Forensische Analyse des Dateisystems Dahua DHFS 4.1

Dane Wullen
Email: dane.wullen@gmail.com

***Zusammenfassung*—Dateisysteme sind der Grundstein zur Verwaltung von Daten auf digitalen Medien. Sie dienen zur Strukturierung und effizienten Speicherung von Dateien wie Bilder, Dokumente und Videos. Viele dieser Dateisysteme, z. B. FAT, NTFS, HFS+ etc. bekannter Betriebssysteme und Hersteller wurden in zahlreichen Arbeiten analysiert und können von den meisten forensischen Programmen interpretiert werden. Neben diesen bekannten Dateisystemen existieren unbekannte, meist proprietäre Dateisysteme, welche immer stärker in den Fokus der digitalen Forensik rücken.**

Unter anderem implementieren viele Hersteller von Netzwerkvideorekordern diese unbekannten Dateisysteme um Videodaten effizient zu speichern. Für die digitale Forensik können diese Videodaten ein wichtiger Schlüssel zur Aufklärung und Ermittlung von Straftaten sein.

Diese Arbeit befasst sich mit dem Dateisystem DHFS 4.1 des Herstellers DAHUA. Auf Basis der Dateisystemanalyse wird die Struktur und Dateiverwaltung erläutert und wie Methoden der digitalen Forensik angewandt werden können um Videodateien zu finden. Auf Basis dieser Erkenntnisse wird eine X-Tension für das Programm X-Ways Forensics entwickelt um digitale Datenträger, welche dieses Dateisystem implementieren, interpretieren und analysieren zu können.

***Schlagwörter*—DHFS 4.1, DAHUA, DVR, NVR, Dateisystem**

I. Einleitung

Die digitale Forensik ist eine noch recht junge Disziplin in der Aufklärung von Straften. Beginnend in den 1980er Jahren erfuhr die digitale Forensik ab dem Jahre 2000 einen Sprung, da immer mehr Menschen Zugang zu Computern, Mobiltelefonen und dem Internet erhielten. [1]

Neben den klassischen, analogen Spuren und Beweismittel zur Aufklärung von Straftaten gewinnen die digitalen Beweismittel wie Mobiltelefone, Computer und Smartphones immer mehr an Bedeutung. Diese Beweismittel enthalten elektronische Spuren wie Bilder, Dokumente, E-Mails und Chatverläufe, welche zur Aufklärung von Straftaten relevant sein können.

Zur Abwendung und Aufklärung von Straftaten spielen digitale Videoüberwachungsanlagen (digital video recorder, DVR) eine besondere Rolle. Diese Anlagen zeichnen, je nach Kapazität, rund um die Uhr die von den Kameras erfassten Bereiche auf und können in der Strafverfolgung einen hohen Stellenwert haben.

Viele dieser Systeme verwenden jedoch nicht die in der digitalen Forensik bekannten und analysierten Dateisysteme wie FAT, exFAT oder NTFS. Stattdessen implementieren die Hersteller dieser Videoüberwachungsanlagen eigene, proprietäre Dateisysteme zur Verwaltung der Videodaten.

Damit die gängigen forensischen Programme die Daten auf diesen Dateisystem lesen und interpretieren können müssen die Dateisysteme analysiert und der Aufbau sowie die Verwaltung der Daten erforscht werden, da nur so eine Extraktion der Daten erfolgen kann.

Ziel dieser Arbeit ist die Analyse des proprietären Dateisystems DHFS 4.1 des Herstellers DAHUA. Auf der Basis von verschiedenen digitalen Datenträgern wird das Dateisystem Abschnitt für Abschnitt analysiert und die zugrundeliegenden Videodaten extrahiert.

Die Ergebnisse dieser Analyse fließen in die Entwicklung einer Erweiterung des Programms X-Ways Forensics ein, einer so genannten X-Tension. Mit dieser X-Tension können anschließend digitale Datenträger oder die Abbilder mit dem Programm eingelesen, interpretiert und analysiert werden, sodass die vorhandenen und teilweise gelöschten Videodateien für eine Ermittlungsperson sicht- und auswertbar werden.

II. Verwandte Arbeiten

Die Grundlage für die Dateisystemanalyse ist das Werk von Brian Carrier "File System Forensic Analysis". Carrier beschreibt hier die genauen Analyseschritte welche notwendig sind ein Dateisystem zu erforschen und seine Bestandteile in bestimmte Kategorien einzuordnen. [2]

Im Bereich der Videoüberwachungsanlagen gibt es einige verwandte Arbeiten, welche sich mit der Analyse von proprietären Dateisystemen oder den Videodateien dieser Anlagen beschäftigen. Das am naheliegendste Werk ist von Jaehyeok Han et al., in welchem das Dateisystem des Herstellers von Videoüberwachungsanlagen HIKVISION analysiert wird. Han et al. beschreiben dabei den Aufbau des Dateisystems und der Videodateien, welche gespeichert werden. [3]

Ein weiteres verwandtes Werk von Lee Tobin et al. beschreibt die allgemeine Analyse von unbekannten Dateisystemen von Videoüberwachungsanlagen. Diese Arbeit

untersucht verschiedene Methoden der manuellen Analyse von Dateisystemen, u. A. das Auffinden von Offsets des Dateisystems und wiederholenden Strukturen sowie die Verknüpfung dieser Daten und Informationen. [4] Von Dongen beschreibt in seiner Arbeit die Analyse eines Dateisystems und der proprietären Dateiformate eines Videorekorders des Herstellers Samsung. [5]

Werke speziell über den Hersteller DAHUA existieren nur in einer geringen Zahl. Evangelos Dragonas et al. beschreiben in Ihrer Arbeit die Analyse von Logdateien eines DVR des Herstellers DAHUA. Diese Logdateien enthalten ebenfalls für die digitale Forensik wesentliche Informationen über die Konfiguration und das Nutzerverhalten und können einer Ermittlungsperson einen Einblick über die Anlage verschaffen. [6] Die Logdateien sind jedoch nicht die Logdateien, die in diesem Dateisystem analysiert werden.

III. Stand der Technik

Neben den derzeitigen Arbeiten und Werken welche sich mit der Dateisystemanalyse, vor allem bei den Herstellern von Videoüberwachungsanlagen beschäftigen, wird ein Blick auf den aktuellen Stand der Technik geworfen, speziell auf Programme welche in der Lage sind diese Dateisysteme zu analysieren.

Eines der bekanntesten Programme ist Witness, ehemals DVR-Examiner, von Magnet Forensics. Dieses Programm ist laut dem Hersteller in der Lage eine Vielzahl an Dateisystemen von Videoüberwachungsanlagen zu interpretieren und analysieren zu können. Erfahrungen von Benutzern berichten, dass Videodateien des proprietären Dateisystems von DAHUA ebenfalls mit Witness eingesehen werden können. Es werden vorhandene als auch gelöschte Videodateien gesucht und wiederhergestellt. [7]

DiskInternals bietet ebenfalls ein Programm zur Einsicht und Speicherung von Videos verschiedener Dateisystem von DVR-Herstellern, u. A. HIKVISION und DAHUA. [8] Eine Testversion ist frei verfügbar, der Kauf einer Lizenz wird für die weitere Nutzung vorausgesetzt.

Neben den kostenpflichtigen Programmen stehen auch einige Open-Source Programme zur Verfügung, welche Daten des proprietären Dateisystems DHFS 4.1 erkennen und extrahieren können.

Galileu Bastista entwickelte ein Python-Programm, welche auf der Basis eines Festplattenabbildes einer DAHUA Videoüberwachungsanlage die gespeicherten Videos anzeigen und extrahieren kann. Das Programm ist auch in der Lage nach gelöschten Dateien anhand von Signaturen zu suchen. Im Programmcode ist der Aufbau des Dateisystems teilweise beschrieben und auf GitHub verfügbar. [9]

Ein weiteres Projekt auf GitHub von dem Benutzer DmytroMoisiuk, ebenfalls ein Python-Programm, sucht anhand einer Signatur nach Videodaten und versucht diese zu verknüpfen. Dieses Programm arbeitet jedoch als reiner Carver, da auf die Struktur des Dateisystems nicht eingegangen wird. [10]

Von der Seite des Herstellers DAHUA existiert das Programm SmartPlayer. [11] Das Programm ist in der Lage die vorhandenen Videodateien auf einer Festplatte des Dateisystems DHFS 4.1 anzuzeigen und diese im proprietären Dateiformat (.dav) auf einem Computer zu speichern. Die Wiederherstellung von gelöschten Dateien findet nicht statt.

