Academiejaar 2015-2016
3e examenperiode - augustus

EXFAT: DE NIEUWE FORENSISCHE UITDAGING

Eindwerk voorgedragen door Timothy De Groot

Interne promotor:

Peter Berghmans

Externe promotor:

Yves Vandermeer

Tot het bekomen van het diploma Hoger Onderwijs, één cyclus, volledig leerplan, Bachelor in het infomaticamanagement en de multimedia in het studiegebied Technologie & Design

Academiejaar 2015-2016
3e examenperiode - augustus

EXFAT: DE NIEUWE FORENSISCHE UITDAGING

Eindwerk voorgedragen door Timothy De Groot

Interne promotor:

Peter Berghmans

Externe promotor:

Yves Vandermeer

Tot het bekomen van het diploma Hoger Onderwijs, één cyclus, volledig leerplan, Bachelor in het infomaticamanagement en de multimedia in het studiegebied Technologie & Design

"Ik, Timothy De Groot, verklaar dat, voor zover ik er weet van heb, deze scriptie geen materiaal bevat dat ooit in eender welke instelling is gebruikt om een diploma, van welke aard ook, te behalen of dat eerder werd gepubliceerd of geschreven door een ander persoon, behalve daar waar deze scriptie referenties bevat naar andere werken."

INHOUDSOPGAVE

1 VOORWOORD

Mijn eerste contact met Yves Vandermeer, mijn externe promotor, was overweldigend. Yves Vandermeer werkt als rechercheur bij de dienst 'Federal Computer Crime Unit' van de Federale Gerechtelijke Politie van België, is betrokken bij Europol, leidt een internationaal organisatie die instaat voor het ontwikkelen van computercriminaliteitsgerichte cursussen, doctoreert en geeft les aan de Universiteit van Dublin. Al bij ons eerste contact maakte Yves Vandermeer mij attent op de problematiek met betrekking tot de geringe kennis rond het bestandssysteem exFAT.

Ik wens Yves Vandermeer dan ook te bedanken voor het mogelijk maken van mijn stage, voor de aanwakkering van mijn interesse in exFAT en voor de leidraad en bron van informatie die hij geweest is tijdens het schrijven van dit werk.

Ook dank ik Walter Coenraets, diensthoofd van de Federal Computer Crime Unit, dat ik deel uit mocht maken van zijn team. De stage en de collega's brachten mij veel bij over onderzoeksmethoden en computercriminaliteit in het algemeen.

Tot slot wens ik ook mijn interne promotor Peter Berghmans alsook docent Chris Vandermeiren te bedanken voor het in goede banen leiden van dit eindwerk. Hun advies en hulp hebben mij veel steun geboden bij de totstandkoming van de structuur, inhoud en de presentatie van het werk zoals het geschreven staat.

2 SAMENVATTING

Vanuit de dienst Federal Computer Crime Unit van de Belgische Federale Politie, blijkt de noodzaak voor een onderzoek naar exFAT. Forensische analyse dient immers toegepast te worden op systemen die gebruik maken van het relatief nieuwe bestandssysteem. Het doel van forensische analyse bestaat er niet alleen in om bestanden terug te halen en te analyseren maar ook om na te gaan welke handelingen een verdachte verricht heeft, als om de handelingen die een onderzoeker ondernomen heeft om bewijzen te verzamelen te noteren.

Dit werk zal aan de hand van eerder uitgevoerde onderzoeken naar de werking van exFAT, en aan de hand van vorige versies van FAT, de werking van het bestandssysteem tonen vanuit het oogpunt van een forensische analist. De gebruikelijke opbouw van een harde schijf met zijn partities, bestandsystemen en drivers zijn van belang en worden daarom kort beschreven. Elk bestandsysteem wordt als het ware op een afzonderlijke partitie op de harde schijf geplaatst. Een driver werkt samen met bestandssystemen om het resultaat van bepaalde handelingen te kunnen bekomen. De werking van een driver op één toestel kan een verschillend resultaat opleveren dan de driver op een ander toestel. Daarom worden aantal voorbeelden van dit fenomeen kort uitgelegd.

De werking van exFAT en FAT32 zijn niet volledig verschillend. Zo wordt er nog steeds gebruik gemaakt van ketening en entry's in de directory's. Opvallend is wel dat, door de toevoeging van een bitmap in exFAT, nu ook gefragmenteerde bestanden teruggehaald kunnen worden. Ook niet-gefragmenteerde kunnen, net zoals in FAT32, nog steeds op de oude manier hersteld worden dankzij de beschrijving in deze directories. Bestanden en bestandsnamen aanpassen en verplaatsen is in theorie niets anders dan het 'wissen' van de bestandsentry. De casus toont daarom ook twee voorbeelden van de recuperatie van 'gewiste' bestanden in exFAT, zowel gefragmenteerde als niet-gefragmenteerde herstelling wordt besproken.

Tot slot werden 4 programma's geanalyseerd. De programma's draaien niet op hetzelfde besturingssysteem en zijn ook niet altijd gratis. Een gebruiker kan daarom niet altijd kiezen welk programma hij zal gebruiken. De programma's vertonen vele gelijkenissen, het verschil is merkbaar bij het weergeven van details of bij de presentatie van gegevens in de gebruikersinterface. Momenteel zijn enkel Tyrhex en X-ways in staat om gefragmenteerde bestanden terug te halen, met gefragmenteerde bestanden heeft geen enkel van de besproken programma's moeilijkheden.

3 VERONDERSTELLINGEN

3.1 Symbolenlijst

ACPO: Association for Chief Police Officers);

Apple: Apple Inc. is een Amerikaans elektronicabedrijf;

Bootable: een besturingssysteem opstarten vanaf een gegevensdrager als bijvoorbeeld een cd-rom of usb-stick;

Chaining: ketening, chronologisch ordenen van dataclusters;

Cluster heap: start van de clusterzone;

Ddos: distributed denial-of-service-aanvallen, overbelasten server;

Dell: Dell Inc. is een Amerikaanse computerfabrikant;

Diskette: Een diskette of floppy (disk) is een gegevensdrager die in computers gebruikt wordt en in de jaren 60 ontwikkeld werd;

DOS: Disk Operating System, oftewel een besturingssysteem voor apparaten met schijfstations;

Driver: Een stuurprogramma is een specifiek stuk software dat een verbinding legt tussen hardware en besturingssysteem.

EiB: exbibyte;

EUROPOL: European Police Office;

exFAT: Extend FAT (bestandsysteem);

ext4: Fourth Extended File System (bestandssysteem van Linux);

FAT: File Allocation Table (geen bestandssysteem);

FCCU: Federal Computer Crime unit;

File allocation entry: ook wel clusterverloopentry of bestandslocatieëntry genoemd;

File Main entry: ook wel main entry, file entry, Bestand/Map main entry;

File name entry: ook wel bestandsnaamentry of bestandsentry genoemd;

Formattering: softwarematig indelen van een opslagmedium in clusters en sectoren;

GPT: GUID Partition Table;

HFS+: Hierarchical File System + (bestandssyteem Mac OS X, Apple);

HP: Hewlett-Packard is een van de grootste Amerikaanse technologiebedrijven;

IBM: International Business Machines Corporation;

ISO 9660: is een bestandssysteem voor cd-rom;

JFS: Journaled File System is een 64-bit logboekbestandsysteem ontwikkeld door IBM;

KiB: kibibyte;

LFS: Large file support;

Linux: is een open-source-, Unix-achtig besturingssysteem gebaseerd op de Linux-kernel;

Long file name: Een naamprincipe dat ervoor zorgt dat bestandsnamen en extensies uit meer karakters kunnen bestaan; Long bestaat om het verschil te tonen met gewone, oude DOS file names, gemaakt van 8 karakters voor de naam en 3 voor de extensie.

Mac OS X: is een lijn van besturingssystemen van Apple;

MBR: Master Boot Record;

MFT: Master File Table;

MiB: mebibyte;

NTFS: New Technology File System;

NTLDR: NT loader;

Offset: is een andere benaming voor de positie;

Partitie: het verdelen van die schijf in meerdere logische eenheden, die aangeduid worden als partities;

Phishing: is een vorm van internetfraude, vissen naar persoonlijke gegevens;

PiB: pebibyte

Root directory: ook wel de root of hoofdmap genoemd, is het hoogste niveau van de boomstructuur van een op een computer aanwezig bestandssysteem.

TiB: mebibyte;

Unallocated: Niet-toegewezen (cluster);

Unicode-karakters: is een internationale standaard voor de codering van grafische tekens en symbolen in binaire codes, vergelijkbaar met de ASCII-standaard;

Uppercasetabel: een tabel, gevuld met karakters;

YiB: yobibyte;

ZiB: zebibyte.

3.2 Algemene veronderstellingen

Winhex: is een licht programma, ontwikkeld door het bedrijf X-ways. Het programma toont de hexadecimale weergave van data, offsets en interpreteert ook Unicode-karakters. Het programma is handig voor forensische doeleinden en foutopsporing;

Drivers: de werking van de meeste drivers verschilt van elkaar. Vaak vertonen zij dezelfde resultaten maar in sommige gevallen is het resultaat totaal verschillend. Een voorbeeld hiervan is het wissen van een bestand op het bestandssysteem FAT32 door sommige fototoestellen. Zij zullen alle bytes op 0 plaatsen, in tegenstelling tot zo goed als alle andere drivers. Dit werk zal enkel de meest gebruikte manier van drivers bespreken;

Little endian: is een manier van ordenen van een eendimensioneel systeem van elementen die zelf een geordende rij van sub-elementen zijn, zoals een computergeheugen. Er zijn twee hoofdsoorten: big-endian en little-endian. Dit werk zal noteringen vertonen in little-endian;

Wissen: in dit werk zal zelden gesproken worden over het 'wissen' van een bestand. Er dient bij exFAT omzichtig omgesprongen te worden met deze term aangezien het, tot het tegendeel bewezen is, niet zeker is of een bestand verplaatst of gewist werd. Bovendien zullen deze bestanden zich ook nog steeds fysiek op de drager bevinden tot de clusters die de oude data bevatten door een nieuw bestand overschreven worden (zie verdere hoofdstukken). Er zal dan ook vaak verwezen worden naar de term 'niet-toegewezen';

MBR: het is mogelijk dat de start van het VBR opgezocht dient te worden in het MBR. In dit werk zal er echter enkel gewerkt worden met de geëxtraheerde volume van een usb-stick. Dit zal de interpretatie van voorbeelden en oefeningen zal vergemakkelijken.

