Studienprogramm Sachbearbeiter:in Digitale Forensik
Praktikum Betriebssysteme
Prof. Bodach | Wetterau B.Sc. | Hossfeld B.Sc.
Stand: 07.06.2023

HOCHSCHULE MITTWEIDA
University of Applied Sciences

Betriebssysteme

Praktikum macOS

In diesem Praktikum lernen Sie verschiedene Methoden zur Analyse von macOS Artefakten. Dazu werden verschiedene Tools und Parser vorgestellt, welche eine einfache und übersichtliche Arbeit ermöglichen. Sie sind nach dem Praktikum in der Lage diese Tools in den aufgezeigten Sachverhalten anzuwenden und deren Ergebnisse zu verstehen.

Inhalte des Praktikums:

- $ Erstellen eines Datenträgerimages in macOS
- $ Spotlightuntersuchung mit Python Parser
- $ FSEvent Untersuchung mit FSEvent Parser
- $ Untersuchung der Safari History mit SQLite
- $ Untersuchung der KnowledgeC.db mit SQLite

Vorbereitung

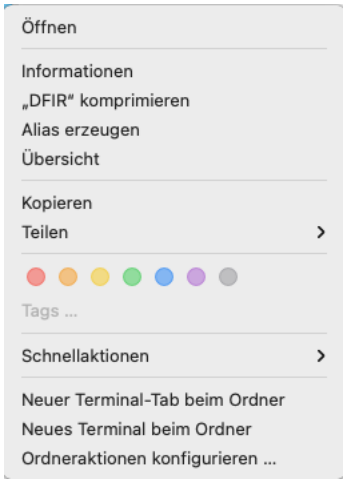
Zur Vorbereitung auf die Auswertung der gesicherten Daten kopieren Sie sich bitte von diesem Link die **`PRBlockMacOS.iso`** Datei auf das lokale Laufwerk D:. Erstellen Sie dort einen **neuen Ordner macOS**! Die Tools, welche sich auf dem ISO-Abbild befinden, benötigen wir später zur Analyse der macOS Artefakte. Die Ausgaben wollen wir in den neuen Ordner speichern.

Erstellen eines Datenträgerimages

Zuerst wollen wir uns mit der Sicherung des Dokumentenverzeichnisses des Nutzer1 beschäftigen. Auch dazu gibt es unter macOS eingebaute Tool, welche uns diese Aufgabe abnehmen. Öffnen Sie dazu das Terminal und geben anschließend folgenden Befehl ein:

```
$ hdiutil create -srcfolder /User/nutzer1/Documents -volname „Datensicherung Dokumente Nutzer1“ /Users/nutzer1/Desktop/Datensicherung.dmg
```

Anschließend sollten Sie auf dem Desktop ein neues Datenträgerimage sehen. Öffnen Sie mit einem Rechtsklick auf das Icon auf dem Desktop die Übersicht „Informationen“ und schauen Sie sich an, welche Informationen Sie dem Fenster entnehmen können.



Binden Sie anschließend das Datenträgerimage in das System ein. Dazu schließen Sie zuerst du Informationsseite und klicken dann doppelt auf das Icon auf dem Desktop. Nach einer kurzen Wartezeit öffnet sich der Finder und stellt Ihnen die Inhalte der Sicherung dar. Durch einen Rechtsklick auf eine Datei oder Verzeichnis in dem Image stellen wir fest, dass keine Daten gelöscht werden können.

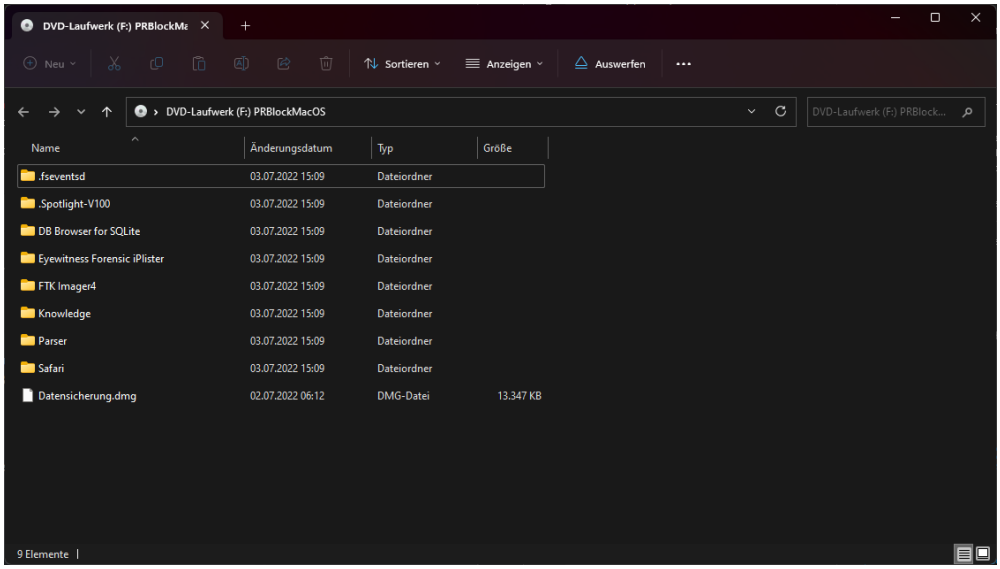
Um das zu ermöglichen, hängen wir das Image wieder aus dem System aus. Dies tun wir über das neue Icon auf dem Desktop mit dem Namen „Datensicherung Dokumente Nutzer1“ und werfen es mit einem Rechtsklick und Auswerfen aus dem System aus. Anschließend binden wir es wieder ein mit dem folgenden Befehl:

```
$ hdiutil attach /Users/nutzer1/Desktop/Datensicherung.dmg -shadow
```

Durch die Angabe von **`-shadow`** wird eine Shadow-Datei angelegt, in welche alle Änderungen geschrieben werden. Damit können im Datenträgerimage Daten geändert werden und die Änderungen werden dann in die Shadow-Datei geschrieben. Überprüfen Sie das, indem Sie erneut auf eine Datei oder Verzeichnis in dem Abbild rechtsklicken. Sie müssten feststellen, dass Sie die Datei in den Papierkorb verschieben können.