IV. Dateisystemanalyse

Dieser Abschnitt beschreibt den Aufbau des Dateisystems DAHUA DHFS 4.1. Dabei wird die Methodik von Carrier verwendet, das Dateisystem in verschiedene Bestandteile zu gliedern. Ähnlich wie bei Lee Tobin et al. wurden die meisten Elemente in der Dateisystemstruktur über Tests und manuelles Absuchen der Sektoren ermittelt. Es wurde auf immer wiederkehrende Muster, Bereiche und Werte geachtet um somit die gesamte Struktur erkennen zu können.

Ein klassisches Dateisystem besteht in der Regel aus mehreren Teilen, welche die verschiedenen Bereiche beschreiben. Es gibt einen Bootsektor, ein Hauptverzeichnis und Bereiche, wo die Daten und Metadaten gespeichert werden. Damit ein Betriebssystem weiß wo ein Dateisystem anfängt, liest es eine Partitionstabelle, z. B. den Master Boot Record (MBR) oder die GUID Partition Table (GPT), welche den Einsprung und die Größe des Dateisystems beinhaltet.

Carrier verwendet für die Analyse eines Dateisystems fünf Kategorien: Dateisystem, Dateiinhalt, Metadaten, Dateiname und Anwendungsdaten (im orig.: file system, content, metadata, file name, application). Jede dieser Kategorien beinhalten Daten die verschiedene Aufgaben erfüllen, z .B. das Auffinden wichtiger Sektionen, die Einstiegsadresse von Daten etc. Im folgenden werden die einzelnen Kategorien anhand des DHFS 4.1 Dateisystems beschrieben.

A. Dateisystem-Kategorie

Daten über ein Dateisystem sind essenzielle Daten, welche den Aufbau des Dateisystems beschreiben. Bspw. werden die Größe des Dateisystem, die Größe einer Allokationseinheit (z. B. Cluster oder Inode) oder die Adressen der Daten- oder Metadaten-Sektionen beschrieben.

Essenzielle Daten werden von nicht essenziellen Daten insofern unterschieden, dass diesen „vertraut“ werden muss. Sind die Daten korrupt oder nicht vorhanden kann eine Analyse des Dateisystems nicht ohne weiteres stattfinden.

Um die Werte der folgenden Abschnitte korrekt zu lesen muss vorher die Byte-Reihenfolge, auch Endianness genannt, bestimmt werden. Im DHFS 4.1 werden Texte und Signaturen im Big-Endian-, Zahlenwerte im Little-Endian-Format gelesen.

Der erste Sektor eines mit DHFS 4.1 formatierten Datenspeichers beginnt mit der Signatur 0x44 48 46 53 34 2E 31, was für „DHFS4.1“ steht. Über diese Signatur lässt sich feststellen, ob das DHFS 4.1 vorhanden ist, siehe Abbildung 1.

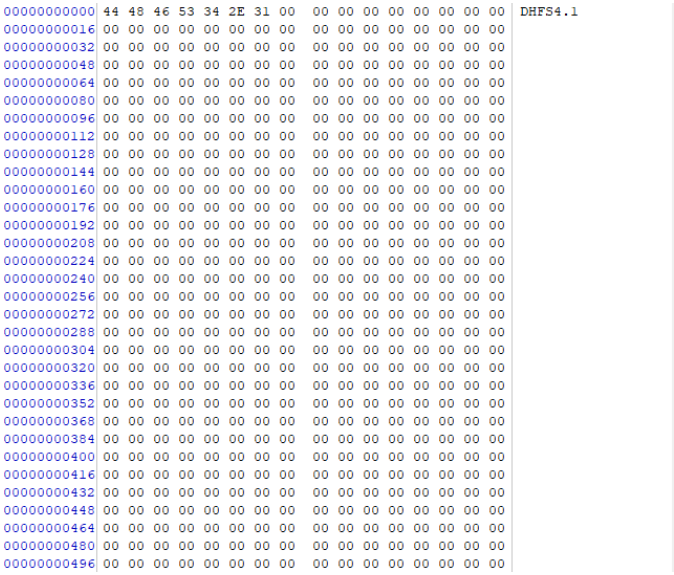

Abbildung 1. Erster Sektor von DHFS 4.1

Das DHFS 4.1 besitzt zwei Sektionen welche Daten über das Dateisystem beinhalten: eine Partitionstabelle und ein Bootsektor für jede Partition. Normalerweise zählt die Partitionstabelle eines Datenträgers nicht zu den Daten über das Dateisystem, da diese nur den Einsprung zu einem Dateisystem enthält.

Im DHFS 4.1 ist jedoch der besondere Umstand, dass die Partitionstabelle im Kontext der ersten Partition zu finden ist. Somit wird diese hier zu den Daten des Dateisystems gezählt.

1) Partitionstabelle: Die Partitionstabelle ist im Sektor 30 vom Anfang des Datenträgers aufzufinden. Die ersten 32 Byte sind unbekannt und werden für die Analyse nicht benötigt.

Tabelle I
BYTE-OFFSETS PARTITIONSTABELLE

Offset	Beschreibung	Farbe
0x08 - 0x0B	Bootsektoroffset	
0x24 - 0x2B	Partitionsoffset	
0x2C - 0x2F	Partitionslänge	
0x134 - 0x137	Magic-Number	

Es folgen anschließend 32 Byte Einträge, welche die einzelnen Partitionen des Dateisystems beschreiben. Zu den

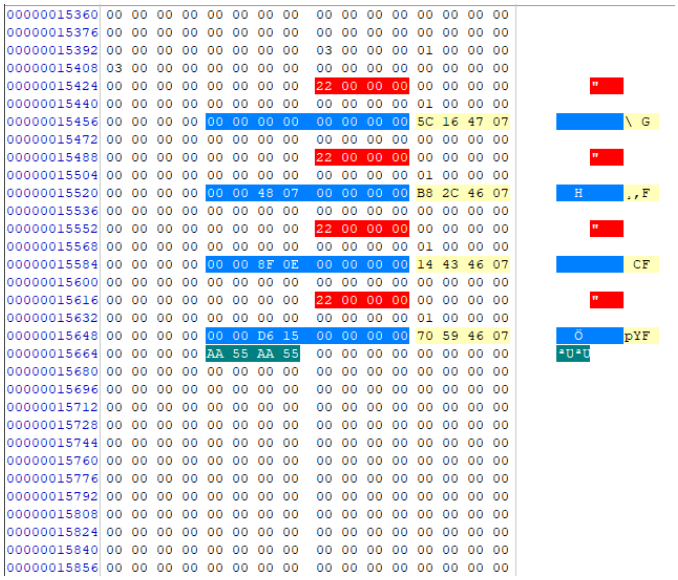

Abbildung 2. Partitionstabelle

wichtigsten Informationen zählen die Adresse des Bootsektors, die Länge der Partition und dessen Einsprung, siehe Tabelle I und Abbildung 2. Alle Werte sind in der Einheit Sektoren zu interpretieren, wobei ein Sektor 512 Byte umfasst.

Das Ende der Partitionstabelle wird mit dem hexadezimalen Wert 0xAA55AA55, auch Magic-Number genannt, gekennzeichnet. Es folgen darauf keine weitere Partitionen.

Es ist zu vermuten, dass eine Partition eine maximale Größe von 2^{32} Sektoren besitzt. Demnach können maximal 4.294.967.296 Sektoren * 512 Byte, also 2 TB für eine Partition verwendet werden. Die untersuchten Festplattenabbilder besaßen in der Regel 4 Partitionen, sodass in diesem Fall theoretisch Datenträger mit einer Größe von 8 TB für dieses Dateisystem verwendet werden können.

2) Bootsektor einer Partition: Der Bootsektor, welcher in der Regel im Sektor 34 ab Partitionsanfang aufzufinden ist, enthält essenzielle Informationen über die Einsprünge zu den Daten- und Metadaten-Kategorien sowie zu den Größen der Allokationseinheiten, siehe Abbildung 3.

Jeder Bootsektor besitzt mehrere Zeitstempel, u. A. der vermeintliche Erzeugungszeitstempel der Partition oder der Logdatei und zwei Zeitstempel welche den Aufnahmezeitraum für die jeweilige Partition eingrenzen.