3.3 Voorkennis

Om dit werk beter te begrijpen is een voorkennis vereist van bestands- en besturingssystemen. Dit werk beperkt zich voornamelijk tot het bestandssysteem exFAT;

Het is aangewezen op de hoogte te zijn van 'Little endian', aangezien een aantal voorbeelden in voorgemelde notering gepresenteerd zullen worden;

Kennis met betrekking tot de linux-commando's is aangewezen aangezien er in de casus bestanden teruggehaald zullen worden met behulp van linux-commando's en het programma 'the sleuthkit';

Om datums uit de entry's te kunnen omzetten is een voorkennis van de DOS-datumnotering vereist.

4 INLEIDING

Een stage bij de Federale Computer Crime unit leert dat er weinig kennis is over de werking van Extended FAT (afkorting exFAT), het nieuwe bestandssysteem van Microsoft. Maar wat is exFAT? Wat maakt het zo speciaal en hoe onderscheidt het zich van andere bestandssystemen? Wat maakt dat (er over) exFAT weinig gekend is en hoe komt het dat er over andere bestandssytemen wel uitgebreide informatie beschikbaar is? In welke mate wordt exFAT reeds ondersteunt door andere besturingssystemen dan Microsoft en hoe wordt er omgegaan met licenties? Waarom zouden gebruikers moeten overschakelen naar het bestandssysteem? En tot slot, hoe zal het gebruik in de toekomst hierdoor vermoedelijk evolueren?

In het kader van forensisch onderzoek levert de geringe informatie problemen op bij het verzamelen van bewijsmateriaal. Ook in rechte is het uitleggen van de werking van het bestandssysteem of het verantwoorden van bepaalde onderzoeksmethoden een vereiste. Beschikbare informatie zou het ontwikkelen van nieuwe software kunnen helpen. Hierdoor kunnen onderzoekers meer en betere resultaten boeken en krachtiger bewijsmateriaal verzamelen. Een belangrijk element is om te weten wat precies onderzocht moet worden om de werking van het bestandssysteem te begrijpen of om een bewijs van criminele feiten aan te kunnen tonen. Bovendien zou het onderzoeken van het bestandssysteem andere verrassende elementen naar boven kunnen brengen die cruciaal zouden kunnen zijn voor het verzamelen van bewijsmateriaal.

Dit eindwerk richt zich daarom vooral op de studie naar exFAT vanuit het oogpunt van een forensisch onderzoeker. Het doel is om een antwoord te bieden op de hierboven beschreven probleemstelling. Reverse engineering, in samenwerking met de Federale Computer Crime Unit, vorige FAT-versies, de geschiedenis van FAT en verschillende studies en andere bronnen, zullen maken dat dit eindwerk een beeld kan schetsen over de werking van exFAT en vooral, hoe deze werking te benutten om er voordeel uit te kunnen halen bij forensisch onderzoek. De studie moet ook moeten uitwijzen of verwijderde bestanden teruggehaald kunnen worden, zoals momenteel bij andere bestandssystemen mogelijk is.

Tot slot kan gesteld wordt dat dit werk zich onderscheidt van andere studies doordat het in de Nederlandse taal geschreven is, door het beschrijven van de opbouw van exFAT vanuit het oogpunt van een forensisch onderzoeker en door het analyseren en presenten van praktijkvoorbeelden, casussen en bestaande forensische programma’s.

5 INTRODUCTIE (EX)FAT

In 1977 kwam Microsoft met het bestandssysteem FAT. FAT staat voor File Allocation Table. Een File Allocation Table is een tabel, geplaatst op een bepaalde plaats op de harde schijf, die gebruikt wordt voor het bijhouden van de posities van data op de schijf. Er is een duidelijk verschil tussen een FAT-bestandssysteem en tussen de File Allocation Table. Deze begrippen zullen in dit werk vaak besproken worden en mogen niet met elkaar verward worden. Een FAT-bestandssysteem heeft, tot op heden, altijd een File Allocation Table, deze kan maar zal daarom niet altijd gebruikt worden.

FAT wordt tegenwoordig voornamelijk gebruikt op verwijderbare harde schijven, dit omdat het op lichte systemen een hoge performantie kan blijven voorzien. FAT heeft dit voordeel te danken aan het gebrek aan bijhouden van veel bijkomende informatie, dit in tegenstelling tot bestandssystemen als NTFS. Sinds het uitbrengen van FAT is er veel veranderd. FAT zelf werd meermaals verbeterd en kreeg hierbij ook meermaals andere namen. Volgende namen kunnen onderscheden worden; FAT12, FAT16, FAT32 en exFAT.

Momenteel is het meest gebruikte bestandssysteem op verwijderbare media FAT32 en dit heeft uiteraard zijn redenen. In dit hoofdstuk zal FAT, en vooral exFAT besproken worden aan de hand van diens geschiedenis. Deze aanpak is logisch, aangezien men van FAT tot exFAT gekomen is.

De scope van dit project zal zich niet uitbreiden tot de details van FAT. Hierover zijn reeds andere stukken geschreven. Dit werk beperkt zich enkel tot exFAT en de forensische kijk op dit bestandssysteem. exFAT behoort tot de FAT familie, hierdoor is het dan ook noodzakelijk dat ook FAT kort wordt weergegeven in dit werkstuk. Kennis over de werking van de vorige versies van FAT heeft het onderzoek naar exFAT kunnen vergemakkelijken aangezien bepaalde werkwijzen uit oudere FAT-versies in de nieuwe versie opnieuw gebruikt worden.

5.1 De geschiedenis van exFAT

De term FAT (File Allocation Table) verwijst naar de meest gebruikte FAT-systemen, met name FAT12, FAT16, FAT32 en sinds kort ook exFAT. Ook Microsoft heeft het in zijn dialoogvensters vaak over FAT en niet over een specifiek FAT-type als bijvoorbeeld FAT32. Dit kan soms verwarring zaaien wanneer geweten moet zijn over welk type FAT het exact gaat. Gedurende de loop der jaren werden ook uitbreidingen of subversies van FAT uitgebracht, dit om hieronder beschreven beperkingen tijdelijk op te lossen. Een voorbeeld hiervan is FAT16B. Deze versies zijn soms niet eens door Windows ontworpen en worden niet beschreven in dit werk.

Hieronder wordt de geschiedenis geïllustreerd van de 4 grootste FAT-types. Vooral de nadruk op de beperkingen van elk bestandstype heeft een grote invloed op het ontstaan van diens opvolger. Een weggewerkte beperking kan voor een forensisch analist in bepaalde gevallen erg interessant zijn. Hierover is meer te lezen in het verdere verloop van dit werkstuk.

5.2 FAT

In 1977 werd het concept FAT bedacht. Het doel van FAT was om op een snelle manier stukken op de schijf bij te kunnen houden en om te kijken door welke bestanden deze stukken gebruikt worden. De grootste beperkingen aan de eerste versie waren het erg kleine opslagvolume, de kleine bestandsnaam (maximum 9 karakters) en het gebrek aan het bestaan van submappen.

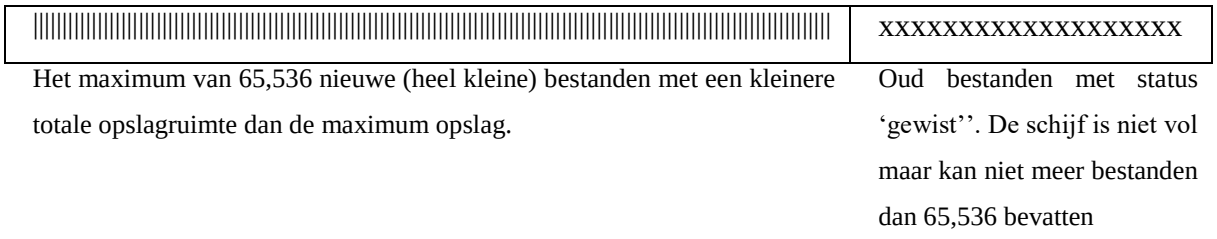
5.3 FAT12

Op een aantal beperkingen van FAT werd snel een antwoord geboden door Microsoft. FAT12 werd gelanceerd in 1980, slechts 3 jaar na het uitbrengen van de eerste versie van FAT. Het maximum van de volumegrootte werd uitgebreid naar 16 Mb in 4 Kb clusters en naar 32 Mb in 8 Kb clusters. Later zal ook dit een beperking worden bij het steeds groter worden van bestanden en opslagmedia. Deze beperking is er één die steeds zal terugkeren bij FAT.

FAT12 werd gebruikt op diskettes en in die periode ook op vaste harde schijven. Op dat moment bestaat het alternatief NTFS immers nog niet. NTFS zal verder in dit werk kort besproken worden.

Een van de meest uitgesproken beperkingen van FAT12 is de beperking op het aantal bestanden (ongeacht de grootte). Zoals hierboven besproken kunnen beperkingen handig zijn voor forensische analyse. Een voorbeeld hiervan is bovengenoemde beperking. Wanneer een bestand 'gewist' wordt, krijgt het door FAT een status. Deze status houdt in, dat visueel een deel van het bestand zich nog op de schijf bevindt, tot op het moment dat, de voor het oud bestand gebruikte ruimte, door een ander bestand wordt overschreven. Het voordeel van deze beperking voor forensische analyse is dat wanneer het aantal bestanden op een diskette zijn limiet bereikt, maar de maximum bestandsgrootte niet overschreden wordt, een deel van de 'gewiste' bestanden nog steeds fysiek aanwezig is op de schijf. De schijf lijkt immers vol te zijn, ook al is dit niet zo.