Spotlight Untersuchung mit Python Parser

Wir fahren ab jetzt mit der Untersuchung der macOS Artefakte auf dem Windows-Hostsystem fort! Wie im Kapitel Vorbereitung bereits geschrieben sollten Sie an der Stelle die heruntergeladene ISO-Datei auf Ihrem System vorliegen haben. Suchen Sie diese ggf. auf dem Dateisystem in Ihrem Windows-Hostsystem auf und binden Sie diese mit einem Doppelklick ein. Anschließend sollten Sie in Windows folgende Ansicht erhalten:



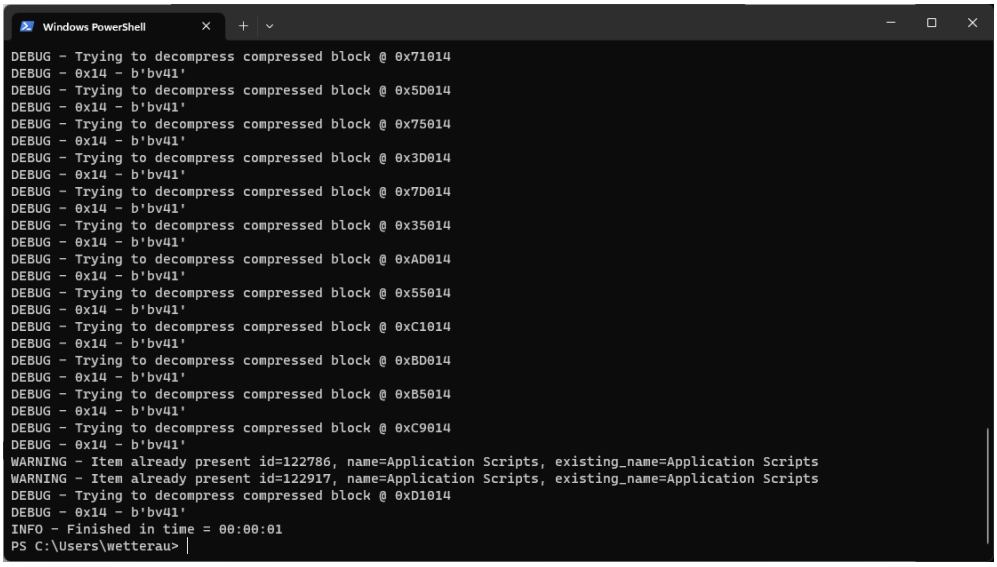
Rufen Sie auf dem Abbild das Verzeichnis Parser auf. Sie sollte hier zwei Python Scripte finden für Spotlight und FSEvent. Bitte informieren Sie sich kurz dazu, was ein Parser ist und welche Besonderheit ein Parser hat, wenn Ihnen das an dieser Stelle noch nicht klar ist. Öffnen eine neue Kommadozeile, wobei diese nicht mit Adminrechten gestarten werden muss. Wir wollen mit dem Parser die Spotlight-Datenbank untersuchen. Diese befindet sich auf dem Abbild unter dem Pfad

```
\.Spotlight-V100\Store-V2\C92F6DA8-D4F3-4C2E-9354-8691E97EDC36\.store.db
```

Notieren Sie sich vorbereitend den Laufwerksbuchstaben der eingebundenen CD. Mit all diesen Informationen starten Sie bitte den Parser durch den folgenden Befehl (F: durch Ihren Laufwerksbuchstaben ersetzen):

```
$ F:\Parser\spotlight_parser.exe F:\.Spotlight-V100\Store-V2\C92F6DA8-D4F3-4C2E-9354-
  8691E97EDC36\.store.db D:\macOS\SpotlightOut
```

Die sollten nach der Ausführung des Befehls folgende Ansicht erhalten:



Der Parser sollte bei erfolgreichem Abschluss eine Textdatei in dem von Ihnen erstellte Verzeichnis generiert haben. Diese sollte den Namen spotlight-store_data.txt tragen. Rufen Sie diese auf und schauen Sie sich die enthaltenen Informationen an. Im vorigen Praktikum haben wir nach dem Begriff „confidential“ gesucht. *Tun Sie dies auch hier in der Textdatei. Nutzen Sie dazu entweder Strg + F oder klicken Sie auf Bearbeiten und dann Suchen. Geben Sie den Begriff „confidential“ ein und schauen Sie, welche Treffer Sie erhalten. Welche Informationen können Sie feststellen?*