Die Größe für einen Sektor sowie die Größe einer Allokationseinheit werden im Bootsektor definiert. Anlehnend zu NTFS und FAT werden diese Allokationseinheiten in dieser Arbeit ebenfalls Cluster genannt. Es zeigte sich, dass die Größen bei den untersuchten Festplattenabbildern mit 512 Bytes pro Sektor sowie 4096 Sektoren pro Cluster stets

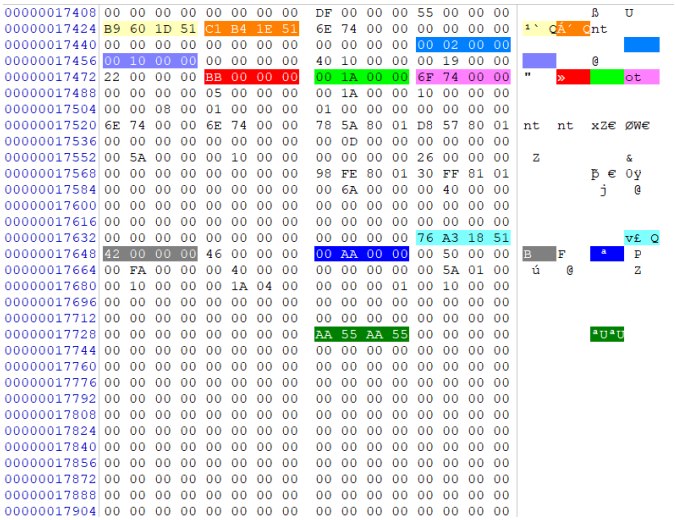

Abbildung 3. Bootsektor der ersten Partition

identisch war. Weitere Werte können in Tabelle II eingesehen werden.

Tabelle II
BYTE-OFFSETS BOOTSEKTOR

Offset	Beschreibung	Farbe
0x10 - 0x13	Aufnahmedatum von	
0x14 - 0x17	Aufnahmedatum bis	
0x2C - 0x2F	Byte pro Sektor	
0x30 - 0x33	Sektor pro Cluster	
0x44 - 0x47	Einsprung Deskriptortabelle	
0x48 - 0x4B	Einsprung Videodatenbereich	
0x4C - 0x4F	Anzahl Einträge Deskriptortabelle	
0xEC - 0xEF	Erzeugungsdatum Log	
0xF0 - 0xF3	Einsprung Reserve-Bootsektor	
0xF8 - 0xFB	Einsprung Logdateien	
0x148 - 0x14B	Magic-Number	

Der Einsprung zu dem Videodatenbreich sowie der Beginn der Deskriptortabelle, welche die einzelnen Videodateien beschreibt ist in der Regel bei allen Bootsektoren gleich, kann jedoch von Datenträger zu Datenträger abweichen. Der Datenbereich in Abbildung 3 beginnt ab Sektor 6656, die Deskriptortabelle ab Sektor 187. Die Größe der Deskriptortabelle weicht abhängig von der Belegung ab, da der Wert die Anzahl der belegten 32 Byte Werte wiedergibt.

Neben den Einsprüngen zu den Daten- und Metadaten-Kategorien besitzt jede Partition einen Einsprung zu einer eigenen Logdatei. Der Einsprung zu dieser Logdatei ist hier bspw. der Sektor 43520 und wird in der Kategorie Anwendungsdaten beschrieben.

Wie in der Partitionstabelle markiert der Marker 0xAA55AA55 das Ende des Bootsektors.

Jede Partition besitzt neben dem Bootsektor im Sektor 34 auch einen Reserve-Bootsektor, der Einsprung ist im Offset 0xF0 - 0xF4 der jeweiligen Partition definiert. Dieser ist eine 1:1 Kopie des ersten Bootsektors und kann im Falle einer Beschädigung oder Korruption des 1. Bootsektors ausgelesen werden.

Die Bootsektoren von DHFS 4.1 haben gewisse Ähnlichkeiten zu dem Bootsektor einer NTFS-Partition, da ein ähnlicher End-Marker verwendet wird und eine Kopie des Bootsektor vorhanden ist. Ebenfalls werden die Sektor- und Clustergrößen definiert.

3) Deskriptortabelle: Die Deskriptortabelle ist eine der wichtigsten Datenstruktur von DHFS 4.1. Die Funktion dieser Datenstruktur fließt ebenfalls in die Kategorien Daten und Metadaten an, welche später behandelt werden.

Im Grunde ist die Deskriptortabelle eine fortlaufende, doppelt verkettete Liste, welche mit dem Index bzw. der ID 0 beginnt. Jeder Listeneintrag ist 32 Byte groß und beinhaltet den Allokationsstatus eines DHFS 4.1 Clusters. Ein Eintrag bzw. Deskriptor ist entweder frei (0xFE), ein Hauptvideodeskriptor (0x01) oder ein Videofragmentdeskriptor (0x02), siehe Abbildungen 4 bis 6. Andere Belegungstypen sind nicht bekannt.

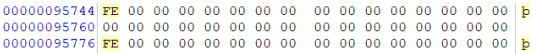

Abbildung 4. Leerer Deskriptor

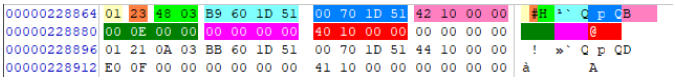

Abbildung 5. Hauptvideodeskriptor

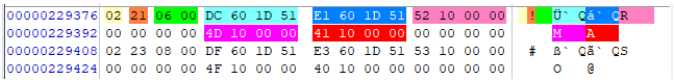

Abbildung 6. Videofragmentdeskriptor

Der Typ Hauptvideodeskriptor unterscheidet sich zum Videofragmentdeskriptor insoweit, da dieser als Art Einsprung für die Clusterkette dient. Er beschreibt die gesamte Länge des Videos, den gewählten Kanal (d. h. die gewählte Kamera), die Anzahl der Fragmente und die ID bzw. dadurch den (partitionsweiten) einzigartigen Namen des Videos. Zudem beschreibt dieser die Größe des letzten Fragments.

Der Aufnahmezeitraum für einen Hauptvideodeskriptor ist jeweils eine Stunde mit der Ausnahme, dass eine Videoaufnahme unterbrochen wurde. Für die Bestimmung der

Zeit verwendet das DHFS 4.1 eine vom UNIX-Zeitstempel abweichende Definition, welche in der Metadaten-Kategorie beschrieben wird.

Der Videofragmentdeskriptor beschreibt ein einzelnes Fragment, welches einem Hauptvideodeskriptor zugeordet wird. Es enthält die Informationen zu den Fragmenten, die vor oder nach ihm folgen, einen Zeitraum der Aufnahme sowie den Bezug zum Hauptvideodeskriptor.

Die Datenstrukturen der Hauptvideodeskriptoren und Videofragmentdeskriptoren unterscheiden sich in einigen wenigen Punkten. Der Hauptvideodeskriptor speichert die Anzahl der vorhandenen Videofragmente sowie die Größe des letzten Videofragmentes und hat keinen Vorgänger. Der Videofragmentdeskriptor speichert seinen Vorgänger und die interne Nummer des Fragments, um eine korrekte Reihenfolge zu gewährleisten. Die Unterschiede werden in den Tabellen III und IV dargestellt.

Tabelle III
BYTE-OFFSETS HAUPTVIDEODESKRIPTOR

Offset	Beschreibung	Farbe
0x00 - 0x00	Status	
0x01 - 0x01	Kanal	
0x02 - 0x03	Anzahl der Videofragmente	
0x04 - 0x07	Anfangsdatum	
0x08 - 0x0B	Enddatum	
0x0C - 0x0F	Nächste Deskriptor-ID	
0x10 - 0x11	Größe des letzten Videofragments	
0x14 - 0x17	Null	
0x18 - 0x1B	Video-ID	

Tabelle IV
BYTE-OFFSETS VIDEOFRAGMENTDESKRIPTOR

Offset	Beschreibung	Farbe
0x00 - 0x00	Status	
0x01 - 0x01	Kanal	
0x02 - 0x03	Interne Nummer des Videofragments	
0x04 - 0x07	Anfangsdatum	
0x08 - 0x0B	Enddatum	
0x0C - 0x0F	Nächste Deskriptor-ID	
0x14 - 0x17	Vorherige Deskriptor-ID	
0x18 - 0x1B	Video-ID	

Der Wert für den Kanal muss mit einer UND-Verknüpfung mit dem Wert 0x0F und der Addition der Zahl 1 umgewandelt werden. Beispielsweise wird in Abbildung 5 der Wert 0x23 über die Rechnung (0x23 & 0xF) + 1 zum Wert 4 gewandelt.

