Technische Berichte in Digitaler Forensik

Herausgegeben vom Lehrstuhl für Informatik 1 der Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) in Kooperation mit dem Masterstudiengang Digitale Forensik (Hochschule Albstadt-Sigmaringen, FAU, Goethe-Universität Frankfurt am Main)

Forensische Untersuchung der Anwendung „Drive File Stream“ unter Microsoft Windows

Markus Hinkelmann

25.02.2018
Technischer Bericht Nr. 13

Zusammenfassung

Bereits seit längerer Zeit stellt die Firma Google die Cloudspeicherlösung „Google Drive“ bereit. Ende des Jahres 2017 wurde durch diese die Software „Drive File Stream“ veröffentlicht, welche die Synchronisation zwischen dem Rechner des Anwenders und dem Cloudspeicher ermöglicht. Im Rahmen dieser Analyse soll dargestellt werden, welche forensisch relevanten Artefakte bei der Installation, Benutzung und Deinstallation dieser Anwendung entstehen. Dabei erfolgte auch eine erste Analyse der durch das Programm erzeugten Datenbanken, sowie eine Betrachtung des verwendeten Cache-Mechanismus.

Entstanden im Rahmen des Moduls Browser- und Anwendungsforensik des Studiengangs Digitale Forensik im Wintersemester 2017/2018 unter der Anleitung von Felix Freiling, Holger Morgenstern und Gaston Pugliese.

Inhaltsverzeichnis

1 Einleitung

Im Rahmen dieses Berichts werden die Ergebnisse der Analyse der Software „Drive File Stream“ aus dem Hause Google LLC dargestellt. Untersucht wurde die Version 25.102.133.409. Bei dem analysierten Programm handelt es sich um eine Client-Anwendung, welche für Windows und Mac OS X verfügbar ist und einen Zugriff auf die Cloudspeicherlösung „Google Drive“ ermöglicht.

Neben „Drive File Stream“ bietet Google auch die die Software „Backup and Sync“ an. Auch diese ermöglicht Zugriffe auf das im Rahmen des Abo-Produktes „GSuite“ bereitgestellte „Google Drive“. Während Google den Einsatzzweck der Software „Backup and Sync“ eher im Bereich des Heimanwenders sieht, richtet sich die Lösung „Drive File Stream“ exklusiv an Abonnenten des kostenpflichtigen „GSuite“-Dienstes. [1]

Das Hauptaugenmerk der Arbeit liegt auf der Software „Drive File Stream“. Die Analyse der Anwendung soll unter folgenden zwei Gesichtspunkten durchgeführt werden:

- Welche Spuren der Anwendung können an welcher Stelle im Dateisystem gefunden werden?
- Lassen sich aus diesen Spuren forensisch relevante Schlussfolgerungen ziehen?

Im Vorfeld der tatsächlichen Analyse der o.g. Software wird zunächst das Thema Anwendungsanalyse in Form eines allgemeinen Überblicks betrachtet. Dabei wird sowohl auf die untersuchte Software als auch auf die verwendeten Untersuchungsmethoden eingegangen und nachfolgend das Vorgehen während der Analyse dargestellt.

Nachdem die grundlegenden Dinge erläutert wurden, werden die Ergebnisse der allgemeinen Untersuchung präsentiert und einzelne Dateien, welche durch „Drive File Stream“ erzeugt werden, noch einmal gesondert betrachtet.

Am Ende werden die gewonnenen Erkenntnisse noch einmal zusammengefasst.

2 Allgemeiner Überblick

An dieser Stelle wird das Untersuchungsobjekt kurz dargestellt, die verwendeten Untersuchungsmethoden erläutert, sowie das Vorgehen während der Analyse beschrieben.

2.1 Beschreibung der Anwendung

Wie bereits in der Einleitung erläutert, ermöglicht die Software „Drive File Stream“ den lokalen Zugriff auf die in der Cloudspeicherlösung „Google Drive“ gespeicherten Daten.

Im Gegensatz zur Vorgängerlösung „Google Drive“ werden durch das hier untersuchte Programm nicht sämtliche Daten des Anwenders zwischen seinem Computer und dem Cloudspeicher synchronisiert, sondern es hält lediglich Dateinamen und wichtige Zeitstempel lokal vor. Dateien als solche werden erst bei einem Zugriff durch den Anwender heruntergeladen und angezeigt.

Der Anwender hat aber die Möglichkeit, einzelne Dateien oder Verzeichnisse zur Offline-Speicherung zu markieren. Diese werden dann vollständig auf den PC des Anwenders heruntergeladen und können so auch dann betrachtet und bearbeitet werden, wenn gerade keine Internetverbindung besteht. Etwaige Änderungen werden synchronisiert, sobald wieder eine Internetverbindung besteht. Der Zustand der einzelnen Dateien und Ordner wird dabei über entsprechende Statussymbole dargestellt.

Die Anwendung selbst wird im Hintergrund ausgeführt. Für den Benutzer tritt sie lediglich als Tray-Icon in Erscheinung, welches den aktuellen Status der Anwendung symbolisiert (Verbunden, Synchronisiert, Offline). Über einen Klick auf das Icon können weitere Informationen über laufende Synchronisierungsaktivitäten und den allgemeinen Stand der Anwendung angezeigt werden (s. Abbildung 1).

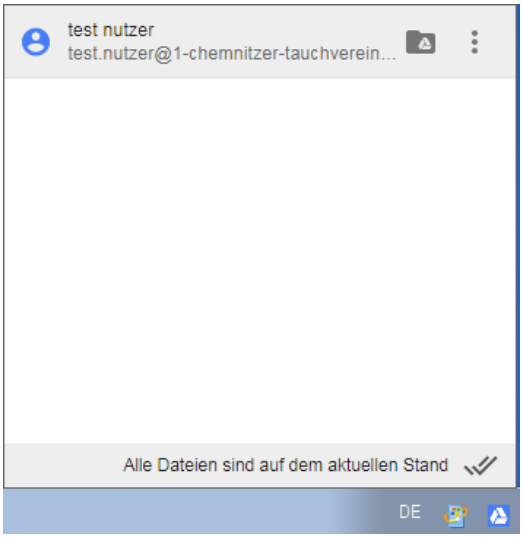


Abbildung 1: Drive File Stream Tray-Icon Dialog

Im Gegensatz zu vergleichbaren Anwendungen ähnlicher Dienste, wie zum Beispiel dem „DropBox“ Client, erfolgt die Zuordnung der Daten zum Onlinespeicher nicht mehr über ein synchronisiertes Verzeichnis im Dateisystem des Nutzers, sondern über ein Netzlaufwerk, welches alle Daten aus dem Onlinespeicher enthält. Standardmäßig wird diesem Laufwerk der Buchstabe „G“ zugewiesen. Die grafische Oberfläche des Programmes bietet derzeit noch keine Möglichkeit, dies zu ändern. Abbildung 2 zeigt das im Windows Explorer geöffnete Netzlaufwerk mit den Daten, die offline verfügbar sind (grüner Haken) oder lediglich online vorgehalten und erst bei entsprechendem Zugriff geladen werden (kleine Wolke).

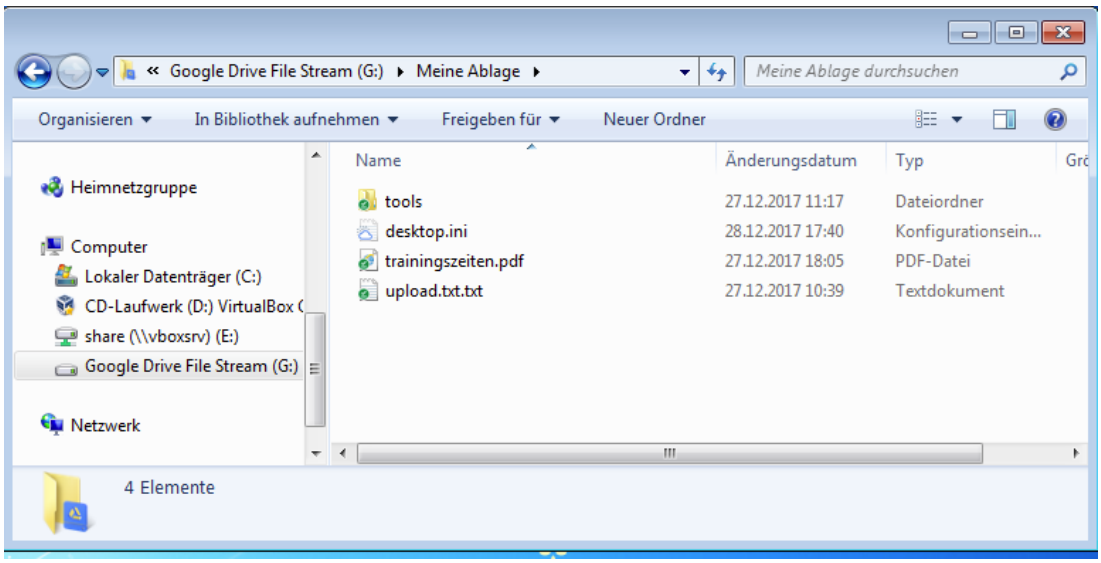


Abbildung 2: Google Drive File Stream Ordner

2.2 Verwendete Untersuchungsmethoden

Bei der Analyse einer Anwendung können grundsätzlich zwei Methoden zur Anwendung kommen. Gemeint sind die Ereignis- und die Zustandsmethode. Im vorliegenden Fall wurden beide Methoden genutzt. Im Folgenden sollen sie kurz vorgestellt und dabei auch gleich erläutert werden, welche Methode für welchen Zweck verwendet wurde.

2.2.1 Ereignismethode

Bei dieser Methode beobachtet man das Gesamtsystem im laufenden Betrieb. Die dabei auftretenden Ereignisse werden aufgezeichnet und gespeichert. Der Vorteil dabei ist, dass man die Ereignisse nach dem Auslöser filtern kann. Dadurch ist es möglich, dass zum Beispiel nur die Ereignisse betrachtet werden müssen, die durch ein bestimmtes Programm ausgelöst wurden. So ist es sehr effektiv möglich, für die Untersuchung irrelevante Änderungen auszublenden. [2, S. 103]

In diesem Zusammenhang ergibt sich aber auch ein Nachteil. Betrachtet man lediglich die Ereignisse, welche direkt von einem bestimmten Programm ausgelöst wurden, dann können unter Umständen indirekt verursachte Änderungen übersehen werden. Dies ist häufig der Fall, da Softwareentwickler mit ihren Produkten gerne auf bereits bestehende Funktionen anderer Programme bzw. des Betriebssystems zurückgreifen. Die durch diese ausgelösten Systemereignisse würden der o.g. Filterung zum Opfer fallen.

Aus diesem Grunde wurden die Ergebnisse der im Rahmen dieser Untersuchung angewandten Ereignismethode in der Folge durch die Ergebnisse der ebenfalls verwendeten Zustandsmethode verifiziert. [2, S. 103]

2.2.2 Zustandsmethode

Hierbei werden die Zustände des Dateisystems eingehend betrachtet. Zu diesem Zweck wird jeweils vor und nach der zu analysierenden Aktion ein Abbild des Dateisystems erzeugt und die Abbilder im Nachgang miteinander verglichen. Die dabei festgestellten Veränderungen im Dateisystem können so ziemlich eindeutig der untersuchten Aktion zugeordnet werden. [2, S. 103]

Eine zweifelsfreie Zuordnung ist bei dieser Methode jedoch nicht möglich, da nicht ausgeschlossen werden kann, dass einige Veränderungen im Dateisystem auch durch andere Prozesse (z.B. Hintergrundaktivitäten des Betriebssystems, Antivirensoftware usw.) ausgelöst oder auch rückgängig gemacht wurden. [2, S. 103]

Eine Möglichkeit, die Präzision der Zustandsmethode zu verbessern, ist die Aufzeichnung des sogenannten Hintergrundrauschens des Betriebssystems. Dabei werden Veränderungen durch das Betriebssystem aufgezeichnet, die ohne die Ausführung einer spezifischen Aktion entstehen, und diese später von den wie gerade beschrieben gewonnenen Analysedaten abgezogen. [2, S. 103]

Im Rahmen dieser Arbeit wurde die Zustandsmethode zur Analyse von Veränderungen der Windows Registrierungsdatenbank (Registry) und zur Verifizierung der Ergebnisse aus der Ereignismethode verwendet. [2, S. 103]

2.3 Vorgehen während der Untersuchung

Im Vorfeld der eigentlichen Analyse wurde zunächst festgelegt, welche Aktionen typische Anwendungsszenarien der zu untersuchenden Software abbilden und daher für die Analyse durchgeführt werden sollten. Zielstellung war es, für jede dieser Aktionen die durch sie verursachten spezifischen Spuren zu identifizieren. Eine Darstellung der definierten Aktionen findet sich in Abschnitt 2.3.1.

Zur Durchführung der geplanten Analyse wurde eine virtuelle Maschine mit dem Betriebssystem Windows 7 eingerichtet und der Zustand nach der erfolgreichen Installation des Betriebssystems und der benötigten Updates in Form eines Snapshots gesichert. Anschließend wurde das Hintergrundrauschen des Betriebssystems aufgezeichnet. Dazu wurde die virtuelle Maschine gestartet, mithilfe des Programmes „sync.exe“ aus der Sysinternals Suite alle ausstehenden Änderungen auf die Festplatte geschrieben und anschließend eine Kopie des Datenträgers angefertigt. Im Folgenden wurde das Betriebssystem ohne Einwirkung von außen eine Minute betrieben und nach einer erneuten Ausführung des Programmes „sync.exe“ eine zweite Kopie des Datenträgers geschrieben.

Durch einen Vergleich der beiden Datenträgerabbilder mithilfe des Programmes „*idifference2.py*“ aus dem Digital Forensics XML Projekt konnten veränderte Dateien erkannt werden. [3] Nach Zurücksetzen der virtuellen Maschine auf den Ausgangszustand wurde der beschriebene Prozess

insgesamt vier Mal wiederholt. Die Änderungen, die erkannt werden konnten, wurden zusammengeführt und als Hintergrundrauschen gespeichert.

Nun erfolgte die eigentliche Analyse der Anwendung. Dabei wurde jede der zuvor definierten Aktionen insgesamt drei Mal ausgeführt. Nach der Ausführung der jeweiligen Aktion und Speicherung der Ergebnisse, wurde die Maschine wieder in den vorherigen Stand zurücksetzt. Dabei kamen, wie bereits in Abschnitt 2.2 erläutert, die Zustands- und die Ereignismethode zur Anwendung.

Bei der ersten Ausführung wurde die jeweilige Aktion unter Anwendung der Ereignismethode beobachtet. Hierzu wurde das Programm „ProcessMonitor“ aus der „Sysinternals Suite“ [4] verwendet. Im zweiten Durchgang wurden unter Verwendung der Zustandsmethode die an der Windows Registrierungsdatenbank vorgenommenen Veränderungen erfasst. Dazu wurde jeweils vor und nach Ausführung der Aktion mithilfe des Programmes „RegShot“ [5] eine Kopie der Registry angefertigt. Die verursachten Veränderungen wurden durch einen simplen Vergleich dieser beiden Snapshots festgestellt.