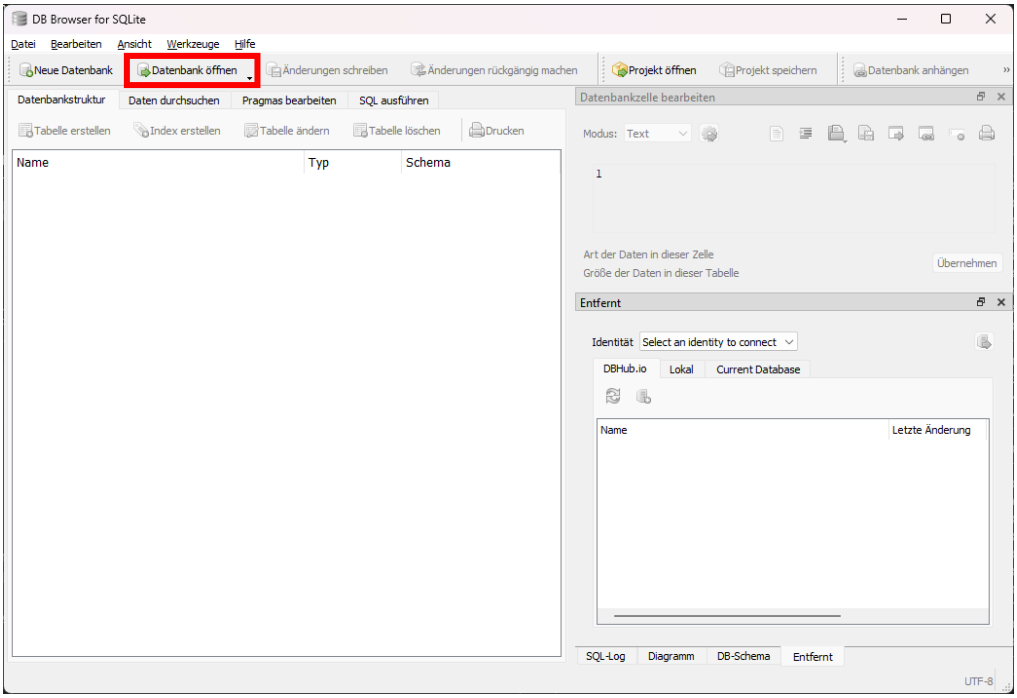
FSEvent Untersuchung mit FSEvent Parser

Als nächstes schauen wir uns die FSEvent an. Auch dazu finden Sie einen Parser auf dem ISO-Datenträger. Um herauszufinden, wie der Parser funktioniert, starten wir diesen einfach mal mit dem einfachen Aufruf des Skripts:

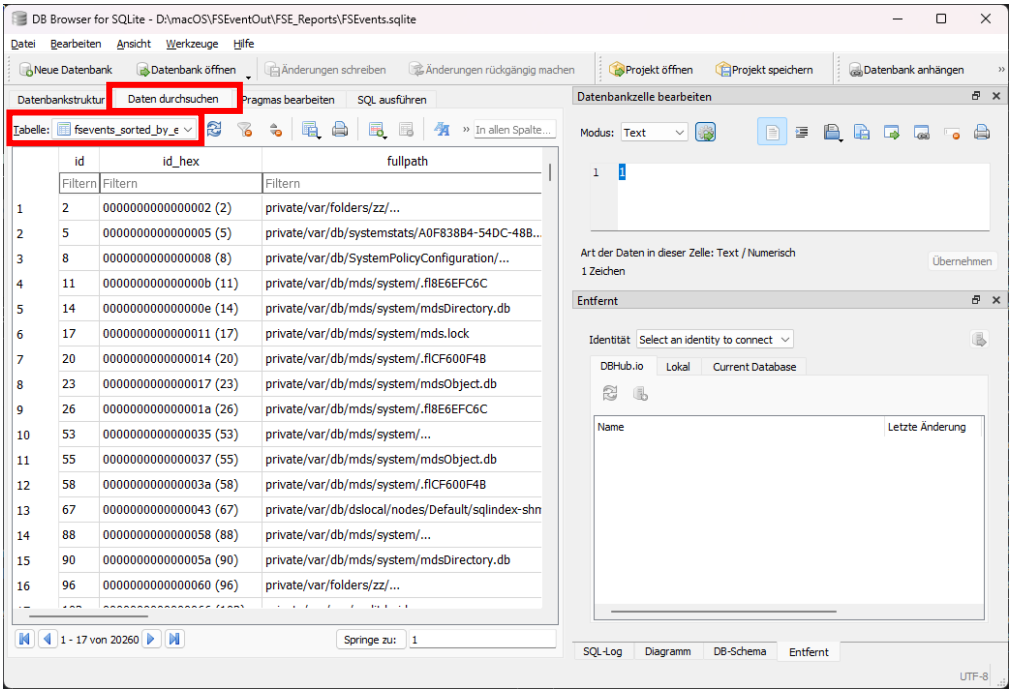
```
$ F:\Parser\FSEParser_V4.exe
```

Damit bekommen wir eine grobe Übersicht über die Funktion des Tools. Alle mit „REQUIRED“ gekennzeichnet Argumente braucht der Parser zur Erfüllung seiner Aufgabe. Wir wollen hier das Verzeichnis .fseventsd auf dem ISO-Abbild als Eingabe verwenden. Damit müssen wir als Quelltyp folder angeben. Schreiben Sie die Ausgaben in das Verzeichnis FSEventOut in Ihrem erstellten macOS-Ordner. **ACHTUNG: Dieses müssen Sie vorher erst erstellen!** *Stellen Sie nun anhand der Hilfestellung des Kommandos und anhand der gegebenen Informationen den Befehl zusammen, dass die beschriebene Funktionalität umgesetzt wird.*