Über die Deskriptortabelle wird deutlich, dass die gespeicherten Videoaufnahmen im Dateisystem fragmentiert abgelegt werden können. Es verwendet wie FAT eine Clusterkette, welche den Allokationsstatus eines Clusters sowie die in der Kette befindlichen Cluster referenziert. Jede Partition besitzt nur eine Deskriptortabelle, eine Kopie konnte nicht gefunden werden. Eine Bitmap zur Bestimmung des Allokationsstatus, wie sie z. B. bei NTFS verwendet wird, ist somit nicht notwendig.

B. Daten-Kategorie

Zu der Daten-Kategorie werden jegliche Daten eingeordnet, welche zur Bestimmung der Daten benötigt werden. Dazu zählen z. B. die Größen der Allokationseinheiten und die sogenannte logische Dateisystemadresse.

Der Bootsektor einer DHFS 4.1 Partition definiert die Größe der Bytes pro Sektor und auch die Anzahl der Sektoren pro Cluster. Ein Cluster ist damit die Allokationeinheit, welche zur Speicherung und Bestimmung der Größe einer Videodatei benötigt wird.

Die Nummerierung der Cluster ist identisch mit der Nummerierung in der Deskriptortabelle. Der 1. Deskriptor spiegelt den 1. Cluster wieder, der 2. Deskriptor den 2. Cluster usw.

Da die Deskriptortabelle eine doppelt verkette Liste aus Hauptdeskriptoren und Videofragmentdeskriptoren ist kann die Größe des Datenbereichs bestimmt werden. Diese setzt sich zusammen aus der Anzahl der vorhandenen Deskriptoren multipliziert mit der Anzahl der Sektoren / Cluster.

Die Cluster werden ab der logischen Dateisystem Adresse 0 bis n - 1 gezählt. Eine Umrechnung auf die logische Block Adresse (LBA) erfolgt über die Summe der Einsprünge zu der Partition und des Datenbereichs, addiert mit dem Produkt aus Clustergröße und Deskriptor bzw. Clusters. Die Formel wird folgend dargestellt:

$$LBA = Partitionsanfang + Videodatenbereichsanfang + (Clustergroesse * Deskriptornummer) \quad (1)$$

Somit kann zu jedem Cluster die genaue LBA bestimmt werden, was essenziell für die Wiederherstellung der Fragmentierung ist.

Zur Bestimmung welche Fragmente zu einem Hauptdeskriptor zugeordnet sind, sind ab dem Hauptdeskriptor alle Fragmente wie eine verkettete Liste zu traversieren. Sobald die Anzahl der Fragmente des Videos erreicht ist oder das letzte Fragment keinen Nachfolger hat (End-Of-File, EOF) ist die Kette vollständig bestimmt.

Das letzte Fragment hat zudem die Besonderheit, dass die Größe variiert. Im Hauptdeskriptor ist die Größe des letzten Fragments angegeben, sodass z. B. das letzte Fragment nur 3241 Sektoren anstatt 4096 reserviert. Durch diese Bedingung kann ein sogenannter Slack-Bereich (Slack-Space) entstehen, welcher weiter analysiert werden kann.

Anhand der Nummer der Deskriptoren können so nun die Cluster und damit die LBA bestimmt werden, um eine Datei aus dem Dateisystem zu extrahieren.

Wie genau das Dateisystem entscheidet, welcher Deskriptor als nächstes belegt wird ist nicht bekannt und scheint nach der Reihenfolge zu gehen, wann ein aktiv beschriebener Cluster vollkommen belegt ist. Dadurch, dass mehrere Kanäle gleichzeitig aufgenommen werden können die Cluster eines Videos ggf. nicht zusammenhängend belegt werden, wodurch die Fragmentierung entsteht.

Auch die Technik zur Löschung bzw. Freigabe von Deskriptoren ist nicht eindeutig erkennbar. In den meisten Videoüberwachungssystemen werden die Aufnahmen nach einer bestimmten Zeit, oft 7 oder 14 Tage, sukzessive überschrieben. Ob das Löschen einzelner Videoaufnahmen die Deskriptoren auf den freien Zustand (0xFE) zurücksetzen ist nicht bekannt.

C. Metadaten-Kategorie

Metadaten beschreiben die Daten welche auf einem Dateisystem liegen. So gesehen sind es „Daten über Daten“ und beinhalten bspw. Zeitstempel oder Adressen zu den Datenbereichen, die eine Datei allokiert.

Im DHFS 4.1 existieren nur eine kleine Anzahl von Metadaten, das Aufnahmedatum eines Hauptvideodeskriptors, in der Regel im Zeitraum einer Stunde, und die Zeitstempel der dazugehörigen Videofragmentdeskriptoren, welche im Minuten oder Sekundenzeiträumen einzugrenzen sind.

Eine direkte logische Datensystemadresse existiert nicht, jedoch kann, wie in Formel 1 beschrieben, anhand der Deskriptor-ID die Adresse durch eine Umrechnung herausgefunden werden.

1) Zeitstempel: Die Zeitstempel im DHFS 4.1 sind jeweils 32-Bit groß und recht ähnlich zu den UNIX-Zeitstempeln, welche ebenfalls 32-Bit groß sind. Der UNIX-Zeitstempel rechnet in Sekunden ab dem 01.01.1970, sodass jede Sekunde ab diesem Zeitpunkt gezählt wird. [12] Eine Minute entspricht demnach 60 Sekunden, eine Stunde somit dem Wert von 3600 Sekunden usw.

Als Beispiel entspricht das Datum 11.08.2025 11:54:27 (GMT +2) dem 32-Bit Wert 1754906067. Zeitzonen werden mit Greenwich Mean Time (GMT) +X angegeben, sodass bspw. die mitteleuropäische Sommerzeit (MESZ) als GMT +2 angegeben wird.

Der DHFS 4.1 Zeitstempel rechnet anders, eine Stunde ist äquivalent zu dem Wert 4096. Daher muss für die Bestimmung der genauen Uhrzeit, also Jahr, Monat, Tag, Stunde, Minute und Sekunde die Bit-Maske von Tabelle V auf den 32-Bit Wert angewandt werden.

Tabelle V
ZEITSTEMPEL BITMASKE

Bits	Länge	Beschreibung
31–26	6	Jahr
25–22	4	Monat
21-17	5	Tag
16–12	5	Stunde
11-6	6	Minute
5–0	6	Sekunde

Das Jahr enthält jedoch nur die letzten beiden Ziffern des vollen Jahrs, sodass der Wert 2000 hinzu addiert werden muss.

Bspw. wird der Zeitstempel $0x76A31851_{LE}$, welcher dem dezimalen Wert 1360569206 entspricht, in Tabelle VI umgerechnet.

Tabelle VI
UMRECHNUNG 1360569206

Stelle	Wert
Jahr	20
Monat	4
Tag	12
Stunde	10
Minute	13
Sekunde	54

Addiert man auf das Jahr den Wert 2000 erhält man das lesbare Datum 12.04.2020 10:13:54 Uhr. Als Zeitzone wird hier GMT +0 angenommen, sodass der Zeitzonenversatz manuell hinzugefügt werden muss.

Außer den Zeitstempeln sind keine weiteren Metadaten bekannt. Es existieren keine Zugriffs-, Erstellungs- und Änderungszeitstempel wie bei den Dateisystemen NTFS oder FAT. Es ist ebenfalls unbekannt wieso die meisten Videosequenzen im Zeitraum einer Stunde aufgezeichnet werden, außer wenn die Aufnahmen abgebrochen werden.

D. Dateiname-Kategorie

Im DHFS 4.1 existieren keine direkt Dateinamen, wie man sie aus anderen Dateisystemen gewohnt ist. Auch Pfade oder Ordnerstrukturen konnten nicht erkannt werden. Vielmehr sind die Dateien auf einer Partition in einer Art flachen Liste angelegt, sodass alle Videodateien im (virtuellen) Wurzelpfad einer Partition zu finden sind.

Es wird vermutet, dass die Dateinamen aus verschiedenen Elementen zusammengefügt werden, u. A. die Nummer der Partition, der Aufnahmekanal, das Anfangs- und Enddatum der Videoaufnahme sowie der Nummer des Hauptvideodeskriptors. Dieses Schema wurde ebenfalls für die Programmierung der

X-Ways Forensics X-Tension gewählt, sodass die Videodateien eindeutig identifiziert werden können.