Harde schijf met bestanden



Deze illustratie toont een vereenvoudigde visuele weergave van een harde schijf met bestanden en met bestandstype FAT12. "|"stelt een bestand voor, "x" een gewist bestand.

5.4 FAT 16

FAT16 werd in 1984 geïntroduceerd. Het werd vooral gebruikt op harde schijven en USB-sticks tot 2 GB. Net zoals bij zijn voorgangers ligt de grootste beperking bij het maximale bestandsvolume, namelijk 2 GB bij 2 Kb clusters en tot 16 GB bij 256Kb clusters. Omdat harde schijven tegenwoordig veel groter zijn dan 2 GB, wordt FAT16 niet meer gebruikt, in tegenstelling tot FAT32, exFAT of NTFS.

In 1984 werd FAT16 echter gebruikt voor vaste harde schijven waardoor dit een erg belangrijke beperking werd bij het groter worden van bestanden en volumes. FAT16 had ook nog een aantal andere verbeteringen en beperkingen maar deze gaan buiten de scope van dit werkstuk.

5.5 FAT 32

FAT32 werd in 1996 uitgebracht. Opnieuw werden een groot aantal veranderingen aangebracht. Intern werden grote wijzigingen doorgevoerd waardoor FAT32 veel meer clusters kon bevatten. Concreet wil dit zeggen dat de bestanden die zich op het bestandstype bevinden veel groter mogen/kunnen zijn dan bij FAT16 omdat een groter opslagvolume gebruikt kan worden. Opnieuw zal deze maximum bestandsgrootte in de toekomst niet voldoende blijken. Nog een grote wijziging is dat de rootdirectory zich niet meer in de sectoren bevindt maar wel in de clusters. Hierdoor is een volle directory niet meer mogelijk. Verderop in dit werkstuk wordt meer informatie gegeven over clusters, sectoren. Een andere grote wijziging is tevens ook het invoeren van long file names. (Deze long file names kunnen na het uitbrengen ook op de nieuwere versies van FAT 16 gebruikt worden.)

FAT32 laat het toe om grote bestanden te behandelen. Toch blijkt ook de maximale bestandsgrootte van FAT32 niet voldoende. Bij de komst van windows XP werd daarom een volledig nieuw bestandstype ontwikkeld, NTFS genaamd. Dit bestandstype biedt veel voordelen voor vaste harde schijven waarop besturingssystemen draaien. FAT32 is echter nog steeds lichter, sneller en beter geschikt voor verwijderbare media. Hierdoor wordt de huidige tendens opgemerkt van NTFS op vaste harde schijven en FAT32 op verwijderbare media. NTFS en FAT32 zijn volledig compatibel met elkaar. Hierdoor worden zij vandaag nog steeds in boven beschreven combinatie gebruikt voor allerhande systemen en apparaten.

Aangezien FAT32 erg lang gebruikt werd, hebben veel fabrikanten er compatibiliteit voor voorzien. Onder andere Linux en APPLE's besturingssystemen kunnen FAT32 perfect interpreteren en gebruiken. FAT32 heeft echter 1 groot nadeel: het kan geen bestanden overbrengen die groter zijn dan
4GB (of 2 GB zonder LFS).

5.6 exFAT

In 2006, bij de lancering van vista service pack 1, werd exFAT geïntroduceerd. De bedoeling van exFAT is om zijn alom gebruikte voorganger, FAT32, te vervangen. exFAT biedt oplossingen voor tal van problemen bij zijn voorloper FAT32. Een erg belangrijke wijzigingen voor de scope van dit werkstuk is dat exFAT beschikt over een bitmap. Deze bitmap biedt een aantal mogelijkheden voor onderzoek. Het voordeel van het gebruik van een bitmap zal in het verdere verloop van dit werkstuk uitvoerig besproken worden.

Door zijn uitstekende prestaties en relatief weinig nadelen, was FAT32 heel lang erg populair. Omwille van de te kleine maximum bestandsgrootte moest echter een alternatief bedacht worden. Zo ontstond exFAT. De populariteit van FAT32 speelt echter in het nadeel van exFAT. FAT32 werkt perfect op APPLE's en LINUX besturingssystemen en wordt op zo goed als elke geheugenkaart voor camera's of andere apparaten gebruikt, dit is niet het geval voor exFAT en heeft te maken met patentering en het geheim houden van de werking van exFAT. Door deze geheimhouding hebben concurrenten geen inzage in de werking van het bestandssysteem en is het moeilijker om systemen en apparaten eropaf te stemmen. De patentering van allerlei functies van exFAT, belemmert concurrenten bovendien compatibiliteit te voorzien zonder één van de patenten te schenden. Hierdoor is het bijvoorbeeld mogelijk om exFAT te gebruiken op LINUX via een onofficiële omweg. Officieel is het gebruik van exFAT echter nog niet in LINUX opgenomen.

Toch mag men aannemen dat exFAT op lange termijn FAT32 zal vervangen. Meer en meer fabrikanten voorzien standaard al exFAT op grote verwisselbare media. De toekomst zal uitwijzen hoe concurrenten algemeen zullen omgaan met de patenten op exFAT.

6 FORENSISCHE ANALYSE

6.1 Introductie

Om grondig onderzoek te kunnen verrichten dient geweten te zijn welke de noodzaak is van een betrokken domein. In dit werk wordt dan ook kort geschetst waarmee rekening gehouden dient te worden bij het bespreken van exFAT wanneer het bestandssysteem vanuit het oogpunt van een Forensische onderzoeker bekeken wordt. Volgend hoofdstuk beschrijft de vragen die gesteld moeten worden wanneer bewijsmateriaal verzameld wordt voor een forensisch onderzoek.

Forensische analyse richt zich op het vinden van juridisch bewijsmateriaal. Er zijn vele soorten forensisch onderzoek zoals bijvoorbeeld bloedspatpatroononderzoek, drugsanalyse, forensisch duiken, enz. Dit werk zal zich beperken tot de forensische informatica analyse of anders genaamd: 'IT forensics'. Wanneer er in dit werkstuk over forensische analyse wordt geschreven, wordt gerefereerd naar de forensische informatica analyse. Het doel van forensische informatica analyse is het verklaren van de status van een digitaal voorwerp. Dit kan bijvoorbeeld een compleet besturingssysteem zijn maar even goed een usb-stick. Foto's, e-mails, en andere elementen kunnen verzameld worden als bewijsmateriaal dat in rechte gebruikt kan worden. In België zijn de forensische IT-specialisten te vinden bij de (Federale) Computer Crime Unit van de Politie.

6.2 De noodzaak van forensische analyse

Wetshandhavers komen vaak in contact met georganiseerde en gespecialiseerde vormen van criminaliteit. Omdat verder en diepgaand onderzoek kennis en ervaring vereist, is de politieman op straat vaak niet op de hoogte of niet bevoegd voor het uitvoeren van bepaalde onderzoeksdaden. Sommige vormen van criminaliteit zijn zo geavanceerd dat zij bestreden moeten worden door experts in het desbetreffende vakgebied. Om binnen de scope van dit werk te blijven, wordt als voorbeeld cybercriminaliteit genomen. De vormen en modus operandi veranderen voortdurend. Hierdoor zou men zich continu moeten bijscholen en verder specialiseren om op een correcte manier te kunnen ageren.

Er kan echter niet van elke politieman verwacht worden dat hij of zij op de hoogte is van bijvoorbeeld de exacte werking van exFAT, DDOS-aanvallen, phishing, enz. Hierdoor wordt er gebruik gemaakt van gespecialiseerde diensten. Van hen kan er dan wel verwacht worden dat zij zich toeleggen op deze uitgebreide en continu veranderende materie. Dit leidt tot een beter onderzoek, met betere resultaten en een hogere vattingskans. Betere resultaten kunnen dan weer gevolgen hebben op de handelingen van criminelen. In het beste geval worden ze afgeschrikt omdat de kans om gepakt te worden groter is en de straf in die mate krachtig, dat het winstpotentieel niet opweegt tegen het risico. Waargebeurd; onderzoekers dienden 2 weken aan de reconstructie van een bestand te werken dat uiteindelijk belangrijk bewijsmateriaal bevatte.

Tot slot moeten deze gespecialiseerde eenheden hun bevindingen kunnen staven in een rechtszaak. Derden moeten hun onderzoek kunnen begrijpen om de verdediging van de verdachte te kunnen verzekeren. Hierdoor moeten deze eenheden de werking van bepaalde systemen als exFAT, kunnen uitleggen. Dit om aan te tonen waarom zij sommige onderzoeksdaden verricht hebben of om bewijzen hard te kunnen maken.

6.3 Basisprincipes van de forensische analyse

Het ACPO (Association for Chief Police Officers) beschrijft in het document “ACPO Good Practice Guide ACPO Good Practice Guide for Digital Evidence” (2012) de vier belangrijkste principes met betrekking tot forensische analyse als volgt:

1. Data kan enkel betrouwbaar en rechtsgeldig zijn in rechte, wanneer deze niet door de onderzoekers is gewijzigd.
2. Indien een onderzoeker het toch nodig acht om de originele data aan te passen, moet deze onderzoeker competent zijn en dient hij uitleg te kunnen geven over de relevantie en de gevolgen van zijn acties.
3. Er moet een logboek bestaan van alle uitgevoerde processen die werden toegepast op het digitaal bewijsmateriaal. Een derde partij moet in staat kunnen zijn deze processen te onderzoeken. Het resultaat van deze onderzoeken zou hetzelfde moeten zijn als dat van de onderzoekers.
4. De verantwoordelijke voor het onderzoek heeft de totale verantwoordelijkheid met betrekking tot het verzekeren van het toepassen van bovengenoemde principes.

