

HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCE

LEHRBRIEF

für das Modul

Betriebssysteme

macOS

Autor: Prof. Ronny Bodach

Hinweise

Herausgeber:

©2022 Hochschule Mittweida

Hochschule Mittweida - University of Applied Sciences

Fakultät Computer- und Biowissenschaften

Technikumplatz 17

09648 Mittweida

2. Auflage (05.07.2024)

Redaktionelle Bearbeitung: Prof. Ronny Bodach

Das Werk einschließlich seiner Teile ist urheberrechtlich geschützt. Jede Verwendung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung der Verfasser unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Inhaltsverzeichnis

1	Betriebssystem macOS als Client	3
1.1	Einführung in macOS	3
1.1.1	Einführung	3
1.1.2	Aufbau und Grundlagen	5
1.1.3	Sicherheitsfeatures.....	8
1.2	macOS-Lab.....	12
1.2.1	Bedienung.....	12
1.2.2	macOS Lab und Image Einbindung.....	22
1.2.3	Bootcamp Besonderheiten und Parallels	26
1.3	Speicherstrukturen und Datenformate	28
1.3.1	Mac FHS und Speicherstrukturen.....	28
1.3.2	Datenformate SQLite und Plist.....	30
1.3.3	Plattformabhängige Speichermethode Plist	30
1.3.4	Plattformunabhängige Speichermethode SQLite.....	36
1.3.5	SQLite – Wiederherstellung gelöschter Records	38
1.4	macOS Artefakte	46
1.4.1	Zuletzt genutzte Elemente und Nutzeraktivitäten	46
1.4.2	Spotlight und erweiterte Metadaten	48
1.4.3	Gelöschte Dateien	51
1.4.4	Schlüsselbund/Keychain	53
1.4.5	Logdateien	55
1.4.6	Mac Disk Images.....	63
1.4.7	Time Machine und lokale Backups	68
1.4.8	Weitere Artefaktanalysen	72
1.5	macOS Apps.....	73
1.5.1	Kommunikations-Apps	73
1.5.2	Browser Artefakte	77
1.5.3	Cloud.....	78
1.5.4	iOS Backups	81

1 Betriebssystem macOS als Client

1.1 Einführung in macOS

1.1.1 Einführung

Das auf Unix basierende Mac OS X ist die zehnte Ausgabe des Apple eigenen Betriebssystems für Macintosh Computer. Die ursprünglich auf Motorola Chipsätzen basierenden Apple Computer wurden zwischenzeitlich ausschließlich mit Intel Chipsätzen hergestellt. Derzeit wird der Umstieg von Intel auf ARM Technologie durchgeführt.

Im September 2000 wurde die erste OSX Beta (Kodiak) an die Entwickler verteilt. Bis einschließlich Version 10.7 hieß das Betriebssystem Mac OS X ab 10.8 wurde es nur noch OS X genannt mit der Einführung von 10.12 wurde es erneut umbenannt und heißt nun macOS.

1.1.1.1 Mac OS (X) Versionen

Die folgende Aufzählung bietet eine Übersicht über die verschiedenen Versionen von Mac OS X.

- Mac OS X Public Beta – code name Kodiak
- Mac OS X 10.0 – code name Cheetah
- Mac OS X 10.1 – code name Puma
- Mac OS X 10.2 – also marked as Jaguar
- Mac OS X Panther – 10.3
- Mac OS X Tiger – 10.4
- Mac OS X Leopard – 10.5
- Mac OS X Snow Leopard – 10.6 (requires purchase)
- Mac OS X Lion – 10.7 – OS X Lion (requires purchase)
- OS X Mountain Lion – 10.8 (requires purchase)
- OS X Mavericks – 10.9 (free)
- OS X Yosemite – 10.10 (free)
- OS X El Capitan – 10.11 (free)
- macOS Sierra – 10.12 (free)
- macOS High Sierra – 10.13 (free)
- macOS Mojave – 10.14 (free)
- macOS Catalina – 10.15 (free)
- macOS 11.0: Big Sur (2020)
- macOS 12.0: Monterey (2021)

1.1.1.2 Modellpalette

Es gibt verschiedene Modelle. Dazu zählen:

- Mac Book Pro 2008
- iMac G3
- Mac G4
- Power Mac G5
- Mac Book Pro
- Mac Book Air
- iMac 5
- Mac Mini
- Apple TV
- Time Capsule
- AirPort Extreme
- Mac Mini M2
- Mac Pro M2

Diese Modelle sind in den folgenden Abbildungen dargestellt.





1.1.2 Aufbau und Grundlagen

1.1.2.1 Grundaufbau von OSX

OSX ist in vier Schichten aufgebaut. Die erste Schicht ist die Benutzerebene Aqua, welche die grafische Benutzerschnittstelle (GUI) darstellt. Die nächste Schicht ist die Anwenderprogrammierschicht. Diese bietet Programmierschnittstellen (APIs) wie Cocoa (und früher Carbon) und ist Java basiert. Die dritte Schicht ist die Bereitstellungsebene, welche das Grafik-Subsystem (Quartz mit Quartz Compositor, OpenGL, Audio/Video (QuickTime) und weiteres enthält. Die letzte Schicht bildet die Basisebene. Dies ist Darwin, das Kern-Betriebssystem.

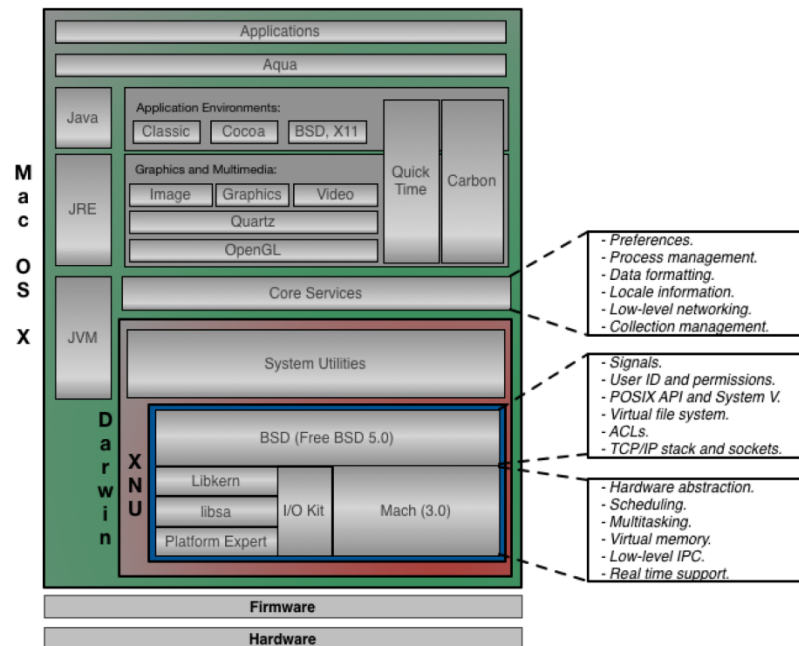
OSX ist ein Nachkomme von NeXTSTEP und genau genommen eine (proprietäre) Software-Distribution, wobei Darwin, die Basisebene, von BSD abgeleitet ist und damit ein (freies) Unix das eigentliche Betriebssystem ist.

Durch Darwin (vererbt aus BSD) verfügt OS X über Fähigkeiten wie Speicherplatzschutz, präemptives Multitasking, Mehrbenutzerfähigkeit, erweitertes Speichermanagement und symmetrisches Multiprocessing (SMP). Darwin wurde unter die quelloffene Lizenz Apple Public Source License gestellt.

Der Kernel wurde gegenüber NeXTStep vollkommen überarbeitet - während NeXTStep noch einen reinen Mach-Mikrokern verwendete, setzt OS X bzw. Darwin auf einen sogenannten Hybridkernel: Dabei werden einige Funktionen in den Kernel integriert, allerdings nicht so viele wie bei einem monolithischen Kernel.

Als Basis Kernel wurde weiterhin Mach verwendet und mit Teilen des monolithischen FreeBSD Kernel ergänzt. Der Kernel heißt XNU (X is Not Unix).

Die folgende Abbildung stellt den Grundaufbau von OSX dar.



1.1.2.2 Unterstützte Dateisysteme

Mac OS X nutzte das Dateisystem HFS und dessen Erweiterung HFS+. Dieses von Apple entwickelte Dateisystem wurde auch für externe Datenträger verwendet und kann mit einem Windows basierten System nicht gelesen werden.

Mac OS X kann auch das FAT12/16/32 Dateisystem lesen und schreiben. Damit ist es möglich, externe Datenträger wie Speicherkarten und USB-Sticks für den Multibetriebssystembetrieb einzurichten.

Seit der Einführung von macOS X 10.12 wurde das Apple Dateisystem HFS+ durch APFS (Apple File System) ersetzt.

OS X unterstützt verschiedene weitere lokale Dateisysteme wie NTFS, exFAT, UFS, UDF, sowie ZFS (die beiden letztgenannten nur lesend). Der Schreibzugriff auf NTFS wurde in Mac OS X 10.6 hinzugefügt, ist standardmäßig jedoch abgeschaltet und muss durch einen Eintrag in fstab aktiviert werden.

Unterstützte Netzwerkdateisysteme sind AFP, FTP, NFS, SMB/CIFS und Web-DAV.

Mit der Zusatzsoftware *MacFUSE* und entsprechenden Plugins wie NTFS-3G (für Schreib/Lesezugriff auf NTFS-Datenträger bis OS X 10.6) sind weitere Dateisystemtypen unter OS X verfügbar. Hierbei werden zusätzlich eine Menge für die Forensik relevante Dateisysteme nutzbar, wie durch das Mounten von EWF-Images, BDE-Volumes oder Ähnlichem.

1.1.2.3 Benutzerverwaltung

Dem Ursprung von Mac OS X angelehnt ist das Betriebssystem Multiuserfähig und bietet eine entsprechende Benutzerverwaltung mit Benutzern und Gruppen an.

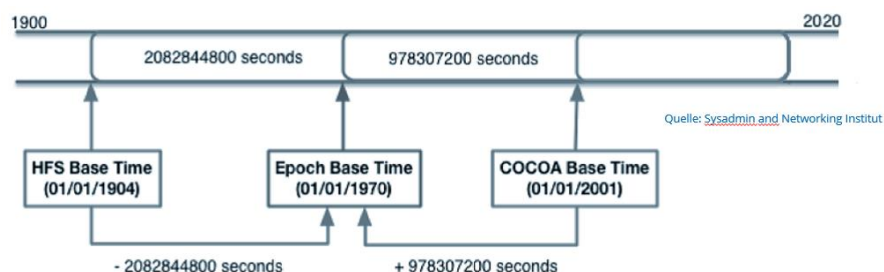
OSX unterscheidet zwischen den normalen Benutzern (user), dem Systemverwalter (admin) und dem Superuser (root). Normale Benutzer können keine Änderungen am System vornehmen oder Software außerhalb ihrer Benutzerordner installieren. Alle von Usern gestartete Programme werden mit den entsprechenden Nutzerrechten des Users ausgeführt.

Die Benutzer der Gruppe admin verfügen über weitergehende Rechte. Sie dürfen systemweite Einstellungen vornehmen, Software installieren und verfügen über Schreibzugriff auf diverse Systemverzeichnisse.

Nur nach gesonderten Authentifizierungen können tiefgreifende Änderungen am System vorgenommen werden. Ein nutzbares Root-Benutzerkonto, das dauerhaft über Berechtigungen des Superusers verfügt, gibt es nach einer Systeminstallation nicht. Zwar gibt es einen Benutzer „root“, dieser ist jedoch standardmäßig deaktiviert. Kann jedoch explizit aktiviert werden.

1.1.2.4 Zeitstempel

OSX verwendet drei unterschiedliche Zeitstempel. Dazu zählen HFS Time, Epoch und Cocoa. HFS Time ist ein 4 Byte HexWert, der die Sekunden seit dem 01. Januar 1904 zählt. Epoch ist ebenfalls ein 4 Byte HexWert, welcher die Sekunden jedoch seit dem 01. Januar 1970 zählt. Cocoa ist ein 64 Bit – Integer, der die Sekunden seit dem 01. Januar 2001 zählt.



1.1.3 Sicherheitsfeatures

1.1.3.1 System Integrity Protection (SIP)

Der Systemintegritätsschutz wird auch als *rootless* bezeichnet. Er wurde mit OS X El Capitan (OS X 10.11) im Jahr 2015 eingeführt. Dieser Systemintegritätsschutz besteht aus drei Schutzebenen. Dazu zählen der Schutz von Inhalten und Dateisystemberechtigungen von Systemdateien und -verzeichnissen, der Schutz von Prozessen gegen Code-Injection, Laufzeitanbindung (wie Debugging) und Dtrace sowie der Schutz vor unsignierten Kernel-Erweiterungen ("kexts").

Das Kernstück des Systemintegritätsschutzes ist der Schutz der systemeigenen Dateien und Verzeichnisse gegen Änderungen durch Prozesse ohne eine bestimmte "Berechtigung", auch wenn sie vom Root oder einem Benutzer mit Root-Rechten (sudo) ausgeführt wird. Dies geschieht entweder durch Hinzufügen eines erweiterten Dateiattributs zu einer Datei oder einem Verzeichnis oder durch Hinzufügen der Datei oder des Verzeichnisses zu „/System/Library/Sandbox/rootless.conf“. Per default werden „/System“, „/sbin“, „/bin“, „/usr“ und „/Applications“ durch SIP geschützt.

Der Systemintegritätsschutz kann nur (ganz oder teilweise) von außerhalb der Systempartition deaktiviert werden. Zu diesem Zweck stellt Apple das Befehlszeilendienstprogramm *csrutil* bereit, welches über ein Terminalfenster innerhalb des Wiederherstellungssystems oder von einer bootfähigen macOS-Installationsdiskette, die dem NVRAM des Geräts ein Boot-Argument hinzufügt, ausgeführt werden kann. Dazu sind drei Schritte nötig. Beginnend wird „Command + R“ beim booten in der Wiederherstellungskonsole gedrückt. Anschließend wird das Terminal geöffnet und darin der Befehl „*csrutil disable*“ ausgeführt. Nach Änderungen einschalten nicht vergessen.

1.1.3.2 Sealed System Volume (SSV)

Sealed System Volume wurde mit macOS 11 BigSure eingeführt. Es ergänzt das separate schreibgeschützte Systemvolumen (ab macOS 10.15 Catalina). Mithilfe von Sealed System Volume wird der Schutz des Systems gegenüber dem bestehenden Read-only-Volumen der Systemintegritätssicherung (SIP) vertieft.



Während der Installation, sobald System-Volumen installiert, werden kryptografische Hashes für jede Komponente auf dem Volumen berechnet und zu einem Baum (wie ein Merkle-Baum) zusammengesetzt. Der Baum gipfelt in einem einzigen Master-Hash, der als Siegel bezeichnet wird. Die Hashes werden hierbei als Metadaten gespeichert und es wird ein Dateisystem-Snapshot des Volumens erstellt. Anstatt das System-Volumen wie in Catalina schreibgeschützt zu mounten, wird nur der versiegelte Schnappschuss gemountet. Dadurch wird ein weiterer robuster Schutz vor Manipulationen und Fehlern durch unveränderliche Systemdateien erzielt.

Während des Starts prüft macOS BigSur das Seal des Systems. Ist das Seal defekt, bootet das Betriebssystem nicht und muss neu installiert werden. Der Wiederherstellungsmodus bietet eine Option, um diese Prüfung zu deaktivieren, so dass es möglich ist, ein System-Volume anzupassen und es unversiegelt zu starten. Eine Datensicherung von Seal Broken Devices ist nur mittels ASR Befehl möglich.

1.1.3.3 FileVault1

FileVault1 wurde mit Mac OS X 10.3 Panther eingeführt. Es bietet eine reine Softwareverschlüsselung. In FileVault 1 oder Legacy FileVault werden nur Home-Ordner in einem Sparsebundle verschlüsselt. Sparsebundle sind dynamische Dateien (8 MB Default Größe), welche ähnlich zu einem verschlüsselten Container in Truecrypt sind. FileVault1 verursacht Probleme mit Time Machine-Backups (werden nur bei Nutzerabmeldung erstellt vs. Suspend). Weiterhin ist es vergleichsweise einfach zu knacken bzw. zu entschlüsseln mit Benutzerkennwort.

1.1.3.4 FileVault2 ohne Hardwareschutz

FileVault 2 wurde in Mac OS X 10.7 Lion eingeführt und ist ursprünglich ebenfalls Software basiert mit zusätzlichen Maschinenbefehlen für die Crypting Engine auf Intel i CPUs. Es bietet die Verschlüsselung des gesamten Volumes basierend auf dem Benutzerkennwort mittels überschlüsselten Volume Encryption Key VEK. Die Verschlüsselung wird unter Verwendung des XTS-AES-Modus von AES mit einem 256-Bit-Schlüssel durch CPU durchgeführt. Alle Daten, die geschrieben werden, müssen verschlüsselt werden, bevor sie auf die Festplatte geschrieben werden. Alle davon gelesenen Daten müssen entschlüsselt werden, bevor sie verwendet werden können.

1.1.3.5 Hardwareschutz ohne Filevault2

Das Unternehmen Apple führte mit dem Apple T2 Security Chip 2017 einige Funktionen zur Erhöhung der Sicherheit entsprechender Geräte ein. Der T2 (und auch M1) Chip speichert den VEK Schlüssel, der zum Verschlüsseln verwendet wird, auf seiner eigenen Hardware. Der T2-Chip fungiert hierbei als Speichercontroller für die interne SSD, sodass alle Daten, die zwischen dem Intel-Prozessor und der SSD übertragen werden, eine Verschlüsselungsstufe in der Hardware des T2 durchlaufen. Alle Macs mit T2-Chips, mit Ausnahme des Mac Pro 2019, verfügen über einen internen Speicher, der eingelötet ist, um das Entfernen schwierig zu machen.

Die Sicherheitsfunktionen werden von einer dedizierten Hardware durchgeführt (Secure Enclave mit Secure OS), welche nur über Schnittstellen mit der restlichen Hardware kommuniziert. Auf Intel-Macs verlassen sie nie den T2-Chip, sind also nie von dem Intel-Prozessor des Mac aus zugreifbar. Wird die Hardware beschädigt oder zerstört, können die Daten auf dem Datenträger nicht mehr durch diesen entschlüsselt werden. Der Chip kann auch nicht einfach ersetzt werden, da der einzigartige Schlüssel fehlt. Durch diese Funktion kann die Erstellung einer Forensischen Datensicherung stark erschwert oder verhindert werden.

1.1.3.6 FileVault2 mit Hardwareschutz

Wenn FileVault2 auf T2 / M1 Mac aktiviert ist, wird ein VEK in der Hardware abgelegt, der durch einen KeyEncryptionKey (KEK) geschützt wird, der durch das Benutzerkennwort geschützt wird. Dies bedeutet, dass der Benutzer sein Passwort ändern kann, ohne dass das Volume erneut verschlüsselt werden muss. Es ermöglicht zudem die Verwendung spezieller Wiederherstellungsschlüssel, falls das Benutzerpasswort verloren geht.

1.1.3.7 Hardwareschutz und Forensik

Der T2 ist die zweite Generation der Apple T-Serie und wurde seit 2017 in verschiedenen Apple Geräten mit Intel Prozessor verbaut. Mit der Einführung des Apple M1 sind die Funktionen des T2 Chips in den M1 Chip integriert worden, es wird hier also kein extra Chip mehr benötigt.

Ein Hardwareschutz wurde im Boot-ROM integriert. Der Startprozess eines Apple Gerätes mit M1 oder T2 kann nicht manipuliert werden, da der im Boot-ROM gespeicherte Code nicht veränderbar ist. Dieser stellt sicher, dass nur vertrauenswürdige Software im Bootvorgang geladen und ausgeführt wird.

Ein weiteren Sicherheitsaspekt bietet Secure Boot. Bei entsprechender Konfiguration kann verhindert werden, dass unsignierte Betriebssysteme geladen werden. Es besteht auch die Option das die Signatur bei vorhandener Netzwerkverbindung während der Installation bei Apple überprüft wird.

Zusätzlich gibt es Allowed Boot Media. Bei entsprechender Konfiguration kann verhindert werden, dass von einem anderen Gerät gebootet wird.

1.1.3.8 Apple T2 Security Chip

Apple T2 Security Chip ist die zweite Generation der Apple T-Serie und ist ein System on a Chip (SoC). Er bietet mehrere Funktionen und besteht aus einigen Komponenten. Dazu zählen ein verschlüsselter Speicher für Passwörter und Schlüssel, Funktionen zum sicheren Startvorgang, der Schutz und die Verarbeitung von Biometrischen Daten, die Audio- und Videoverarbeitung, die Speichercontroller für internes Solid-State Drive, sowie eine hardwarebeschleunigte AES-Engine zur Ver- und Entschlüsselung. Der Apple T2 besitzt weitere Funktionen und Komponenten zur Diagnose, Spracherkennung und zum Monitoring.

1.1.3.9 Apple Silicon M1

Der Apple M1 Chip ist die erste Generation der Apple M-Serie und ist ebenfalls ein SoC. Der M1 Chip wird als der Hauptprozessor von entsprechenden Apple Geräten verwendet. Bei seinem Design wurden viele Funktionen direkt in diesen Chip integriert. Zu den Funktionen und Komponenten zählen:

- CPU
- GPU
- Unified Memory
- Cache

- Neural Engine
- Verschlüsselter Speicher für Passwörter und Schlüssel
- Funktionen zum sicheren Startvorgang
- Schutz und Verarbeitung von Biometrischen Daten
- Audio- und Videoverarbeitung
- Speichercontroller für internes Solid-State Drive
- Hardwarebeschleunigte AES-Engine zur Ver- und Entschlüsselung

Der Apple M1 besitzt ebenfalls weitere Funktionen und Komponenten zur Diagnose, Spracherkennung und zum Monitoring.

1.1.3.10 Open Firmware Passwort

Firmwarepasswörter sind zum Schutz des BIOS / EFI BIOS gedacht. Sie schützen vor falschen Bootmedien (DVD/USB). Dabei schützen sie den Start Manager vor unberechtigter Nutzung (Wahltaste/Option (⌘) oder Alt) und schützen vor dem Starten des Recovery/Wiederherstellungsmodus (Befehlstaste/Command (⌘) + R). Auf Macs mit Apple-Silicon-Chips lässt sich kein Firmware-Passwort mehr setzen.

1.2 macOS-Lab

1.2.1 Bedienung

Die folgende Abbildung zeigt den Aufbau der Oberfläche von macOS.



1.2.1.1 Schreibtisch/Desktop



Der Schreibtisch ist der Hintergrundbereich der Bildschirmanzeige. Auf ihm sind Symbole für Festplatten, CDs und Server, die an den Mac angeschlossen sind. Es ist möglich, Dateien und Ordner direkt auf dem Schreibtisch abzulegen, um schnell darauf zuzugreifen.

1.2.1.2 Dock

Das Dock wird zum Öffnen von Programmen, Dokumenten, Ordnern und anderen Objekten verwendet. Es wird standardmäßig am unteren Bildschirmrand angezeigt. Wenn ein Fenster minimiert, oder ein Programm geöffnet wird, das nicht im Dock enthalten ist, wird das zugehörige Symbol im Dock angezeigt. Jedes Objekt im Dock besitzt ein Kontextmenü, das einen schnellen Zugriff auf die Befehle ermöglicht, die für das betreffende Objekt verfügbar sind.



1.2.1.3 Menüleiste

Die Menüleiste befindet sich am oberen Rand des Bildschirms. Diese ist, mit Ausnahme des Apple-Menüs ganz links, sowie der Uhrzeit und dem Kontrollzentrum ganz rechts, programmspezifisch und ändert sich, wenn man von einem Programm zum anderen wechselt. Wie das Dock lässt sich die Menüleiste ausblenden und erscheint nur dann, wenn man den Mauszeiger dorthin bewegt.



Außerdem lassen sich für verschiedene Systemfunktionen und manche Anwendungen Symbole der Menüleiste hinzufügen (Menü-Extras), beispielsweise für Spotlight, das WLAN, Time Machine, 1Password oder Carbon Copy Cloner. Auch diese Symbole werden immer angezeigt, egal mit welcher Anwendung man gerade arbeitet.



1.2.1.4 Das Apple-Menü



Das Apple-Menü ganz links mit dem Apfelsymbol dient dazu, immer Zugriff auf bestimmte Funktionen zu haben, egal, in welcher Anwendung man sich gerade befindet. Hier finden sich die Systemeinstellungen und der App Store. Unter „Benutzte Objekte“ sind die zuletzt verwendeten Programme, Dokumente und Server aufgelistet. Hält man die Taste Command (⌘) gedrückt, kann man sich die Objekte im Finder anzeigen lassen.

Über den Befehl „Sofort beenden“ lässt sich ein hängengebliebenes Programm zur Aufgabe bewegen.

Mit „Ruhezustand“ schickt man den Mac in den Schlafmodus. Zum Aufwachen genügt es, eine Taste auf der Tastatur oder die Maustaste zu drücken beziehungsweise auf das

Trackpad zu tippen.

Dann folgen die Befehle zum Neustart des Rechners und zum Ausschalten. Das System fragt vorsichtshalber jeweils nach, ob man dies auch wirklich möchte. Um die Nachfrage zu umgehen, hält man die Taste Option (⌥) gedrückt, wenn man den Befehl auswählt.

Mit den beiden letzten Optionen im Apple-Menü lässt sich der Bildschirm sperren oder der momentan aktive Benutzer am System abmelden.

An oberster Stelle im Apple-Menü befindet sich der Eintrag „Über diesen Mac“, mit dem sich ein Informationsfenster öffnet. Es zeigt die Version des Systems an, den Rechnertyp, den Prozessortyp, die Größe des eingebauten Arbeitsspeichers, von welchem Volume der Mac gestartet ist sowie die Seriennummer.

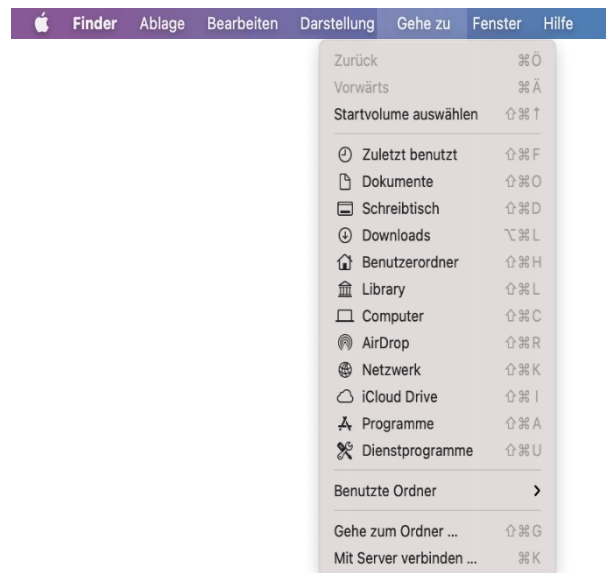


Unter dem Punkt „Festplatten“ findet man Informationen über die verfügbaren Datenträger angezeigt.

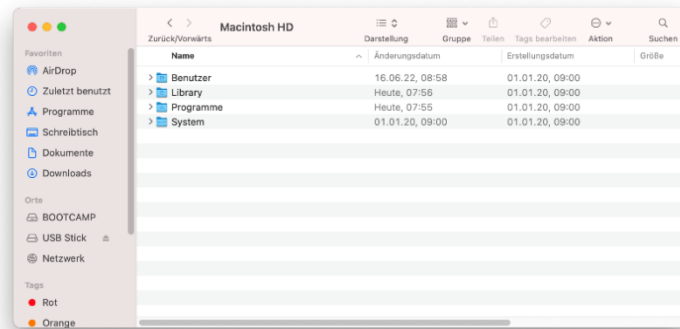


1.2.1.5 Der Finder

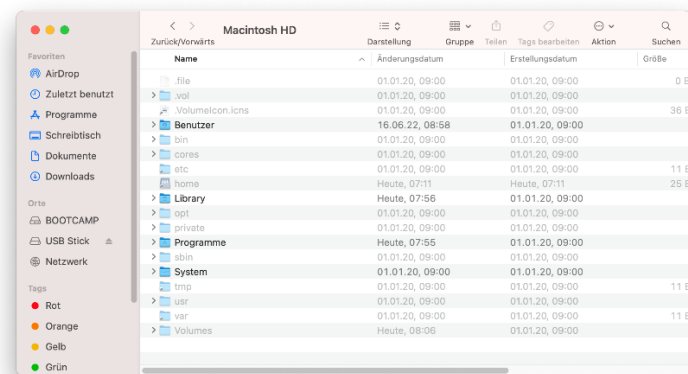
Der Finder ist das Programm, welches für die Verwaltung von Dateien, Programmen, Festplatten (Volumes), Netzwerkverbindungen und Geräten (wie Drucker) zuständig ist. Während der Arbeit mit dem Computer ist der Finder immer aktiv, auch wenn er sich im Hintergrund befindet. Zum Einblenden des Finder-Fensters klicken Sie auf das Finder-Symbol im Dock. Der Finder ist dort immer das erste Symbol. Über den Finder kann der Inhalt des Computers aufgelistet werden (Computer).