Im darauffolgenden letzten Durchgang wurde, wie bereits im Zusammenhang mit der Erfassung des Hintergrundrauschens erläutert, jeweils vor und nach der Ausführung der Aktion ein Datenträgerabbild erstellt.

Nach dem Vergleich der beiden entstandenen Datenträgerabbilder mithilfe des Programmes „idifference2.py“, wurde von den festgestellten Veränderungen das zuvor aufgezeichnete Hintergrundrauschen abgezogen.

Nach Fertigstellung der Aufzeichnung wurde die Aktion ein viertes Mal ohne weitere Maßnahmen ausgeführt und nachfolgend ein neuer Snapshot der virtuellen Maschine angelegt. Dies diente dazu, einen sauberen Ausgangszustand für die Ausführung der nächsten Aktion zu erzeugen, soweit dieser von der vorherigen Aktion abhängig war. Weiterhin wurde der an diesem Punkt erstellte Snapshot zur Validierung der Erkenntnisse aus den zuvor erzeugten Aufzeichnungen genutzt.

Auf etwaige Besonderheiten während der Ausführung der einzelnen Aktionen wird in deren Beschreibung gesondert eingegangen.

Die anschließende Auswertung der Aufzeichnungen erfolgte in zwei Schritten, namentlich der Dateisystem- und Registryanalyse. Es wurden also zunächst die festgestellten Änderungen im Dateisystem betrachtet. Dazu wurden die Aufzeichnungen des „ProcessMonitor“ genutzt. Danach erfolgte die Validierung der hieraus gezogenen Schlussfolgerungen mithilfe der Ergebnisse aus dem Datenträgerabgleich (s.o., „idifference2.py“) und durch Inaugenscheinnahme der relevanten Verzeichnisinhalte im Windows-Explorer.

Im Rahmen dieses Berichtes wird lediglich auf Dateien eingegangen, welche nach der Ausführung der Aktion noch vorhanden waren. Dateien, welche während der Aktion angelegt und direkt wieder gelöscht wurden, wurden dagegen nicht berücksichtigt, da deren forensischer Nutzen als eher gering zu betrachten ist.

Nun erfolgte die Analyse der Registry. Als primäre Quelle für die Auswertung diente dabei das Ergebnis aus dem Vergleich der beiden Registry-Snapshots (s.o., „Regshot“). Die Validierung erfolgt hier sowohl mithilfe der Aufzeichnungen des „ProcessMonitor“, sowie durch eine Betrachtung der erzeugten Registry-Schlüssel mit dem Windows Registrierungs-Editor („regedit.exe“). Wie schon bei der Darstellung der Ergebnisse aus der Dateisystemanalyse, werden auch bei der Besprechung der Ergebnisse der Registryanalyse nur Spuren berücksichtigt, die für die forensische Analyse des Systems von Bedeutung sein können.

Nach Abschluss der allgemeinen Analyse erfolgte eine eingehende Betrachtung der durch das Programm erzeugten Konfigurations- und Logdateien. Auf das Vorgehen bei der Analyse dieser Daten wird in Kapitel 4 genauer eingegangen.

2.3.1 Untersuchte Anwendungsszenarien

Wie bereits erläutert, wurden im Vorfeld verschiedene Anwendungsszenarien definiert, welche typische Aktionen der Software abbilden sollen und als Ausgangspunkt für die Analyse der Spuren dienten. Tabelle 1 stellt die definierten Aktionen dar.

Aktion	Ausgangszustand	Beschreibung
Installation		Installation der Software
Login	Drive File Stream installiert	Login mit einem gültigen Benutzer
Upload einer Datei	Login erfolgt	Kopieren einer Datei in das Netzlaufwerk von Drive File Stream
Öffnen einer Datei	Upload einer Datei erfolgt	Öffnen einer Datei vom Netzlaufwerk von Drive File Stream
Löschen einer Datei	Upload einer Datei erfolgt	Löschen einer Datei vom Netzlaufwerk von Drive File Stream
Offline verfügbar machen einer Datei	Upload einer Datei erfolgt	Eine Datei vom Netzlaufwerk von Drive File Stream offline verfügbar machen
Öffnen einer Datei (offline)	Offline verfügbar machen einer Datei erfolgt	Eine offline verfügbare Datei öffnen
Löschen einer Datei (offline)	Offline verfügbar machen einer Datei erfolgt	Eine offline verfügbare Datei löschen
Deinstallation	Upload einer Datei erfolgt / Offline verfügbar machen einer Datei erfolgt	Deinstallation der Software

Tabelle 1: Untersuchte Anwendungsszenarien

2.3.2 Verwendete Software

Die Virtualisierung des Untersuchungssystems erfolgte mithilfe der Software „VirtualBox“ in der Version 5.0.20. Als Host-Betriebssystem kam dabei das Betriebssystem OS X „El Capitan“ in der Version 10.11.5 zum Einsatz.

Innerhalb des Untersuchungssystems wurde folgende Software eingesetzt:

- Windows 7 Professional N, SP 1
- Regshot, v. 1.9.0 (2013-02-02)
- Sysinternals Suite:
    - Process Monitor, v. 3.4
    - Sync, v. 2.2

Die Erstellung der jeweiligen Festplattenabbilder für die Analyse mit „idifference2.py“ erfolgte mit den im Rahmen des Moduls 105 entwickelten Python-Skripten, welche die von der Software „VirtualBox“ bereitgestellte Steuerungssoftware „VBoxManage“ nutzen.

Die Ausführung des Skriptes „idifference2.py“ erfolgte in der durch den Auftraggeber bereitgestellten virtuellen Maschine „ST_fiwalk“, welche als Betriebssystem ein Ubuntu in der Version 14.04 verwendet. Da das „Digital Forensics XML project“, welches auch „idifference2.py“ bereitstellt, keine Versionierung verwendet, kann für dieses Programm keine genaue Versionsnummer angegeben werden. Zur Analyse wurde eine aktuelle Kopie des Master-Branches des Projektes verwendet.

3 Ergebnisse der allgemeinen Analyse

Im folgenden Abschnitt sollen die Erkenntnisse aus der Analyse der Software „Drive File Stream“ vorgestellt werden. Dabei soll eine Betrachtung aller Anwendungsszenarien erfolgen, welche im Abschnitt 2.3.1 vorgestellt wurden.

Aufgrund des Umfanges der Spurenmenge werden bei der Darstellung der Ergebnisse nur die für eine weitere forensische Analyse relevanten Spuren berücksichtigt. Bei der Installation der untersuchten Software wird automatisch das Programm „Google Update“ mitinstalliert, welches für die Aktualisierung von „Drive File Stream“ verantwortlich ist. Auf Spuren von diesem Programm wird in der nachfolgenden Darstellung nicht eingegangen. [6]

3.1 Installation

3.1.1 Allgemeine Beschreibung

Die Installation der Software wurde für den Nutzer stark vereinfacht. Nach dem Starten der entsprechenden Datei öffnet sich der Installationsdialog. Der Nutzer kann hier lediglich beeinflussen, ob eine Verknüpfung auf dem Desktop angelegt werden soll oder nicht. Eine Möglichkeit den Installationspfad zu ändern besteht nicht. Abbildung 3 stellt den Dialog dar.

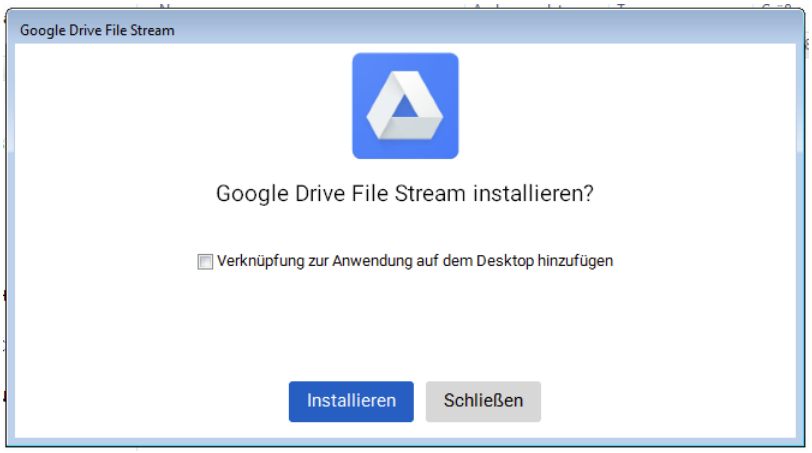


Abbildung 3: „Drive File Stream“ - Installationsdialog

Im Rahmen der Untersuchung wurden die Standardeinstellungen belassen, es wurde also keine Verknüpfung auf dem Desktop angelegt. Durch einen Klick auf „Installieren“ wird die Installation gestartet, nach deren erfolgreichem Abschluss ein Dialog zur Eingabe der für die Anmeldung bei „Google-Drive“ benötigten Account-Daten angezeigt wird. Im Installationsdialog selbst wird lediglich eine Erfolgsmeldung angezeigt, und der Nutzer hat die Möglichkeit den Installationsdialog zu schließen. Abbildung 4 zeigt die beiden Dialoge.

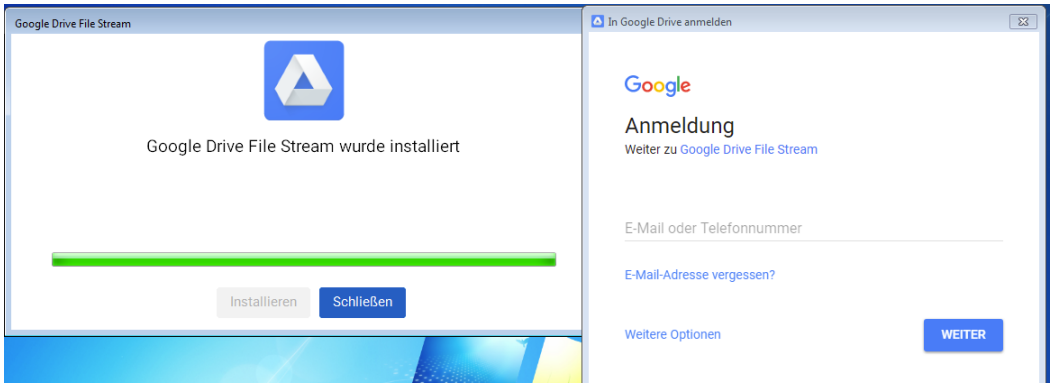


Abbildung 4: „Drive File Stream“ - Abgeschlossene Installation

3.1.2 Veränderungen am Dateisystem

Programmdateien:

Die Ablage der eigentlichen Programmdateien erfolgte im Ordner „*C:\Program Files\Google\Drive File Stream*“. In diesem Pfad wurden zwei Unterverzeichnisse erstellt. Eines enthält die eigentlichen Programmdateien und hat die Versionsnummer der Anwendung (*25.102.133.409*) als Verzeichnisnamen. Das zweite Verzeichnis trägt die Bezeichnung „*Drivers*“ und enthält im Unterverzeichnis „2220“ die benötigten Treiberdateien für das Netzlaufwerk. Die folgende Tabelle stellt die aus Sicht des Autors relevanten Ordner und Dateien im Pfad „*C:\Program Files\Google\Drive File Stream*“ noch einmal dar. Auf die Darstellung von Bibliotheksdateien oder ähnlichem wurde aufgrund des begrenzten Umfanges dieser Arbeit verzichtet.

Relativer Pfad	Dateiname	Beschreibung
Drivers\2220	googledrivefs2220.cat	Katalogdatei mit einem kryptographischen Hash für jede Datei im Treiberpaket. [7]
Drivers\2220	googledrivefs2220.inf	Installationsinformationen für den bereitgestellten Treiber
Drivers\2220	googledrivefs2220.sys	eigentliche Treiberdatei
25.102.133.409	GoogleDriveFS.exe	ausführbare Datei zum Starten der Anwendung
25.102.133.409	uninstall.exe	ausführbare Datei zum Deinstallieren der Anwendung
25.102.133.409\config	roots.pem	Zertifikatsdatei mit für Google Drive vertrauenswürdigen Rootzertifikaten
25.102.133.409\locales		Ordner mit Übersetzungsdateien für verschiedene Sprachen
25.102.133.409\html		Ordner mit verschiedenen HTML-Dateien. Enthält unter anderem das Layout für den „About“ Dialog.

Tabelle 2: Artefakte im Programmverzeichnis

Treiberdateien:

Wie bereits erwähnt, wurden neben den Programmdateien auch der für die Bereitstellung des Netzlaufwerkes benötigte Treiber installiert. Auch das kann im Dateisystem nachvollzogen werden. Die entsprechende Treiberdatei konnte unter dem Pfad „*C:\Windows\System32\drivers\googledrivefs2220.sys*“ gefunden werden.

Aus dem Inhalt der Datei „*googledrivefs2220.inf*“ ergaben sich noch weitere Informationen über den Treiber. Hierbei wurde festgestellt, dass es sich um einen Treiber der Klasse „DiskDrive“ handelt. Abbildung 5 stellt die entsprechende Stelle aus der Datei dar.

```
1  [Version]
2  Signature         = "$Windows NT$"
3  Class             = DiskDrive
4  ClassGuid         = {4d36e967-e325-11ce-bfc1-08002be10318}
5  Provider          = %ProviderName%
6  DriverVer         = %DriverVersion%
7  CatalogFile       = %DriverName%.cat
8  DriverPackageType = FileSystem
```

Abbildung 5: Auszug aus der Datei *googledrivefs2220.inf*

Des Weiteren konnte aus der Datei der genaue Name („*googledrivefs2220*“) und die Version des Treibers entnommen werden („*11/07/2017“)*. Abbildung 6 zeigt die entsprechende Stelle aus der Datei.

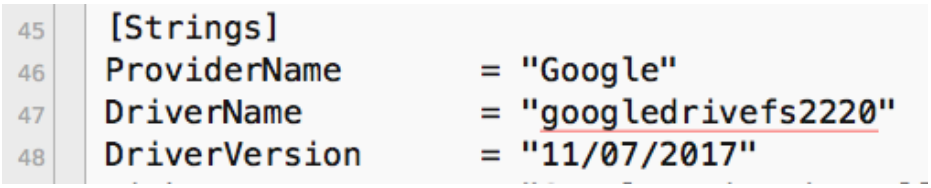


Abbildung 6: Auszug aus dem Abschnitt Strings der Datei googledrivefs2220.inf

Nutzerspezifische Dateien:

Neben dem Verzeichnis für die benötigten Programmdateien wurde bei der Installation auch eines für Konfigurationsdateien angelegt. Der zugehörige Pfad lautet „*C:\Users\Markus\AppData\Local\Google\DriveFS*“, und die folgende Tabelle gibt eine Übersicht über die in diesem Ordner befindlichen Dateien und Unterverzeichnisse.

Relativer Pfad	Dateiname	Beschreibung
	pid.txt	Textdatei mit der Prozess-ID
	lock	Textdatei, enthält den Pfad zu einer Pipe: *\\.\Pipe\GoogleDriveFSPipe_Markus*
Crashpad		Ordner, dessen Zweck nur anhand seiner Benennung vermutet werden kann, da hier nach der Installation keine relevanten Dateien gefunden werden konnten
Logs		Ordner mit verschiedenen Logdateien

Tabelle 3: Artefakte im Nutzerverzeichnis

Eine genauere Beschreibung der durch die Software geschriebenen Logdateien erfolgt im Abschnitt 4.3.