6.4 Forensische analyse van bestanden

6.4.1 Vaststellingen

Eén van de belangrijkste mogelijkheden die een onderzoeker heeft, is het analyseren van bestanden en de dragers ervan. Ideaal zou een onderzoeker alle antwoorden op volgende vragen kunnen vinden:

6.4.1.1 Met welke driver, werd een uitvoering verricht?

De driver kan namelijk een indicatie geven over het toestel waarmee gewerkt werd. Een voorbeeld zou zijn dat een verdachte ontkent dat een verdacht bestand door hem wordt aangemaakt. Zoals reeds in dit werk beschreven, bewerkt een driver op een cameratoestel bestanden niet zoals de driver van een pc, een smarphone of een wagen ingebouwde GPS. Een onderzoeker zou dus op basis van deze kennis kunnen opmaken of het bestand door het cameratoestel van de verdachte werd aangemaakt of niet.

6.4.1.2 Wanneer werd een uitvoering verricht?

Op basis van de metadata in bestanden kan nagegaan worden wanneer bepaalde verrichtingen plaatsvonden. Op basis van deze informatie kan men alibi's checken en de geloofwaardigheid van verdachten checken.

6.4.1.3 Door wie werd de uitvoering verricht?

Erg belangrijk is om te weten wie welke uitvoering aan een bestand volbracht. Voor bepaalde gerechtelijke feiten als kinderpornografie, weegt het maken van foto's in een rechtszaak zwaarder door dan het delen. Het is daarom van alle belang om de juiste verdachte te kunnen identificeren op basis van de beschikbare sporen.

6.4.1.4 Anomalieën in de bestandsgrootte?

Is een bestand groter nadat het van de computer komt van één van de verdachten? Indien dit het geval is, kan bewezen worden dat elementen werden toegevoegd of bijgewerkt (bijvoorbeeld pornografische foto werd toegevoegd, verdachte werkt actief mee aan een pornografisch netwerk). Bestand is kleiner: er werden elementen/bewijzen gewist of het bestand werd gecomprimeerd opgeslagen.

6.4.1.5 De bovengenoemde elementen kunnen toegepast te worden voor en/of na volgende verrichtingen:

- **Aanmaak**: om beweringen van verdachte te kunnen controleren en om te kunnen checken wanneer verdachte bijvoorbeeld begonnen is met een bepaald feit.

- **Wissen:** erg belangrijk om te weten. Dit kan aantonen dat een verdachte sporen heeft proberen wissen.

- **Aanpassen:** Om betrokkenheid van verdachte aan te kunnen tonen of om aan te tonen dat informatie veranderd (sporen wissen)/vervalst werd.

- **Verplaatsen:** Om aan te tonen dat elementen gedeeld werden, wanneer ze gedeeld werden. Om aan te tonen dat informatie verplaatst werd, wanneer deze verplaatst werd. Beiden kunnen interessant zijn om bepaalde elementen te verduidelijken. Een historiek kan bijvoorbeeld ook meer duidelijkheid scheppen.

6.4.2 Interpretatie

De vaststellingen van bepaalde feiten is één element. Een andere element is het interpreteren van de vaststellingen. Enerzijds moet men de ruwe data kunnen verzamelen en deze kunnen linken aan de werking van een systeem. Daarom is het zo belangrijk voor onderzoekers om over correcte informatie te beschikken met betrekking tot de werking van bepaalde systemen.

Anderzijds is het dan weer aan de onderzoeker om de informatie correct te interpreteren. Dit kan door beweringen van een verdachte te vergelijken met de gevonden sporen. Omdat er zowel fouten kunnen sluipen in het achterhalen van de sporen als in de interpretatie ervan, is het belangrijk om de gehanteerde methoden bij te houden. Hierdoor kunnen later problemen vermeden worden en moet een verkeerde interpretatie van sporen niet leiden tot een nietigverklaring van de vaststellingen.

7 BESTANDSSYSTEMEN EN STUURPROGRAMMA'S

7.1 Introductie

In de vorige hoofdstukken van dit werkstuk werd de term bestandssysteem aangehaald. FAT is één van de beschikbare bestandssystemen. Dit hoofdstuk zal kort schetsen waar FAT zich mag situeren ten opzichte van andere bestandssystemen. Erg belangrijk is tevens ook te weten welk bestandssysteem voor welk besturingssysteem ontwikkeld is en welke drivers beschikbaar zijn. De samenwerking van een driver met een besturingssysteem is van groot belang, en dit voor iedereen die gebruik maakt van een computer alsook voor een forensische onderzoeker. Het is namelijk zo dat een bestandssysteem op het een bepaald apparaat met een bepaalde driver correct kan werken. Een andere toestel met het hetzelfde bestandssysteem maar met een andere driver kan echter ook volledig perfect werken, toch is de wijze waarop de driver informatie behandeld op apparaat één volledig verschillend van de werking van de driver op het tweede toestel. Het begrijpen van de werking van deze drivers kan ervoor zorgen dat forensisch onderzoekers kan in sommige gevallen belangrijke informatie opleveren.

7.2 Bestandssysteem

Een bestandssysteem is de onderliggende structuur die door een besturingssysteem wordt gebruikt om gegevens op een vaste schijf te ordenen. Het besturingssysteem gebruikt deze indeling om data in de vorm van bestanden weg te schrijven of te lezen. Er zijn meer dan honderden verschillende bestandssystemen.

De meest gebruikte bestandssystemen zijn FAT32, NTFS, ext4 voor LINUX en HFS+ voor Mac OS X Voor de komst van verwisselbare media als USB-sticks, was ook het bestandssysteem ISO 9660 veel gebruikt, niet onlogisch aangezien dit het bestandssysteem is voor CD-ROMS en DVD'S.

7.3 Werking van een bestandssysteem

7.3.1 Werking op het niveau van het besturingssysteem en de hardware

Een bestandssysteem verzorgt de vertaling van de fysieke locaties op een opslagmedium, zoals bijvoorbeeld de sectoren, en toont deze in een aaneengesloten verzamelingen van data. Het bepaalt met andere woorden welke datablokken samen horen en toont deze vervolgens als één geheel.

7.3.2 Werking op niveau van het besturingssysteem en de applicaties

Op dit niveau kan eveneens een verzameling van datablokken worden voorzien door het bestandssysteem. Deze datablokken kunnen bestaan uit mappen, bestanden, enz. De ordening van deze data is voorzien door mappen en submappen. Ook interne verwijzingen voor gebruik van besturingssystemen worden op deze lage voorzien.

7.4 Systeemfuncties

Een bestandssysteem is, in de loop der tijd, uitgegroeid tot één van de basisonderdelen van het besturingssysteem en computergebruik. Het is dan ook niet verbazend dat interactie met het bestandssysteem een centraal onderdeel is van vrijwel iedere applicatie. Daartoe biedt het besturingssysteem een aantal standaardoperaties aan op het bestandssysteem in de vorm van systeemfuncties.

Een systeemfunctie kan grafisch dus gesitueerd tussen het bestandssysteem en de driver. In principe is er slechts 1 driver per bestandssysteem of is er zelf maar 1 driver voor meerdere bestandsystemen.

Normaal gesproken behoren de volgende functies tot die verzameling:

- Aanmaken van een bestand, al dan niet met een bepaalde grootte;
- Uitbreiden of verkleinen van een bestand;
- Hernoemen van een bestand;
- Kopiëren van een bestand
  (op hetzelfde of op een ander volume, gebruikende zelfde of andere bestandssysteem);
- Aanmaken van een nieuwe directory;
- Hernoemen en verplaatsen van een directory;
- Kopiëren van een directory;
- Verplaatsen van een bestand over de directory structuur;
- Verwijderen van data en vrijgeven van de ervoor gereserveerde ruimte;
- Verwijderen van een directory, al dan niet met inhoud.

Dankzij deze systeemfuncties kunnen verschillende bestandssystemen met elkaar communiceren. Bijvoorbeeld; Een bestand dat exFAT werd aangemaakt kan uitgelezen worden door NTFS.

Omdat het terughalen van bestanden niet als systeemfunctie, en ook niet door de drivers voorzien wordt, kunnen bestanden in praktijk niet met een druk op een knop teruggehaald worden. Zoals later in dit werk aangetoond zal worden, is het terughalen van sommige bestanden echter perfect mogelijk en zou er in theorie ook hiervoor een driver geschreven kunnen worden. Deze techniek wordt onder andere toegepast door forensische programma's.

7.5 Logging/Journaling

Een andere belangrijk item in het gebruik van bestandssystemen is het gebruik van logboeken. De meeste besturingssystemen maken gebruik van gereedschappen om de integriteit van data te waarborgen. Het gebruik van de logboeken kan deze integriteitscontroles optimaliseren en versnellen. In het geval van een forensische analyse zou een bestaand logboek vanzelfsprekend een goudmijn aan informatie kunnen bevatten. ExFAT maakt echter geen gebruik van logging, hierdoor kan het niet benut worden voor forensische analyse.

7.6 Vergelijking meest gebruikte bestandssystemen

LEGENDE (geeft enkel de hieronder getoonde waarden weer)		
Korte weergave	**Naam**	**Waarde**
KiB	kibibyte	$1024^1 = 2^{10}$
MiB	mebibyte	$1024^2 = 2^{20}$
GiB	gibibyte	$1024^3 = 2^{30}$
TiB	tebibyte	$1024^4 = 2^{40}$
PiB	pebibyte	$1024^5 = 2^{50}$
EiB	exbibyte	$1024^6 = 2^{60}$
ZiB	zebibyte	$1024^7 = 2^{70}$
YiB	yobibyte	$1024^8 = 2^{80}$

Deze tabel geeft een legende weer van het aantal bytes, hun afkorting en benaming.