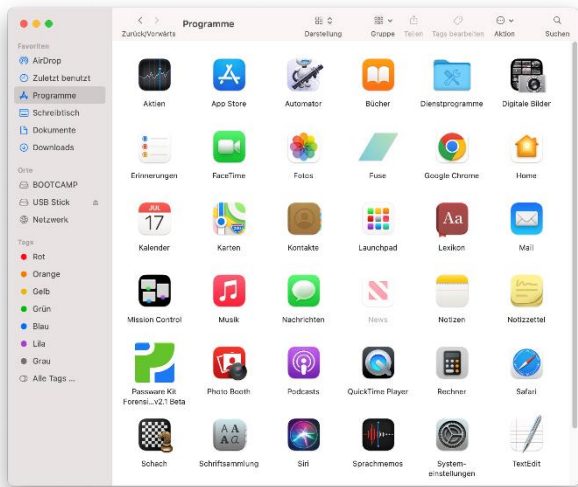
Die standard Ansicht zeigt dabei leider nur sehr wenig Daten an:



Die Vollansicht (**Shift** + Befehlstaste/**Command** (⌘) + .) offenbart weitere Daten:



1.2.1.6 Programme



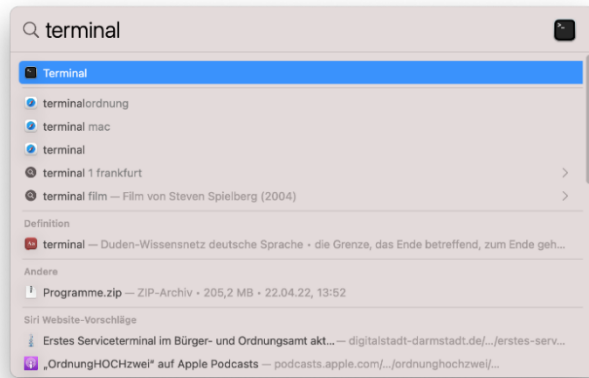
Die installierten Anwendungen werden unter dem Punkt „Programme“ abgelegt. macOS hat viele davon als Standard on Board. Die Installations DVD macOS umfasst ca. 20 GB! Wichtige Punkt sind hier Dienstprogramme und Systemeinstellungen.

Anwendungen werden durch Drag & Drop installiert.

1.2.1.7 Spotlight

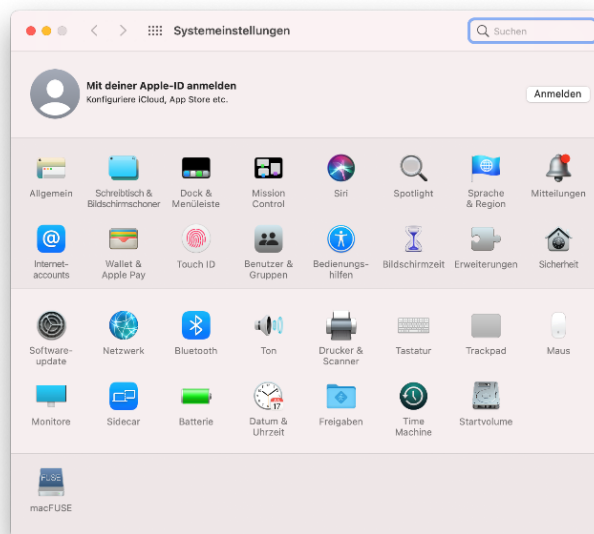
Spotlight ist der Index zum macOS System, mit welchem Sie ganz einfach Informationen auf Ihrem Computer finden können. In vielen Fällen ist es einfacher, Dateien, Ordner und Programme mit Spotlight zu öffnen, anstatt den Finder zu verwenden. Sobald Sie beginnen, Text in das Suchfeld von Spotlight einzugeben, werden die Suchergebnisse sofort in einem Menü unterhalb des Suchfelds angezeigt.





1.2.1.8 Systemeinstellungen

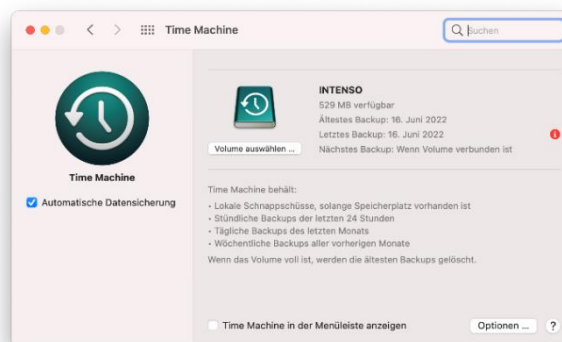
Mithilfe der Systemeinstellungen kann das macOS verwaltet werden. Die Systemeinstellungen befinden sich im Menü "Apple" und im Dock. Wichtige forensische Bestandteile sind Internet Accounts, das Wallet, Benutzer und Gruppen, Erweiterungen, die Time Maschine und das Startvolumen.



Die folgende Abbildung zeigt die Systemeinstellung „Startvolumen“.

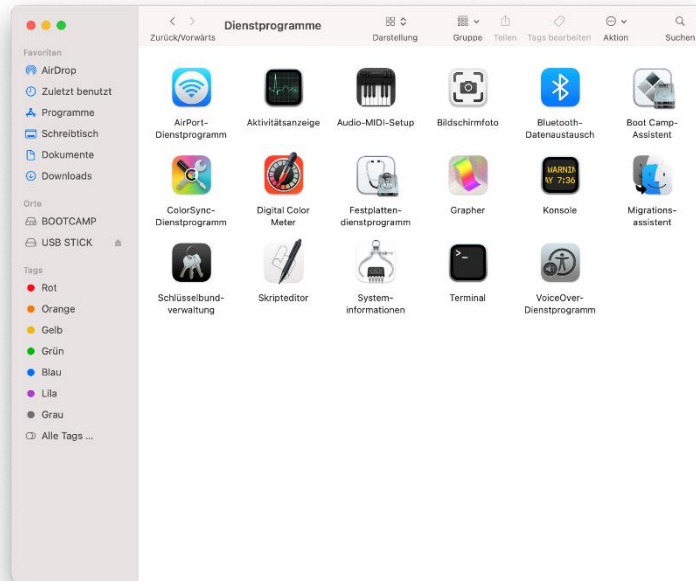


Die folgende Abbildung zeigt die Systemeinstellung „Time Maschine“.

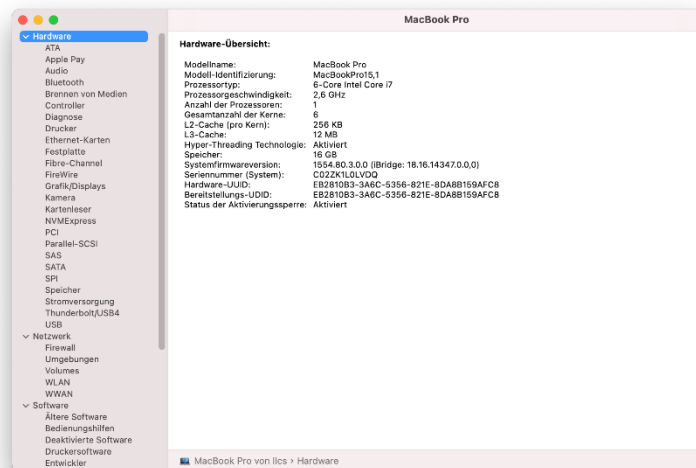


1.2.1.9 Dienstprogramme

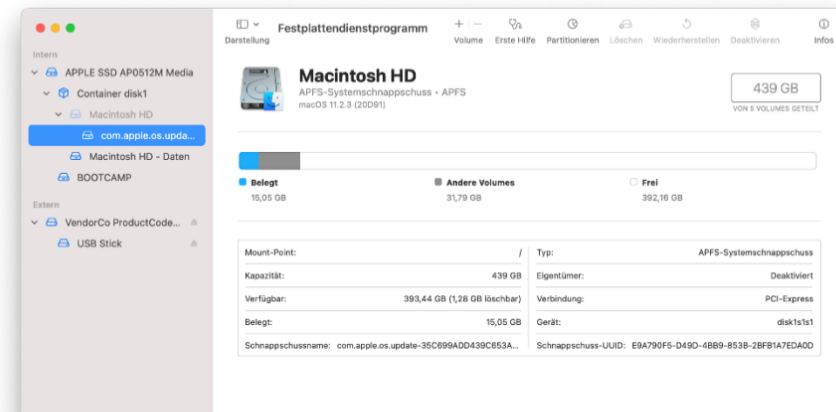
Der Abschnitt Dienstprogramme enthält eine Reihe wichtiger Informationsquellen für eine Untersuchung. Dazu zählen das Festplattendienstprogramm, die Schlüsselbundverwaltung, Systeminformationen, das Terminal und der Bootcamp Assistent.



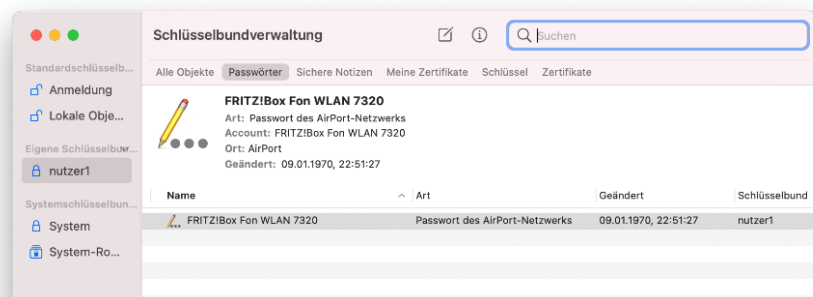
Die folgende Abbildung stellt die Systeminformationen dar.



Die folgende Abbildung zeigt das Festplattendienstprogramm.



Die folgende Abbildung stellt die Schlüsselbundverwaltung / Keychain dar.



1.2.1.10 Terminal

Zsh – die Z shell ist eine UNIX Shell und basiert auf der bash. Sie ist mittlerweile die Standard Shell auf MacOS.



1.2.2 macOS Lab und Image Einbindung

1.2.2.1 Herausforderungen bei der Datensicherung und Analyse von macOS Systemen

Es gibt mehrere Herausforderungen bei der Datensicherung von macOS Systemen. Eine Herausforderung stellt die Verschlüsselung mit Hardwareeinbindung über T2 und M1 Chipsätze dar. Weiterhin entstehen durch fest verbaute Datenträger (SSD on Board) und durch fehlende Zugriffsmöglichkeiten auf die Hardware, wie etwa auf das PCB, Schwierigkeiten.

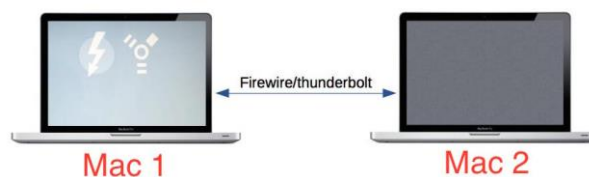
Bezüglich der Analyse ergeben sich ebenfalls Herausforderungen. Dazu zählt, dass Log Formate mit speziellen Binärcodierungen (ähnlich evtx) codiert sind. Weiterhin gibt es Datenspeicherformate mit einer Binärcodierung. Eine zusätzliche Schwierigkeit entstehen durch die Schlüsselbund/Keychain Informationen, welche mit Hardwareeinbindung T2 und M1 arbeiten.

Die Datensicherung wie auch Untersuchung eines macOS Systems lässt sich nur bedingt auf externen Geräten ohne macOS durchführen. Daher empfiehlt es sich ein macOS System als macOS Lab vorzuhalten, um auf Probleme im Umgang mit macOS spezifischen Besonderheiten einzugehen.

Als Vorbereitung von macOS Lab für die Nutzung als Forensic Lab Station wird System Integrity Protection (SIP) des Systems deaktiviert und Homebrew (Paketmanager) installiert. Weiterhin werden die Xcode Umgebung, libewf und xmount installiert.

1.2.2.2 Anschluss von macOS Asservaten an die Forensic Lab Station

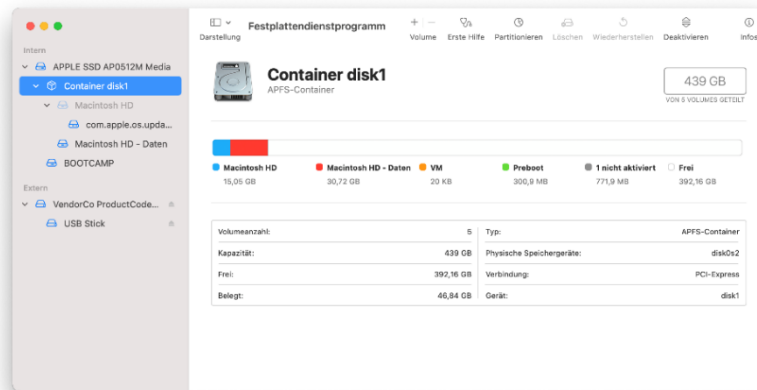
Für den Anschluss von macOS Asservaten an die Forensic Lab Station wird der Target Disk Mode (TDM) genutzt, indem beim booten T gedrückt wird. Danach wird der TDM am Asservat gestartet und das Asservat verhält sich wie eine externe Festplatte (Firewire/Thunderbolt).



Hierbei ergeben sich einige Einschränkungen. Im TDM Mode wird nur die erste Festplatte durchgegeben. Außerdem lässt sich der TDM Mode bei gesetztem Firmwarepasswort nicht starten. Weiterhin ist ein Benutzerpasswort mit Admin Berechtigungen zum Freischalten der Verschlüsselungen notwendig. Beim Anschluss wird die Festplatte automatisch im Read-Write Mode eingebunden, weshalb die Nutzung eines Hardware Schreibblockers unverzichtbar ist.

1.2.2.3 Verhinderung eines Auto Mount an der Forensic Lab Station

Die Auflistung der angeschlossenen Datenträger kann im Festplattendienstprogramm eingesehen werden oder mit dem Terminal Programm „diskutil“:



Die folgende Abbildung zeigt die Ausgabe des Befehls „diskutil list“.

```
nutzer1@MBP-von-llcs ~ % diskutil list
/dev/disk0 (internal, physical):
#:  
0:      GUID_partition_scheme             *500.3 GB   disk0  
1:      EFI EFI                             314.6 MB   disk0s1  
2:      Apple_APFS Container disk1         439.0 GB   disk0s2  
3:      Microsoft Basic Data BOOTCAMP      61.0 GB    disk0s3  
  
/dev/disk1 (synthesized):  
#:  
0:      APFS Container Scheme -             +439.0 GB   disk1  
      Physical Store disk0s2  
1:      APFS Volume Macintosh HD             15.1 GB    disk1s1  
2:      APFS Snapshot com.apple.os.update-... 15.1 GB    disk1s1s1  
3:      APFS Volume Preboot                  300.9 MB   disk1s2  
4:      APFS Volume Recovery                  613.8 MB   disk1s3  
5:      APFS Volume VM                       20.5 KB    disk1s4  
6:      APFS Volume Macintosh HD - Daten    52.0 GB    disk1s5  
  
/dev/disk4 (external, physical):  
#:  
0:      FDisk_partition_scheme              *2.0 GB    disk4  
1:      DOS_FAT_16 USB STICK                 2.0 GB    disk4s1
```

Beim Anschluss von externen Datenträgern wird durch den Disk Arbitrator jeder Datenträger automatisch im Read Write Mode eingebunden. Dies kann durch den Schreibblocker verhindert werden. Eine weitere Möglichkeit dies zu verhindern ist es, den Disk Arbitrator temporär auszuschalten. Dafür wird der Befehl „sudo launchctl unload /System/Library/LaunchDaemons/com.apple.diskarbitrationd.plist“ genutzt. Danach kann der Datenträger ohne Automount angeschlossen und gesichert werden (im gestoppten Zustand kein diskutil verfügbar). Nach der Sicherung kann der Dienst wieder mit dem Befehl „sudo launchctl load /System/Library/LaunchDaemons/com.apple.diskarbitrationd.plist“ gestartet werden.

1.2.2.4 Datensicherung in macOS Systemen

Es gibt sechs verschiedene Möglichkeiten die Datensicherung in macOS Systemen durchzuführen. Dazu zählen der Live Betrieb/Target Disk Mode mit hdiutil, der Live Betrieb/Target Disk Mode mit asr, der kommerzielle Fremdboot mit Cellebrite Macquisition/Digital Collector, der kommerzielle Fremdboot mit Sumuri Recon Imager / Recon ITR, die Datensicherung im macOS Lab mit DD bzw. ewfacquire und die Datensicherung mittels Bildschirmfotos.

Live Betrieb/Target Disk Mode mit hdiutil

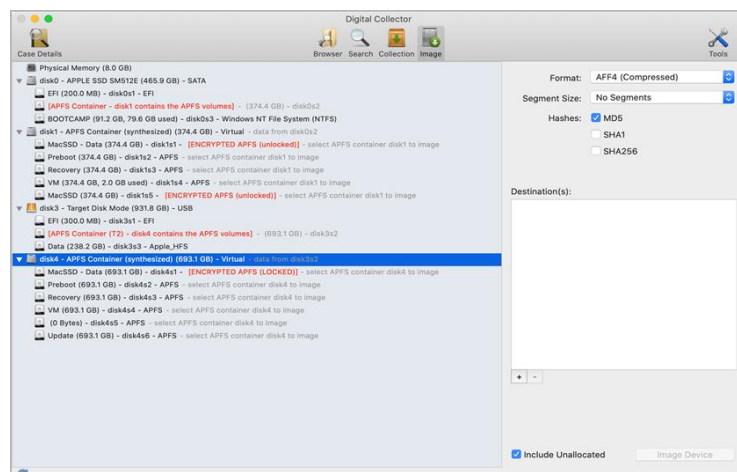
Mit dem Tool hdiutil können Images erstellt und auch eingebunden werden. Der Befehl lässt sich auf einzelne Verzeichnisse anwenden und erstellt ein DMG Container Image: „hdiutil create -srcfolder /Verzeichnisangabe -volname DMGNAME /mountpoint/folder/DMGNAME.dmg“. Der erstellte DMG Container ist ein Image, welches nicht veränderbar ist und für die weitere Verarbeitung wie eine Festplatte aufgebaut ist.

Live Betrieb/Target Disk Mode mit asr

Mit dem Tool asr können Images von Disks und Snapshots erstellt werden. Der Befehl lässt sich auf einzelnen Disks anwenden und erstellt eine Kopie auf ein zweites Volume: „asr restore --source /dev/diskx --target /dev/diskx (--toSnapshot SnapshotUID)“. Mit der option kann von einem Snapshot des Systems eine Kopie erstellt werden. Diese Option kann etwa bei Seal Broken Volumes auf den Snapshot angewendet werden.

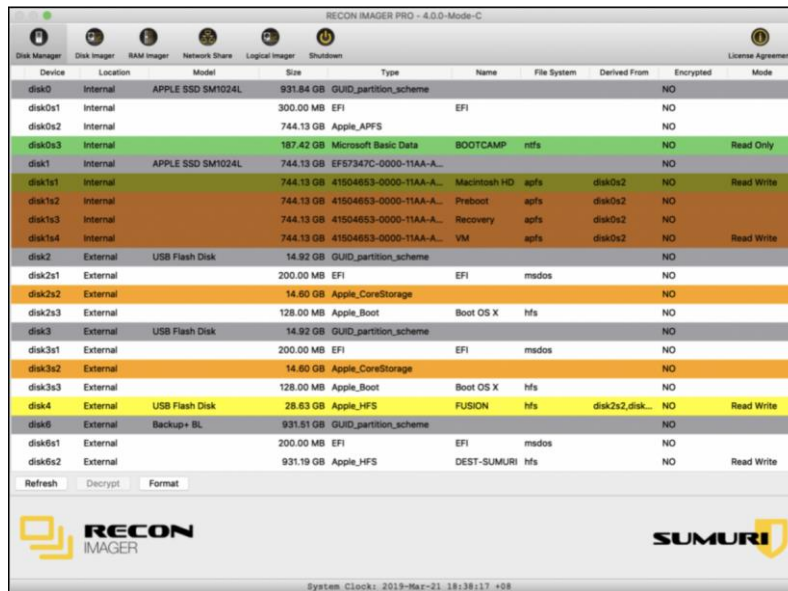
Cellebrite Macquisition/Digital Collector

Hierbei handelt es sich um ein kommerzielles Tool, welches Dateien einzeln oder ganze Images sichern kann. Es entschlüsselt sowohl T2&M1 Hardware Keys wie auch Filevault2 automatisch. Passwörter sind hierfür notwendig!



Sumuri Recon Imager / Recon ITR

Hierbei handelt es sich ebenfalls um ein kommerzielles Tool, welches einzelne Dateien oder ganze Images sichern kann. Es entschlüsselt sowohl T2&M1 Hardware Keys wie auch Filevault2 automatisch. Auch hierbei sind Passwörter notwendig!



Device	Location	Model	Size	Type	Name	File System	Derived From	Encrypted	Mode
disk0	Internal	APPLE SSD SM1024L	931.84 GB	GUID_partition_scheme				NO	
disk0s1	Internal		300.00 MB	EFI	EFI			NO	
disk0s2	Internal		744.13 GB	Apple_APFS				NO	
disk0s3	Internal		187.42 GB	Microsoft Basic Data	BOOTCAMP	ntfs		NO	Read Only
disk1	Internal	APPLE SSD SM1024L	744.13 GB	EF57347C-0000-11AA-A...				NO	
disk1s1	Internal		744.13 GB	41504853-0000-11AA-A...	Macintosh HD	apfs	disk0s2	NO	Read Write
disk1s2	Internal		744.13 GB	41504853-0000-11AA-A...	Preboot	apfs	disk0s2	NO	
disk1s3	Internal		744.13 GB	41504853-0000-11AA-A...	Recovery	apfs	disk0s2	NO	
disk1s4	Internal		744.13 GB	41504853-0000-11AA-A...	VM	apfs	disk0s2	NO	Read Write
disk2	External	USB Flash Disk	14.92 GB	GUID_partition_scheme				NO	
disk2s1	External		200.00 MB	EFI	EFI	msdos		NO	
disk2s2	External		14.60 GB	Apple_CoreStorage				NO	
disk2s3	External		128.00 MB	Apple_Boot	Boot OS X	hfs		NO	
disk3	External	USB Flash Disk	14.92 GB	GUID_partition_scheme				NO	
disk3s1	External		200.00 MB	EFI	EFI	msdos		NO	
disk3s2	External		14.60 GB	Apple_CoreStorage				NO	
disk3s3	External		128.00 MB	Apple_Boot	Boot OS X	hfs		NO	
disk4	External	USB Flash Disk	28.63 GB	Apple_HFS	FUSION	hfs	disk2s2,disk...	NO	Read Write
disk6	External	Backup+ BL	931.51 GB	GUID_partition_scheme				NO	
disk6s1	External		200.00 MB	EFI	EFI	msdos		NO	
disk6s2	External		931.19 GB	Apple_HFS	DEST-SUMURI	hfs		NO	Read Write

Mit DD bzw. ewfacquire

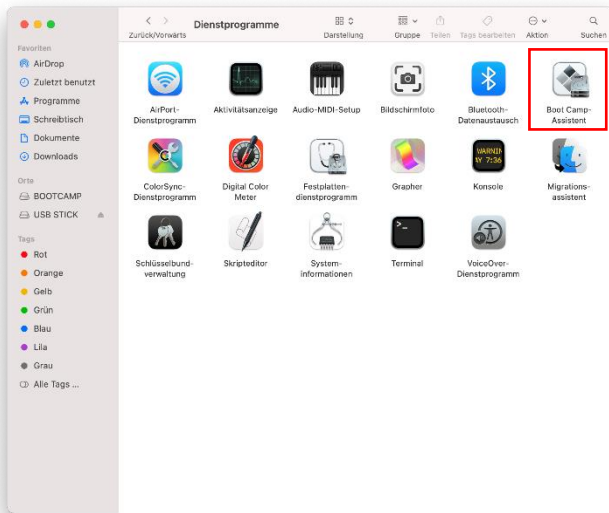
mit dem Tool dd können RAW Images von Disks erstellt werden, wie auch unter Linux mit „dd if=/dev/diskx of=/ImageDateiname.dmg (--bs=512)“. Bei installierter libewf kann auch das Tool ewfacquire genutzt werden. Hier wird der Befehl „ewfacquire /dev/diskx“ ausgeführt. Beide Befehle werden genutzt, um *.E01 Images anzulegen. Mit beiden Tools können von externen Datenträgern Datenträgerkopien erstellt werden, nicht von den internen verschlüsselten Systemen!

Mittels Bildschirmfotos

Oft wird es notwendig, zusätzliche Information bei live Datensicherungen zu erheben. Hier haben sich auch Bildschirmfotos bewährt. Bildschirmfotos vom kompletten Bildschirm können mit der Tastenkombination „Shift + Befehlstaste/Command (⌘) + 3“ aufgenommen werden. Der standard Ablageort ist der Desktop des Benutzers!

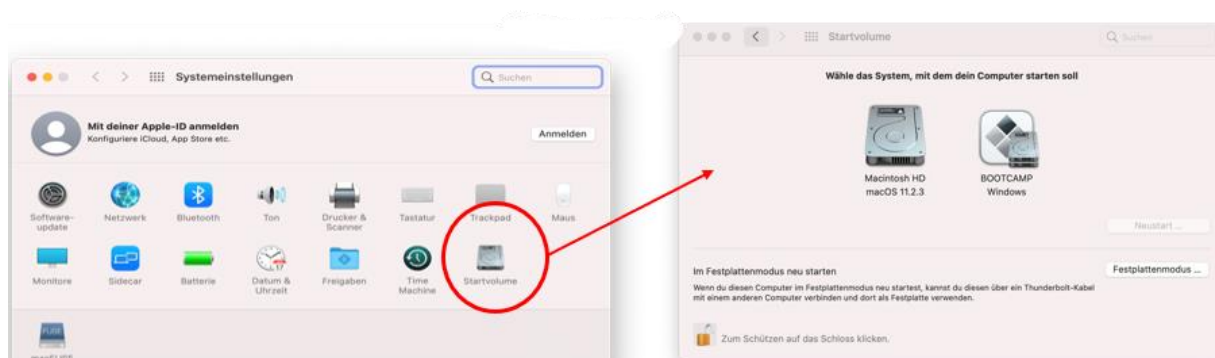
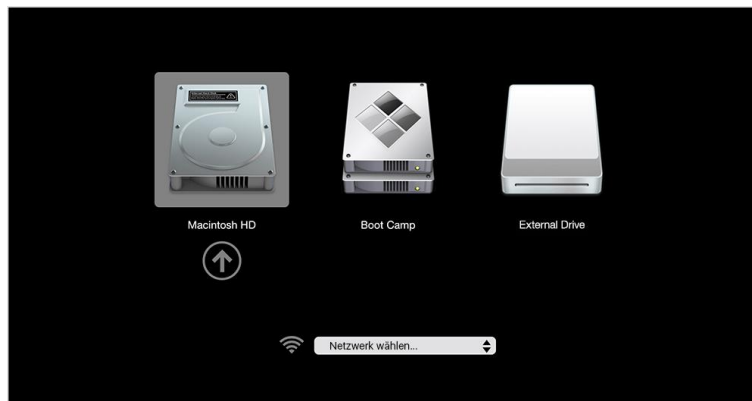
Bildschirmfotos von Bildschirmfenstern können mit der Tastenkombination „Shift + Befehlstaste/Command (⌘) + 5“ aufgenommen werden. Hierbei kann auch der Speicherort unter Optionen angegeben werden.

1.2.3 Bootcamp Besonderheiten und Parallels

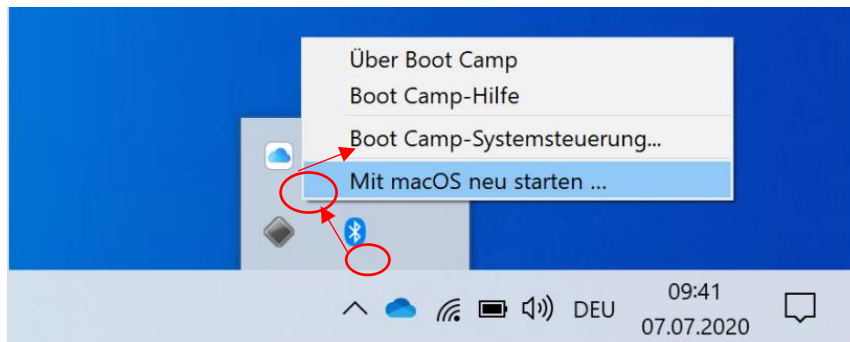


Auf macOS System mit Intel Architektur lässt sich per Einrichtungsdialo ein Windows System installieren. Die genutzte Technik heißt Bootcamp und erzeugt ein Volume im System für die Installation von Windows. Auf Silicon Mac Systemen mit M1 Architektur geht dies nicht mehr!

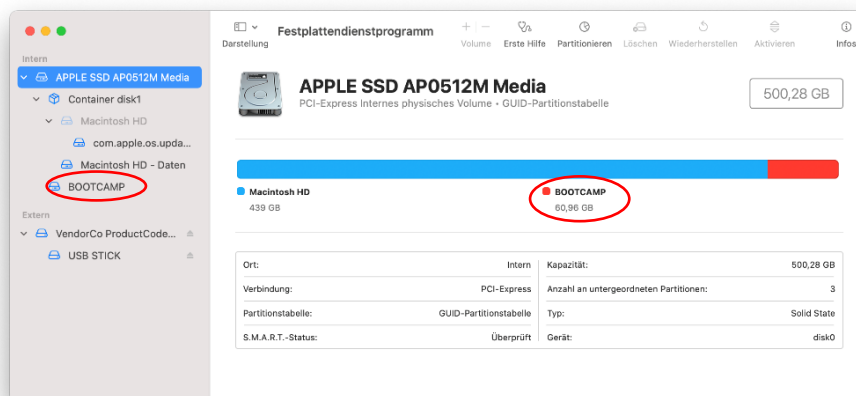
Das Startvolume kann entweder im Start Manager (Wahltaste/Option (⌘)) oder im macOS „Systemeinstellungen > Startvolume (schneller Spotlight > Startvolume)“ ausgewählt werden.