Sonstige relevante Dateien:

Die folgende Tabelle stellt sonstige Dateien dar, welche im Rahmen der Installation geschrieben wurden und Hinweise auf das Vorhandensein der Software „Drive File Stream“ auf einen Client-PC liefern können.

Pfad	Beschreibung
C:\Windows\Prefetch\GOOGLEDRIVEFSSETUP.EXE-FE88EA60.pf	Prefetch-Datei des Installers
C:\Users\Markus\AppData\Local\Temp\GoogleDFSSetup_171218132031_2260.log	Logdatei der Installation
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Drive File Stream.lnk	Startmenüverknüpfung

Tabelle 4: Weitere Installationsartefakte

Aus der im Rahmen der Installation geschriebenen Logdatei können unter anderem ihr genauer Zeitpunkt und alle in ihrem Verlauf entpackten und kopierten Dateien entnommen werden. Des Weiteren finden sich dort auch die Zeitpunkte für den Abschluss der Installation und den ersten Start der Software. Die folgende Grafik (Abbildung 7) zeigt einen Ausschnitt aus dem Beginn des Logs.

```
17-12-18T12:20:31 I setup.cc(424): Setup started.
17-12-18T12:20:31 I setup.cc(436): DriveFS version 25.102.133.409
17-12-18T12:20:31 I setup.cc(445): 32 bit mode.
17-12-18T12:20:31 I compat.cc(49): System IE major version is 9
17-12-18T12:20:31 I compat.cc(109): Wintrust.dll version is 6.1.7601.23769
17-12-18T12:20:31 I util.cc(162): Two compared versions differ only in build#
17-12-18T12:20:31 I util.cc(417): Could not read registry value SOFTWARE\Google\Update\Clients\{6BBAE539-2232-434A-A4E5-9A33560C6283},pv: 2
17-12-18T12:20:31 I omaha.cc(105): No previously installed version found.
17-12-18T12:20:31 I util.cc(793): No pending delete: SYSTEM\CurrentControlSet\Services\googledrivefs2220
17-12-18T12:20:31 I util.cc(417): Could not read registry value SOFTWARE\Google\DriveFS,DriverVersion: 2
17-12-18T12:20:45 I setup.cc(263): Gathering status
17-12-18T12:20:45 I util.cc(417): Could not read registry value SOFTWARE\Google\Update\Clients\{6BBAE539-2232-434A-A4E5-9A33560C6283},pv: 2
17-12-18T12:20:45 I omaha.cc(105): No previously installed version found.
17-12-18T12:20:45 I setup.cc(269): first_install: true
17-12-18T12:20:45 I setup.cc(283): Legacy DriveFS version was not running.
17-12-18T12:20:45 I setup.cc(287): Installing
```

Abbildung 7: Auszug aus dem Installationslog

3.1.3 Veränderungen an der Registry

Allgemeine Eintragungen:

Unter dem Registry-Schlüssel „*HKEY_LOCAL_MACHINE\SOFTWARE\Google\DriveFS*“ wurden während der Installation allgemeine Informationen zur installierten Software abgelegt. Tabelle 5 stellt die jeweiligen Werte dar:

Schlüsselname	Wert
DriverVersion	2.220.1725.0
InstallLog	C:\Users\Markus\AppData\Local\Temp\GoogleDFSSetup_171230160212_504.log

Tabelle 5: Schlüssel im Registryverzeichnis HKEY_LOCAL_MACHINE\SOFTWARE\Google\DriveFS

Die Unterschlüssel in Tabelle 6 sind im ebenfalls angelegten Registry-Schlüssel: „*HKEY_USERS\S-1-5-21-1146510048-2666936353-2363100080-1001\Software\Google\DriveFS\Share*“ abgelegt.

Schlüsselname	Wert
CopyLinkEnabled	0
MountPoint	G
ShareDialogEnabled	0
ShellIpcEnabled	0
ShellIpcPath	(nicht gesetzt)

Tabelle 6: Registry-Schlüssel im Benutzerbeich

Außerdem wurde ein weiterer Schlüssel im Pfad „*HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\googledrivefs2220*“ angelegt. Die nachfolgende Tabelle stellt dessen Werte im Einzelnen dar.

Schlüsselname	Wert
Description	Google Drive kernel-mode file system driver
DisplayName	googledrivefs2220
ErrorControl	1
Group	File System
ImagePath	system32\DRIVERS\googledrivefs2220.sys
Start	1
Tag	1
Type	2

Tabelle 7: Schlüssel im ControlSet001

Im untergeordneten Schlüssel „*Enum*“ sind folgende Werte abgelegt:

Schlüsselname	Wert
0	Root\LEGACY_GOOGLEDRIVEFS2220\0000
Count	1
NextInstance	1

Tabelle 8: "Enum" im ControlSet001

Änderungen in der Windows Explorer Ansicht:

Wie bereits in Abschnitt 2.1 dargestellt, symbolisiert die Anwendung den Synchronisationsstatus der einzelnen Dateien mithilfe kleiner Icons im Explorer Fenster. Diese Icons werden über Einträge in der Windows-Registry realisiert. Zu diesem Zweck wurden im Pfad: „*HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ShellIconOverlayIdentifiers*“ drei weitere Schlüssel angelegt:

Schlüsselname	Wert
GoogleDriveCloudOverlayIconHandler	{7CB4D2F7-77AE-4A08-9BDF-21370FF8D6BD}
GoogleDrivePinnedOverlayIconHandler	{C9F7D7A1-D13F-4C72-9AB0-06FDC65AA931}
GoogleDriveProgressOverlayIconHandler	{96836CC1-31EA-4F1C-A7F4-D67863D5D4FD}

Tabelle 9: Shell-Icon-Definition in der Registry

Die angegebenen Schlüssel-Werte verweisen dabei auf den Registry-Pfad: „*HKEY_USERS\S-1-5-21-1146510048-2666936353-2363100080-1001_Classes\CLSID*“, wobei sich an dieser Stelle für jede ID ein gleichnamiger Unterschlüssel befindet. In diesem befindet sich jeweils ein Eintrag mit dem Namen „*InProcServer32“*, welcher einen Verweis auf die Bibliothek enthält, in der sich das Icon befindet. Die folgende Abbildung zeigt einen entsprechenden Auszug aus der Registry:

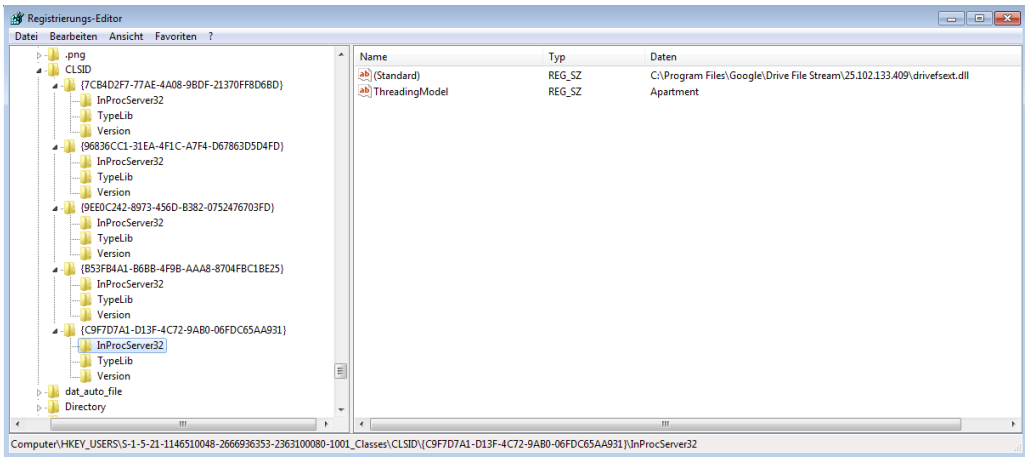


Abbildung 8: Auszug aus dem Registry-Ordner CLSID

Unter „*HKEY_USERS\S-1-5-21-1146510048-2666936353-2363100080-1001\Software\Classes*“ wurden zusätzliche Dateiendungen registriert. Für jede dieser neu angelegten Endungen wurde ein „Default-Icon“ (Schlüsselname: „DefaultIcon“) sowie ein Standardbefehl für das Öffnen einer solchen Datei (Schlüsselpfad: „*shell/open/command“*) definiert. Dieser ist jedoch bei jedem Dateityp identisch und lautet: „*C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe" --open_gdoc="%1*“

Die folgende Tabelle stellt die einzelnen Endungen dar.

Endung	Default-Icon
gdoc	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-2
gdraw	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-3
gform	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-4
glink	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-5
gmap	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-6
gnote	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-7
gscript	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-8
gsheet	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-9
gsite	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-12
gslides	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-10
gtable	"C:\Program Files\Google\Drive File Stream\25.102.133.409\GoogleDriveFS.exe",-11

Tabelle 10: Registrierte Dateiendungen

Zusätzlich erfolgte für jede der in der Tabelle dargestellten sowie die bereits bestehenden für Bilder verwendeten Dateiendungen „bmp“, „jpg“, „jpeg“ und „png“ die Registrierung einer „Shell Extension“ mit dem Schlüsselnamen *„HKEY_USERS\S-1-5-21-1146510048-2666936353-2363100080-1001_Classes\<DATEIENDUNG>\shellex\{E357FCCD-A995-4576-B01F-234630154E96}“* und dem Wert *„{9EE0C242-8973-456D-B382-0752476703FD}*“. Ebenso wie beim Icon Handler verbirgt sich hinter dem Wert ein Verweis auf die Bibliothek „*drivefsext.dll“*.

Es wurde auch für alle Dateitypen eine „Context Menu Handlers“ registriert. Dies erfolgte unter dem Eintrag *„HKEY_USERS\S-1-5-21-1146510048-2666936353-2363100080-1001\Software\Classes*\shellex\ContextMenuHandlers\DriveFS*“ mit dem Wert „{B53FB4A1-B6BB-4F9B-AAA8-8704FBC1BE25}“. Dieser verweist ebenfalls auf „*drivefsext.dll“*.

3.2 Login

3.2.1 Allgemeine Beschreibung

Nach der Installation wird der Benutzer aufgefordert, sich mit dem gewünschten Google Account anzumelden. Nach der Eingabe von Nutzernamen und Passwort erfolgt die initiale Synchronisation des Rechners mit den Daten aus der Cloud. Anschließend kann der Anwender über ein nun verbundenes Netzlaufwerk mit dem Namen „Google Drive File Stream (G:)“ auf die Daten zugreifen.

Im Wurzelverzeichnis dieses Netzlaufwerkes befinden sich standardmäßig die beiden Ordner „*Meine Ablage*“ und „*Teamablagen*“. Eventuell in den Ablagen vorhandene Dateien werden zu diesem Zeitpunkt zwar angezeigt, sind aber nicht offline verfügbar und werden daher erst bei Bedarf dynamisch nachgeladen. Abbildung 9 zeigt das Explorer Fenster nach der initialen Synchronisation.

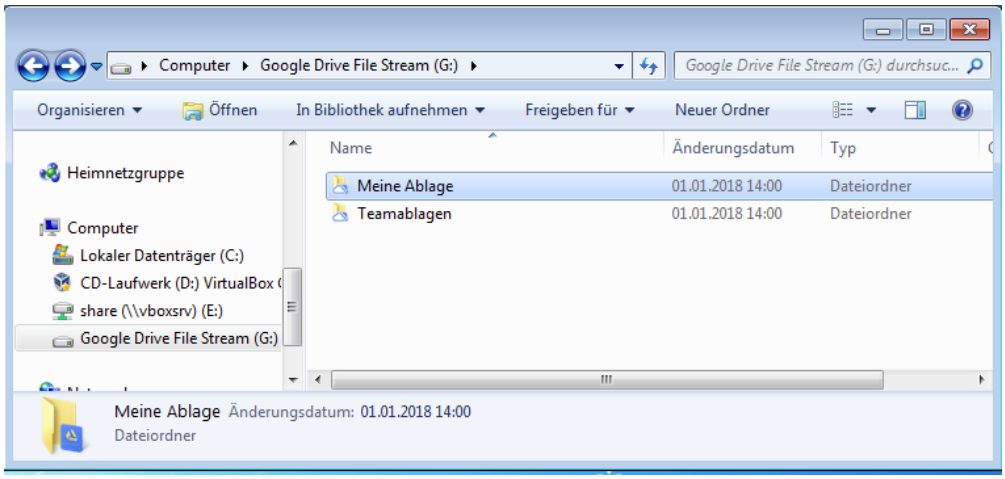


Abbildung 9: Netzlaufwerk "Google Drive File Stream" nach erstem Login

3.2.2 Veränderungen im Dateisystem

Bei der Betrachtung des Dateisystems konnten Veränderungen im Konfigurationsordner („„*C:\Users\Markus\AppData\Local\Google\DriveFS*“) der Anwendung festgestellt werden. Hier wurden verschiedene neue Dateien, sowie ein zusätzliches Verzeichnis erstellt. Die folgende Tabelle gibt einen Überblick über die neu erstellten Dateien und deren Inhalt.

Name	Beschreibung
active_mount_point	Textdatei, welche den Laufwerksbuchstaben des Netzlaufwerkes für Google Drive enthält (G)
features	Binärdatei
signin	Datei, welche den Namen des neu angelegten Ordners in dem Verzeichnis enthält (dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWluLmRl)

Tabelle 11: Neue Dateien im Benutzerverzeichnis

Außerdem wurde im Konfigurationsverzeichnis ein weiterer Ordner mit dem Namen „*dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWluLmRl*“ angelegt, dessen Inhalt in der nachfolgenden Tabelle dargestellt wird. Die Darstellung beschränkt sich hierbei auf persistente Dateien. Temporäre Bearbeitungsdateien, wie sie beispielsweise durch eine offene SQLite-Datenbank erzeugt werden, werden hingegen nicht dargestellt.

Relativer Pfad	Dateiname	Beschreibung
	metadata_sqlite_db	SQLite-Datenbank
	metadata_sqlite_db_local_counter	SQLite-Datenbank
	metrics_sqlite_db	SQLite-Datenbank
local_folders		leerer Ordner
content_cache		Ordner, welcher 67 Unterordner enthält, die nach dem Schema d0 bis d66 benannt sind
content_cache	chunks.db	SQLite-Datenbank
thumbnails_cache		Ordner, welcher 7 Unterordner enthält, die nach dem Schema d0 bis d6 benannt sind
thumbnails_cache	chunks.db	SQLite-Datenbank

Tabelle 12: Dateien im nach dem Login angelegten Ordner

Eine genauere Betrachtung der in diesem Zusammenhang erstellten SQLite-Datenbanken, sowie der erstellten Cache Ordner erfolgt im Abschnitt 4.

Im Ordner „*C:\Users\Markus\AppData\Local\Google\DriveFS\Logs*” wurde zudem noch eine weitere Logdatei mit dem Namen “*drive_fs.txt“* erstellt, sowie die Datei „*chrome_debug.log*“ verändert.