Einzig das von DAHUA entwickelte Programm zeigt den vermeintlichen Pfad zur Videodatei an. Dieser besteht teilweise aus den o. g. Elementen, enthält jedoch auch unbekannte Nummern, welche nicht zugeordnet werden können, siehe Abbildung 7.

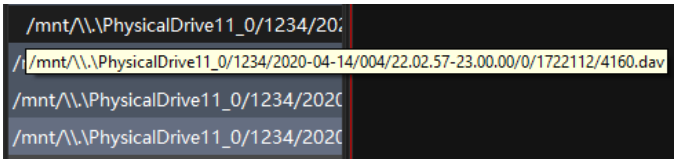

Abbildung 7. SmartPlayer Dateiname

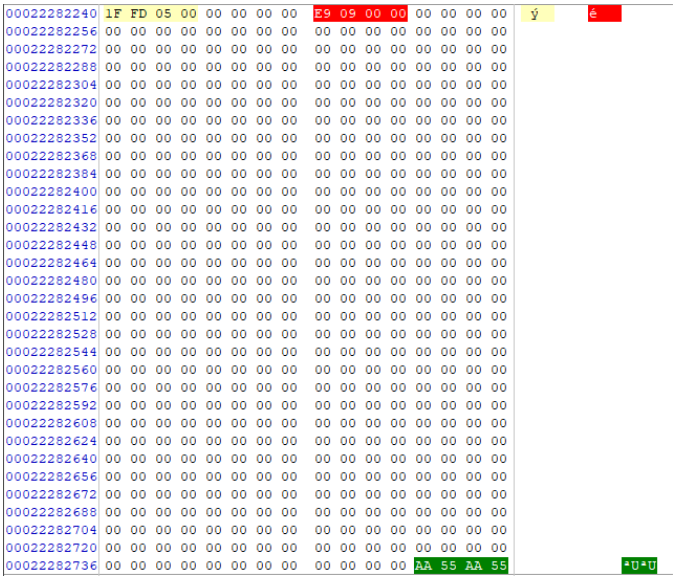

Abbildung 8. Logdatei-Header

E. Anwendungsdaten-Kategorie

Zu den Anwendungsdaten eines Dateisystems zählen jegliche Daten, die für das Dateisystem zu Gewinnung von Daten nicht relevant sind, jedoch zur Gewinnung von Informationen über eine Datei relevant sein können.

Typische Anwendungsdaten sind u. a. Dateisystemen Journals, welche z. B. den Zustand einer Transaktion (Erzeugen, Schreiben) festhalten und bei Inkonsistenzen umkehren können.

Das DHFS 4.1 hat kein Dateisystem Journal, enthält jedoch eine Logdatei, welche den Wechsel der Partitionen beschreibt. Weiterhin haben die Videodateien, welche im proprietären DAV-Format gespeichert werden wichtige Informationen über den Aufbau der Datei, sodass diese Elemente kurz angesprochen werden.

1) Logdateien: Jede DHFS 4.1 Partition enthält einen Bereich für eine Logdatei. Diese ist im Bootsektor auslesbar und war bei allen untersuchten Datenträgerabbilder der Sektor 43520 vom Anfang der Partition.

In diesem Sektor befindet sich eine Datenstruktur, welche den Log-Bereich des Dateisystems beschreibt, siehe Abbildung 8. Sie wird hier Logdatei-Header genannt.

Die Datenstruktur enthält nur zwei bekannte und wesentliche Werte: Die Länge der Logdatei in Bytes und die Anzahl der Log-Einträge, siehe Tabelle VII. Am Ende des Sektor lässt sich wieder das Magic-Byte 0x55AA55AA wiederfinden, was den Abschluss des Headers markiert.

Tabelle VII
BYTE-OFFSETS LOGDATEI-HEADER

Offset	Beschreibung	Farbe
0x00 - 0x03	Länge in Bytes (32 Byte)	
0x08 - 0x0B	Anzahl Logeinträge	
0x1FC - 0x1FF	Magic-Number	

Nach der Datenstruktur folgt eine 1:1 Kopie des Sektors. Anschließend können die Log-Daten eingesehen werden. Diese sind, anders als bei den Videodateien, nicht fragmentiert und können durchgehend gelesen werden. Der Dateiinhalt ist einfacher Text, siehe Abbildung 9.

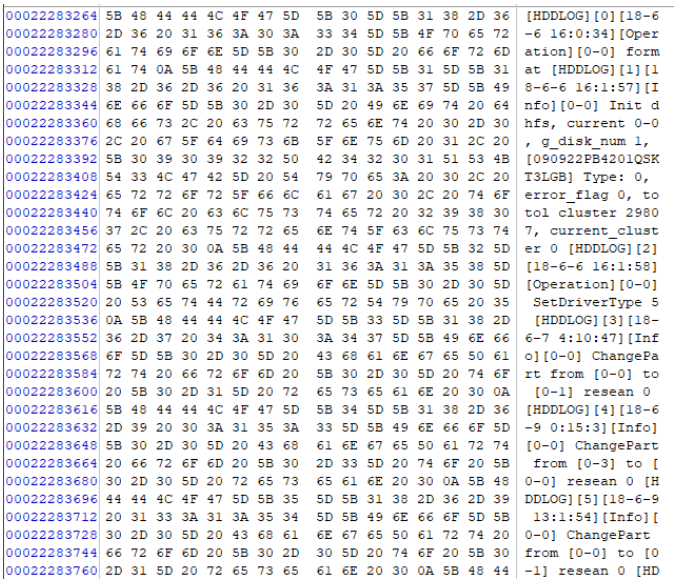

Abbildung 9. Logdatei Inhalt

Eine genaue Logdatei-Analyse liegt außerhalb des Bereichs dieser Arbeit, es wird aber vermutet, dass die Logdateien u. A. Informationen über die Rotation der Partitionen oder ehemals existierende Aufnahmedaten beinhalten.

2) DHII-Datenstruktur: Folgt man der Clusterkette eines Hauptvideodeskriptors ab dem 1. Videofragment anhand der im Abschnitt IV-B genannten Berechnung, landet man zunächst nicht direkt auf Videodaten sondern bei einem Sektor der eine weitere, beschreibende Datenstruktur enthält. Die Datenstruktur wird hier DHII-Datenstruktur genannt, da die ersten 4 Bytes dieser Struktur der hexadezimalen Wert 0x44484949 enthält, welche den ASCII Zeichen DHII entsprechen, siehe Abbildung 10.

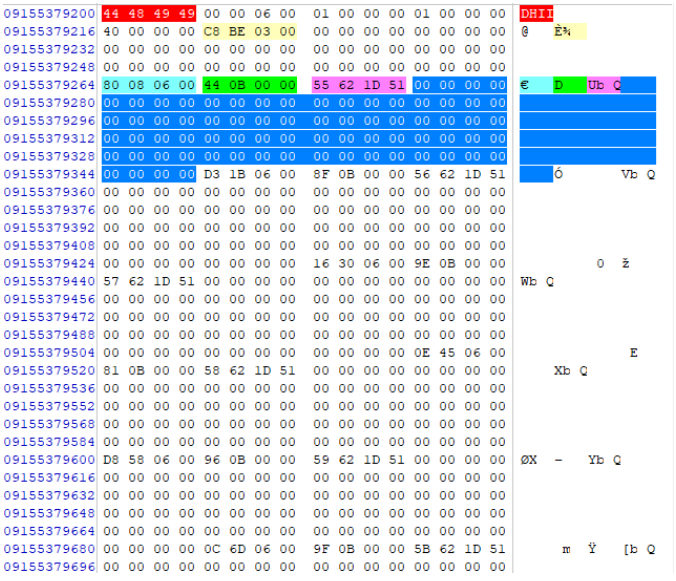

Abbildung 10. DHII-Datenstruktur

Diese Datenstruktur besteht aus zwei Teilen, einem Header (64 Byte) und einer Liste mit jeweils 84 Byte großen Einträgen. Die Anzahl der Einträge kann aus dem Header ab Byte 0x14 entnommen werden, siehe dazu Tabelle VIII.

