WILHELM BÜCHNER HOCHSCHULE
Mobile University of Technology

Studienheft

SRN04

Intrusion Detection und Intrusion Prevention

SRN04

Intrusion Detection und Intrusion Prevention

Christopher Ruppricht, M.Sc.

Falls wir in unseren Studienheften auf Seiten im Internet verweisen, haben wir diese nach sorgfältigen Erwägungen ausgewählt. Auf Inhalt und Gestaltung haben wir jedoch keinen Einfluss. Wir distanzieren uns daher ausdrücklich von diesen Seiten, soweit darin rechtswidrige, insbesondere jugendgefährdende oder verfassungsfeindliche Inhalte zutage treten sollten.

Intrusion Detection und Intrusion Prevention

Inhaltsverzeichnis

Anhang

Vorwort

Liebe Studierende,

die vernetzte Kommunikation zwischen einzelnen Systemen ist ein wesentlicher Erfolgsfaktor der Informationstechnik. Doch mit der steigenden Vernetzung erhöhen sich auch die Gefährdungen, die für die einzelnen Systeme existieren.

In diesem Studienheft lernen Sie die Funktionsweise sowie den Unterschied von Intrusion-Detection-Systemen und Intrusion-Prevention-Systemen sowie ihre Bedeutung für die Netzwerksicherheit kennen. Darüber hinaus erhalten Sie einen Überblick über die Möglichkeiten und Grenzen solcher Systeme kennen.

Sie alle kennen einen Film, ein Buch oder zumindest eine effekthaschende Berichterstattung über „Hacker“ oder „Cyberkriminelle“, die es innerhalb von Sekunden schaffen, in ein Netzwerk einzudringen.

Dass dies oftmals fernab jeder Realität ist, braucht man Ihnen nicht weiter auszuführen. Dennoch ist vieles aus diesem Studienheft Bestandteil jedes Sicherheitsforschers oder auch „Hackers“.

Darüber hinaus widmet sich dieses Studienheft der IT-Forensik. Auch hier haben viele Menschen ein bestimmtes Szenario vor ihrem geistigen Auge: Der am Tatort sichergestellte und durch den Einsatz in Mitleidenschaft gezogene Computer soll auf Beweise überprüft werden. Er ist eigentlich vollkommen zerstört, jedoch rekonstruiert und überprüft der örtliche IT-Forensiker der lokalen Polizei ihn innerhalb von Stunden und erbringt den unwiderlegbaren Beweis, der zur Festnahme und Verurteilung des Täters führt. Auch hier gibt es also die ein oder andere Mär zu entlarven. Das entsprechende Kapitel des Studienhefts zielt darauf ab, Ihnen ein planvolles Vorgehen zur Sicherung notwendiger Daten näherzubringen und Sie vor fehlerhaften Handlungen zu bewahren.

Beim Bearbeiten dieses Studienhefts wünschen wir Ihnen nun viel Spaß und Erfolg!

Ihre Studienleitung

1 Grundlagen

In diesem Kapitel lernen Sie

- *die Grundprinzipien zur Konzeption von sicheren Systemen,*
- *die Phasen einer Kompromittierung sowie*
- *die Prinzipien der Netzwerkverteidigung*

kennen. Diese Punkte bilden die theoretische Grundlage für die Auswahl wirkungsvoller Maßnahmen zum Schutz des Netzwerks.

Darüber hinaus erhalten Sie Informationen über ein allgemein anerkanntes Bewertungsschema zur Einstufung von Schwachstellen in Systemen und Anwendungen. Sie sollten das ▸CVSS-Modell nachvollziehen und die Anwendungsweise verstehen können.

Den Abschluss des Kapitels bildet eine kurze Zusammenfassung des ▸OSI-Modells und Hinweise zur Einordnung von Netzwerkports.

1.1 Grundlagen zur Konzeption sicherer Systeme

Die Leitfäden und Handbücher für die Konzeption und Konstruktion von verlässlichen und sicheren IT-Systemen füllen ganze Bibliotheken und werden in anderen Studienheften ausführlich behandelt. Daher beschränken wir uns in diesem Studienheft auf die grundlegenden Prinzipien für die Planung und Konzeption sicherer Systeme. Diese wurden bereits im Jahr 1975 von Jerome H. Saltzer und Michael D. Schroeder definiert. Zu diesen Prinzipien zählen das Prinzip

- der Erlaubnis,
- der Vollständigkeit,
- der minimalen Rechte („need-to-know“),
- der Benutzerakzeptanz sowie
- des offenen Entwurfs.

Das Prinzip der Erlaubnis bedeutet, dass Zugriffe auf ein System grundsätzlich verboten sind und einer ausdrücklichen Erlaubnis bedürfen. Damit verbunden ist das Prinzip der Vollständigkeit, nämlich dass jeder Zugriff auf das System zu prüfen ist.

Weiterhin sieht das Prinzip der minimalen Rechte vor, einem Aufgabenträger nur die Benutzerrechte zu gewähren, die er zur Erfüllung seiner Aufgaben benötigt. Die Sicherheitsmechanismen müssen einfach angewendet werden können, damit die Benutzer diese akzeptieren. Das Prinzip des offenen Entwurfs stellt die Unabhängigkeit der Systemsicherheit von der Geheimhaltung verwendeter Verfahren sicher.

Übung 1.1:

Was bedeutet das Prinzip der Erlaubnis bei der Konzeption sicherer Systeme?

1.2 Phasen der Kompromittierung

Ein Angreifer, der in ein Netzwerk einbrechen möchte, durchläuft bei seinem Angriff verschiedene Phasen, die fließend ineinander übergehen können. Diese Phasen werden auch als die fünf Ps eines Angriffs bezeichnet. Diese fünf Phasen sind:

- Probe
- Penetrate
- Persist
- Propagate
- Paralyze

Nachfolgend finden Sie eine kurze Beschreibung der jeweiligen Phasen.

1. Probe

Zu Beginn jeden Angriffs steht die Sammlung von Informationen über das Ziel. Hierzu werden sowohl technische als auch nichttechnische Methoden angewendet. Als Beispiele für nichttechnische Methoden können vor allem weitverbreitete Social-Engineering-Techniken wie Telefonanrufe oder das Durchsuchen des Papiermülls genannt werden.

Bei technischen Methoden zur Informationssammlung analysiert der Angreifer nach außen kommunizierende Systeme, wie beispielsweise einen Domain-Name-Service-Dienst, einen Mailserver oder einen Webserver. Darüber hinaus überprüft der Angreifer die Meta-Informationen in veröffentlichten Dokumenten.

Natürlich können diese beiden Methodiken auch miteinander kombiniert werden, indem man soziale Netzwerke nach den Mitarbeitern des Zielunternehmens durchsucht, um auf deren persönliche Interessen schließen zu können oder zumindest die relevanten Rollen für einen Angriff herauszufinden. Die gesammelten Informationen werden vom Angreifer auf mögliche Schwachstellen hin ausgewertet.

2. Penetrate

Im Rahmen der Auswertung der gesammelten Informationen auf mögliche Schwachstellen erfolgt auch die Auswahl der Werkzeuge und der Vorgehensweise für den Angriff. Der Angreifer wird hierbei stets das Ziel und den damit verbundenen Aufwand abwägen. Wenn sein Ziel beispielsweise darin besteht, möglichst viele Informationen zu sammeln und diese an den Wettbewerber zu verkaufen, wird er vielleicht weiterhin die Papiercontainer durchwühlen, wenn das Zielunternehmen sensible Dokumente nicht angemessen vernichtet und die Papiercontainer leicht zugänglich sind.

Sofern das Ziel über eine Schadsoftware angegriffen werden soll, würde der Angreifer in dieser Phase die vorhandenen Systeme auf Schwachstellen prüfen. Die gängigsten Beispiele für solche Schwachstellen sind veraltete Konfigurationen oder Benutzer mit trivialen Passworten. Ein Angreifer sucht in dieser Phase vorzugsweise nach „schwächeren“ Systemen, wie beispielsweise einer schlecht gesicherten Entwicklungsumgebung mit Daten aus dem produktiven Betrieb.

Hierfür genügen oftmals einfache Tests wie beispielsweise *Portscans* oder bewusstes Provozieren von Fehlermeldungen bei Webservern. Abb. 1.1 zeigt eine Fehlermeldung eines Onlineshops, der auf einem veralteten Microsoft CIIS bereitgestellt wird.

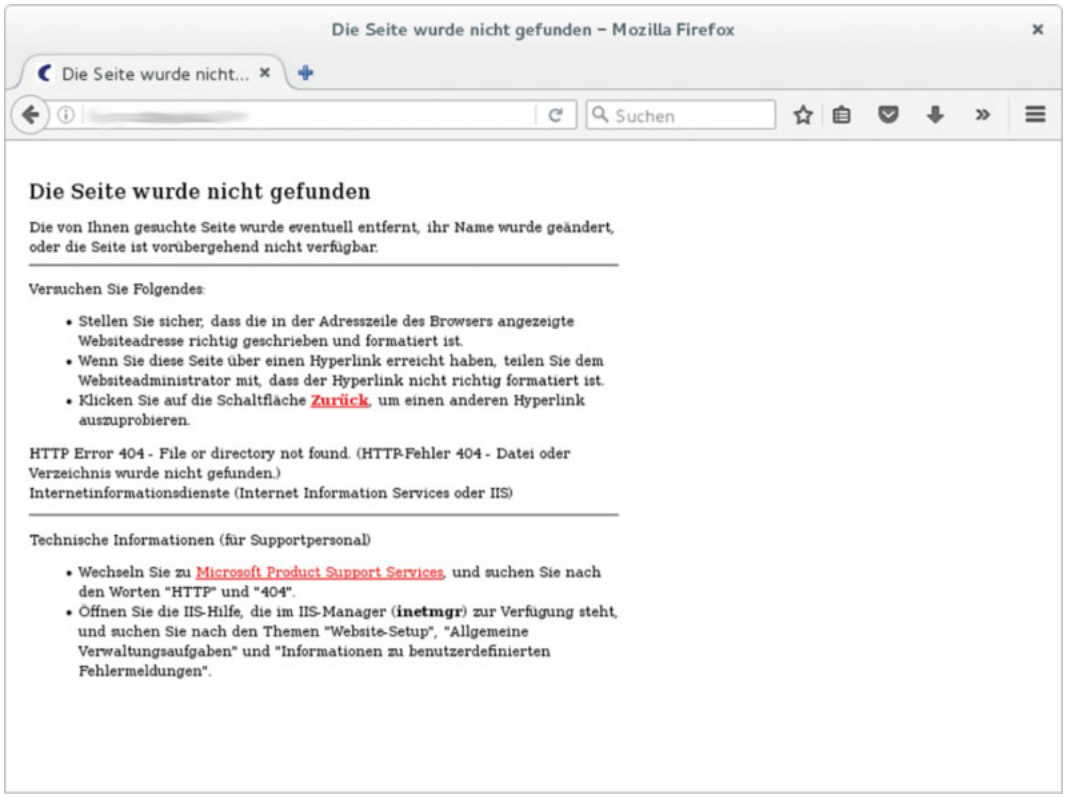

Abb. 1.1: Fehlermeldung (Link existiert nicht) des IIS

Warum ist diese Information für den Angreifer nützlich? Der Angreifer weiß jetzt, dass das System, auf dem dieser Webserver bereitgestellt wird, ein Windows-System ist und als Webserver Microsoft ▸IIS verwendet wird. Weiterhin erfährt er durch eine kurze Internetrecherche, dass die angezeigte Seite eine veraltete Fehlerseite ist. Dies lässt darauf schließen, dass auch der verwendete Webserver nicht mehr aktuell ist.

Mit diesen Informationen kann er sich jetzt auf die Suche nach entsprechenden *Exploits* machen. Ein *Exploit* ist eine Beschreibung oder ein Programm, das systematisch eine Schwachstelle ausnutzt. Ein besonders gravierende Variante hiervon sind sogenannte 0Day Exploits (gesprochen Zero Day Exploits). Diese bezeichnen Schwachstellen, für die der Hersteller eines Betriebssystems oder einer Anwendung kein Patch oder Update bereitgestellt hat.

Der Angreifer kann also hiermit jedes System eines Herstellers angreifen, sofern es nicht durch andere Sicherheitsmaßnahmen geschützt wurde. Eine beispielhafte Übersicht der passenden Exploits zu unserem Webserver ist in Abb. 1.2 aufgelistet.

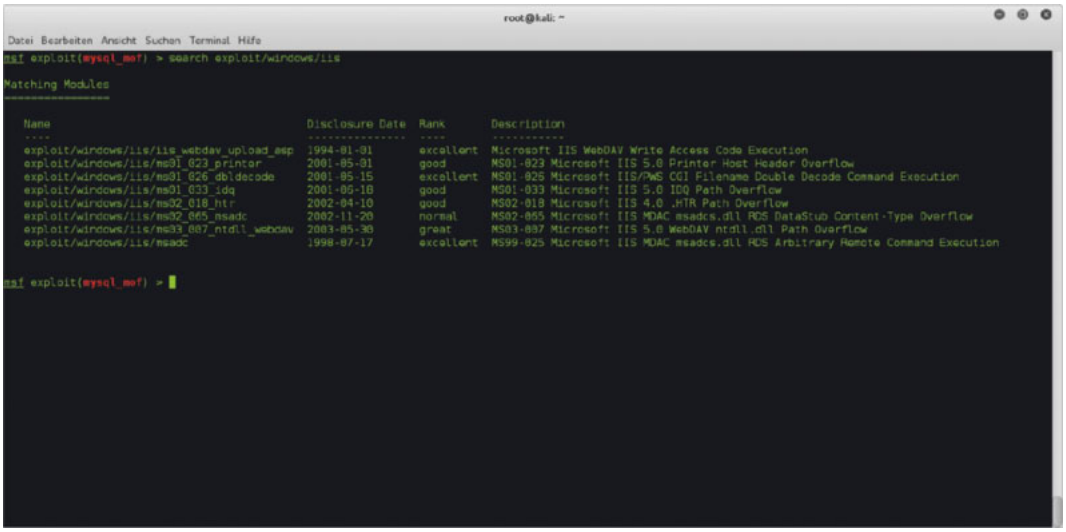

Abb. 1.2: Exploits des IIS

Die in Abb. 1.2 dargestellten Exploits sind Teil der Anwendung *Metasploit* des Herstellers Rapid7 LLC. *Metasploit* ist eine Anwendung zum Auffinden und Testen von Schwachstellen und ist weitverbreitet bei Penetrations-Testern. Ein versierter Angreifer würde natürlich nicht auf solche Anwendungen zurückgreifen, sondern auf entsprechenden Schwarzmarkt-Websites ein passendes Exploit erwerben oder es selbst entwickeln und versuchen, das schwache System damit zu infizieren.

3. Persist

Das Etablieren des Zugangs erfolgt durch die Ausnutzung der vorhandenen Schwachstelle sowie deren Verfestigung. Dies ist die oftmals schwierigste Phase für den Angreifer. Immerhin muss er die freigelegte und ausgenutzte Schwachstelle für einen dauerhaften Zugang ausbauen. Dies bedeutet beispielsweise, das Schadprogramm, das durch ein geöffnetes PDF in den *Random-Access-Memory(▸RAM)* geladen wurde, dauerhaft auf dem Datenträger des Systems zu speichern, sodass die Schwachstelle auch nach einem Neustart des Systems verfügbar ist.

4. Propagate

Die Ausbreitungsphase beinhaltet das Tarnen der installierten Schadsoftware sowie den Versuch der Rechteeskalation. Hierzu wird die Schadsoftware oftmals in betriebssystemnahe Verzeichnisse kopiert oder an bestehende Systemprozesse angehängt. Darüber hinaus wird in dieser Phase die Kommunikation zwischen dem Angreifer und dem infizierten System hergestellt.

Üblicherweise meldet sich das infizierte System über ein gängiges Protokoll bei einer stellvertretenden dritten Instanz, die sich ebenfalls unter der Kontrolle des Angreifers befindet. Eine solche Instanz nennt man „Command and Control"-Server. Je nach Art der Schadfunktion werden in dieser Phase auch weitere Programme durch den Angreifer auf das infizierte System nachgeladen.

5. Paralyze

In dieser Phase ist der Angreifer am Ziel seiner Arbeit und beginnt mit dem Abzug der Daten oder der Zerstörung des Systems. In dem oben beschriebenen Beispielkontext des Webservers würde der Angreifer

- diesen mit einer Schadsoftware infizieren, die auf die Systeme der anderen Website-Besucher übertragen wird („Drive-by-Download"),
- die Inhalte der Website durch eigene Inhalte ersetzen („Defacement") oder
- Daten aus der Datenbank des Webservers stehlen.

Übung 1.2:

Nennen Sie drei Phasen der Kompromittierung.

1.3 Bewertung von Schwachstellen

Ein weitverbreiteter Standard zur einheitlichen Benennung von Schwachstellen ist ▸CVE. Dieser Standard ermöglicht es, Informationen über Schwachstellen zuzuordnen und die Schwachstellen zu katalogisieren. Die ▸CVE-Methode wurde im Jahr 1999 veröffentlicht, um den verschiedenen Interessengruppen einer Schwachstelle den Austausch von Informationen über diese zu vereinfachen und eine einheitliche Benennung zu ermöglichen.

Die ▸CVE-Benennung setzt sich aus der Jahreszahl und einer fortlaufenden Nummer zusammen, die durch die amerikanische gemeinnützige und regierungsnahe Organisation *MITRE* vergeben wird.

Darüber hinaus beinhaltet jede ▸CVE-Benennung eine kurze Beschreibung der Schwachstelle und Verweise auf weitere sachdienliche Hinweise zur Schwachstelle des Produktherstellers und zu renommierten Dritten (z.B. Antivirus-Hersteller, ▸CERT). Abb. 1.3 zeigt einen beispielhaften ▸CVE-Eintrag (Nummer: ▸CVE-2015-7755) aus dem Jahr 2015.

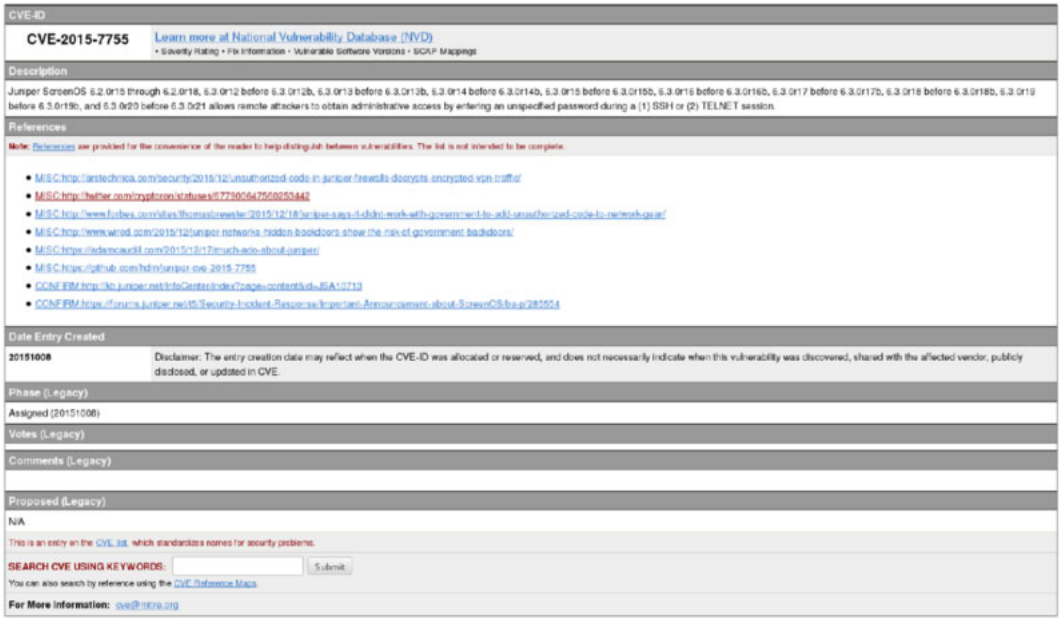

Abb. 1.3: CVE-Eintrag einer Schwachstelle bei weitverbreiteten Netzwerksystemen

Die Schwachstelle enthält Informationen zu einer Hintertür, die einen Fernwartungszugang zu allen Systemen eines weitverbreiteten Netzwerkkomponentenherstellers unter Verwendung eines nicht dokumentierten Passworts ermöglicht.

Neben der allgemeinen Benennungsmethodik existiert auch ein Industriestandard zur Bewertung von Schwachstellen. Einer dieser Bewertungsstandards ist *Common Vulnerabilities Scoring System(▸CVSS)*. Der Standard dient der Einschätzung des Schweregrads einer Sicherheitslücke auf einer aufsteigenden Kardinalskala von 0.0 bis 10.0. Tab. 1.1 zeigt die gebräuchliche Einstufung dieser Schweregrade.

Tab. 1.1: CVSS-Bewertung von Sicherheitslücken

Schweregrad Sicherheitslücke	Bewertung
nicht vorhanden	0.0
niedrig	0.1 bis 3.9
mittel	4.0 bis 6.9
hoch	7.0 bis 8.9
kritisch	9.0 bis 10.0

Die Einschätzung unterteilt sich in folgende Bewertungsgruppen:

- Basisfaktoren (Base Metric Group)
- temporäre Faktoren (Temporal Metric Group)
- Umgebungsfaktoren (Environmental Metric Group)

Die Basisfaktoren setzen sich zusammen aus

- der Einordnung des Angriffsvektors (AV),
- der Einordnung der Angriffskomplexität (AC),
- den benötigten Berechtigungen zur Ausführung des Angriffs (PR),
- den eventuellen voraussetzenden Benutzerinteraktionen (UI),
- den Auswirkungen für andere Komponenten und Anwendungen (S) sowie
- den Auswirkungen auf die Vertraulichkeit (C), die Integrität (I) und die Verfügbarkeit (A) eines Systems oder einer Anwendung.

Diese dienen zur Bestimmung des Schweregrads der Auswirkungen bei einer erfolgreichen Attacke. Die einzelnen Teilfaktoren sind dabei in zwei- bis dreistufige Ordinalskalen unterteilt und mit anteiligen Werten 0 und 1 hinterlegt.

Die Einbeziehung der temporären Faktoren und der Umgebungsfaktoren ist optional und dient vor allem der angemessenen Berücksichtigung von risikomindernden Aspekten. Die temporären Faktoren bewerten

- den „Reifegrad“ des Exploits (*Exploit Code Maturity* (E)),
- die vorhandenen, vom Produkthersteller empfohlenen Gegenmaßnahmen (*Remediation Level* (RL)) sowie
- den Grad der Nachvollziehbarkeit zu Details der Schwachstelle (*Report Confidence* (RC)).

Die Umgebungsfaktoren individualisieren die vorgenommene Bewertung und passen sie auf den jeweiligen Kontext des Ziels an. Sie dienen sozusagen der Feinjustierung der Schwachstelle für die vorhandene Vertraulichkeit, Verfügbarkeit oder Integrität des eigenen Systems oder Netzwerks.

Der Angriffsvektor wird beispielsweise unterschiedlich bewertet, je nachdem, ob er im Netzwerk (0.85), im angrenzenden Netzwerk (0.62), lokal (0.55) oder physisch (0.2) erfolgt. Die Angriffskomplexität wird als niedrig (0.77) oder hoch (0.44) bewertet. Das vollständige Bewertungsschema sowie ein beispielhaftes Berechnungsprogramm ist auf der Website www.first.org/cvss (▸CVSS, 2016) zu finden.

1.4 Prinzipien zur Netzwerkverteidigung

Für eine wirksame Verteidigungsstrategie des eigenen Netzwerks ist es zunächst notwendig, eine kontinuierliche **Überwachung und Analyse des Netzwerkverkehrs** einzurichten. Die Überwachung sollte dabei risikoorientiert und in Abhängigkeit des Schutzbedarfs der einzelnen Systeme erfolgen. Dies bedeutet, dass beispielsweise für das interne Local Area Network (▸LAN) andere Überwachungsinstanzen konfiguriert werden sollten, als für die Demilitarized Zone (▸DMZ) des Netzwerks.

Die Analyse sollte dabei weitestgehend automatisiert erfolgen und dem Analysten regelmäßig vorbereitete Berichte liefern, die die beachtenswerten Ereignisse zusammenfassen und bei Bedarf eine Detailanalyse der Rohdaten ermöglichen. Darüber hinaus sollte Analyse nicht durch einen Administrator, sondern durch einen fachkundigen Dritten erfolgen.

Des Weiteren ist die Platzierung der Analyseinstanzen im Netzwerk für die Überwachung entscheidend. Wenn beispielsweise das ▸IDS zwischen zwei Firewalls platziert wird, ist es unter Umständen nicht möglich, den Netzwerkverkehr auf Systemebene zu überprüfen, da die Netzwerkpakete von den Firewalls verändert wurden.