3.2.3 Veränderungen an der Registry

An der Registry konnten zwei Veränderungen festgestellt werden, welche im Zusammenhang mit der Erzeugung des Netzlaufwerkes stehen. Im Pfad „*HKLM\SYSTEM\MountedDevices*“ wurden folgende Einträge hinzugefügt:

- \??\Volume{f11baffc-e3ec-11e7-8cc6-0800274ad52e}
- \DosDevices\G:

Weitere untersuchungsrelevante Änderungen konnten bei der Analyse des Loginverhaltens nicht festgestellt werden.

3.3 Upload einer Datei

3.3.1 Allgemeine Beschreibung

In der nächsten Phase des Testes wurde eine Textdatei („*upload.txt*“) in das Verzeichnis „*G:\Meine Ablage*“ kopiert. Die Datei wurde im Folgenden weder geöffnet noch anderweitig verwendet. Bevor mit der Auswertung begonnen wurde, wurde lediglich gewartet, bis durch das Tray-Icon der Anwendung symbolisiert wurde, dass der Upload abgeschlossen war.

Abbildung 10 zeigt den Dialog der Anwendung nach dem erfolgreichen Upload der Datei.

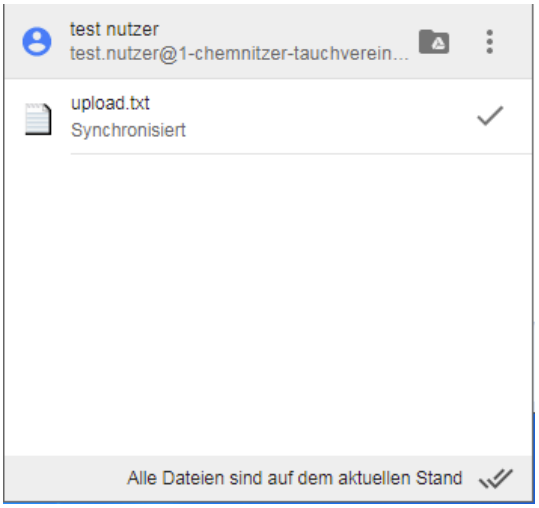


Abbildung 10: Tray-Fenster nach Upload einer Datei

3.3.2 Veränderungen am Dateisystem

Nach einem erfolgreichen Upload konnte beobachtet werden, dass der Inhalt der Datei „C:*Users\Markus\AppData\Local\Google\DriveFS\Logs\drive_fs.txt*“ verändert wurde.

Zudem wurden im Pfad „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWluLmRl*“ die „Write-Ahead-Logs“ der folgenden SQLite Datenbanken verändert:

- *metadata_sqlite_db-wal*
- *metadata_sqlite_db_local_counter-wal*
- *metrics_sqlite_db-wal*
- *content_cache\chunks.db-wal*

Nach dem Beenden der Anwendung „Drive File Stream“ konnte im untersuchten System beobachtet werden, dass die Änderungen aus dem „*Write-Ahead-Log*“ in die jeweilige SQLite Datenbank übernommen wurden.

Zusätzlich konnte beobachtet werden, dass im Pfad „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWIuLmRl\content_cache\d36*“ ein neues Verzeichnis mit dem Namen „*d103*“ erstellt wurde, welches eine Datei („*103*“) enthält. Der Inhalt dieser Datei entsprach dem Inhalt der hochgeladenen Datei „*upload.txt*“. Abbildung 11 veranschaulicht dies.

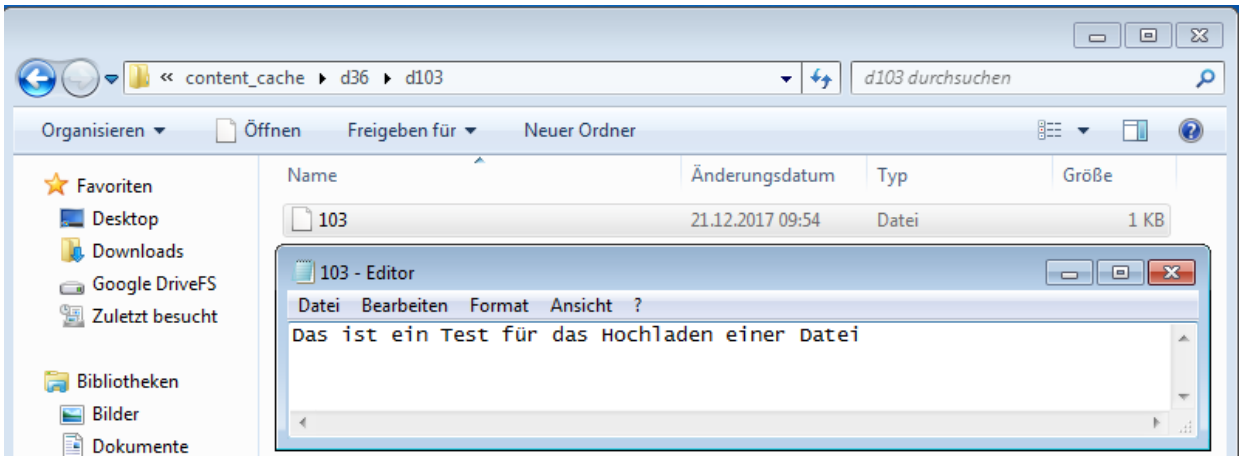


Abbildung 11: Erzeugte Cache-Datei

Außerdem konnten auch Veränderungen an der Datei „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWIuLmRl\account_settings*“ festgestellt werden.

3.3.3 Veränderungen an der Registry

Änderungen an der Registry konnten nicht festgestellt werden.

3.4 Öffnen einer Datei

3.4.1 Allgemeine Beschreibung

Hier wird untersucht, wie sich die Anwendung beim Öffnen der zuvor hochgeladenen Datei „*upload.txt*“ und einer Datei „*upload_1.txt*“ verhält, welche mittels Webbrowser dem Cloudspeicher hinzugefügt wurde. Nach der erfolgreichen Synchronisation der Anwendung wurde mit der Erstellung der Aufzeichnung begonnen.

Zunächst wurde die Datei „*upload.txt*“ durch einen Doppelklick geöffnet und der Inhalt der Datei betrachtet.

Nach einer Dauer von 20 Sekunden wurden die Datei und das geöffnete Explorer Fenster wieder geschlossen. Nach weiteren 20 Sekunden wurde die Analyse der Dateien beendet und mit der Auswertung begonnen.

Dieser Test wurde dann mit der Datei „*upload_1.txt*“ wiederholt.

Abbildung 12 zeigt die geöffnete Datei „*upload_1.txt*“.

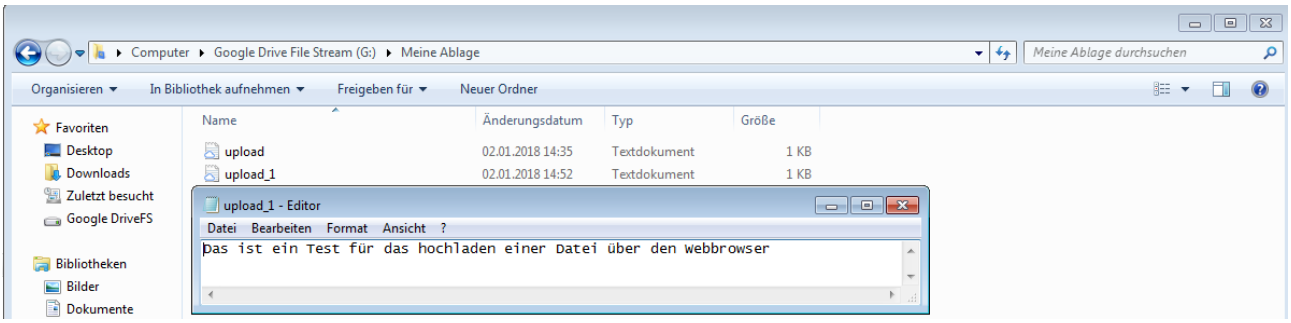


Abbildung 12: Datei "*upload_1.txt*"

3.4.2 Veränderungen am Dateisystem

Bei beiden Vorgängen konnten Änderungen an der Datei „C:*Users\Markus\AppData\Local\Google\DriveFS\Logs\drive_fs.txt*“ beobachtet werden.

Veränderungen beim Betrachten der Datei „*upload.txt*“:
Beim Betrachten dieser Datei wurden unter dem Pfad „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWluLmRl*“ Änderungen an dem „Write-Ahead-Logs“ der SQLite Datenbank „*metadata_sqlite_db-wal*“ bewirkt.

Zusätzlich konnte durch den ProcessMonitor ein lesender Zugriff auf die Datei „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWluLmRl\content_cache\d36\d103\103*“ beobachtet werden.

Veränderungen beim Betrachten der Datei „*upload_1.txt*“:

Ähnlich wie bei dem Fall des Uploads einer Datei konnten auch hierbei Änderungen im Pfad „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWluLmRl*“ festgestellt werden. Hier erfolgte eine Änderung der „Write-Ahead-Logs“ der folgenden SQLite Datenbanken:

- *metadata_sqlite_db-wal*
- *metadata_sqlite_db_local_counter-wal*
- *content_cache\chunks.db-wal*

Zusätzlich wurde beobachtet, dass im Pfad „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXR*

hdWNodmVyZWluLmRl\content_cache\d42“ der Ordner „*d109*“ erstellt wurde, welcher die Datei „*109*“ enthält. Diese enthielt denselben Text wie die Datei „*upload_1.txt“.*

3.4.3 Veränderungen an der Registry

Anwendungsspezifische Änderungen an der Registry konnten in beiden Fällen nicht beobachtet werden.

3.5 Löschen einer Datei

3.5.1 Allgemeine Beschreibung

Nach dem Zurücksetzen der virtuellen Maschine auf denselben Ausgangszustand wie im Fall „Öffnen einer Datei“ (s. 3.3), erfolgte zunächst das Löschen der Datei „*upload.txt“*. Danach wurde gewartet, bis die durch das Löschen ausgelöste Synchronisation abgeschlossen und die Datei auch in der Webanzeige von Google Drive nicht mehr vorhanden war, und die Synchronisierung in der Tray-Anzeige der Anwendung als beendet angezeigt wurde.

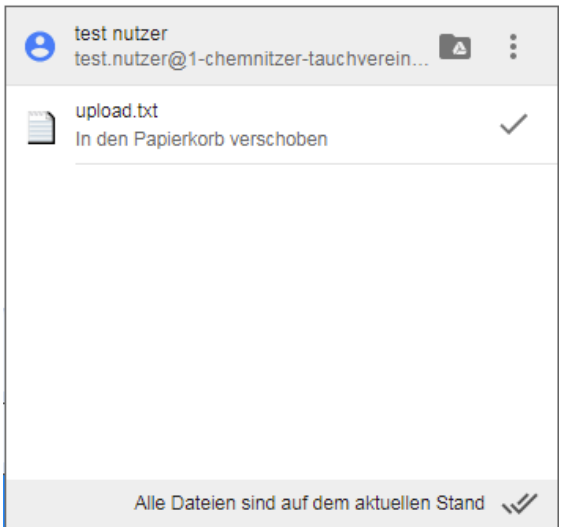


Abbildung 13: Löschen einer Datei

Der Test wurde dann mit der Datei „*upload_1.txt“*. wiederholt. Abbildung 13 zeigt den Zustand der Tray-Anzeige nach dem Löschen der Datei „*upload.txt“*.

3.5.2 Veränderungen am Dateisystem

Bei beiden Vorgängen konnten Änderungen an der Datei „C:*Users\Markus\AppData\Local\Google\DriveFS\Logs\drive_fs.txt*“ beobachtet werden.

Veränderungen beim Löschen der Datei „*upload.txt“*:
Wie auch in den vorherigen Szenarien traten Änderungen der „Write-Ahead-Logs“ der folgenden SQLite Datenbanken ein:

- *metadata_sqlite_db-wal*
- *metrics_sqlite_db-wal*
- *content_cache\chunks.db-wal*

Außerdem wurde auch die beim Upload von „upload.txt“ angelegte Textdatei „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWIuLmRl\content_cache\d36\d103\103*“ gelöscht.

Veränderungen beim Löschen der Datei „*upload_1.txt*“:
Abweichend von den Feststellungen beim Löschen der Datei „*upload.txt*“ erfolgte hier lediglich eine Änderung der „Write-Ahead-Logs“ der folgenden SQLite Datenbanken:

- *metadata_sqlite_db-wal*
- *metrics_sqlite_db-wal*

3.5.3 Veränderungen an der Registry

Anwendungsspezifische Änderungen an der Registry konnten nicht beobachtet werden.

3.6 Offline verfügbar machen einer Datei

3.6.1 Allgemeine Beschreibung

Als nächster Analyseschritt wurde der Zustand vor dem Öffnen einer Datei (s. 3.4) wiederhergestellt. Die zuvor hochgeladene Datei „*upload.txt*“ wurde offline verfügbar gemacht. Dazu wurde im Windows Explorer über das Kontext-Menü dieser Datei der Punkt „Offline verfügbar“ im Menü „Drive File Stream“ ausgewählt (s. Abbildung 14).

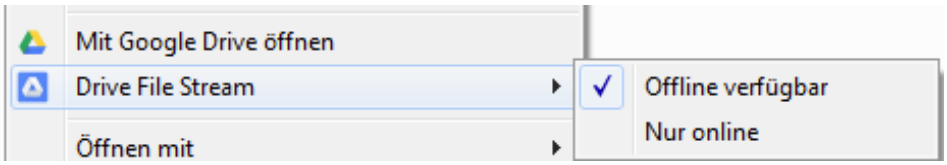


Abbildung 14: Kontext-Menü der Anwendung "Drive File Stream"

Danach wurde abgewartet, bis die Synchronisation abgeschlossen war und durch das grüne Häkchen im Icon der Datei symbolisiert wurde, dass die untersuchte Datei nun offline verfügbar ist (s. Abbildung 15). Ein Öffnen der Datei erfolgte nicht.

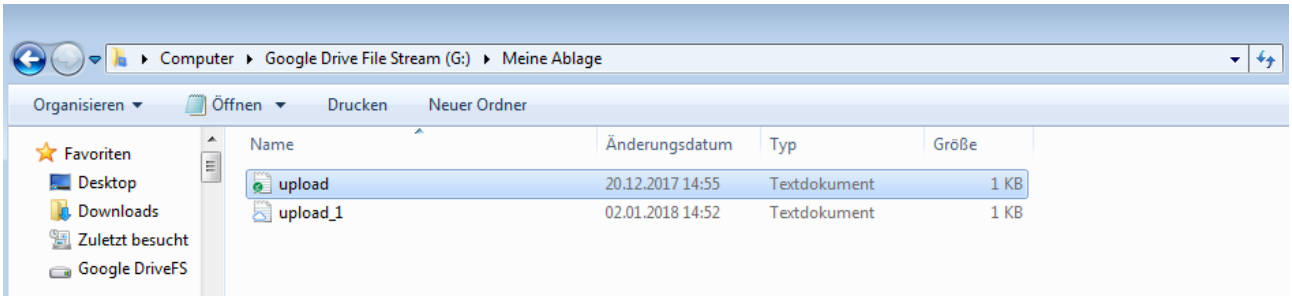


Abbildung 15: Netzlaufwerk mit offline verfügbarer Datei

Abschließend wurde das Explorer-Fenster wieder geschlossen.