Tabelle VIII
BYTE-OFFSETS DHII-HEADER

Offset	Beschreibung	Farbe
0x00 - 0x03	Signatur	
0x18 - 0x1B	Anzahl Listeneinträge	

Ein Listeneintrag in dieser Datenstruktur beschreibt vermeintlich die genaue Position eines Videoframes in der gesamten Videodatei. Dazu werden drei Werte benötigt, den Offset ab dem Anfang der Listenstruktur, die Länge des Videoframes und der Zeitstempel des Videoframes. Diese können anhand der Offsets in Tabelle IX ausgelesen werden. Die restlichen 72 Bytes werden nicht benötigt, grenzen aber den aktuellen Eintrag von den nächsten Einträgen ab.

Tabelle IX
BYTE-OFFSETS DHII-LISTENEINTRAG

Offset	Beschreibung	Farbe
0x00 - 0x03	Offset zum Videoframe	
0x04 - 0x07	Länge des Videoframes	
0x08 - 0x0B	Anfangsdatum des Videoframes	
0x0C - 0x53	Unbekannt	

Zu beachten ist, dass die Offsets nur für ein defragmentiertes Video gültig sind. Die Videodatei muss also zunächst zusammen gesetzt werden, bevor die Offsets zu den Videoframes angewendet werden können.

Tests ergaben, dass die DHII Datenstruktur zum Abspielen des Videos nicht benötigt wird und diese sogar das Abspielen des Videos behindert. Es reicht aus, wenn die Videodatei ab dem ersten Videoframe zusammengesetzt wird. Die Videoframes werden im folgenden Abschnitt erläutert.

3) Videoframes: Folgt man nun den Offsets in den Listeneinträger der DHII-Datenstruktur lassen sich die einzelnen Videoframes einer DAV-Videodatei auffinden. Abbildung 11 zeigt, wie der Anfang eines Frames im Dateisystem dargestellt wird.

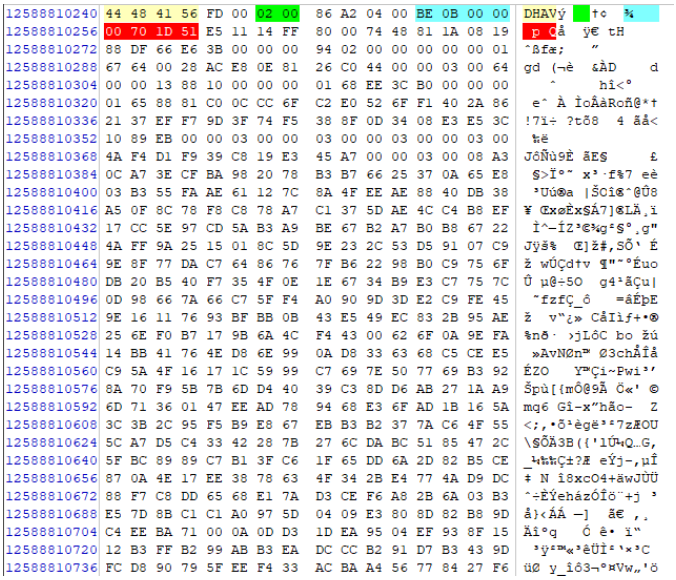

Abbildung 11. Anfang einer DHAV Videodatei / Videoframes

Über den Aufbau eines DHAV Videoframes ist nicht viel bekannt und die hier ermittelten Werte beruhen auf Vermutungen. In Tabelle X werden einige Werte genannte, welche sich mit den Werten aus der DHII-Datenstruktur und der Deskriptortabelle decken.

Tabelle X
BYTE-OFFSETS DHAV VIDEOFRAME

Offset	Beschreibung	Farbe
0x00 - 0x03	Signatur	
0x06 - 0x07	Kanal	
0x0C - 0x0F	Größe des Videoframes	
0x10 - 0x13	Anfangsdatum	

Zwischen den Signaturen am Anfang (0x44484156, DHAV) und am Ende (0x64686176, dhav) (siehe Abbildung 12) liegen die Nutzdaten des Videoframes, welche vermutlich in H.264 kodiert sind. Eine weitere Analyse der Nutz- bzw. Videodaten liegt außerhalb dieser Arbeit und wird nicht weiter thematisiert.

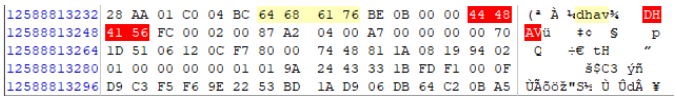

Abbildung 12. Ende einer DHAV Videodatei / Videoframes mit anschließender DHAV Signatur

V. SLACKSPACE UND CARVING VON VIDEODATEIEN

Da nun die Struktur des DHFS 4.1 besprochen wurde können verschiedene forensische Aspekte angesprochen werden. Ziel einer jeden forensischen Untersuchung ist die vollständige Suche nach belastenden und entlastenden Beweisen, in diesem Falle Videodaten. Somit fallen nicht nur vorhandene, sondern auch scheinbar gelöschte Daten in den Fokus einer Ermittlungsperson.

Durch die Analyse des Dateisystems ist ersichtlich, dass es zwei Bereiche gibt wo eine Ermittlungsperson weitere, evtl. gelöschte Videodateien auffinden kann: Im Slackspace des letzten Fragments der Videofragmentliste oder durch das Carving in den Clustern freier Deskriptor-IDs. Die Begriffe Slackspace und Carving werden folgend kurz erläutert.

Der Slackspace ist ein Speicherbereich, der bei der Allokierung und Nutzung von Dateneinheiten entsteht. Werden bspw. für eine Datei 2 Cluster á 1024 Bytes (insgesamt also 2048 Bytes) reserviert jedoch nur 1536 Bytes verwendet, verbleiben noch 512 Byte, welche verwendet werden können. Interessanter wird es, wenn das Betriebssystem bei der Allokierung den Speicher nicht überschreibt und somit der Inhalt der zuvor vorhandenen Datei weiterhin vorhanden ist.

Als Carving bezeichnet man den Prozess der Programmdaten-basierten Dateiwiederherstellung. [2] Grundsätzlich haben viele Dateien, wie auch die DHAV Videoframes, eine feste Datenstruktur mit einem Header und einem Footer, welche den Anfang und das Ende einer Datei oder Datenstruktur markieren. Beim Carving sucht ein Programm, ein so genannter Carver, nach diesen Signaturen und wendet verschiedene Algorithmen und Wiederherstellungsmethoden an um die Dateien ohne Hilfe von Metadaten wiederherzustellen. Schwierigkeiten bereiten fragmentierte Dateien, da keine Zuordnung der Dateifragmente und Cluster besteht und somit zwischen dem Header und dem Footer Dateiinhalte gelesen werden können, die nicht zur gesuchten Datei gehören.

Folgend werden die Methoden besprochen welche angewandt werden können um bei dem DHFS 4.1 Videodateien in den o. g. Bereichen zu finden.

A. *Slackspace des letzten Videofragments*

Da über die Hauptdeskriptoren bekannt ist, wie groß das letzte Fragment in der Videofragmentliste ist, kann dieser Bereich einfach extrahiert und untersucht werden. Vorzugsweise kann hier ein Carver angewandt werden um den Header und den Footer eines DHAV Videoframes einzugrenzen.

B. *Carving freier Cluster*

Ein einfacherer Ansatz des Carvings ist die Cluster bzw. den Slackspace nach dem ersten Vorkommen eines Headers abzusuchen und die Suche für diesen Cluster abzubrechen. Die Informationen, u. A. der Zeitstempel des Videoframes aller Erstvorkommen werden analysiert und chronologisch zusammengefasst, sodass eine neue Clusterkette entsteht. Die Idee dahinter ist anzunehmen, dass eine (ehemalige) Videodatei einen Cluster durchgehend beschrieben hat, da in der Regel eine Aufnahmedauer von einer Stunde vorgesehen ist.

Nachteil dieser Methode ist, dass andere DHAV Header schlichtweg übersehen und im falschen Datum eingruppiert werden und / oder die Videodatei dadurch nicht oder nur schlecht abspielbar wird, da Sprünge in den Zeiten vorhanden sind.

Eine effizientere Methode ist das Carving anhand der einzelnen Videoframes. Ein Carver könnte alle gültigen Videoframes aufsuchen und diese zusammenfassen, sodass aus den Fragmenten anhand der gewonnen Uhrzeiten und Kanälen Teile eines Videos wiederhergestellt werden können.

Problematisch wird die angesprochene Fragmentierung der Cluster, da hier verschiedene Faktoren überdacht werden müssen. Bei einer anzunehmenden starken Fragmentierung beschreibt ein Kanal bei der Konfiguration 512 Byte pro Sektor und 4096 Sektoren pro Cluster maximal 4096 * 512 Byte = 2 MB.