Anschließend sollten Sie sehen, dass die Informationen Stück für Stück extrahiert werden. Nach erfolgreichem Abschluss der Operation hat Ihnen der Parser drei neue Dateien in dem Verzeichnis macOS/FSEventOut erstellt. Einmal wurden die Ergebnisse als TSV-Datei exportiert. Sinnvoller für die Untersuchung der Inhalte ist aber die SQLite-Datenbank. Zur Untersuchung dieser Datenbank steht Ihnen auf dem ISO-Abbild das Tool DB Browser for SQLite zur Verfügung. *Starten Sie das Tool durch einen Doppelklick auf die EXE im gleichnamigen Verzeichnis!* (Nicht die Cipher Variante)



Öffnen Sie die gerade mit dem Parser erstellte SQLite-Datenbank über den Button Datenbank öffnen und suchen Sie die Datenbank in Ihrem Dateiverzeichnis. Anschließend sollte Sie eine Ansicht über die verfügbaren Datenbanktabellen bekommen. Klicken Sie auf den Reiter Daten durchsuchen und wählen Sie die Tabelle fsevents_sorted_by_event_id aus.

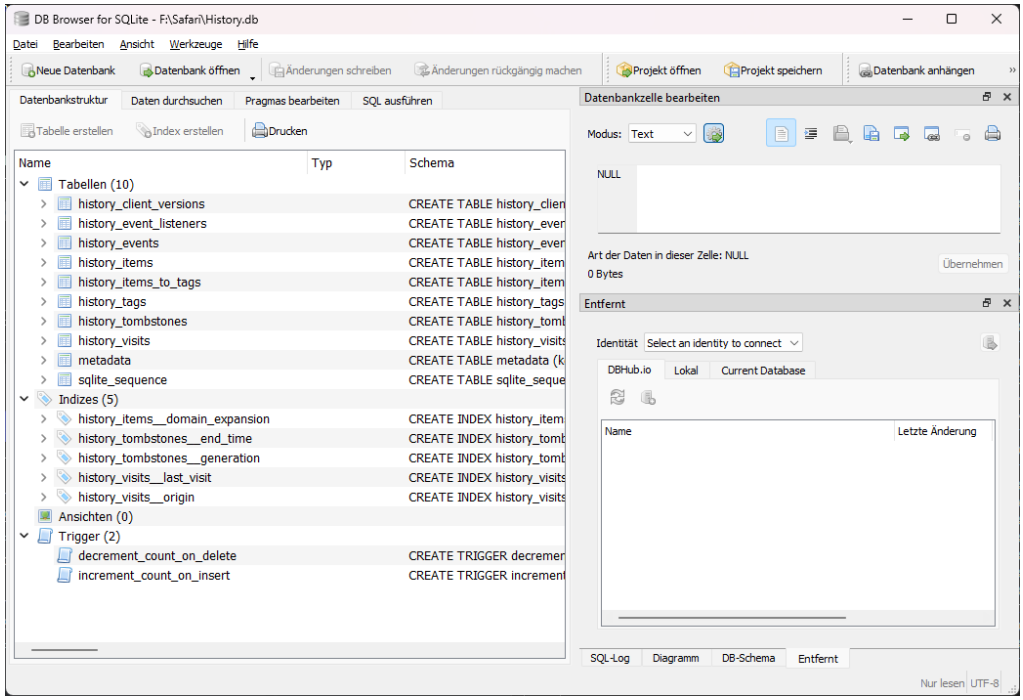


Filter Sie unter Kopfzeile der Tabelle in der Spalte fullpath nach dem Begriff „confidential“. Finden Sie auch hier die gleichen Eintragungen, wie Sie in der Textdatei und in der Spotlightsuche gefunden haben?

Untersuchung der Safari History mit SQLite

In diesem Abschnitt des Praktikums wollen wir uns erneut mit dem DB Browser for SQLite beschäftigen. SQLite ist ein Datenbankformat, welche häufig zur effizienten Speicherung und Verwaltung von Daten dient. Die Zugriffe sind einfach und schnell und die Speicherung der Daten schlank. Damit bietet sich SQLite für viele Anwendungen Softwaredevelopment an.

Browser speichern, wie alle wissen sollten, den Verlauf der besuchten Webseiten. Hierfür ist das SQLite-Format wie gemacht. Alle Browser, welche auf Chromium basieren (Opera, Chrome, Firefox, Safari) haben ein sehr ähnliches Format zum Ablegen der Verlaufsinformationen. Wie schon erwähnt unterstützt auch der Standardbrowser unter macOS Safari diese Datenbank und speichert die Verlaufsinformationen in die History.db. Sie finden einen Vertreter dieser Datenbank auf dem ISO-Abbild unter Safari. Öffnen Sie diese Datenbank wie kennengelernt im DB Browser!