3.6.2 Veränderungen am Dateisystem

Wie aufgrund der Feststellungen aus den vorherigen Szenarien zu erwarten war, erfolgte eine Veränderung der Log-Datei „C:*Users\Markus\AppData\Local\Google\DriveFS\Logs\drive_fs.txt*“.

Es wurden auch hier die „Write-Ahead-Logs“ verschiedener SQLite Datenbanken verändert (Pfad: „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWIuLmRl*“). In diesem Fall waren das:

- *metadata_sqlite_db-wal*
- *metrics_sqlite_db-wal*
- *metadata_sqlite_db_local_counter-wal*
- *content_cache\chunks.db-wal*

Zusätzlich konnte in diesem Szenario beobachtet werden, dass im Pfad „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWIuLmRl\content_cache\d42*“ ein neuer Ordner mit der Bezeichnung „*d109*“ erstellt wurde, welcher die Datei „*109*“ enthält. Der Inhalt der erzeugten Datei war identisch mit dem Inhalt von „*upload.txt*“.

3.6.3 Veränderungen an der Registry

Anwendungsspezifische Änderungen an der Registry konnten auch hier nicht beobachtet werden.

3.7 Öffnen einer Datei (offline)

3.7.1 Allgemeine Beschreibung

Die im vorherigen Szenario offline verfügbar gemachte Datei wurde nun im Windows Explorer durch einen Doppelklick geöffnet. Ähnlich wie in Punkt 3.4 beschrieben, verblieb die Datei für 20 Sekunden in geöffnetem Zustand. Danach wurde sie und anschließend auch das Explorer-Fenster geschlossen.

Eine Veränderung oder ein Speichern der Datei erfolgte nicht. Abbildung 16 zeigt die geöffnete Datei.

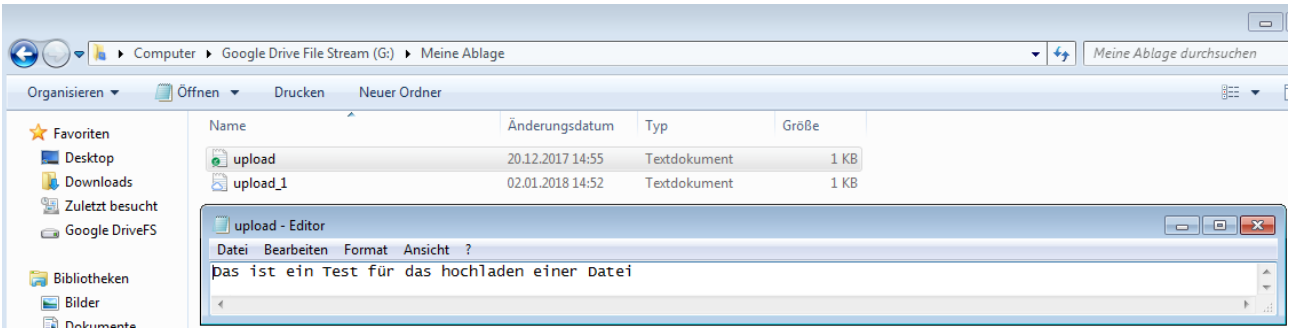


Abbildung 16: Hochgeladene und offline verfügbare Datei "*upload.txt*"

3.7.2 Veränderungen am Dateisystem

Erwartungsgemäß wurde auch hier die Log-Datei „C:*Users**Markus**AppData**Local**Google**DriveFS**Logs**drive_fs.txt*“ verändert.

Zusätzlich traten Änderungen an den „Write-Ahead-Logs“ folgender SQLite Datenbanken im Pfad „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWIuLmRl*“ auf:

- *metadata_sqlite_db-wal*
- *metrics_sqlite_db-wal*

Es konnte zudem festgestellt werden, dass auf die Datei “*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWIuLmRl\content_cache\d42\d109\109”* ein lesender Zugriff erfolgte.

3.7.3 Veränderungen an der Registry

Anwendungsspezifische Änderungen an der Registry wurden nicht beobachtet.

3.8 Löschen einer Datei (offline)

3.8.1 Allgemeine Beschreibung

Hier wurde für die Analyse die Datei „*upload.txt*“ im Windows Explorer gelöscht. Wie auch im Szenario in Punkt 3.5 wurde abgewartet, bis die durch das Löschen ausgelöste Synchronisation abgeschlossen und die Datei auch in der Webanzeige von Google Drive nicht mehr vorhanden war, sowie dass die Synchronisierung in der Tray-Anzeige der Anwendung als beendet angezeigt wurde. Abbildung 17 zeigt den Zustand der Tray-Anzeige nach dem Löschen der Datei.

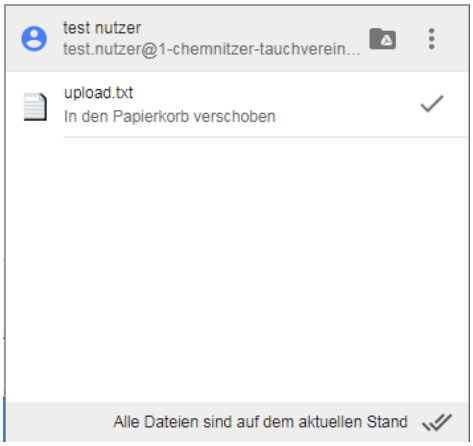


Abbildung 17: Löschen einer offline verfügbaren Datei

3.8.2 Veränderungen am Dateisystem

Auch nach dieser Aktion wurde die Log-Datei „C:*Users\Markus\AppData\Local\Google\DriveFS\Logs\drive_fs.txt*“ verändert.

Des Weiteren wurden unter dem Pfad „*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWluLmRl*“ die „Write-Ahead-Logs“ folgender SQLite Datenbanken verändert:

- *metadata_sqlite_db-wal*
- *metrics_sqlite_db-wal*
- *content_cache\chunks.db-wal*

Außerdem wurde die Datei “*C:\Users\Markus\AppData\Local\Google\DriveFS\dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWluLmRl\content_cache\d42\d109\109”* gelöscht.

3.8.3 Veränderungen an der Registry

Anwendungsspezifische Änderungen an der Registry wurden auch bei dieser Aktion nicht beobachtet.

3.9 Deinstallation der Anwendung

Im letzten Szenario dieser Analyse erfolgte die Deinstallation des Programmes „Drive File Stream“. Dazu musste zunächst die noch laufende Anwendung beendet werden. Danach wurde der Deinstallationsprozess über die Systemsteuerung gestartet. Ähnlich wie bei der im Punkt 3.1

beschriebenen Installation konnten in diesem Dialog keine weiteren Einstellungen gesetzt werden (s. Abbildung 18).

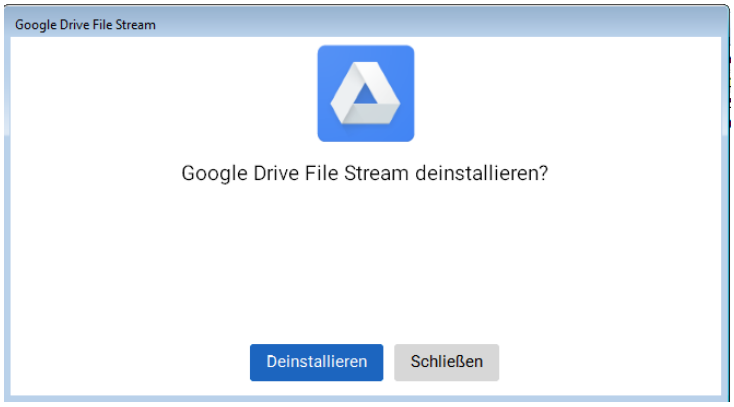


Abbildung 18: Deinstallationsdialog

Nach erfolgreichem Abschluss der Deinstallation wurde das Dialogfensters geschlossen, das System neu gestartet und die Auswirkungen auf das System analysiert.

3.9.1 Veränderungen am Dateisystem

Nach der Deinstallation:

Es konnte beobachtet werden, dass nach der Deinstallation im Verzeichnis „C:\Users\Markus\AppData\Local\Temp\“ zwei zusätzliche Logdateien angelegt wurden. Hierbei handelt es sich um:

- *google_drivefs_uninstall_171222101012_3396.exe*
- *GoogleDFSSetup_171222101012_3396.log*

Wie im Rahmen einer Deinstallation zu erwarten, erfolgte eine fast vollständige Löschung der Programmdaten im Pfad „*C:\Program Files\Google\Drive File Stream*“. Das bei der Installation angelegte Verzeichnis „*Drivers“* wurde ebenfalls vollständig gelöscht. Der Ordner „*25.102.133.409*“ verblieb jedoch weiterhin im Dateisystem. Er enthielt noch die folgenden Bibliotheksdateien:

- *cc_icu_data_library_core.dll*
- *common_icuuc.dll*
- *drivefsext.dll*
- *icui18n.dll*

Das Verzeichnis enthielt außerdem eine neu hinzugekommene Textdatei mit dem Namen „*rebootpending*”, deren Inhalt aus den Worten „reboot required“ besteht. Änderungen im Ordner für

die Konfigurationsdateien („*C:\Users\Markus\AppData\Local\Google\DriveFS*“) konnten nicht festgestellt werden.

Nach dem Neustart:
Nach dem Neustart des Rechners war der Ordner „*C:\Program Files\Google\Drive File Stream*“ vollständig gelöscht, am Verzeichnis für die Konfigurationsdateien konnten jedoch keine Veränderungen beobachtet werden.

3.9.2 Veränderungen an der Registry

Nach der Deinstallation:
Der Registry-Schlüssel „*HKEY_LOCAL_MACHINE\SOFTWARE\Google\DriveFS*“ wurde während der Deinstallation gelöscht.

Der Eintrag „*HKEY_USERS\S-1-5-21-1146510048-2666936353-2363100080-1001\Software\Google\DriveFS\Share*“ war dagegen noch existent und unverändert.

Unter dem Registry-Eintrag „*HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\googledrivefs2220*“ waren weitere Veränderungen feststellbar. Hier wurde ein neuer Schlüssel mit dem Namen „*DeleteFlag“* hinzugefügt. Außerdem wurde der Wert des Schlüssels „*Start“* verändert. Die nachfolgende Tabelle stellt die Änderungen dar.

Schlüsselname	Alter Wert	Neuer Wert
Start	1	4
DeleteFlag		1

Tabelle 13: Änderungen am Registry-Schlüssel "*HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\googledrivefs2220*"

Der Unterschlüssel mit der Bezeichnung „*Enum*“ blieb unverändert.

Die bei der Installation angelegten Schlüssel

- GoogleDriveCloudOverlayIconHandler
- GoogleDrivePinnedOverlayIconHandler
- GoogleDriveProgressOverlayIconHandler

welche sich im Pfad „*HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ShellIconOverlayIdentifiers* “ befunden haben, waren nicht mehr vorhanden.

Auch die entsprechenden Unterschlüssel in „*HKEY_USERS\S-1-5-21-1146510048-2666936353-2363100080-1001_Classes\CLSID*“ waren gelöscht. Entsprechend wurden auch die in Abschnitt 3.1.3 beschriebenen, zusätzlich registrierten Dateiendungen, sowie die Definition der dazugehörigen

Default-Icons (Schlüsselname: „DefaultIcon“) und der Schlüssel mit dem Standardbefehl für das Öffnen dieser Datei (Schlüsselpfad: „*shell/open/command*“) entfernt.

Gleiches gilt auch für die für Bilddateierweiterungen („.bmp“, „.jpg“, „.jpeg“, .png“) registrierte „Shell Extension“ und den o.g. „Context Menu Handler“ unter „*HKEY_USERS\S-1-5-21-1146510048-2666936353-2363100080-1001\Software\Classes*\shellex\ContextMenuHandlers\DriveFS*“.

Nach dem Neustart:
Der Registry-Schlüssel „*HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\googledrivefs2220*“ war nun vollständig gelöscht.

Weitere Änderungen an der Registry, konnten nicht festgestellt werden.

3.10 Erkenntnisse aus der Analyse

Die allgemeine Analyse ergab folgende Erkenntnisse:

Während der Installation der Software werden Programmdateien im Verzeichnis „*C:\Program Files\Google\Drive File Stream“* abgelegt. Zusätzlich zum korrekten Einbinden des virtuellen Netzlaufwerkes ein Treiber installiert, welcher für die Bereitstellung der Clouddaten verantwortlich ist. Nutzerspezifische Dateien werden während der Installation und vor allem nach dem Login im Verzeichnis „*C:\Users\Markus\AppData\Local\Google\DriveFS“* abgelegt.

Innerhalb der Windows Registrierungsdatenbank werden lediglich allgemeine Informationen zur Anwendung abgelegt. Über die synchronisierten Dateien selbst sind dort keinerlei Informationen verfügbar. Es wird hier lediglich noch eine Vielzahl zusätzlicher Dateiendungen registriert.

Während der verschiedenen untersuchten Aktivitäten konnten Änderungen im Logfile „C:*Users\Markus\AppData\Local\Google\DriveFS\Logs\drive_fs.txt*“ beobachtet werden. Zusätzlich wurden verschiedene SQLite Datenbanken genutzt. Diesen beiden Punkten widmet sich der nächste Abschnitt eingehender.

Bei der Deinstallation der Anwendung werden die Programmdateien vollständig, die in der Registry gespeicherten Informationen zu großen Teilen gelöscht. Die nutzerspezifischen Dateien unter „*C:\Users\Markus\AppData\Local\Google\DriveFS“* bleiben jedoch unberührt.

4 Betrachtung der benutzerspezifischen Dateien

Hier sollen die im Verzeichnis „*C:\Users\Markus\AppData\Local\Google\DriveFS“* abgelegten SQL-Datenbanken und die bei den in Abschnitt 3 durchgeführten Aktionen geschriebenen Cache-Dateien eingehend betrachtet werden. Weiterhin wird kurz auf die Log-Dateien eingegangen.

4.1 Betrachtung der SQLite-Datenbanken

Im Ordner „*dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWluLmRl*“, welcher nach dem Login des Benutzers angelegt wurde, befinden sich insgesamt drei verschiedene SQLite Datenbanken, deren Inhalt im Folgenden kurz dargestellt und analysiert werden soll. Bei den Datenbankdateien handelt es sich um „*metrics_sqlite_db“, „metadata_sqlite_db_local_counter“* und „*metadta_sqlite_db“*.

Neben den durch die Aktionen in Abschnitt 3 erzeugten verschiedenen Datenbankzuständen wurden im Rahmen der Analyse noch weitere Zustände erzeugt. Diese werden benötigt, um die geschriebenen Datenbanken vollumfänglich analysieren können. Dazu wurde unter anderem eine Datei über die Google Drive Weboberfläche markiert, ein zusätzliches Verzeichnis angelegt, darin weitere Text-Dateien angelegt und diese dann während der Analyse bearbeitet (Hinzufügen eines Wortes) und über die lokale Anwendung und den Web-Client gelöscht.

Weiterhin wurde durch einen anderen Benutzer eine PDF-Datei für den im Rahmen dieser Arbeit verwendeten Benutzer „*Test Nutzer*“ freigegeben. Diese Datei wurde in der Folge durch den „*Test Nutzer*“ zu seiner Ablage hinzugefügt.