Ein weiteres Prinzip der Verteidigungsstrategie ist es, einem potenziellen Angreifer die **Bewegungsfreiheit im Netzwerk einzuschränken**. Hierunter versteht man üblicherweise die Trennung von Netzwerken mit unterschiedlichem Schutzbedarf durch den Einsatz von Firewallsystemen und Network Address Translation (▸NAT) sowie den Einsatz von Benutzerberechtigungsverfahren.

Die vergebenen Berechtigungen sollten dabei auf ein notwendiges Minimum reduziert und jeder Benutzer eindeutig authentifiziert werden.

Des Weiteren sollten alle Systeme des Netzwerks erfolgreich authentisiert werden. Ein weitverbreitetes Verfahren hierzu ist beispielsweise eine Vorgehensmethode nach dem Standard 802.1X des Institute of Electrical and Electronics Engineers (▸IEEE).

Das vierte Prinzip einer robusten Verteidigungsstrategie des Netzwerks ist die **Reduzierung der Netzwerk- und Systemdienste**. Diese sollten auf ein notwendiges Minimum reduziert werden, da eine Vielzahl von Anwendungen, die von einem Angreifer verwendet werden, nach Schwachstellen in diesen Diensten suchen und Systeme anhand der bereitgestellten Dienste authentifizieren.

Die letzte Säule der Verteidigung ist ein konsequentes **Aktualisieren des Netzwerks**. Hierunter ist sowohl ein angemessenes Patchmanagement zu verstehen als auch der Austausch von Systemen, sofern für diese keine Aktualisierungen mehr durch den Hersteller bereitgestellt werden. Eine grundlegende Voraussetzung für das stetige Aktuell-Halten ist die Kenntnis von neuen veröffentlichten Aktualisierungen, aber auch von aktuellen Bedrohungen.

Eine Vielzahl von Herstellern stellen mittlerweile automatisierte Aktualisierungszyklen bereit. Die Firma Microsoft veröffentlicht Aktualisierungen beispielsweise jeden zweiten Dienstag im Monat und lediglich sicherheitskritische Aktualisierungen außerhalb dieses Plans. Viele Anwendungen und Betriebssysteme bieten darüber hinaus eine integrierte Aktualisierungsabfrage, die bei einer bestehenden Internetverbindung die Aktualität selbstständig prüft.

Diese Verfahren sollten jedoch mit Bedacht verwendet werden, da ein fehlerhaftes Update sich mitunter gravierend auf das System auswirken kann. Beispielsweise zog Microsoft im Dezember 2014 diverse Updates zurück, da sie Fehlfunktionen an den Systemen verursachten. So verursachte das entsprechende Update für den E-Mail- und Groupware-Server Exchange („KB2986475“) eine Störung, die dazu führte, dass keine E-Mails mehr versendet und Kalendereinträge erstellt werden konnten.

Daher sollte man Aktualisierungen vorab testen oder mindestens das Risiko für einen ungeprüften Einsatz gegenüber dem Risiko eines veralteten Systems abwägen. Die Virenschutzsoftware in einem großen Unternehmensnetzwerk erhält stündlich oder täglich entsprechende Aktualisierungen. Daher wäre der Aufwand, jede Aktualisierung vor deren unternehmensweitem Einsatz zu testen, für die meisten Unternehmen zu hoch.

Des Weiteren besteht ein hohes Risiko einer Infektion mit einer Schadsoftware, wenn der Virenschutz nicht regelmäßig aktualisiert wird. Daher sollte im Rahmen eines angemessenen *Patchmanagements* detailliert definiert werden, welche Systeme auf welche Weise aktualisiert werden.

Übung 1.3:

Nennen Sie die Prinzipien der Netzwerkverteidigung.

1.5 Grundlagen der Netzwerktechnik

Dieses Kapitel soll Ihnen eine kurze Übersicht über wichtige Grundbegriffe der Netzwerktechnik anhand des Open-Systems-Interconnection(▸OSI)-Modells geben.

Die International Organization for Standardization (▸ISO) veröffentlichte im Jahr 1977 das ▸OSI-Modell. Dieses Modell gilt als Referenzmodell für die Netzwerkarchitektur. Es besteht aus sieben Schichten, die einer vollständigen Beschreibung der Netzwerkkommunikation dienen. Die Schichten reichen von einem grundlegenden physikalischen Verbindungsaufbau (*Schicht 1: Physical Layer*) bis zur Übertragung komplexer Inhalte (*Schicht 7: Application Layer*).

1.5.1 Schicht 1: Physical Layer

Diese Schicht (Bitübertragungsschicht) dient der Beschreibung von physischen (elektrischen und mechanischen) Elementen in der Netzwerkkommunikation. Grundsätzlich kann auf dieser Schicht zwischen aktiven und passiven Netzwerkkomponenten unterschieden werden. Eine passive Netzwerkkomponente ist auf den folgenden Ebenen nicht sichtbar. Sie leitet die Netzwerkkommunikation lediglich weiter oder verstärkt die jeweiligen Signale.

Ein weiteres Entscheidungsmerkmal ist die verwendete Übertragungstechnik. Die elektrischen Signale können per Kupfer, Lichtwellenleiter oder auch per Funk übertragen werden. Die verwendete Übertragungsart ist ein grundlegender Faktor für die Zugriffssicherheit durch Unbefugte.

Eine Attacke auf einen Lichtwellenleiter ist beispielsweise mit einem höheren Aufwand für den Angreifer verbunden als die Attacke auf ein Funknetzwerk. Daher ist unter der Berücksichtigung der verarbeiteten sensiblen Daten der Einsatz von Verschlüsselung des Funknetzwerks höher zu priorisieren als der Einsatz einer Verschlüsselung des Lichtwellenleiternetzwerks.

1.5.2 Schicht 2: Data Link Layer

Der *Data Link Layer* oder auch Sicherungsschicht ist für die grundlegende Datenübertragung zuständig. Darüber hinaus wird auf dieser Schicht der Zugriff auf das Übertragungsmedium sowie der Umgang mit Kollisionen von Daten geregelt. Die typischen Protokolle, die auf dieser Schicht arbeiten, sind Ethernet, ▸IEEE 802.11 und Fiber Distributed Data Interface (▸FDDI).

Auf dieser Schicht erfolgt auch die logisch-technische Trennung von Netzwerken in sogenannte Virtual LAN (▸VLAN) sowie das Zusammenfassen der übertragenen Daten in verschiedene Pakete. Die Kommunikation auf dieser Schicht erfolgt über Media Access Control (▸MAC). Für die Regelung der Kollisionen werden verschiedene Verfahren verwendet. Bei Funknetzwerken kommt hierfür oftmals das Verfahren Carrier Sense Multiple Access/Collision Avoidance (▸CSMA/CA) zum Einsatz. Dieses regelt den Datenfluss, indem es Kollisionen von Datenpaketen vermeidet. Hierzu hört es das Medium ab, bevor es ein Datenpaket sendet, und überprüft, ob dieses empfangen wurde. Wenn eine Kollision während der Übertragung festgestellt wird, wird die Kommunikation für eine zufällige Zeitdauer ausgesetzt, bevor die Übertragung erneut gestartet wird.

Ein weiteres Verfahren für die Datenübertragung ist Carrier Sense Multiple Access/Collision-Detection (▸CSMA/CD). Das Verfahren prüft zunächst, ob derzeit eine Kommunikation auf einem Medium stattfindet. Wenn das Medium frei ist, wird die Übertragung gestartet. Dabei wird weiterhin überprüft, ob die Datenpakete erfolgreich übertragen werden oder kollidieren. Wenn eine Kollision stattfindet, wird ein Störsignal an alle anderen Netzwerkteilnehmer gesendet und ein interner Zähler hochgezählt. Danach pausiert die Datenübertragung eine zufällige Zeitdauer, bevor sie erneut gestartet wird, sofern der interne Zähler für ein definiertes Maximum nicht überschritten hat.

1.5.3 Schicht 3: Network Layer

Der *Network Layer* (Vermittlungsschicht) ist für das Routing der Datenübertragung verantwortlich. In dieser Schicht erfolgt die Kommunikation beispielsweise über Internet Protocol (▸IP), Internet Control Message Protocol (▸ICMP) und Internet Protocol Security (▸IPSec). Während die oben beschriebenen Schichten, der ersten Schicht des ebenfalls weitverbreiteten *TCP/IP-Modells* entsprechen, entspricht der *Network Layer* der zweiten Schicht (*Internet Layer*) dieses Modells.

1.5.4 Schicht 4: Transport Layer

Die vierte Schicht des Modells ist für die Stauvermeidung und die Segmentierung der Datenübertragung verantwortlich. Hier erfolgt die Kommunikation an einen bestimmten Port des Zielsystems. Die vom *Transport Layer* typischerweise genutzten Protokolle sind beispielsweise Transmission Control Protocol (▸TCP), User Datagram Protocol (▸UDP) oder auch Generic Routing Encapsulation (▸GRE). Im *TCP/IP-Modell* entspricht diese Schicht der dritten Schicht, die ebenfalls *Transport Layer* genannt wird.

1.5.5 Schicht 5: Session Layer

Der *Session Layer* wird auch Sitzungsschicht genannt. Die Sitzungsschicht ist für den Aufbau einer Sitzung zwischen zwei Systemen und deren gegenseitige Authentisierung verantwortlich. Die wohl am häufigsten genannte exemplarische Verwendung der fünften Schicht ist das Protokoll Secure Sockets Layer (▸SSL). Ein weiterer Dienst, der dieser Schicht zugeordnet werden kann, ist Remote Procedure Call (▸RPC).

Beispiel 1.1:

Ein bekanntes Beispiel für die Verwendung von ▸RPC ist das Protokoll Network-File-System (▸NFS). ▸NFS dient zum Austausch von Dateien über das Netzwerk. Im Rahmen von ▸NFS kontaktiert zunächst ein ▸RPC den Portmapper des ▸NFS-Servers und fragt nach dem Port des Bereitstellungsdienstes *Mount Daemon* „mountd“. Ein weiterer ▸RPC kontaktiert „mountd“ und ruft den *Filehandle* des gewünschten Verzeichnisses ab. Der Daemon gibt als *Filehandle*-Wert „0“ zurück.

Der Client fragt den Portmapper nach dem Port für ▸NFS-Daemon (nfsd) an. Dieser gibt die entsprechende Portnummer zurück. Als Nächstes führt der Client einen *LOOKUP*-RPC mit dem *Filehandle 0* und dem Dateinamen der Datei im aufgerufenen Verzeichnis durch. Der ▸NFS-Daemon gibt den entsprechenden Filehandle (*Filehandle 1*) für die Datei zurück. Der Client führt eine *READ*-RPC mit dem *Filehandle 1* aus. Der ▸NFS-Daemon gibt den Inhalt der Datei zurück.

Ein Filehandle ist ein Funktionsaufruf, der einen ganzzahligen Wert zurückgibt. Aus der Programmiersprache *C* haben sich die Rückgabewerte 0, 1 und 2 für solche Funktionsaufrufe als Standards durchgesetzt. Es existieren jedoch keine verbindlichen Vorgaben oder Standards.

Die Schicht 5 sowie die folgenden Schichten 6 und 7 des ▸OSI-Modells entsprechen der Schicht 4 des *TCP/IP-Modells*.

1.5.6 Schicht 6: Presentation Layer

Die Darstellungsschicht oder auch *Presentation Layer* bereitet die empfangenen Daten für die Darstellung auf. Hierzu zählt beispielsweise die Umwandlung der Daten in die systemabhängige Codierung. Weitere Aufgaben dieser Schicht sind Datenkompression und Verschlüsselung. Ein weitverbreitetes Verwendungsbeispiel für die Schicht 6 ist Multipurpose Internet Mail Extensions (▸MIME).

1.5.7 Schicht 7: Application Layer

Der *Application Layer* dient der programmabhängigen Datenein- und -ausgabe sowie der Verbindung der tieferen Schichten mit den entsprechenden Anwendungen.

Übung 1.4:

Auf welcher ▸OSI-Schicht arbeiten ▸CSMA/CA und ▸CSMA/CD?

1.6 Ports

Ein Port ist ein Teil der Netzwerkadresse und dient der genauen Zuordnung der Datenpakete. Ports dienen sowohl der Unterscheidung der Kommunikation zwischen Client und Server, sofern mehrere Datenverbindungen gleichzeitig aufgebaut werden, als auch der Identifikation des Dienstes.

Die Portnummern reichen von 0 bis 65 535. Die Ports zwischen 0 und 1 023 werden als *System Ports* bezeichnet und von der ▸IANA fest vergeben. Diese Ports werden oftmals auch „well known ports“ genannt. Die Ports im Bereich von 1 024 bis 49 151 werden *Registered Ports* genannt. Sie sind nicht von der Internet Assigned Numbers Authority (▸IANA) definiert. Sie können sowohl durch Systemdienste als auch von entsprechenden Anwendungen verwendet werden.

Der Bereich 49 152 bis 65 535 wird von Anwendungen individuell vergeben, wenn für eine Kommunikation weitere Ports benötigt werden. Daher werden die Ports in diesem Bereich *dynamische Ports* oder auch „kurzlebige Ports“ genannt.

Ein Port kann verschiedene Zustände einnehmen. Ein Port wird als *offen* bezeichnet, wenn ein Programm bereit ist, ▸TCP-Verbindungen oder ▸UDP-Pakete auf diesem Port anzunehmen. Beim Port-Scanning ist es oftmals das Ziel, solche Ports zu finden. Sicherheitsbewusste Menschen wissen, dass jeder offene Port eine breite Einfahrtstraße für Angriffe darstellt. Angreifer und Penetrationstester wollen offene Ports ausbeuten, während Administratoren versuchen, sie zu schließen oder mit Firewalls zu schützen, ohne legitime Benutzer zu behindern. Offene Ports sind auch für Scans von Interesse, bei denen es nicht um Sicherheit geht, weil sie Dienste anzeigen, die im Netzwerk benutzt werden können.

Ein *geschlossener* Port ist erreichbar, aber es gibt kein Programm, das ihn abhört. Er kann von Nutzen sein, um zu zeigen, dass ein Host online ist und eine ▸IP-Adresse benutzt (Host-Erkennung oder Ping-Scanning), sowie als Teil der Betriebssystemerkennung. Weil geschlossene Ports erreichbar sind, sind sie es wert, gescannt zu werden, falls sie später einmal geöffnet werden sollten.

Ein *gefilterter Port* wird durch eine Firewall oder durch einen Router geschützt, der die Datenpakete verwirft, ohne dem Sender eine entsprechende Rückmeldung zu geben. Weil sie so wenig Information bringen, sind diese Ports für Angreifer frustrierend. Eine Filterung verlängert die Dauer eines Scans erheblich, da eine Vielzahl der Pakete mehrmals versendet werden müssen, um Klarheit über den Zustand des Ports zu erhalten.

Abb. 1.4 zeigt eine Übersicht der geöffneten Ports an einem Beispielsystem.

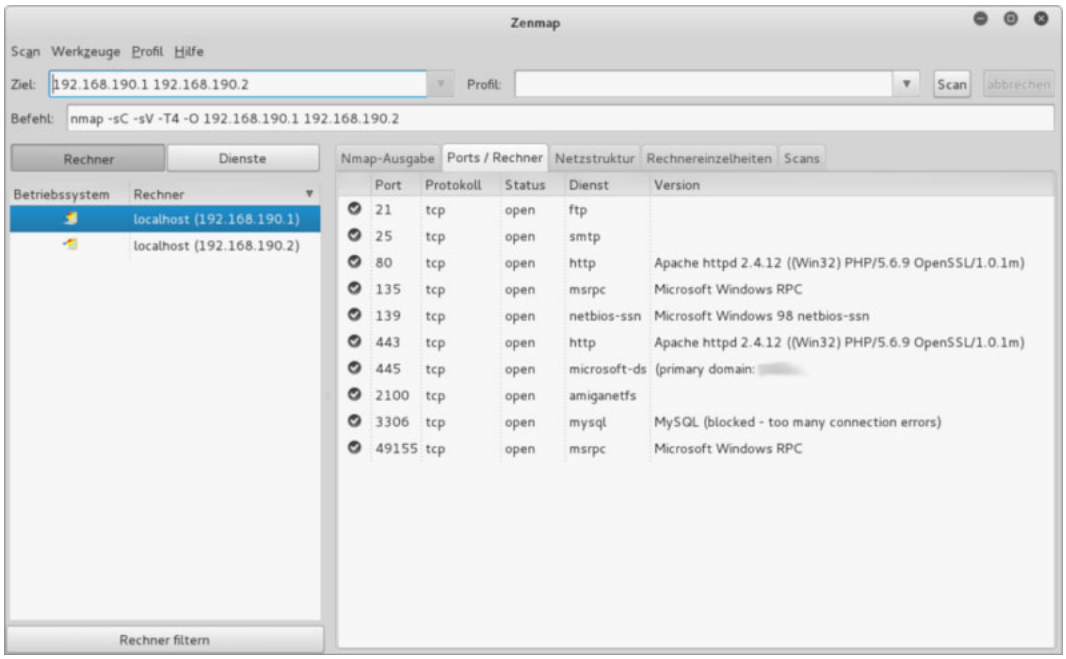

Abb. 1.4: Geöffnete Ports an einem System

Zusammenfassung

Während IT-Systeme einer hohen Änderungshäufigkeit unterliegen, sind die Grundprinzipien zur Konzeption sicherer Systeme seit 1975 unverändert gültig. Letztendlich sollten Funktionen und Berechtigungen stets restriktiv behandelt werden und die Sicherheit eines Systems niemals auf Geheimhaltung beruhen, sondern durch eine transparente Architektur eine Begutachtung und Qualitätssicherung durch andere Benutzer ermöglichen.

Die Kompromittierung eines Systems erfolgt in mehreren logischen Schritten, die fließend ineinander übergehen können. Der wichtigste Punkt hierbei ist die Informationssammlung, also die Suche nach möglichen Ansatzpunkten und Schwachstellen. Daher wirkt das Prinzip des offenen Entwurfs als Schutzmaßnahme etwas gegensätzlich, jedoch wissen Sie bei einem geheim gehaltenen Systementwurf nicht, ob dieses System bereits Hintertüren, wie beispielsweise „Wartungszugänge“ oder Standardpasswörter, enthält. Die Vergangenheit hat gezeigt, dass solche Hintertüren immer wieder bekannt wurden und eine enorme Bedrohung darstellen. Sie werden dann zumeist als eine 0Day-Schwachstelle behandelt.

Die Vergleichbarkeit und Bewertung von Schwachstellen ist für den einzelnen Administrator oftmals schwierig, vor allem wenn mehrere unterschiedliche Schwachstellen vorhanden sind, deren Beseitigung mit einem hohen Aufwand verbunden ist. Zur Erhöhung der Vergleichbarkeit von Schwachstellen kann der ▸CVSS-Ansatz verwendet werden. Darüber hinaus wird durch die Verwendung von ▸CVE eine einheitliche Datenbasis zur Kommunikation von Schwachstellen bereitgestellt.

Aufgaben zur Selbstüberprüfung

1.1 Nennen Sie die Prinzipien zur Konzeption sicherer Systeme.

1.2 Nennen Sie die Phasen der Kompromittierung.

1.3 Nennen Sie die drei Faktorengruppen der ▸CVSS-Bewertungsmethode.

1.4 Nennen Sie die Prinzipien der Netzwerkverteidigung.

2 Erkennen von Netzwerkangriffen

Dieses Kapitel soll Ihnen einen Überblick über den Aufbau eines ▸IDS geben und Ihnen die Vor- und Nachteile der verschiedenen Funktionsweisen und Systeme erläutern.

Am Ende des Kapitels kennen Sie die Standards und Sprachen zum Austausch von ▸IDS-Komponenten sowie den Unterschied zwischen hostbasierten und netzbasierten Systemen.

Den Abschluss des Kapitels bildet ein Definitionsansatz für „Attacken" in diesem Kontext.

2.1 Intrusion-Detection-Systeme

Das Ziel eines Intrusion-Detection-Systems (▸IDS) besteht in der frühzeitigen Erkennung von Angriffen, um daraus resultierende Schäden zu minimieren und den Angreifer schnell zu identifizieren. Darüber hinaus erlauben ▸IDS das Sammeln von Informationen über neue Angriffstechniken, die zur Verbesserung von präventiven Maßnahmen genutzt werden können.

Die Forschungsabteilung Defense Advanced Research Projects Agency (▸DARPA) des amerikanischen Verteidigungsministeriums initiierte im Jahr 1999 ein Projekt, das einen standardisierten Austausch von verschiedenen ▸IDS zum Ziel hatte. Das Ergebnis ist das Common Intrusion Detection Framework (▸CIDF), das vier grundlegende Komponenten eines ▸IDS beschreibt:

- Ereigniskomponenten
- Analysekomponenten
- Datenbankkomponenten
- Reaktionskomponenten

Eine **Ereigniskomponente** stellt Informationen über das zu schützende System bereit. Hierunter kann beispielsweise das Protokollieren sicherheitsrelevanter Aktivitäten, wie der Verbindungsaufbau über einen falschen Netzwerkport oder ein fehlgeschlagener Anmeldeversuch, verstanden werden.

Eine **Analysekomponente** analysiert und bewertet die durch die Ereigniskomponente protokollierten Informationen. In der Analysekomponente erfolgt die Erkennung des Angriffs.

Die Informationen zur Analyse sowie deren Zwischenergebnisse werden in der **Datenbankkomponente** gespeichert.

Die **Reaktionskomponenten** beinhalten die Behandlungsmethoden zur Begegnung des Angriffs (z. B. Versand einer Alarm-E-Mail).

2.2 Gesetzliche Rahmenbedingungen

Der Einsatz eines ▸IDS kann zu Konflikten mit bestehenden gesetzlichen Vorgaben führen, da mit seinem Einsatz eine Vielzahl von Informationen erhoben werden, die auch die Privatsphäre des Einzelnen berühren können.

Ein klassisches Beispiel ist die Protokollierung und die damit verbundene Auswertung des „Surfverhaltens“ der Mitarbeiter. Ein ▸IDS „erfährt“ beispielsweise, welche Website von welchem Mitarbeiter oder welchem System aufgerufen und welche Inhalte darauf betrachtet wurden. Hat eine Organisation die private Nutzung des Internets zugelassen, verstößt diese Datensammlung gegen geltende Vorschriften, wie beispielsweise die Vorgaben zur Datensparsamkeit des Bundesdatenschutzgesetzes (▸BDSG) sowie den Grundsatz zur Transparenz, der sich im Telemediengesetz (▸TMG) und im Telekommunikationsgesetz (▸TKG) wiederfindet.

2.3 Standards zum Austausch von IDS-Komponenten

Im Zusammenhang mit einem ▸IDS wird eine Vielzahl von Informationen analysiert und weiterführend aufbereitet. Im Rahmen verschiedener wissenschaftlicher, aber auch technischer Ansätze wurden dabei unterschiedliche Sprachen entwickelt, die eine systemunabhängige Darstellung des jeweiligen Sachverhalts ermöglichen. Die Ansätze unterscheiden grundsätzlich zwischen folgenden Sprachen:

- Attackensprachen (Attack Languages)
- Exploit-Sprachen (Exploit Languages)
- Ereignissprachen (Event Languages)
- Erkennungssprachen (Detection Languages)
- Reaktionssprachen (Response Languages)
- Reportsprachen (Report Languages)
- Korrelationssprachen (Correlation Languages)

2.3.1 Attackensprachen

Die *Attackensprachen* ordnen die Art der Attacke in unterschiedliche Klassen ein. Ein Beispiel für die Darstellung von Attacken mittels Graphentheorie ist State Transition Analysis Technique Language (▸STATL). Die zentralen Begrifflichkeiten in diesem Zusammenhang sind Szenarien, Zustände und Transitionen (Übergänge).

2.3.2 Exploit-Sprachen

Bei *Exploit-Sprachen* steht die Codierung der Aktionen, die zur Durchführung einer Attacke erforderlich sind, im Vordergrund. Dies sind typischerweise Programmier- und Skriptsprachen wie beispielsweise Hypertext-Preprocessor (▸PHP), C oder Perl.

2.3.3 Ereignis- und Erkennungssprachen

Die Beschreibung der Ereignisse, die von einem ▸IDS analysiert werden sollen, werden in Ereignissprachen dargestellt. Ein typisches Beispiel hierfür sind Spezifikationen von *TCPDUMP*-Daten.

Die ▸IDS-Signaturen werden unter Verwendung von *Erkennungssprachen* spezifiziert.