Ein Bootcamp Icon in der Windows Taskbar zeigt an, dass Bootcamp installiert ist und kann genutzt werden, um zurück zu macOS zu booten.



Das Bootcamp Volume gehört nicht zum APFS Container und ist eine separate Partition! Es kann wie jede normale Windows Partition untersucht werden.



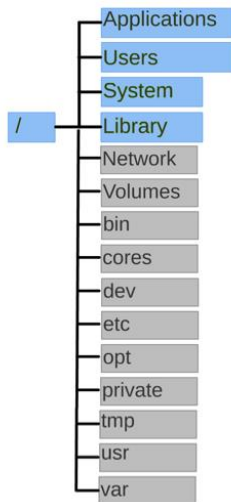
1.2.3.1 Parallels Desktop

Parallels Desktop bietet eine Virtualisierungslösung als Alternative zu Bootcamp und stellt eine bewährte Desktop-Virtualisierungssoftware dar, welche seit 15 Jahren im Einsatz ist. Es erfolgt eine Installation von Windows und Linux auf Intel- oder Apple M1-Mac (derzeit einziger Anbieter). Es gibt eine flüssige Reaktionszeit der Windows-Benutzeroberfläche und Videowiedergabe. Weiterhin ist es für macOS Monterey und Windows 11 optimiert. Parallels Desktop ist ein Konkurrent zum VMWare Fusion (kein Windows Support auf Silicon M1 mac's).

1.3 Speicherstrukturen und Datenformate

1.3.1 Mac FHS und Speicherstrukturen

1.3.1.1 Wichtige Verzeichnisstrukturen



Die Verzeichnisstruktur eines Mac OS X lässt den UNIX-Ursprung erkennen. Unterschiede gibt es jedoch bei den von Apple entwickelten Bestandteilen. Apple hält sich bei seiner Verzeichnisstruktur nicht streng an den Filesystem Hierarchy Standard (FHS). Der Standard ist aber noch erkennbar. Zudem sind viele Systemordner bei OSX versteckt (graue Verzeichnisse).

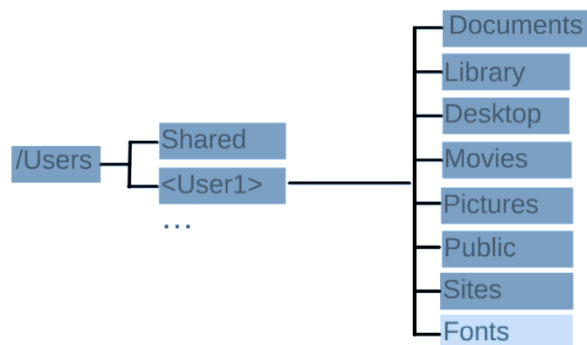
Das MAC OS X Betriebssystem teilt Daten ebenfalls in die Kategorien Systemdaten, Benutzerdaten und Softwaredaten auf.

Anwendungen befinden sich im Verzeichnis „Application“ im Root Verzeichnis.

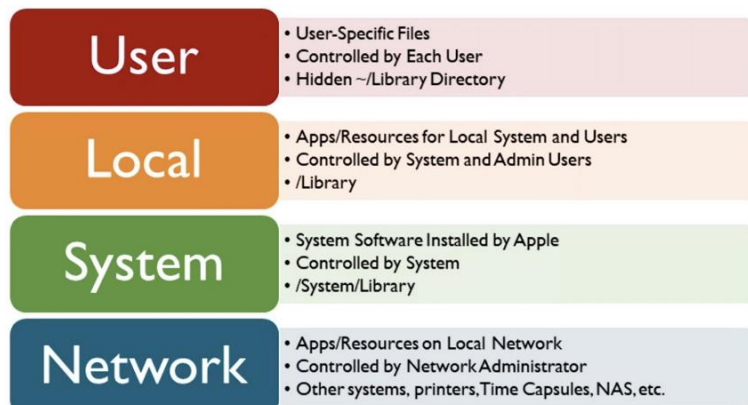
Systemdaten finden sich zum Einen im Verzeichnis „.../System/Library/“ und zum Anderen im „.../Library“ Verzeichnis im Root Verzeichnis selbst.

Die für jeden Benutzer gültigen Systemeinstellungen werden im Benutzerverzeichnis ebenfalls im Unterverzeichnis „.../Users/[Name]/ Library“ abgelegt.

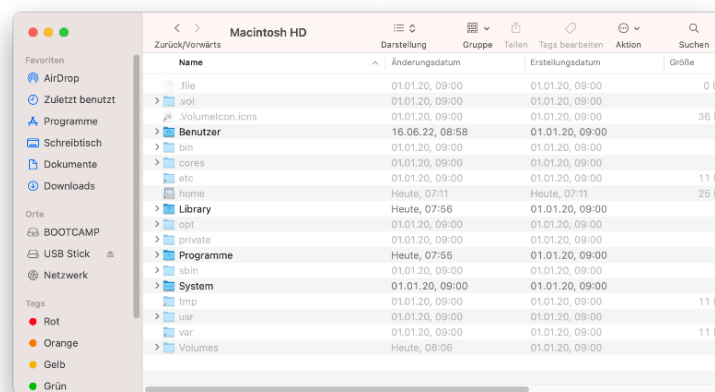
Benutzerdaten befinden sich im jeweiligen Unterverzeichnis des Benutzers, im Verzeichnis Users im Root Verzeichnis.



Folgende Übersicht zeigt zudem die von System und Netzwerk benutzten Bestandteile auf:



Die Vollansicht im Finder muss explizit aktiviert werden (Shift + Befehlstaste/Command (⌘) + .):



1.3.1.2 Zusätzliche Anwendungsdaten

Im Library Benutzerverzeichnis existiert ein Unterverzeichnis mit dem Namen „Application Support“. Hier befinden sich Benutzerdaten für die installierten Anwendungen. Dieses Verzeichnis kann mit dem auf Windows bekannten Anwendungsdatenverzeichnis gleichgesetzt werden.

1.3.2 Datenformate SQLite und Plist

1.3.2.1 Formate

OS X speichert jede Menge Informationen, die für den IT-Forensiker von Interesse sein können. Diese Informationen sind in einer ganzen Reihe verschiedener Formate abgelegt.

Einige sind leicht zu interpretieren, wie Plain Text, XML oder Datenbanken. Andere haben aber proprietäre Binärformate. Einige Dateiformate sind gut dokumentiert, andere leider gar nicht und mussten Reverse Engineered werden.

Für die meisten Binärformate bringt OSX aber eigene Viewer mit. Das ist ein Grund, dass sich Apple Rechner am besten auf einem Apple auswerten lassen.

Basic Security Modules (BSM) sind Binärdateien, welche die Kernel Logs speichern. Es gibt mehrere BSM-Dateien, die jeweils mehrere Token fester oder variabler Länge enthalten. Sie sind ähnlich zu den EVT, EVTX Dateien bei Windows.

Das Binary Apple System Log (ASL) ist das Standardformat für die Daemon Logs und besteht aus Binärdaten in doppelt verlinkten Listen.

Die Keychain ist ein binäres Datenbank-Format in welchem Passwörter und Zertifikate von Applikationen, Webseiten, Netzwerken und ähnlichem abgelegt werden.

OS X und die Applikationen legen ihre Konfiguration in Plist Dateien ab. Es gibt zwei Unterformate. Die Plists die XML enthalten und Binäre Plists (Bplist).

Userapplikationen wie Chrome, Skype, Firefox, etc., aber auch einige Caches von OSX selbst werden im SQLite Datenbankformat abgelegt.

1.3.2.2 macOS Library

Systemeinstellungen werden in *.plist Dateien den Property Listen gespeichert. Diese Dateien sind vom Aufbau her XML-Dateien mit Apple spezifischen Schlüsselpaaren. Es besteht jedoch die Möglichkeit diese Property Listen auch als binärcodierte Dateien abzulegen. Für die Auswertung ist es dann notwendig diese Binary Property List in eine XML basierte Datei umzuwandeln.

1.3.3 Plattformabhängige Speichermethode Plist

1.3.3.1 Plist Konvertierung

OSX bietet ein Tool, um zwischen beiden Formaten zu konvertieren. Mit dem Befehl „plutil -convert xml1 some_file.plist“ wird eine bplist in eine XML-plist konvertiert. Der Befehl „plutil -convert binary1 some_other_file.plist“ konvertiert eine XML-plist in eine bplist.

1.3.3.2 Aufbau des Plist Formates

Die Bplist Datei besteht zunächst aus vier Bestandteilen. Dazu zählt der 8 Byte Header, die var. Byte Objekttable, die var. Byte Offset Table und der 32 Byte Trailer. Diese Bestandteile werden im Folgenden nicht in der Reihenfolge beschrieben, in der sie auftreten, sondern in der Reihenfolge, die beim Lesen der Datei am sinnvollsten ist.

Header

Die bplist-Datei trägt im Header eine Signatur (0x62706C6973743030 = bplist), gefolgt von der Version des bplist Files (00,15,16,etc.). Die Untersuchung von Bplist Dateien setzt in der Regel voraus, dass es sich um eine Bplist der Version 00 handelt, da andere Versionen von Apple bisher nicht dokumentiert sind.

Trailer

Der Trailer besteht im Wesentlichen nur aus drei für uns relevanten Teilen. Zum einen die Offsetgröße in Byte, welche uns verrät, wie groß ein Offset in der Offset Tabelle ist. Weiter ist hier auch die Anzahl aller gespeicherten Objekte vermerkt. Abschließend ist der Offset der Offset Tabelle ebenfalls noch enthalten.

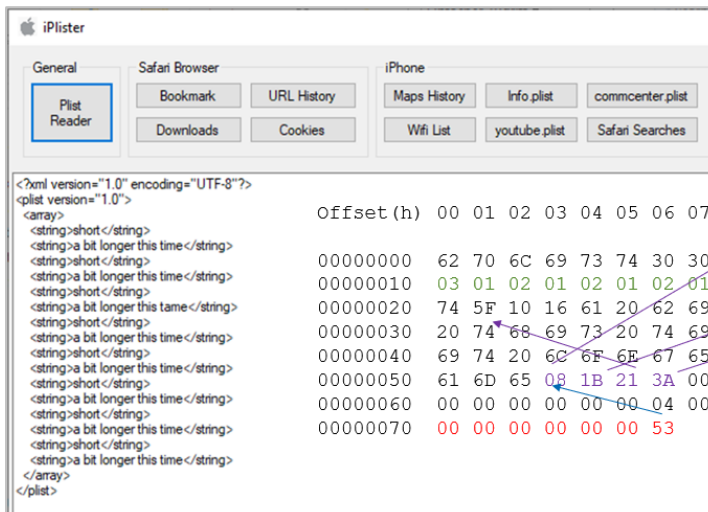
Offset	Länge	Datenstruktur	Beschreibung
0	5	unbenutzt	unbenutzt
6	1	8-bit unsigned integer	Offsetgröße in Byte
7	1	8-bit unsigned integer	Referenzgröße in Dictionaries
8	8	64-bit unsigned big endian integer	Anzahl der gespeicherten Objekte
16	8	64-bit unsigned big endian integer	Top Level Object Index
24	8	64-bit unsigned big endian integer	Offset der Offset Tabelle
32			

Offset-Tabelle

Die Offset-Tabelle ist der dritte Abschnitt in der Datei. Der Startoffset in der Datei wird im Trailer angegeben („Offset der Offset-Tabelle“). Die Länge kann durch Multiplizieren der Trailer-Felder „Anzahl der gespeicherten Objekte“ mit „Offsetgröße in Byte“ ermittelt werden. Die Offsettabelle enthält ein Array von Offsets mit einem Index von Null (dh der erste Eintrag wird als Eintrag 0 bezeichnet, der zweite als Eintrag 1 usw.), die auf Objekte in der Objekttable verweisen.

Die folgende Abbildung zeigt ein Beispiel einer Offset-Tabelle mit den dazugehörigen referenzierten Objekten.

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	62	70	6C	69	73	74	30	30	AF	10	10	01	02	01	02	01	bplist00 ⁻
00000010	03	01	02	01	02	01	02	01	02	01	02	55	73	68	6F	72Ushor
00000020	74	5F	10	16	61	20	62	69	74	20	6C	6F	6E	67	65	72	t...a bit longer
00000030	20	74	68	69	73	20	74	69	6D	65	5F	10	16	61	20	62	this time...a b
00000040	69	74	20	6C	6F	6E	67	65	72	20	74	68	69	73	20	74	it longer this t
00000050	61	6D	65	08	1B	21	3A	00	00	00	00	00	00	01	01	00	ame...!:.....
00000060	00	00	00	00	00	00	04	00	00	00	00	00	00	00	00	00
00000070	00	00	00	00	00	00	53									S



Objekttabelle

Die Objekttabelle ist der zweite Abschnitt in der Datei, der direkt nach dem Header bei Offset 8 beginnt und bis zum Beginn der Offset-Tabelle fortgesetzt wird. Es enthält die binären Darstellungen jedes „Objekts“ in der Plist.

Jedes Objekt besteht aus einem Typ-Deskriptor-Byte, optional gefolgt von einer Länge (abhängig vom Datentyp kann die Länge im Datentyp-Byte impliziert sein), gefolgt von den Daten selbst.

Um die Länge der Objekte festzustellen, muss zunächst der Typ erkannt werden. Dieser wird in den meisten Fällen durch die ersten 4 Bit bestimmt. Die Länge ergibt sich dann durch die nächsten 4 Bit. Ist die Länge kleiner als 15 Byte so reichen diese 4 Bit aus, um die Länge anzugeben. Ist das Objekt größer, so werden die 4 Bit alle mit 1 belegt. Das bedeutet, dass die nächsten Bytes für die Größe ausgewertet werden müssen.

Typ	Binär	Größe	Inhalt
null	0000 0000		
bool	0000 1000		FALSE
bool	0000 1001		TRUE
fill	0000 1111		Füll Byte
int	0001 nnnn	...	2 ⁿⁿⁿⁿ Big-Endian Bytes
real	0010 nnnn	...	2 ⁿⁿⁿⁿ Big-Endian Bytes
date	0011 0011	...	8 Byte Float (Big-Endian)
data	0100 nnnn	[int] ...	nnnn Bytes bis '1111', danach int Typ und int Count gefolgt von Inhalt
string	0101 nnnn	[int] ...	ASCII String, nnnn Chars bis '1111', danach int Typ und int Count gefolgt von Inhalt
string	0110 nnnn	[int] ...	Unicode String, nnnn Chars bis '1111', danach int Typ und int Count gefolgt von Inhalt

Typ	Binär	Größe	Inhalt
	0111 xxxx		reserviert
uid	1000 nnnn	...	nnnn+1 Bytes
	1001 xxxx		reserviert
array	1010 nnnn	[int] objref*	nnnn Einträge bis '1111', danach int Typ und int Count
	1011 xxxx		reserviert
set	1100 nnnn	[int] objref*	nnnn Einträge bis '1111', danach int Typ und int Count
dict	1101 nnnn	[int] keyref	nnnn Einträge bis '1111', danach int Typ und int Count
	1110 xxxx		reserviert
	1111 xxxx		reserviert
	0111 xxxx		reserviert

Beispiel

Im Folgenden wird ein Beispiel einer Plist dargestellt.

```

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01 bplist00.....
00000010 03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72 .....Ushor
00000020 74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72 t...a bit longer
00000030 20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62 this time...a b
00000040 69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74 it longer this t
00000050 61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00 ame...!:.....
00000060 00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 .....
00000070 00 00 00 00 00 00 53 .....S
  
```

AF = 1010 1111

nnnn = Anzahl im nachfolgenden Byte

array	1010 nnnn	[int] objref*	nnnn Einträge, bis '1111', danach int Typ und int Count
-------	-----------	---------------	--

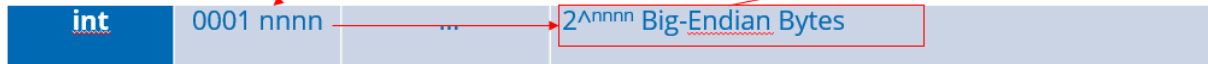
```

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01 bplist00^.....
00000010 03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72 .....Ushor
00000020 74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72 t...a bit longer
00000030 20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62 this time...a b
00000040 69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74 it longer this t
00000050 61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00 ame...!:.....
00000060 00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 .....
00000070 00 00 00 00 00 00 53 .....S

```

10 Type = 0001 0000

2^0 = 1 Byte



```

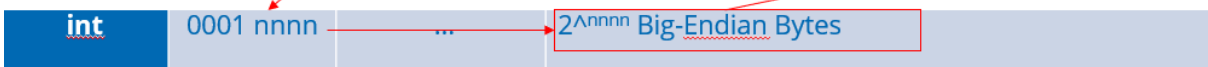
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01 bplist00^.....
00000010 03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72 .....Ushor
00000020 74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72 t...a bit longer
00000030 20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62 this time...a b
00000040 69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74 it longer this t
00000050 61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00 ame...!:.....
00000060 00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 .....
00000070 00 00 00 00 00 00 53 .....S

```

10 Type = 0001 0000

10 Count = 16 Dezimal

2^0 = 1 Byte



```

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01 bplist00^.....
00000010 03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72 .....Ushor
00000020 74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72 t...a bit longer
00000030 20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62 this time...a b
00000040 69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74 it longer this t
00000050 61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00 ame...!:.....
00000060 00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 .....
00000070 00 00 00 00 00 00 53 .....S

```

10 Count = 16 Dezimal

16 Array Member – Nummer bezieht sich auf Position innerhalb der Plist Offsets (beginnend bei 0):
 01 02 01 02 01 03 01 02 01 02 01 02 01 02 01 02

```

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000 62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00^.....
00000010 03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  .....Ushor
00000020 74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t...a bit longer
00000030 20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62  this time...a b
00000040 69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050 61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame...!:.....
00000060 00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  .....
00000070 00 00 00 00 00 00 53

```

```

<plist version="1.0">
<array>
<string>short</string>
<string>a bit longer this time</string>
<string>short</string>
<string>a bit longer this time</string>
<string>short</string>
<string>a bit longer this time</string>
<string>short</string>
<string>a bit longer this time</string>
<string>short</string>
<string>a bit longer this time</string>
<string>short</string>
<string>a bit longer this time</string>
<string>short</string>
<string>a bit longer this time</string>
<string>short</string>
<string>a bit longer this time</string>
</array>
</plist>

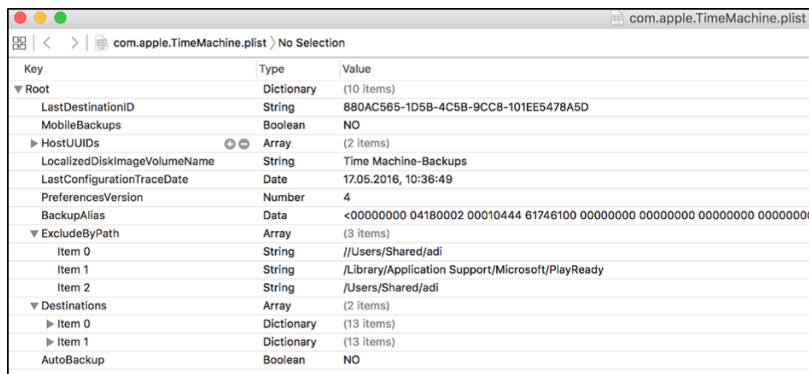
```

16 Array Member – Nummer bezieht sich auf Position innerhalb der Plist Offsets (beginnend bei 0):

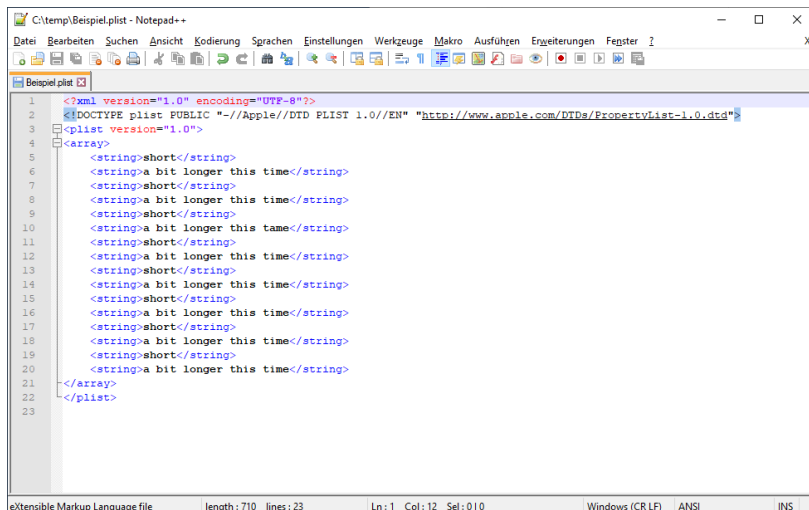
01 02 01 02 01 03 01 02 01 02 01 02 01 02 01 02

1.3.3.3 Interpretieren und Auswerten von *.plist Dateien

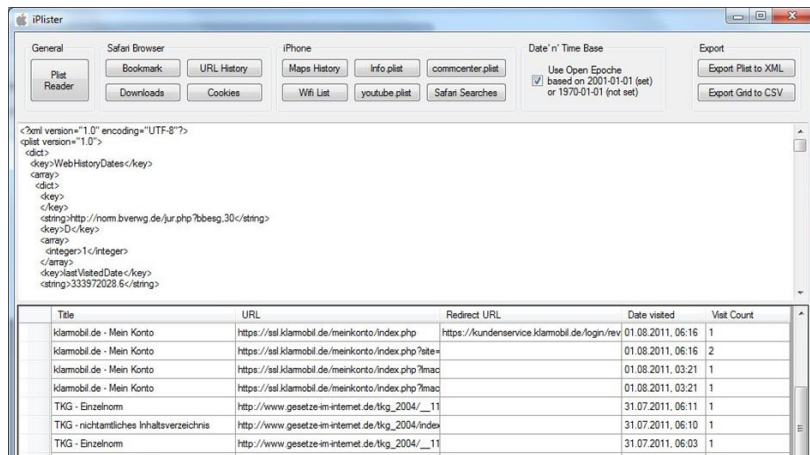
Neben plutil zum Konvertieren von Property Lists liefert Apple einen kostenlosen Property List Editor zusammen mit der XCode Entwicklungsumgebung aus.



Daneben gibt es unzählige Tools für OSX, um Property Lists zu bearbeiten oder anzuzeigen. Auch für Windows sind Programme verfügbar, wie etwa ein Bplist Plugin für Notepad++.



Oder die freie Software iPlister aus dem Forensik Bereich von Eyewitness Forensic Software (www.eyewitnessforensic.com).



1.3.4 Plattformunabhängige Speichermethode SQLite

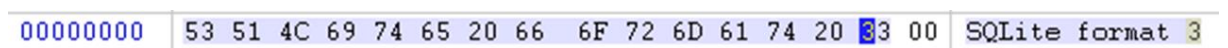
1.3.4.1 SQLite DB (relationale SQL-Datenbank)

SQLite DB ist das bekannteste und am weitesten verbreitete Datenbankformat für kleine und mittlere Desktop- sowie mobile Anwendungen. Viele native Apps in Android und auch iOS nutzen SQLite Datenbanken. Sie geriet bereits früh im Focus forensischer Untersuchungen, vor allem weil hier gelöschte Datenbankeintragen wiederherstellbar sind.

Bei einer relationalen Datenbank können unterschiedliche Datenfelder in einem Datensatz zusammengefasst und hinzugefügt, geändert oder gelöscht werden. Die Abfragen erfolgen mit SQL. Weiterhin sind Relationen zwischen Datenfeldern und Datensätzen zulässig. Sie sind nicht parallel verarbeitbar.

Aufgrund der vielen Betriebssystemportierungen ist SQLite DB weit verbreitet. Bekannte Anwendungen die SQLite Datenbanken nutzen sind Mozilla Firefox, Google Chrome, ICQ bis 7, Skype und native iOS Apps wie das Adressbuch sowie in Android die Kontakte.

Im Folgenden wird der Aufbau einer SQLite Datenbank beschrieben. Die SQLITE 3 Header-Signatur ist folgende:

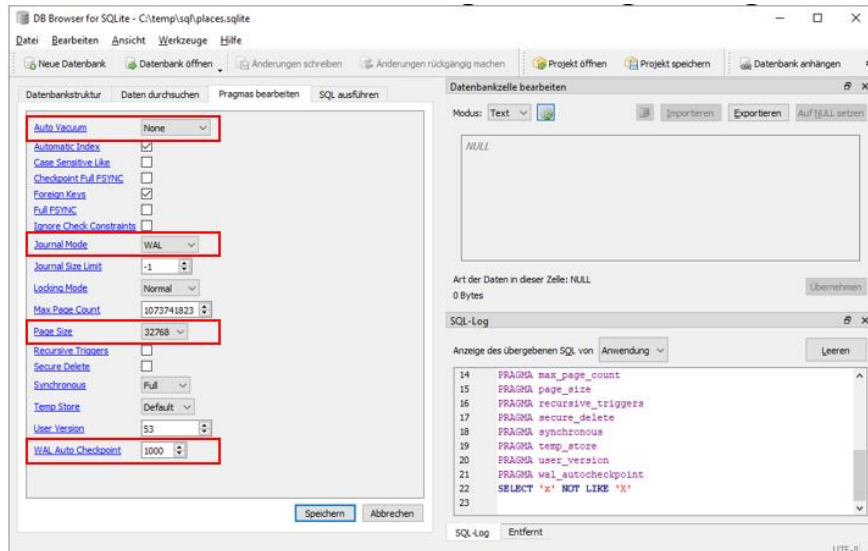


Weiterhin bestehen SQLite DBs aus Pages mit einer vordefinierten Größe. Diese Pages werden mit einzelnen Datensätzen (Records) beschrieben. Außerdem sind SQLite DBs B-Tree basiert. Sie können mit einer Journal-Funktion versehen werden (Rollback oder Write Ahead Log – WAL).

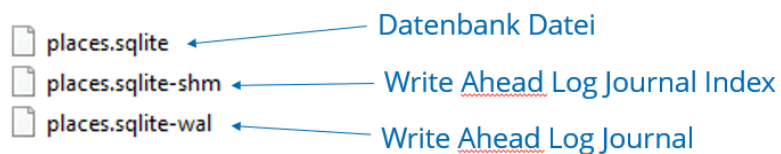
Die SQLite Datenbank besitzt verschiedene Funktionen. Änderungen an Datensätzen entstehen durch das Ändern von Pages, wobei neue Pages mit geänderten Daten entstehen können. Alte Pages werden entweder freigegeben oder reorganisiert. Weiterhin sind automatische Comactionen möglich (Auto

Vacuum), indem freie Pages ans Ende gestellt und die Datenbank automatisch gekürzt wird. Ohne das Auto Vacuum werden freie Pages in der Free Page List verwaltet und wiederverwendet. Eine Compaction (Vacuum) kann auch manuell erzwungen werden.

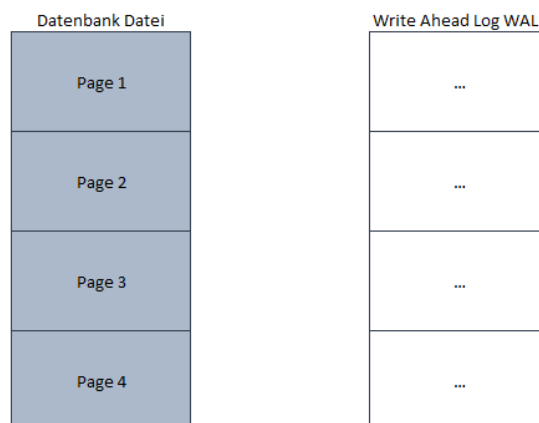
Die Einstellungen der Datenbank können über Pragma Abfragen festgestellt werden:



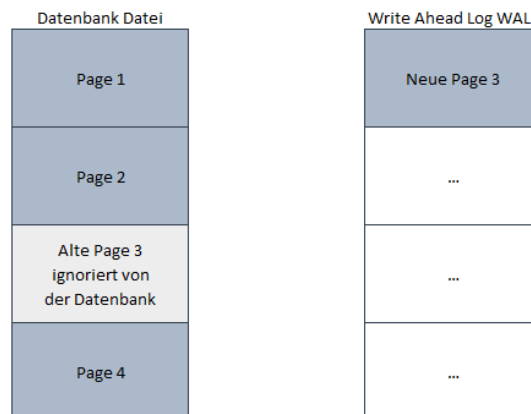
Mit Rollback oder WAL werden Page Änderungen in zusätzlichen Dateien außerhalb der Datenbank gespeichert und erst nach Transaktionsbestätigung eingearbeitet. Die folgende Abbildung zeigt den Dateisystem Aufbau mit einer WAL Datei.



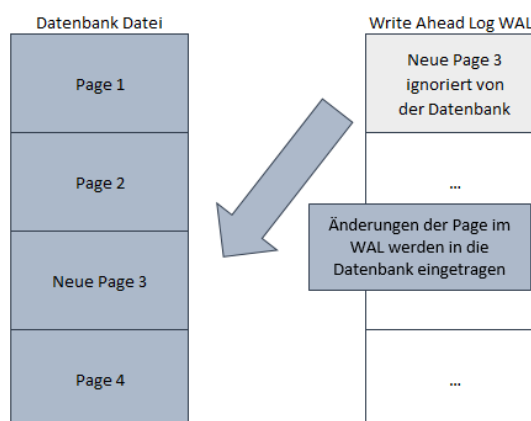
Die nächste Abbildung zeigt den Initialisierungszustand der Datenbank.