Bestandsysteem **Ontwikkelaar**	**NTFS** **WINDOWS**	**exFAT** **WINDOWS**	**FAT32** **WINDOWS**	**JFS** **APPLE**	**HFS+** **APPLE**	**EXT4** **LINUX**
Max. bestandsgrootte	2 TiB	64 ZiB	4 GiB	4 PiB	8 EiB	16 TiB
Max. schijfgrootte	256 TiB	64 ZiB	2 TiB	32 PiB	16 EiB	1 EiB
Ondersteuning besturingssysteem	Windows Mac OS X (enkel lezen) Linux	Windows Mac OS X (10.6.5 en later)	Windows Mac OS X Linux	Mac OS X Linux	Windows (op Intel-Mac) Mac OS Mac OS X	Windows (zonder journaling) Mac OS X (enkel lezen) Linux

Deze tabel geeft de verschillen weer tussen de meest gebruikte bestandssystemen.

Uit bovenstaande tabel kan worden opgemaakt dat exFAT zowel op het gebied van maximale bestandsgrootte als op het gebied van maximale schijfgrootte veel beter presteert. Waar het echter minder goed voor scoort is de compatibiliteit met andere besturingssystemen. Andermaal is te merken dat FAT32 hier nog steeds het voordeel heeft dat het ondersteuning wordt voorzien door zowel Max OS X als Linux. Opvallend is ook dat NTFS minder goed presteert (niveau van snelheid en maximale bestandsgrootte) dan alle andere bestandssystemen, uitgezonderd de voorloper van exFAT, FAT32. De details hieromtrent vallen echter buiten de scope van dit stuk.

7.7 Stuurprogramma's en samenwerking met bestandssystemen

Een stuurprogramma is een stuk software dat een verbinding voorziet tussen het besturingssysteem en de hardware van een toestel. De Engelstalige benaming voor een besturingsprogramma is 'driver'. Een stuurprogramma zorgt ervoor dat een applicatie geen rekening moet houden met de hardware waarop het zal draaien. Het doel van het stuurprogramma is om een applicatie het besturingssysteem op een uniforme manier te laten aanspreken. Een voorbeeld zou zijn: het commando "doe iets" op het toestel met hardware X een uitvoering te laten verrichten. Wanneer de applicatie een uitvoering moet verwezenlijken op een toestel met hardware Y, kan het dankzij het stuurprogramma met een universeel commando dezelfde uitvoering laten bewerkstelligen.

Dit houdt echter wel in dat vele toestellen andere stuurprogramma's hebben. Het resultaat van de werking tussen hardware, bestandssysteem en een stuurprogramma zou in theorie hetzelfde moeten zijn. Door verschillende fabrikanten, programmeurs en andere bijkomende factoren dienen aanpakken te verschillen. Door andere werkwijzen zal in praktijk dan ook te zien zijn dat bijvoorbeeld een bestand wissen op een usb-stick met bestandssysteem FAT32 op een volledig andere manier zal gebeuren vanop een Windows computer dan vanop een digitale camera. Het doel van de drivers blijft echter wel om zo weinig mogelijk bewerkingen te verrichten om het gewenste resultaat te bekomen. Hierdoor zal de werking van het bestandssysteem versnellen en zal de gegevensdrager minder belast worden.

8 EXFAT: HET ONDERZOEK

Dit hoofdstuk beschrijft de werking van exFAT op basis van eerder uitgevoerde studies, zoals beschreven in “The Extend FAT file system: Differentiating with FAT32 file system” (2011), reverse engineering met behulp van de Federale Computer Crime unit, en de gekende informatie met betrekking tot de voorganger van exFAT.
De opbouw en werking worden beschreven vanuit het oogpunt van een forensisch onderzoeker. Een grote nadruk zal daarom ook gelegd worden op hoe een bepaalde werking als voordeel kan worden benut bij een forensisch onderzoek. Zoals reeds opgenomen in het hoofdstuk over forensische analyse kunnen logische elementen vaak enorm veel bewijsmateriaal bevatten. Sommige elementen kunnen dan ook door de verdachte worden aangepast of gewist, er kan zelfs sprake zijn van het gebruik van counter-forensics-programma’s. Het doel van de forensische analyse bestaat erin om het bewijs van het kwaad opzet te verzamelen of om de oorspronkelijke gegevens terug te halen. Uit dit hoofdstuk zal blijken dat bepaalde elementen, omwille van de technische werking van exFAT, niet gewijzigd kunnen worden. Wanneer belangrijke elementen bewust of onbewust gewijzigd of gewist werd, kunnen deze technische elementen in het voordeel van de onderzoeker benut worden.

8.1 Structuur van een harde schijf

8.1.1 Sectoren & clusters

8.1.1.1 Introductie

Het begrijpen van sectoren en clusters is van primordiaal belang bij de interpretatie van bepaalde elementen van een bestandssysteem. Niet alleen ketening (zie verdere hoofdstukken), maar ook het terugvinden van partities, directories, enz. vereisen de onder beschreven kennis.

Gegevens worden in bestanden bewaard. Bestanden worden beheerd door een bestandssysteem die de gegevens een plaats geeft op de gegevensdrager (bijvoorbeeld een externe harde schijf). Een harde schijf kan meerdere bestandssystemen bevatten, hetzij met dezelfde bestandssystemen, hetzij met verschillende bestandssystemen.

Een derde mogelijkheid is met zelfde bestandssystemen maar met andere instellingen. Elk bestandssysteem wordt op een afzonderlijke partitie bewaard. Op het laagste niveau van de schijf worden sectoren met een vaste grootte onderscheiden. Partities en hun bestandsystemen kunnen gebruik maken van gegroepeerde sectoren, clusters genaamd.

De gegevens van een bestand worden in clusters bewaard op het niveau van het bestandssysteem. Op het niveau van de harde schijf bestaan deze zelfde clusters uit groepen van sectoren.

8.1.1.2 Slack space

Wanneer we gegevens van een bestand opslaan in clusters, is de kans groot dat we te maken krijgen met slack space in de laatste cluster. Deze term wordt gebruikt om de restbytes te beschrijven van een cluster die niet door data in beslag genomen worden. Wanneer we bijvoorbeeld een cluster hebben die bestaat uit 2048 bytes (4 sectoren van 512 bytes) en een bestand hebben dat 1224 bytes groot is, dan zullen de resterende 824 bytes in de laatste sectoren niet gebruikt worden voor het nieuwe bestand. Zij zijn als het ware, verloren ruimte, slack space genaamd. Het daaropvolgende bestand wordt immers weggeschreven in de eerstvolgende vrije cluster, niet in de reeds gebruikte cluster.
Dit fenomeen heeft echter een groot forensisch voordeel. Indien we namelijk beschikken over een cluster die de status ‘niet-toegewezen’ heeft (en dus nog oude data bevat), dan kan deze cluster door een nieuw bestand overschreven worden. Het deel van de cluster dat door het nieuw bestand overschreven wordt kan niet meer gerecupereerd worden. De slack space kan daarentegen wel teruggehaald worden!

Het zijn de instellingen bij een bestandssysteem die zullen aangeven hoeveel sectoren een cluster zal bevatten, hierdoor kan het zijn dat verschillende partities andere clustergroottes hebben. Hoe meer sectoren een cluster bevat, hoe groter de kans dat er meer informatie op de slack space teruggevonden kan worden.

Voorbeeld cluster x met 4 sectoren				
Sector 1 (in bytes)	**Sector 2** (in bytes)	**Sector 3** (in bytes)		**Sector 4** (in bytes)
512	512	200	~~312~~	~~512~~

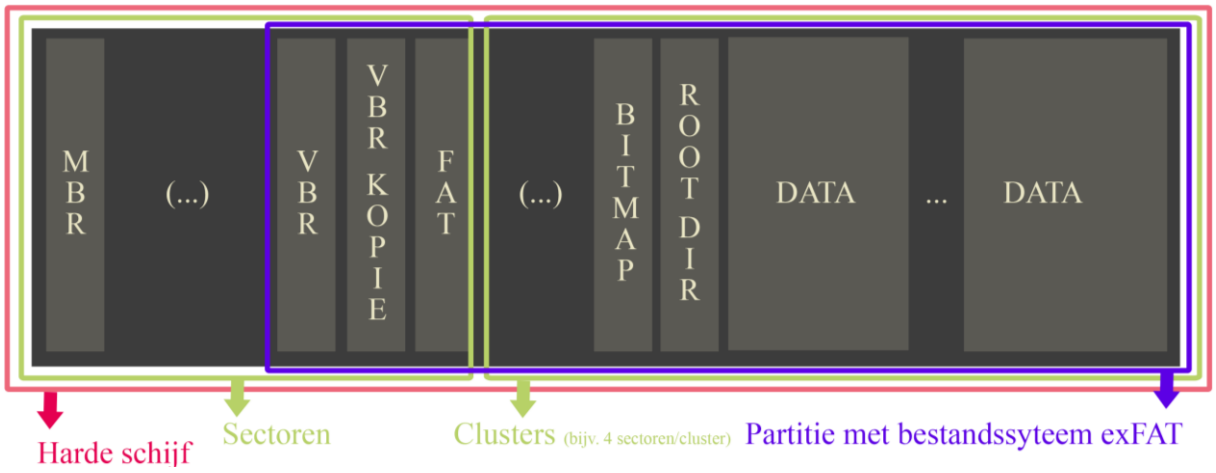
Nieuw bestand

Stuk van oud bestand in de slack space

Cluster bestaande uit 4 sectoren met elk 512 bytes. Sector 3 en 4 bevatten een deel van het oude, 'gewiste' bestand dat zich nog in de slack space bevindt.