Im letzten Schritt wurde zudem ein „TeamDrive“ mit dem Namen „*Test*“ erstellt und auch in diesem eine Datei gespeichert.

4.1.1 metrics_sqlite_db

Diese Datenbank enthält lediglich eine Tabelle mit dem Namen „*log_event_table“*, die aus den beiden Spalten „*id“* und „*proto“* besteht. Nach allen Aktionen enthielt diese Tabelle nur eine Zeile mit der ID 1 und einem Binärdatenstrom in der Spalte „*proto“*. Aufgrund des begrenzten Zeitumfangs und der aus der Analyse der anderen Datenbanken gewonnen Erkenntnisse, wurde auf eine Analyse der gespeicherten Binärdaten verzichtet. Abbildung 19 zeigt einen Auszug aus dem Binärdatenstrom dieser Zeile.

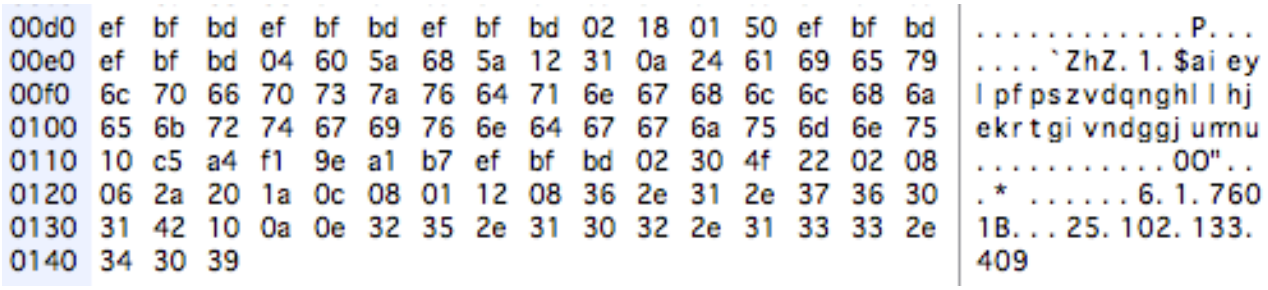


Abbildung 19: Auszug Spalte "*proto*"

4.1.2 metadata_sqlite_db_local_counter

Auch diese Datenbank enthielt lediglich eine Tabelle, diesmal allerdings mit dem Namen „*properties“*. Auch sie beinhaltet zwei Spalten, namentlich „*property“* und „*value“*. Bei allen betrachteten Szenarien enthielt die Tabelle lediglich eine Zeile. In dieser enthielt *„property“* den String „local_counter“ und *„value“* einen Integer, dessen Wert nach dem Login 102 betrug und mit jeder weiteren Aktion anwuchs.

4.1.3 metadata_sqlite_db

Die Tabelle “*metadata_sqlite_db”* scheint in der Anwendung als zentrale Datenbank für alle in Google Drive vorhandenen Dateien zu dienen. In ihr selbst befinden sich insgesamt zwölf verschiedene Tabellen deren Inhalt und Aufbau in diesem Abschnitt näher betrachtet werden sollen. Ein vollständiges Datenbankmodell befindet sich im Anhang 1. Abbildung 20 zeigt die in der Datenbank vorhanden Tabellen.

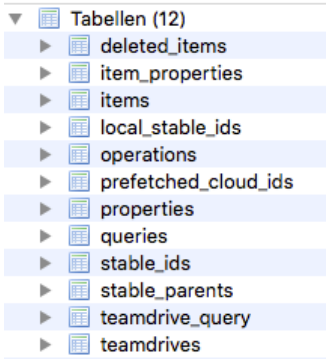


Abbildung 20: Tabellen in der Datenbank "metadata_sqlite_db"

Während der gesamten Untersuchung konnte in folgenden Tabellen kein Inhalt festgestellt werden:

- „*deleted_items“*
- „*local_stable_ids“*
- *„prefetched_cloud_ids“*
- „*operations“*
- *„teamdrive_query“*
- „*teamdrives“*

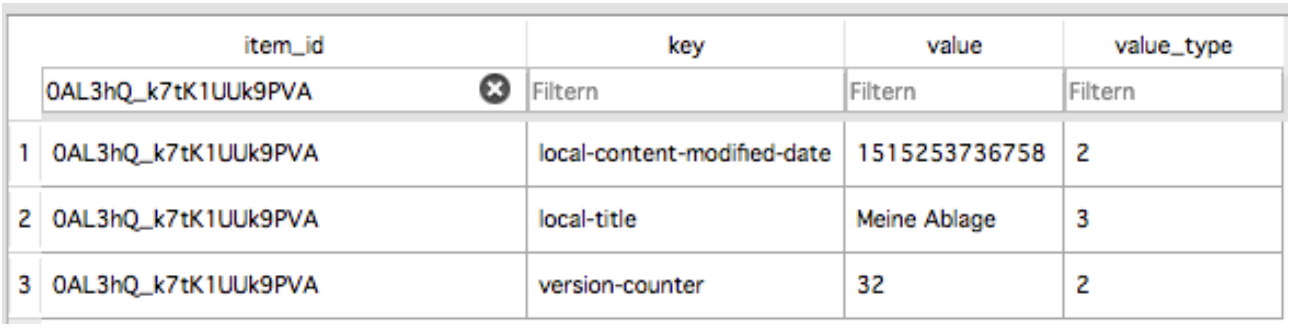
Tabelle „item_properties“

Diese Tabelle besitzt 4 Spalten, welche in der nachfolgenden Tabelle dargestellt werden.

Spaltenname	Beschreibung
item_id	Google Drive ID des betreffenden Objektes
key	Schlüssel
value	Wert
value_type	Datentyp des Wertes

Tabelle 14: Tabelle "item_properties"

Initial enthielt diese Tabelle lediglich einen Eintrag für den Hauptordner „Meine Ablage“ in welchem alle im Rahmen der Untersuchung verwendeten Testdaten abgelegt wurden. Abbildung 21 stellt diesen Eintrag dar.



	item_id	key	value	value_type
	0AL3hQ_k7tK1UUk9PVA	Filtern	Filtern	Filtern
1	0AL3hQ_k7tK1UUk9PVA	local-content-modified-date	1515253736758	2
2	0AL3hQ_k7tK1UUk9PVA	local-title	Meine Ablage	3
3	0AL3hQ_k7tK1UUk9PVA	version-counter	32	2

Abbildung 21: Einträge für den Ordner "Meine Ablage"

Die „*item_id*“ entspricht der internen Google Drive ID des Objektes. Die Spalte „*key*“ enthält im Folgenden jeweils einen Wertbezeichner und „*value*“ den dazugehörigen Wert. In der Spalte „*value_type*“, wird anscheinend numerisch der Datentyp für den Wert des jeweiligen Schlüssels bestimmt. Die Analyse aller Einträge in der Tabelle lässt die folgende Zuordnung der Nummern zu den Datentypen vermuten:

1: boolesche Werte
2: numerische Werte
3: Texte
4: Binärdaten

Durch das Hochladen einer Datei wurden weitere Einträge mit zusätzlichen key-value Paaren erzeugt. Abbildung 22 zeigt einen Auszug der Einträge für die offline verfügbare Datei *„upload.txt*.

	item_id	key	value	value_type
	fEWg4TEMeuU4jWHxcFNWOfib8tvZk	Filtern	Filtern	Filtern
1	1---fEWg4TEMeuU4jWHxcFNWOfib8tvZk	inactive-content-entries	^z 30B73hQ_k...	4
2	1---fEWg4TEMeuU4jWHxcFNWOfib8tvZk	version-counter	19	2
3	1---fEWg4TEMeuU4jWHxcFNWOfib8tvZk	stale-revision-id	0B73hQ_k7tK1...	3
4	1---fEWg4TEMeuU4jWHxcFNWOfib8tvZk	pinned	1	1
5	1---fEWg4TEMeuU4jWHxcFNWOfib8tvZk	offlineStatus	1	2
6	1---fEWg4TEMeuU4jWHxcFNWOfib8tvZk	local-title	upload.txt	3
7	1---fEWg4TEMeuU4jWHxcFNWOfib8tvZk	drivefs.Zone.Identifier	[ZoneTransfer] ZoneId=3	4
8	1---fEWg4TEMeuU4jWHxcFNWOfib8tvZk	content-entry	t 30B73hQ_k7t...	4

Abbildung 22: Einträge für die Datei "upload.txt"

Insgesamt konnten im Rahmen der in Abschnitt 3 ausgeführten Tests und den damit verbundenen verschiedenen Datenbankzuständen die in der nachfolgenden Tabelle dargestellten Keys identifiziert werden. Soweit der Zweck dieser Schlüssel nachvollziehbar war, wird dieser mit angegeben.

Schlüssel	Inhalt
inactive-content-entries	Binärdaten
version-counter	Zahlenwert; wird mit jeder Editierung der Datei erhöht
stale-revision-id	entspricht der „headRevisionId“ in der Google Drive API (ID der aktuellen Version der Datei in Google Drive)
pinned	bei allen Dateien immer „1“
offlineStatus	„1“, wenn Datei offline verfügbar gemacht wurde, ansonsten nicht vorhanden
local-title	angezeigter Name
drivefs.Zone.Identifier	bei allen Dateien: [ZoneTransfer] ZoneId=3
content-entry	Binärdaten
local-content-modified-date	Zeitpunkt der letzten Änderung an der Datei im „UNIX-Timestamp-Format“

Tabelle 15: Schlüssel in der Datenbanktabelle "item_properties"

Abbildung 23 zeigt einen Auszug aus der Abfrage der Informationen über die Datei „*upload.txt*“ mithilfe des Google Drive API Explorers, aus welcher sowohl die „ID“, wie auch der Name und die „headRevisionId“ der Datei ersichtlich sind.


```
-{
  "id": "1---fEWg4TEMeuU4jWHxcFNWOfib8tvZk",
  "name": "upload.txt",
  "originalFilename": "upload.txt",
  "headRevisionId": "0B73hQ_k7tK1UUVg5QWpCeElRZVdNMUZIekp1YzhBVDMyUDFjPQ"
 }
```


Abbildung 23: Google Drive API Abfrage für die Datei "upload.txt"

Tabelle „items“

Die Tabelle „items“ enthält für jede Datei einen Eintrag. Diese Einträge bleiben auch erhalten, wenn die entsprechende Datei gelöscht wurde. Abbildung 24 zeigt einen Auszug aus dem Eintrag für die Datei „*upload.txt“*.

	stable_id	id	proto	trashed	starred	is_owner	mime_type
	Filtern	1---	Filtern	Filtern	Filtern	Filtern	Filtern
1	117	1--- fEWg4TEMeuU4jWHxcFN...	*BLOB*	1	0	1	text/plain

Abbildung 24: Auszug aus der Datenbanktabelle "items"

Die nachfolgende Tabelle stellt die verschiedenen Felder dieser Datenbanktabelle und, soweit dies ermittelt werden konnte, auch den jeweiligen Inhalt dar.

Spaltenname	Inhalt
stable_id	lokale ID des Eintrages
id	Google Drive ID des Eintrages
proto	Binärdaten, in welchen unter anderem auch der angezeigte Name der Datei oder des Ordners enthalten ist
trashed	„1“, wenn nicht mehr in „Meiner Ablage“ vorhanden, ansonsten „0“
starred	„1“, wenn Datei über die Google Drive Weboberfläche markiert wurde, ansonsten „0“
is_owner	„1“, wenn durch den Nutzer hochgeladen wurde „0“, wenn die Datei für den Nutzer freigegeben wurde
mime_type	MIME-Typ der Datei (z.B. text/plain)
is_folder	„1“, wenn es sich um ein Ordner handelt, ansonsten „0“
title	identisch mit angezeigtem Titel
explicitly_trashed	„1“, wenn durch den Nutzer auf diesem Rechner gelöscht, ansonsten „0“
hidden	Bei allen Einträgen „0“
created_date	Erstellungsdatum als UNIX-Timestamp
modified_date	Änderungsdatum als UNIX-Timestamp
modified_by_me_date	bei allen Dateien „0“
shared_with_me_date	Datum, wann die Datei für den Nutzer freigegeben wurde als UNIX-Timestamp
viewed_by_me_date	Zeitpunkt, wann die Datei zuletzt durch den Nutzer betrachtet wurde als UNIX-Timestamp
quota_bytes	bei allen Dateien „0“
recency_date	bei allen Dateien „0“
drive_space	bei allen Dateien „0“
app_data_folder_space	bei allen Dateien „0“
photos_space	bei allen Dateien „0“
android_backup_space	bei allen Dateien „0“
is_tombstone	bei allen Dateien „0“
machine_root_folder	bei allen Dateien „0“
arbitrary_sync_folder	bei allen Dateien „0“
subscribed	bei allen Dateien „1“

local_title	bei allen untersuchten Dateien identisch mit „title“
team_drive_id	ID des Teamdrives, soweit sich die Datei auf einem befindet

Tabelle 16: Felder der Datenbanktabelle "items"

Wie bereits erwähnt, bleiben Einträge in dieser Tabelle auch bei vollzogener Löschung der entsprechenden Entität erhalten. Wird eine Datei mit identischem Namen neu hinzugefügt, wird dafür ein neuer Eintrag mit neuen IDs erzeugt.

Tabelle „properties“

Dieser Tabelle enthält die Spalten „*property*“ und „*value*“ (s. Abbildung 25).

	property	value
	Filtern	Filtern
1	root_id	0AL3hQ_k7tK1UUk9PVA
2	teamdrive_cache	1
3	cache_type	3
4	feature_switches	enabled true
5	largest_change_id	193
6	account	0 112425930574341475939 test nutzerB'test.nutz...

Abbildung 25: Inhalt der Tabelle "properties"

Beim Wert von „*root_id*“ handelt es sich um die Drive ID des Ordners „Meine Ablage“. Die Bedeutung der Werte der Properties „*teamdrive_cache*“, „*cache_type*“ und „*largest_change_id*“ war im Rahmen dieser Arbeit nicht möglich.

Das Feld „*feature_switches*“ enthält Binärdaten. Aufgrund des Namens kann vermutet werden, dass darin verschiedene Einstellungen gespeichert sind. Bei Betrachtung der Binärdaten, könnten diese teilweise ausgelesen werden. Darauf wurde jedoch im Rahmen dieser Arbeit verzichtet. Abbildung 26 zeigt einen Auszug aus den entsprechenden Felddaten.

```
0000  0a 11 0a 09 12 07 65 6e 61 62 6c 65 64 12 04 74  ......enabled..t
0010  72 75 65 0a 19 0a 11 12 0f 76 69 72 74 75 61 6c  rue......virtual
0020  5f 66 6f 6c 64 65 72 73 12 04 74 72 75 65 0a 38  _folders..true.8
0030  0a 2a 12 28 64 6f 6b 61 6e 5f 6b 65 65 70 5f 61  .*.(dokan_keep_a
0040  6c 69 76 65 5f 74 69 6d 65 6f 75 74 5f 61 66 74  live_timeout_aft
0050  65 72 5f 77 61 6b 65 75 70 5f 6d 73 12 0a 31 34  er_wakeup_ms..14
0060  30 30 30 30 30 30 30 30 0a 2b 0a 1d 12 1b 64 6f  00000000.+....do
0070  6b 61 6e 5f 6b 65 65 70 5f 61 6c 69 76 65 5f 74  kan_keep_alive_t
0080  69 6d 65 6f 75 74 5f 6d 73 12 0a 31 34 30 30 30  imeout_ms..14000
0090  30 30 30 30 30 0a 11 0a 08 12 06 72 65 63 65 6e  00000......recen
00a0  74 12 05 66 61 6c 73 65 0a 12 0a 09 12 07 73 74  t..false......st
00b0  61 72 72 65 64 12 05 66 61 6c 73 65 0a 15 0a 0c  arred..false....
```

Abbildung 26: Binärdaten des Property "feature_switches"

Das Property „*account*“ enthält ebenfalls Binärdaten. Auch hier sind durch Inaugenscheinnahme weitere Rückschlüsse möglich. So können unter anderem der Nutzername des Accounts („blau“) und dessen E-Mail-Adresse („rot“) identifiziert werden. Auch die Drive-ID des Ordners „*Meine Ablage*“ ist ersichtlich („grün“). Abbildung 27 zeigt das.