Nun wird die Page drei geändert und in das WAL eingetragen.



Die Page 3 wird geändert und vom WAL in die Datenbank geschrieben.



Je nach eingestellter WAL Checkpoint Größe kann eine Eintragung der WAL Daten, wie auch die Löschungen erst spät erfolgen. Eintragungen aus dem WAL werden bei jedem Start und Ende der SQLite Datenbank Engine durchgeführt. Bei einem Crash/Kill, sind diese Änderungen womöglich noch nicht durchgeführt worden. Bei forensischen Untersuchungen spielen daher auch die WAL Dateien eine entscheidende Rolle.

1.3.5 SQLite – Wiederherstellung gelöschter Records

1.3.5.1 SQLITE 3 Datenbanken Header

SQLITE3 DBs besitzen einen Header mit den wichtigsten Konfigurationsparametern als Big Endian Integer. Das Speichern der Einträge erfolgt ähnlich einem Dateisystem in Blöcken/Pages mit variabel einstellbarer Größe. Die Page Größe variiert zwischen 512 und 32768 Bytes und befindet sich an Offset 16 der Datenbankdatei als 2 Byte BE Integer:

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
00000000	53	51	4C	69	74	65	20	66	6F	72	6D	61	74	20	33	00	SQLite format 3
00000016	10	00	01	01	00	40	20	20	00	00	01	6C	00	00	03	BA	@ 1 2
00000032	00	00	00	2B	00	00	03	7D	00	00	00	1A	00	00	00	01	+ }
00000048	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0A	
00000064	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	01	6B		k
00000096	00	2D	E2	19	05	00	00	00	00	00	00	00	00	00	1C		-â û
00000112	0F	FB	00	00	00	00	00	00	00	00	00	00	00	00	00		û

Daten-Dolmetscher

8 Bit (±): 0

16 Bit (±): 4096

32 Bit (±): 16781312

Am Offset 52 (0x34) wird festgelegt, ob es sich um eine AutoVacuum DB oder um eine mit einer wachsenden Größe handelt. AutoVacuum Datenbanken reorganisieren sich beim Löschen von Daten/Records neu. In AutoVacuum Datenbanken lassen sich gelöschte Records nicht wiederherstellen. AutoVacuum Datenbanken beinhalten in Offset 52 (0x34) vier Non-Zero Byte. Somit ist eine DB nur wiederherstellbar, wenn Offset 52 vier Zero Byte enthält.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
00000000	53	51	4C	69	74	65	20	66	6F	72	6D	61	74	20	33	00	SQLite format 3
00000016	10	00	01	01	00	40	20	20	00	00	01	6C	00	00	03	BA	@ 1 2
00000032	00	00	00	2B	00	00	03	7D	00	00	00	1A	00	00	00	01	+ }
00000048	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	0A	

1.3.5.2 Vom Header zur FreepageList

SQLITE 3 Datenbanken speichern gelöschte Records in Free Pages. Offset 32 (0x20) beinhaltet die erste Sprungadresse der Freelist Page. Sprungadresse = (4 Byte BE in Offset 32 – 1) * Pagesize

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	53	51	4C	69	74	65	20	66	6F	72	6D	61	74	20	33	00	SQLite format 3
00000010	10	00	01	01	00	40	20	20	00	00	00	0F	00	00	00	00	@
00000020	00	00	00	4A	00	00	00	22	00	00	00	19	00	00	00	01	J "

Sprungadresse hex = (0x4A – 1) * 0x1000 = 0x49000

Sprungadresse dec = (74 – 1) * 4096 = 299008

1.3.5.3 Die FreepageList

FreepageLists speichern freie Pages für ein Neubeschreiben, in denen aber auch gelöschte Daten enthalten sein können. Die ersten 4 Byte BE verweisen auf eine weitere Free List Page oder sind andernfalls Zero. Die darauffolgenden 4 Byte stellen die Anzahl der folgenden Listeneinträge dar.

00049000	00 00 00 00	00 00 00 21	00 00 00 55	00 00 00 4C	!	U	I
00049010	00 00 00 4B	00 00 00 45	00 00 00 44	00 00 00 40	K	E	D
00049020	00 00 00 3F	00 00 00 2B	00 00 00 2C	00 00 00 35	?	+	,
00049030	00 00 00 36	00 00 00 37	00 00 00 39	00 00 00 3A	6	7	9
00049040	00 00 00 49	00 00 00 48	00 00 00 3D	00 00 00 51	I	H	=
00049050	00 00 00 30	00 00 00 2E	00 00 00 43	00 00 00 47	0	.	C
00049060	00 00 00 46	00 00 00 2D	00 00 00 31	00 00 00 4F	F	-	1
00049070	00 00 00 4E	00 00 00 34	00 00 00 33	00 00 00 3C	N	4	3
00049080	00 00 00 50	00 00 00 42	00 00 00 2F	00 00 00 00	P	B	/

Freepages werden als 4 Byte Eintragungen angezeigt, die um 1 subtrahiert mit der Pagesize multipliziert den Offset der gelöschten freien Page vom Beginn an ergeben.

00049000	00 00 00 00	00 00 00 21	00 00 00 55	00 00 00 4C	!	U	I
00049010	00 00 00 4B	00 00 00 45	00 00 00 44	00 00 00 40	K	E	D
00049020	00 00 00 3F	00 00 00 2B	00 00 00 2C	00 00 00 35	?	+	,
00049030	00 00 00 36	00 00 00 37	00 00 00 39	00 00 00 3A	6	7	9
00049040	00 00 00 49	00 00 00 48	00 00 00 3D	00 00 00 51	I	H	=
00049050	00 00 00 30	00 00 00 2E	00 00 00 43	00 00 00 47	0	.	C
00049060	00 00 00 46	00 00 00 2D	00 00 00 31	00 00 00 4F	F	-	1
00049070	00 00 00 4E	00 00 00 34	00 00 00 33	00 00 00 3C	N	4	3
00049080	00 00 00 50	00 00 00 42	00 00 00 2F	00 00 00 00	P	B	/

↑

Offset
= (0x50-1)*0x1000
= 0x4F000

↑

Offset
= (0x43-1)*0x1000
= 0x42000

1.3.5.4 Freepage einer FFX3 DB

Die folgende Abbildung zeigt eine der gelöschten Freepages mit gelöschten Record Einträgen einer Mozilla Firefox moz_places Tabelle:

Offset= (0x50-1)*0x1000= 0x4F000

0004F000	0D 04 C4 00 10 00 4E 00	0E 9C 0E 1B 0D 70 0C A6	Ä	N		p	
0004F010	0C 0D 0B 54 0A 6E 09 D5	09 62 06 98 06 29 04 27	T	n	Ö	b	
0004F020	03 79 02 EB 01 AB 00 4E	00 4E 00 4E 00 4E 00 4E	y	ë	<<	N	N
0004F030	00 4E 00 4E 00 4E 00 4E	00 00 00 00 00 00 00 00	N	N	N	N	N
0004F040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 82	59				
0004F050	84 7C 0C 00 84 3B 43 29	01 01 01 01 01 01 06 68	74				:C)
0004F060	74 70 3A 2F 2F 77 77 77	2E 67 6F 6F 67 6C 65 2E	ht				
0004F070	64 65 2F 73 65 61 72 63	68 3F 73 63 6C 69 65 6E	tp://www.google.				
0004F080	74 3D 70 73 79 26 68 6C	3D 64 65 26 63 6C 69 65	de/search?scli				
0004F090	6E 74 3D 66 69 72 65 66	6F 78 2D 61 26 68 73 3D	t=psy&hl=de&clie				
0004F0A0	68 4F 72 26 72 6C 73 3D	6F 72 67 2E 6D 6F 7A 69	nt=firefox-a&hs=				
0004F0B0	6C 6C 61 3A 64 65 25 33	41 6F 66 66 69 63 69 61	hOr&rls=org.moz				
			lla:de%3Aofficia				

Offset= (0x43-1)*0x1000= 0x42000

```

00042000 0D 00 00 00 19 00 51 00 0F 4C 0E 9F 0D E2 0D 44 Q L | ä D
00042010 0C 7E 0C 32 0B E2 0B 78 0A EA 0A 46 09 B7 09 08 ~ 2 ä x ê F
00042020 08 3C 07 B4 07 1E 05 DB 04 95 04 28 03 BC 02 E6 < ' Û | ( % æ
00042030 02 77 01 C3 01 52 00 E0 00 51 00 00 00 00 00 00 w Å R à Q
00042040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00042050 00 81 0B 83 7B 0C 00 47 81 0B 39 01 01 01 01 01 |{ G 9
00042060 06 68 74 74 70 3A 2F 2F 77 77 77 2E 6A 65 6E 73 http://www.jens
00042070 2D 74 72 61 69 6E 69 6E 67 2E 63 6F 6D 2F 4A 65 -training.com/Je
00042080 6E 73 20 54 72 61 69 6E 69 6E 67 20 4C 74 64 20 ns Training Ltd
00042090 2D 20 43 6F 6D 70 75 74 65 72 20 46 6F 72 65 6E - Computer Foren
000420A0 73 69 63 73 20 61 6E 64 20 49 54 20 53 65 63 75 sics and IT Secu
000420B0 72 69 74 79 20 54 72 61 69 6E 69 6E 67 6D 6F 63 rity Trainingmoc
000420C0 2E 67 6E 69 6E 69 61 72 74 2D 73 6E 65 6A 2E 77 .gniniart-snej.w
000420D0 77 77 2E 01 00 00 2D 62 00 04 A4 02 56 42 3A F9 ww. -b x VB:ù
000420E0 6F 83 7A 0B 00 47 55 39 01 01 01 00 02 06 68 74 oiz GU9 ht
000420F0 74 70 3A 2F 2F 77 77 77 2E 6A 65 6E 73 6B 69 72 tp://www.jenskir
00042100 73 63 68 6E 65 72 2E 63 6F 6D 2F 52 65 64 69 72 schner.com/Redir
00042110 65 63 74 69 6E 67 20 74 6F 20 77 77 77 2E 6A 65 ecting to www.je
00042120 6E 73 2D 74 72 61 69 6E 69 6E 67 2E 63 6F 6D 6D ns-training.com
00042130 6F 63 2E 72 65 6E 68 63 73 72 69 6B 73 6E 65 6A cc.renhcsrikenej
00042140 2E 77 77 77 2E 01 00 01 07 9E 00 04 A4 02 56 42 .www. | x VB

```

1.3.5.5 Freepage Aufbau

Die Freepages sind als B-Baum aufgebaut. Dieser besteht aus einem Header, einem Cell Pointer Array und den Cell Einträgen.

```

00042000 0D 00 00 00 19 00 51 00 0F 4C 0E 9F 0D E2 0D 44 Q L | ä D
00042010 0C 7E 0C 32 0B E2 0B 78 0A EA 0A 46 09 B7 09 08 ~ 2 ä x ê F
00042020 08 3C 07 B4 07 1E 05 DB 04 95 04 28 03 BC 02 E6 < ' Û | ( % æ
00042030 02 77 01 C3 01 52 00 E0 00 51 00 00 00 00 00 00 w Å R à Q
00042040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00042050 00 81 0B 83 7B 0C 00 47 81 0B 39 01 01 01 01 01 |{ G 9
00042060 06 68 74 74 70 3A 2F 2F 77 77 77 2E 6A 65 6E 73 http://www.jens
00042070 2D 74 72 61 69 6E 69 6E 67 2E 63 6F 6D 2F 4A 65 -training.com/Je
00042080 6E 73 20 54 72 61 69 6E 69 6E 67 20 4C 74 64 20 ns Training Ltd
00042090 2D 20 43 6F 6D 70 75 74 65 72 20 46 6F 72 65 6E - Computer Foren
000420A0 73 69 63 73 20 61 6E 64 20 49 54 20 53 65 63 75 sics and IT Secu
000420B0 72 69 74 79 20 54 72 61 69 6E 69 6E 67 6D 6F 63 rity Trainingmoc
000420C0 2E 67 6E 69 6E 69 61 72 74 2D 73 6E 65 6A 2E 77 .gniniart-snej.w

```

- Offset 0: 0x0A Indexpage und 0x0D Datenpage
- Offset 3+4 i:Anzahl folgender Cellpointer
- ab Offset 8 je 2 Byte Cellpointer, Offset der Cell zum Page Anfang

1.3.5.6 Free Page Records

Die folgende Abbildung zeigt einen Free Page Record.

Length of Payload (Varint)	Row ID (Varint)	Payload Header	Payload
----------------------------	-----------------	----------------	---------

Row ID	Hex Data	Text
00042050	00 81 0B 83 7B 0C 00 47 81 0B 39 01 01 01 01 01	{ G 9
00042060	06 68 74 74 70 3A 2F 2F 77 77 77 2E 6A 65 6E 73	http://www.jens
00042070	2D 74 72 61 69 6E 69 6E 67 2E 63 6F 6D 2F 4A 65	-training.com/Je
00042080	6E 73 20 54 72 61 69 6E 69 6E 67 20 4C 74 64 20	ns Training Ltd
00042090	2D 20 43 6F 6D 70 75 74 65 72 20 46 6F 72 65 6E	- Computer Foren
000420A0	73 69 63 73 20 61 6E 64 20 49 54 20 53 65 63 75	sics and IT Secu
000420B0	72 69 74 79 20 54 72 61 69 6E 69 6E 67 6D 6F 63	rity Trainingmoc
000420C0	2E 67 6E 69 6E 69 61 72 74 2D 73 6E 65 6A 2E 77	.gniniart-snej.w
000420D0	77 77 2E 01 00 00 2D 62 00 04 A4 02 56 42 3A F9	ww. -b VB:ù
000420E0	6F 83 7A 0B 00 47 55 39 01 01 01 00 02 06 68 74	oiz GU9 ht
000420F0	74 70 3A 2F 2F 77 77 77 2E 6A 65 6E 73 6B 69 72	tp://www.jenskir
00042100	73 63 68 6E 65 72 2E 63 6F 6D 2F 52 65 64 69 72	schner.com/Redir
00042110	65 63 74 69 6E 67 20 74 6F 20 77 77 77 2E 6A 65	ecting to ww.je
00042120	6E 73 2D 74 72 61 69 6E 69 6E 67 2E 63 6F 6D 6D	ns-training.comm
00042130	6F 63 2E 72 65 6E 68 63 73 72 69 6B 73 6E 65 6A	oc.renhcsriksnej
00042140	2E 77 77 77 2E 01 00 01 07 9E 00 04 A4 02 56 42	.www. VB

Ein Payload Header ist nach dem folgenden Schema aufgebaut:

Serial Type	Content Size	Meaning
0	0	NULL
1	1	8-bit twos-complement integer
2	2	Big-endian 16-bit twos-complement integer
3	3	Big-endian 24-bit twos-complement integer
4	4	Big-endian 32-bit twos-complement integer
5	6	Big-endian 48-bit twos-complement integer
6	8	Big-endian 64-bit twos-complement integer
7	8	Big-endian IEEE 754-2008 64-bit floating point number
8	0	Integer constant 0. Only available for schema format 4 and higher.
9	0	Integer constant 1. Only available for schema format 4 and higher.
10,11		Not used. Reserved for expansion.
N≥12 and even	(N-12)/2	A BLOB that is (N-12)/2 bytes in length
N≥13 and odd	(N-13)/2	A string in the database encoding and (N-13)/2 bytes in length. The nul terminator is omitted.

00 47 81 0B 39 01 01 01 01 01 06
 0x00 NULL (ROWID - Primary Key)
 0x47 String Länge 29 Bytes (URL)
 0x810B String Länge 63 Bytes (Titel)
 0x39 String Länge 22 Bytes (reverse URL)
 0x01 Int16
 0x01 Int16
 0x01 Int16
 0x01 Int16
 0x01 Int16
 0x06 Int64 (Datum Unix Epoche)

Die Länge von gespeicherten Informationen wird entweder durch den Typ vorgegeben oder als VarInt angegeben.

Im Folgenden wird der damit entschlüsselte Dateninhalt der FFX3 moz_places Zeile dargestellt.

|00 47 81 0B 39 01 01 01 01 01 06|

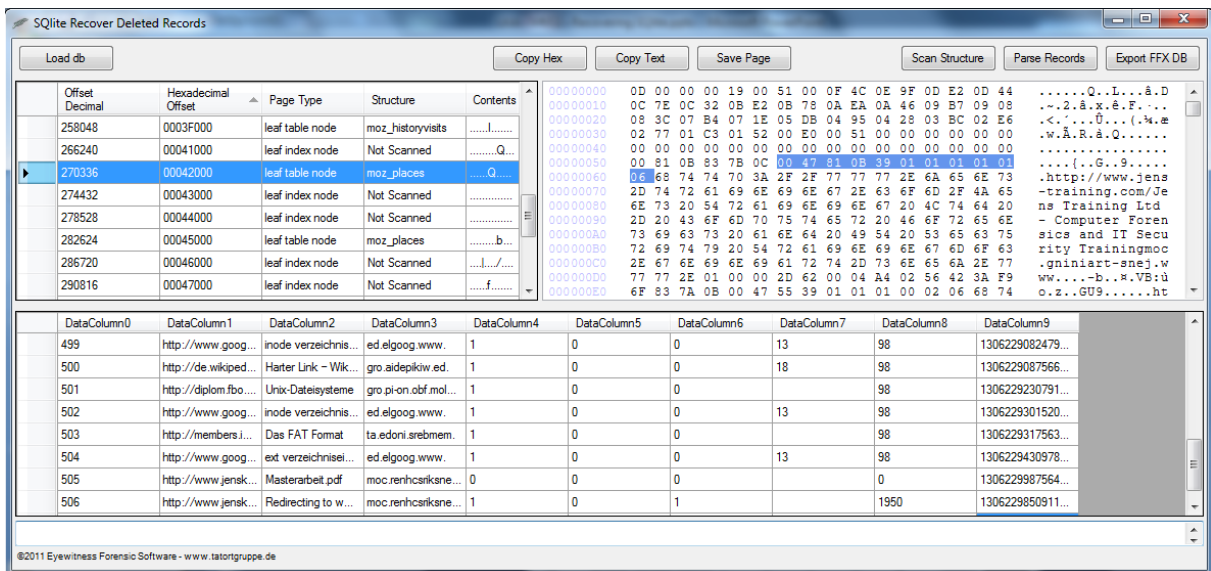
- 0x00 NULL (ROWID – Primary Key)
- 0x47 String Länge 29 Bytes (URL)
- 0x810B String Länge 63 Bytes (Titel)
- 0x39 String Länge 22 Bytes (reverse URL)
- 0x01 Int8
- 0x01 Int8
- 0x01 Int8
- 0x01 Int8
- 0x01 Int8
- 0x06 Int64 (Datum Unix Epoche)

```

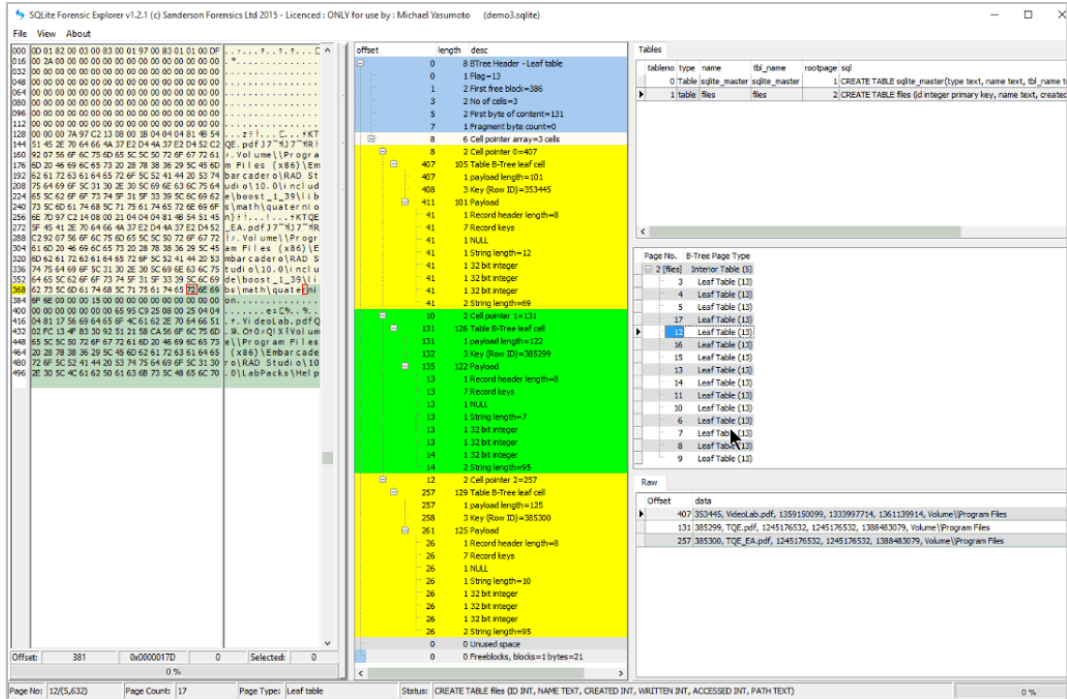
http://www.jens
-training.com/Je
ns Training Ltd
- Computer Foren
sics and IT Secu
rity Trainingmoc
.gniniart-snej.w
ww. -b VB:ù
o|z GU9 ht
  
```

1.3.5.7 Tools

Es existieren derzeit einige Forensic Tools am Markt, die in der Lage sind gelöschte Datenbank Records wiederherzustellen:



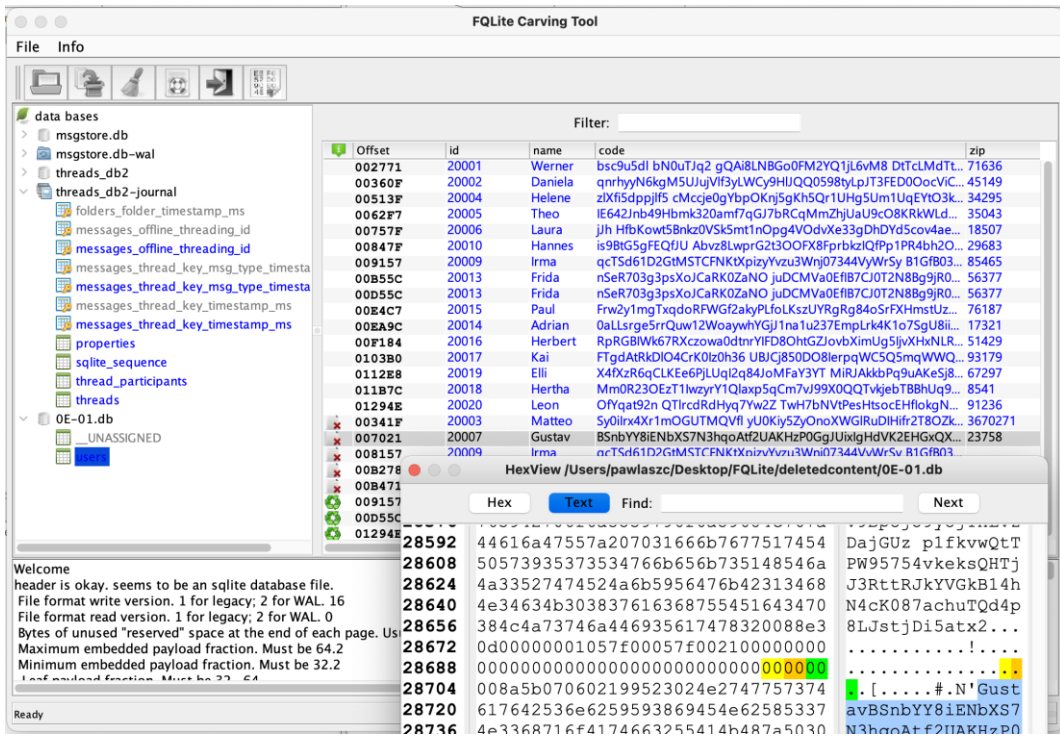
Eines der umfangreichsten Toolkits ist jedoch des SQLite Forensik Toolkit von Sanderson Forensics:



Auch ein Python Script existiert, welches gelöschte Records parsen kann: <https://github.com/mdegrazia/SQLite-Deleted-Records-Parser>.

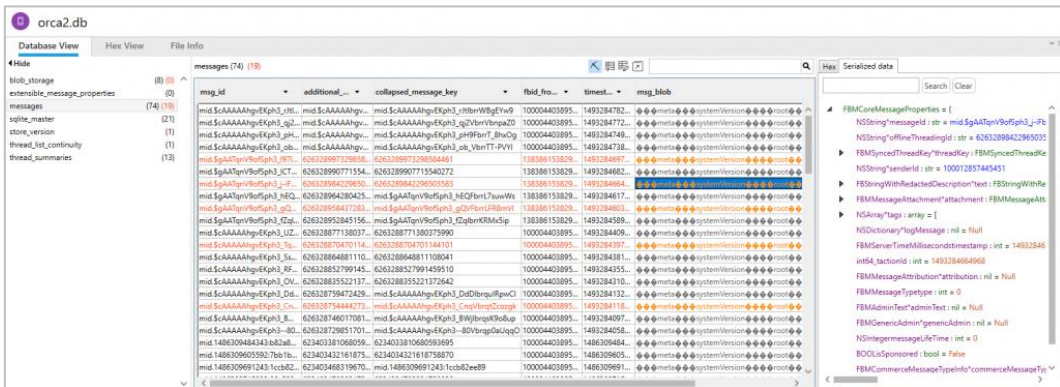
D21		Ghttp://www.reddit.com/r/funny/comments/1pnx9t/my_cousins_costume_hes_a_dentistand_yes_hes_at/My cousin's costume. He's a dentist...and yes, he's at work. : funny.H\														
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Type	Offset	Length	Data												
11	Unallocat	393270	66	H0												
12	Unallocat	434266	56	\6												
13	Unallocat	446544	71	g0												
14	Free Block	446700	180	-Ghttp://www.reddit.com/r/funny/comments/1pn79p/i_may_technically_be_an_adult_but_i_laughed_oc/I may technically be an adult, bu												
15	Free Block	448557	204	Mhttp://www.reddit.com/r/funny/comments/1pmnx5/my_five_year_old_dressed_as_michael_jackson_today/My five year old dressed as												
16	Unallocat	454740	76	p0												
17	Free Block	457046	201	Ghttp://www.reddit.com/r/funny/comments/1pn24h/this_kid_in_school_was_just_running_into_walls/This kid in school was just running												
18	Unallocat	491598	254	LLLLkkk Eahhttp://www.reddit.com/r/funny/comments/1pnh3/someone_in_my_school_drew_jfk_and_the_teacher/Someone in my school												
19	Free Block	492158	240]http://www.reddit.com/r/funny/?count=225&after=t3_1pn79sfunny.He 5ghttp://www.reddit.com/r/funny/comments/1pn79s/my_pregna												
20	Free Block	492700	169	Awhttp://www.reddit.com/r/funny/comments/1pnx46/i_work_at_a_retirement_home_it_is_halloween/I work at a retirement home. It is												
21	Free Block	493833	188	Ghttp://www.reddit.com/r/funny/comments/1pnx9t/my_cousins_costume_hes_a_dentistand_yes_hes_at/My cousin's costume. He's a de												
22	Free Block	494974	177]http://www.reddit.com/r/funny/?count=200&after=t3_1pmififunny.H=w+http://www.reddit.com/r/funny/comments/1pmifif/genius/Gen												
23	Unallocat	503894	77	m6												
24																
25																

Ein freies Tool, welches die derzeit bekannten Ansätze zur Wiederherstellung gelöschter Daten beinhaltet ist FQLite: <https://github.com/pawlaszczyk/fqlite>.



Cellebrite Software Physical Analyzer ist ebenfalls in der Lage SQLite Datenbanken aufzubereiten und gelöschte Records wiederherzustellen.

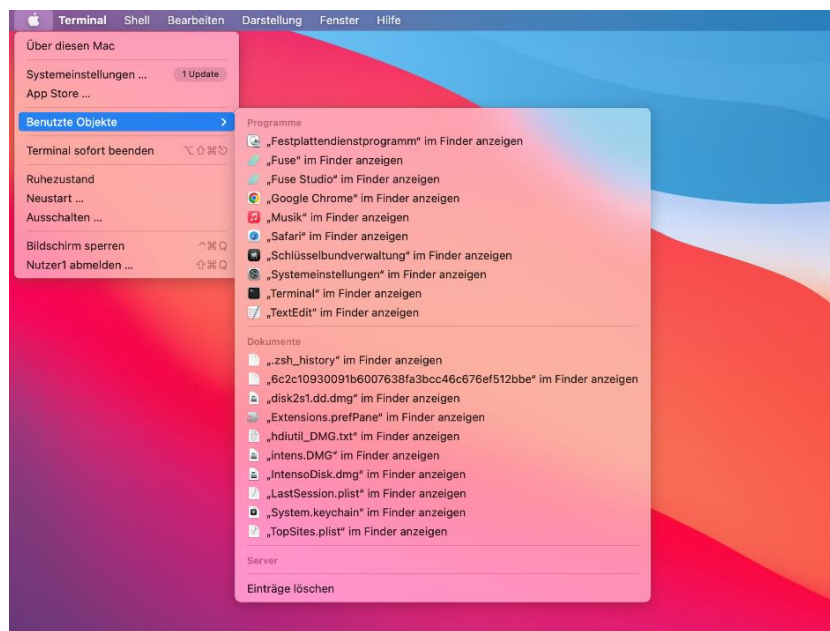
Zusätzlich existiert ein Tool, um unbekannte SQLite Datenbanken aufzubereiten und in Tabellenformate zu überführen, um sie berichtsfähig zu machen.