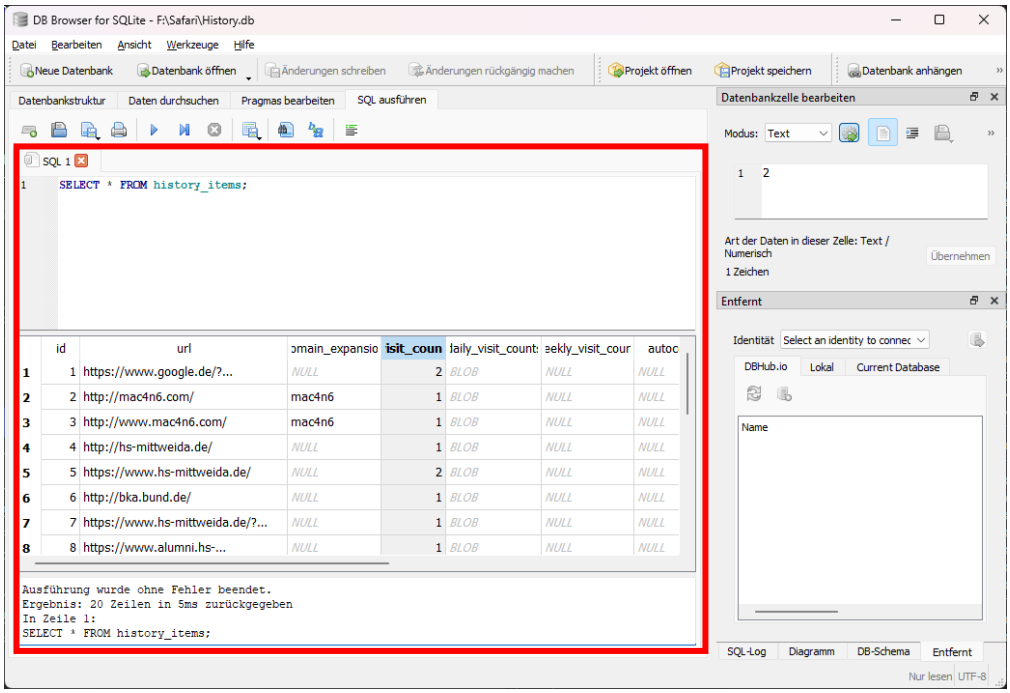


Schauen Sie sich bitte die Tabellennamen an und überlegen Sie sich, welche Tabellen den höchsten forensischen Wert haben!

Wir schauen uns die Datenbankinhalte ein wenig genauer an. Dazu macht sich die Anwendung von SQL nützlich. Wir nutzen SQL, um Informationen mit spezifischen Kriterien herauszufiltern. Das macht deswegen Sinn, da die Extraktion aller Daten unverhältnismäßig, unübersichtlich, nicht zielführend und damit auch nicht aufgabenerfüllend wäre. Wechseln Sie im DB Browser in den Tab SQL ausführen. Dort sehen wir ein Fenster mit den Namen SQL 1, in welchem wir nun SQL-Anfragen an die Datenbank stellen können. Wir starten mit einem einfachen SQL-Statement:

```
$ SELECT * FROM history_items;
```

Mit **SELECT** wählen wir alle Spalten aus der Tabelle (**FROM**) history_items aus. Dabei haben wir bisher noch keine Kriterien gestellt. Sie sollten nun die folgende Ansicht bekommen, welche alle Inhalte der mit **FROM** gewählten. Tabelle einsehen können. *Überlegen Sie sich welche Bedeutung die Spalten url und visit_count haben. Recherchieren Sie ggf. kurz im Internet.*

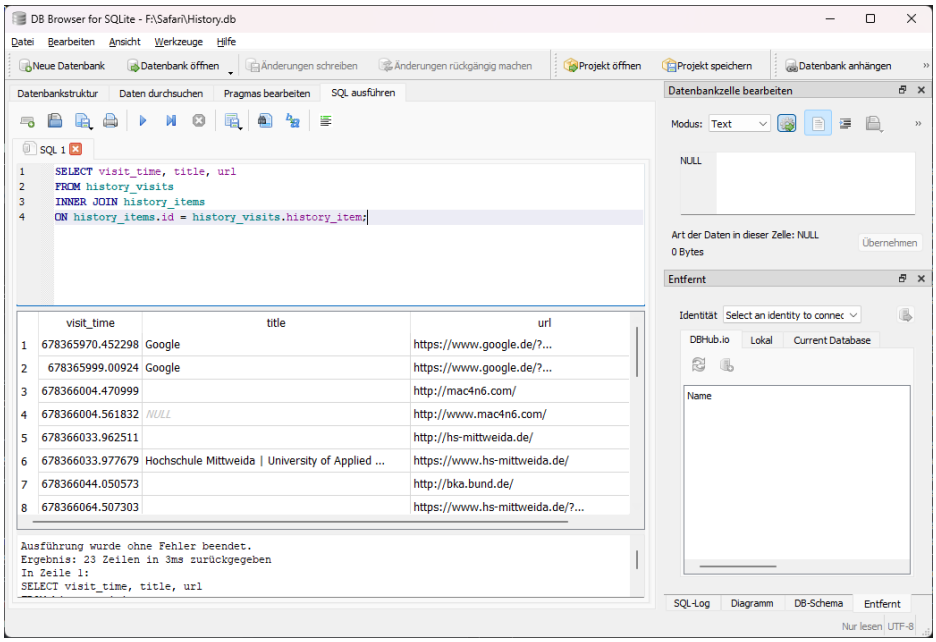


Die zweite Tabelle, welche forensischen Wert an dieser Stelle hat, ist die Tabelle history_visits. *Setzen Sie in dem Fenster ebenso ein Statement ab mit dem Sie alle Inhalte aus der Tabelle history_visits erhalten! Können Sie mit den Informationen viel anfangen? Sind die Informationen repräsentativ und können Sie diese so in Ihrem forensischen Auswertungsbericht verwenden? Warum?*

Wir wollen die Informationen der beiden Tabellen jeweils mit den Informationen der anderen Tabelle anreichern. Dazu brauchen wir eine Verbindung zwischen den Tabellen. Die Verbindungen werden in relationalen Datenbanken über sogenannte Schlüsselattribute hergestellt. Ein Primärschlüssel bezeichnet in einer Tabelle eindeutig einen Eintrag der Tabelle. Das Gegenstück zum Primärschlüssel ist der Fremdschlüssel. Sie sind Schlüsselattribute einer anderen Tabelle und referenzieren dort eindeutig einen Eintrag. In unserem Fall ist der Fremdschlüssel der Tabelle history_visits das Attribut history_item. *Schauen Sie in die Tabell history_items. Können Sie dort feststellten, welches Attribut den Primärschlüssel darstellt und jeden Eintag eindeutig referenziert?*