8.1.2 Opbouw van een harde schijf

Om de besproken onderwerpen in de komende hoofdstukken beter te begrijpen, wordt in dit hoofdstuk in het kort een overzicht gegeven van een mogelijk voorbeeld van een harde schijf met één partitie en het besturingssysteem exFAT. Een aantal elementen zijn noodzakelijk voor de goede werking van een computer. Zo moet een harde schijf bijvoorbeeld altijd starten met een Master Boot Record, in onderstaand voorbeeld is dit dus ook het geval. Dit record bevat onder andere gegevens over een partitie op de harde schijf. Een harde schijf kan uiteraard meerdere partities bevatten, daarom is dit record ook zo belangrijk. Elke partitie start dan weer met een Volume Boot Record. Dit record bevat informatie over de partitie en vertelt, onder andere, informatie over clustergrootte en het gebruikte bestandssysteem op de partitie, in het gegeven voorbeeld dus exFAT. De MBR, VBR, VBR Kopie en FAT zijn bij exFAT opgedeeld in sectoren (niet in clusters). In de datazone, opgedeeld in clusters, bevinden zich de bitmap, de root directory en uiteraard ook de data.



Deze afbeelding illustreert de grafische weergave van een harde schijf met 1 partitie en het bestandssysteem exFAT.

8.1.3 Partities op extern opslagapparaat

Tot slot nog een interessant gegeven met betrekking tot verschillende partities op een extern opslagapparaat. Het is technisch mogelijk om meerdere partities te maken op een extern opslagmedia. Omwille van technische redenen leest het besturingssysteem Windows enkel de eerste partitie op het medium. Hierdoor zijn de andere partities inclusief inhoud niet zichtbaar. Een forensisch analist dient dus te allen tijde waakzaam te zijn voor dit fenomeen. Belangrijke gegevens kunnen immers ‘verborgen’ worden op deze ‘niet-zichtbare’ partitie.

Er zijn meerdere manieren mogelijk om de verborgen partitie toch te bekijken:

- Gebruik een ander besturingssysteem dan Windows zoals OS X of Linux;
- Maak gebruik van de software EaseUS Home Partition Master en RMPrepUSB;
- Maak gebruik van de software BootIce;
- Andere mogelijkheden.

8.1.4 MBR & GPT

Eén van de eerste onderdelen van het opstartproces is het uitlezen van de eerste sector op het toestel dat de bootable partitie bevat. De computer zal in de eerste sector op zoek gaan naar de Master Boot Record (MBR). Het MBR bevat belangrijke instructies, die nodig zijn voor de verdere afhandeling van het opstarten van de computer. Het MBR zal doorverwijzen naar de juiste opstartpartitie met daarop het besturingssysteem.

Standaard kan een MBR maximum 4 partities bevatten (tenzij gebruik gemaakt wordt van extended MBR). Daarom ondersteunen sinds 2010 bijna alle besturingssystemen GPT (GUID Partition Table). De opvolger van MBR doet hetzelfde als zijn voorganger maar kent een aantal voordelen. Eén van deze voordelen is dat het 128 partities kan bevatten in plaats van 4. Verder kunnen met GPT 128 bytes gebruikt worden om de partitie te beschrijven in plaats van 16 bytes.

Een forensisch onderzoeker kan uit het MBR of GPT informatie vinden over het aantal partities op de harde schijf en over de plaats waar deze partities zich op de schijf bevinden. Het is namelijk zo dat partitie A niet noodzakelijk naast partitie B geplaatst hoeft te worden op de harde schijf. Belangrijk om weten is dat een master boot record uit 512 bytes bestaat en afgesloten wordt met de handtekening “0xAA55”. Het ontbreken van deze handtekening betekent een corruptie van de harde schijf.

Het analyseren van het MBR is reeds uitvoerig in andere werken beschreven. Voor dit werk is de analyse van de Master Boot Record enkel relevant voor het terugvinden van de eerste cluster van de partitie die onderzocht dient te worden. Meer informatie over het MBR staat beschreven in “Analyzing Master Boot Record for Forensic Investigations” (2016). Het is mogelijk dat de start van het VBR opgezocht dient te worden in het MBR. In dit werk zal er echter enkel gewerkt worden met de geëxtraheerde partitie, afkomstig van de harde schijf. Hierdoor dient er geen rekening gehouden te worden met het MBR. Wat de interpretatie van voorbeelden en oefeningen zal vergemakkelijken.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	FE
000001C0	FF	FF	0B	FE	FF	FF	02	00	00	00	40	98	00	00	00	FE
000001D0	FF	FF	07	FE	FF	FF	43	98	00	00	98	47	0F	00	00	FE
000001E0	FF	FF	0B	FE	FF	FF	DC	DF	0F	00	50	4C	00	00	00	FE
000001F0	FF	FF	05	FE	FF	FF	2C	2C	10	00	D4	93	0F	00	55	AA

De aangeduide selectie op deze afbeelding illustreert een MBR in hexadecimale weergave in winhex.

Meer informatie over de analyse van GPT is terug te vinden in “Forensic analysis of GPT disks and GUID partition tables.*Digital Investigation”* (2009).

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
000003C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000003D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000003E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000003F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000400	A2	A0	D0	EB	E5	B9	33	44	87	C0	68	B6	B7	26	99	C7	¢ ÐëåI3D‡Àh¶·&™Ç
00000410	2F	2A	1D	66	51	FC	F7	40	9F	EB	63	D6	3B	FA	E3	93	/* fQü÷@ŸëcÖ;úã“
00000420	00	08	00	00	00	00	00	00	FF	47	01	00	00	00	00	00	ÿG
00000430	00	00	00	00	00	00	00	00	54	00	65	00	73	00	74	00	T e s t
00000440	31	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	1
00000450	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000460	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000470	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000480	A2	A0	D0	EB	E5	B9	33	44	87	C0	68	B6	B7	26	99	C7	¢ ÐëåI3D‡Àh¶·&™Ç
00000490	8E	4E	28	D2	4C	F4	62	4C	BC	C7	CF	38	40	E6	F8	5F	ŽN(ÒLôbL¼ÇÏ8@æø_
000004A0	00	48	01	00	00	00	00	00	FF	BF	03	00	00	00	00	00	H ÿ¿
000004B0	00	00	00	00	00	00	00	00	74	00	65	00	73	00	74	00	t e s t
000004C0	32	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	2
000004D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

De aangeduide selectie op deze afbeelding illustreert een deel van de GPT in hexadecimale weergave in winhex.

8.1.5 VBR

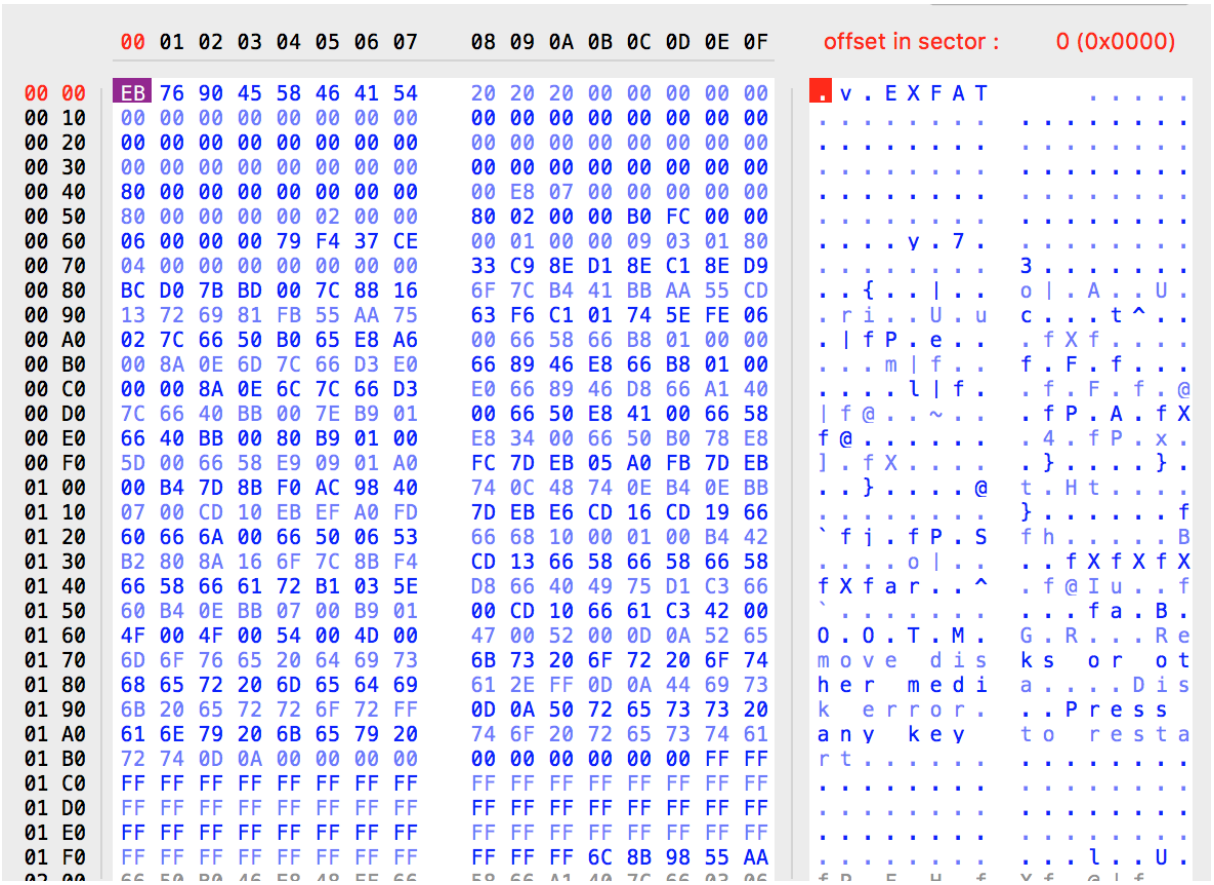
VBR staat voor Volume Boot Record en de start ervan is te vinden in de eerste sector (sector 0) van een partitie of van een externe harde schijf (al dan niet met partities). Het VBR bevat belangrijke informatie zoals de sectorgrootte, het serienummer van het volume, het bestandssysteem van de partitie, waar het MFT en zijn kopie zich bevindt (enkel relevant bij NTFS) en tot slot de locatie van de NTLDR (NTLoader).