1.4 macOS Artefakte

1.4.1 Zuletzt genutzte Elemente und Nutzeraktivitäten

Recent/Verlaufseinträge können für den Benutzer über das Apfel Symbol unter dem Punkt „Benutzte Objekte“ angezeigt werden.



In macOS Versionen vor Big Sure werden diese unter folgenden Einträgen in Plist Dateien erfasst:

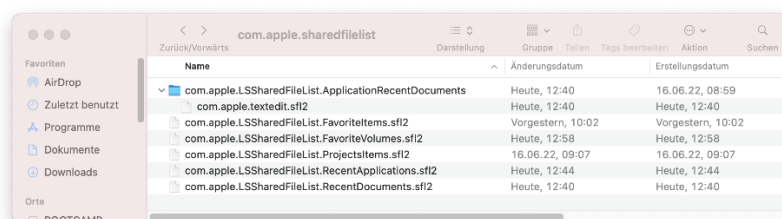
Recent Einträge: `%%users.homedir%%/Library/Preferences/com.apple.recentitems.plist`

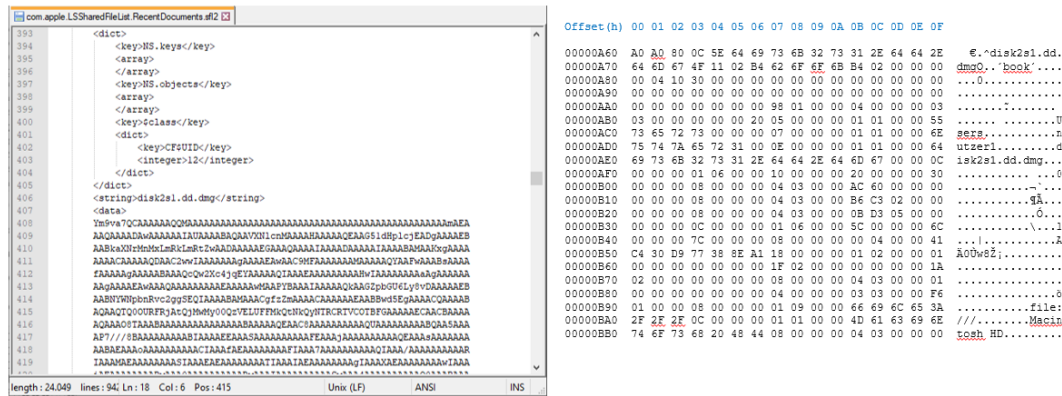
Recent Einträge pro Anwendung: `%%users.homedir%%/Library/Preferences/*LSSharedFileList.plist`

In macOS Versionen ab Big Sure werden diese unter folgenden Einträgen in *.slf2 (Bplist) Dateien erfasst:

Recent Einträge: `~/Library/Application Support/com.apple.sharedfilelist/*.slf2`

Recent Einträge pro Anwendung: `com.apple.LSSharedFileList.ApplicationRecentDocuments/*.slf2`





Die Terminal Kommando Historie mit der Bash Shell befindet sich unter %users.homedir%/.bash_history. Die der Z-Shell unter %users.homedir%/.zsh_history.

Weiterhin gehören pattern-of-life Daten zu den nützlichsten Informationen auf einem Gerät - sie erzählen die tatsächliche Geschichte über die Nutzung eines Gerätes durch den Benutzer. Auf iOS Geräten sind sie auch in der knowledgeC.db Datenbank vorhanden. Zusätzlich bietet die SQLite Datenbank Eintragungen zur genauen Benutzer- und Anwendungsnutzung.

1.4.1.1 KnowledgeC.db

Die KnowledgeC.db beinhaltet:

- Application Usage
- Application Activities
- Safari Browser History
- Device Power Status
- Lock Status (iOS Only)
- Battery Usage (iOS Only)
- App Installations (iOS Only)
- Audio Status (iOS Only)

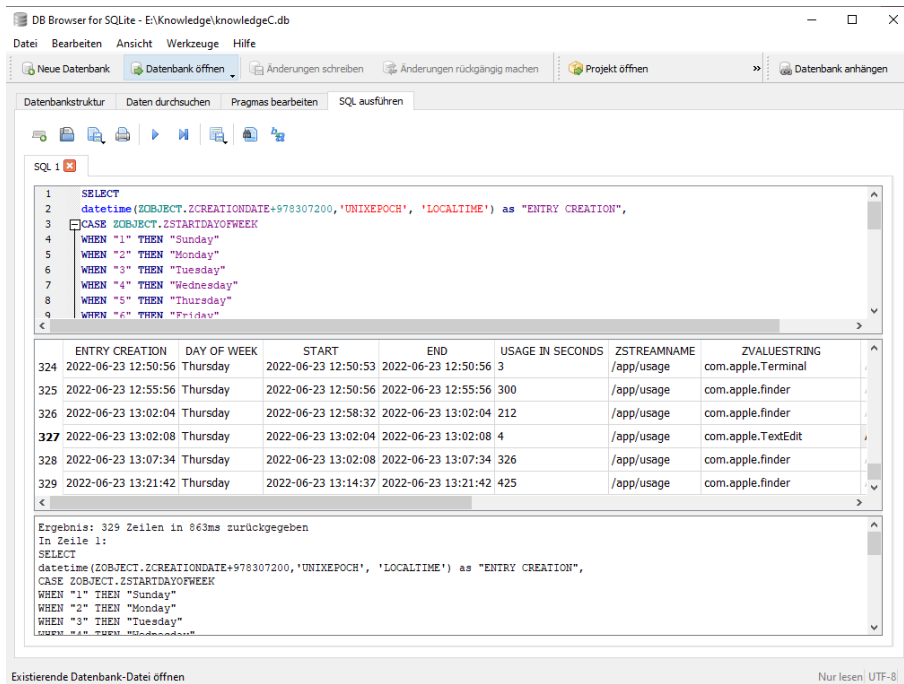
Fundstellen sind die System Kontext Datenbank unter „/private/var/db/CoreDuet/Knowledge/KnowledgeC.db“ und die Benutzerdatenbank unter %users.homedir%/Library/Application Support/Knowledge/KnowledgeC.db.

Folgende Eintragungen werden als Inhalt erfasst:

- "/activity/level"
- "/app/activity"
- "/app/inFocus"
- "/app/intents"
- "/app/usage"
- "/app/WebUsage"
- "/device/isPluggedIn"
- "/display/isBacklit"
- "/safari/history"

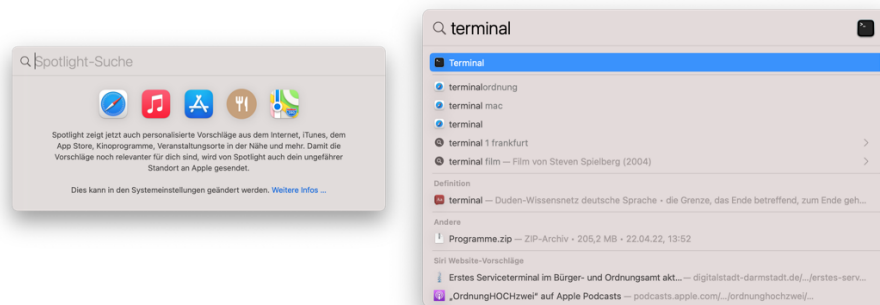
Im Folgenden wird der Aufbau der KnowledgeC.db erläutert. Die Datenbank hat viele Tabellen mit multiplen Spalten. Die Tabelle ZOBJECT enthält potenziell Nutzungseinträge für etwa 4 Wochen. Andere Tabellen, auf die ZOBJECT-Einträge verweisen können, befinden sich in den Tabellen ZSOURCE (Quelle der ZOBJECT-Einträge) und ZSTRUCTUREDMETADATA (Zusätzliche Metadaten, die ZOBJECT-Einträgen zugeordnet sind). Zeitstempel in dieser Datenbank verwenden die Mac-Epochezeit (01.01.2001 00:00:00 UTC).

Die folgende Abbildung zeigt die KnowledgeC.db.



1.4.2 Spotlight und erweiterte Metadaten

Spotlight ist der Name des Indexierungssystems, das in macOS integriert ist. Aufrufbar ist es mit „Befehlstaste/Command (⌘) + Leertaste“.



Spotlight ist aber auch für die kontinuierliche Indizierung von Dateien und Ordnern auf allen angeschlossenen Volumes verantwortlich. Weiterhin bewahrt es eine Kopie aller Metadaten für fast jede einzelne Datei und jeden Ordner auf der Festplatte auf.

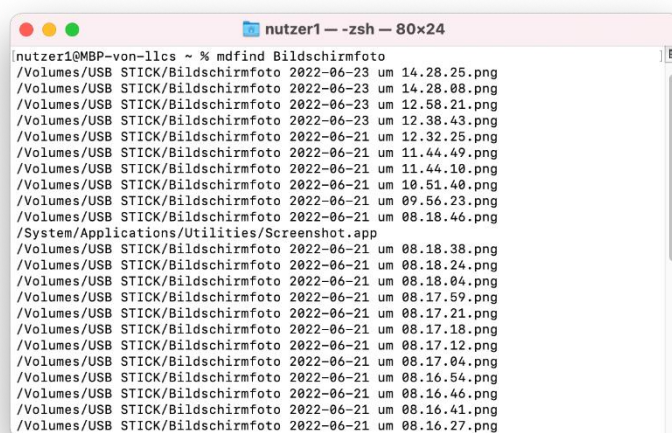
Es kann einige hervorragende Daten für Ihre Untersuchung liefern. Während viele der gleichen Informationen erhalten werden können, wenn Sie Zugriff auf das vollständige Disk-Image haben, ist bekannt, dass diese Datenbank Informationen enthält, die anderswo nicht verfügbar sind. Details wie das/die Datum(e) des letzten Öffnens oder wie oft (eine Anwendung oder Datei) geöffnet/verwendet wurde, sind nirgendwo anders im Dateisystem verfügbar.

Die Metadaten einer Datei können zudem auch über den Befehl `mdls./DATEI` im Terminal pro Datei abgerufen werden:



```
nutzer1@MBP-von-llcs ~ % mdls /Volumes/USB/STICK/Bildschirmfoto/2022-06-23/um
\12.58.21.png
_kMDItemDisplayNameWithExtensions = "Bildschirmfoto 2022-06-23 um 12.58.21.
png"
kMDItemAlternateNames = (
  "Bildschirmfoto 2022-06-23 um 12.58.21.png"
)
kMDItemBitsPerSample = 32
kMDItemColorSpace = "RGB"
kMDItemComment = "Screenshot"
kMDItemContentCreationDate = 2022-06-23 18:58:26 +0000
kMDItemContentCreationDate_Ranking = 2022-06-23 08:00:00 +0000
kMDItemContentModificationDate = 2022-06-23 18:58:26 +0000
kMDItemContentModificationDate_Ranking = 2022-06-23 08:00:00 +0000
kMDItemContentType = "public.png"
kMDItemContentTypeTree = (
  "public.png",
  "public.image",
  "public.data",
  "public.item",
  "public.content"
)
kMDItemLastUsedDate_Ranking = 2022-06-23 08:00:00 +0000
kMDItemLogicalSize = 291961
kMDItemOrientation = 0
kMDItemPhysicalSize = 294912
kMDItemPixelCount = 1364480
kMDItemPixelHeight = 656
kMDItemPixelWidth = 2080
kMDItemProfileName = "Farb-LCD"
kMDItemResolutionHeightDPI = 144
kMDItemResolutionWidthDPI = 144
kMDItemScreenCaptureGlobalRect = (
  285,
  372,
  1040,
  328
)
kMDItemScreenCaptureType = "window"
kMDItemUseCount = 1
kMDItemUsedDates = (
  "2022-06-22 22:00:00 +0000"
)
nutzer1@MBP-von-llcs ~ %
```

Die Suche nach Dateien kann über den Befehl „`mdfind`“ im Terminal gestartet werden. Dafür wird „`mdfind` Suchbegriff“ angegeben.



```
nutzer1@MBP-von-llcs ~ % mdfind Bildschirmfoto
/Volumes/USB/STICK/Bildschirmfoto/2022-06-23/um/14.28.25.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-23/um/14.28.08.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-23/um/12.58.21.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-23/um/12.38.43.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/12.32.25.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/11.44.49.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/11.44.10.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/10.51.40.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/09.56.23.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.18.46.png
/System/Applications/Utilities/Screenshot.app
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.18.38.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.18.24.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.18.04.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.17.59.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.17.21.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.17.18.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.17.12.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.17.04.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.16.54.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.16.46.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.16.41.png
/Volumes/USB/STICK/Bildschirmfoto/2022-06-21/um/08.16.27.png
```

die Informationen die Spotlight dabei nutzt, befinden sich in Datenbank Dateien store bzw. `.store`. Diese Dateien sind im jeweiligen Datenträger unter „`/.Spotlight-V100/Store-V2/<UUID>/store`“ zu

finden. Seit macOS 10.13 existiert zudem eine Datenbank für jeden Nutzer unter „~/Library/Metadata/CoreSpotlight/index.spotlightV3/.store“.

Leider verwendet Spotlight ein proprietäres, undokumentiertes Format und es gibt keinen öffentlich verfügbaren Code von Apple, um es zu lesen. Es existiert jedoch ein Spotlight parser für Python der hier genutzt werden kann. Dieser ist unter „http://github.com/ydkhatri/spotlight_parser“.

```
C:\spotlight_parser>c:\Python27\python.exe spotlight_parser.py g:\ElCapitan\store.db c:\output\elcap
INFO - Output folder 'c:\output\elcap' does not exist! Creating it for you.
INFO - Processing g:\ElCapitan\store.db
INFO - Creating output file c:\output\elcap\spotlight-store_data.txt
INFO - Creating output file c:\output\elcap\spotlight-store_fullpaths.csv
DEBUG - Trying to decompress compressed block @ 19014
DEBUG - Trying to decompress compressed block @ 7D014
DEBUG - Trying to decompress compressed block @ 71014
DEBUG - Trying to decompress compressed block @ A5014

... output snipped ...

DEBUG - Trying to decompress compressed block @ 7F9014
DEBUG - Trying to decompress compressed block @ 801014
...
DEBUG - Err, could not find path for id 956655
DEBUG - Err, could not find path for id 957257
DEBUG - Err, could not find path for id 963495
INFO - Finished in time = 00:00:48
```

Rot sind Metadaten, die sich auf einen einzelnen Eintrag in der Datenbank beziehen, einschließlich Datum und Uhrzeit der letzten Aktualisierung. Danach folgen die Metadaten selbst.

Die Elemente in Blau sind Informationen, die nur in der Spotlight-Datenbank verfügbar sind. Die letzten beiden können für einen Ermittler von besonderem Interesse sein.

```
Inode_Num --> 100
Flags --> 0
Store_ID --> 5824
Parent_Inode_Num --> 0
Last_Updated --> 2018-02-20 00:10:56
_kMDItemContentChangeDate --> 2017-11-09 22:57:58
_kMDItemCreationDate --> 2017-11-09 23:03:35
_kMDItemCreatorCode --> 0
_kMDItemFileName --> 96206759_099496705b_b.jpg
_kMDItemFinderFlags --> 0
_kMDItemFinderLabel --> 0
_kMDItemGroupId --> 13
_kMDItemIsExtensionHidden --> 0
_kMDItemOwnerGroupID --> 99
_kMDItemOwnerUserID --> 99
_kMDItemTextContentIndexExists --> 0
_kMDItemTypeCode --> 0
kMDItemBitsPerSample --> 32

kMDItemInterestingDate_Ranking --> 2017-11-09 00:00:00
kMDItemKind --> JPEG image
kMDItemLastUsedDate --> 2017-11-09 23:13:43.323351
kMDItemLastUsedDate_Ranking --> 2017-11-09 00:00:00
kMDItemLogicalSize --> 763434
kMDItemOrientation --> 0
kMDItemPhysicalSize --> 786432
kMDItemPixelCount --> 786432
kMDItemPixelHeight --> 768
kMDItemPixelWidth --> 1024
kMDItemResolutionHeightDPI --> 72
kMDItemResolutionWidthDPI --> 72
kMDItemUseCount --> 6
kMDItemUsedDates --> 2017-11-09 05:00:00, 2018-02-19 05:00:00
```

1.4.3 Gelöschte Dateien

1.4.3.1 Der Papierkorb

Auf jedem Laufwerk existiert ein Verzeichnis `.Trashes` welches die gelöschten Dateien in einem Unterverzeichnis beinhaltet. Die darin befindlichen Dateien haben ihren originalen Dateinamen behalten.

Name	Beschreibung	Erw.	Größe
.. = .Trashes (4)	existierend		611 KB
. = 502 (4)	existierend		611 KB
.DS_Store	existierend		6,0 KB
Bildschirmfoto 2022-06-23 um 16.17.42.png (2)	existierend	png	605 KB

Im `.Trashes` Verzeichnis befindet sich zudem eine Datei `.DS_Store`. Diese beinhaltet die notwendigen Pfadangaben der gelöschten Inhalte, um diese wiederherstellen zu können.

Die `.DS_Store` Dateien enthalten neben den Dateinamen Informationen über die Ansichtseinstellungen, der Position von Icons, Sortiereinstellungen, Informationen über Fenstergröße und -position sowie andere Metadaten. `.DS_Store` Dateien aus dem Papierkorb können Auskunft über Struktur und Ordnernamen eventuell bereits gelöschter Daten geben, auch wenn diese durch mehrfaches Überschreiben oder ähnliche Maßnahmen tatsächlich physikalisch nicht mehr nachweisbar sind.

`.DS_Store` Dateien können mit einem speziellen Parser gelesen werden: https://github.com/hanwenzhu/.DS_Store-parser.

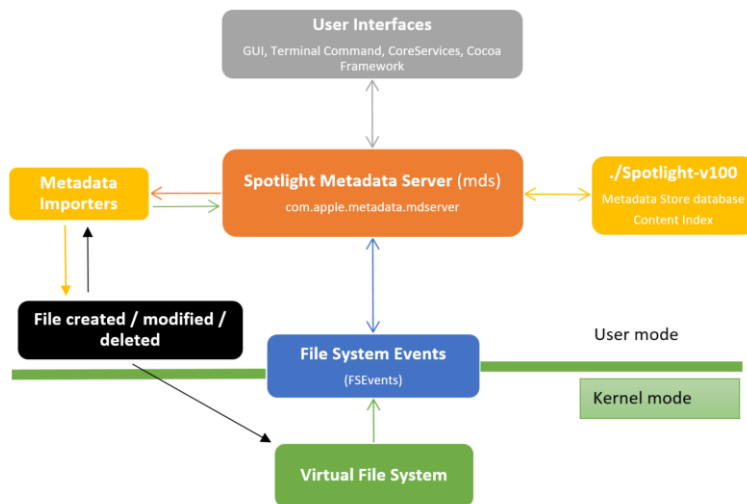
Im folgenden Beispiel ist das `Bildschirmfoto 2022-06-23 um 16.17.42.png` aus dem Verzeichnis / vom Datenträger gelöscht worden.

```
F:\>python F:\dsparse.py F:\.DS_Store
Bildschirmfoto 2022-06-23 um 16.17.42.png
  ptbL (unrecognized): '/'
  ptbN (unrecognized): 'Bildschirmfoto 2022-06-23 um 16.17.42.png'

F:\>
```

1.4.3.2 Dateisystemartefakte und deren Zusammenspiel

Die folgende Abbildung zeigt das Zusammenspiel von Dateisystemartefakten.



1.4.3.3 FSEvents

FSEvents wurden mit macOS 10.7 Lion eingeführt und sind auf macOS Systemen und externen Datenträgern zu finden. Es registriert Dateisystemänderungen in FSEvent log Dateien (gzip). Dazu zählen Historische Ereignisse von Änderungen am Dateisystem. Außerdem können Protokolle Tage bis Monate umfassen. Die Datensätze werden hierbei alphabetisch und nicht chronologisch gespeichert.

1.4.3.4 FSEvent Logs

FSEvent Logs lassen sich in macOS (System oder Laufwerk) unter „/.fsevents/xxxxxxx“ finden. Sie liegen im Gzip archive format vor. Der Name ist die letzte gespeicherte Event ID im FSEvent log + 1, beispielsweise „0000000000a4b3e“ oder 674,622 Dezimal.

FSEvent logs können mit einem speziellen Parser gelesen werden: <https://github.com/dlcowen/FSEventsParser>.

```
C:\Users\John>F:\FSEParser_V4.exe -s F:\.fsevents -o f:\fsout -t folder
=====
FSEParser v 4.0 -- provided by G-C Partners, LLC
=====
[Info]: Report queries file not specified using the -q option. Custom reports will not be generated.
[Info]: No casename specified using -c. Defaulting to "FSE_Reports".
[STARTED] 06/23/2022 14:29:25 UTC Parsing files.
  File 4 of 4 [=====] 100.0%
  All Files Attempted: 4
  All Parsed Files: 4
  Files with Errors: 0
  All Records Parsed: 22
[FINISHED] 06/23/2022 14:29:25 UTC Parsing files.
[STARTED] 06/23/2022 14:29:25 UTC Sorting fsevents table in Database.
[FINISHED] 06/23/2022 14:29:25 UTC Sorting fsevents table in Database.
[STARTED] 06/23/2022 14:29:25 UTC Exporting fsevents table from Database.
[FINISHED] 06/23/2022 14:29:25 UTC Exporting fsevents table from Database.

  Exception log and Reports exported to:
  'f:\fsout\FSE_Reports'
C:\Users\John>
```

DB Browser for SQLite - F:\fsout\FSE_Reports\FSEvents.sqlite

Neue Datenbank | Datenbank öffnen | Änderungen schreiben | Änderungen rückgängig machen | Projekt öffnen | Projekt speichern | Datenbank anhängen | Datenbank schließen

Datenbankstruktur | Daten durchsuchen | Pragma bearbeiten | SQL ausführen

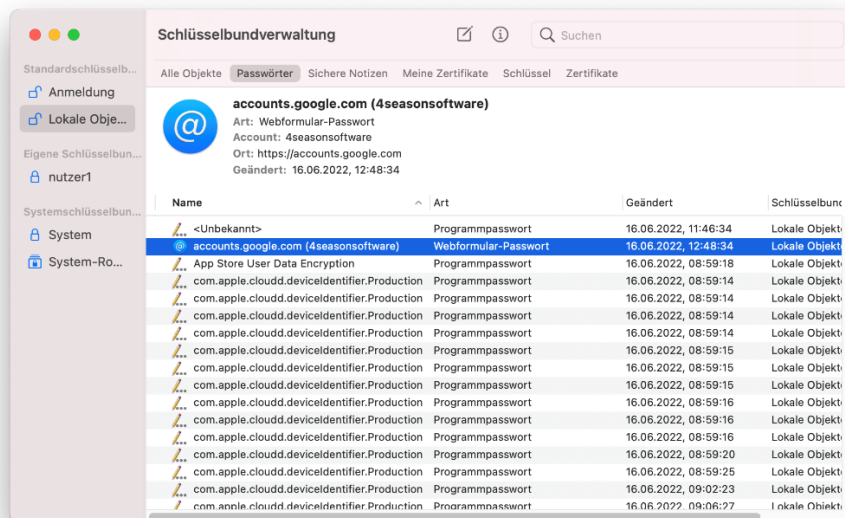
Tabelle: fsevents

id	id_hex	fullpath	filename	type	flags
9	00000000000000000000000000000000	FileEvent;	Created;
10	9552869	00000000000000000000000000000000	9552869	FolderEvent;	FolderCreated;Removed;
11	9552878	00000000000000000000000000000000	9552878	FileEvent;	Created;
12	9552872	00000000000000000000000000000000	9552872	FileEvent;	Created;
13	9552860	00000000000000000000000000000000	9552860	FileEvent;	Modified;Created;
14	9605461	00000000000000000000000000000000	9605461	FolderEvent;	FolderCreated;PermissionChange;
15	9605473	00000000000000000000000000000000	9605473	FolderEvent;	FolderCreated;InodeMetaMod;PermissionChange...
16	9608115	00000000000000000000000000000000	9608115	FileEvent;	Modified;InodeMetaMod;Created;FinderInfoMod;
17	9605477	00000000000000000000000000000000	9605477	FileEvent;	Renamed;
18	9605476	00000000000000000000000000000000	9605476	FileEvent;	Modified;Renamed;InodeMetaMod;Created;Permi...
19	9609665	00000000000000000000000000000000	9609665	FolderEvent;	
20	9609666	00000000000000000000000000000000	9609666	FolderEvent;	EndOfTransaction;
21	9612241	00000000000000000000000000000000	9612241	FolderEvent;	
22	9612242	00000000000000000000000000000000	9612242	FolderEvent;	EndOfTransaction;

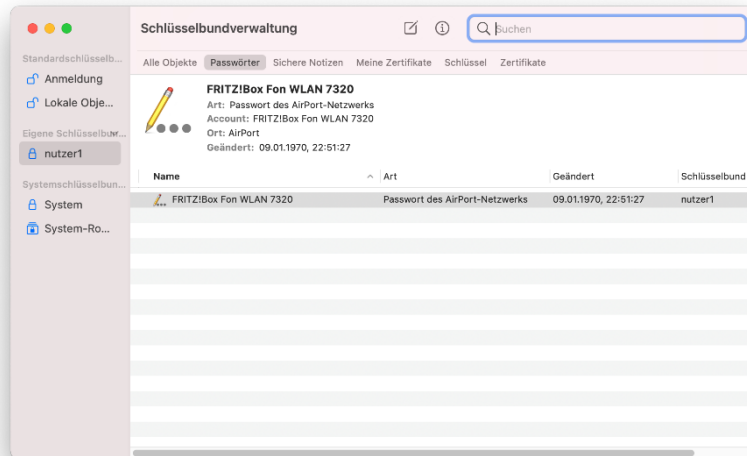
9 - 22 von 22 | Springe zu: 1

1.4.4 Schlüsselbund/Keychain

Passwörter (Zugänge, Web, etc.) und Zertifikate werden in macOS in der Schlüsselbundverwaltung / Keychain gespeichert und verwaltet. Auf neueren Geräten mit T2 und M1 Chipsatz sind die Passwörter zudem an den Hardwarekey HEK gebunden. Passwörter könne nur mit bekanntem Passwort des Benutzers angezeigt werden.



Durch einen Doppelklick können die Elemente geöffnet werden und nach Eingabe des Kennworts des Benutzers angezeigt werden:

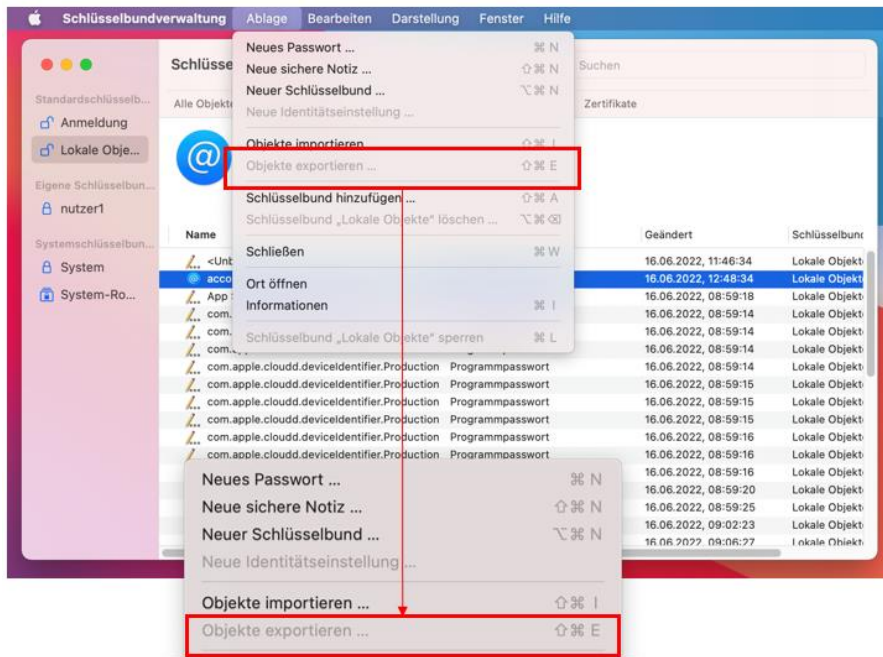


Die Passwörter werden in mehreren Keychain Dateien in einem Keychain Verzeichnis (Keychain Directory) abgelegt und können damit auch kopiert werden. Sie befinden sich unter `%%users.home-dir%%/Library/Keychains/*`. Mit einem Python Tool kann dann die Keychain entschlüsselt werden: <https://github.com/n0fate/chainbreaker>.