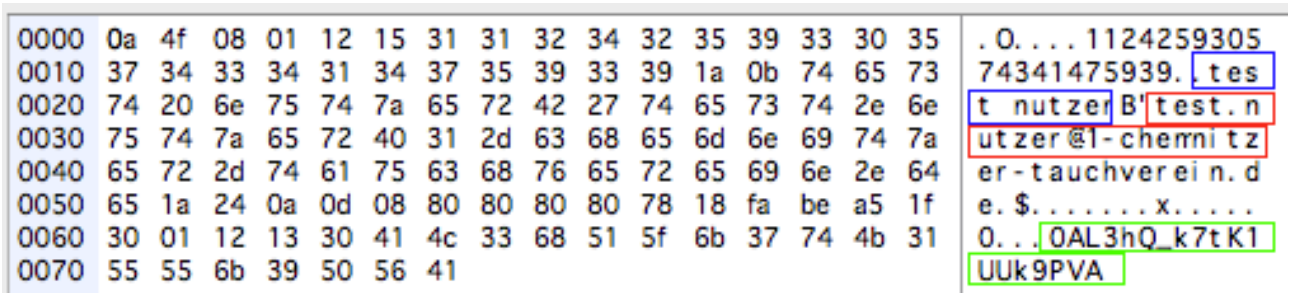


Abbildung 27: Binärdaten des Property "account"

Auch an dieser Stelle wurde aufgrund des begrenzten Zeitumfanges auf eine weitere Analyse der Daten verzichtet.

Tabelle „queries“

Die Tabelle „*queries*“ beinhaltet insgesamt 4 Spalten mit den Namen „*hash*“, „*proto*“, „*complete*“ und „*size*“. Bei den Feldern „*hash*“ und „*size*“ handelt es sich um Integer Felder, während „*complete*“ vom Typ boolean ist und „*proto*“ Binärdaten beinhaltet.

Die Tabelle beinhaltete über den gesamten Untersuchungszeitraum hinweg konstant 3 Zeilen, die in Abbildung 28 dargestellt werden.

	hash	proto	complete	size
	Filtern	Filtern	Filtern	Filtern
1	-7107626291117447615	*BLOB*	1	0
2	-4442904673763526029	root	1	50
3	8180096014014521420	0AL3hQ_k7tK1UUk9PVA user.drive.pinned	1	0

Abbildung 28: Inhalt der Tabelle "queries"

Der genaue Zweck der Tabelle konnte im Rahmen dieser Untersuchung nicht ergründet werden.

Tabelle „stable_ids“:

Bei der Tabelle „*stable_ids*“ handelt es sich um eine reine Zuordnungstabelle. Über diese ist eine Zuordnung der lokalen ID einer Datei (Spalte: „*stable_id*“) zur Drive ID der Datei möglich (Spalte: „*cloud_id*“). Die nachfolgende Abbildung zeigt die Tabelle für den Ordner „*Meine Ablage*“ und die Datei „*upload.txt*“.

	stable_id	cloud_id
1	101	0AL3hQ_k7tK1UUk9PVA
2	117	1---fEWg4TEMeuU4jWHxcFNWOfib8tvZk

Abbildung 29: Auszug aus der Tabelle "stable_ids"

Tabelle „stable_parents“

Über die Tabelle „*stable_parents*“ erfolgt die Abbildung der hierarchischen Ordnerstruktur. Zu jedem Ordner und zu jeder Datei, existiert ein Eintrag, welcher definiert in welchem Vater-Ordner sich dieser befindet. Zur den Root-Ordnern („*Teamablagen*“ und „*Meine Ablage*“) existieren keine Einträge.

Die Tabelle selbst besitzt 3 Spalten:

- „*item_stable_id“*
- *„parent_stable_id“*
- *„local_title_hash“*

Über die Spalten *„item_stable_id“* und *„parent_stable_id“* erfolgt die Zuordnung. Der Einfluss der Spalte *„local_title_hash“* sowie die Mechanismen zur Bildung ihres Inhaltes konnten im Rahmen dieser Arbeit nicht erkannt werden. Abbildung 29 zeigt die Zuordnung exemplarisch für die Datei *„upload.txt“* zum Ordner „*Meine Ablage“*.

	item_stable_id	parent_stable_id	local_title_hash
	117	Filtern	Filtern
1	117	101	1446883536

Abbildung 30: Auszug aus der Tabelle "stable_parents"

Tabelle „teamdrives “

Das im Rahmen dieses Tests zusätzlich angelegte Teamdrive mit dem Namen „test“ konnte in dieser Tabelle gefunden werden. In der Tabelle werden für jedes Teamdrive neben der lokalen und der Google Drive ID auch verschiedene Einstellungen gespeichert. Die nachfolgende Abbildung zeigt den Eintrag.

	stable_id	id	local_title	proto	active	largest_change_id	cache_type	last_used
	Filtern	Filtern	Filtern	Filtern	Filtern	Filtern	Filtern	Filtern
1	103	0AELLSubhvZ-yUk9PVA	test	*BLOB*	1	7	2	0

Abbildung 31: Tabelle "teamdrives"

Die Bedeutung der einzelnen Felder, wird in der nachfolgenden Tabelle erläutert.

Spaltenname	Inhalt
stable_id	lokale ID des Teamdrives
id	Google Drive ID des Teamdrives
local_title	identisch mit dem angezeigten Namen des Teamdrives
proto	Binärdaten, in welchen unter anderem auch der angezeigte Name enthalten ist
active	im Test „1“; in den Einstellungen konnte keine Möglichkeit gefunden werden, das Teamdrive zu deaktivieren
largest_change_id	im Testfall „7“; Bedeutung konnte nicht ermittelt werden
cache_type	im Testall „2“; Bedeutung konnte nicht ermittelt werden
last_used	„0“, obwohl Änderungen an den Dateien in diesem Teamdrive vorgenommen wurden.

Tabelle 17: Felder der Datenbanktabelle "teamdrives"

4.2 Betrachtung des Ordners „content_cache“

Bei mehreren der im Abschnitt 3 beschriebenen Aktionen wurden Änderungen an der Datenbank „*chunks.db“*, welche sich in diesem Ordner befindet, sowie am Inhalt verschiedener Unterordner festgestellt. In der folgenden Betrachtung soll herausgefunden werden, inwieweit bei einer forensischen Totanalyse des Systems ohne Internetzugang mithilfe des Cache eventuell der Inhalt verschiedener Dateien aus dem Google Drive wiederhergestellt werden kann.

Zu diesem Zweck wurde das Cache Verzeichnis nach den Aktionen „Upload einer Datei“, „Öffnen einer Datei“ und „Offline verfügbar machen einer Datei“ untersucht. Zusätzlich wurde auch noch eine PDF-Datei mit einer Größe von 5 MB sowie eine weitere Textdatei mit dem Namen „*upload.txt.txt“*, welche eine größere Menge an Text enthielt (Gesamtgröße 1MB) hochgeladen und danach eine Kopie des Ordners „*C:\Users\Markus\AppData\Local\Google\DriveFS\ dGVzdC5udXR6ZXJAMS1jaGVtbml0emVyLXRhdWNodmVyZWluLmRI“* angefertigt.

Dann wurden diese Dateien offline verfügbar gemacht und eine weitere Kopie des Ordners erstellt. Die in diesen Kopien enthaltenen Datenbanken wurden zur weiteren Analyse herangezogen.

4.2.1 Allgemeine Feststellungen

Wie bereits bei der Installation der Anwendung festgestellt, beinhaltet das Verzeichnis „*content_cache“* insgesamt 67 Unterverzeichnisse („*d0*“ bis „*d66*“). Zusätzlich befindet sich in darin

die SQLite-Datenbank *„chunks.db“*. Sobald in eines der Verzeichnisse eine Cache-Datei geschrieben wird, wird der Struktur ein weiterer Ordner hinzugefügt.

So wurde zum Beispiel beim Hinzufügen einer Datei in den Ordner *„d36“* der neue Ordner *„d103“* erstellt, in welchem die Cache-Datei „103“ abgelegt war.

Während die Textdatei, welche in den in Abschnitt 3 beschriebenen Szenarien verwendet wurde, unabhängig davon ob sie offline verfügbar war oder nicht, jeweils vollständig in einem der Cache-Ordner zu finden war, war das bei dem PDF-Dokument nicht der Fall. Diese landete nur dann vollständig im Cache-Ordner, wenn sie als offline verfügbar markiert war. Ist dies nicht der Fall, dann können dort nur Teile der Datei angetroffen werden. Eine Verteilung einer in der Cloud befindlichen Datei auf mehr als eine Cache-Datei konnte nicht beobachtet werden.

4.2.2 Aufbau der SQLite-Datenbank „*chunks.db*“

Diese Datenbank enthält lediglich die Tabelle *„chunk_table“*, welche aus den beiden Spalten *„item_path“* und *„chunks“* besteht. Während *„chunks“* vom Datentyp „BLOB“ ist enthält *„item_path“* Integer-Werte. Abbildung 32 zeigt einen Ausschnitt aus der Tabelle.

	item_path	chunks
	Filtern	Filtern
1	103	BLOB
2	107	BLOB
3	109	BLOB
4	113	BLOB

Abbildung 32: Auszug aus der Tabelle "chunk_table"

Im folgenden Abschnitt soll dargestellt werden, wie über ein Element aus der Datenbank *„metadata_sqlite_db“* eine Zuordnung zwischen den Cache-Dateien und den Informationen erfolgen kann.

4.2.3 Zuordnung der Cache-Dateien

Für die Zuordnung der Cache-Dateien im Ordnersystem des Elternordners *„content_cache“* kann das Feld *„item_path*“ herangezogen werden. Der Name der Cache-Datei ist dabei identisch mit der Nummer in der Spalte *„item_path“*. Das Unterverzeichnis in dem sich die Datei befindet, trägt dieselbe Bezeichnung mit einem vorangestellten „d“. Dieses Verzeichnis ist wiederum ein Unterordner eines der 67 o.g. Verzeichnisse innerhalb von *„content_cache*“. In welchem genau lässt sich mit der Formel *<VERZEICHNISNAME> = „d“ + <ITEM PATH> - 67* berechnen. Am Beispiel der Cache-Datei „103“ würde das bedeuten, dass sie im Verzeichnis *„content_cache“* unter dem Pfad *„d36\d103\103“* gefunden werden kann.

Die Zuordnung der Cache-Datei zu einer Entität in der Tabelle *„items“* aus der Datenbank *„metadata_sqlite_db“* gestaltet sich schwieriger. Im Rahmen dieser Ausarbeitung konnte ein Verfahren gefunden werden, mit dem eine sichere Zuordnung von offline verfügbaren und

vollständig im Cache vorhandenen Dateien möglich ist. Dieses Verfahren soll am Beispiel der Cache-Datei „*103*“ erläutert werden.

Dazu muss zuerst der Wert aus der Blob-Spalte *„chunks“* ausgelesen werden. Teile dieses Wertes dienen der Zuordnung zur jeweiligen gespeicherten Datei. Im Beispiel lautet der Wert aus „chunks“ „0a 06 08 00 10 9e bb 03“. Die Bedeutung der einzelnen Bestandteile wird in der nachfolgenden Tabelle erläutert.

Bytefolge	Bedeutung
0a	Bei allen Cache-Dateien identisch
06	Anzahl der nachfolgenden Bytes
08 00 10	Bei allen Cache-Dateien identisch
9e bb 03	Bytefolge zur Identifizierung der entsprechenden Datei

Tabelle 18: Aufschlüsselung des Inhaltes der Spalte "chunks"

Die extrahierte Bytefolge kann in der Tabelle *„item_properties“* der Datenbank *„metadata_sqlite_db“* wiedergefunden werden. Wie bereits in Abschnitt 4.1.3 beschrieben, existiert zu jeder Datei in dieser Tabelle ein Eintrag mit dem Key *„content-entry*. Dieser enthält am Ende des Binärdatenstromes zuerst die jeweilige Cloud-ID der Datei (rot), ein Trennbyte (blau) und danach die zur Identifizierung der Cache-Datei benötigte Bytefolge (grün). Abbildung 33 stellt dies noch einmal grafisch dar.

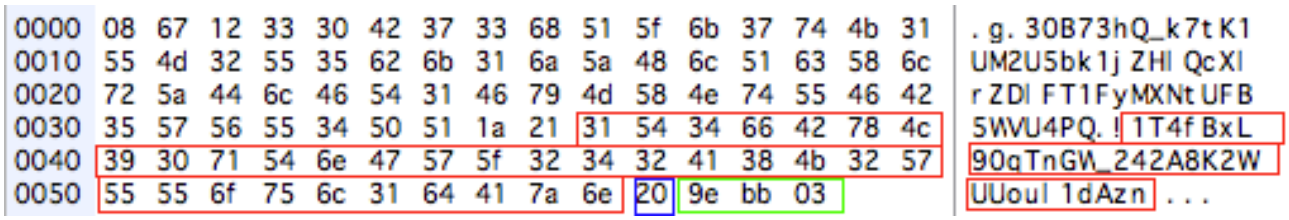


Abbildung 33: Identifizierung der Cache-Dateien

Über die eindeutige Cloud-ID der Datei kann nun aus der Tabelle *„items“* über die dortige Spalte „id“ die Datei zugeordnet werden. Dieses Verfahren funktioniert wie bereits erwähnt stabil bei der Zuordnung von offline verfügbaren und vollständig im Cache befindlichen Dateien. Eine Möglichkeit für die Zuordnung von Dateien, welche sich nicht vollständig im Cache befinden, zeichnete sich im begrenzten Rahmen der Arbeit nicht ab.

4.3 Betrachtung der geschriebenen Logdateien

Im Ordner „*C:\Users\Markus\AppData\Local\Google\DriveFS\Logs*” befinden sich insgesamt drei verschiedene Log-Dateien. Hierbei handelt es sich um die Dateien *„chrome_debug.log“*, *„drive_fs.txt“* und *„parent.txt“*. Während der Analyse konnte festgestellt werden, dass nahezu jede Aktion Änderungen in der Datei *„drive_fs.txt“* bewirkt, während *„chrome_debug.log“* lediglich bei Login-Ereignissen betroffen ist.