Folgend schreiben wir ein SQL-Statement mit dem wir die Inhalte der Tabellen miteinander verbinden können. Dazu nutzen wir den **INNER JOIN** und beziehen uns dabei auf die Schlüsselspalten, welche die beiden Tabellen verbinden:

```
$ SELECT visit_time, title, url
$ FROM history_visits
$ INNER JOIN history_items
$ ON history_items.id = history_visits.history_item;
```



Problematisch und eher nicht verständlich ist der Zeitstempel des Eintrags in der Spalte visit_time. Zur besseren Veranschaulichung nutzen wir eine eingebaute Funktion datetime(). Mit dieser können wir die Ganzzahl in ein geeigenetes Datumsformat überführen. Probieren wir das einmal mit dem Unix-Epoch-Format. Ersetzen Sie die Angabe von visit_time nach dem SELECT im oberen Statement durch folgende Syntax:

```
$ datetime(visit_time, 'unixepoch', 'localtime ') AS "Time"
```

Setzen Sie den Befehl durch die Ergänzung erneut ab und schauen Sie sich die Ergebnisse Ihrer Operation durch. *Können die Einträge so stimmen? Können Sie sich erklären, warum das der Fall ist?*

Für die Analyse der Zeitstempel in macOS brauchen wir die Zeit nach er macepoch Zeit. Die Unixepoch zählt die Sekunden seit dem 01.01.1970. Die macepoch zählt allerdings die Sekunden ab dem 01.01.2001. Damit besteht zwischen den beiden Zeitformaten eine Differenz von 978.307.200 Sekunden.

Für Interessierte: *Erstellen Sie einen genauen Rechenweg, der zeigt, wie der exakte Wert in Sekunden zustande kommt!*

Folglich berechnet sich die macepoch wie folgt:

$$macepoch(cacoa\ touch|apple\ base\ time) = unixepoch + 978307200$$

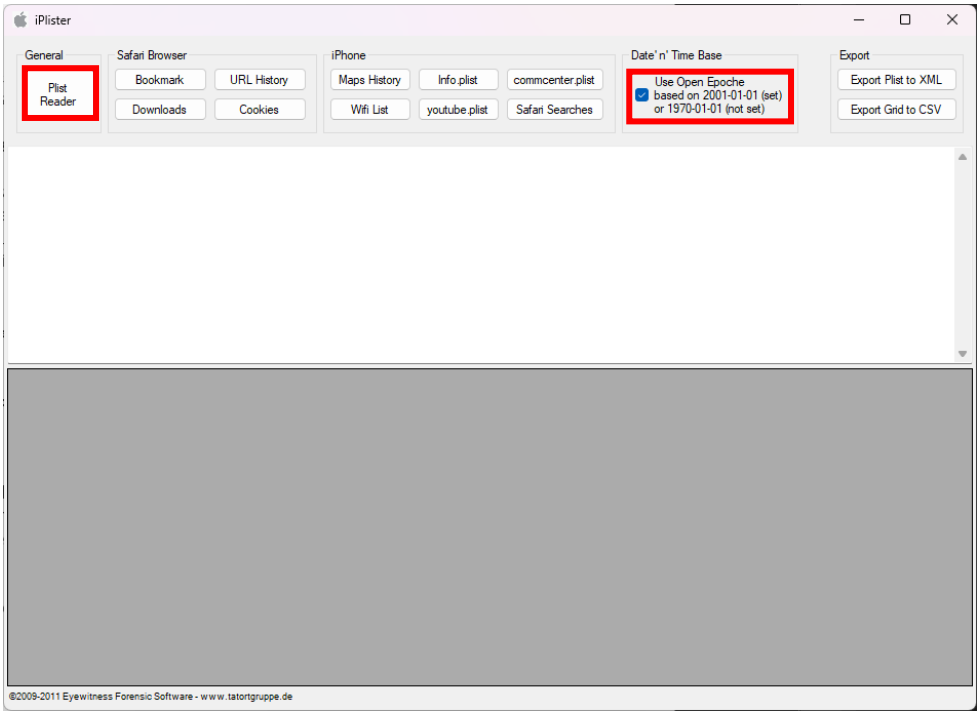
Daraus ergibt sich für die Funktion datetime() folgende Angabe:

```
$ datetime(visit_time + 978307200, 'unixepoch', 'localtime ') AS "Time"
```