2.3.4 Reaktionssprachen

Mithilfe von *Reaktionssprachen* werden Aktionen beschrieben, die im Fall einer erkannten Attacke als Gegenmaßnahme einzuleiten sind. Eine Vielzahl von ▸IDS-Systemen verwendet hierfür häufig Bibliotheksfunktionen, die in Programmiersprachen wie C entwickelt wurden.

2.3.5 Reportsprachen

Die *Reportsprachen* beschreiben Formate von Alarmmeldungen, die Informationen zu erkannten Angriffen enthalten. Ein Beispiel hierfür ist das Verfahren Intrusion Detection Message Exchange Format (▸IDMEF), das von der Internet-Engineering-Task-Force (▸IETF) im Jahr 2007 als Standardverfahren anerkannt wurde. ▸IDMEF ist ein Extensible-Markup-Language (XML)-strukturiertes, objektorientiertes Format, das 33 verschiedene Klassen zur Beschreibung einer Meldung enthält. Das Format unterstützt die Varianten Alarmmeldung und „Heartbeat“-Meldung. Beispiel 2.1 zeigt den Quelltext einer beispielhaften ▸IDMEF-Meldung.

Beispiel 2.1:

```
<?xml version="1.0" encoding="UTF-8"?>
<idmef:IDMEF-Message xmlns:idmef="http://iana.org/idmef"
version="1.0">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-sensor01">
      <idmef:Node category="dns">
        <idmef:name>ids1.wb-hochschule.de</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc71f4f5.0xef449129">2016-10-
    09T10:01:25.93464Z
    </idmef:CreateTime>
    <idmef:Source ident="a1a2" spoofed="yes">
      <idmef:Node ident="a1a2-1">
        <idmef:Address ident="a1a2-2" category="ipv4-addr">
          <idmef:address>192.0.1.231</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Source>
    <idmef:Target ident="b3b4">
      <idmef:Node>
        <idmef:Address ident="b3b4-1" category="ipv4-addr">
          <idmef:address>192.0.2.50</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Target>
```

```
    <idmef:Target ident="c5c6">
      <idmef:Node ident="c5c6-1" category="nisplus">
        <idmef:name>lollipop</idmef:name>
      </idmef:Node>
    </idmef:Target>
    <idmef:Target ident="d7d8">
      <idmef:Node ident="d7d8-1">
        <idmef:location>Cabinet B10</idmef:location>
        <idmef:name>Cisco.router.b10</idmef:name>
      </idmef:Node>
    </idmef:Target>
    <idmef:Classification text="Ping-of-death detected">
      <idmef:Reference origin="cve">
        <idmef:name>CVE-1999-128</idmef:name>
          <idmef:url>http://www.cve.mitre.org/cgi-bin/
          cvename.cgi?name=CVE-1999-128</idmef:url>
      </idmef:Reference>
    </idmef:Classification>
  </idmef:Alert>
</idmef:IDMEF-Message>
```

Eine ▸IDMEF-Meldung beinhaltet verschiedene Informationen, die in verschiedenen wohlgeformten Elementen hinterlegt sind. Dabei wird zwischen obligatorischen und optionalen Elementen unterschieden. Die eindeutigen Identifikationsmerkmale zählen beispielsweise zu den obligatorischen Elementen. Hierzu gehören ein eindeutiges Merkmal zur Bestimmung des meldenden ▸IDS, die sogenannte *Analyser-ID*, sowie die Unterscheidung, ob es sich bei der Meldung um eine Alarmmeldung (*Alarm Message*) oder eine regelmäßige Statusmeldung (*Heartbeat Message*) handelt. Des Weiteren sind die Elemente *CreateTime*, *Classification*, *Source*, *Target*, *Node* und *Address* für eine Alarmmeldung ebenfalls obligatorisch. Sie beschreiben die grundlegenden Informationen zu den Quellen, den Zielen und die Art des Angriffs.

2.3.6 Korrelationssprachen

In Abhängigkeit des Betrachtungsgegenstands können *Reportsprachen* auch als Ereignissprachen verwendet werden, um Zusammenhänge bei komplexen oder verteilten Angriffen zu erkennen. Die Analyse der Zusammenhänge erfolgt mithilfe von *Korrelationssprachen*. Eine *Korrelationssprache* ist also eine *Reportsprache* auf einer höheren Abstraktionsebene.

2.4 Ereigniskomponente

Für die Erhebung von Daten mit einem angemessenen Informationsgehalt ist maßgeblich die Einstellung der Ereigniskomponente verantwortlich. Bei einer falsch eingestellten Ereigniskomponente besteht die Gefahr, dass wir Sicherheitsvorfälle nicht erkennen, weil wir sie in einer Informationsflut übersehen oder weil zu wenig Informationen erhoben werden, um sie zu erkennen. Beide Extreme sind unbedingt zu vermeiden!

Daher bieten ▸IDS oftmals eine Vielzahl von Konfigurationsmöglichkeiten, um sie an ihren Einsatzzweck anzupassen. Des Weiteren sollte kein ▸IDS in Betrieb genommen werden, ohne sich im Vorfeld mit diesen Konfigurationsmöglichkeiten zu beschäftigen.

Grundsätzlich können wir zwischen zustandsbasierten und aktionsbasierten Ereignissen unterscheiden. Ein zustandsbasiertes Ereignis fragt regelmäßig Informationen über Auslastung und Ressourcennutzung eines Systems für eine spätere Analyse ab. Eine aktionsbasierte Ereigniskomponente zeichnet dagegen sicherheitsrelevante Aktivitäten des Systems auf. Bei dieser Ereigniskomponente stehen insbesondere Aussagen zu den Fragen

- Wer?
- Wann?
- Welche Aktion?
- Erfolgreich?

im Vordergrund. Diese Aussagen werden oftmals durch weitere Detailinformationen ergänzt.

2.5 Analyse- und Datenbankkomponente

Die Erkennung des sicherheitsrelevanten Vorfalls findet in der Analysekomponente statt. Hierzu werden die erfassten Daten mit Daten aus der Datenbankkomponente verglichen und als Ergebnis eine entsprechende Bewertung erstellt.

Die Bewertungen der Analyse können dabei wie folgt klassifiziert werden:

- True Positives: Eine Sicherheitsverletzung wurde korrekt klassifiziert.
- True Negatives: Eine sicherheitskonforme Handlung wurde korrekt klassifiziert.
- False Positives: Eine sicherheitskonforme Handlung wurde als Sicherheitsverletzung klassifiziert.
- False Negatives: Eine Sicherheitsverletzung wurde als sicherheitskonform klassifiziert.

Für die Erzielung einer angemessenen Aussagekraft eines ▸IDS ist die Minimierung der False-Positive- und False-Negative-Meldungen unerlässlich. Ein ▸IDS, das regelmäßig False Positive (Fehlalarme) meldet, wird im Zeitverlauf dazu führen, dass Meldungen nicht mehr ausgewertet und berücksichtigt werden. Hierdurch besteht die Gefahr, dass ein echter Alarm von einer Vielzahl von Fehlalarmen überlagert wird und unerkannt bleibt.

Ein ▸IDS, das Sicherheitsverletzungen nicht meldet, ist unwirksam (False Negatives).

Übung 2.1:

Welche Funktion hat die Ereigniskomponente in einem ▸IDS?

2.6 Erkennungsmethoden

Die Erkennungsmethoden bei einem ▸IDS sind entscheidend für seine Wirksamkeit und sind Teil der Analyse- und Datenbankkomponente des ▸IDS. Man unterscheidet bei den Erkennungsmethoden zwei grundsätzliche Vorgehensweisen:

- Anomalieerkennung
- Signaturanalyse

Bei der Anomalieerkennung bewertet das ▸IDS Abweichungen von einem Normalzustand als einen Sicherheitsvorfall. Die Datenbankkomponente enthält dabei Informationen zu den normalen Verhaltensweisen des zu überwachenden Systems. Alle Abweichungen hiervon werden als Sicherheitsvorfall klassifiziert.

2.6.1 Anomalieerkennung

Die normalen Verhaltensweisen können dem ▸IDS dabei entweder durch Messen bzw. Erlernen oder durch Spezifizieren bekannt gemacht werden. Beim Verwenden der Erkennungsmethode Erlernen sollte diese „Lernphase“ in regelmäßigen Zeitabständen wiederholt werden, da sich die Verhaltensweisen der Benutzer im Zeitverlauf ebenfalls ändern. Daher besteht bei dieser Erkennungsmethode die Gefährdung, fehlerhafte Verhaltensweisen in diesen Phasen zu „erlernen“ und zukünftig als sicherheitskonform zu bewerten.

Die spezifikationsbasierte oder auch statistische Anomalieerkennung verwendet formalisierte Spezifikationen des Benutzer- oder Systemverhaltens. Hierzu werden zunächst die Zustände definiert, die sicherheitskonform sind. In einer Vielzahl von ▸IDS-Produkten werden hierzu explizit sicherheitskonforme Handlungen vorab im System erlaubt (z. B. auf Protokollebene).

Ein nicht freigegebener Zustand wird damit als sicherheitsrelevant gemeldet. Auch hier besteht die Gefährdung, dass das ▸IDS „False Positive“ und „False Negative“ meldet, sofern es nicht regelmäßig gepflegt wird.

2.6.2 Signaturanalyse

Die Signaturanalyse sucht anhand definierter Angriffsmuster, sogenannter Signaturen, nach konkreten Sicherheitsverletzungen. Sie wird oftmals auch als Missbrauchserkennung bezeichnet. Wenn die aufgezeichnete Information mit einem dieser Muster übereinstimmt, wird sie als Sicherheitsverletzung gewertet.

Eine Sicherheitsverletzung kann grundsätzlich in zwei Klassen unterteilt werden. Eine *Single Step Attack* (Einzelschrittattacke) wird anhand eines einzelnen Protokolleintrags erkannt. Die Signaturen einer solcher Attacke beschreiben typischerweise charakteristische Byte-Sequenzen. Wenn eine entsprechende Kombination in einem Protokolleintrag festgestellt wird, deutet dies auf eine Sicherheitsverletzung hin.

Eine *Multi Step Attack* (Mehrschrittattacke) wird erst durch die Analyse von charakteristischen Merkmalen in mehreren Protokolleinträgen festgestellt. Dementsprechend sind *Single Step* ▸*IDS* ausschließlich in der Lage, Einzelschrittattacken zu erkennen. Ein *Multi Step* ▸*IDS* ist darüber hinaus auch in der Lage, eine Mehrschrittattacke zu erkennen.

Übung 2.2:

Was ist ein „False Positive“ bei einem ▸IDS?

2.6.3 Netzbasierte und hostbasierte IDS

Bei ▸IDS wird zwischen netzbasierten und hostbasierten ▸IDS unterschieden. Ein netzbasiertes ▸IDS analysiert protokollierte Netzwerkpakete, während ein hostbasiertes ▸IDS Ereignisse auf Betriebssystem- oder Anwendungsebene überprüft.

Für eine Auswertung auf eine Einbruchserkennung ist ein hostbasiertes Protokoll häufig hochwertiger und detaillierter. Der Konfigurationsaufwand für die Protokollauswertung bei einem hostbasierten ▸IDS ist deutlich höher. Hier ist ein entsprechender Aufwand bei jedem Client zu leisten mit entsprechend negativen Auswirkungen auf die Leistungsfähigkeit des Netzes und der Clients.

Ein netzbasiertes ▸IDS wird zentral im geschützten Netzwerk platziert und analysiert den Datenverkehr. Die Leistungsfähigkeit der Clients wird mit der Verwendung einer netzbasierten Variante nicht beeinträchtigt.

Aufgrund der vergleichsweise guten Erkennungsgenauigkeit sowie der einfach zu realisierenden Analyseverfahren und der einfachen Installation und Konfiguration sind netzwerkbasierte ▸IDS zur Erkennung von *Single Step Attacks* der am häufigsten verwendete Ansatz zur Sicherung des Netzwerks.

2.6.4 Attacke im IDS-Kontext

Eine Attacke repräsentiert eine bestimmte Menge an Aktionen und Ereignissen in einem IT-System, die der Sicherheitspolitik zuwiderlaufen. Im ▸IDS-Kontext kann es sich dabei um eine definierte Folge von Systemrufen oder Netzwerkpaketen handeln.

Ein Missbrauchserkennungssystem versucht, Sequenzen von Protokolleinträgen, die mit Signaturen von definierten Sicherheitsverletzungen übereinstimmen, zu erkennen. Eine grundlegende Voraussetzung ist daher, dass sich diese Sicherheitsverletzungen in den Protokolleinträgen manifestieren, also protokolliert werden können.

Die bei einer Attacke protokollierten charakteristischen Spuren in den Protokolleinträgen werden *Manifestierung* genannt.

Es ist möglich, dass mehrere Attacken eines Typs gleichzeitig und unabhängig voneinander fortschreiten, beispielsweise indem sie parallel von verschiedenen Angreifern durchgeführt werden. Deshalb ist es erforderlich, von *Instanzen einer Attacke* zu sprechen, deren Manifestierungen sich in der Ausprägung einzelner Merkmale, z. B. dem Benutzernamen, unterscheiden. Eine *Signaturinstanz* beschreibt die Menge der Kriterien, die eindeutig die Manifestierung einer Attackeninstanz im Protokoll-Datenstrom identifiziert.

Übung 2.3:

Welches ist die am häufigsten eingesetzte ▸IDS-Variante?

2.7 Platzierung der Sensoren

Die Platzierung der Sensoren und der Auswertungsstation ist ein wesentlicher Aspekt bei der Einführung eines ▸IDS. Je nach der Platzierung liefert der Sensor mehr oder weniger Informationen oder kann eventuell selbst Ziel für eine Attacke werden. Ein netzbasierter Sensor sollte daher nach folgenden Aspekten platziert werden:

- Erfüllung der formulierten Einsatzziele des ▸IDS (z.B. Was soll überwacht werden?)
- Erzielung von Synergieeffekten (z.B. Wie kann mit wenig Sensoren ein möglichst großer Netzbereich überwacht werden?)
- Effizienz der Netzwerküberwachung (z.B. Erlauben die Informationen eine regelmäßige Auswertung?)

Bei hostbasierten Sensoren empfiehlt sich eine mögliche Platzierung bei Systemen am Übergang zwischen externem und internem Netzwerk. Da hostbasierte Sensoren stets zusätzlich auf einem System eingesetzt werden, sollte berücksichtigt werden, dass die Sensoren den Prozessor und den Arbeitsspeicher des Systems beanspruchen. Daher sollten entsprechende zusätzliche Kapazitäten bei der Beschaffung des Systems eingeplant werden.

Bei der Platzierung der Auswertungsstation ist zu berücksichtigen, dass diese mit den Sensoren und anderen relevanten IT-Systemen (z.B. E-Mail-Server) kommunizieren können. Des Weiteren muss eine Kommunikation zwischen Auswertungsstation und Clients, die auf die Auswertungen zugreifen, möglich sein.

2.8 ID-Systeme und -Anwendungen

Es existiert eine Vielzahl verschiedener Intrusion-Detection-Systeme und Anwendungen am Markt. Die Systeme können dabei sowohl hostbasiert als auch netzbasiert eingesetzt werden. Darüber hinaus existieren proprietäre wie auch quelloffene System- und Anwendungsvarianten.

Ein weitverbreitetes System für ein netzbasiertes ▸IDS ist das quelloffene *Snort*. *Snort* wurde im Jahr 2001 durch den amerikanischen Programmierer Martin Roesch entwickelt, der hierfür das Unternehmen Sourcefire, Inc. gründete. Das Unternehmen wurde im Jahr 2013 durch den Konzern Cisco Systems, Inc. übernommen.

In seiner grundlegenden ▸IDS-Funktionsweise arbeitet *Snort* mit Signaturen, mit denen es übertragene ▸IP-Pakete auf potenzielle Bedrohungen überprüft. Diese Signaturen bezeichnet man im Anwendungskontext als Rules. Die einzelnen Rules können dabei durch einen Benutzer in den Prüfungsvorgang einbezogen oder von diesem ausgenommen werden. Darüber hinaus können verschiedene Rules auch individualisiert werden.

Für die Rules werden täglich Aktualisierungen bereitgestellt, die per Transport-Layer-Security(▸TLS)-geschützte Verbindung auf *Snort*-Systeme übertragen werden. Der Aktualisierungsmechanismus von *Snort* wird *PulledPork* genannt.

Beispiel 2.2:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
  (
   msg: "ICMP PING NMAP";
   dsize: 0;
   itype: 8;
   reference: arachnids, 162;
   classtype: attempted-recon;
   sid: 469;
   rev: 3;
  )
```

Ein einfache individualisierte Regel zeigt Beispiel 2.2. Das Beispiel sendet einen Alarm bei einem ▸ICMP-Paket, das von einem beliebigen Quellport („any“) des externen Netzwerks („EXTERNAL_NET“) an einen beliebigen Zielport im internen Netzwerk („HOME_NET“) gesendet wird und die Datengröße 0 („dsize“) besitzt und dem ▸ICMP-Typ 8 („itype“) entspricht.

Die ▸ICMP-Typen sind in dem Internet-Standard Request for Comments (▸RFC) 792 definiert. Der Typ 8 entspricht einem ECHO-Paket („ping“). Der Alarm enthält die Beschreibung („msg“) „ICMP PING NMAP“. Die Eigenschaft „classtype“ bezeichnet die Angriffsklasse. *Snort* besitzt eine Vielzahl von grundsätzlich definierten Angriffsklassen, die eine Meldung priorisieren und einordnen können. Die Angriffsklassen lassen sich durch den Benutzer ebenfalls individualisieren.

Das System bietet darüber hinaus noch verschiedene Intrusion-Prevention-System (▸IPS)-Funktionen, wie beispielsweise das Blockieren von verdächtigen Paketen.

Ein weiteres, weitverbreitetes ▸IDS ist *Astaro Security Gateway*. Das System ist proprietär und baut auf einem entsprechend gehärteten Linux-Kernel auf. Es besitzt sowohl Funktionen zur Missbrauchs- als auch zur Anomalieerkennung der typischen Netzwerkprotokolle. Des Weiteren verfügt es über Firewall- und Protokollierungsfunktionen. Die Protokollierungsfunktionen sind jedoch lediglich passiv. Einer protokollierten Bedrohung können also keine Maßnahmen gegenübergestellt werden. Ein Nachteil des Systems ist die Aufbewahrung der Protokolldateien. Diese können lediglich unverschlüsselt gespeichert werden.

Ein weiteres proprietäres *IPS* ist Cisco Security Agent (▸CSA). ▸CSA wurde ursprünglich durch die amerikanische Firma Okena, Inc. unter dem Namen Storm Watch Agent entwickelt. Okena wurde im Jahr 2003 durch Cisco übernommen und das ▸IPS dem Produktportfolio von Cisco hinzugefügt. Im Juni 2010 verkündete Cisco, dass die Weiterentwicklung von ▸CSA eingestellt wird. Das Produkt wird nicht länger gepflegt, wird jedoch in vielen mittelständischen Unternehmen aufgrund seiner hohen Interoperabilität mit anderen Cisco-Komponenten in Deutschland immer noch eingesetzt.

Das System besteht aus zwei Hauptkomponenten. Die Komponente Management Center ist für die Regelverwaltung und eine zentrale Protokollierung zuständig. Die Verwaltung dieser Komponente erfolgt über den Webbrowser. Die Komponente Host Agent ist für die Umsetzung der eingestellten Regeln zuständig und auf jedem zu schützenden System installiert.

Die Anwendung *fail2ban* ist eine weitverbreitete, quelloffene hostbasierte ▸IPS-Anwendung zum Schutz von Linux-Systemen vor Brute-Force- und Wörterbuch-Attacken. *fail2ban* überwacht die Protokolle des Linux-Systems auf vordefinierte Einträge und sperrt die ▸IP-Adresse der Quelle bei einer Regelverletzung. Die Regeln müssen durch den Benutzer selbstständig definiert werden. Die Anwendung sichert lediglich das lokale System und kein Netzwerksegment ab.

Für die Konfiguration der Anwendung erfolgt in den Dateien *jail.local* und *jail.conf*. Beispiel 2.3 zeigt eine exemplarische Konfiguration zur Überwachung der Dienste *ssh* und *http* auf fehlerhafte Logins in der Datei *jail.local*.

Beispiel 2.3:

```
[ssh]
         enabled = true
         port = ssh
         filter = sshd
         logpath = /var/log/auth.log
         maxretry = 6
[apache]
         enabled = true
         port = http, https
         filter = apache-auth
         logpath = /var/log/apache*/*error.log
         maxretry = 6
```

Die ▸IP-Adresse wird nach sechs fehlerhaften Zugriffsversuchen auf den Ports 22 (ssh), 80 (http) und 443 (https) gesperrt. Als Quelle werden die jeweiligen Systemprotokolle (logpath) verwendet. Die Anwendung bietet damit einen einfachen und zuverlässigen Schutz für einzelne Systeme, ist jedoch von seinem Funktionsumfang nicht mit einem proprietären, netzbasierten ▸IPS vergleichbar.

2.9 Honeypots

Ein *Honeypot* ist ein präpariertes System oder eine spezielle Umgebung, die eine oder mehrere Schwachstellen simuliert. Das Ziel eines Honeypots ist es, einen Angreifer von anderen Systemen im Netzwerk abzuhalten. Des Weiteren sollen Methoden, Vorgehensweisen und Strategien von Angreifern untersucht werden, um neue Attacken und Trends zu erkennen.

Es existiert eine Vielzahl von Vorgehensweisen, einen Honeypot in ein Netzwerk zu integrieren. Grundsätzlich kann man zwischen

- *Low Interaction Honeypots* und
- *High Interaction Honeypots*

unterscheiden. Des Weiteren unterscheidet man zwischen Client- und Server-Anwendungen. Ein *Low Interaction* Server Honeypot simuliert die Bereitstellung von Serverdiensten und dient vor allem zur Informationsgewinnung über automatisierte Angriffe. Ein versierter Angreifer wird auf einen solchen Honeypot nicht lange hereinfallen.

Eine weitverbreitete Anwendung für einen *Low Interaction Server* Honeypot unter Linux ist *honeyd*. *honeyd* simuliert gängige Serverdienste auf definierten ▸IP-Adressen.

Die Zugriffsversuche auf die Serverdienste werden in eine Protokolldatei zur späteren Analyse geschrieben. Beispiel 2.4 zeigt einen Auszug einer Protokolldatei von *honeyd* bei der Simulation eines ▸FTP-Dienstes.

Beispiel 2.4:

```
2004-01-07-14:36:58.7132 tcp(6) - 252.214.169.203 2064
192.168.27.180 21: 48 S [MacOS 8.0-8.6 OTTCP]
2004-01-07-15:26:40.0209 tcp(6) - 244.233.22.102 61891
172.162.8.180 21: 60 S [FreeBSD 5.0-5.1]
2004-01-07-16:48:30.1212 tcp(6) S 192.168.21.135 33395
172.162.8.91 80 [Linux 2.6]
2004-01-07-16:48:41.4929 tcp(6) S 10.173.240.67 22110
192.168.14.178 81 [Windows XP SP1]
```

Ein *Low Interaction* Client ist eine Anwendung, die einen Zugriff auf einen echten Server als Anwendung simuliert, um einen Angriff auf den Client erkennen zu können. Ein weitverbreitetes Beispiel für einen solchen Honeypot ist das Mozilla-Projekt *Spider Monkey*.

Ein *High Interaction* Server ist ein Honeypot, der manuell gesteuerte Angriffe protokolliert und die Vorgehensweise des Angreifers analysiert. Hierzu simuliert der Honeypot zumeist einen zentralen und hochschutzbedürftigen Serverdienst, wie beispielsweise einen Datenbankserver.

Ein *High Interaction* Client überwacht eine Client-Anwendung, wie beispielsweise einen Browser, auf Angriffe. Diese Honeypots arbeiten meistens auf einem gängigen Betriebssystem mit „echten“ (also nicht simulierten) Anwendungen. *High Interaction Clients* werden meistens zur Überprüfung von Websites auf Bedrohungen, wie Drive-by-Downloads, eingesetzt.

2.10 Security Information and Event Management

Security-Information-and-Event-Management(▸SIEM)-Systeme sind sozusagen Data-mining-Systeme, die Meldungen anderer IT-Systeme mit ihren jeweiligen Metadaten sammeln und auswerten. Die Sammlung eines ▸SIEM-Systems umfasst dabei sowohl technische Meldungen, wie beispielsweise Syslog-Einträge und Meldungen des Virenschutzes, als auch organisationsindividuelle Informationen (z.B. Schutzbedarf, Kritikalität). Hierdurch sollen Bedrohungen und Störungen frühzeitig erkannt werden, sodass zeitnah Gegenmaßnahmen eingeleitet werden können.

Die Vorläufer aktueller ▸SIEM-Systeme waren sogenannte Security-Information-Management(▸SIM)- und Security-Event-Management(▸SEM)-Systeme. Ein ▸SIM verarbeitet und aggregiert gesammelte Protokollinformationen zur Erkennung von Trends. Ein ▸SEM korreliert Echtzeitdaten aus diversen IT-Systemen mit vorhandenen Regeln und meldet sicherheitsrelevante Ereignisse.

Die Qualität eines ▸SIEM-Systems lässt sich grundlegend durch drei Faktoren bestimmen:

- Anzahl der Events pro Sekunde, die es verarbeiten kann
- Konfigurationsmöglichkeit der Attribute
- Logik des Regelprozessors

Wie bei einem ▸IPS oder ▸IDS erfordert ein ▸SIEM zunächst eine Individualisierung durch den Benutzer, um die Aussagekraft eines Systems zu gewährleisten. Vor der Einführung sollte zunächst geklärt werden, welche Ziele die Organisation durch den Einsatz eines ▸SIEMs erreichen möchte. Als typische Ziele können beispielsweise Transparenz und Compliance-Fähigkeit genannt werden. Danach sollte festgelegt werden, welche Daten die Software überwachen und sammeln soll sowie nach welchen Kriterien dies entschieden wird.

Ein Vorgehen nach dem Motto „viel hilft viel“ wird bei der Konfiguration eines ▸SIEMs nicht sehr hilfreich sein, da hierdurch sehr schnell die Grenzen der Verarbeitungskapazitäten des Systems erreicht werden oder relevante Informationen in der Datenflut untergehen.

Zusammenfassung

Die Gruppe der ▸IDS-Systeme ist vielfältig und das Erkennen von Anomalien im Netzwerk ist keine neue Disziplin. Mit steigender Vernetzung der Systeme wird sie doch zunehmend wichtiger und gehört heutzutage zur Grundausstattung der Netzwerksicherheitssysteme.

Der Nutzen eines ▸IDS hängt von dessen Ereignis- und Analysekomponente ab. Die korrekte Konfiguration des Systems ist die Voraussetzung zum Erkennen von Anomalien. Die Konfiguration ist kein einmaliger, sondern ein kontinuierlicher Prozess, der mit entsprechenden Ressourcen hinterlegt werden sollte.

Ein ▸IDS besteht aus folgenden Komponenten:

- Ereigniskomponenten
- Analysekomponenten
- Datenbankkomponenten
- Reaktionskomponenten

Die Systeme können sowohl als Host-IDS oder als netzbasiertes ▸IDS eingebunden werden.

Beim Vorgehen für die Ereigniserkennung bzw. deren Analyse lässt sich grundsätzlich zwischen Anomalieerkennung und Signaturanalyse unterscheiden. Darüber hinaus sollte man bei der Signaturanalyse zwischen Single-Step-Angriffen und Multi-Step-Angriffen differenzieren.

Ein weitverbreitetes System ist *Snort*. Für dieses System existieren automatisierte Aktualisierungsmöglichkeiten. Darüber hinaus können eigene Regeln erstellt und vorhandene Regeln individualisiert werden.

Ein guter Indikator für Unerwünschtes im eigenen Netzwerk sind Honeypots. Bei Honeypots wird zwischen Client und Servern sowie Low Interaction und High Interaction unterschieden.

Aufgaben zur Selbstüberprüfung

2.1 Nennen Sie die Komponenten eines ▸IDS.

2.2 Welche Sprachenarten gibt es bei einem ▸IDS?

2.3 Worin liegt die Schwierigkeit bei der Erkennung einer Multi-Step-Attacke?

3 Verhindern von Netzwerkangriffen

In diesem Kapitel erhalten Sie einen kurzen Überblick über die grundlegenden Firewalltechniken. Darüber hinaus lernen Sie verschiedene Varianten von Honeypots und Schwachstellenscanner kennen.

Sie erfahren, wie Sie Ihre Systeme mithilfe gängiger Werkzeuge auf Schwachstellen überprüfen können.

Den Abschluss des Kapitels bildet eine Übersicht über ▸UTM-Lösungen.

3.1 Intrusion Prevention

Das Erkennen eines Angriffs ist die Voraussetzung, um weitere Maßnahmen einzuleiten. Doch nach der Erkennung geht es zunächst um die Begrenzung des Schadens und im weiteren Verlauf um die Identifizierung des Angreifers. Genau hier grenzt sich ein ▸IPS von einem ▸IDS ab.

Ein ▸IDS sammelt Informationen und Daten zu Aktivitäten und erstellt Alarmmeldungen, wenn die Aktivitäten gegen voreingestellte Regeln verstoßen. Ein ▸IPS dagegen ergreift selbstständig Maßnahmen, um die festgestellte Aktivität zu unterbinden oder zu blockieren. Daher ist die korrekte Konfiguration eines ▸IPS sehr wichtig, um vorhandene elektronische Abläufe, Prozesse oder Verfahren nicht zu stören. Man stelle sich nur einmal vor, das ▸IPS würde den Datenverkehr eines E-Mail-Servers als potenzielle Bedrohung ansehen und den entsprechenden Verkehr, also alle versendeten und empfangenen E-Mails, blockieren. Dies würde für nahezu jede Organisation einen Schaden verursachen.

Ein ▸IDS dagegen würde lediglich eine Vielzahl von „False Positive“-Meldungen erstellen, die durch den Benutzer bearbeitet werden müssten. Eine solche Fehlkonfiguration würde lediglich zu einem Entdeckungsrisiko anderer Aktivitäten, die eine echte Sicherheitsverletzung bedeuten, führen.

3.2 Firewalls

Ein Firewall-System übernimmt zwei grundlegende Aufgaben in einem Netzwerk:

- „Brandschutzmauer“: Eine Brandschutzmauer in einem Gebäude versucht, ein Feuer an der Ausbreitung zu hindern. Eine Firewall agiert vergleichbar. Sie versucht, Netzwerkprobleme, Angriffe sowie die Ausbreitung von schädlichem Code an der Verbreitung zu hindern.
- „Pförtner“: Ein Pförtner empfängt und überprüft die passierenden Personen. Wer passieren kann, wird in einem Logbuch vermerkt. Wer keine Zutrittsberechtigung besitzt, wird abgewiesen. Eine Firewall arbeitet genauso. Nur überwacht sie die ein- und ausgehenden Datenpakete.

Es können grundsätzlich zwei klassische Firewall-Technologien unterschieden werden. Die erste Variante ist ein Paketfilter. Ein Paketfilter überprüft die Header-Informationen der Datenpakete auf ▸OSI-Schicht 3 (z.B. ▸IP) und auf ▸OSI-Schicht 4 (z.B. ▸TCP,

▸UDP). Bei diesem Firewall-System handelt es sich letztendlich um einen etwas funktionserweiterten Netzwerkrouter, da er Pakete entgegennimmt, sie genau betrachtet und entsprechend weiterleitet.

Bei der Analyse dieser Pakete werden gängigerweise folgende Informationen abgefragt:

- Das *Quellsystem und das Zielsystem* sind die beiden Endpunkte der ▸TCP/IP-Verbindung. Das Quellsystem versucht mit dem Zielsystem zu kommunizieren. Die ▸IP-Adressierung der beiden Systeme werden im ▸IP-Header gespeichert. Der Paketfilter überprüft, ob die beiden Systeme miteinander kommunizieren dürfen. Falls nicht, wird das Paket abgelehnt.
- Der *Quell- und der Zielport* werden ebenfalls im ▸IP-Header des Pakets gespeichert. Der Paketfilter überprüft, ob die Nutzung des Ports erlaubt ist, und weist das Paket zurück, wenn der Port geschlossen ist.
- Darüber hinaus können noch weitere *Header-/Protokoll-Optionen* überprüft werden. Zum Beispiel können bei der Verwendung von ▸TCP anhand der gesetzten Flags der Sinn und Status einer Verbindung analysiert werden.

Die zweite klassische Firewall-Technologie sind Application Gateways und Proxies. Anders als ein Paketfilter überprüft ein Application Gateway auch die Anwendungsprotokollebene auf den höheren ▸OSI-Schichten (z. B. Session Layer, Presentation Layer). Dies bedeutet, dass ein Application Gateway im Gegensatz zu einem Paketfilter zwischen einem Hypertext-Transfer-Protocol(▸HTTP)- und einem Simple-Mail-Transfer-Protocol (▸SMTP)-Datenverkehr auf einem Port unterscheiden kann und den zulässigen Datenverkehr erlauben wird, während der unzulässige Datenverkehr unterbunden wird.

Immer mehr Anbieter von Firewall-Produkten erweitern ihr Produktportfolio um eine sogenannte *Next Generation Firewall(▸NGFW)*. Diese Varianten von Firewalls zielen darauf ab, die oben genannten Firewall-Funktionen um ▸IPS- und Proxy-Funktionen zu ergänzen und somit beispielsweise auch die Überprüfung von ▸TLS-verschlüsseltem Datenverkehr zu ermöglichen. Darüber hinaus kann diese Art von Firewalls in vorhandene Berechtigungskonzepte eingebunden werden, um einzelnen Benutzern und Anwendungen Zugriffe zu erlauben oder zu verweigern.

Übung 3.1:

Auf welchen Schichten des ▸OSI-Modells arbeitet ein Paketfilter?

3.3 Rechtliche Hinweise

Das Finden und Analysieren von Schwachstellen ist eine rechtliche Grauzone und hat mit der Aufnahme des sogenannten Hackerparagrafen durch seine allgemeine Formulierung für Irritationen bei vielen IT-Sicherheitsexperten gesorgt.

§ 202a StGB – Ausspähen von Daten

„(1) Wer unbefugt sich oder einem anderen Zugang zu Daten, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, unter Überwindung der Zugangssicherung verschafft, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

(2) Daten im Sinne des Absatzes 1 sind nur solche, die elektronisch, magnetisch oder sonst nicht unmittelbar wahrnehmbar gespeichert sind oder übermittelt werden."

Letztlich ist der Einsatz solcher Werkzeuge ausschließlich straffrei, wenn der Systemeigner für deren Nutzung entsprechend zugestimmt hat. Dies muss nicht zwingend der für das System zuständige Administrator sein. Es genügt eine entsprechende Vereinbarung mit der Unternehmensleitung.

Da die rechtliche Analyse dieses „Tatbestands" selbst ein eigenes Studienheft beanspruchen würde, belasse ich es bei einem kurzen rechtlichen Hinweis. Im folgenden Kapitel lernen Sie verschiedene und weitverbreitete Anwendungen zur Schwachstellenanalyse mit ihren jeweiligen Vor- und Nachteilen kennen, die für das Testen der Netzwerk- und Systemsicherheit verwendet werden können.

3.4 Port- und Schwachstellen-Scanner

Der folgende Abschnitt enthält einen Überblick über verschiedene Anwendungen zum Testen von Systemen und Netzwerken. Hierbei wird grundsätzlich zwischen dem Überprüfen von offenen Netzwerkverbindungen, dem sogenannten Portscan, und Anwendungen zur Schwachstellenanalyse, den sogenannten Vulnerability-Scanner, unterschieden. Die Produkte zur Schwachstellenanalyse beinhalten oftmals sogenannte Portscan-Möglichkeiten.

Des Weiteren lassen sich die vorgestellten Produkte grundsätzlich in passive und aktive Anwendungen unterscheiden. Während passive Anwendungen, wie beispielsweise *tcp-dump*, *Wireshark* oder auch *kismet* für Wireless-Local-Area-Network(▸WLAN)-Netze, sich vor allem zur Analyse von Netzwerken eignen, versuchen die aktiven Varianten in Abhängigkeit ihrer Konfiguration, die gefundenen Schwachstellen auch auszunutzen, um die Verwundbarkeit entsprechend offenzulegen.

Der Einsatz solcher Produkte erfordert stets die vorherige Zustimmung des Systemeigners.

Darüber hinaus sollten vor dem Einsatz zunächst rechtliche und Haftungsfragen geklärt werden. Mit Einführung des sogenannten Hackerparagrafen (vgl. § 202a StGB) wurde die unbefugte Verwendung solcher Anwendungen unter Strafe gestellt. Darüber hinaus kann der unsachgemäße Einsatz zu Schäden auf den Zielsystemen führen. Kurzum, solche Werkzeuge machen zwar Spaß, sind jedoch keiner und sollten daher nur eingesetzt werden, wenn die Systemeigner einem Einsatz zugestimmt haben. Darüber hinaus sollte die Handhabung dieser Anwendungen zunächst an geeigneten Testsystemen geübt werden.

3.4.1 tcpdump und Wireshark

Die Anwendungen *tcpdump* und *Wireshark* sind freie und quelloffene Werkzeuge zur Analyse von Datenpaketen. Die Werkzeuge arbeiten auf den Schichten 2 bis 7 des ▸OSI-Modells und bieten daher umfangreiche Analysemöglichkeiten zur Netzwerkkommunikation.

Beide Anwendungen sind passive Scanner, die die Kommunikation der Netzwerkkarte des Systems, auf dem sie ausgeführt werden, aufnehmen und wiedergeben. Darüber hinaus geben sie auch mitgeschnittene *Broadcasts* wieder. Das Werkzeug *Wireshark* bietet anders als *tcpdump* eine grafische Benutzeroberfläche.

Beide Anwendungen basieren auf der UNIX-Programmbibliothek *libpcap* zum Mitschneiden von Netzwerkinformationen (Packet Capture, ▸PCAP) und wurden auch für Windows-Systeme portiert. Hier verwenden die Anwendungen die Schnittstelle *Winpcap*. Sowohl *Wireshark* als auch *tcpdump* erlauben das Speichern und Exportieren ihrer Mitschnitte in einer Vielzahl von Formaten zur nachgelagerten Analyse in weiteren Anwendungen.

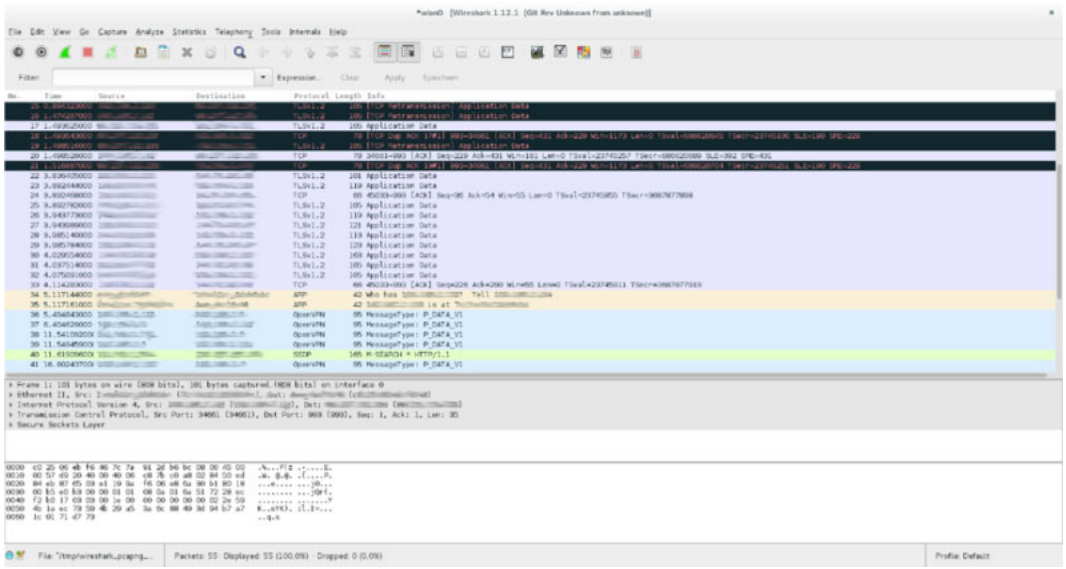

Abb. 3.1: Paketmitschnitt mit der Anwendung *Wireshark*

Abb. 3.1 zeigt einen exemplarischen Paketmitschnitt auf einem Linuxsystem.

3.4.2 Schwachstellenscanner für Wireless LAN

Der Einsatz eines Wireless Local Area Network (▸WLAN) ist sowohl für Unternehmen als auch für Privatpersonen mit einer Vielzahl von Vorteilen, aber auch mit entsprechenden Risiken verbunden. Aber gerade Systeme, die für einen mobilen Betrieb ausgelegt sind, setzen heutzutage für ihren Betrieb eine vorhandene ▸WLAN-Infrastruktur voraus.

Da sich die Reichweite und Ausrichtung eines ▸WLAN-Signals nicht vollständig restriktiv planen und umsetzen lässt, ist ein solches Netzwerk immer wieder ein Einfallstor für Angreifer. Daher sollten die Prinzipien der Netzwerkverteidigung (vgl. Abschnitt 1.4) für diese Netzwerkart besonders berücksichtigt und die Übergänge zu anderen Netzwerkbereichen durch Firewalls voneinander getrennt werden.

Anders als die in den nachfolgenden Abschnitten vorgestellten Schwachstellen-Scanner arbeitet ein Schwachstellen-Scanner für ▸WLAN auf Schicht 2 des ▸OSI-Modells. Er sucht nach vorhandenen Verbindungen und versucht, diese durch das Ausnutzen von Schwachstellen in der Verschlüsselung oder durch das Aufbrechen von schwachen Verschlüsselungen zu kompromittieren und sich hierzu einen grundlegenden Zugang zum Netzwerk zu verschaffen.

Ein weitverbreiteter Irrglaube ist es, dass das Verstecken der Service Set Identifiers (▸SSID) vor gängigen Attacken wie beispielsweise „Wardriving“ schützt. Diese Schutzfunktion ist jedoch nahezu wirkungslos, da sie zwar das sogenannte *broadcast beacon*, jedoch nicht die Kommunikation zwischen einem Client und dem *Access Point* unterbindet. Diese Kommunikation kann vom Angreifer ebenfalls ausgespäht und als Ausgangspunkt für weitere Angriffe verwendet werden.

Die wohl bekanntesten Scanner bzw. ▸WLAN-Sniffer sind *kismet* und *Aircrack-ng*. Das Werkzeug *kismet* arbeitet als passiver Scanner und überprüft die für ▸WLAN freigegebenen Frequenzbereiche auf das Vorhandensein von entsprechender Kommunikation. Bei Fund eines Access-Points werden sowohl die Basic-Service-Set-Identification (▸BSSID) als auch die Extended-Service-Set-Identifier (▸ESSID) ausgelesen. Gerade die ▸BSSID kann für ein weiteres Vorgehen interessant sein, da sie häufig der ▸MAC-Adresse des Access-Points entspricht. Darüber hinaus unterstützt die Anbindung von Global-Positioning-System(▸GPS)-Empfängern bei der geografischen Ortung der gefundenen Netzwerke.

Die Werkzeugsammlung *Aircrack-ng* wurde im Jahr 2006 veröffentlicht und seitdem kontinuierlich weiterentwickelt. Die Sammlung ist kostenfrei und quelloffen verfügbar. Sie bietet verschiedene Werkzeuge zum Scannen, Testen und auch Aufbrechen von ▸WLAN-Verbindungen.

Für das Testen und Aufbrechen der Verbindungen können dabei sowohl Exploits zu schwachen Verschlüsselungsimplementierungen (z. B. Wired-Equivalent-Privacy (▸WEP)- und Wi-Fi-Protected-Setup(▸WPS)-Attacken), als auch Wörterbuch- und Brute-Force-Attacken genutzt werden. Sofern eine schwache Verschlüsselung des ▸WLAN verwendet wird oder die Verschlüsselung durch vorhandene Zusatzfunktionen geschwächt wurde, kann die Werkzeugsammlung den Access-Point direkt angreifen. Bei einer starken Verschlüsselung werden verschiedene Verbindungen mitgeschnitten, in den Aufnahmen nach dem Schlüssel gesucht und dieser unter Verwendung von Wörterbuch- oder Brute-Force-Attacken versucht zu dechiffrieren.

3.4.3 Network Mapper (Nmap)

Die Anwendung „Network Mapper (▸Nmap)“ ist ein freies und quelloffenes Werkzeug zum Durchsuchen von Netzwerken und zur Durchführung von Schwachstellenprüfungen. Die Anwendung wurde 1997 entwickelt. Mit ▸Nmap lassen sich Systeme und Netzwerke in ▸LAN oder Wide Area Network (▸WAN) auf offene Ports und entsprechende aktive Dienste überprüfen. ▸Nmap ist konsolenbasiert und steht sowohl für Windows als auch für Linux- und Mac OS-Systeme zur Verfügung. Darüber hinaus existiert mit *Zenmap* auch eine Erweiterung, die eine grafische Oberfläche bietet.

Die Anwendung stellt dem Benutzer verschiedene Scan-Methoden zur Verfügung. Die grundlegenden Scan-Methoden werden nachfolgend vorgestellt:

- Der *SYN-Scan* (Parameter **-sS**) ist eine Methode, die eine schnelle Prüfung einer Vielzahl von Ports bei geringer Zeitdauer zulässt. Diese Methode öffnet ▸TCP-Verbindungen, die dann nicht abgeschlossen werden. Daher wird diese Methode auch als halboffenes Scannen bezeichnet. Es wird ein *SYN*-Paket gesendet, das die Antwort *SYN/ACK* („offen“) oder *RST* („geschlossen“) erwartet. Sofern nach einer entsprechenden Anzahl von Verbindungsversuchen keine Antwort gemeldet wird, wird der Zustand „gefiltert“ angenommen.