VBR wordt vaak ook het boot block, het volume boot record, de volume boot sector, het partition boot record of de partition boot sector genoemd.

De analyse van het VBR beperkt zich in dit werk tot het terugvinden van volgende elementen:

Offset (hexadecimaal)	Bytes	Beschrijving
0x50	4	Locatie van de FAT (uitgedrukt in sectoren!)
0x54	4	Grootte van de FAT (uitgedrukt in sectoren!)
0x58	4	Start van datazone (cluster heap)
0x60	4	Eerste cluster van root directory
0x6D	1	Aantal bytes per sector (uitgedrukt in macht van 2)
0x6D	1	Aantal sectoren per cluster (uitgedrukt in macht van 2)

Deze tabel beschrijft een aantal waarden die teruggevonden kunnen worden op een bepaalde positie in het VBR.



Deze afbeelding toont een voorbeeld van een deel van het VBR, getoond in hexadecimale weergave met winhex.

Meer informatie over het Virtual Boot Record is terug te vinden in het werk “Plug and Play BIOS Specification 1.0A” (1994).

8.1.6 FAT

FAT staat voor File Allocation Table. In het verleden hield deze tabel zowel de ketening (chaining) als de toewijzing van de clusters van bestanden bij. ExFAT werkt echter enkel op deze wijze wanneer er sprake is van fragmentatie, de toewijzing van de clusters is zelf volledig weggevallen aangezien deze overgenomen werd door de bitmap. De ketening en bitmap komen in de verdere hoofdstukken nog uitgebreid aan bod. ExFAT bevat bovendien nog maar 1 File Allocation Table in tegenstelling tot zijn voorgangers die telkens ook een kopie opsloegen.

Zoals in vorig hoofdstuk beschreven is, kunnen de startpositie en de grootte van de FAT teruggevonden worden in het VBR (0x50). In de File Allocation Table bij exFAT, wordt verwezen naar de cluster van een bestand op de gegevensdrager. Elke waarde bevat 4 bytes en geeft ofwel aan dat een cluster leeg is (waarde 00 00 00 00), dat de desbetreffende cluster de laatste cluster is van het bestand (FF FF FF FF) of geeft een numerieke waarde weer die de startpositie van de cluster bevat. OPGELET: de eerste 2 clusters in de FAT zijn steeds gevuld, de eerste cluster die gebruikt wordt, zal dus de 3e cluster zijn en niet de eerste.

Op die manier kan simpelweg het pad gevolgd worden om de data van een bestand chronologisch na mekaar te plaatsen.

Een voorbeeld van het gebruik van de FAT is hieronder terug te vinden:

File Allocation Table						
➔ Cluster 4 (04 00 00 00)	➔ Cluster 7 (07 00 00 00)	Niet in gebru - (00 00 00 00)	-ik door FAT (00 00 00 00)	Einde verwijzing FF FF FF FF	Nt in gebruik 00 00 00 00	…
Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	…

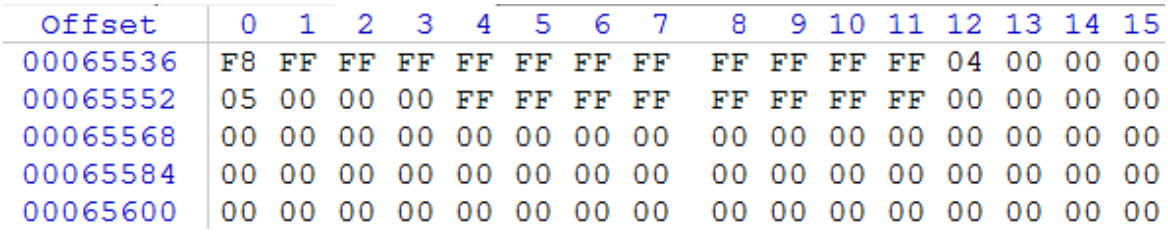
Datazone van de gegevensdrager						
Data	Data			Data		…
Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	…

Bestand		
Data	Data	Data
Cluster 3	Cluster 4	Cluster 7

Illustratie van een gefragmenteerd bestand en hoe het zich fysiek in de datazone en FAT bevindt.

De derde (effectieve) waarde in de File Allocation Table representeert de 3^{e} cluster in de datazone maar bevat de waarde van de volgende cluster. In dit voorbeeld de 4^{e} cluster, die op zijn beurt verwijst naar de 7^{e} cluster. Omdat de 7^{e} cluster de waarde FF FF FF FF bevat, is geweten dat deze cluster de laatste is van het voorbeeldbestand.

OPGELET: het kan ook zijn dat naar eerdere clusters verwezen wordt. Gefragmenteerde bestanden worden geplaatst waar de drager ruimte heeft. Het is niet gezegd dat die ruimte zich na het eerst geplaatste stuk van het bestand bevindt.



Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00065536	F8	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	04	00	00	00
00065552	05	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	00	00	00	00
00065568	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00065584	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00065600	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Deze afbeelding illustreert een stuk uit de File Allocation Table getoond in hexadecimale weergave met het programma winhex.

Tot slot kan aan de hand van het verkregen clusternummer ook snel de startende 4-byte-waarde van het bestand/ de map in de FAT teruggevonden worden.

Formule:

*(Start van FAT * sectorgrootte) + (Clustern° * 4)*

De start van de FAT kan in het VBR verkregen worden. De waarde die erin verkregen wordt dient nog vermenigvuldigd te worden met het aantal bytes (altijd 512 want de FAT bevindt zich in de systeemzone en niet in de datazone). Deze waarde dient bij het clusternummer geteld te worden dat met 4 vermenigvuldigd is aangezien elke cluster in 4 bytes beschreven wordt.

8.2 Werking van exFAT

8.2.1 Introductie

Onderzoekt toont aan dat voor de goede werking in exFAT de driver moet starten met het zoeken naar de startcluster van de root directory in het Volume Boot Record. De driver zal in het record ook zoeken naar de grootte van de clusters en de start van de datazone. Op basis van de verkregen informatie zullen de essentiële bestanden en instellingen teruggevonden kunnen worden die de correcte werkwijze van exFAT kunnen verzorgen. Eén van de nieuwigheden in exFAT is het ‘bestand’ BITMAP. Het voordeel van de bitmap is het versnellen van een aantal bewerkingen. Door de toevoeging van de BITMAP is de werking van het toekennen van clusters aan bestanden gewijzigd. De File Allocation Table zal alleen nog gebruikt worden voor gefragmenteerde bestanden. Voor dergelijke bestanden zal exFAT ook terugvallen op de oude methode van clustertoewijzing.

8.2.2 (Root)Directory

De rootdirectory en subdirectory's bevatten een aantal belangrijke elementen die de huidige werking van exFAT mogelijk maken. Elk bestand of elke directory wordt door een main entry en door subentry's beschreven. De entry's bestaan telkens uit 32-byte-waarden, die informatie (in unicode) weergeven over onder andere, de bestand-/mapentry, clusterstromen en de bestandsnamen. De root directory bevat ook "systeembestanden" die informatie in entry's beschrijven over het volumelabel, de bitmap en de uppercasetabel.

Aan de hand van de start van elk van zo'n entry kan opgemaakt worden wat juist beschreven staat. De eerste byte wordt daarom telkens voor deze waarde gereserveerd. De resterende bytes zullen informatie bevatten over het betrokken element.

Informatie over het volumelabel, de bitmap en de uppercasetabel bevinden zich één keer in de rootdirectory. Informatie over bestanden en mappen kan zich zowel in de root directory als in andere mappen bevinden. Het valt op dat startwaarde '0x85' vaak terugkomt, deze geeft immers de start aan van een bestand-/mapentry in een map. Logischerwijze kan een onderzoeker op basis van deze waarden belangrijke elementen terugvinden tijdens zijn onderzoek.

Volgende elementen beschrijven de verschillende entry's die zich in mappen kunnen bevinden:

Startwaarde	**Suboffset (byte #0)**	**Lengte (bytes)**	**Inhoud**
0x81	**Bitmapinformatie entry**		
	0	1	Bevat de waarde 0x81
	20	4	Eerste cluster (Meestal 0x03)
	24	8	Grootte in bytes = ((Aantal clusters - 2)/8)+1
0x82	**Uppercase tabel entry**		
	0	1	Bevat de waarde 0x82
	20	4	Eerste cluster (Meestal 0x02)
	24	8	Grootte in bytes

0x83	**Volume label entry**		
	0	1	Bevat de waarde 0x83
	1	1	Lengte van de volumenaam, max. 11
	2	Naamlengte * 2	Volumenaam (Unicode karakters, max 11 karakters lang)
0x85	**Bestand/Map main entry**		
	0	1	Bevat de waarde 0x85 (= 0x05 wanneer niet meer in gebruik)
	1	1	Aantal subentry's voor het bestand
	4	1	DOS-attributen
	8	4	Datum en tijd van aanmaak (DOS-formaat)
	12	4	Datum en tijd van laatste keer geopend/bekeken (DOS-formaat)
	16	4	Datum en tijd van laatste aanpassing (DOS-formaat)
	20	1	Aanmaaktijd bias (1/100s)
	21	1	Aanpassingstijd bias (1/100s)
	22	1	Tijdzone van aanmaak
	23	1	Tijdzone van laatste keer geopend/bekeken
	24	1	Tijdzone van laatste aanpassing
0xC0	**Clusterverloop**		
	0	1	Bevat de waarde 0xC0 (= 0x40 wanneer niet gebruikt)
	1	1	Geeft in de byte, in bits aan, of FAT gebruikt wordt (0x02 = niet). De betekenis van andere waarden is niet gekend.
	3	1	Bestandsnaamlengte
	20	4	Nummer van de startcluster van het bestand
	24	8	Bestandsgrootte van het bestand in bytes
0xC1	**Bestandsnaam**		
	0	1	Bevat de waarde 0xC1 (= 0x41 wanneer niet gebruikt)
	2	30	Bestandsnaam in unicode (max. 15)