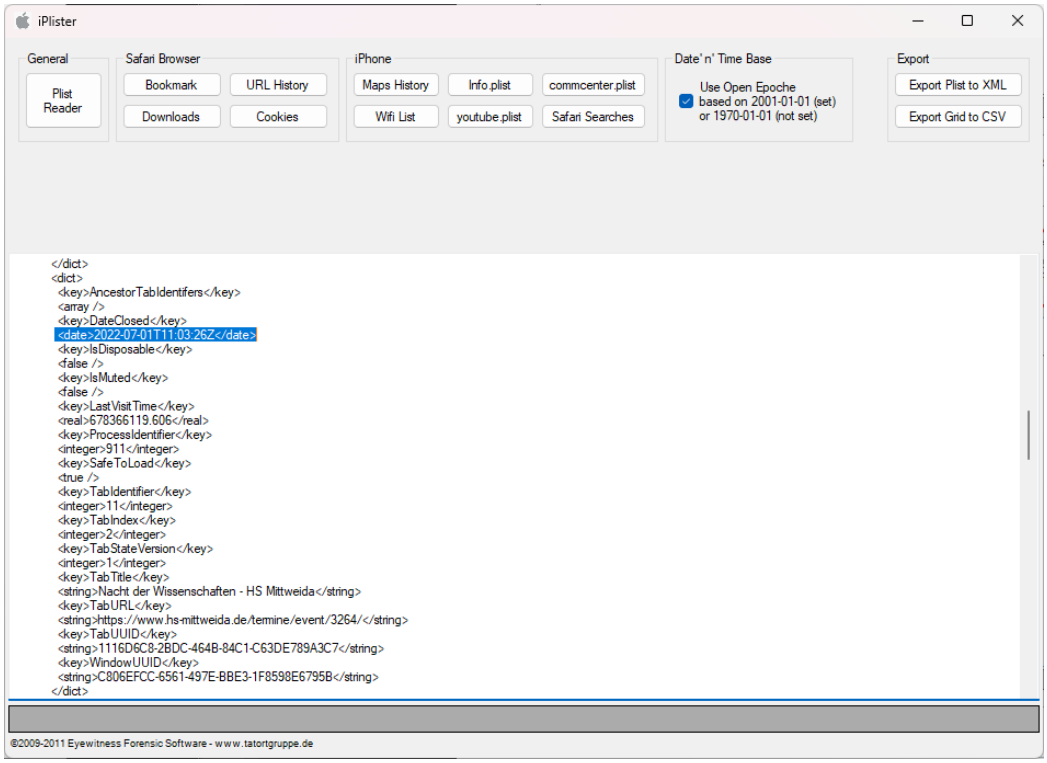
Abschließend wollen wir noch herausfinden, ob der Nutzer Seiten besucht hat, welche mit dem BKA in Verbindung stehen. Dazu nutzen wir das Statement LIKE und geben folgend eine Zeichenkette an, nach der wir filtern wollen. Sinn macht das in unserer Ausabe nur in der Spalte url. Ebenso wollen wir die Reihenfolge der Einträge nach der Zeit sortieren, und zwar so, dass wir die neusten Einträge zuerst sehen. In Summe ergibt sich mit allen Anpassungen das folgende Statement:

```
$ SELECT datetime(visit_time + 978307200, 'unixepoch', 'localtime ') AS "Time" , title, url
$ FROM history_visits
$ INNER JOIN history_items
$ ON history_items.id = history_visits.history_item
$ WHERE url LIKE '%bka%'
$ ORDER BY visit_time DESC;
```

Eine weitere Untersuchung, welche durchgeführt werden kann, ist die Untersuchung der Binary Plist Dateien des Safari Browser. Dies kann durch den iPlister von Eyewitness Forensic vorgenommen werden. Diesen finden Sie ebenfall auf dem ISO-Abbild im Verzeichnis Eyewitness Forensic iPlister. *Starten Sie den iPlister mit einem Doppelklick auf dessen EXE.*



Im iPlister angekommen, wählen wir die entsprechende Binary Plist aus mit einem Klick auf Plist Reader. Die Plist zur Untersuchung befindet sich auf dem ISO-Abbild unter dem Verzeichnis Safari. Wählen Sie dort die Datei RecentlyClosedTabs.plist aus. Mit der Checkbox im Feld Date‘ n‘ Time Base können Sie einstellen, welche Zeitformat bei der Analyse der Plist angewandt werden soll. Wollen Sie das Zeitformat umstellen, müssen Sie allerdings die Datei neu laden.



Untersuchung der KnowledgeC.db mit SQLite

Zuletzt schauen wir uns eine weitere Datenbank an, welche forensische Informationen für uns bereithält. Sie enthält auf macOS-Geräten Pattern of Life Informationen. Auch zu deren Analyse verwenden wir den DB Browser for SQLite. Starten Sie den SQLite Browser, wenn er nicht schon läuft und laden die KnowledgeC.db aus dem Verzeichnis Knowledge auf dem ISO-Abbild. Sie sehen, dass sich sehr viele Tabellen in der Datenbank befinden. Zur Analyse bekommen Sie ein vorgefertigtes Statement, welches Sie in den Reiter „SQL ausführen“ schreiben und ausführen sollen:

```
SELECT
      datetime(ZOBJECT.ZCREATIONDATE + 978307200, 'unixepoch', 'localtime') AS "ENTRY CREATION",
      CASE ZOBJECT.ZSTARTDAYOFWEEK
        WHEN "1" THEN "Sunday"
        WHEN "2" THEN "Monday"
        WHEN "3" THEN "Tuesday"
        WHEN "4" THEN "Wednesday"
        WHEN "5" THEN "Thursday"
        WHEN "6" THEN "Friday"
        WHEN "7" THEN "Saturday"
      END "DAY OF WEEK",
      datetime(ZOBJECT.ZSTARTDATE + 978307200, 'unixepoch', 'localtime') AS "START",
      datetime(ZOBJECT.ZENDDATE + 978307200, 'unixepoch', 'localtime') AS "END",
      ZOBJECT.ZENDDATE-ZOBJECT.ZSTARTDATE AS "USAGE IN SECONDS",
      ZOBJECT.ZSTREAMNAME,
      ZOBJECT.ZVALUESTRING
FROM ZOBJECT
ORDER BY "START"
```

Dieses Statement sammelt die wichtigsten Informationen aus der Tabelle und stellt die Nutzungsinformationen von Anwendungen zur Verfügung. *Führen das Statement aus und schauen Sie sich die Ergebnisse dazu an.*