Sollte nun ein Videoframe eine Größe oder einen Offset im Cluster besitzen welche über die Clustergrenze hinweg ist, wäre eine Wiederherstellung dieses Videoframes nur bedingt möglich. Der Footer, welcher in einem anderen Cluster liegen könnte enthält als einzige Referenz eine Größenangabe, die mit einem potenziellen Header verglichen werden muss.

Zur Herstellung fragmentierter Videoframes können mehrere Annahmen getroffen werden:

1) Die definierte Größe im Footer muss mit der Größe im Headers übereinstimmen. Stimmt diese nicht überein, gehört der Footer nicht zu dem Video und vice versa.

2) Die durch die Fragmentierung abgeschnittenen Bytes ab dem Header entsprechen dem Offset in einem anderen Deskriptor bis der Footer gefunden wird.

   Wenn durch die Fragmentierung bspw. 34 Bytes vom Header für die volle Länge fehlen und ab dem Anfang eines anderen, naheliegenden Deskriptors mit einem Offset von 34 Bytes die Endsignatur gefunden wird, welche die gleiche Größenangabe hat ist die Wahrscheinlichkeit sehr groß, dass der Header zum Footer passt und vice versa.

Allgemein sollte für jeden Videoframe das Datum validiert werden, um zufällige Treffer der Header Signatur zu vermeiden Zudem sollte ein unvollständiger Header in der unmittelbaren Entfernung zu einem unvollständigen Footer liegen.

Da der Inhalt zwischen dem Header und Footer unbekannt ist, ist der Carver stark abhängig von dem Header oder der Signatur DHAV. Ist diese aufgrund einer Überschreibung nicht vorhanden können ganze Videoframes nicht gefunden und extrahiert werden.

VI. Entwicklung der X-Ways Forensics X-Tension

Basierend auf den Erkenntnissen der Dateisystemanalyse kann nun eine X-Tension für das forensische Programm X-Ways Forensics entwickelt werden.

Zur Entwicklung stellt X-Ways Forensics einem Entwickler Schnittstellen (APIs) zur Verfügung, mit denen die X-Tension mit dem Programm und vice versa interagieren kann. Das Ergebnis ist eine DLL-Datei (Dynamic Link Library), welche in dynamisch in das Programm eingebunden und ausgeführt wird.

Als Programmiersprache kann C oder C++ verwendet werden. In der hier entwickelten X-Tension wird eine Mischung aus C sowie C++ Programmcode und somit Konstrukte aus beiden Sprachen verwendet.

Zunächst werden C-Struct definiert, welche der Datenstrukturen des DHFS 4.1 entsprechen. Bspw. wird der Bootsektor einer DHFS 4.1 Partition wie in Listing 1 dargestellt.

```
1   struct DHFS4_1_Bootsector {
2     uint32_t beginTime;
3     uint32_t endTime;
4     uint32_t sectorSize;
5     uint32_t clusterSize;
6     uint32_t descriptorTableOffset;
7     uint32_t descriptorTableItemcount;
8     uint32_t dataAreaOffset;
9     uint32_t logsOffset;
10  };
```

Listing 1. Bootsektor C-Struct

Weitere Datenstrukturen sind im Quellcode vorhanden und können dort eingesehen werden.

Damit die X-Tension mit dem Programm und vice versa interagieren müssen bestimmte Funktionen in der DLL exportiert werden. Diese speziellen Funktionen mit dem Präfix XT_* werden anschließend zur Laufzeit von X-Ways Forensics aufgerufen und initialisieren damit die X-Tension.

Funktionen mit dem Präfix XWF_* können aus der DLL aufgerufen werden, um mit X-Ways Forensics zu interagieren. Beispielsweise kann mit der Funktion XWF_Read(...) eine bestimmte Anzahl an Byte ab einer bestimmten Position in einen Speicherbereich gelesen und später verarbeitet werden.

Im weiteren Verlauf ist der Entwickler einer X-Tension für das Erstellen eines Verzeichnisbaums sowie das Lesen der Daten von der Quelle (Datenträger oder Abbild) verantwortlich. Auch hier liefert die X-Tension API wichtige Funktionen um bspw. Dateien zu erstellen und diese mit Daten zu befüllen.

Ein besonderes Augenmerk muss auf die Fragmentierung der Videodateien gelegt werden. Eine „normale“ X-Ways

X-Tension ist nicht in der Lage Dateien mit Fragmentierung zu verarbeiten. Dazu muss die X-Tension mit Funktionen der Disk I/O X-Tension API erweitert werden.

Eine X-Tension in diesem Modus ist zuständig für das Befüllen der Daten, welche dem Benutzer im Verzeichnisbrowser angezeigt werden. Zudem muss jeder Sektorzugriff verarbeitet werden, um den Benutzer die gesuchten Daten anzeigen zu können.

Die X-Tension für das DHFS 4.1 Dateisystem wurde so entwickelt, dass diese beide Funktionen abdeckt. Sie kann eingesetzt werden um den Verzeichnisbaum zu erzeugen und anschließend im Disk I/O Modus die Daten defragmentiert anzuzeigen, um die volle Funktionalität zu gewähren.

Nachdem die X-Tension ausgeführt wird werden die vorhandenen sowie gecarvten Videodateien, wenn vorhanden, angezeigt. Diese können anschließend eingesehen oder extrahiert werden um diese mit externen Videoverarbeitungsprogrammen abzuspielen.

1) Aufbau und Ablauf der X-Tension: Um das Dateisystem zu parsen müssen die einzelnen in der Dateisystemanalyse analysierten Bestandteile gelesen, zwischengespeichert und verarbeitet werden, sodass am Ende der Verarbeitung ein Dateibaum erzeugt werden kann welcher in X-Ways Forensics abgebildet wird. Folgend wird der Ablauf der X-Tension dargestellt.

1) Lesen der Partitionstabelle
2) Lesen des Bootsektors einer Partition
3) Lesen der Deskriptortabelle einer Partition
4) Carving der freien Deskriptoren
5) Carving des Slackspaces der belegten Deskriptoren

Die Schritte 2 bis 5 wiederholen sich für jede vorhandene Partition im Dateisystem.

Zum Lesen der einzelnen Datenstrukturen wurden, wie bereits erwähnt, C-Structs definiert, welche der Reihe nach befüllt und ausgelesen werden. Zunächst wird die Partitionstabelle, anschließend die jeweiligen Bootsektoren der Partitionen gelesen. Für jede Partition wird ein eigener Ordner angelegt, damit die Videodateien einer Partition dort abgelegt werden können, siehe Abbildung 13. Hinter den Namen der Partition wird die Anzahl der Unterobjekte angegeben.

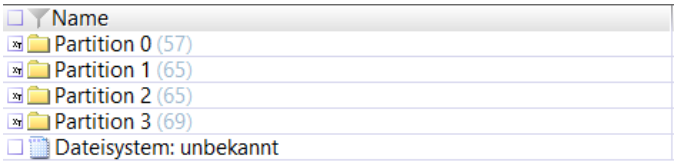

Abbildung 13. Partitionen als Ordner in X-Ways Forensics

Nach dem Lesen der Bootsektoren kann die Deskriptortabelle aufgefunden und ausgewertet werden. Leere Deskriptoren werden zunächst zwischengespeichert, da diese später für den Einstieg für das Carving sein werden.

Wird ein Hauptvideodeskriptor gefunden wird die Clusterkette für alle Fragmente abgelaufen. Das Ergebnis ist eine Liste an Fragmenten bzw. Videofragmentdeskriptoren, welche einen Hauptvideodeskriptor zugeordnet werden können.

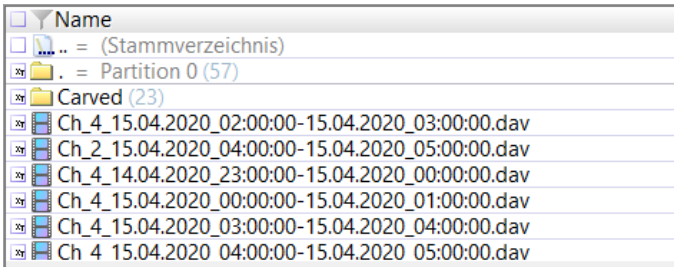

Abbildung 14. Dateinamen in X-Ways Forensics

Die nun aufgefundene Datei wird unmittelbar in den Verzeichnisbrowser von X-Ways Forensics dargestellt. Als Name der Datei wird eine Kombination aus Kanal, Anfangs- und Enddatum verwendet, um die Dateien zeitlich eingrenzen zu können, siehe Abbildung 14.