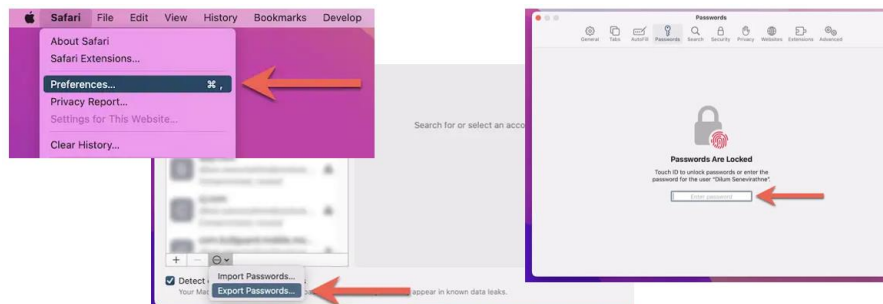
```
./chainbreaker.py --password=TestPassword -a test_keychain.keychain
2020-11-12 15:58:18,925 - INFO -
ChainBreaker 2 - https://github.com/gaddie-3/chainbreaker
2020-11-12 15:58:18,925 - INFO - Runtime Command: chainbreaker.py --password=TestPassword -a test_keychain.keychain
2020-11-12 15:58:18,925 - INFO - Keychain: test_keychain.keychain
2020-11-12 15:58:18,925 - INFO - Keychain MDS: eb3abc06c22afa388ca522ea5aa032fc
2020-11-12 15:58:18,925 - INFO - Keychain 256: 2d76f564ac24fa6a8a22adb6d5cb9b430032765b1ba3effa8ddea38222008441
2020-11-12 15:58:18,925 - INFO - Dump Start: 2020-11-12 15:58:18.925479
2020-11-12 15:58:19,245 - INFO - 1 Keychain Password Hash
2020-11-12 15:58:19,245 - INFO - $keychain$7255a69abe21a28e1d2967265c9bba9c9bf4daf1*28dcfa41552db4eb*9dabb91712bb6a38f461b4335c334d444eb0c451e5fa02183eafe05c
35310d76014bc04b699d420d8487d4452d067e5
2020-11-12 15:58:19,245 - INFO -
2020-11-12 15:58:19,245 - INFO - 2 Generic Passwords
2020-11-12 15:58:20,306 - INFO - [+] Generic Password Record
2020-11-12 15:58:20,306 - INFO - [-] Create DateTime: 2020-10-13 23:01:17
2020-11-12 15:58:20,306 - INFO - [-] Last Modified DateTime: 2020-10-13 23:01:17
2020-11-12 15:58:20,306 - INFO - [-] Description: secure note
2020-11-12 15:58:20,306 - INFO - [-] Creator:
2020-11-12 15:58:20,306 - INFO - [-] Type: note
2020-11-12 15:58:20,307 - INFO - [-] Print Name: Test Secure Note
2020-11-12 15:58:20,307 - INFO - [-] Alias:
2020-11-12 15:58:20,307 - INFO - [-] Account:
2020-11-12 15:58:20,307 - INFO - [-] Service: Test Secure Note
2020-11-12 15:58:20,307 - INFO - [-] Base64 Encoded Password:
PD94bWogdmYyc21vbj0MS4wIi1lbnV2ZGluz0lVVRGLTg1Pz4RPFCEFNUNWVFIHbaaXNOIFBVQkxJQyAiLS8vOXhBbGUvL0RURCBQcTETVCAxLjAvL0VOIiAiaHR0cDovLzI3d3dy
5hcHbz2S5jb20vRFR9cy9cm9wZXJ0eXpc3QtMS4wLmR0ZCI=

```

Auf macOS Geräten vor BigSure ist es möglich Passwörter zu exportieren. Dazu wählt man die Objekte aus. Unter Ablage existiert ein Punkt „Objekte exportieren“. Auf neueren Geräten ist diese Option ausgegraut und nicht verfügbar.



Wenn auf dem Mac jedoch macOS Monterey oder höher ausgeführt wird, kann der integrierte Passwort-Manager von Safari verwendet werden, um Anmeldeinformationen im CSV-Dateiformat zu speichern.



1.4.5 Logdateien

1.4.5.1 Log-Dateien des Betriebssystems

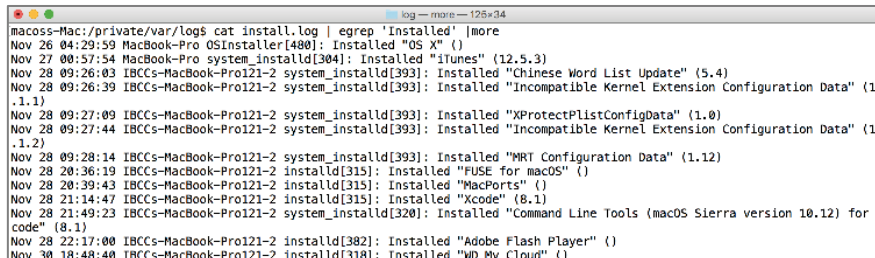
Die Log-Dateien des Betriebssystems befinden sich im Verzeichnis „/private/var/log“. MacOS hat bis zur Version macOS 10.12 / 11 BigSur verschiedene „überlieferte“ Logdateiformate genutzt. Die Log Formate stammen aus dem Betriebssystem Sun Solaris und wurde in verschiedenen Formen von BSD Unix, einschließlich Mac OS übernommen.

1.4.5.2 Nutzer-/Account-Informationen

Informationen zu eingeloggtten Nutzern bzw. Nutzer-Accounts können in der Datei „/private/var/log/accountpolicy.log“ eingesehen werden.

1.4.5.3 Software-Installationen

Installationen von Betriebssystem-Versionen, Updates und Apps lassen sich in der Datei „/private/var/log/install.log“ ermitteln. Die Logdatei kann mit dem Terminalbefehl „cat install.log | grep "Suchbegriff"“ analysiert werden.



```
macosx-Mac:/private/var/logs cat install.log | egrep 'Installed' | more
Nov 26 04:29:59 MacBook-Pro OSInstaller[480]: Installed "OS X" ( )
Nov 27 00:57:54 MacBook-Pro system_installd[304]: Installed "iTunes" (12.5.3)
Nov 28 09:26:03 IBCCs-MacBook-Pro121-2 system_installd[393]: Installed "Chinese Word List Update" (5.4)
Nov 28 09:26:39 IBCCs-MacBook-Pro121-2 system_installd[393]: Installed "Incompatible Kernel Extension Configuration Data" (12.1.1)
Nov 28 09:27:09 IBCCs-MacBook-Pro121-2 system_installd[393]: Installed "XProtectPlistConfigData" (1.0)
Nov 28 09:27:44 IBCCs-MacBook-Pro121-2 system_installd[393]: Installed "Incompatible Kernel Extension Configuration Data" (12.1.2)
Nov 28 09:28:14 IBCCs-MacBook-Pro121-2 system_installd[393]: Installed "MRT Configuration Data" (1.12)
Nov 28 20:36:19 IBCCs-MacBook-Pro121-2 installd[315]: Installed "FUSE for macOS" ( )
Nov 28 20:39:43 IBCCs-MacBook-Pro121-2 installd[315]: Installed "MacPorts" ( )
Nov 28 21:14:47 IBCCs-MacBook-Pro121-2 installd[315]: Installed "Xcode" (8.1)
Nov 28 21:49:23 IBCCs-MacBook-Pro121-2 system_installd[320]: Installed "Command Line Tools (macOS Sierra version 10.12) for Xcode" (8.1)
Nov 28 22:17:00 IBCCs-MacBook-Pro121-2 installd[382]: Installed "Adobe Flash Player" ( )
Nov 30 18:48:40 IBCCs-MacBook-Pro121-2 installd[318]: Installed "iWork My Cloud" ( )
```

1.4.5.4 Filesystem Check

Die Log-Dateien „/private/var/log/fsck_hfs.log“ oder „/private/var/log/fsck_apfs.log“ zeigen an, wann eine Dateisystemprüfung für die Dateisysteme HFS+ und Apple File System durchgeführt wurde. Neben dem Character Device wird das Ergebnis der Prüfung (CLEAN, ERROR) ausgegeben.

1.4.5.5 Storage Manager

Die Log-Datei „/private/var/log/com.apple.revisiond/revisiond.log“ zeigt Informationen zum Mac-OS-Storage-Manager revisiond. Der Hintergrunddienst verwaltet unterschiedliche Fassungen von Dokumenten, die von Applikationen oder Mac-OS-Systemdiensten erzeugt wurden. In das Verzeichnis „/private/var/log/com.apple.revisiond/“ kann nur mit Root-Rechten navigiert werden. Die Log-Datei kann u. a. Hinweise auf Volumes geben, beispielsweise bei der Löschung von Cache-Speicher.

1.4.5.6 WiFi

Von Mac OS hergestellte WiFi-Verbindungen lassen sich in der Log-Datei „/private/var/log/wifi.log“ auffinden. Die Log-Datei speichert WiFi-Verbindungen der letzten 24 Stunden und zeigt detaillierte Informationen zum Verbindungsaufbau mit WiFi-Access-Points. WiFi-Log-Dateien von vergangenen Tagen werden bzip2-komprimiert abgespeichert.

Die folgende Abbildung zeigt den Verbindungsaufbau mit der BSSID E2:5F:45:75:14:08 von der eigenen Mac-Adresse.

```

log -- sh -- 125x38
Sun Apr 9 22:23:37.789 <kernel> en0: Terminating supplicant.
Sun Apr 9 22:23:37.789 <kernel> RSNSupplicant: Releasing authenticator for e2:5f:45:75:14:88
Sun Apr 9 22:23:37.727 <kernel> parseRSNIE: groupCipherType = 5 pairwiseCipherType = 5 authSel = 2
Sun Apr 9 22:23:37.727 <kernel> initWithInterfaceAndIE: _myMacAddress a0:99:9b:13:f4:d1
Sun Apr 9 22:23:37.727 <kernel> setPMK: PMK SET!
Sun Apr 9 22:23:37.830 <kernel> en0: Received EAPOL packet (length = 113)
Sun Apr 9 22:23:37.830 <kernel> inputEAPOLFrame: 0 extra bytes present in EAPOL frame.
Sun Apr 9 22:23:37.830 <kernel> inputEAPOLFrame: Received message 1 of 4
Sun Apr 9 22:23:37.830 <kernel> FULL RSN IE FOUND:
Sun Apr 9 22:23:37.830 [00000000] 30 14 01 00 00 0F AC 04 01 00 00 0F AC 04 01 00 00 0F AC 02 0C 00
Sun Apr 9 22:23:37.830 <kernel> storeFullRSNIE: getAP_IE_LIST returned 0
Sun Apr 9 22:23:37.830 <kernel> PMK:

```

1.4.5.7 Periodische Log-Dateien

Die Log-Dateien werden von Mac OS in periodischen Abständen angelegt und protokollieren system-spezifische Vorgänge. Die Log-Dateien befinden sich unter „/private/var/log“ und sind mit einem Texteditor oder im Terminal beispielsweise mit dem Kommando „cat“ lesbar. Sie beinhalten Informationen wie daily.out (Disk-Status, Network-Status, System-Status), weekly.out (Protokollierung des Neuaufbaus der Whatis-Datenbank) und monthly.out (Statistik zu Nutzer-Logins).

Die Whatis-Datenbank ist eine BSD-Komponente zur Ausgabe von Informationen zu Schlüsselbegriffen. Zum Beispiel können mit dem Befehl „whatis grep“ Informationen zum UNIX-Kommando „grep“ abgerufen werden.

Die Log-Dateien werden von Mac OS in periodischen Abständen angelegt und protokollieren system-spezifische Vorgänge. Die folgende Abbildung zeigt einen Auszug aus der Log-Datei daily.out mit einer grep-Suche nach eingehängten Volumes.

```

log -- -bash -- 118x33
IBCCs-MacBook-Pro121:log ibcc$ cat daily.out | grep "\Volumes"
/dev/disk2s1 1.9Gi 13Mi 1.9Gi 1% 512 0 100% /Volumes/ITB LIVE
/dev/disk2s2 465Gi 465Gi 0Bi 100% 122012666 0 100% /Volumes/iacis
/dev/disk2s2 7.2Gi 5.4Gi 1.8Gi 75% 1405151 476949 75% /Volumes/Install OS X Yosemite
/dev/disk4s2 46Gi 17Gi 29Gi 38% 4570055 7511050 38% /Volumes/OSX 10.11
/dev/disk4s8 46Gi 149Mi 46Gi 1% 38236 12104244 0% /Volumes/OSX 10.6
/dev/disk4s4 47Gi 426Mi 46Gi 1% 109031 12097998 1% /Volumes/OSX 10.10
/dev/disk4s5 47Gi 416Mi 46Gi 1% 106605 12085026 1% /Volumes/OSX 10.9
/dev/disk4s9 47Gi 289Mi 46Gi 1% 73913 12117718 1% /Volumes/OSX 10.5
/dev/disk4s12 46Gi 42Gi 3.9Gi 92% 11109944 1016152 92% /Volumes/OSX Software I
/dev/disk4s6 46Gi 149Mi 46Gi 1% 38194 12120670 0% /Volumes/OSX 10.8

```

1.4.5.8 System Log-Dateien im Text Format

Bis zur macOS Version 10.8 gab es drei Log-Dateien im Text Format, die Systemnachrichten speicherten. Dazu zählten secure.log, kernel.log und system.log. Ab macOS Version 10.8 wurden diese zum system.log zusammengefasst. Ab macOS Version 10.12 und mit Version 11 BigSure werden Systemnachrichten im Apple Unified Log (ALU) zusammengefasst aufgezeichnet.

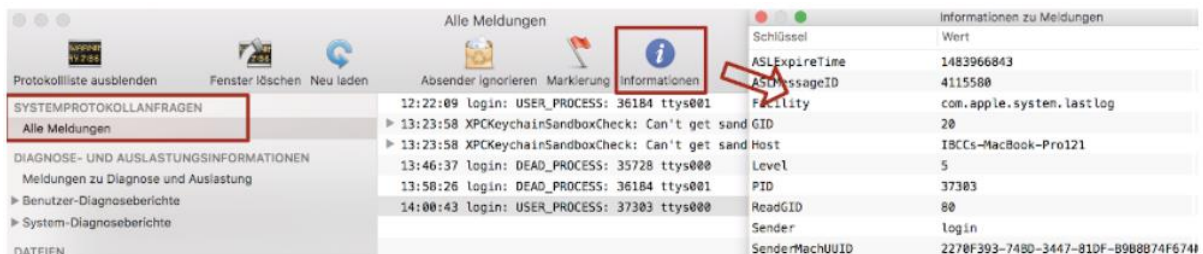
1.4.5.9 Binary Apple System Log (ASL)

Mac OS hat als BSD-System Zugang zum unter UNIX-Systemen weit verbreiteten Log-Mechanismus System Log. System Log (syslog) ist ein Hintergrundprozess, der von verschiedenen Komponenten des Betriebssystems Meldungen entgegennimmt und aufzeichnet. Ab Mac OS X 10.4 wurde der UNIX-

Mechanismus in Apple System Log (ASL) überführt. Die Grundlage für ASL ist syslog, wodurch es mit älteren Standards kompatibel ist.

ASL-Logfiles befinden sich im Verzeichnis „/private/var/log/asl“. Sie haben ein binäres Format mit der Signatur „ASL DB“. Grundsätzlich gibt es zwei verschiedene Arten von ASL-Log-Dateien. Dazu zählen Dateien mit Zeitstempeln im Format YYYY.MM.DD.*.asl und Dateien mit der Bezeichnung AUX.YYYY.MM.DD.asl. Beide speichern Informationen für einen Zeitraum von 7 Tagen, bevor sie in die Log-Dateien mit der Bezeichnung BB.*.asl überführt werden. BB.*.asl Dateien beinhalten eine monatliche Speicherung der protokollierten Aktionen und werden für ein Jahr gespeichert.

ASL-Log-Dateien können mit der Console.app unter Auswahl von Systemprotokollanfragen/ Alle Meldungen intuitiv betrachtet werden. Jede ASL-Meldung enthält weitere Informationen, die in Schlüsseln (ASL Keys) abgelegt sind welche durch Auswahl der Schaltfläche Informationen verfügbar sind. Die folgende Abbildung zeigt eine ASL-Meldung eines Bash-Login-Vorgangs.



Alternativ kann eine Ausgabe auch unter Zuhilfenahme des Kommandos „syslog“ erfolgen. Dabei können einzelne Dateien oder ganze Verzeichnisse mit entsprechenden ASLLog- Dateien betrachtet werden. Der Befehl „syslog -d /private/var/log/asl/“ gibt die Inhalte des gesamten ASL-Log-Verzeichnisses wieder. Die folgende Abbildung zeigt einen Auszug aus der Ausgabe der ASL-Log-Dateien mit syslog.

```
sh-3.2# syslog -d /private/var/log/asl/
Oct 1 10:57:37 localhost bootlog[0] <Notice>: BOOT_TIME 1443689857 0
Oct 1 10:57:59 IBCCs-MacBook-Pro121 loginwindow[94] <Notice>: USER_PROCESS: 94 console
Oct 1 10:58:08 macbook-pro loginwindow[435] <Notice>: USER_PROCESS: 435 console
Oct 1 16:40:18 macbook-pro login[1429] <Notice>: USER_PROCESS: 1429 ttys000
Oct 1 16:41:19 macbook-pro login[1465] <Notice>: USER_PROCESS: 1465 ttys001
Oct 1 16:46:32 macbook-pro login[1465] <Notice>: DEAD_PROCESS: 1465 ttys001
Oct 1 16:46:58 macbook-pro login[1429] <Notice>: DEAD_PROCESS: 1429 ttys000
```

1.4.5.10 Basic Security Modules Audit Logs (BSM)

Das Dateiformat Basic Security Module (BSM) stammt aus dem Betriebssystem Sun Solaris und wurde in verschiedenen Formen von BSD Unix, einschließlich Mac OS X, übernommen. Audit-Logs basieren auf einer Apple-eigenen Implementierung des Basic-Security-Module (BSM) in Darwin. Außerdem ermöglichen sie die Verfolgung und Prüfung von Aktionen, die von Nutzern oder Prozessen durchgeführt werden.

Audit-Logs sind im Verzeichnis „/private/var/audit/“ abgelegt, welches nur mit Root-Rechten geöffnet werden kann. Log-Dateien liegen in einem binären Format vor und haben das Format [Start-Zeitstempel].[End-Zeitstempel]. Die Log-Datei [Start-Zeitstempel].not terminated ist die aktuell beschriebene Audit-Log-Datei. Zeitstempel haben das Format YYYYMMDDHHMMSS.

Um Audit-Log-Dateien auszuwerten, verwendet man das Kommando „praudit -xn“. Dabei stehen die Parameter -x für eine XML-Ausgabe und -n für eine Konvertierung der Nutzer- und Gruppen-IDs. Die folgende Abbildung zeigt einen Auszug der Ausgabe der Audit-Logfiles mit „praudit -xn“.

```
sh-3.2# praudit -xn /private/var/audit/*
<?xml version='1.0' encoding='UTF-8'?>
<audit>
<record version="11" event="audit startup" modifier="0" time="Mon Apr 27 11:29:30 2015" msec=" + 697 msec" >
<text>launchd:Audit startup</text>
<return errval="success" retval="0" />
</record>
<record version="11" event="session start" modifier="0" time="Mon Apr 27 11:29:37 2015" msec=" + 54 msec" >
<argument arg-num="1" value="0x0" desc="sflags" />
<argument arg-num="2" value="0x0" desc="am_success" />
<argument arg-num="3" value="0x0" desc="am_failure" />
<subject audit-uid="1" uid="0" gid="0" ruid="0" rgid="0" pid="0" sid="100003" tid="0 0.0.0.0" />
<return errval="success" retval="0" />
</record>
```

Audit-Einträge sind durch Tags (ähnlich wie ein XML-/ HTML-Dokument) nach folgendem Muster hierarchisch strukturiert:

Struktur eines Audit-Eintrags	
<record...>	Beginn des Eintrags (Header)
<subject...>	Betreff (Subject), Kontext der Aktion (IDs)
<text>	String mit Beschreibung zur Aktion
<return>	Rückgabewert
</record>	Ende des Eintrags

Einzelne Aktionen aus den Audit-Log-Dateien können mit dem Befehl auditreduce gefiltert werden. Die Filterung kann nur nach bestimmten Audit-Events erfolgen, die in der Konfigurationsdatei „/etc/security/audit_event“ beschrieben sind. Die folgende Abbildung zeigt die Filterung nach Nutzer-Logins (AUE_lw_login) in der Audit-Datei 20160105164801.not_terminated.

```
sh-3.2# auditreduce -m AUE_lw_login 20160105164801.not_terminated | praudit -xn
<?xml version='1.0' encoding='UTF-8'?>
<audit>
<record version="11" event="loginwindow login" modifier="0" time="Tue Jan 5 17:48:05 2016" msec=" + 632 msec" >
<subject audit-uid="501" uid="0" gid="0" ruid="501" rgid="20" pid="97" sid="100007" tid="503316500.0.0.0" />
<return errval="success" retval="0" />
</record>
```

1.4.5.11 Apple Unified Log (AUL)

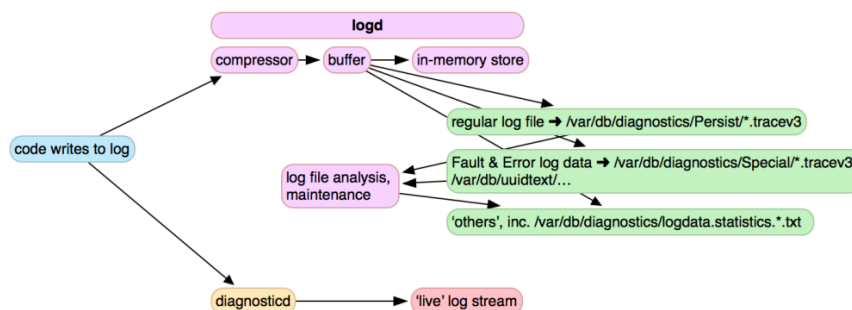
Mit macOS 10.12 Sierra ersetzte Apple im September 2016 das traditionelle textbasierte Unix-Protokollsystem durch sein neues einheitliches Protokoll, dem Apple Unified Logging. Der Log-Mechanismus ersetzt die bisherigen Log-Dateien system.log und Apple System Logs (ASL) bzw. führt sie zusammen. Das neue Unified Logging wird plattformübergreifend auch in iOS (ab iOS-Version 10), watchOS und tvOS eingesetzt. Mit macOS Sierra 10.12 wird das Unified Logging parallel zu den bekannten Log-Dateien betrieben, ab macOS 11 BigSure ausschließlich.

Es wurden jedoch nicht alle Protokolle im einheitlichen Protokoll gebündelt. Zu den verbleibenden traditionellen textbasierten Protokolldateien gehören:

- daily.out, monthly.out und wifi.log sind noch aktiv
- /var/log/install.log führt immer noch ein wertvolles Protokoll der Installation von Softwareupdates
- CUPS führt immer noch seine eigenen Protokolle in /var/log/cups aus
- Apps von Drittanbietern wie die von Adobe führen immer noch ihre eigenen Textprotokolle aus
- system.log existiert noch, wird nur noch von veralteter Software besucht, z.B. dem Google Software Update

Es gibt drei Hauptgruppen von Dateien, die Protokolleinträge speichern. Zum einen die in „/var/db/diagnostics/Persist/“ in Form von tracev3-Dateien, die reguläre Protokolleinträge enthalten. Zum anderen die in „/var/db/diagnostics/Special/“ in tracev3 Dateien, die zusätzliche Einträge mit kürzerer Lebensdauer enthalten. Außerdem speichert das Logging-System Daten zudem im Verzeichnis „/var/db/uuidtext“. Dieses enthält diverse Unterverzeichnisse mit Dateien, die mit GUIDs benannt sind und Log-Mitteilungen in Textform enthalten, welche von Unified Logging zur Darstellung von Log-Mitteilungen verwandt werden.

Log-Dateien des Unified Logging sind nach dem folgenden Muster abgespeichert: „logdata.Persistent.YYYYMMDDTHHMMSS.tracev3“. Das Unified-Logging-System nutzt die Hintergrundprozesse „logd“ und „diagnosticd“. Zur Laufzeit einer Anwendung verarbeitet „logd“ Mitteilungen, komprimiert sie und speichert sie in eine normale Log-Datei. Der Hintergrunddienst „diagnosticd“ nimmt Mitteilungen entgegen und erzeugt einen Live-Log-Stream. Die folgende Abbildung gibt eine Übersicht über die Arbeit von „logd“ und „diagnosticd“.

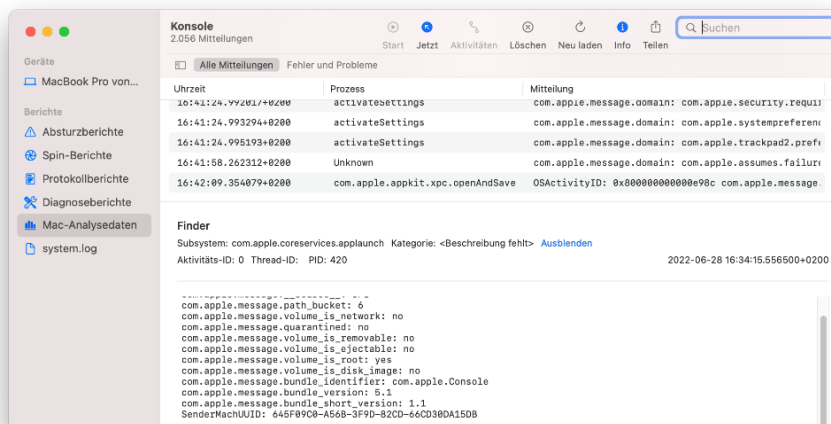


Log-Mitteilungen des Unified-Logging-Systems können durch die API (os_log) als Default-, Info- oder Debug-Mitteilungen definiert werden. Für jede Art kann definiert werden, ob ein Logging aktiviert ist und wo Log-Dateien gespeichert werden (Festplatte oder Hauptspeicher). Die folgende Tabelle zeigt die Standardeinstellungen für Mitteilungen.

Standardeinstellungen für Mitteilungen im Unified-Logging-System		
Mitteilungsart	Status	Speicherort
Debug-Level	nicht aktiviert	-
Info-Level	aktiviert	Hauptspeicher
Default-Level	immer	Festplatte
Error	immer	Festplatte
Fault	immer	Festplatte

Um die erstellten Log-Dateien oder den Live-Log-Stream betrachten zu können, kann die in Mac OS integrierte Konsole-App oder das Terminalkommando log eingesetzt werden. Zum Exportieren und Öffnen von Log-Mitteilungen auf weiteren Mac-OS-Systemen kann das neue Speicherformat .logarchive eingesetzt werden. Unified Logging ermöglicht eine Kategorisierung von Log-Mitteilungen sowie verbesserte Filtermöglichkeiten bei der Analyse.

In der Konsole-App werden unter Auswahl des Symbols des Mac-Computers die Mitteilungen des Unified Logging ausgewertet. Die Live-Sicht ermöglicht aktuelle Betriebssystem-Meldungen in Echtzeit zu verfolgen (Live Log Stream). Fehler (Fault und Error) werden mit einem gelben Punkt gekennzeichnet und können explizit gefiltert werden. Die Info erlaubt die Anzeige von erweiterten Informationen für eine geloggte Meldung (Unter anderem die Kategorisierung einer Mitteilung nach Subsystem und Kategorie). Mit Details kann die Info- Sicht nochmals erweitert werden.



Es kann auch das Terminalkommando log eingesetzt werden, um den Live-Log-Stream oder vom Unified-Logging-System erstellte Log-Dateien darzustellen. Das log-Kommando bietet dabei erweiterte Möglichkeiten zur Filterung und Suche. „log show“ zeigt den Inhalt von .tracev3 Log-Dateien oder .logarchive-Dateien an.

Syntax: log show	
Befehl	Bedeutung
\$ log show --archive Datei	Zeigt archivierte .logarchive-Dateien an.
\$ log show --file Datei	Zeigt .tracev3-Dateien an (die Datei muss sich in .logarchive oder in einem System-Log-Verzeichnis befinden).
\$ log show --predicate Filterausdruck	Ermöglicht die Angabe eines Filterausdrucks (predicate).
\$ log show --start Datum/Zeit --end Datum/Zeit	Grenzt Mitteilungen nach Start- und Endzeit zeitlich ein (akzeptiert sind die Formate „YYYY-MM-DD“, „YYYY-MM-DD HH:MM:SS“ oder „YYYY-MM-DD HH:MM:SSZZZZ“).

„log collect“ sammelt Informationen in einer Archivdatei (.logarchive) zur späteren Verwendung mit log oder der Konsole.

Syntax: log collect	
Befehl	Bedeutung
\$ log collect --output Pfad	Setzt den Ausgabepfad für die .logarchive-Datei.
\$ log collect --start Datum/Zeit	Grenzt Mitteilungen nach Anfangszeit ein (akzeptiert sind die Formate „YYYY-MM-DD“, „YYYY-MM-DD HH:MM:SS“ oder „YYYY-MM-DD HH:MM:SSZZZZ“).
\$ log collect --last Num [m h d]	Grenzt Mitteilungen zeitlich ein. Ausgehend von der aktuellen Zeit wird eine bestimmte Periode zurückgerechnet, z. B. last 2m (die letzten 2 Minuten) oder last 3h (die letzten 3 Stunden).