Als nächsten Befehl schauen wir uns an, wie welche Apps verwendet wurden. Auch dazu bekommen Sie ein vorgefertigtes Statement:

```
SELECT
      datetime(ZOBJECT.ZCREATIONDATE + 978307200,'unixepoch', 'localtime') AS "ENTRY CREATION",
      CASE ZOBJECT.ZSTARTDAYOFWEEK
        WHEN "1" THEN "Sunday"
        WHEN "2" THEN "Monday"
        WHEN "3" THEN "Tuesday"
        WHEN "4" THEN "Wednesday"
        WHEN "5" THEN "Thursday"
        WHEN "6" THEN "Friday"
        WHEN "7" THEN "Saturday"
      END "DAY OF WEEK",
      datetime(ZOBJECT.ZSTARTDATE + 978307200,'unixepoch', 'localtime') AS "START",
      datetime(ZOBJECT.ZENDDATE + 978307200,'unixepoch', 'localtime') AS "END",
      ZOBJECT.ZENDDATE-ZOBJECT.ZSTARTDATE AS "USAGE IN SECONDS",
      ZOBJECT.ZSTREAMNAME,
      ZOBJECT.ZVALUESTRING, ZSTRUCTUREDMETADATA.Z_DKAPPLICATIONACTIVITYMETADATAKEY__ACTIVITYTYPE
      AS "ACTIVITY TYPE",
      ZSTRUCTUREDMETADATA.Z_DKAPPLICATIONACTIVITYMETADATAKEY__TITLE AS "TITLE",
      ZSTRUCTUREDMETADATA.Z_DKAPPLICATIONACTIVITYMETADATAKEY__USERACTIVITYREQUIREDSTRING AS
      "ACTIVITY STRING",
      datetime(ZSTRUCTUREDMETADATA.Z_DKAPPLICATIONACTIVITYMETADATAKEY__EXPIRATIONDATE +
      978307200,'unixepoch', 'localtime') AS "EXPIRATION DATE"
FROM ZOBJECT
LEFT JOIN ZSTRUCTUREDMETADATA ON ZOBJECT.ZSTRUCTUREDMETADATA = ZSTRUCTUREDMETADATA.Z_PK
WHERE ZSTREAMNAME IS "/app/usage"
ORDER BY "START"
```

Schauen Sie sich auch die Ausgaben dieses Statements an und analysieren sie überblicksartig die Einträge. Mit dem letzten Befehl wollen wir die Nutzung des Safari Browsers analysieren mit dessen aufgerufenen URLs

```
SELECT
      datetime(ZOBJECT.ZCREATIONDATE + 978307200, 'unixepoch', 'localtime') AS "ENTRY CREATION",
      CASE ZOBJECT.ZSTARTDAYOFWEEK
        WHEN "1" THEN "Sunday"
        WHEN "2" THEN "Monday"
        WHEN "3" THEN "Tuesday"
        WHEN "4" THEN "Wednesday"
        WHEN "5" THEN "Thursday"
        WHEN "6" THEN "Friday"
        WHEN "7" THEN "Saturday"
      END "DAY OF WEEK",
      datetime(ZOBJECT.ZSTARTDATE + 978307200, 'unixepoch', 'localtime') AS "START",
      datetime(ZOBJECT.ZENDDATE + 978307200, 'unixpeoch', 'localtime') AS "END",
      ZOBJECT.ZENDDATE-ZOBJECT.ZSTARTDATE AS "USAGE IN SECONDS",
      ZOBJECT.ZSTREAMNAME,
      ZOBJECT.ZVALUESTRING,
```

```
    ZSTRUCTUREDMETADATA.Z_DKDIGITALHEALTHMETADATAKEY__WEBPAGEURL AS "WEBPAGEURL",
    ZSTRUCTUREDMETADATA.Z_DKAPPLICATIONACTIVITYMETADATAKEY__USERACTIVITYREQUIREDSTRING AS
    "ACTIVITY STRING",
    datetime(ZSTRUCTUREDMETADATA.Z_DKAPPLICATIONACTIVITYMETADATAKEY__EXPIRATIONDATE +
    978307200, 'unixepoch', 'localtime') AS "EXPIRATION DATE",
    ZSTRUCTUREDMETADATA.Z_DKINTENTMETADATAKEY__INTENTCLASS AS "INTENT CLASS",
    ZSTRUCTUREDMETADATA.Z_DKINTENTMETADATAKEY__INTENTVERB AS "INTENT VERB",
    ZSTRUCTUREDMETADATA.Z_DKINTENTMETADATAKEY__SERIALIZEDINTERACTION AS "SERIALIZED
    INTERACTION",
    ZSOURCE.ZBUNDLEID
FROM ZOBJECT
LEFT JOIN ZSTRUCTUREDMETADATA ON ZOBJECT.ZSTRUCTUREDMETADATA = ZSTRUCTUREDMETADATA.Z_PK
LEFT JOIN ZSOURCE ON ZOBJECT.ZSOURCE = ZSOURCE.Z_PK WHERE ZSTREAMNAME IS "/app/webUsage"
ORDER BY "START"
```

Interessant kann genau das letzte Statement sein, da dieses auch Einträge der History enthält, obwohl diese gelöscht wurden. Die Datenbank als auch das hier demonstrierte Statement bieten sich im forensischen Kontext zum Wiederherstellen von gelöschten Daten an. Schauen Sie sich in der entstehenden Ausgabe die Einträge genau an und versuchen Sie die Tabelle zu verstehen.