Zusätzlich werden der Datei Metadaten hinzugefügt, u. A. die Partition und die ID des Hauptvideodeskriptors, siehe Abbildung 15. Diese Metadaten sind notwendig, um das Befüllen der Datei zu ermöglichen, da die X-Tension nach dem Aufbau des Dateibaums aus dem Speicher entladen wird, die Metadaten jedoch im Verzeichnisbaum von X-Ways Forensics im aktiven Fall erhalten bleiben.

Bei den Logdateien oder gecarvten Videodateien wird in den Metadaten „Logfile“ oder „Carved“ hinzugefügt, um diese eindeutig zu unterscheiden. Dies ist notwendig, da die Daten der gecarvten und vorhandenen Videodateien sowie der Logdateien auf verschiedenen Wegen ausgelesen werden.

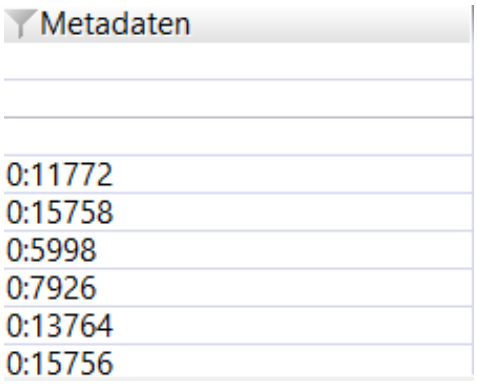

Abbildung 15. Metadaten in X-Ways Forensics

A. Carving von Videodateien

Da der Datenbereich des Dateisystems fest definiert ist wird das Carving auf diesen Bereich festlegt. Videodaten sind demnach nur in den Clustern ab dem Datenoffset aufzufinden. Es werden alle Sektoren freier Deskriptoren bzw. Cluster nach den Signaturen „DHAV“ sowie „dhav“ durchsucht. Wird bspw. die Signatur DHAV gefunden, wir zunächst geprüft ob die folgende Länge sowie der Zeitstempel gültig sind. Ist die Prüfung erfolgreich, wird überprüft ob die Länge des gesamten Videoframes über die Grenze des aktuellen Clusters hinaus geht.

Falls die Länge im Bereich aktuellen Clusters liegt, wird dieses Videoframe zu einer Liste hinzugefügt. Andernfalls wird dieses Videoframe als fragmentiert gekennzeichnet und in einer separaten Liste gespeichert. Sollte im Anschluss nun ein dhav, also ein Footer gefunden werden, wird die Liste der fragmentierten Videoframes rückwärts durchgesucht, bis ein passender Header gefunden wird. Falls kein Header gefunden wird kann der Footer als nicht zuordenbar ignoriert werden.

Durch das Suchen der Signaturen und spätere Zuordnung der können aus einzelnen Videoframes ganze Videofragmente erzeugt werden. Alle Videoframes werden einem Videofragment zugeordnet, welches eine Stunde, bspw. 11:00 Uhr bis 12:00 Uhr, eingrenzt. Die erzeugten Videofragmente werden anschließend in dem Verzeichnisbrowser von X-Ways Forensics im Ordner „Carved“ der jeweiligen Partition dargestellt.

B. Lesen der Daten

Da die Dateien, bis auf die Logdateien, in DHFS 4.1 fragmentiert abgelegt werden muss die X-Tension mit den API Funktionen der Disk I/O erweitert werden. In diesem Modus ist der Entwickler verantwortlich, den von X-Ways Forensics erwarteten Dateipuffer mit Daten zu befüllen.

Der von X-Ways Forensics zur Verfügung gestellte Puffer ist max. 8 MB groß. Die Funktion zum Lesen der Daten (XT_FileIO) wird solange von X-Ways Forensics aufgerufen bis keine Daten mehr vorhanden sind oder das Programm einen Fehler zurückgibt.

Die Datei wird somit sukzessive gelesen. Die Fragmentierung muss aufgelöst und die richtigen Dateioffsets zum Lesen ermittelt werden. Dabei muss sich das Programm stets den aktuellen Offset für jeden Funktionsaufruf merken, um an der richtigen Stelle weiter lesen zu können. Zudem muss sichergestellt werden, dass die Offsets für jede neue Datei zurückgesetzt werden. Für diesen Fall wurden zwei globale Variablen definiert, currentNItemID sowie moreFragmentsOffset, welche die zuvor genannten Punkte absichern.

VII. Zusammenfassung und Ausblick

Die hier vorgestellte Arbeit beschreibt das Dateisystem DHFS 4.1 von DAHUA. Durch die Dateisystemanalyse nach der Struktur von Carrier konnte das Dateisystem in seine einzelnen Bestandteile zerlegt und analysiert werden. Auf dieser Grundlage wurde die X-Tension für X-Ways Forensics entwickelt.

Es konnte gezeigt werden, dass das Dateisystem verschiedene Strukturen aus bekannten Dateisystemen wie NTFS und FAT verwendet, sowie dass die Dateien des Dateisystems teilweise fragmentiert gespeichert werden. Durch die Analyse der Deskriptortabelle kann die Fragmentierung umgekehrt werden, um die Videodateien lesen zu können.

Die hier vorgestellte X-Ways Forensics X-Tension dient als Grundlage um die Videodateien von dem DHFS 4.1 lesen und extrahieren zu können. Es sollte gezeigt werden, wie im Grundsatz ein Parser für das Dateisystem entwickelt und verwendet werden kann. Die Methode des Carvings kann ggf. durch in der Zukunft gewonnene Informationen über die Dateistruktur der DHAV-Dateien erweitert und verbessert werden.

Auf Grundlage der Dateisystembeschreibung können weitere neue Programme oder Erweiterungen bekannter Programme entwickelt werden. Einige dieser Programme können für die Strafverfolgung nützlich sein, da so ggf. Straftaten durch die Extraktion von teilweisen gelöschten Videodateien ermöglicht wird.

Literaturverzeichnis

[1] IBM. “What is digital forensics?” Besucht am 25. Okt. 2025. Adresse: https://www.ibm.com/think/topics/digital-forensics.

[2] B. Carrier, *File System Forensic Analysis*. Addison-Wesley, 2005.

[3] J. Han, D. Jeong und S. Lee, “Analysis of the HIKVISION DVR file system,” Bd. 157, Okt. 2015, S. 189–199, ISBN: 978-3-319-25511-8. DOI: 10.1007/978-3-319-25512-5_13.

[4] N.-A. Le-Khac, R. Gomm, M. Scanlon und T. Kechadi, “Analytical Approach to the Recovery of Data from CCTV File Systems,” Juli 2016. DOI: 10.13140/RG.2.2.31446.65601.

[5] W. Dongen, “Case study: Forensic analysis of a Samsung digital video recorder,” *Digital Investigation*, Jg. 5, S. 19–28, Sep. 2008. DOI: 10.1016/j.diin.2008.04.001.

[6] E. Dragonas, C. Lambrinoudakis und M. Kotsis, “IoT Forensics: Investigating the Mobile App of Dahua Technology,” Juli 2023, S. 452–457. DOI: 10.1109/CSR57506.2023.10224982.

[7] Magnet-Forensics. “Magnet Witness,” besucht am 25. Okt. 2025. Adresse: https://www.magnetforensics.com/de/products/magnet-witness/.

[8] DiskInternals. “DiskInternals DVR Recovery,” besucht am 25. Okt. 2025. Adresse: https://www.diskinternals.com/dvr-recovery/dhfs-4-1-file-system-dahua-nvr-cameras-data-recovery/.
[9] G. Batista. “dhfs_extractor,” besucht am 25. Okt. 2025. Adresse: https://github.com/gbatmobile/dhfs_extractor.
[10] DmytroMoisiuk. “DVR_Dahua,” besucht am 25. Okt. 2025. Adresse: https://github.com/DmytroMoisiuk/DVR_Dahua.
[11] Dahua-Technology. “Smartplayer,” besucht am 25. Okt. 2025. Adresse: https://dahuawiki.com/Software/Dahua_Toolbox/SmartPlayer.
[12] Opengroup. “Seconds Since the Epoch,” besucht am 25. Okt. 2025. Adresse: https://pubs.opengroup.org/onlinepubs/9699919799/xrat/V4_xbd_chap04.html#tag_21_04_16.