„log stream“ zeigt den Live-Log-Stream an und „log config“ konfiguriert das Unified-Logging-System.

```

Last login: Tue Jun 28 16:53:08 on ttys000
nutzer1@MBP-von-llcs ~ % log show --last 1s
Skipping info and debug messages, pass --info and/or --debug to include.
Timestamp      Thread      Type        Activity      PID  TTL
2022-06-28 16:33:03.887411+0200 0x11310      Default      0x0          253  30  softwareupdated: (SoftwareUpdateCoreSupport)
[com.apple.SoftwareUpdateMacController:SU] ...[FSM] API postEvent | SUCCESS
2022-06-28 16:33:03.887446+0200 0x11310      Default      0x0          253  30  softwareupdated: (SoftwareUpdateMacControlle
*) [com.apple.SoftwareUpdateMacController:SUMacController] [Download] SUCoreUpdate delegate method -[SUMacController updateAssetDown
loadProgress:] called
2022-06-28 16:33:03.887621+0200 0x11079      Default      0x0          253  30  softwareupdated: (SoftwareUpdateCoreSupport)
[com.apple.SoftwareUpdateMacController:SU] [FSM:update[UUID(E78967CA-A585-4E73-AD88-324FD43499A8) 28091(user)->286630((null)) incr]
]] >S> DownloadingSU >E> DownloadProgress >A> ReportDownloadProgress info:
[>>>
  targetPhase: NO_CHANGE
  policy: (null)
  downloadProgress: phase:Downloading stalled:NO portionComplete:0.935025 totalWrittenBytes:5373924236 totalExpectedBytes:574736125
6 estimatedTimeRemaining:556.431411
  prepareProgress: (null)
  resultCode: 0
  error: (null)
<<<]
2022-06-28 16:33:03.887906+0200 0x11310      Default      0x0          253  30  softwareupdated: (SoftwareUpdateMacControlle
*) [com.apple.SoftwareUpdateMacController:SUMacController] PerformAction with Action:DownloadProgress Event:DownloadProgress State:D
ownloadProgress NextState:(null) Info:[>>>
  BridgeOS(shouldPerformBridgeOSUpdate:NO|bridgeOSVersionToInstall:(null)|bridgeOSDownloadSizeBytes:(null)|bridgeOSDownloadDirecto
ry:(null))
  Rosetta(shouldPerformRosettaUpdate:NO|rosettaVersionToInstall:(null)|rosettaDownloadDirectory:(null))
  Targets(targetPhase:SUMAC_PHASE_NONE|eventToIssue:(null)|queryStateCompletion:NO)
  clientRequest: (null)
  progress: phase:Downloading stalled:NO portionComplete:0.935025 totalWrittenBytes:5373924236 totalExpectedBytes:5747361256 estim

```

Das Kommando log erlaubt es, sogenannte Predicate-Filter zu nutzen:

Wichtige Predicate-Filter	
Predicate-Filter	Bedeutung
eventMessage contains 'string'	Durchsucht Log-Mitteilungen nach einem bestimmten String. Der String muss in der Ausgabe der Mitteilung enthalten sein.
messageType == error	Sucht nach Mitteilungen des Typs Error.
processID == 100	Sucht nach Mitteilungen einer ProzessID, hier mit PID 100.
subsystem == "com.apple.TimeMachine"	Sucht nach einem konkreten Subsystem/App Bundle, im Beispiel Time Machine.

Predicate-Filter können durch logische Operatoren wie &&, and, or, >, <, !=, between, contains, like u. a. miteinander verknüpft werden.

Um die Log-Dateien mit log decodieren zu können, müssen .logarchive-Bundle-Dateien vorliegen oder sich die .traveV3-Dateien im Unified-Logging-Verzeichnis des laufenden Systems befinden. Zur Analyse bietet es sich daher entweder an, dass zu untersuchende System zu virtualisieren oder die Verzeichnisse „/var/db/diagnostics“ und „/var/db/uuidtext“ auf einen Analyse-Mac-Computer zu übernehmen. Das log-Kommando gibt seine Ausgaben grundsätzlich auf dem Bildschirm aus. Daher ist es sinnvoller, die Ausgabe in eine Textdatei umzuleiten.

```
$ touch ~/Desktop/meineLogDatei.txt
$ log show --last 5m --info > ~/Desktop/meineLogDatei.txt
```

Es existiert derzeit eine Alphaversion eines Offline Datei Parsers: <https://github.com/ydkhatri/UnifiedLogReader>.

```
G:\>c:\Python37-32\python.exe c:\Github\UnifiedLogReader\UnifiedLogReader.py -h
usage: UnifiedLogReader.py [-h] [-f OUTPUT_FORMAT] [-l LOG_LEVEL]
                          uuidtext_path timesync_path tracev3_path
                          output_path

UnifiedLogReader is a tool to read macOS Unified Logging tracev3 files.
This is version 0.3 tested on macOS 10.12.5 - 10.15 and iOS 12.
```

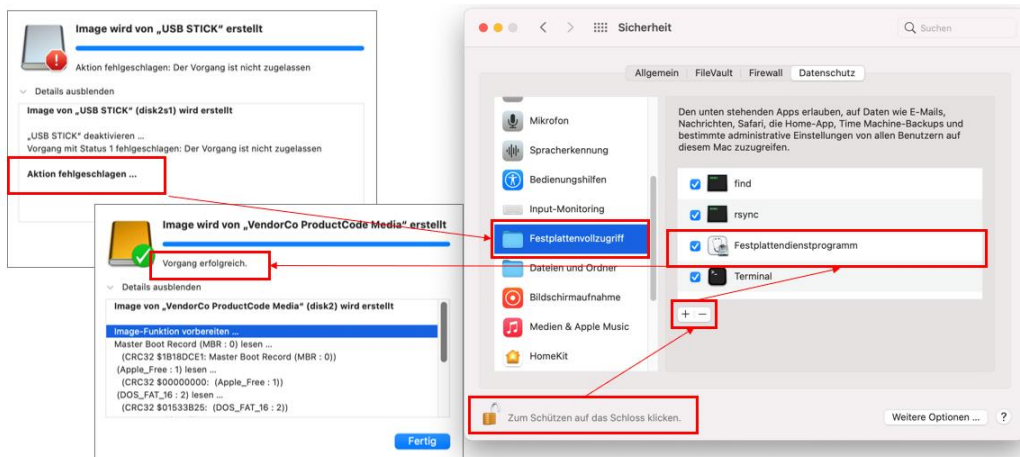
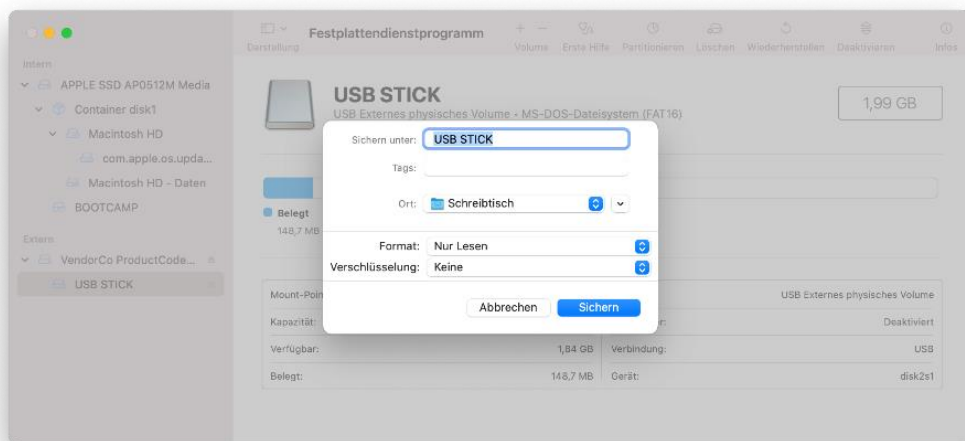
1.4.6 Mac Disk Images

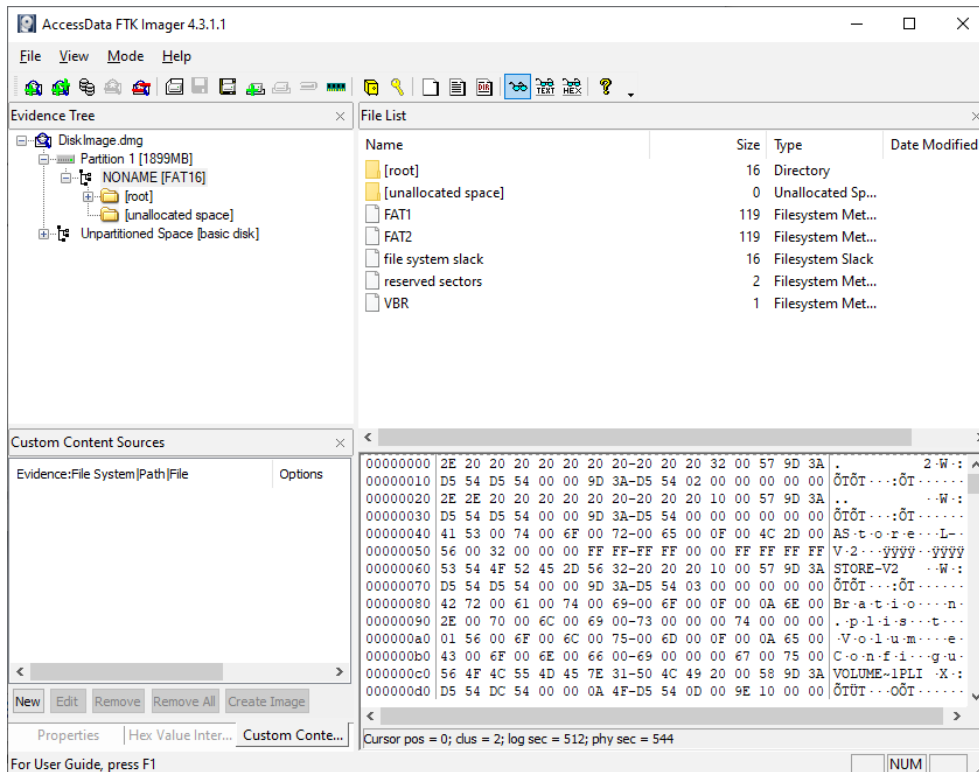
1.4.6.1 Mac Disk Image Formate

Mac OS kann mit verschiedenen Arten von Disk Images umgehen. Diese beinhalten vollständige Dateisysteme (unterstützt werden HFS+, FAT oder ExFAT) in einer einzigen Datei (.dmg), in einer mitwachsenden Datei (.sparseimage) oder in einem mitwachsenden Bundle (.sparsebundle).

DMG-Images haben im Gegensatz zu Sparse Disk Images eine feste Größe. Ein Sparse Disk Image kann seinen Inhalten entsprechend mitwachsen lassen. Ein Sparse Bundle wird für den Nutzer transparent in kleinen Einheiten gespeichert, so dass beispielsweise eine inkrementelle Sicherung von Teilen des Images möglich ist. Die angegebene Größe von gemounteten Sparse Disk Images oder Sparse Bundles kann aus diesem Grund wesentlich größer sein als der tatsächlich belegte Speicherplatz auf dem Datenträger.

1.4.6.2 Mac OS Disk Image erstellen





1.4.6.3 UDIF (Universal Disk Image Format)

Disk Images nutzen das von Apple entwickelte proprietäre Format UDIF (Universal Disk Image Format). Unter Mac OS können sie bei aktiviertem Disk Arbitration per Doppelklick im Finder gemountet werden. Der Befehl „hdiutil“ kann genutzt werden, um Informationen zu Disk Images zu erhalten bzw. um sie in das System einzubinden oder auszuhängen.

- \$ hdiutil info - zeigt Informationen zu gemounteten Disk Images
- \$ hdiutil attach /Desktop/Disk_Image.dmg - mountet ein Disk Image
- \$ hdiutil detach /Desktop/Disk_Image.dmg - entfernt ein Disk Image



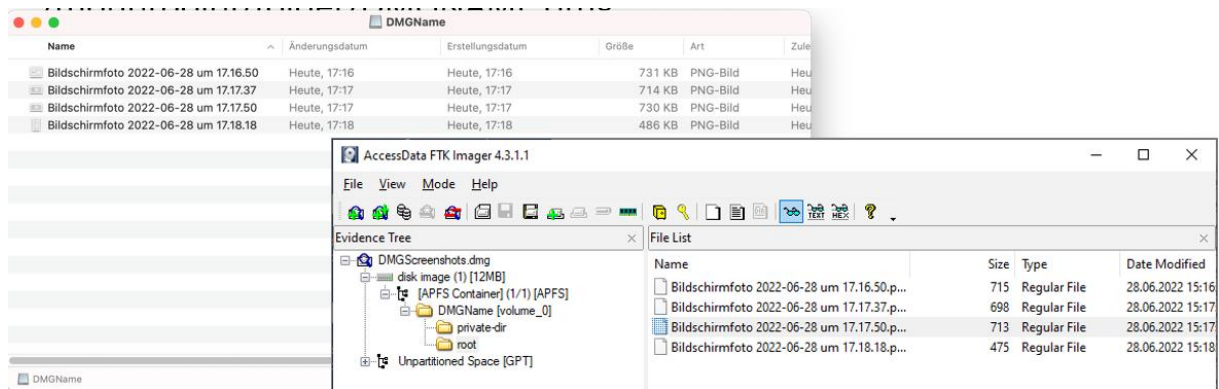
1.4.6.4 Forensische Abbilder von Mac-Computern

Disk Images können auch dazu genutzt werden, forensische Abbilder von Mac-Computern zu erstellen. Das DMG-Format wird insbesondere von Produkten, die unter Mac OS lauffähig sind, unterstützt. Es bietet den Vorteil, dass die Mac-eigene Technologie Spotlight benutzt werden kann, um Abbilder im DMG-Format zu indexieren und damit einfach und vor allem schnell durchsuchbar zu machen. Der Befehl „`hdiutil create -srcfolder /Users -volname DMGNAME /mountpoint/folder/DMGNAME.dmg`“ erzeugt eine DMG Datei mit APFS:



```
nutzer1 -- zsh -- 80x24
Last login: Tue Jun 28 17:07:09 on ttys000
nutzer1@MBP-von-llcs ~ % hdiutil create -srcfolder ./Desktop/Screenshots -volname DMGName /Volumes/USB\ STICK/DMGScreenshots.dmg
.....
created: /Volumes/USB STICK/DMGScreenshots.dmg
nutzer1@MBP-von-llcs ~ %
```





1.4.6.5 Schreibgeschütztes Mounten von DMG-Dateien

Falls ein forensisches Abbild im DMG-Format eingebunden werden soll, ist es sinnvoll, die Datei schreibgeschützt einzubinden, um Veränderungen auszuschließen. Das kann erreicht werden, indem man die DMG-Datei im Finder als geschützt markiert. Die Datei wird, wenn geschützt, mit einem kleinen Schloss-Icon markiert.

Damit Spotlight die Datei indexieren kann, wird der Befehl „hdiutil attach“ mit der Option „-shadow“ benutzt. Dieser Befehl erzeugt eine Shadow-Datei, in der alle von Spotlight durchgeführten Änderungen an der DMG-Datei gespeichert werden. Spotlight Indexing kann mit „mdutil -i on“ aktiviert werden. Der Befehl „mdutil -s“ überprüft, ob Spotlight Indexing aktiv oder nicht aktiv ist.

Neben dem schreibgeschützten Mounten von DMG-Dateien durch Setzen des Hakens „geschützt“ ist ein zweiter Weg möglich, der allerdings kein Spotlight Indexing unterstützt. Hierzu wird das Disk Image zunächst mit dem Befehl „hdiutil attach -nomount“ als Block-Device eingebunden und anschließend mit „mount_hfs -j -o rdonly, noexec, noowners“ schreibgeschützt gemountet.

1.4.6.7 Forensische Abbilder mounten

Die Möglichkeit, DMG-Dateien schreibgeschützt einzubinden, kann auch zum Mounten von forensischen Abbildern im RAW- oder E01-Format genutzt werden. Hierzu wird das forensische Abbild mit dem Befehl „xmount“ (alternativ „ewfmount“) in eine DMG-Datei konvertiert und anschließend, wie zuvor beschrieben, schreibgeschützt eingebunden werden.

RAW-Abbilder können neben dem beschriebenen Weg auch direkt in Mac OS eingebunden werden: „\$ hdiutil attach -nomount -imagekey diskimageclass=CRawDiskImage [Pfad zum RAW Abbild]“. Alternativ können forensische Abbilder unter Mac OS auch mit den grafischen Tools EWMounter von Blacklight oder mit Sumuri Recon in das System eingebunden und anschließend untersucht werden.

1.4.7 Time Machine und lokale Backups

1.4.7.1 Time Machine

Seit der Mac-OS-X-Version 10.5 ist Time Machine die Apple-eigene Technologie zur Erstellung von Backups. Bis macOS 11 BigSure wurden Time Machine Backups auf ein virtuelles HFS+-Dateisystem auf einem Sparse-Bundle geschrieben. Die Verschlüsselung erfolgt gegebenenfalls im Sparseimage. macOS 11 Big Sur ist die erste Version von macOS, die Time Machine-Backups auf APFS-Volumes erstellt, ohne ein virtuelles HFS+-Dateisystem auf einem Sparse-Bundle zu verwenden. Hier erfolgt die Verschlüsselung im APFS Volume.

1.4.7.2 Time Machine Backups im Sparsebundle Format

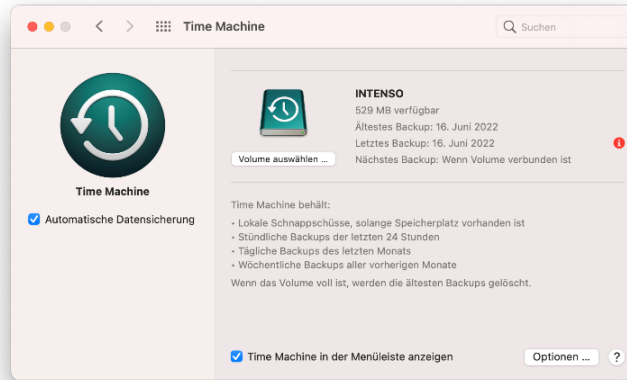
Time Machine erstellt ein virtuelles HFS+-Dateisystem auf einem Sparse-Bundle. Darin wird in einen Ordner eine Kopie des Dateisystems des zu sichernden Volumes abgelegt. Dateien und Ordner, die sich seit der letzten Sicherung nicht geändert haben, werden dort durch feste Links dargestellt. Überprüft man die Inode-Nummer von zwei entsprechenden „Kopien“ eines Ordners, der sich zwischen den Sicherungen nicht geändert hat, sieht man identischen Inodes, genau wie bei normalen Datei-Hardlinks.

1.4.7.3 Time Machine Backups auf APFS Datenträgern

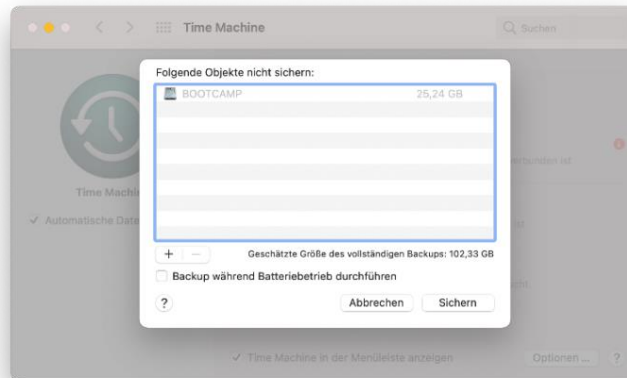
APFS unterstützt keine Verzeichnis-Hardlinks und kann daher beim Speichern von Time Machine-Backups nicht denselben Mechanismus verwenden. APFS verwendet die aus dem Dateisystem stammenden Snapshots. Das Ziellaufwerk muss somit APFS formatiert werden. Snapshots werden innerhalb des Dateisystems mit dem Copy on Write Prinzip erstellt. Ist das Ziellaufwerk nicht verschlüsselt, so ist auch das Backup unverschlüsselt. Theoretisch ist mit T2 und M1 eine peer File Verschlüsselung im Backup möglich.

1.4.7.4 Time Machine Sicherungen

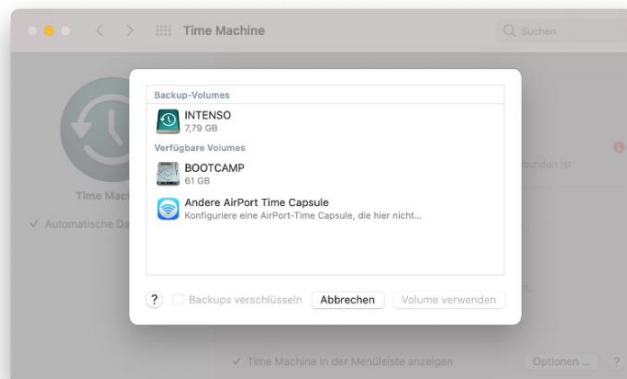
Time Machine Sicherungen werden nach einem bestimmten Zeitschema durchgeführt. Es werden Local Snapshots (Offline) angefertigt, wenn das Time-Machine-Volume nicht präsent ist. Stündliche Backups werden nach 24 Stunden verworfen. Die täglichen Backups hingegen werden nach einem Monat verworfen. Wöchentliche Backups werden nicht verworfen (Ende, wenn das Time-Machine-Volume voll ist). Die folgende Abbildung zeigt die Ansicht der letzten Backups in der Anwendung.



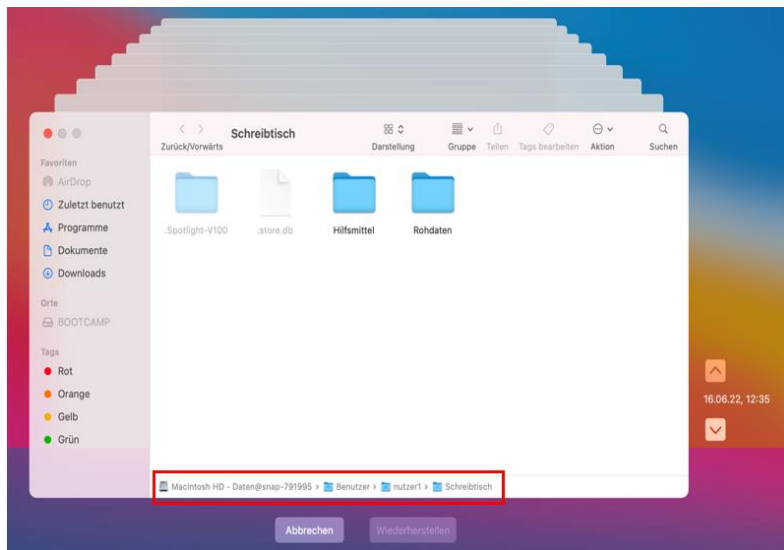
Die nächste Abbildung zeigt von der Sicherung ausgenommene Objektanzeigen in der Anwendung.



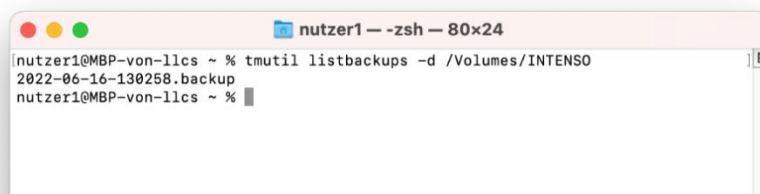
Die Abbildung zeigt die Anzeige des Backuplaufwerks in der Anwendung.



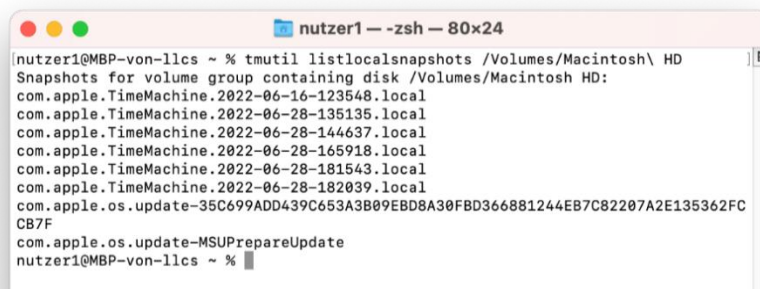
Vom Finder aufgerufen erhält man die Ansicht der jeweiligen backupstände des geöffneten Verzeichnisses.



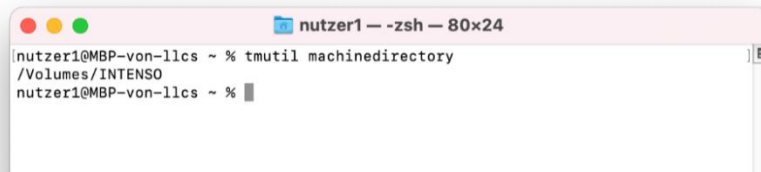
Der Befehl „tmutil listbackups -d [/mountpoint/BackupLaufwerk]“ gibt die Backups im Backuplaufwerk mit Datum aus.



Der Befehl „tmutil listlocalsnapshots [/mountpoint]“ gibt die gespeicherten lokalen Backups aus.



Der Befehl „tmutil machinedirectory“ gibt den Mountpoint der Backups aus.



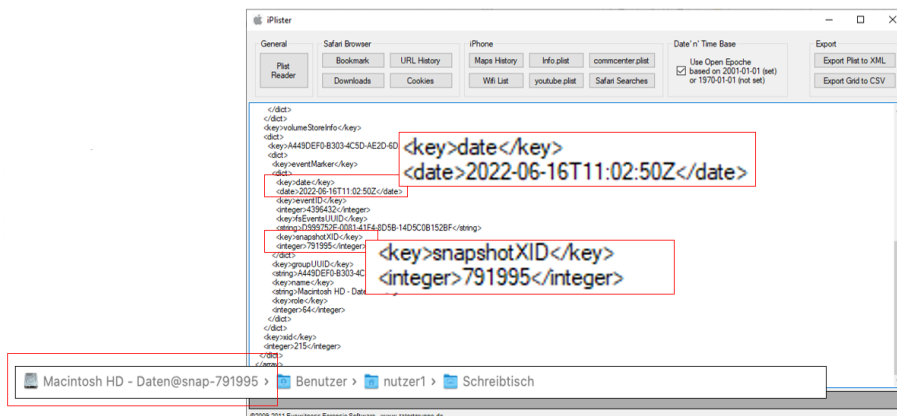
Der Befehl „tmutil destinationinfo“ gibt den Speicherort der Backups aus.



Das Backup Laufwerk enthält die einzelnen Backup Stände als Verzeichnisstruktur:

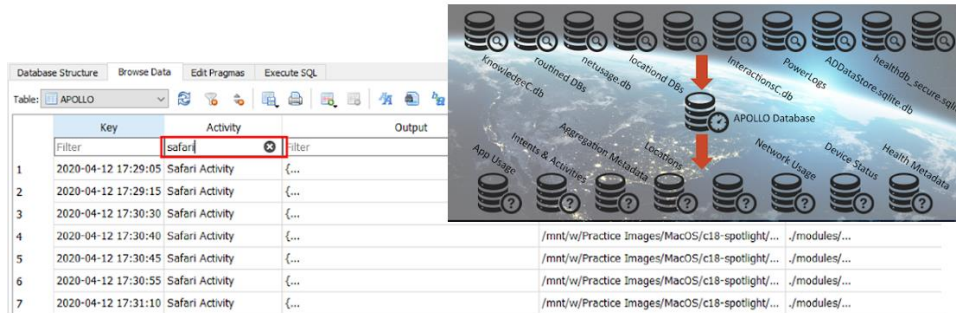
Name	Pfad	Beschreibung	Erw.	Typ	Größe	Erzeugung
.		(Stammverzeichnis)	existierend		5,9 GB	
INTENSO (176.317)			existierend		5,5 GB	16.06.2022 10:35:02
2022-06-21-071648.inprogress (3)	\INTENSO	existierend	inprogress		253 KB	21.06.2022 05:16:49
2022-06-16-183911.interrupted (5)	\INTENSO	existierend	interrupted		196 KB	16.06.2022 16:39:12
2022-06-16-182156.interrupted (160.452)	\INTENSO	existierend	interrupted		3,7 GB	16.06.2022 16:21:57
2022-06-16-174012.interrupted (4.721)	\INTENSO	existierend	interrupted		524 MB	16.06.2022 15:40:13
2022-06-16-130258.previous (11.050)	\INTENSO	existierend	previous		1,1 GB	16.06.2022 10:35:49
Spotlight-V100 (80)	\INTENSO	existierend			79,5 MB	16.06.2022 10:35:02
private-dir (0)	\INTENSO	existierend			0 B	16.06.2022 10:35:02
backup_manifest.plist (1)	\INTENSO	existierend	plist	plist	0,6 KB	16.06.2022 11:03:19
DS_Store (1)	\INTENSO	existierend			6,0 KB	16.06.2022 10:35:16
com.apple.fs.cow-exempt-file-count	\INTENSO	existierend	cow-exempt-file-count	cow-exe...	12 B	
Catalog	\INTENSO	virtuell (für Untersuchungszwecke)			76,0 MB	

Die Datei „backup_manifest.plist“ auf dem BACKUP Laufwerk enthält die letzte Snapshot-ID und das Datum.



1.4.8 Weitere Artefaktanalysen

Der Apple Pattern of Life Lazy Output'er (APOLLO) kann verwendet werden. Er ist unter <https://github.com/mac4n6/APOLLO> zu finden.



Filter	Key	Activity	Output
	2020-04-12 17:29:05	Safari Activity	{...}
	2020-04-12 17:29:15	Safari Activity	{...}
	2020-04-12 17:30:30	Safari Activity	{...}
	2020-04-12 17:30:40	Safari Activity	{...}
	2020-04-12 17:30:45	Safari Activity	{...}
	2020-04-12 17:30:55	Safari Activity	{...}
	2020-04-12 17:31:10	Safari Activity	{...}

The diagram on the right illustrates the APOLLO database architecture, showing various data sources like Knowledge-Cdbs, Location DBs, APOLLO Database, and others, all feeding into a central APOLLO Database.