Bei der Betrachtung des Verzeichnisses „*Logs“* konnte weiterhin festgestellt werden, dass offensichtlich sowohl für „*drive_fs.txt“* wie auch für *„parent.txt“* das Rotating-Log-System benutzt wird. Das bedeutet, dass ab einer bestimmten Dateigröße oder Existenzdauer bzw. bei einem anderen Event (z.B. dem Neustart der Anwendung) die Logdatei umbenannt und eine neue Datei begonnen wird. In diesem Fall werden an den Dateinamen bei der Umbenennung einfach ein Unterstrich und eine laufende Nummer angehängt (z.B. „*drive_fs_1.txt“*).

4.3.1 Logdatei „drive_fs.txt“

Aus dieser Datei konnten sowohl allgemeine Informationen über verschiedene Konfigurationseinstellungen der Anwendung als auch solche über die gelesenen Registry-Werte und andere interne Aktionen abgerufen werden. Informationen über hochgeladene bzw. geöffnete oder editierte Dateien waren aus dem Log nicht ersichtlich. Abbildung 34 zeigt einen Ausschnitt aus diesem Log:

```
2018-01-01T14:47:58Z I [908] registry_win.cc:58:cello::fs::GetDwORDFromRegistryValue Could not read reg value [Software\Google\DriveFS, DisableSSLValidation]: 0x2.
2018-01-01T14:47:58Z I [908] registry_win.cc:86:cello::fs::GetStringFromRegistryValue Could not read reg value [Software\Google\DriveFS, DefaultMountPoint]: 0x2.
2018-01-01T14:47:58Z I [908] registry_win.cc:86:cello::fs::GetStringFromRegistryValue Could not read reg value [Software\Google\DriveFS, DefaultMountPoint]: 0x2.
2018-01-01T14:47:58Z I [908] registry_win.cc:58:cello::fs::GetDwORDFromRegistryValue Could not read reg value [Software\Google\DriveFS, ForceBrowserAuth]: 0x2.
2018-01-01T14:47:58Z I [908] options.cc:71:cello::fs::SetFallbacksAndValidateOptions Using ssl root certificates from: C:\Program Files\Google\Drive File Stream\25.102.133.409\config
2018-01-01T14:47:58Z I [908] drive_fs_main.cc:602:wWinMain Nonstandard filter drivers present: (none)
2018-01-01T14:47:58Z I [908] i18n.cc:278:cello::i18n::Translators::TrimLanguage Dash found in language name. Trimmed to 'de'.
2018-01-01T14:47:58Z I [908] i18n.cc:262:cello::i18n::Translators::SetCatalogForLanguage Translator found. Using 'de' language.
2018-01-01T14:47:58Z I [908] drive_fs.cc:253:cello::fs::RunDriveFS Options: argv_0: "C:\\Program Files\\Google\\Drive File Stream\\25.102.133.409\\GoogleDriveFS.exe"
base_path: "C:\\Users\\Markus\\AppData\\Local\\Google\\DriveFS"
ipc_pipe_path: "\\\\.\\Pipe\\GoogleDriveFSPipe_Markus"
shell_ipc_pipe_path: "\\\\.\\Pipe\\GoogleDriveFSPipe_Markus_shell"
crash_handler_token: "\\\\.\\pipe\\crashpad_2656_GJZBESXNCAPDDQCA"
feature_config {
  enabled: true
  virtual_folders: true
  thumbnails: true
  deprecated: false
  starred: false
  recent: false
  drive_dot: false
  metadata_cache_reset_counter: 0
  spotlight: true
  feedback: true
  share_dialog: false
  shell_ipc: false
  changelog_download_throttle_time_ms: 16000
  temporary_items_virtual_folder: false
  case_insensitive: false
  local_disk_aware_get_free_space: false
  context_menu_copy_link: false
  dokan_keep_alive_timeout_ms: 45000
  dokan_keep_alive_timeout_after_wakeup_ms: 600000
  pause_syncing_option: false
  short_circuit_crawlers_with_eof: false
}
enable_tracing: false
binary_traces: false
default_mount_point: "G"
enable_ui: true
```

Abbildung 34: Auszug aus dem Log "drive_fs.txt"

Fehlermeldungen, die offensichtlich bei dem Zugriff auf das Google Drive API durch das Programm auftraten, werden ebenfalls aufgezeichnet. Abbildung 35 zeigt einen entsprechenden Ausschnitt.

```
2018-01-01T15:01:33Z E [1696] celloquerypage.cc:214:cello::CelloQueryPage::OnDataStoreQueryComplete LOCAL query id: "trash:10"
 returned 0 results (0 unfenced) with status UNAVAILABLE_RESOURCE:
2018-01-01T15:01:34Z E [1696] drive_util.cc:665:cello::CloudStoreLog HTTP 404 (Http(404) Not Found) when getting https://www.googleapis.com/drive/v2internal/files/trash%3A10?reportPer

2018-01-01T15:01:34Z E [1696] drive_util.cc:670:cello::CloudStoreLog ----------  [begin response body]  ----------
{
 "error": {
  "errors": [
   {
    "domain": "global",
    "reason": "notFound",
    "message": "File not found: trash:10",
    "locationType": "other",
    "location": "file"
   }
  ],
  "code": 404,
  "message": "File not found: trash:10"
 }
}

----------  [end response body]  ----------
```

Abbildung 35: Fehlermeldung in der Datei "drive_fs.txt"

4.3.2 Logdatei „parent.txt“

Auch aus dieser Datei waren verschiedene Konfigurationseinstellungen und gelesene Registry-Schlüssel ersichtlich. Hinweise auf bearbeitete Dateien waren auch hier nicht ersichtlich (s. Abbildung 36).

```
2018-01-01T14:47:58Z I [3636] instrumentation.cc:96:cello::fs::`anonymous-namespace'::OpenLogFileInDirectory Logging to C:\Users\Markus\AppData\Local\Google\DriveFS\Logs\parent.txt
2018-01-01T14:47:58Z I [3396] drive_fs_main.cc:452:cello::fs::`anonymous-namespace'::EarlySetup Feature config (not including overrides): enabled: true
virtual_folders: true
thumbnails: true
deprecated: false
starred: false
recent: false
drive_dot: false
metadata_cache_reset_counter: 0
spotlight: true
feedback: true
share_dialog: false
shell_ipc: false
changelog_download_throttle_time_ms: 16000
temporary_items_virtual_folder: false
case_insensitive: false
local_disk_aware_get_free_space: false
context_menu_copy_link: false
dokan_keep_alive_timeout_ms: 45000
dokan_keep_alive_timeout_after_wakeup_ms: 600000
pause_syncing_option: false
short_circuit_crawlers_with_eof: false

2018-01-01T14:47:58Z I [3396] registry_win.cc:86:cello::fs::GetStringFromRegistryValue Could not read reg value [Software\Google\DriveFS, TrustedRootCertsFile]: 0x2.
2018-01-01T14:47:58Z I [3396] registry_win.cc:58:cello::fs::GetDWORDFromRegistryValue Could not read reg value [Software\Google\DriveFS, DisableSSLValidation]: 0x2.
2018-01-01T14:47:58Z I [3396] registry_win.cc:86:cello::fs::GetStringFromRegistryValue Could not read reg value [Software\Google\DriveFS, DefaultMountPoint]: 0x2.
2018-01-01T14:47:58Z I [3396] registry_win.cc:86:cello::fs::GetStringFromRegistryValue Could not read reg value [Software\Google\DriveFS, DefaultMountPoint]: 0x2.
2018-01-01T14:47:58Z I [3396] registry_win.cc:58:cello::fs::GetDWORDFromRegistryValue Could not read reg value [Software\Google\DriveFS, ForceBrowserAuth]: 0x2.
2018-01-01T14:47:58Z I [3396] options.cc:71:cello::fs::SetFallbacksAndValidateOptions Using ssl root certificates from: C:\Program Files\Google\Drive File Stream\25.102.133.409\config
2018-01-01T14:47:58Z I [3396] drive_fs_main.cc:602:wWinMain Nonstandard filter drivers present: (none)
```

Abbildung 36: Auszug aus dem Log "parent.txt"

4.3.3 Logdatei „chrome_debug.log“

Die Datei „*chrome_debug.log*“ enthielt lediglich Fehlermeldungen oder Warnungen, die bei der Ausführung der Anwendung auftraten.

Informationen über die im Rahmen der Tests veränderten Dateien konnten auch in diesem Log nicht gefunden werden. Abbildung 37 zeigt einen Ausschnitt aus der Logdatei.

```
[1230/160443.746:ERROR:gpu_child_thread.cc(174)] Exiting GPU process due to errors during initialization
[1230/160443.824:ERROR:browser_gpu_channel_host_factory.cc(103)] Failed to launch GPU process.
[1230/160444.012:ERROR:gpu_child_thread.cc(174)] Exiting GPU process due to errors during initialization
[1230/160444.105:ERROR:gpu_child_thread.cc(174)] Exiting GPU process due to errors during initialization
[1230/160444.387:ERROR:gpu_child_thread.cc(174)] Exiting GPU process due to errors during initialization
[1230/160444.574:ERROR:gpu_child_thread.cc(174)] Exiting GPU process due to errors during initialization
[1230/160444.621:ERROR:browser_gpu_channel_host_factory.cc(103)] Failed to launch GPU process.
[1230/160444.621:ERROR:browser_gpu_channel_host_factory.cc(103)] Failed to launch GPU process.
[1230/160444.621:ERROR:browser_gpu_channel_host_factory.cc(103)] Failed to launch GPU process.
[1230/160444.621:ERROR:browser_gpu_channel_host_factory.cc(103)] Failed to launch GPU process.
[1230/160444.621:ERROR:browser_gpu_channel_host_factory.cc(103)] Failed to launch GPU process.
[1230/173218.105:ERROR:browser_gpu_channel_host_factory.cc(103)] Failed to launch GPU process.
[0101/140004.420:WARNING:spdy_session.cc(2910)] Received HEADERS for invalid stream 5
[0101/140004.429:WARNING:spdy_session.cc(2910)] Received HEADERS for invalid stream 7
```

Abbildung 37: Auszug aus der Logdatei "chrome_debug.log"

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Nach einer allgemeinen Beschreibung der Anwendung „Drive File Stream“ erfolgte im ersten Hauptteil der Arbeit die Definition verschiedener Szenarien anhand derer die Anwendung analysiert werden sollte. Auch wurden in diesem Zusammenhang die angewandten Untersuchungsmethoden sowie die für die Untersuchung verwendete Software dargestellt.

Im nächsten Teil wurden die durch die zuvor definierten Aktionen ausgelösten Veränderungen beschrieben und dabei nach Veränderungen der Registry und im Dateisystem unterschieden. Nachfolgend wurden einzelne, durch das Programm erzeugte Dateien näher betrachtet. Dabei handelte es sich sowohl um die durch die Anwendung geschrieben SQLite-Datenbanken, wie auch um die durch sie erstellten Logdateien.

Zusätzlich wurde der Cache-Mechanismus der Anwendung untersucht und eine Möglichkeit aufgezeigt, wie sowohl als offline verfügbar gemachte, als auch vollständig im Cache abgelegte Dateien ohne Zugriff auf die Google Cloud wiederhergestellt werden können.

Im Rahmen der Untersuchung konnte festgestellt werden, dass bei einer forensischen Analyse eines Rechners, auf dem die Software „Drive File Stream“ aktiv war, über die Datenbank „*metadata_sqlite_db*“ in dessen „*AppData*“ Verzeichnis Informationen über die im Google Drive gespeicherten Daten gewonnen werden können. Dabei spielt es keine Rolle, ob die Daten zum Untersuchungszeitpunkt bereits gelöscht waren. Mithilfe des lokalen Cache ist es mögliche, auch ohne eine Online-Verbindung offline gespeicherte und vollständig gecachte Dateien wiederherzustellen.

Es bleibt anzumerken, dass bei einer Deinstallation der Anwendung zwar die Programmdateien, nicht aber die im Ordner „*AppData“* gespeicherten Konfigurations- und Datendateien (wie z.B. die durch das Programm verwendete SQLite-Datenbanken) gelöscht werden. So ist es auch nach einer eventuellen Deinstallation der Anwendung noch möglich, bei einer Analyse des Rechners Informationen über die in der Cloud gespeicherten Dateien zu gewinnen.

5.2 Ausblick

Aufgrund der begrenzten Zeit für die Erstellung des Berichts konnte kein komplettes Reverse Engineering der untersuchten Anwendung erfolgen. Weiterhin konnten nicht alle durch die Anwendung erzeugten Binär-Dateien (und Binärdatenfelder der Datenbanken) entschlüsselt werden. Im Rahmen einer Fortführung dieser Arbeit könnte dies ein erster Ansatzpunkt sein, um die Erkenntnisse über diese Anwendung zu vervollständigen.

Ein weiterer Punkt, welcher bei einer Fortführung der Arbeit zu betrachten wäre, ist die detaillierte Analyse der durch die Anwendung erzeugten Cache-Dateien. Insbesondere wäre hier zu klären, nach welchen Gesichtspunkten eine Datei nur unvollständig gecacht wird, wie diese Cache-Datei der eigentlichen Datei zugeordnet wird und welche Informationen aus der gecachten Datei zusätzlich gewonnen werden könnten.

Auch eine Betrachtung des Ordners „*thumbnails_cache*“, welcher im selben Verzeichnis wie der Ordner *„content_cache“* liegt, erfolgte nicht, da im Rahmen der Analysen keine Schreib- und Lesezugriffe auf dieses Verzeichnis oder die darin befindliche SQLite-Datenbank „*chunks.db“* registriert wurden. Ein kurzer Test ergab jedoch sowohl im Verzeichnis als auch der Datenbank Aktivitäten, sobald Bilddateien im Cloudlaufwerk abgelegt wurden. Eine Untersuchung dieses Ordners erscheint daher bei einer Fortführung der Arbeit ebenfalls sinnvoll.

Literaturverzeichnis

[1] Google, „Choose a sync solution,“ Google, 2018. [Online]. URL: https://support.google.com/a/answer/7491633?hl=en. [Zugriff am 11 Januar 2018].

[2] Felix Freiling, Christian Riess: Browser- und Anwendungsforensik. Kursunterlagen. 6. Auflage vom 14.11.2017

[3] Digital Forensics XML project, „Digital Forensics XML project and library,“ Digital Forensics XML project and library, 2017. [Online]. URL: https://github.com/simsong/dfxml. [Zugriff am 11 Januar 2018].

[4] Microsoft, „Windows Sysinternals,“ 2018. [Online]. URL: http://sysinternals.com. [Zugriff am 11 Januar 2018].

[5] maddes, regshot und xhmikosr, „regshot,“ 2018. [Online]. URL: https://sourceforge.net/projects/regshot/. [Zugriff am 11 Januar 2018].

[6] Google, „Drive File Stream bereitstellen,“ Google, 2018. [Online]. URL: https://support.google.com/a/answer/7491144?hl=de. [Zugriff am 11 Januar 2018].

[7] T. Hudek, 20 April 2017. [Online]. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/install/components-of-a-driver-package. [Zugriff am 11 Januar 2018].

Anhang

Anhang 1: ER-Diagramm Datenbank „metadata_sqlite_db“