ein Logfile schreiben und das Image bestmöglich komprimieren (best). Alle anderen Angaben können Sie im folgenden Fenster jeweils mit Enter bestätigen. (Achtung: Langer Befehl)*

Lassen Sie sich die Größe der entstandenen Datei ausgeben. Was stellen Sie fest? Schauen Sie sich ebenfalls die geschriebene Logdatei an. Was finden Sie darin und wofür ist es nützlich?

Hashen der bestehenden Dateien

Nachdem wir nun die Daten von der SD-Karte gesichert haben, können wir diese nun „zur Seite legen“ und arbeiten an den gerade erstellten Kopien weiter. Dazu können wir beide der Formate nutzen. Das EWF-Format schauen wir und hier nur grob an. Dazu können wir weitere Tools aus **`ewf-tools`** nutzen. Dazu gehören die folgenden:

```
$ ewfinfo <Abbild.E01>
$ ewfverify <Abbild.E01>
```

Per **`ewfinfo`** bekommen wir die im EWF-Format gespeicherten Informationen ausgegeben. Darunter auch die bei der Erstellung angegebenen. Das macht Sinn, wenn Sie ein Abbild weitergeben zur Auswertung. Es weiß jeder sofort, wer das Abbild angelegt hat und bekommt weitere Metainformationen. **`ewfverify`** überprüft die Integrität des Abbildes. Dazu nimmt es den im File gespeicherten Hash und berechnet diesen neu und vergleich die beiden

Werte. Damit können wir jede Änderung am Abbild wahrnehmen. *Testen Sie beide Befehle auf Ihr erstelltes EWF-Abbild.*

Schauen wir uns die Daten im RAW-File an. Um auch hier die Integrität der Daten auf dem Datenträger zu wahren mounten wir das Image im Lesemodus (readonly). Zu beachten ist, dass wir den gesamten Datenträger gesichert haben. Das bedeutet, dass auf der Festplatte mehrere Partitionen existierten können. Wir brauchen zuerst einen Überblick über den Datenträger. Dazu nutzen wir den Befehl:

- **fdisk <Imagedatei>**

Wir bekommen eine interaktive Anzeige des Tools **fdisk**. Mit der Eingabe von „**p**" und der Bestätigung mit Enter sehen wir die Partitionstabelle des Datenträgers und einige Informationen. *Schauen Sie sich die Partitionstabelle ihres erstellten Images an. Wie viele Partitionen sind eingerichtet? An welchem Sektor beginnen diese?*

Um eine bestimmte Partition einzubinden, müssen wir deren Offset angeben. Der Offset ist der Startsektor multipliziert mit der Größe eines Sektors. *Nutzen Sie den folgenden Befehl, um das erstellte RAW-Image einzubinden:*

- **sudo mount <Imagedatei> <Einhängepunkt> –o ro,loop,offset=$((<Startsektor>*<Blockgröße>))**

*Hängen Sie ihr Image im Pfad **/tmp/image** ein. Erstellen Sie die Pfade entsprechend. Navigieren Sie anschließend in den Pfad.*

Anschließend können sie mit **lsblk** den Erfolg des Einhängens prüfen und sehen, dass nun ein loop-Device entstanden ist, welches an dem Pfad eingebunden ist, an welchem Sie es gerade eingehängt haben. *Bewegen Sie sich zu dem Pfad und schauen Sie sich ein wenig um. Sie können sich die vorhandenen Bilder auch mittels eines Kommandos anschauen:*

```
$ xdg-open <Bilddatei>
```

Mit den Pfeiltasten können Sie zwischen den Bildern navigieren. Wir wollen schauen, ob auf dem Datenträger inkriminiertes Material vorhanden ist. Dazu gibt es oftmals Hashtabellen, welche wir auch hier heranziehen wollen. Diese ist leider noch nicht auf ihrer virtuellen Maschine und muss erst heruntergeladen werden. Das gleiche gilt für die Vergleichsbilder, die wir Ihnen zur Veranschaulichung zur Verfügung stellen. Dazu sind Ihnen folgende Links gegeben:

https://tirockx-forensics.ddns.net/foreNCics/index.php/s/bPgkxiej9TJgRSn/download/vergleichsbilder.tar
https://tirockx-forensics.ddns.net/foreNCics/index.php/s/gXBiBZD5GfxJBrq/download/hashtable