- Die ▸*TCP-Connect-Scan*-Methode (Parameter **-sT**) verwendet den vorhandenen ▸TCP-Stack über den Systemaufruf „connect“ des Betriebssystems zum Aufbau einer Verbindung mit dem Zielrechner und dem entsprechenden Zielport. Diese Methode kann verwendet werden, wenn der Benutzer keine administrativen Berechtigungen auf dem System oder zumindest systemseitig keine Schreibrechte für rohe Datenpakete besitzt. Anders als bei der *SYN-Scan*-Methode wird die ▸TCP-Verbindung hier vollständig aufgebaut und beendet. Daher wird bei Verwendung dieser Methode die Verbindung auf dem Zielrechner (z. B. in syslog) protokolliert und benötigt eine höhere Zeitdauer als die *SYN-Scan*-Variante.

Die ▸*UDP-Scan*-Methode (Parameter **-sU**) sendet einen leeren ▸UDP-Header an alle Ports des Zielsystems. Wenn das Zielsystem eine Antwort zurücksendet, wird der Port als „offen“ markiert. Enthält die Antwort eine Fehlermeldung, wird in Abhängigkeit der Fehlermeldung der Port als „gefiltert“ oder „geschlossen“ markiert. Auch wenn ein Großteil der Dienste über das ▸TCP-Protokoll kommunizieren, ist ▸UDP nicht obsolet. So verwenden beispielsweise die Dienste Domain Name System (▸DNS), Simple Network Management Protocol (▸SNMP) und Dynamic Host Configuration-Protocol (▸DHCP) dieses Protokoll.

Darüber hinaus existieren noch weitere Scan-Methoden, die jedoch (mit Ausnahme der ▸IP-Scan-Methode) auf den oben genannten Methoden beruhen. Bei der Angabe der Zielports kann der Benutzer wählen zwischen

- einem spezifischen Port (Parameter -p) sowie
- den gängigsten Ports (Parameter -F).

Die gängigsten Ports können dabei vom Benutzer festgelegt werden. Erfolgt keine Festlegung, werden die sogenannten „well known“-Ports (vgl. Abschnitt 1.6) verwendet.

Für die Ausgabe der Scan-Ergebnisse bietet ▸Nmap ebenfalls verschiedene Varianten. Hier kann der Benutzer durch Angabe des entsprechenden Parameters (z.B. -oX für eine Ausgabe im XML-Format) das Format zur weiteren Aufbereitung in einer anderen Anwendung selbst wählen.

Darüber hinaus bietet die Anwendung eine sogenannte Script-Engine, die dem Benutzer eine einfache Überprüfung von Schwachstellen und simplen Brute-Force-Attacken ermöglicht.

Abb. 3.2 zeigt das Ergebnis eines SYN-Portscans eines Systems nach den Ports 21 (▸FTP), 22 (SSH), 80 (▸HTTP) und 443 (▸HTTPS).

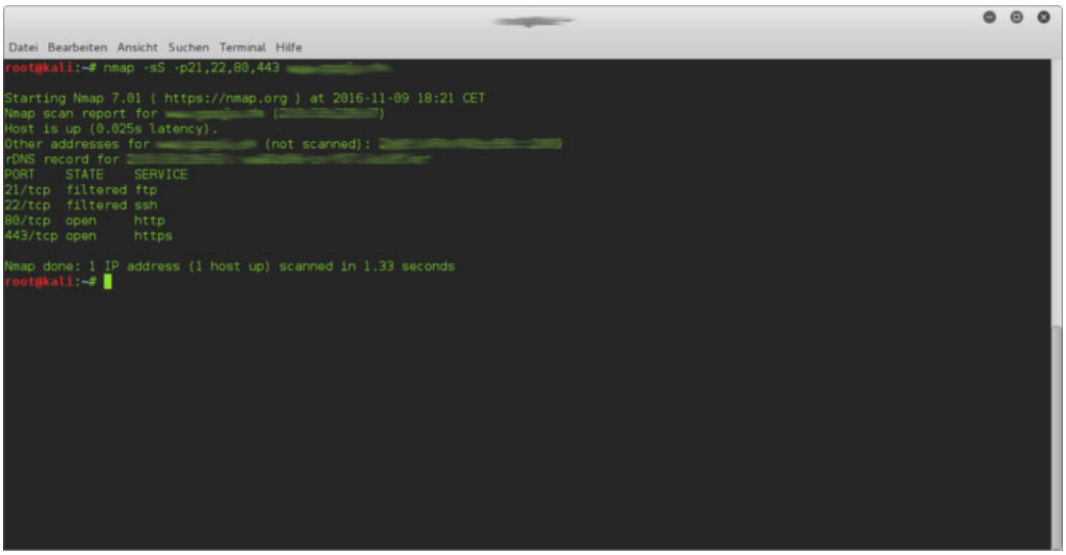

Abb. 3.2: Portscan mit der Anwendung Network Mapper (*Nmap)*

3.4.4 Open Vulnerability Assessment System (OpenVAS)

Das freie und quelloffene Framework *Open Vulnerability Assessment System (▸OpenVAS)* ist eine Werkzeugsammlung zum Durchführen von sogenannten Schwachstellenanalysen. Es wurde im Jahr 2005 von der proprietären Variante *Nessus* abgespalten und als freies Produkt in der Programmiersprache C weiterentwickelt.

Die Analyse-Anwendung selbst wird täglich über sogenannte Network-Vulnerability-Tests(▸NVT) aktualisiert und erhält somit Informationen zu neuen Schwachstellen. Für die Durchführung der Portscans verwendet die Anwendung ▸*Nmap*. Die Anwendung arbeitet durch einen modularen Aufbau als Client/Server-Architektur.

Der Benutzer kann auf die Anwendung über eine Konsolenanwendung (▸OpenVAS CLI) oder eine grafische Weboberfläche, den sogenannten *Greenbone Security Assistant*, zugreifen. Die Ergebnisse werden in der Weboberfläche grafisch aufbereitet und können in verschiedenen Dateiformaten exportiert werden.

3.4.5 Nessus

Das Framework *Nessus* des amerikanischen Unternehmens Tenable Network Security, Inc. ist heutzutage der kommerzielle Bruder von ▸*OpenVAS*. Der Programmierer Renaud Deraison entwickelte und veröffentlichte die erste Version von *Nessus* im Jahr 1998 im Alter von 17 Jahren. Er ist jetzt Chief Technology Officer des Unternehmens.

Das Framework wird heutzutage sehr häufig für Schwachstellenanalysen verwendet und gilt als De-facto-Standard der Branche. Das Unternehmen bietet das Framework in unterschiedlichen Varianten an. Die größte Produktvariante beinhaltet neben diversen Audit-Werkzeugen auch eine umfangreiche Unterstützung zum Patch-Management.

Wie ▸OpenVAS arbeitet auch *Nessus* als Client/Server-Architektur und erhält über eine regelmäßige Aktualisierungsroutine Informationen zu neuen Schwachstellen. Die größte Produktvariante besitzt darüber hinaus einfache heuristische Funktionen zu Angriffen, indem es aus gesammelten Informationen des Netzwerks und seiner Teilnehmer einen „Normalzustand" berechnet und Abweichungen davon dokumentiert.

Übung 3.2:

Worin besteht der Unterschied zwischen einem SYN-Scan und einem ▸TCP-Connect-Scan bei Nmap?

3.4.6 Metasploit

Das Framework *Metasploit* wurde im Jahr 2003 von dem amerikanischen Programmierer H. D. Moore entwickelt. Die erste Version enthielt eine Datenbank mit 11 Exploits und war in der Programmiersprache Perl geschrieben. Mit der Unterstützung eines Programmierers mit dem Nickname „Spoonm" wurde das Framework im darauffolgenden Jahr komplett überarbeitet und in die Programmiersprache Ruby portiert. Die Version Metasploit 2.0 enthielt 19 Exploits und 27 *Payloads* bei ihrer Veröffentlichung. Im Jahr 2009 wurde *Metasploit* von dem Unternehmen Rapid7 LLC gekauft und wird seitdem als kommerzielles Produkt mit einer freien Variante mit reduziertem Funktionsumfang weiterentwickelt.

Ein **Payload** ist sozusagen die eigentliche „Ware", also die Funktion oder der Inhalt eines Programms. Der Begriff wird oftmals im Zusammenhang mit Schadsoftware zur Beschreibung der eigentlichen Schadfunktion verwendet. Als Payload bezeichnet man jedoch auch den Inhalt im Rahmen eines Verschlüsselungsvorgangs.

Das Framework arbeitet intrusiver als *Nessus* und ▸*OpenVAS* und überprüft Systeme nicht nur auf Schwachstellen, sondern beutet diese bei entsprechender Konfiguration auch aus. *Metasploit* setzt sich aus

- einem Datenbankmodul,
- einer Konsolenanwendung und
- verschiedenen grafischen Oberflächen

zusammen, deren Verwendung zur Bedienung jedoch nicht zwingend erforderlich ist. Das Framework erlaubt es dem Benutzer, eigene Module zu entwickeln und diese für den Verwendungszweck passend zu individualisieren. Darüber hinaus stellt es auch vorkonfigurierte Module zur nichtintrusiven Schwachstellenanalyse bereit.

Das Framework bietet auch verschiedene Portscan-Funktionen, die über ▸Nmap bereitgestellt werden. Ein weiterer Bestandteil des Frameworks ist eine Suchfunktion, in der vorkonfigurierte *Exploits* nach Schweregrad, Plattform, Erscheinungsdatum (▸CVE) und Anwendungsnamen gesucht werden können. Abb. 3.3 zeigt eine beispielhafte Suche nach Exploits für das Datenbankmanagementsystem *MySQL*.

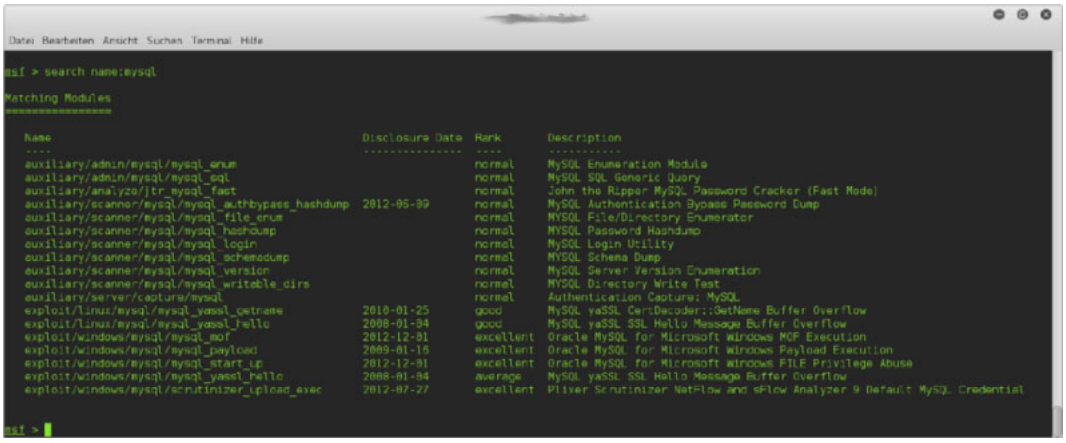

Abb. 3.3: Das Framework *Metasploit*

Für eine umfassende Schwachstellenanalyse eines Netzwerks oder Systems können *Metasploit* und *Nessus* über ein Plug-in verbunden und die Ergebnisse von *Nessus* als Grundlage für weitere Exploit-Handlungen verwendet werden.

Eine grundlegende *Payload*-Routine von *Metasploit* ist *Meterpreter*. *Meterpreter* bietet dem Benutzer unter anderem folgende Funktionen:

- Auslesen und Kopieren von Arbeits- und Datenspeicher-Daten
- Kopieren von Systempasswörtern
- Aufnehmen von Bildschirmkopien

Darüber hinaus können Systeme durch Meterpreter auch als „Trittbrett“ für das Exploiting weiterer Systeme verwendet werden. Diese Technik wird Pivoting genannt. Das erste System fungiert hierbei als Proxy. Diese Technik wird oftmals verwendet, um ein veraltetes Firewallsystem zu überwinden und das dahinterliegende Netzwerk auszuspähen.

Übung 3.3:

Was bedeuten die Begriffe „Heuristik“ und „intrusiv“?

3.5 Unified Threat Management (UTM)

Der Begriff Unified Threat Management (▸UTM) bezeichnet Systeme, die sowohl Firewall- als auch ▸IPS-, Proxy-, Virtual-Private-Network(▸VPN)-Gateway- und Virenschutz-Funktionen in einem System kombinieren. Den gängigen Bedrohungen soll hierdurch eine ganzheitliche „Schutzmaßnahme“ gegenübergestellt werden. Ein Großteil der ▸UTM-Systeme verwenden quelloffene weitverbreitete Anwendungen, wie

beispielsweise *Snort* für ▸IPS-Funktionen, und kombinieren diese mit herstellereigenen proprietären Systemen, wie beispielsweise Virenschutzfunktionen. Der Anwender muss zur Nutzung dieser propietären Systeme oftmals zusätzliche Lizenzen erwerben.

Grundsätzlich bietet der Einsatz eines ▸UTM-Systems folgende Vorteile:

- reduzierte Komplexität durch den Einsatz eines einzigen Sicherheitssystems statt verschiedener Systeme und Anwendungen
- einfache Implementierung in ein bestehendes Netzwerk
- Einsatz von aufeinander abgestimmten Sicherheitssystemen
- geringer administrativer Aufwand
- einfache Fehlerbehebung, da ein defektes ▸UTM-System komplett ausgetauscht wird

Die Nachteile dieser Systeme liegen oftmals in der Konfiguration. Durch die einfache Implementierung in das bestehende Netzwerk besteht die Gefahr, dass einzelne Schutzfunktionen falsch konfiguriert werden und hierdurch ein falsches Sicherheitsbewusstsein bei den Benutzern entsteht.

Des Weiteren muss beim Ausfall einer einzelnen Komponente, wie beispielsweise der Netzwerkkarte, oftmals das gesamte System kurzfristig ausgetauscht werden. Mit diesem Ausfall ist also das Deaktivieren sämtlicher Sicherheitsfunktionen eines Netzwerks verbunden.

Unabhängig von diesen Nachteilen setzen vor allem immer mehr kleine und mittlere Unternehmen ▸UTM-Systeme zum Schutz ihres Netzwerks ein.

Zusammenfassung

Absolute Sicherheit gibt es nicht! Diese Aussage trifft auf viele Themenfelder zu, in denen Sicherheit von Bedeutung ist. Daher sollte man sich bei der Angriffserkennung nicht allein auf die Einstellung seiner Sicherheitsmaßnahmen verlassen, sondern diese selbst oder mithilfe von fachkundigen Dienstleistern überprüfen.

Ein Budget für ein sogenanntes *Penetration Testing* oder ein *Security Audit* gehört in vielen Branchen heutzutage als regelmäßige Plankosten dazu. Man sollte jedoch die Werkzeuge kennen und sich von der Qualität der Dienstleister vorab überzeugen, da die falsche Anwendung der Werkzeuge hohe Schäden und viel Wiederherstellungsarbeit zur Folge haben kann.

Bei Firewalls unterscheidet man grundsätzlich zwischen Paketfiltern und Application Gateways. Ein Paketfilter arbeitet auf den Schichten 3 und 4 des ▸OSI-Modells und bietet nur eine begrenzte Schutzfunktion. Ein Application Gateway überprüft auch anwendungsabhängige Protokolle. Des Weiteren werden ▸UTM-Lösungen für kleinere und mittlere Unternehmen immer populärer. Dem Vorteil des geringen Konfigurationsaufwands stehen jedoch einzelne Nachteile gegenüber.

Aufgaben zur Selbstüberprüfung

3.1 Was ist der Unterschied zwischen einem passiven und einem aktiven Scanner?

3.2 Welche Information kann die ▸BSSID einem Angreifer verraten?

3.3 Warum wird der ▸TCP-Connect-Scan oftmals protokolliert?

3.4 Was ist Pivoting?

4 IT-Forensik

Nach Bearbeitung dieses Kapitels verstehen Sie die Grundlagen der IT-Forensik. Sie verstehen, warum ein planvolles Vorgehen zur Sicherung forensischer Daten für eine vollständige Beweiskette sinnvoll ist und welche Fallstricke Ihnen auf dem Weg dahin begegnen können.

Sie sind vertraut mit einem planvollem Vorgehen zur verlustfreien Sicherung von Datenspeichern und kennen den grundlegenden Aufbau des Master Boot Record. Darüber hinaus kennen Sie den Unterschied der verschiedenen Slack-Varianten und beherrschen „Carving".

4.1 Grundlagen der IT-Forensik

Die meisten Menschen verbinden mit dem Begriff Forensik die gerichtsmedizinische Analyse von biologischen und physikalischen Spuren zur Ermittlung eines Vorgangs. Der ein oder andere Fan des Münsteraner Tatorts denkt bei Forensik vielleicht auch an Jan Josef Liefers in der Rolle von Professor Dr. Karl-Friedrich Boerne als Leiter der Rechtsmedizin des Uniklinikums Münster.

Doch auch wenn die IT-Forensik nichts mit dem klassisch-medizinischen Bereich zu tun hat, ist sie doch ein Bereich, der in den letzten Jahren zunehmend wichtiger wurde. Der Grund hierfür ist einfach. Immer mehr Geschäftsprozesse werden digitalisiert und enthalten keine physischen Elemente. Hierdurch wird es zunehmend schwieriger, ob ein Missbrauch oder eine Fehlhandlung stattgefunden hat, nachzuvollziehen. Die IT-Forensik soll diese Lücke schließen und für digitale Informationen eine rechtssichere Nachvollziehbarkeit gewährleisten. Das Ziel einer forensischen Untersuchung ist es, Antworten auf folgende Fragen zu finden:

- Was ist geschehen?
- Wo ist es passiert?
- Wann ist es passiert?
- Wie ist es passiert?

Darüber hinaus sollte insbesondere bei strafrechtlicher Bewertung die Frage geklärt werden, wer es getan hat und wie eine Wiederholung des Tathergangs zukünftig verhindert werden kann.

In der IT-Forensik unterscheidet man grundsätzlich zwischen den Prozessschritten

- strategische Vorbereitung,
- operationale Vorbereitung,
- Datensammlung,
- Datenwiederherstellung,
- Datenanalyse sowie
- Präsentation.

Die Datensammlung umfasst die vorbereitenden Tätigkeiten sowie die Sicherung der Daten. Die Datenwiederherstellung und die Datenanalyse beinhalten die Auswertung der gesicherten Informationen sowie die Spurensuche. Die Präsentation bereitet die gefundenen Spuren auf und erlaubt Rückschlüsse auf die abgelaufenen Vorgänge.

Des Weiteren sollte eine forensische Untersuchung folgenden Anforderungen genügen (vgl. Goschonneck, 2004):

- *Akzeptanz*: Die angewandten Methoden und Schritte sollten in der Fachwelt beschrieben und allgemein anerkannt sein. Bei einem Einsatz neuartiger Verfahren sollte zunächst der Nachweis der Korrektheit erbracht werden.
- *Glaubwürdigkeit*: Die angewandten Verfahren und Funktionalitäten müssen erwiesenermaßen robust sein.
- *Wiederholbarkeit*: Die Ergebnisse der Verfahren müssen durch Dritte reproduziert werden können.
- *Integrität*: Die sichergestellten Daten müssen zu jedem Zeitpunkt gegen ungewollte Veränderung geschützt sein. Ein entsprechender Beweis muss jederzeit erbracht werden können.
- *Ursache und Auswirkungen:* Die Auswahl der Methoden muss eine logisch nachvollziehbare Beweiskette zwischen Ereignissen und Beweisspuren ermöglichen.
- *Dokumentation*: Jeder Schritt des Ermittlungsprozesses muss angemessen dokumentiert sein.

4.2 Vorbereitung und Datensammlung

Vor Beginn der Datensammlung sollte man sich zunächst überlegen, welche Daten man sammeln will und wie diese Daten verlust- und manipulationsfrei gesichert werden können. Man unterscheidet grundsätzlich zwischen

- flüchtigen und
- nichtflüchtigen

Daten. Der Begriff flüchtige Daten umfasst dabei die Informationen aus dem Arbeitsspeicher oder auch ▸RAM, die bei einem Neustart des Systems nicht wiederherstellbar sind.

Als nichtflüchtige Daten werden alle Daten bezeichnet, die auch nach einem Stromverlust des Systems noch vorhanden sind oder sich wiederherstellen lassen. Ein typisches Beispiel sind die auf der Festplatte gespeicherten Informationen.

Jede Datensammlung erfordert ein planvolles Vorgehen. Entscheidend dabei ist, dass die gesammelten Daten über den gesamten Prozessverlauf gegen fahrlässige (und vorsätzliche) Veränderungen gesichert sind. Daher sollte jede Datensammlung mit einer Überprüfung der Ausrüstungsgegenstände beginnen. Die Grundausstattung des Forensikers sollte dabei mindestens

- eine Kamera,
- einige Schraubendreher in branchenüblichen Größen,
- ein Notebook mit Datensicherungssoftware,
- einen Schreibschutzadapter,
- diverse Adapter,
- eine Funkuhr,
- Speichermedien zum Erstellen der Kopien,
- diverse Verlängerungskabel,

- diverse Aufkleber und Markierungshilfen sowie
- eine Taschenlampe und entsprechende Verpackungsmaterialien zum Transport der Datenträger

umfassen. Die Funkuhr und die Kamera dienen zur detaillierten Dokumentation des Vorgehens. Die Adapter und die Speichermedien werden für das Duplizieren der originalen Datenträger verwendet. Das erstellte Duplikat ist dabei keine Sicherheitskopie, sondern das zentrale Medium, das für die Analyse des Vorgangs verwendet wird.

4.2.1 Strategische Vorbereitung

Ein IT-Forensiker arbeitet normalerweise reaktiv. Er nimmt den gegenwärtigen Status eines Sachverhalts auf und versucht, daraus eine robuste Beweiskette zu rekonstruieren, die einem objektiven Beobachter einen entsprechenden Rückschluss auf den Sachverhalt erlaubt. Dieser Umstand bedingt, dass man als IT-Forensiker stets dafür sorgen sollte, dass der gegenwärtige Status nicht verändert wird. Daher sollte das zu überprüfende System nicht abgeschaltet, sondern lediglich isoliert werden, sofern es Teil eines Netzwerks ist. Der erste Rat an den Administrator sollte also stets lauten, das Netzwerkkabel abzuziehen oder das ▸W-LAN zu deaktivieren und niemals das System herunterzufahren.

4.2.2 Operationale Vorbereitung

Vor der Verwendung sollte zunächst überprüft werden, ob sich auf dem Speichermedium zum Erstellen des Duplikats bereits Daten befinden. Bedenken Sie, dass die Speichermedien die Grundlage für belastbare Beweise darstellen, daher sollten sich keine alten Daten oder Fragmente davon darauf befinden.

Eine **Schnell- oder Normalformatierung** eines Speichermediums **löscht keine Daten**, sondern lediglich die **Partitionstabelle** des Master Boot Record, also das Inhaltsverzeichnis des Speichermediums (Schnellformatierung), und sucht nach defekten Sektoren auf dem Speichermedium (Normalformatierung). **Lediglich das bitweise Überschreiben mit *null* löscht diese wirklich.**

Für Windows-Betriebssysteme ist eine Zusatzsoftware notwendig, damit ein Speichermedium vollständig gelöscht werden kann. Bei einer gängigen Linux-Distribution kann man mit dem Systemprogramm *dd* ein angeschlossenes Speichermedium vollständig löschen.

Als versierter IT-Forensiker sollte man übrigens nicht auf die Mär hereinfallen, dass ein Speichermedium mehrfach überschrieben werden sollte, um alle Daten so zu löschen, dass sie nicht wiederhergestellt werden können. Es reicht das einfache, aber vollständige Überschreiben des Speichermediums mit *Null*-Werten.

Eine Studie von verschiedenen Forschern zeigte, dass bereits das einfache Überschreiben des Speichermediums dazu führt, dass man ein einziges Bit lediglich mit einer Wahrscheinlichkeit von 56 % rekonstruieren kann, und das auch nur dann, wenn man den

genauen Speicherbereich des Bit kennt. Bei einem Byte sind es noch lediglich 0,97 %, da man hierfür bereits achtmal richtig raten müsste. Weitere Informationen hierzu finden Sie in Wright, Kleiman und Sundhar (2008).

Das nachfolgende Beispiel zeigt die exemplarische Verwendung des Systemprogramms *dd* zum Überschreiben des Speichermediums *sdb* mit *Null*-Werten. Der Parameter *if* steht für *Input File* und der Parameter *of* steht für (Sie ahnen es sicherlich) *Output File*.

Beispiel 4.1:

```
dd if=/dev/zero of=/dev/sdb
```

Bei der Verwendung von *dd* sollte man beachten, dass die Anwendung keine Fortschrittsanzeige bietet. Daher kann die Rückmeldung des Systems schon mal ein paar Minuten dauern, wenn Sie ein großes Speichermedium vollständig löschen.

Die Abkürzung *dd* steht übrigens nicht für Disk Dump oder Dump Device. Das Systemprogramm heißt eigentlich Convert and Copy. Die Abkürzung müsste daher cc lauten. Da cc jedoch bereits durch den entsprechenden C-Compiler belegt war, griff man auf den nächsthöheren Buchstaben zurück.

4.3 Sammeln flüchtiger Daten

Jede Datensammlung sollte mit dem Sichern der flüchtigen oder auch volatilen Daten beginnen. Unter Windows haben sich die Anwendungen *Win32dd* für 32-Bit-Systeme oder *Win64dd* für 64-Bit-Systeme bewährt. Vor dem Sichern des ▸RAM sollte berücksichtigt werden, dass das Ausführen der Anwendungen einen Teil des ▸RAM überschreibt. Das folgende Beispiel sichert den ▸RAM eines 32-Bit-Windows-Systems in die Datei *ram.dd*.

Beispiel 4.2:

```
win32dd.exe /r /f ram.dd
```

Bei älteren Linux-Systemen ist das Auslesen auf den Arbeitsspeicher mit dem Systemprogramm *dd* möglich. Da dies jedoch immer wieder zu Sicherheitsproblemen geführt hat, muss bei aktuelleren Linux-Systemen eine Zusatzanwendung, wie beispielsweise emphfmem oder *Linux Memory Extractor (LiME)*, installiert werden. Für Mac OS-Systeme wird auch eine Zusatzanwendung benötigt. Hierfür eignet sich beispielsweise *Mac Memory Reader*.

Sollte das System mit einem Passwort geschützt sein, muss der IT-Forensiker mitunter tief in die Trickkiste greifen, um den ▸RAM auslesen zu können. Die einfachste Art, an das Passwort zu gelangen, ist natürlich, den Besitzer danach zu fragen. Doch was, wenn dieser nicht erreichbar ist?

Auch dann gibt es diverse Möglichkeiten, an die Daten des ▸RAM zu gelangen. Eine Möglichkeit ist beispielsweise die Direct-Memory-Access(▸DMA)-Attacke. Diese Attacke verwendet eine Eigenschaft des ▸IEEE 1394-Bussystems (Firewire), die dazu dient, die Geschwindigkeit der Datenübertragung zu erhöhen, und daher den ▸RAM direkt anspricht. Die durch das Betriebssystem implementierten Zugriffskontrollen werden daher umgangen. Diese Attacke funktioniert natürlich auch bei anderen ▸DMA-Schnittstellen, wie beispielsweise Thunderbolt oder DisplayPort.

Der einfachste Weg, diese Attacke durchzuführen, ist die Verwendung einer vorgefertigten, speziellen Boot-CD mit einem entsprechenden Betriebssystem (z. B. SANS SIFT Kit).

Die andere Variante ist eine *Kaltstartattacke*. Die Kaltstartattacke-Methode beruht auf Erkenntnissen einer wissenschaftlichen Arbeit von Forschern der Princeton-Universität aus dem Jahr 2008. Weitere Informationen zu dieser Arbeit finden Sie in Halderman et al. (2008). Die Forscher fanden heraus, dass sich der ▸RAM nach dem Ausschalten des Systems nicht in Millisekunden entlädt, sondern unter bestimmten Voraussetzungen eine Datenremanenz für einen Zeitraum von wenigen Sekunden bis Minuten erhalten bleibt.

Die *Kaltstartattacke* kann in zwei Varianten durchgeführt werden. Die erste Möglichkeit ist es, den ▸RAM einzufrieren und in einen anderen Computer zu transplantieren. Diese Variante kann man anwenden, wenn das entsprechende System nur eine Stromkabellänge von einem Kühlraum entfernt steht oder wenn man der Grundausstattung eine ausreichende Menge an Kältespray hinzufügt.

In der zweiten Variante startet man das System mit einem speziellen Betriebssystem und einer speziellen Anwendung, wie beispielsweise *Msramdmp* oder *USB Memory Scraper*, neu.

Man sollte die Anwendung der *Kaltstartattacke* stets als **letztes Mittel** wählen und vor dem ersten Einsatz ausreichend üben, da man hier lediglich einen Versuch hat, den ▸RAM zu sichern.

Neben dem ▸RAM befinden sich auf einem System jedoch noch weitere wichtige Daten, die mit dem Ausschalten des Systems verloren gehen könnten. Beispiele hierfür sind die aktiven Netzwerkverbindungen, temporäre und geöffnete Dateien oder auch Prozesslisten. Ein Großteil dieser flüchtigen Daten können unter Kenntnis der entsprechenden Betriebssystemkommandos gesichert werden. Dies verlangt jedoch die Kenntnis der entsprechenden Kommandos für das vorgefundene Betriebssystem im Ernstfall.

Tab. 4.1 stellt eine Übersicht der flüchtigen Daten eines Systems dar:

Tab. 4.1: Übersicht der flüchtigen Daten

Flüchtige Daten	Windows-Werkzeuge	Linux-Werkzeuge
Arbeitsspeicher	`Win32DD, Win64DD`	`dd, fmem`
Routing-Tabellen, Kerneldaten	`Route PRINT, arp -a, netstat`	`netstat -r -n`
Prozesslisten	`PsList, ListDDLs, Curr-Process, tasklist`	`ps -ef, lsof`
aktive Netzwerkverbindungen	`netstat -a`	`netstat -a, ifconfig`
Programme mit Verbindungen	`sc queryex, netstat -ab`	`netstat -tunp`

Flüchtige Daten	Windows-Werkzeuge	Linux-Werkzeuge
geöffnete Dateien	`Handle, PsFile, Openfiles, net file`	`lsof, fuser`
Netzwerkfreigaben	`Net Share, Dumpsec`	`showmount -e, show-mount -a`
offene Ports	`OpenPorts, ports, netstat -an`	`netstat -an`
momentan eingeloggte Benutzer	`Psloggedon, whoami, nt-last, netusers /l`	`w, who -T, last`
geöffnete echtzeitverschlüsselte Dateisysteme	`Manage-bde (Bitlocker), efsinfo (EFS)`	`mount -v, ls /media`
temporär verbundene Dateisysteme	`FsInfo, reg (mounted devices)`	`mount -v, ls /media`
entfernt geführte Logging- und Monitordaten	`psloglist`	`/etc/syslog.conf, Port UDP 514`
physische Konfiguration, Netzwerktopologie	`Systeminfo, msinfo32, ip-config /all`	`ifconfig -a, netstat -an`
Archivmedien	`reg (Mounted Devices), Net share, netstat -a`	`mount -v, ls /media`
aktive Systemzeit (Abweichung zur Funkuhr)	`time /T, date /T, uptime`	`time, date, uptime`
Umgebungsvariablen	`cmd /c set`	`env, set`
Zwischenablage	`Pclip`	
Massenspeicherinhalte	`FTK Imager, EnCase`	`Dc3dd, ewfacquire, Guymager`

Es empfiehlt sich, die entsprechenden Befehle in einem passenden Skript zusammenzufassen, um sicherzustellen, dass keine speichernswerten Inhalte vergessen werden. Sowohl bei Windows- als auch bei Linux-Systemen lässt sich die Ausgabe der Befehle in Textdateien umleiten. Beispiel 4.3 zeigt die Umleitung der Dateiausgabe unter Windows und Linux. Der Dateiinhalt wird dabei überschrieben, wenn die Datei bereits vorhanden ist. Ansonsten wird die Datei erstellt.

Beispiel 4.3:

```
Befehl > Datei.txt
```

Des Weiteren kann die Befehlsausgabe an eine bereits bestehende Datei durch das Kommando in Beispiel 4.4 angehängt werden.

Beispiel 4.4:

```
Befehl >> Datei.txt
```

Natürlich findet sich im Internet eine Vielzahl von kommerziellen und freien Werkzeugen zur Sicherung flüchtiger Daten. Darüber hinaus stellen renommierte IT-Forensiker auch vorbereitete Skripte zum eigenen Gebrauch bereit.

Übung 4.1:

Welchen Anforderungen sollte eine forensische Untersuchung genügen?

4.4 Sammeln nichtflüchtiger Daten

Für die Sammlung von nichtflüchtigen Daten ist vor allem die Vollständigkeit entscheidend. Man sollte also nie eine einzelne Partition, sondern sektorweise eine vollständige Kopie des Datenträgers erstellen. Hierzu eignet sich das in Abschnitt 4.3 vorgestellte Systemprogramm *dd*. Die Anwendung kopiert blockweise Daten von einem Quelldatenträger auf einen Zieldatenträger oder in eine Abbilddatei.

Die beim Lese- und Schreibvorgang verwendete Blockgröße kann dabei vom Nutzer durch Setzen des Parameters *bs* gesteuert werden. Der Parameter erlaubt dabei sowohl die Verwendung von Binärpräfixen als auch die Verwendung von Dezimalpräfixen als Größenangaben. Weitere Informationen hierzu finden Sie in Abschnitt 4.4.7.

Zur Erstellung einer vollständigen Kopie eines Datenträgers existiert eine Vielzahl von Anwendungen. Das vollständige Kopieren eines Datenträgers wird als „Klonen“ bezeichnet.

Eine Anwendung mit grafischer Oberfläche zum Klonen für Linux ist *guymager*. Die Anwendung wurde vom luxemburgischen Forensiker Guy Voncken entwickelt und ist frei verfügbar. Eine Anwendung ist in Abb. 4.1 zu sehen.

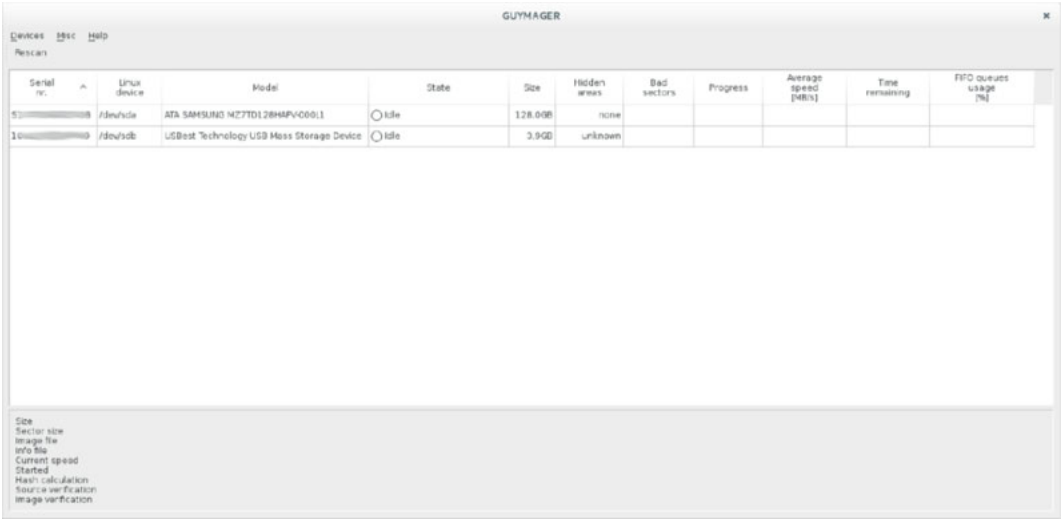

Abb. 4.1: Die Anwendung Guymager für Linux

Eine weitverbreitete Alternative für das Klonen unter Windows ist *FTK Imager* des amerikanischen Unternehmens Access Data. *FTK Imager* kann auf der Firmen-Website kostenlos heruntergeladen werden. Abb. 4.2 zeigt die Anwendung.

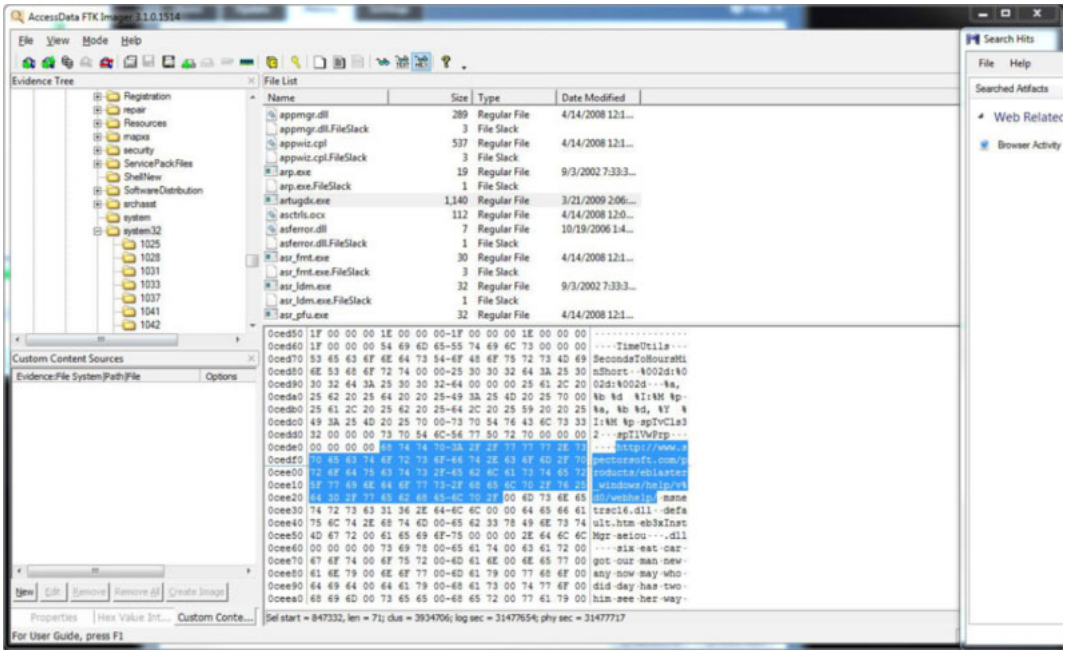

Abb. 4.2: Die Anwendung FTK Imager für Windows (Bildquelle: www.magnetforensics.com)

Die Datenanalyse erfolgt nicht auf dem Original, sondern stets auf der Kopie des Datenträgers. Daher sollte nach dem Erstellen der Kopie diese stets mit dem Original verglichen werden. Wollte man jedes Byte mit dem anderen vergleichen, wäre dies natürlich sehr aufwendig und in der Praxis nicht durchführbar.

Aus diesem Grund kommen Hashfunktionen zum Einsatz. Eine Hashfunktion bildet eine Beziehung zwischen zwei Mengen, die zu jeder Eingabe eine Ausgabe erzeugt. Als Eingabe dienen beispielsweise alle Bytes des Datenträgers. Die Ausgabe ist der entsprechende Hashwert.

Der Hashwert ist wie ein digitaler Fingerabdruck. Er ist nicht einmalig, jedoch ist die Wahrscheinlichkeit, dass ein Fingerabdruck doppelt vorkommt, sehr gering. Der britische Forscher Francis Galton berechnete, dass die Wahrscheinlichkeit eines doppelten Fingerabdrucks bei ca. 1 zu 64 Milliarden liegt.

Die gängigsten Hashfunktionen der digitalen Forensik sind Message Digest Algorithm 5 (▸MD5) und Secure Hash Algorithm (▸SHA). Die Hashfunktion wird sowohl vom Original-Datenträger als auch von der Kopie erstellt. Danach werden die Ergebnisse miteinander verglichen. Stimmen die Hashwerte überein, ist davon auszugehen, dass die Inhalte der beiden Medien übereinstimmen.

Oftmals werden in der IT-Forensik Datenträgerabbilder in speziellen Formaten gespeichert. Die Formate können beispielsweise

- fehlertolerante,
- komprimierte oder

- verschlüsselte

Eigenschaften besitzen. Die gängigsten Formate sind Expert Witness Format (▸E01), Extended Expert Witness Format (▸E01x), Advanced Forensic Format (▸AFF) und natürlich das Rohformat *DD*.

Übung 4.2:

Was wird mit dem Begriff „Klonen“ in Bezug auf die IT-Forensik bezeichnet?

Übung 4.3:

Versuchen Sie ein vollständiges Abbild eines USB-Sticks mit *dd* zu erstellen und dieses auf Ihrer Festplatte zu speichern.

4.4.1 Aufbau von Datenträgern

Aus Sicht des Betriebssystems ist ein Datenträger eine Abfolge von einer Vielzahl von Clustern, sogenannten Sektoren oder Datenblöcken. Ein Cluster ist für das Betriebssystem die kleinste lesbare und beschreibbare Einheit. Das Betriebssystem kann nur einen vollständigen Cluster ansteuern. Je größer die Clustergröße bei einem Betriebssystem ist, desto geringer ist die Fragmentierung von Dateien, da diese sozusagen zerteilt und in verschiedenen Clustern abgelegt werden.

Tab. 4.2: Übersicht der Clustergrößen

Dateisystem	Clustergröße	max. Dateigröße	max. Partitionsgröße
FAT12	512 B	32 MiB	32 MiB
FAT16	2 KiB	2 GiB	2 GiB
FAT32	4 KiB bis 16 KiB	4 GiB	8 TiB
NTFS	4 KiB	16 TiB	256 TiB
EXT2	4 KiB	2 TiB	16 TiB
EXT3	4 KiB	xx GiB	32 TiB
EXT4	4 KiB	xx GiB	1 EiB

Die Clustergröße bei ▸*FAT32* ist abhängig von der Partitionsgröße. Je größer eine Partition ist, desto größer werden die Cluster.

Ein zusammenhängender Bereich desselben Dateisystems wird als Partition bezeichnet. Für das Betriebssystem verhält sich eine Partition wie ein physikalisch getrennter Datenträger. Eine Partition muss vollständig auf einem einzigen Datenträger untergebracht werden. Eine Ausnahme hiervon bildet ein sogenanntes *Redundant Array of Independent-Disks* (▸*RAID*), ein logischer Festplattenverbund, der durch einen Hardware- oder einen Softwarecontroller gesteuert wird.

Bei einer bestimmten Betriebsart eines ▸RAID-Verbunds, dem sogenannten Striping, werden mehrere Datenträger zu einem logischen Datenspeicher zusammengeschlossen. Dies führt jedoch zu einer erhöhten Ausfallwahrscheinlichkeit, sofern keine weiteren Maßnahmen zur Verringerung der Ausfallwahrscheinlichkeit getroffen werden.

Ein Datenträger kann mehrere Partitionen enthalten. Ein modernes Betriebssystem kann mehrere Partitionen verwalten. Grundsätzlich kann hier zwischen

- Primärpartitionen und
- erweiterten Partitionen

unterschieden werden. Eine Primärpartition, die zum Start eines Betriebssystems verwendet wird, wird auch als Systempartition bezeichnet. Jeder Datenträger darf maximal vier Partitionen besitzen. Der Grund hierfür ist die historische Entwicklung des Master Boot Record (▸MBR) eines Datenträgers. Der ▸MBR enthält Informationen zu allen Partitionen auf einem Datenträger und besitzt eine Gesamtspeichergröße von 64 Bytes. Jeder Eintrag einer Partition benötigt einen Speicherplatz von 16 Bytes.

Der ▸MBR befindet sich in Sektor 0 des Datenträgers und besteht aus

- Bootcode (Bytes 0 bis 439),
- Disksignatur (Bytes 440 bis 443),
- reserviertem Bereich (Bytes 444 bis 445),
- Partitionstabelle (Bytes 446 bis 509) sowie
- der festgelegten Signatur *0x55 0xAA*.

Beispiel 4.5 zeigt den Befehl zur Sicherung des *mbr* in einem Linux-System. Hierbei sind insbesondere die Parameter *skip* und *count* zu beachten. Der Parameter *skip* verweist auf den Sektor, ab dem der Kopiervorgang erfolgen soll.

Beispiel 4.5:

```
dd if=/dev/sdX of=mbr.dd bs=512 count=1 skip=0
```

Neuere Varianten, wie beispielsweise die GUID Partition Table (▸GPT), besitzen ebenfalls diese Obergrenze. Hier lassen sich jedoch in einer erweiterten Partition wiederum mehrere logische Partitionen anlegen, sodass theoretisch beliebig viele Partitionen auf einem Datenträger enthalten sein können.

Ein *Partition Gap* ist ein ungenutzter Bereich zwischen zwei Partitionen. Dieser Bereich kann vom Benutzer absichtlich erstellt worden sein, um Daten zu verstecken. Des Weiteren kann der Bereich Restdaten von vorherigen Installationen enthalten.

4.4.2 Dateisysteme

Mit dem Anschalten des Systems beginnt ein festgelegter Startprozess, beginnend mit einer grundlegenden Überprüfung des Basic Input/Output System (▸BIOS), dem sogenannten Power-on Self Test (▸POST). Danach erfolgt der Sprung in Sektor 0 des Datenträgers und das Laden seines Inhalts in den ▸RAM. Des Weiteren wird die Partitionstabelle nach einem Eintrag, der mit dem Wert *0x80* („bootfähig“) beginnt, durchsucht.

Danach erfolgt der Sprung an den angegebenen Anfangssektor der Systempartition. Dort befindet sich der Virtual Boot Record (▸VBR), der die Clustergröße festlegt und den Verweis auf den Bootloader des jeweiligen Betriebssystems. Bei Windows-Systemen ist an dieser Stelle der Bootmanager mit der Metadatendatei *$Boot* zu finden.

Der beschriebene Bootprozess kann an jeder Stelle durch eine Schadsoftware kompromittiert werden. Oftmals kopieren Bootviren hierzu den ▸MBR in einen freien Bereich des Datenträgers (z.B. Sektor 1 bis 62). Daher sollte bei einer forensischen Untersuchung des ▸MBR nicht ausschließlich der erste Sektor, sondern ebenfalls die nächsten Sektoren eines Datenträgers untersucht werden.

Die vorhandenen Dateisysteme werden durch das installierte Betriebssystem verwaltet. Das Dateisystem umfasst dabei die folgenden grundsätzlichen Aufgaben:

- Verwaltung des Dateinamens (bzw. des Verzeichnisnamens)
- Verwaltung des Dateianfangs
- Verwaltung der Dateilänge zusammen mit Metadaten (z.B. Dateirechte, Zeitstempel)
- Verwaltung der von der Datei benutzten Speichereinheiten (Cluster)
- Verwaltung der belegten und freien Cluster

Jedes Betriebssystem verwendet ein eigenes Dateisystem. Beispielsweise verwenden moderne Windows-Systeme das Dateisystem New Technology File System (▸NTFS), während moderne Linux-Betriebssysteme das Dateisystem Extended File System Version 4 (▸ext4) verwenden. Die Firma Apple vertreibt ihre Mac OS-Computer mit dem Betriebssystem OS X und den Dateisystemen Hierarchical File System (▸HFS) oder Apple-File-System (▸APFS). Diese genannten Dateisysteme werden von anderen Betriebssystemen werkseitig nicht unterstützt und können teilweise erst durch Anwendungen von Drittanbietern verwendet werden. Sie sind teilweise auch gar nicht kompatibel.

Ein besondere Ausnahme hierbei bildet das Dateisystem File Allocation Table (▸FAT). Das Dateisystem existiert seit 1980 und wurde kontinuierlich weiterentwickelt. Heute wird das Dateisystem vor allem von externen Speichermedien, wie beispielsweise USB-Sticks, Speicherkarten und Digitalkameras, verwendet. Das Dateisystem ist sowohl mit Windows- als auch Linux- und Mac OS-Systemen kompatibel.

4.4.3 Slack-Varianten

Durch die Aufteilung der Cluster entsteht bei deren Verwendung automatisch ein Verschnitt am Ende des letzten Blocks einer Datei. Diesen unbenutzten Bereich nennt man *Slack* (engl. für Leerlauf, Flaute). Für Forensiker ist dieser Slack sehr interessant, da er Dateireste und unter bestimmten Umständen auch Reste des Arbeitsspeichers enthalten kann. Es lässt sich grundsätzlich zwischen

- einem Disk-Slack und
- einem File-Slack

unterscheiden.

Der *File-Slack* bezeichnet den nicht genutzten Teil eines Clusters. Man stelle sich beispielsweise eine Datei mit einer Größe von einem Byte im Dateisystem ▸NTFS vor. Unter Berücksichtigung der Clustergröße des Dateisystems (siehe Tab. 4.2) sind also 4 095 Byte des Clusters ungenutzt. Diese können jedoch von keiner anderen Datei verwendet werden. Dies ist der sogenannte *File-Slack*. Die Datei besitzt also eine *logische Größe* von einem Byte und eine *physische Größe* von 4 096 Byte.

Bei einem *File-Slack* wird zwischen einem ▸*RAM-Slack* und einem *Drive-Slack* unterschieden. Der ▸*RAM-Slack* ist der Bereich des letzten Bytes der tatsächlichen Datei zur nächsten Sektorgrenze. Früher wurde dieser Bereich mit Daten aus dem ▸RAM überschrieben, was Schadsoftware jedoch gerne zur Verschleierung ihrer Funktionen ausgenutzt hat. Daher wird bei modernen Betriebssystemen dieser Bereich mit Nullen gefüllt.

Der *Drive-Slack* ist der Bereich vom Ende des ▸*RAM-Slacks* bis zur Clustergrenze. Abb. 4.3 verdeutlicht die Unterschiede.

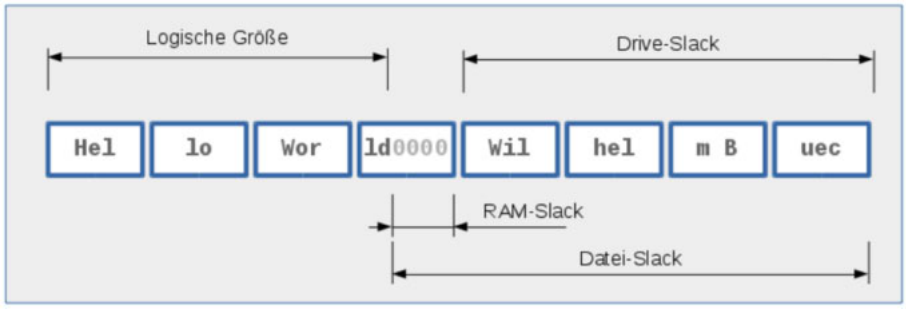

Abb. 4.3: Slack-Bereiche

Der *Disk-Slack* ist der nicht partitionierte Bereich eines Datenträgers. Sofern einzelne Sektoren am Ende, aber noch innerhalb einer Partition, keinem Cluster mehr zugeordnet wurden, spricht man von einem *Volume-Slack*.

4.4.4 Das Dateisystem NTFS

Das Dateisystem ▸NTFS wird von allen modernen Windows-Systemen verwendet und besitzt hierdurch einen weltweiten Marktanteil von über 70 %. Es wird sowohl von privaten Computern und Arbeitsplatz-PCs als auch von Servern eingesetzt. Daher sollte man ein paar Hintergrundinformationen zu diesem Dateisystem kennen.

Im Dateisystem ▸NTFS werden alle Dateien und Verzeichnisse in einer zentralen Datenbank, der Master File Table (▸MFT) verwaltet. Diese Datenbank speichert folgende Metainformationen:

- Dateinamen
- Dateigröße
- Zeitstempel
- Dateiinhalt
- Zugriffsrechte
- Freigaben
- Verweis auf Cluster
- Allocation Flag

Diese Informationen werden dauerhaft in der ▸MFT gespeichert. Wenn eine Datei gelöscht wird, wird diese nicht vollständig vom Datenträger entfernt, sondern lediglich das *Allocation Flag* auf „gelöscht“ gesetzt.

Darüber hinaus unterscheidet die ▸MFT zwischen

- *residenten Daten* und
- *nicht residenten Daten.*

Als *resident* werden Daten bezeichnet, wenn sie direkt in die ▸MFT geschrieben werden. Dies setzt voraus, dass diese Daten eine maximale Größe von 1 024 Bytes besitzen. Um erkennen zu können, ob eine Datei *residente Daten* enthält, die in der ▸MFT gespeichert sind, existiert ein fest definiertes Byte, das sogenannte *Resident Flag*. Das *Resident Flag* besitzt eine feste Position (Offset 8) innerhalb eines ▸MFT-Eintrags und unterscheidet zwischen zwei Werten. Alle *residenten Daten* besitzen an dieser Stelle den Wert *0x00*. Für *nicht residente* Daten ist an diesem Offset der Wert *0x01* hinterlegt.

Die Unterscheidung zwischen *residenten Daten* und *nicht residenten Daten* ist wichtig, da hierdurch beim Auswerten eines Datenträgers der Eindruck einer doppelten Nutzung eines Clusters entstehen kann. Die Daten sind zwar Bestandteil der Datei, jedoch in der ▸MFT abgespeichert.

Des Weiteren werden für jeden ▸MFT-Eintrag vier Zeitstempel in der Datenbank abgelegt. Die Einträge umfassen das Erstellungsdatum, das Änderungsdatum, das Änderungsdatum des ▸MFT-Eintrags sowie das Datum des letzten Zugriffs. Jeder Eintrag besitzt eine Größe von 1 024 Byte und beinhaltet Attribute mit variabler Länge. Die Zeitstempel werden im *Header Standard Information Attribute* gespeichert.

In den *Filename Attributes* werden zusätzliche Zeitstempel sowie der *Short Filename* und der *Long Filename* gespeichert. Für eine forensische Analyse werden jedoch ausschließlich die Informationen im *Standard Information Attribute* verwendet.

Unter Linux können diese Informationen mit dem Befehl *stat* abgerufen werden. Abb. 4.4 zeigt die Ausgabe des *Standard Information Attribute* einer Beispieldatei.

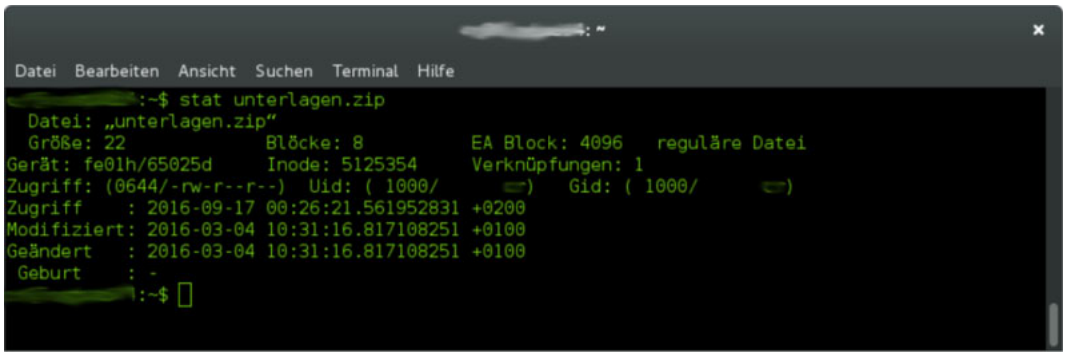

Abb. 4.4: Standard Information Attribute der Datei „unterlagen.zip“

Darüber hinaus zeigt der Befehl die Zugriffsrechte und die Anzahl der verwendeten Cluster.

Auf einem Datenträger kann nicht jeder Dateiinhalt vom Dateisystem zusammenhängend geschrieben werden. Man spricht hierbei von *Fragmentierung*. Diese *Fragmentierung* ist durch sogenannte *Data Run Lists* innerhalb der ▸MFT abgebildet.

Eine *Data Run List* besteht dabei aus folgenden Informationen:

- Länge des Elements (Data Run List) in Bytes
- Anzahl der Bytes für den Anfangscluster
- Anzahl der Bytes für die Größe der Daten
- Information, ob eine Datei fragmentiert ist

Anhand dieser *Data Run Lists* kann eine fragmentierte Datei auch wieder zusammengesetzt werden, wenn sie bereits gelöscht wurde.

4.4.5 Swap und Suspend

Der Swap-Bereich eines Betriebssystems ist eine Auslagerungsdatei, in die nicht benötigte Daten aus dem ▸RAM übertragen werden, die bei Bedarf wieder eingelesen werden. Bei Windows-Systemen wird hierzu die Datei *pagefile.sys* verwendet. Unter Linux-Systemen erfolgt die Ablage auf der *Swap*-Partition und kann mit dem Befehl *top* ausgelesen werden. Bei Mac OS-Systemen sind diese Auslagerungsdateien im Ordner */private/var/vm/swapfile* zu finden. Abb. 4.5 zeigt den Inhalt eines beispielhaften *Swap*-Bereichs.

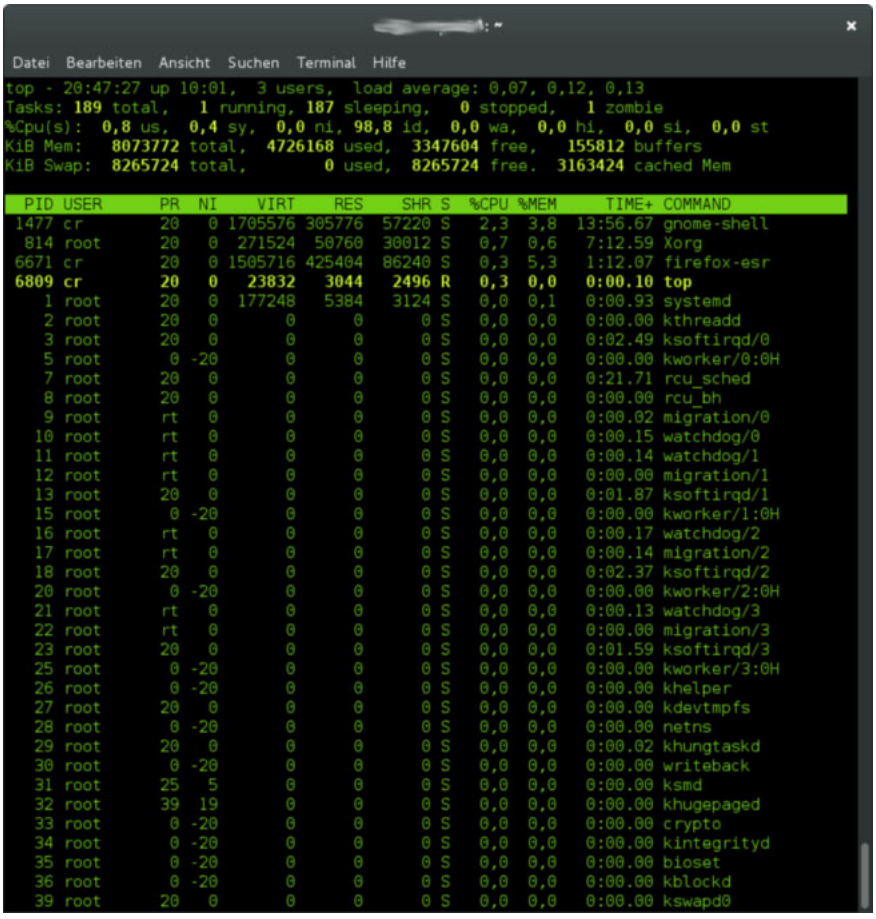

Abb. 4.5: Beispielhafter Inhalt eines Swap-Bereichs

Alle modernen Computer beherrschen einen sogenannten Ruhezustandsmodus (suspend-to-disk). Dieser wird normalerweise nach einer längeren Inaktivität des Benutzers automatisch aktiviert, um Strom zu sparen und beispielweise die Laufzeit des Akkus bei mobilen Geräten zu erhöhen. Gleichzeitig dient dieses auch *Hibernation*-Modus genannte Verfahren zum Verringern der Startzeiten des Computers.

Hierzu werden der gesamte ▸RAM-Inhalt und andere temporäre Systemeinstellungen in eine Datei übertragen und bei Benutzerinteraktion wieder ausgelesen. Bei Windows-Systemen wird hierzu die Datei *hiberfil.sys* verwendet. Bei Linux-Systemen wird der Systemstatus in den *Swap*-Bereich ausgelagert.

Übung 4.4:

Warum sind Swap- und Hibernation-Dateien für Forensiker besonders von Interesse?

4.4.6 Hardlinks und Symbolic Links

Eine Dateiverknüpfung kann bei Betriebssystemen als *Hardlink* oder als *Symbolic Link* umgesetzt werden. Ein *Hardlink* regelt dabei die interne Vater-Kind-Beziehung des Dateisystems. Die Vater-Kind-Beziehung beinhaltet, dass eine erstellte Datei bei modernen Betriebssystemen lediglich auf das Verzeichnis referenziert wird. In Abhängigkeit des verwendeten Dateisystems wird dies über eine *I-Node*- oder über eine *File-Record*-Referenz gewährleistet.

Grundsätzlich erhöht diese Referenz einen internen Zähler des Dateisystems um den Wert 1, wenn eine Datei in einem Verzeichnis erstellt wird. Wenn diese Datei gelöscht wird, wird der interne Zähler um den Wert 1 verringert. Ein *Hardlink* kann nur auf eine Datei, nicht auf ein Verzeichnis angewendet werden.

Beispiel 4.6 zeigt die Erstellung eines *Hardlinks* unter Linux.

Beispiel 4.6:

```
ln beispiel.datei beispiel-hardlink.datei
```

Der *Symbolic Link*, auch *Softlink* genannt, ist eine Verknüpfung, die auf eine andere Datei oder ein anderes Verzeichnis verweist. Wenn diese Datei gelöscht wird, verweist die Verknüpfung ins Leere. Beispiel 4.7 zeigt die Erstellung eines *Symbolic Links* unter Linux.

Beispiel 4.7:

```
ln -s beispiel.date beispiel-softlink.datei
```

Ein *Symbolic Link* kann sowohl auf eine Datei als auch auf ein Verzeichnis angewendet werden. Darüber hinaus kann es auch dateisystemübergreifend eingesetzt werden.

4.4.7 Megabyte und Mebibyte

Die Begriffe „Kilobyte“, „Megabyte“, „Gigabyte“ usw. sind vielen Benutzern bekannt. Etwas weniger verbreitet sind die Begriffe „Kibibyte“, „Mebibyte“ und „Gibibyte“. Diese bezeichnen sogenannte Binärpräfixe, also Datenmengen zur Basis 2. Kibibyte (KiB) wird folgendermaßen festgelegt:

Satz 4.1:

$$2^{10} = 1\,024 \text{ KiB}$$

Demgegenüber sind die Präfixe „Kilo“, „Mega“ oder „Giga“ sogenannte Dezimalpräfixe, also Präfixe zur Basis 10. Kilobyte (KB) wird folgendermaßen festgelegt:

Satz 4.2:

$$10^{6} = 1\,000 \text{ KB}$$

Die Verwendung der Dezimalpräfixe hat sich aus der Elektrik (zum Beispiel „Kilowatt“) abgeleitet und war bei den kleinen Datengrößen in der Vergangenheit nicht weiter bemerkenswert. Mit steigender Datenmenge wächst auch die sprachliche Ungenauigkeit. Eine Festplatte, deren Speicherkapazität vom Hersteller mit einem Terabyte angegeben ist, hat daher lediglich 931,323 Gibibyte an verfügbarem Speichervolumen, da Windows den Speicherplatz binär berechnet, jedoch Dezimalpräfixe anzeigt.

Beispiel 4.8:

$$\frac{1\,000\,000\,000\,000}{\left(2^{30}\right)} = 931{,}323 \text{ Gibibyte (GiB)} \tag{4.1}$$

Die International Electronical Commission (▸IEC) startete daher im Jahr 1996 einen Versuch, dieser Ungenauigkeit durch die Einführung einer standardisierten Benennung des Speichervolumens mit Binärpräfixen zu begegnen, und veröffentlichte die Norm ▸IEC 60027-2. Die Begrifflichkeiten der Norm haben bisher jedoch keinen Eingang in den Alltagsgebrauch gefunden.

Die Verwendung der Dezimalpräfixe ist jedoch nicht gänzlich falsch. Es existieren sehr wohl Speichermedien, deren Konzeption den Einsatz von Dezimalpräfixen rechtfertigt. Tab. 4.3 zeigt die jeweiligen Einsatzbereiche.

Tab. 4.3: Übersicht der Speichermedien und ihrer Präfixe

Dezimalpräfix	Binärpräfix
optische Datenträger ohne CD (z.B. DVD, BluRay)	CD
Mechanische und flashbasierte Speichermedien (z.B. HDD, SSD, USB-Sticks, SD-Cards)	Arbeitsspeicher, Cache, Flash-Speicherchips
Übertragungsgeschwindigkeiten (z.B. ▸W-LAN, Internet)	

Bei einer korrekten Schreibweise der Abkürzung wird ein kleines „i“ eingefügt, wenn man den Binärpräfix verwendet. Eine Festplatte mit einem Speichervolumen von einem *Tebibyte* wird beispielsweise daher mit *TiB* und nicht mit *TB* abgekürzt.

Hinweis:

Das Systemprogramm *dd* unterscheidet bei der Verwendung des Parameters *bs* zwischen Binär- und Dezimalpräfixen.

Übung 4.5:

Welche Kapazität wird bei einem USB-Stick angezeigt, der eine Speichergröße von 3 GB besitzt?

4.5 Datenwiederherstellung und -analyse

Für die Datenwiederherstellung und die Datenanalyse werden oftmals grafische Werkzeuge eingesetzt. Je nach Betriebssystem existieren sowohl kostenlose als auch kostenpflichtige Anwendungen. Ein kostenloses Werkzeug für Linux-Systeme ist das Digital Forensics Framework (▸DFF) der Firma ArxSys. Abb. 4.6 stellt die Anwendung ▸DFF dar.

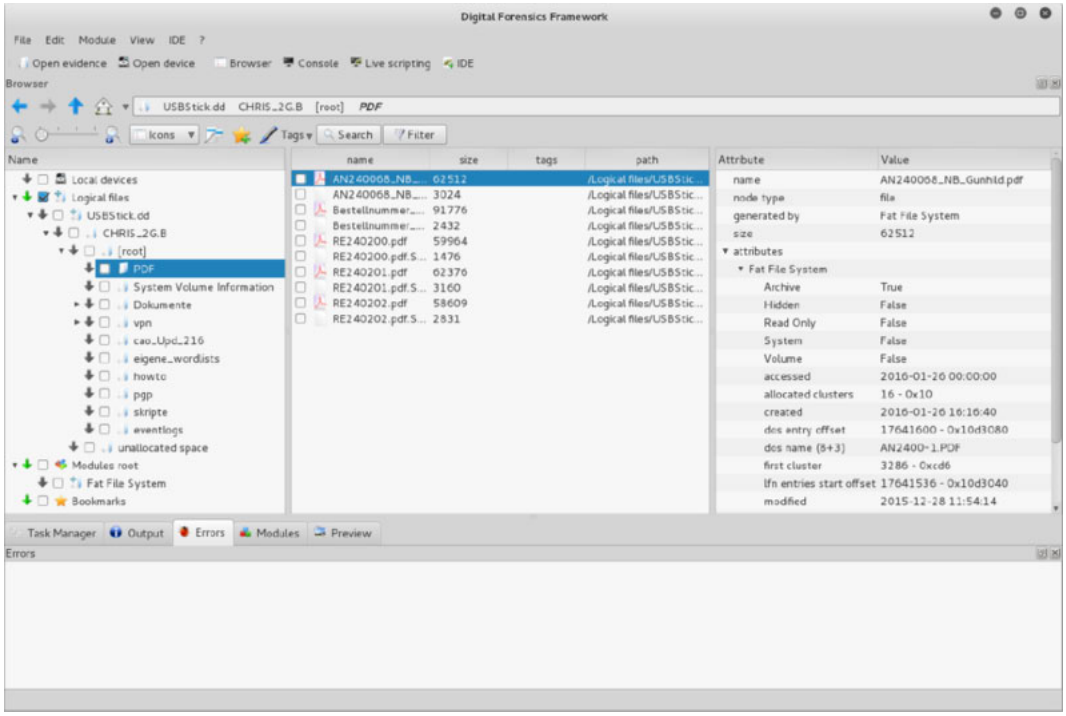

Abb. 4.6: Auswertung eines USB-Sticks mit DFF

Die Anwendung erlaubt die Einbindung von Datenträgerabbildern in den in Abschnitt 4.4 genannten Formaten und auch von ganzen Datenträgern. Zur Auswertung können zu jeder Datei die jeweiligen Metainformationen und dazugehörige Bereiche im Speicher als Hexadezimal-Code und als American-Standard-Code-for-Information-Interchange(▸ASCII)-Darstellung angezeigt werden.

Die Anwendungen zur Datenanalyse unter Windows oder Mac OS besitzen vergleichbare Funktionalitäten. Tab. 4.4 zeigt eine Übersicht von verschiedenen kostenlosen Werkzeugen für Windows-, Linux- und Mac OS-Betriebssysteme und ihren jeweiligen Einsatzzweck.

Tab. 4.4: Werkzeuge zur Datenwiederherstellung und -analyse

Name	Hersteller	Betriebssystem	Einsatzzweck
FTK Imager	Access Data Group	Windows	Datenanalyse
Digital Forensics Framework	ArxSys S.A.S	Linux	Datenanalyse
Recuva	Piriform Ltd.	Windows	Wiederherstellung gelöschter Dateien
PC Inspektor File Recovery	CONVAR EUROPE Ltd.	Windows	Wiederherstellung gelöschter Dateien, Carving
Testdisk u. Photorec	Christophe Grenier	Windows, Linux	Datenwiederherstellung, ▸MBR-Reparatur
Diskdigger	Defiant Technologies, LLC.	Windows	Datenwiederherstellung, Carving
Glary Undelete	Glarysoft Ltd.	Windows	Datenwiederherstellung

Doch wie analysiert man die Daten und differenziert zwischen wichtigen und unwichtigen Informationen? Der einfachste Weg hierfür ist der Abgleich der zu untersuchenden Daten anhand des Hashwerts mit anderen Daten.

Beispielsweise können für alle Originaldaten des Betriebssystems Hashwerte erstellt werden, die dann mit den Hashwerten der zu untersuchenden Daten abgeglichen werden. Dabei können übereinstimmende Hashwerte von weiteren Untersuchungen ausgeschlossen werden.

Möchte man beispielsweise herausfinden, ob ein Mitarbeiter sensible Daten aus einem Unternehmen gestohlen hat, könnte man ein Datenträgerabbild auf das Vorhandensein bestimmter Hashwerte prüfen, nachdem man diese von den Originaldateien des Unternehmens erstellt hat.

4.5.1 Carving

In der IT-Forensik bezeichnet der Begriff „Carving“ (engl. Schneiden) das Durchsuchen des Byte-Streams eines Datenträgers oder Datenträgerabbilds nach bestimmten Dateiinhalten. Die in den Metadaten des Dateisystems vorhandenen Informationen werden dabei nicht berücksichtigt. Das Ergebnis eines Carving-Prozesses sind die reinen Nutzdaten. Das Werkzeug *Photorec* (siehe Abschnitt 4.5) beherrscht beispielsweise Carving.

Der Vorteil dieses Prozesses ist es, dass auch Daten aus einem Dateisystem wiederhergestellt werden können, dessen Metadatenstruktur zerstört wurde.

Ein schwerwiegender Nachteil dieses Vorgehens ist der Verlust sämtlicher Metainformationen zu einer Datei, wie beispielsweise Erstellungsdatum oder Name des Erstellers. Daher sollte diese Möglichkeit stets die letzte Wahl des IT-Forensikers bei der Datenwiederherstellung sein.

Grundsätzlich kann Carving mithilfe unterschiedlicher Algorithmen durchgeführt werden. Ein weitverbreitetes Vorgehen ist die Suche nach den Nutzdaten einer Datei im Byte-Stream. Wenn beispielsweise eine Datei des Formats ▸JPEG File Interchange Form (▸JFIF) der Joint Photographic Experts Group (▸JPEG) wiederhergestellt werden soll (z.B. „bild.jpg“), überprüft man zunächst, ob diese EXIF-Informationen enthält.

Hierzu überprüft man den Hex-Wert, der in Offset 00 auf den Hexwert *FF D8FF E* folgt. Wenn keine EXIF-Informationen vorhanden sind, lautet der Wert *hex E0*. Wenn die Datei EXIF-Informationen besitzt, lautet der Wert *hex E1*. Abb. 4.7 zeigt die Hex-Werte einer ▸JFIF-Datei im Beispiel.

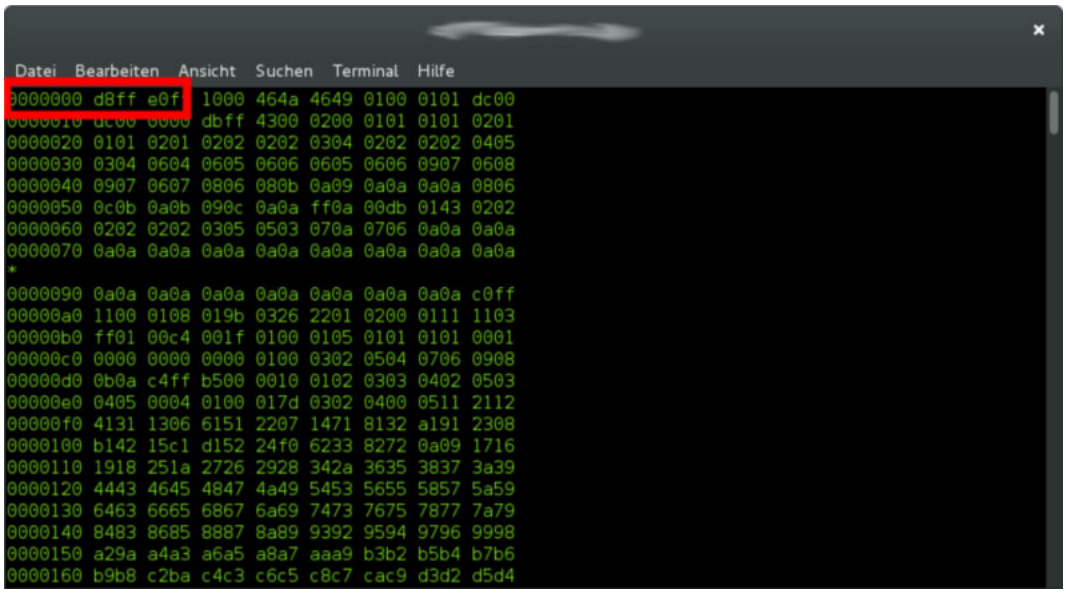

Abb. 4.7: Hex-Darstellung einer JFIF-Datei ohne EXIF-Informationen

Der *Header* der Datei besitzt eine Größe von 3 Bytes und beginnt wie bereits erwähnt mit dem Wert *hex FF D8FF E*. Diese Größe ist in der entsprechenden Formatspezifikation definiert.

Im nächsten Schritt sucht man das Ende des *Footers* der Datei. Dieser endet mit dem Wert *FFD9* und besitzt eine fest definierte Größe von 2 Byte. Beim „Carving“ wird diese Datenstruktur nun aus dem Datenabbild kopiert und mit der entsprechenden Dateiendung auf einem neuen Datenträger neu zusammengesetzt.

Das Erkennen und Zusammensetzen des Dateiformats ist der wesentliche Bestandteil einer *Carving*-Anwendung. Hier kann durch eine intelligente Anwendungslogik ein Datenträgerabbild effizient ausgelesen und entsprechend wiederhergestellt werden. Des Weiteren unterscheidet sich die Qualität der eingesetzten Anwendung im Umgang mit fragmentierten oder teilweise überschriebenen Dateien (z. B. fehlender Datei-Footer).

4.5.2 Block Hashing

Das *Block Hashing* ist eine Methode der Datenanalyse, um Daten aus einem defekten Dateisystem wiederherzustellen. Hierzu werden partielle Hashwerte von einer Liste mit Originaldateien erstellt. In einem weiteren Schritt wird das defekte Dateisystem partiell gehasht und die Ergebnisse miteinander verglichen.

Jedoch ist der Beweiswert dieser Methode in Abhängigkeit der für die Hashwerte eingestellten Blockgröße sehr kritisch zu sehen. Wenn die Blockgröße sehr klein gewählt wurde, steigt die Wahrscheinlichkeit, dass Fragmente auf dem Datenträger mit Teilen der Originaldateien übereinstimmen.

Wenn die Blockgröße groß genug gewählt wurde, lässt sich lediglich damit beweisen, dass sich die Datei auf dem Datenträger befunden hat. Ein Erstellungsdatum oder andere Metainformationen der Datei können mit dieser Methode jedoch nicht wiederhergestellt werden.

4.5.3 Analyse von Daten

Es gibt eine Vielzahl von Ansätzen, wie Daten eines Datenträgers analysiert werden können, ob man nun eine verdächtige Datei sucht oder eine bestimmte Information, wie beispielweise eine besuchte ▸IP-Adresse oder einen Domainnamen. Ohne den Einsatz von entsprechenden Werkzeugen gleicht diese manuelle Auswertung einer Suche nach der Nadel im Heuhaufen.

Möchte man beispielsweise ein Datenträgerabbild nach bestimmten Schlüsselwörtern durchsuchen, unterstützt die Linux-Werkzeugsammlung *The Sleuth Kit* bei der Aufbereitung und dem Durchsuchen des Abbildes.

The Sleuth Kit ist eine Sammlung diverser Werkzeuge für Windows, Linux und Mac OS. Die Werkzeuge wurden in den Programmiersprachen C und Perl geschrieben. Die Sammlung beinhaltet ein Werkzeug

- zur Auswertung von Partitionsinformationen eines Datenträgerabbilds,
- zur Erstellung von Datenbanken mit Informationen über die Dateien,
- zur Auswertung und zum Vergleich von Dateiinhalten,
- zur Wiederherstellung und Auswertung der Metadateninformationen sowie
- zur Wiederherstellung und zum Carving von Dateien.

Des Weiteren existiert mit der Anwendung *Autopsy* eine grafische und webbasierte Oberfläche für *The Sleuth Kit*, die den Benutzer im Analyseprozess unterstützt. Eine grundlegende Funktion der Werkzeugsammlung ist die Erstellung einer SQlite-Datenbank mithilfe des Befehls *tsk_loaddb*, die sämtliche für eine Auswertung relevanten Informationen erfasst und aufbereitet.

Hierdurch wird die Auswertung eines Datenträgerabbilds erheblich vereinfacht. Wenn man beispielsweise sämtliche Dateien eines bestimmten Dateiformats mit seinen entsprechenden Metainformationen anzeigen lassen möchte, wertet man die Tabelle *tsk_files* der erstellten Datenbank mit einem Structured-Query-Language(▸SQL)-Statement aus. Abb. 4.8 zeigt die Tabellenübersicht einer entsprechenden Datenbank.

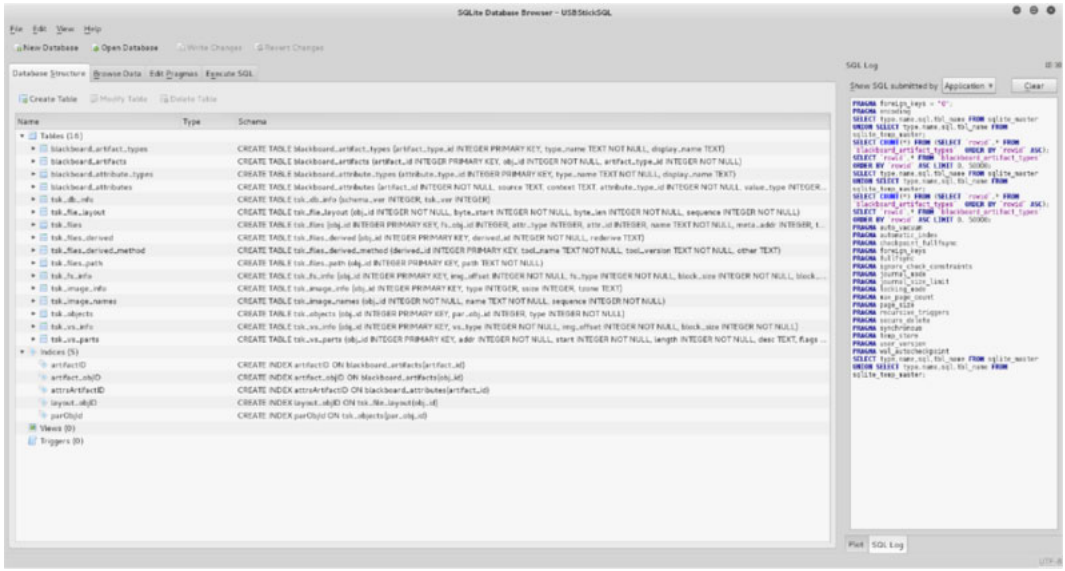

Abb. 4.8: Tabellenübersicht mit tsk_loaddb

4.6 Ergebnisdarstellung und -präsentation

Die Darstellung und die Präsentation der Ergebnisse sollte zielgruppengerecht und nachvollziehbar erfolgen. Ein fachkundiger Dritter sollte in der Lage sein, die Ergebnisse nicht nur nachzuvollziehen, sondern sie auch selbstständig nachzustellen. Darüber hinaus sollte die Präsentation alle in Abschnitt 4.1 gestellten Fragen beantworten können.

Auch hierfür existieren für jedes Betriebssystem verschiedene Werkzeuge. Ein weitverbreitetes Werkzeug zur Auswertung von verschiedenen Protokolldateien und die chronologische Darstellung ihrer Inhalte ist das Framework *log2timeline* und dessen Nachfolger *plaso*. Beide Werkzeuge wurden von der Schwedin Kristinn Gudjonsson entwickelt und stehen für alle gängigen Betriebssysteme zur Verfügung.

Sowohl *log2timeline* als auch *plaso* lesen eine Vielzahl von Diensten und Betriebssystemen aus und stellen deren Ergebnisse in unterschiedlichen Dateiformaten zur Verfügung (z. B. csv, tab, mactime, sqlite, beedocs). *log2timeline* kann folgende Artefakte auslesen:

- Webserver- und Proxy-Logdateien
- Browsercaches und Lesezeichen
- Windows-Ereignisprotokolle

- ▸MFT-Einträge
- Datenbankserverprotokolle
- Windows-Client-Artefakte und Registry-Informationen
- Protokolle diverser Virenschutzsoftwares
- Skype-Protokolle
- diverse Netzwerkprotokolle der ▸OSI-Schicht 4

Die erstellten Ergebnisse lassen sich mithilfe anderer Werkzeuge, wie *GnuPlot* für Linux-Systeme oder *BeeDocs* für Mac OS-Systeme, grafisch aufbereiten.

Zusammenfassung

Die IT-Forensik ist eine Disziplin, die in den letzten Jahren zunehmend wichtiger wurde. Je stärker Informationstechnik unsere Arbeitswelt und unser Privatleben durchdringt, desto wichtiger wird eine gerichtlich verwertbare Nachvollziehbarkeit von Handlungen in dieser Sphäre. Hierzu muss gewährleistet sein, dass diese immateriellen Informationen nicht durch einen Dritten manipuliert wurden.

Eine forensische Untersuchung sollte grundsätzlich folgenden Anforderungen genügen:

- *Akzeptanz*: Die angewandten Methoden und Schritte sollten in der Fachwelt beschrieben und allgemein anerkannt sein. Bei einem Einsatz neuartiger Verfahren sollte zunächst der Nachweis der Korrektheit erbracht werden.
- *Glaubwürdigkeit*: Die angewandten Verfahren und Funktionalitäten müssen erwiesenermaßen robust sein.
- *Wiederholbarkeit*: Die Ergebnisse der Verfahren müssen durch Dritte reproduziert werden können.
- *Integrität*: Die sichergestellten Daten müssen zu jedem Zeitpunkt gegen ungewollte Veränderung geschützt sein. Ein entsprechender Beweis muss jederzeit erbracht werden können.
- *Ursache und Auswirkungen*: Die Auswahl der Methoden muss eine logisch nachvollziehbare Beweiskette zwischen Ereignissen und Beweisspuren ermöglichen.
- *Dokumentation*: Jeder Schritt des Ermittlungsprozesses muss angemessen dokumentiert sein.

Die Datensammlung sollte grundsätzlich flüchtige und nichtflüchtige Daten umfassen. Alle Speichermedien, auf denen Daten gesichert werden, sollten angemessen vorbereitet werden. Beim Kopiervorgang sollte das zu sichernde Speichermedium gegen eine Kontaminierung mit Fremddaten geschützt werden.

Die Analyse sollte stets auf der Kopie erfolgen. Besonders interessant für die Datenanalyse können Slack-Bereiche sein. In diesen Bereichen können noch Datenreste vorhanden sein, die nicht überschrieben wurden. Des Weiteren kann auch ein Blick in die ▸MFT und die Unterscheidung zwischen residenten und nicht residenten Daten sowie die Auswertung des Swap- und Suspend-Bereichs für die Analyse interessant sein.

Wenn nichts mehr hilft, sollte man Methoden wie die Kaltstartattacke, das Carving und das Block Hashing einsetzen. Diese sind jedoch mit Vorsicht zu genießen.

Die Ergebnisdarstellung und -präsentation sollte zielgruppengerecht erfolgen. Die Vorgehensweise sollte für fachkundige Dritte nachvollziehbar dokumentiert sein und jederzeit nachgestellt werden können.

Aufgaben zur Selbstüberprüfung

4.1 Warum sollte man keine Schnellformatierung für eine forensische Untersuchung verwenden?

4.2 Was bedeutet dd?

4.3 Wie funktioniert eine Kaltstartattacke?

4.4 Wie hoch ist die Wahrscheinlichkeit eines doppelten Fingerabdrucks?

4.5 Was ist ein Datei-Slack?

A. Lösungen der Übungen im Text

1.1 Erlaubnisprinzip: Das Prinzip der Erlaubnis bedeutet, dass Zugriffe auf ein System grundsätzlich verboten sind und einer ausdrücklichen Erlaubnis bedürfen. Damit verbunden ist das Prinzip der Vollständigkeit, nämlich dass jeder Zugriff auf das System zu prüfen ist.

1.2 Probe, Penetrate, Persist, Propagate, Paralyze

1.3
- Überwachung und Analyse des Netzwerkverkehrs
- Einschränkung der Bewegungsfreiheit im Netzwerk
- Prinzip der minimalen Rechte (need-to-know)
- Trennung von Netzwerken mit unterschiedlichem Schutzbedarf
- Autorisierung der Netzwerkteilnehmer

1.4 Schicht 2: Data Link Layer

2.1 Diese Komponente bereitet die Analyseergebnisse auf, ermittelt Statistiken und gibt sie benutzergerecht aus.

2.2 Ein nicht freigegebener (aber auch nicht sicherheitsrelevanter) Zustand wird als sicherheitsrelevant gemeldet.

2.3 Netzwerkbasierte ▸IDS zur Erkennung von Single Step Attacks sind die am häufigsten verwendete ▸IDS-Variante.

3.1 Ein Paketfilter arbeitet auf den ▸OSI-Schichten 3 und 4.

3.2 Der SYN-Scan baut die ▸TCP-Verbindung nicht vollständig auf. Daher wird die Verbindung auf dem Zielsystem nicht protokolliert.

3.3 Heuristisch bedeutet, mit unvollständigen Informationen und entsprechender Deduktion eine Lösung zu finden. Intrusiv bedeutet aufdringlich.

4.1 Die forensische Untersuchung sollte für fachkundige Dritte nachvollziehbar sein und im Bedarfsfall durch einen weiteren Forensiker nachgestellt werden können. Die Datensammlung sollte manipulations- und verlustfrei erfolgen. Des Weiteren gelten folgende Anforderungen:

- *Akzeptanz:* Die angewandten Methoden und Schritte sollten in der Fachwelt beschrieben und allgemein anerkannt sein. Bei einem Einsatz neuartiger Verfahren sollte zunächst der Nachweis der Korrektheit erbracht werden.
- *Glaubwürdigkeit:* Die angewandten Verfahren und Funktionalitäten müssen erwiesenermaßen robust sein.
- *Wiederholbarkeit*: Die Ergebnisse der Verfahren müssen durch Dritte reproduziert werden können.
- *Integrität*: Die sichergestellten Daten müssen zu jedem Zeitpunkt gegen ungewollte Veränderung geschützt sein. Ein entsprechender Beweis muss jederzeit erbracht werden können.

- *Ursache und Auswirkungen*: Die Auswahl der Methoden muss eine logisch nachvollziehbare Beweiskette zwischen Ereignissen und Beweisspuren ermöglichen.
- *Dokumentation*: Jeder Schritt des Ermittlungsprozesses muss angemessen dokumentiert sein.

4.2 Klonen bezeichnet eine vollständige Kopie eines Datenträgers.

4.3 Der folgende Befehl erstellt eine Kopie des Datenträgers sdX im Verzeichnis /home/<BENUTZERNAME>/ als Datei usbstick.img unter einem Linux-System:

```
dd if = /dev /sdX of = /home /<BENUTZERNAME> /usbstick.img
```

4.4 Der Arbeitsspeicher wird in diese Dateien ausgelagert.

Daher können sie ein Abbild der flüchtigen Daten enthalten.

4.5 Es wird eine Kapazität von 2 793,96 Mebibyte angezeigt.

B. Lösungen der Aufgaben zur Selbstüberprüfung

1.1 Die Prinzipien zur Konzeption sicherer Systeme lauten:

- Erlaubnis
- Vollständigkeit
- minimale Rechte („need-to-know“)
- Benutzerakzeptanz
- offener Entwurf

Sie sind in Abschnitt 1.1 genannt.

1.2 Die Phasen der Kompromittierung lauten:

- Probe
- Penetrate
- Persist
- Propagate
- Paralyze

Sie sind in Abschnitt 1.2 genannt.

1.3 Die Faktoren zur ▸CVSS-Bewertung lauten:

- Basisfaktoren (Base Metric Group)
- temporäre Faktoren (Temporal Metric Group)
- Umgebungsfaktoren (Environmental Metric Group)

Sie sind in Abschnitt 1.3 genannt.

1.4 Die Prinzipien der Netzwerkverteidigung lauten:

- Überwachung des Netzwerkverkehrs
- Reduzierung der Netzwerk- und Systemdienste
- Bewegungsfreiheit einschränken
- Aktualisieren des Netzwerks

Sie sind in Abschnitt 1.4 genannt.

2.1 Die ▸IDS-Komponenten sind:

- Ereigniskomponenten
- Analysekomponenten
- Datenbankkomponenten
- Reaktionskomponenten

Siehe auch Abschnitt 2.1.

2.2 Die Sprachenarten eines ▸IDS sind:

- Attackensprachen (Attack Languages)
- Exploit-Sprachen (Exploit Languages)
- Ereignissprachen (Event Languages)
- Erkennungssprachen (Detection Languages)

- Reaktionssprachen (Response Languages)
- Reportsprachen (Report Languages)
- Korrelationssprachen (Correlation Languages)

2.3 Die Erkennung einer *Multi Step Attack* (Mehr-Schritt-Attacke) wird erst durch die Analyse von charakteristischen Merkmalen in mehreren Protokolleinträgen festgestellt. Lesen Sie Abschnitt 2.6.2 zu Multi-Step-Attacken.

3.1 Passive Scanner eignen sich vor allem zur Analyse von Netzwerken. Aktive Scanner versuchen die gefundenen Schwachstellen ausnutzen, um die Verwundbarkeit entsprechend offenzulegen. Sie finden die Lösung in Abschnitt 3.4.

3.2 Die ▸MAC-Adresse! Lesen hierzu Abschnitt 3.4.2.

3.3 Die ▸TCP-Verbindung wird vollständig aufgebaut. Lesen Sie Abschnitt 3.4.3.

3.4 Pivoting bedeutet, dass ein System überwunden und ein zweites System angegriffen wird. Lesen Sie Abschnitt 3.4.6. Sie finden die Lösung im letzten Absatz.

4.1 Eine Schnell- oder Normalformatierung eines Speichermediums löscht keine Daten, sondern lediglich die Partitionstabelle des Master Boot Record, also das Inhaltsverzeichnis des Speichermediums (Schnellformatierung) und sucht nach defekten Sektoren auf dem Speichermedium (Normalformatierung). Lediglich das bitweise Überschreiben mit *null* löscht diese wirklich.

4.2 dd steht für Convert and Copy. Lesen hierzu Abschnitt 4.2.2.

4.3 Die *Kaltstartattacke* kann in zwei Varianten durchgeführt werden. Die erste Möglichkeit ist es, den ▸RAM einzufrieren und in einen anderen Computer zu transplantieren. Diese Variante kann man anwenden, wenn das entsprechende System nur eine Stromkabellänge von einem Kühlraum entfernt steht oder wenn man der Grundausstattung eine ausreichende Menge an Kältespray hinzufügt.

4.4 Die Wahrscheinlichkeit ist 1 zu 64 Milliarden.

4.5 Der *File-Slack* bezeichnet den nicht genutzten Teil eines Clusters.

C. Abkürzungsverzeichnis

AFF	Advanced Forensic Format
APFS	Apple File System
ASCII	American Standard Code for Information Interchange
BDSG	Bundesdatenschutzgesetz
BIOS	Basic Input/Output System
BSSID	Basic Service Set Identification
CERT	Computer Emergency Response Team
CIDF	Common Intrusion Detection Framework
CSA	Cisco Security Agent
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerabilities Scoring System
DARPA	Defense Advanced Research Projects Agency
DMZ	Demilitarized Zone
DFF	Digital Forensics Framework
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DMA	Direct Memory Access
E01	Expert Witness Format
E01x	Extended Expert Witness Format
ESSID	Extended Service Set Identifier
ext4	Extended Filesystem Version 4
FAT	File Allocation Table
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol
GRE	Generic Routing Encapsulation
GPS	Global Positioning System
GPT	GUID Partition Table

HFS	Hierarchical File System
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IDMEF	Intrusion Detection Message Exchange Format
IDS	Intrusion Detection System
IEC	International Electronical Commission
IETF	Internet Engineering Task Force
IIS	Internet Information Services
IEEE	Institute of Electrical and Electronics Engineers
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPS	Intrusion Prevention System
IPSec	Internet Protocol Security
ISO	International Organization for Standardization
JFIF	JPEG File Interchange Form
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
MAC	Media Access Control
MBR	Master Boot Record
MD5	Message Digest Algorithm 5
MFT	Master File Table
MIME	Multipurpose Internet Mail Extensions
Nmap	Network Mapper
NTFS	New Technology File System
NAT	Network Address Translation
NFS	Network File System
NGFW	Next Generation Firewall
NVT	Network Vulnerability Tests
OpenVAS	Open Vulnerability Assessment System
OSI	Open Systems Interconnection

PCAP	Packet-Capture
POST	Power-on Self Test
PHP	Hypertext Preprocessor
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RFC	Request for Comments
RPC	Remote Procedure Call
SHA	Secure Hash Algorithm
SEM	Security Event Management
SIEM	Security Information and Event Management
SIM	Security Information Management
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SSID	Service Set Identifiers
SSL	Secure Sockets Layer
STATL	State Transition Analysis Technique Language
TCP	Transmission Control Protocol
TKG	Telekommunikationsgesetz
TLS	Transport Layer Security
TMG	Telemediengesetz
VBR	Virtual Boot Record
VLAN	Virtual LAN
VPN	Virtual Private Network
WAN	Wide Area Network
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area-Network
WPS	Wi-Fi Protected Setup
UDP	User Datagram Protocol
UTM	Unified Threat Management

D. Literaturverzeichnis

Agham, V. (2014).
Marktübersicht IDS/IPS. In: KES (2014), Vol. 8, S. 32–36.

Agham, V. (2016). *Unified Threat Management.* In: International Research Journal of Engineering and Technology (IRJET). Vol. 3 Issue: 4, S. 32–36, April 2016.

Anderson, R. (2008).
Security Engineering: A Guide to Building Dependable Distributed Systems.
Indianapolis: Wiley Publishing.

Bejtlich, R. (2005).
The TAO of Network Security Monitoring. Boston: Pearson Education.

Collins, M. (2014).
Network Security Through Data Analysis. 1. Aufl., Sebastopol: O'Reilly Media.

CVSS: Common Vulnerability Scoring System. (2016).
V3 Development Update https://www.first.org/cvss

Eckert, C. (2009).
IT-Sicherheit Konzepte – Verfahren – Protokolle. 6. Aufl., München: Oldenbourg.

Goschonneck, A. (2004).
Computer-Forensik. 3. Aufl., Heidelberg: dpunkt.

Halderman, A.; Schoen, S.; Heninger, N.; Clarkson, W.; Paul, W.; Calandrino, J.; Feldman, A.; Appelbaum, J.; Felten, E. (2008).
Lest We Remember: Cold Boot Attacks on Encryption Keys.
Proc. 17th USENIX Security Symposium (Sec 08): San Jose.

International Electrotechnical Commission: Prefixes for binary multiples. (2016).
http://www.iec.ch/si/binary.htm

Kennedy, D.; O'Gorman, Jim.; Kearns, D.; Aharoni, M. (2011).
Metasploit: The Penetration Tester's Guide. San Francisco: No Starch Press.

Meier, M. (2007).
Intrusion Detection effektiv! Modellierung und Analyse von Angriffsmustern.
Heidelberg: Springer.

Nessus Vulnerability Scanner|Tenable Network Security. (2016).
https://www.tenable.com/products/nessus-vulnerability-scanner

Nmap: The Network Mapper – Free Security Scanner. (2016). https://www.nmap.org/

Ruef, M. (2007). *Die Kunst des Penetration Testing.* Böblingen: Computer und Literatur.

Wright, C.; Kleiman, D.; Sundhar, S. (2008).
Overwriting Hard Drive Data: The Great Wiping Controversy.
Heidelberg: Springer.

E. Abbildungsverzeichnis

F. Tabellenverzeichnis

G. Sachwortverzeichnis

H. Einsendeaufgabe Typ A

Intrusion Detection und Intrusion Prevention

Einsendeaufgabencode:
SRN04-XX1-K03

Name:	Vorname:	Tutor/-in:
Postleitzahl und Ort:	Straße:	Datum:
Matrikel-Nr.:	Studiengangs-Nr.:	Note:
Heftkürzel: **SRN04**	Druck-Code: **0517K03**	Unterschrift:

Bitte reichen Sie Ihre Lösungen über StudyOnline ein. Falls Sie uns diese per Post senden wollen, dann fügen Sie bitte die Aufgabenstellung und den Einsendeaufgabencode hinzu.

1. Erstellen Sie ein OSI-Modell in Tabellenform mit drei weiteren Spalten und ordnen Sie
   a) die Prinzipien zur Konzeption sicherer Systeme und
   b) die Prinzipien zur Netzwerkverteidigung

   den jeweiligen Schichten zu. Beachten Sie, dass ein oder mehrere Prinzipien mehrfach genannt werden können.

   Nennen Sie danach je Schicht drei konkrete Maßnahmen zu deren Absicherung. Die Maßnahmen können dabei sowohl auf technischer als auch organisatorischer oder infrastruktureller Ebene erfolgen.

   30 Pkt.

2. Ein kleines Unternehmen bittet Sie, die Netzwerksicherheit zu überprüfen und gegebenenfalls Sicherheitsmaßnahmen vorzuschlagen. Leider liegt keine Dokumentation des Netzwerks vor, da der zuständige Administrator dies für unnötig hält. Das Unternehmen handelt mit Büchern, die es über einen eigenen Onlineshop vertreibt.

   Erstellen Sie daher zunächst einen Netzplan eines Netzwerks mit folgenden Systemen:

   - 7 physikalische Client-PC für folgende Abteilungen: Geschäftsführung, IT-Administration, Vertrieb, Buchhaltung, Marketing, Logistik und Webentwicklung
   - 4 physikalische Server: 2 Webserver (redundant), 1 Datenbankserver für die Finanzbuchhaltung und das CRM-System, 1 Datei- und Active Directory-Server
   - 2 Drucker: 1 Geschäftsführung und Buchhaltung, 1 Vertrieb
   - 1 Netzwerkswitch
   - 1 Router

   a) Welches sind die wichtigsten zwei Systeme des Unternehmens? Begründen Sie Ihre Antwort.

   b) Wählen Sie als Nächstes ein passendes IDS-System und platzieren Sie es im Netzwerk so, dass ein Angriff möglichst schnell entdeckt wird. Begründen Sie bitte auch diese Antwort.

c) Binden Sie nun eine passende Honeypot-Variante in das Netzwerk ein. Beachten Sie dabei den Unternehmenszweck.

55 Pkt.

3. Es hätte ein solch schöner freier Sonntag werden können. Doch plötzlich klingelt Ihr Handy und die Geschäftsführung aus Aufgabe 2 ist am anderen Ende. Der Geschäftsführer, Herr M., teilt Ihnen Folgendes mit: „Hallo, hier ist M. und mir ist etwas ganz Blödes passiert! Mein Hund hat meine am Computer angeschlossene USB-Festplatte zwischen die Zähne bekommen, das Kabel abgerissen und auf der Festplatte herumgekaut. Mein Administrator sagt, die Festplatte sei defekt. Auf dieser Festplatte (1 TiB) befindet sich meine persönliche Datenbank mit allen Passwörtern. Es existiert kein Backup! Können Sie die Datenbank (Dateigröße 5 MiB) wiederherstellen?“

   Natürlich können Sie. Beschreiben Sie stichpunktartig Ihr professionelles Vorgehen und nennen Sie auch Ihre Werkzeuge. Beachten Sie dabei, dass die Festplatte zwar einen mechanischen Defekt besitzt, aber noch wenige Male starten wird. Behandeln Sie sie also pfleglich. Dem Hund geht es übrigens gut!

   15 Pkt.

Gesamt: 100 Pkt.