Weiterhin kann das mac_apt – macOS (and iOS) Artifact Parsing Tool verwendet werden. Es ist unter https://github.com/ydkhatri/mac_apt zu finden.

```
Usage: mac_apt_artifact_only.exe [-h] [-i INPUT_PATH [INPUT_PATH ...]] [-o OUTPUT_PATH] [-x] [-c] [-t]
                                [-l LOG_LEVEL] [--plugin_help]
                                plugin

mac_apt is a framework to process macOS forensic artifacts
You are running macOS Artifact Parsing Tool - Artifact Only mode version 1.5.0.dev (20220614)

Note: The default output is now sqlite, no need to specify it now

positional arguments:
  plugin                Plugin to run

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT_PATH [INPUT_PATH ...], --input_path INPUT_PATH [INPUT_PATH ...]
                        Path to input file(s)
  -o OUTPUT_PATH, --output_path OUTPUT_PATH
                        Path where output files will be created
  -x, --xlsx            Save output in excel spreadsheet(s)
  -c, --csv             Save output as CSV files
  -t, --tsv            Save output as TSV files (tab separated)
  -l LOG_LEVEL, --log_level LOG_LEVEL
                        Log levels: INFO, DEBUG, WARNING, ERROR, CRITICAL (Default is INFO)
  --plugin_help        Plugin usage info
```

1.5 macOS Apps

1.5.1 Kommunikations-Apps

1.5.1.1 Programme (Apps)

Unter Mac OS liegen Programme (Apps) als Bundles vor. Die Bundle-Dateien sind entweder systemweit unter „/Applications“ oder im Nutzerkontext unter „~/Applications“ gespeichert. Zur Ausführung und Speicherung von Programmdateien sowie zur Speicherung von Konfigurationseinstellungen und Caches nutzen sie jedoch Verzeichnisse im Nutzerkontext. Die von Programmen genutzten Verzeichnisse sind:

Pfad	Beschreibung
~/Library/Application Support/<App>	Ausführung & Programmdateien von Apps
~/Library/Containers/<Bundle ID>	Ausführung & Programmdateien von Apps, die Sandboxing nutzen
~/Library/<App>	Programmdateien & Konfigurationseinstellungen von Apps
~/Library/Preferences/	Konfigurationseinstellungen von Apps
~/Library/Caches/	Cache-Inhalte von Apps

Es wird zwischen nativen Apps, die fest zum Betriebssystem Mac OS gehören und standardmäßig installiert sind, und Apps von Drittanbietern unterschieden. In Mac OS integrierte Apps sind aus forensischer Sicht zum Großteil gut erforscht und werden von den meisten forensischen Analyseprodukten automatisiert ausgewertet. Anspruchsvoller gestaltet sich hingegen die Analyse von Drittanbieter-Apps.

1.5.1.2 Kontakte

Das in Mac OS integrierte Adressbuch wurde mit Version 10.8 in Kontakte umbenannt, auf Dateisystemebene wird allerdings weiterhin die Bezeichnung Addressbook verwendet. Kontakte können im Verzeichnis „Metadata“ als Plist-Dateien mit der Endung .abcdp aufgefunden werden. Zu den Kontakteinträgen assoziierte Bilddateien sind im Verzeichnis „Images“ aufzufinden und über die GUIDs in den Dateinamen können sie den entsprechenden Kontakteinträgen in „Metadata“ zugeordnet werden.

Kumulativ speichert Mac OS Kontaktdaten in zwei SQLite-Datenbanken. Die Datenbank AddressBook-v22.abcd.db enthält Adressbucheinträge (Tabelle ZABCDRECORD). Die Datenbank MailRecents-v4.abcd.mr enthält, falls vorhanden, die E-Mail-Adressen der Kontakte (Tabelle ZABCDMAILRECENT). Konfigurationseinstellungen zur Kontakte-App enthält die Datei „com.apple.AddressBook.plist“.

1.5.1.3 Mail

Mac OS unterstützt das Empfangen und Versenden von E-Mails mit der integrierten Mail-App. Diese erfuh mit den vergangenen Mac-OS-X-Versionen jeweils eigene Versionssprünge mit veränderten Pfaden zu Konfigurationseinstellungen und E-Mail-Konten:

Mac-OS-Version	Mail-Version	Pfad
OS X 10.8	Mail-Version 6	~/Library/Mail
OS X 10.9	Mail-Version 7	~/Library/Mail/V2
OS X 10.10	Mail-Version 8	~/Library/Mail/V2
OS X 10.11	Mail-Version 9	~/Library/Mail/V3
macOS 10.12	Mail-Version 10	~/Library/Mail/V4
macOS 10.13	Mail-Version 11	~/Library/Mail/V5
macOS 10.14	Mail-Version 12	~/Library/Mail/V6
macOS 10.15	Mail-Version 13	~/Library/Mail/V7
macOS 11	Mail-Version 14	~/Library/Mail/V8
macOS 12	Mail-Version 15	~/Library/Mail/V9
macOS 13 + 14	Mail-Version 16	~/Library/Mail/V10

Mail Version 6 bis 8

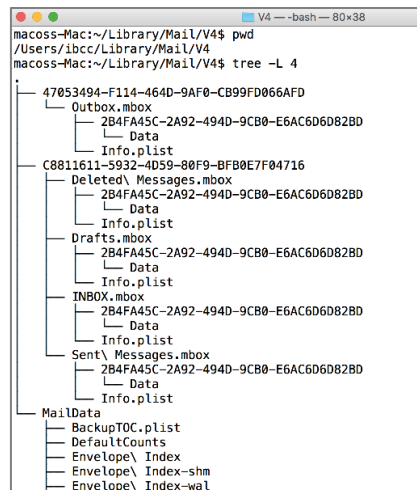
Von Version 6 zu 8 haben sich geringfügige Änderungen bezüglich der Speicherstruktur ergeben, die Systematik ist jedoch gleichgeblieben. Einstellungen für die E-Mail-Konten wie Account-Namen, Konten-Typ und Hostnamen sind in der Datei „Accounts.plist“ verzeichnet. Innerhalb des Verzeichnisses V2 sind E-Mail-Konten als Unterverzeichnisse eingebunden. Sie sind mit dem Kontentyp und der E-Mail-Adresse bezeichnet und haben eine Verzeichnisstruktur mit Posteingang, Postausgang etc. Innerhalb der Verzeichnisstruktur folgt ein GUID-Verzeichnis mit den Verzeichnissen „Data“, „Messages“ und „Attachments“. E-Mails sind innerhalb dieser Ordnerstruktur als .emlx-Dateien abgelegt. Das Verzeichnis „Messages“ enthält die .emlx-Dateien. „Attachments“ enthält entsprechende Anhänge.

Mail Version 9

In Version 9 der Mail-Applikation und damit Mac OS X 10.11 hat Apple weitergehende Änderungen am Speicherverhalten des Programms vorgenommen. Die Konfigurationsdatei „Accounts.plist“ ist nicht mehr vorhanden. Informationen zu E-Mail-Konten sind in der SQLite-Datenbank Accounts3.sqlite, welche Informationen und Einstellungen zu eingerichteten Internet-Accounts enthält. Die Speicherung der E-Mails als .emlx-Dateien im Ordner „Messages“ sowie die Speicherung der Anhänge in „Attachments“ bleibt wie in den Vorgängerversionen gleich.

Ab Mail Version 10

ab macOS 10.12 liegt Mail in Version 10+ vor und speichert seine Inhalte im Verzeichnis V4+. Die Struktur ist analog zum Vorgänger V3. Die Datenbank zur Speicherung von Konten heißt jetzt Accounts4.sqlite.



```
macos-Mac:~/Library/Mail/V4$ pwd
/Users/ibcc/Library/Mail/V4
macos-Mac:~/Library/Mail/V4$ tree -L 4
.
├── 47053494-F114-464D-9AF0-CB99FD066AFD
│   ├── Outbox.mbox
│   │   ├── 2B4FA45C-2A92-494D-9CB0-E6AC606082BD
│   │   │   └── Data
│   │   └── Info.plist
│   └── C8011611-5932-4D59-80F9-BFB0E7F04716
│       ├── Deleted Messages.mbox
│       │   ├── 2B4FA45C-2A92-494D-9CB0-E6AC606082BD
│       │   │   ├── Data
│       │   │   └── Info.plist
│       ├── Drafts.mbox
│       │   ├── 2B4FA45C-2A92-494D-9CB0-E6AC606082BD
│       │   │   ├── Data
│       │   │   └── Info.plist
│       ├── INBOX.mbox
│       │   ├── 2B4FA45C-2A92-494D-9CB0-E6AC606082BD
│       │   │   ├── Data
│       │   │   └── Info.plist
│       └── Sent Messages.mbox
│           ├── 2B4FA45C-2A92-494D-9CB0-E6AC606082BD
│           │   ├── Data
│           │   └── Info.plist
└── MailData
    ├── BackupTOC.plist
    ├── DefaultCounts
    ├── Envelope\ Index
    ├── Envelope\ Index-shm
    └── Envelope\ Index-wal
```

Mail Version alle Versionen

Unter allen Mail-Versionen speichert die SQLite-Datenbank Envelope Index umfangreiche Daten. Dazu zählen die E-Mails (Tabelle addresses), der Zeitstempel in UNIX-Zeit (Tabelle messages), die E-Mail-Adressen und weitere Metadaten zur Mail-Applikation. Envelope Index enthält einen Index aller E-Mail-Dateien, um diese durchsuchbar zumachen.

1.5.1.4 Nachrichten App

Die Nachrichten App ist die in Mac OS integrierte Chat-Applikation. Sie unterstützt eine Vielzahl von Instant-Messaging-Protokollen wie iCloud (iChat), AOL (AIM), Google Talk (Jabber) und Yahoo Chat. Weiterhin hat sie umfangreiche Funktionen wie Peer-to-Peer-File-Sharing oder das Teilen von Bildschirmhalten. Die Integration von FaceTime und iOS ermöglicht zudem Videotelefonie und das Versenden von iMessages oder SMS-Nachrichten über korrespondierende iOS-Devices. Die Nachrichten App besitzt eine Vielzahl von Konfigurationsdateien, die Einstellungen und Account-Informationen zu den verschiedenen Funktionalitäten beinhalten.

Chat-Nachrichten, Zeitstempel, Chat-Teilnehmer und Metadaten zu versandten bzw. empfangenen Dateien befinden sich in der SQLite-Datenbank chat.db. Von besonderem Interesse sind dabei die Tabellen chat, welche Chat-Kontakte und Informationen zu den Chat-Protokollen enthält, die Tabelle messages, welche mit einer UID versehene Chat-Nachrichten beinhaltet, die Tabelle handle, die kürzlich empfangene oder versandte Chats beinhaltet und die Tabelle attachments, welche Metadaten zu empfangenen und versandten Dateien enthält.

Das Verzeichnis „Attachments“ enthält empfangene bzw. versandte Dateien. Diese können anhand der Nachrichten-UID zugeordnet werden. Durchgeführte Konversationen speichert das Programm im Verzeichnis „Archive“ ab. Sie liegen zeitlich sortiert als *.ichat-Dateien vor und bestehen aus binären Plist-Dateien.

1.5.1.5 FaceTime App

FaceTime ist die Mac-OS-eigene Video-Chat-Applikation, welche auch von der mobilen Variante iOS unterstützt wird. Für die Nutzung wird ein iCloud-Account benötigt. FaceTime interagiert mit der Nachrichten-App und kann synchron mit iOS-Geräten betrieben werden. Die Einstellungen findet man in der Datei „com.apple.ids.service.com.apple.madrid.plist“. Weitere interessante Einstellungen wie beispielsweise kürzlich getätigte Anrufe (Anrufliste) sind in der Datei „com.apple.imservice.ids.FaceTime.[GUID].plist“ abgelegt.

1.5.2 Browser Artefakte

1.5.2.1 Safari Browser

Der Safari Browser speichert programmspezifische Einstellungen, wie das gewählte Download-Verzeichnis und mit Safari durchgeführte Suchen im Web in der Plist-Datei „com.apple.Safari.plist“. Mit aktivierter SIP ist die Datei möglicherweise nicht zu finden. Hier liegt diese dann unter „/Users/\$USER/Library/Containers/com.apple.Safari/Data/Library/Preferences/com.apple.Safari.plist“.

im Verzeichnis „~/Library/Safari/“ befinden sich weitere Plist-Dateien und SQLite-Datenbanken, die interessante Informationen zu Safari beinhalten. Die Datei „Bookmarks.plist“ speichert vom Nutzer angelegte Lesezeichen und Favoriten, während die Datei „Downloads.plist“ eine nutzerspezifische Historie von heruntergeladenen Dateien darstellt und Informationen zu Downloads speichert. Dazu zählen die Quell-URL, die Größe des heruntergeladenen Objekts in Bytes, das Ziel-Verzeichnis und die Startzeit und Endzeit des Downloads. Der Verlauf der besuchten Webseiten ist in der SQLite-Datenbank History.db gespeichert. Die Datei ist ab der Mac-OS-X-Version 10.10 vorhanden, zuvor wurde die gleichnamige Plist-Datei „History.plist“ genutzt. Die Tabellen „history_items“ und „history_visits“ enthalten die besuchten Webseiten und korrespondierende Zeitstempel in Mac Epoch Time (definiert als Anzahl der Sekunden seit 01/01/2001 00:00:00, auch CF Absolute Time). Die Dateien „LastSession.plist“ und „TopSite.plist“ enthalten die bei der letzten Sitzung geöffneten Webseiten und die am häufigsten besuchten zwölf Webseiten (Topsites) des Nutzers. Cookies speichert Safari seit Mac OS X 10.7 in der Datei „Cookies.binarycookies“ (in älteren Mac-OS-X-Versionen in der Datei Cookies.plist) in einem proprietären Format.

Safari speichert besuchte Webseiteninhalte zudem in der SQLite-Datenbank Cache.db im Verzeichnis „~/Library/Caches/com.apple.Safari.../“. In der Tabelle „cfurl_cache_response“ befinden sich Metadaten zu den gespeicherten Dateien. Die Tabelle „cfurl_cache_receiver_data“ enthält eingebettet die Binärdaten der Dateien.

1.5.2.2 Google Chrome und Firefox Browser

Von Google Chrome werden Datenbank Dateien und Konfigurationen im Verzeichnis „~/Library/Google/“ abgelegt. Von Mozilla Firefox werden Datenbank Dateien und Konfigurationen im Verzeichnis „~/Library/Mozilla/“ abgelegt. Der Aufbau dieser Verzeichnisse ist plattformübergreifend standardisiert und auf Windows, Linux und macOS weitestgehend identisch. Es können bereits im Einsatz befindliche Tools genutzt werden die Daten zu analysieren.

Von Google Chrome werden unter anderem Cache Dateien im Verzeichnis „~/Library/Caches/Google/“ abgelegt. Von Mozilla Firefox werden unter anderem Cache Dateien im Verzeichnis „~/Library/Caches/Mozilla/“ abgelegt“. Der Aufbau dieser Verzeichnisse ist ebenfalls plattformübergreifend standardisiert.

1.5.3 Cloud

1.5.3.1 iCloud

iCloud ist eine Cloud-Computing-Anwendung der Firma Apple, die es ermöglicht, Daten auf allen Apple-Geräten mit dem Betriebssystem Mac OS und iOS synchron zu halten und von überall auf sie zuzugreifen (Ubiquity). Möglich ist auch ein Zugriff zum Onlinedienst über eine Weboberfläche.



iCloud kann bis zu zehn Apple-Geräte synchronisieren. Die Nutzer erhalten einen kostenlosen Speicherplatz von 5 GB, der bei Bedarf kostenpflichtig erweitert werden kann. Der Dienst iCloud kann über die Betriebssysteme Mac OS X (ab Version 10.7) und iOS (ab Version 5) genutzt werden. iCloud-Daten werden in den Serversystemen von Apple gespeichert. Für den iCloud-Account wird eine eindeutige persönliche ID benutzt, welche mit mehreren E-Mail- Adressen bzw. Telefonnummern verknüpft sein kann.

Im Folgenden werden einige Beispiele für iCloud-Funktionalitäten aufgelistet:

- Teilen von iCloud-Fotos durch Foto-Freigaben
- Orten, Sperren und Fernlöschen von iOS- und Mac-OS-Systemen
- Speicherung von Inhaltsdaten in der iCloud (bspw. Spielstände etc.)
- Synchronisierung von gekauften Inhalten (Apps) und Safari-Lesezeichen + Tabs
- Speicherung und Synchronisierung von Kennwörtern (Schlüsselbund)
- Backup-Funktionalität für iOS-Geräte
- Synchronisierung von nativen Apps auf allen Geräten (bspw. Mail, Notizen, Kontakte, Kalender, Notizen etc.)
- iCloud Drive ermöglicht den Zugriff auf eine Verzeichnisstruktur zum Speichern und Verwalten von Dateien
- Familienfreigabe ermöglicht das Verknüpfen von iCloud-Accounts untereinander

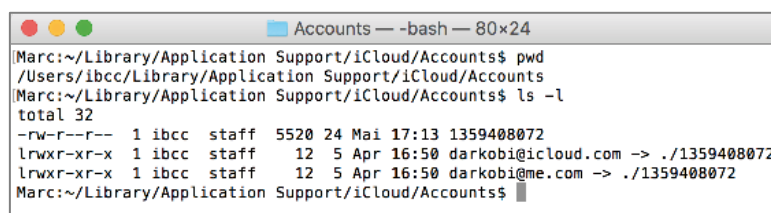
Der Zugang zum Onlinedienst iCloud erfolgt durch Eingabe der persönlichen iCloud-ID (alternativ der iCloud-E-Mail-Adresse) und eines vom Nutzer gewählten Passworts (mindestens 8 Zeichen, mindestens 1 Ziffer, 1 Kleinbuchstabe und 1 Großbuchstabe). Optional zur Authentifizierung mit ID/E-Mail und Passwort kann eine Zwei-Faktor-Authentifizierung aktiviert werden. Bei aktivierter Zwei-Faktor-Authentifizierung kann nur von vertrauenswürdigen Geräten auf den Onlinedienst iCloud zugegriffen werden. Als vertrauenswürdige Geräte können iOS-Geräte oder Mac-OS-Systeme definiert werden, die sich in Besitz des Nutzers befinden.

Sämtliche Daten werden von iCloud verschlüsselt übertragen und auf den Apple-Subsystemen verschlüsselt abgelegt. Als Verschlüsselungsverfahren wird mindestens 128-Bit-AES benutzt. Bestimmte Inhalte, wie der Schlüsselbund, werden 256-Bit-AES verschlüsselt.

Zur Sicherung der Nutzererkennung nutzt iCloud sogenannte sichere Tokens. Die Verwendung von Tokens ermöglicht eine Nutzung von iCloud-Funktionen, ohne dass ein Nutzer bei jeder Inanspruchnahme das Kennwort erneut eingeben muss. Die Authentifizierung erfolgt über das sichere Token. Weiterhin können Drittanbieter-Apps iCloud-Funktionalitäten nutzen, indem sie ein sicheres Token ausgestellt bekommen und damit einen Zugriff erhalten. Damit ist es nicht erforderlich, dass Drittanbieter-Apps die iCloud-Zugangskennung haben, um den Onlinedienst zu nutzen. IT-Forensiker können sichere iCloud-Tokens nutzen, um auch ohne Zugangskennung eines Nutzers einen Zugriff auf iCloud-Daten zu erhalten. Beispielsweise ermöglicht die Software Elcomsoft Phone Breaker das Auslesen von iCloud-Tokens aus laufenden Mac-OS-Systemen oder im Rahmen einer Post-Mortem-Analyse, sofern das Keychain-Passwort bekannt ist. Nach erfolgreicher Extraktion des Tokens kann die Software auf iCloud-Inhalte zugreifen und diese sichern.

1.5.3.2 iCloud-Spuren unter Mac OS

Die persönliche iCloud-ID sowie die verknüpften E-Mail-Adressen lassen sich im Verzeichnis „~/Library/Application Support/iCloud/Accounts“ einsehen. Die folgende Abbildung zeigt die Anzeige der iCloud-ID und der verknüpften E-Mail-Adressen.



```
Accounts — -bash — 80x24
Marc:~/Library/Application Support/iCloud/Accounts$ pwd
/Users/ibcc/Library/Application Support/iCloud/Accounts
Marc:~/Library/Application Support/iCloud/Accounts$ ls -l
total 32
-rw-r--r--  1 ibcc  staff   5520  24 Mai 17:13 1359408072
lrwxr-xr-x  1 ibcc  staff    12  5 Apr 16:50 darkobi@icloud.com -> ./1359408072
lrwxr-xr-x  1 ibcc  staff    12  5 Apr 16:50 darkobi@me.com -> ./1359408072
Marc:~/Library/Application Support/iCloud/Accounts$
```

Der Onlinespeicher iCloud kann von diversen Applikationen zum Abspeichern von Dateien genutzt werden. Die Funktion iCloud Drive lässt darüber hinaus einen Zugriff auf die iCloud-Verzeichnisstruktur zu, in der die Nutzer manuell Dateien abspeichern können. Gespeicherte Dateien werden mit dem verbundenen Mac-OS-System synchronisiert und werden innerhalb der Nutzerdomäne im Verzeichnis „~/Library/Mobile Documents/“ abgespeichert.

Das Verzeichnis „/Mobile Documents“ enthält als erweitertes Metadatum die persönliche ID des verbundenen iCloud-Accounts. Die folgende Abbildung zeigt die Anzeige der erweiterten Metadaten des Verzeichnisses „/Mobile Documents“.

```

Library -- -bash -- 103x32
Marc:~/Library$ xattr -xl Mobile\ Documents/
com.apple.ubd.prsid:
00000000 31 33 35 39 34 30 38 30 37 32 2E 43 6C 6F 75 64 |1359408072.Cloud|
00000010 44 6F 63 73 |Docs|
00000014
Marc:~/Library$ █

```

„/Mobile Documents/“ wird bezüglich der Verzeichnisstruktur der mit iCloud synchronisierten Dateien nach zugehörigen Applikationen kategorisiert. Verzeichnisse der Applikationen sind mit dem Applikationsnamen im Reversed-DNS-Format bezeichnet. Drittanbieter-Apps sind mit einer ID benannt. In iCloud Drive gespeicherte Dateien befinden sich in „com~apple~CloudDocs“. Die folgende Abbildung zeigt die Einteilung der iCloud-Dateien im Verzeichnis „/Mobile Documents“.

```

Mobile Documents -- -bash -- 103x32
Marc:~/Library/Mobile Documents$ pwd
/Users/ibcc/Library/Mobile Documents
Marc:~/Library/Mobile Documents$ tree -L 3
.
├── 28584H69WZ~com~2k~xcomew~SaveDataWithiCloud
│   └── Documents
├── 356QY8883U~com~melodis~soundhound~free
│   └── Documents
├── 3L68KQB4HG~com~readdle~CommonDocuments
│   └── Documents
├── 4268WDS42~com~BuhlData~Finanzblick
│   └── Documents
├── 4R6749AYRE~com~pixelmatorteam~pixelmator
│   └── Documents
├── 57T9237FN3~net~whatsapp~WhatsApp
│   └── Documents
└── 5UAR78B6YU~com~goodnotesapp~goodnotes
    └── Documents
        ├── BEDIS\ Netzwerk.folder
        ├── Mac\ -\ Modul\ B\ Ermittler.folder
        ├── Mac\ Forensic\ 1.folder
        └── Mac\ Forensic\ 2.folder

```

Die globalen Einstellungen zu iCloud enthält die Datei „~/Library/Synced Preferences/com.apple.syncedpreferences.plist“. Einstellungen zu den Synchronisierungseigenschaften einzelner Apps und iCloud befinden sich in eigenen Plist-Dateien, welche sich entweder im Verzeichnis „~/Library/SyncedPreferences“ oder bei Applikationen, die in einer gesicherten Sandbox-Umgebung ausgeführt werden, unter dem Pfad „~/Library/Containers/[APP]/Data/Library/SyncedPreferences“ befinden.

1.5.3.3 iCloud-Logdateien

Details zu mit iCloud synchronisierten Dateien befinden sich im Verzeichnis „~/Library/Application Support/CloudDocs/“. Die Datei „account.1“ enthält die persönliche iCloud-ID des verknüpften iCloud-Accounts. Innerhalb des Verzeichnisses existiert eine weiterverzweigte Struktur: Die unter „/sessions/containers/“ vorhandenen Plist-Dateien enthalten Konfigurationseinstellungen zu mit iCloud synchronisierten Applikationen und die unter „/sessions/db/“ enthalten die SQLite-Datenbanken „client.db“ und „server.db“ enthalten Details zu synchronisierten Dokumenten.

1.5.3.4 iCloud-Daten sichern

Technisch kann eine Sicherung von iCloud-Inhalten entweder durch mit iCloud synchronisierte MacOS- oder iOS-Geräte erfolgen. Ab Mac OS 10.5 sind Inhalte im Rahmen einer Post-Mortem-Analyse auffindbar. Bei iOS-Devices ist eine Sicherung nur bei jailbroken Geräten möglich.

Eine weitere Möglichkeit ist die direkte Sicherung von iCloud-Inhalten auf iCloud.com. Zur direkten Sicherung aus dem Apple-Cloud-Speicher können beispielsweise die Programme iLoot, Elcomsoft Phone Breaker oder Passware Forensic-Toolkit genutzt werden.

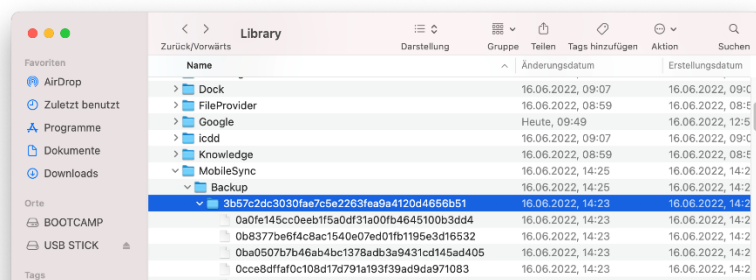
1.5.4 iOS Backups

Nutzer von iOS-Devices haben die Möglichkeit, ein Backup ihres Geräts herzustellen, das zu einer späteren Wiederherstellung bzw. zu einer Sicherung von Inhalten benutzt werden kann. Backups von iOS-Devices können entweder lokal mit iTunes erstellt oder in die iCloud synchronisiert werden. Besitzer von mobilen iDevices haben damit die Möglichkeit, ihre Geräte auch ohne Vorhandensein eines Computers zu sichern. Zur Sicherung von iDevices via iCloud ist ein gültiger iCloud-Account nötig.

iTunes oder iCloud-Backups von iDevices müssen nicht unbedingt alle auf dem Gerät vorhandenen Daten beinhalten, da nicht alle Daten gesichert werden. Sicherungen mit iTunes können lokal auf Windows- oder Mac-Systemen erstellt werden. Bei der Durchführung eines Backups kann festgelegt werden, ob es unverschlüsselt oder verschlüsselt abgespeichert werden soll. Backups können automatisch per WiFi oder per USB durchgeführt werden.

Sicherungen sind an mehreren Speicherorten zu finden. Unter Windows 7 und höher liegen sie unter „/Documents and Settings/Users/[Nutzer]/AppData/Roaming/Apple Computer/Mobile-Sync/Backup/“. Bei macOS befinden sie sich im Verzeichnis „~/Library/Application Support/MobileSync/Backup/“. Sicherungen werden grundsätzlich verschlüsselt in die Cloud übertragen und auch verschlüsselt auf Apple-Servern abgespeichert (128-Bit-AES).

Wird ein iDevice zum ersten Mal angeschlossen und von iTunes erkannt, wird eine 40-stellige alphanumerische UDID (Unique Device ID) erstellt. Unter der UDID (Unique Device ID) wird das Backup abgespeichert. Diese UDID berechnet sich aus dem SHA1-Wert von Geräteinformationen wie Seriennummer, IMEI sowie Mac-Adressen der WiFi- und Bluetooth- Schnittstellen.



Innerhalb des Geräteverzeichnis sind mit einer GUID bezeichnete Dateien vorhanden, welche die eigentlichen Inhalte der Dateien des iOS-Betriebssystems darstellen. Innerhalb des Geräteverzeichnis befinden sich zudem die Dateien „Info.plist“, „Status.plist“, „Manifest.plist“ und „Manifest.mbdb“. Sie enthalten die folgenden Informationen:

Datei	Beschreibung
Status.plist	Backup-Status-Zeitstempel des Backups, Backup-Typ
Info.plist	Name des iDevices, IDs (GUID, ICCID, IMEI), iOS-Version, installierte Apps u.a.
Manifest.plist	Backup verschlüsselt, Zeitstempel des Backups, <u>Passcode vorhanden</u>
Manifest.mbdb	Proprietäre Datenbank mit Mapping der GUIDs zu Originaldateien

Zur Analyse ist eine Vielzahl von freien Open-Source-Tools und kommerziellen Programmen verfügbar, die meist eine Extraktion der Verzeichnisse und Dateien anbieten. Kommerzielle Programme sind UFED Cellebrite, MSAB XRY, Magnet Axion und Sumuri Recon. Zu den Open-Source und Trail Tools zählen Backupbot (Quelle: <http://www.icopybot.com/itunes-backup-manager.htm>), iPhone Backup Extractor (Quelle: <http://www.iphonebackupextractor.com/de/>) und iExplorer (Quelle: <https://www.macroplant.com/iexplorer/>).