Laden Sie die Datei mit dem folgenden Befehl in ihr Home-Verzeichnis herunter:

```
$ curl <Link> -o hashtable
$ curl <Link> -o vergleichsbilder.tar
```

mit der Option **-o** schreibt **curl** den heruntergeladenen Inhalt in die angegebene Datei. Die Hashtabelle besteht aus zwei Spalten, welche die SHA256-Hashs und den Originalnamen der inkriminierten Dateien enthalten. Diese wurden aus vorherigen Fällen zusammengetragen und kann herangezogen werden, um Dateien abzugleichen, ohne Sie anschauen zu müssen*. Entpacken Sie folgend noch das heruntergeladene .tar-Archiv mit den Bildern. Geben Sie dazu folgende Befehle in Ihrem Home-Verzeichnis ein:*

```
$ mkdir vergleichsbilder                          # neuen Ordner für Bilder erstellen
$ tar -xvf vergleichsbilder.tar -C vergleichsbilder  # Bilder in den neuen Ordner entpacken
```

Sie können, wie oben gelernt, die SHA256-Hashs der Dateien über die auf der SD-Karte gefundenen Dateien berechnen und mit der Hashtabelle abgleichen. *Berechnen Sie zuerst die SHA256-Summen aller Dateien, die sie auf dem Datenträger finden und speichern Sie die Datei mit den berechneten Hashwerten in Ihrem Home-Verzeichnis ab. Schauen Sie sich die Datei anschließend an und machen Sie sich kurz mit deren Struktur vertraut.*

Sie müssten nun die **Hashtabelle**, die Datei mit den berechneten **SHA256-Summen** und den Ordner mit **Vergleichsbilder** in Ihrem Home-Verzeichnis sehen. Es gibt nun verschiedene Möglichkeiten die Hashs abzugleichen. Einerseits kann das über ein Skript realisiert werden, aber wir wollen uns hier auf einen schnellen händischen Abgleich beschränken. Die Hashtabelle ist so vorbereitet, dass die Bilder in der richtigen Reihenfolge sind und sie so einen Side-by-Side Vergleich machen können. *Dazu können Sie sich mit* **`paste`** *einfach mal beide Dateien zeilenweise nebeneinander ausgeben lassen.* ***ACHTUNG:*** *Ziehen Sie ggf. das Fenster der VM etwas größer, da die SHA256-Hashs ziemlich lang sind. Was können Sie feststellen? Gibt es etwas Auffälliges in den auf der SD-Karte festgestellten Bildern? Können Sie sich erklären, warum das so ist?*

Zusatz

Als Forensiker würden wir natürlich an dieser Stelle nicht aufgeben, nur weil ein Hash nicht stimmt 😊. Schauen wir uns das Abbild der SD-Karte noch einmal genauer an. Ein Verfahren, welches an der Stelle zum Einsatz kommen kann, ist die Suche nach gelöschten Dateien, das **File-Carving**. Hierbei werden Dateien unabhängig von den Indexierungen des Dateisystem in den Bits und Bytes des Datenträgers gesucht. Der File-Carver schaut nach dem Header von Dateien und den extrahierbaren Metainformationen. Genaueres erfahren Sie in den Vorlesungen. An der Stelle schauen wir uns mal an, wie das File-Carving unter Linux aussehen könnte. Wir nutzen einen File-Carver, welcher das Dateisystem komplett missachtet und alles hervorbringt, was er findet. **`foremost`** ist dazu perfekt geeignet, muss aber zuerst installiert werden. *Installieren Sie, wie gelernt den File-Carver „***`foremost`***“.*

Für die Verwendung von **`foremost`** ist es nicht notwendig eine Partition des Abbildes zu mounten. *Daher geben wir, falls das Abbild noch eingebunden ist, zuerst den* **`umount`***-Befehl mit der folgenden Syntax:*

```
$ sudo umount <Mountpoint>
```

Anschließend können wir **`foremost`** auf unser Abbild loslassen mit dem Befehl:

```
$ foremost -t jpg,gif,png -i <Abbilddatei>
```

`foremost` schreibt uns nun die gecarvten Bilder in einen Ordner mit dem Namen „**`output`**“. *Öffnen Sie diesen und schauen Sie sich dessen Inhalt an.* Dort sind die Dateien nach Dateityp sortiert. Nun sind sie in der Lage auch die SHA256-Hashes der gefundenen Datei zu bilden. Leider ist es so, dass foremost die Bilder nicht korrekt wiederherstellt, sodass die Hashwerte der gecarvten Bilder nicht denen der Hashtabelle entsprechen. *Allerdings können wir einen visuellen Vergleich der Bilder vornehmen. Schauen Sie sich die gefundenen Bilder an. Können Sie ein Bild ausmachen, welches möglicherweise auf den übrigen Hash passen könnte? Finden Sie weitere Besonderheiten im Datenbestand? Können Sie sich das Phänomen erklären?*

Herzlichen Glückwunsch! Sie haben sich als Linux-Forensiker bewiesen.
Nun bleibt nur noch das Thema Shell-Skripting.