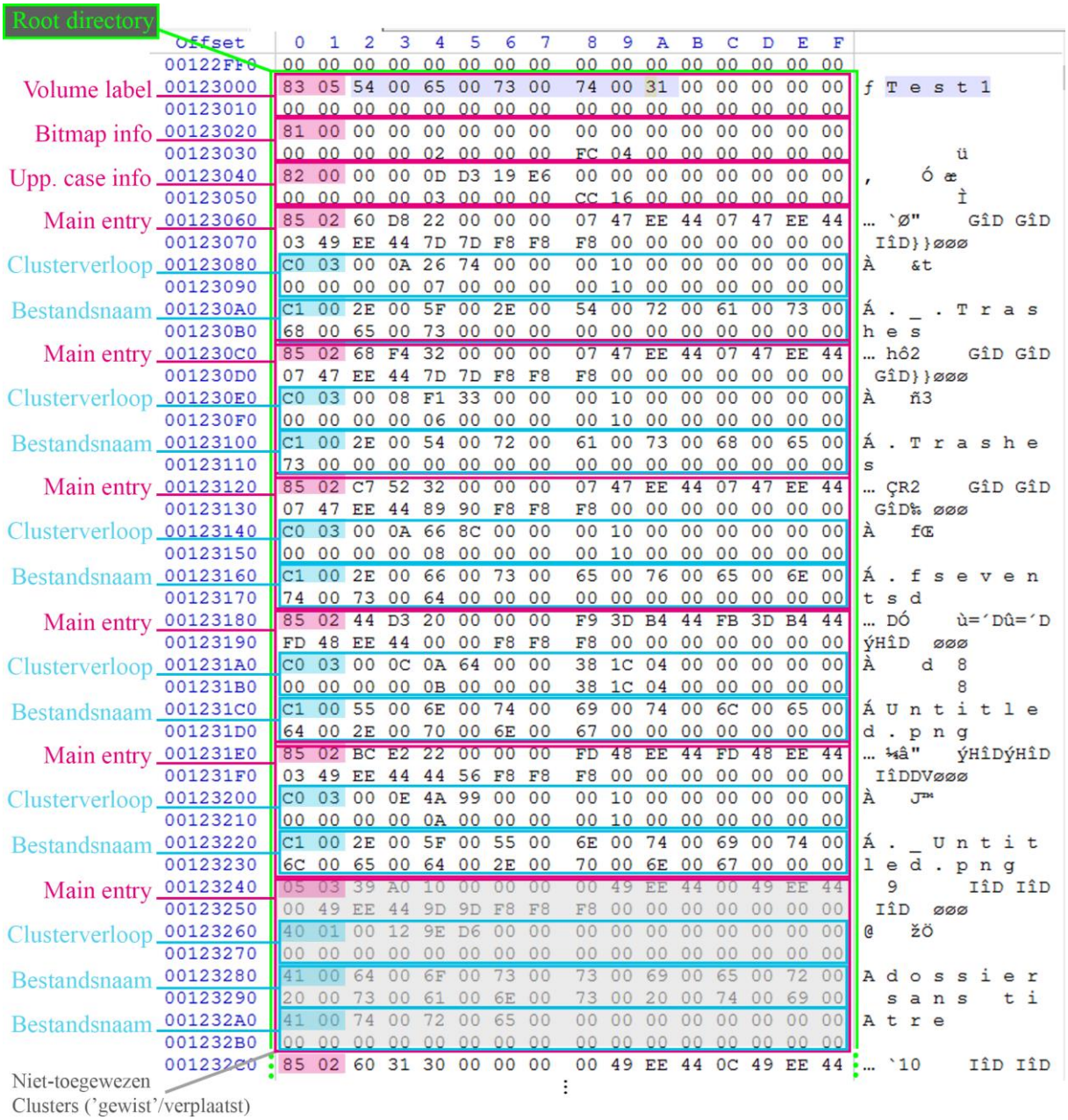
Deze tabel beschrijft de directorieëntry's.

Op basis van bovenstaande tabel kan dus erg belangrijke informatie teruggevonden worden. Uit de entry's kan een onderzoeker bijvoorbeeld de startpositie op de drager terugvinden, kan hij opmaken wanneer bestanden aangemaakt werden, enz.

Het is belangrijk om weten dat de datums DOS-gecodeerd genoteerd worden. Aangezien deze decodering buiten de scope van dit project gaat, zal deze niet besproken worden. Meer informatie hieromtrent is reeds uitgebreid beschikbaar.

De eerste cluster van de root directory kan teruggevonden worden in het VBR. Deze waarde is gerekend vanaf de start van de datazone. Hierdoor dient de start van de datazone toegevoegd te worden aan de genoteerde waarde. Met volgende formule kan de startpositie, relatief vanaf de start van de partitie, van de root directory snel teruggevonden worden:

*(Start datazone * 512) + ((1e cluster root directory - 2) * bytes per cluster)*



Voorbeeld van een root directory en bestandsinformatie die zich erop bevindt. Hexadecimale weergave in winhex.

8.2.3 Bitmap

Het doel van de bitmap is het bijhouden van toewijzing van de clusters. De bitmap heeft de grootte van het aantal clusters in bits (naar boven afgerond in bytes), dit omdat voor elke cluster een bit voorzien is in de bitmap. Het enige wat de bitmap doet is bijhouden of de cluster toegewezen (bezet, bit heeft waarde 1) is of niet (bit heeft waarde 0). De bitmap houdt, in tegenstelling tot de File Allocation Table, de ketening (chaining) van de bestanden niet bij.

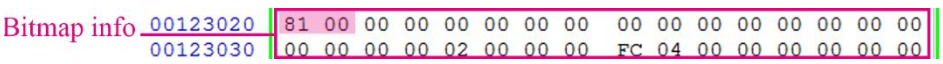
Omdat de bitmap minder informatie bevat dan de File Allocation Table zullen de bewerkingen dus sneller worden uitgevoerd.

De grootte van de bitmap in bytes kan verkregen worden met onderstaande formule:

$$\frac{((\text{aantal clusters} - 2)) + 1}{8 \text{ (bytes)}}$$

De bitmap bevindt zich, zoals in vorige hoofdstukken aangegeven, in de datazone. In praktijk is op te merken dat de bitmap zich vaak net na de FAT bevindt. In theorie is het echter perfect mogelijk dat de bitmap zich op een andere positie bevindt. Een praktisch voorbeeld valt op te merken bij het vergroten van de partitie. Op dat moment moeten immers het aantal clusters herteld worden en zal de bitmap mee vergroten. Omdat de bitmap zich in de datazone bevindt, kan het zijn, dat net na de bitmap, reeds andere data geschreven staat. Het zou onlogisch zijn om dan alle data op de schijf te gaan verplaatsen, dit proces zou immers heel lang duren en is vrij omslachtig. ExFAT zal de volledige nieuwe bitmap, inclusief de data van de oude bitmap, op een lege plaats op de harde schijf opslaan. De oude bitmap wordt op dat moment niet meer gebruikt, de voor de oude bitmap gebruikte clusters krijgen de status 'niet-toegewezen' (unallocated) en de nieuwe bitmap zal zich op een andere locatie in de datazone bevinden.
Deze door exFAT gehanteerde methode, biedt een groot voordeel voor de forensische analyse. Ze kan immers volledig gerecupereerd worden zolang er nog geen nieuwe gegevens over de niet-toegewezen clusters geschreven werd. Het moet wel gezegd worden dat het interpreteren van een bitmap erg ingewikkeld is en dat de verkregen informatie niet automatisch resulteert in bruikbaar bewijsmateriaal.

De startpositie van de bitmap staat beschreven na de waarde 0x81 in de root. Zowel de ketening van de bitmap als die van de root directory en de uppercasetabel, zijn altijd beschreven in de File Allocation Table. Dit omdat de aanwezigheid van deze 3 systeembestanden een vereiste is voor de goede werking van exFAT en omdat de positie van deze systeembestanden zich theoretisch gezien op een welke locatie zouden kunnen bevinden.



De 'bitmap entry' in de root directory die informatie geeft over de locatie van de bitmap en over de grootte van de bitmap in bytes. In dit voorbeeld bevindt de bitmap zich op positie 00 00 00 02. Aan deze waarde dient wel nog de data heap toegevoegd te worden. Hexadecimale weergave in winhex.

De waarde die in deze entry genoteerd staat is niet de effectieve startwaarde maar de waarde gerekend vanaf de start van de datazone. Hierdoor dient de start van de datazone toegevoegd te worden aan de genoteerde waarde. Met volgende formule kan de werkelijke startpositie van de bitmap snel teruggevonden worden:

*(Start datazone * bytes/sector) + ((1e cluster bitmap – 2) * bytes/cluster)*

Wanneer de formule wordt toegepast op bovenstaand voorbeeld, wordt volgend resultaat bekomen:

(384 * 512) + ((02 – 2)*512) = 196 608.

De waarde 384 werd in het VBR teruggevonden op positie 0x58. De waarde 02 (*00 00 00 02)* werd teruggevonden in de bitmapentry. 196 608 is dus startpositie van de bitmap.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00196608	FF	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00
00196624	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Voorbeeld van een bitmap in hexadecimale weergave met winhex.

In sommige gevallen kan het nuttig zijn om de bitmap uit te lezen om te weten te komen of een cluster de status ‘toegewezen’ heeft. Indien dit niet het geval is, wil dit zeggen dat de bitmap de cluster vrijgeeft en dat er een ander bestand op geschreven mag worden. Dit zou als bewijsmateriaal gebruikt kunnen worden om aan te tonen dat een bestand ‘gewist werd. Het uitlezen van de bitmap gebeurt op volgende wijze:

a) De gewenste byte in de bitmap terugvinden;

Elke byte in de bitmap slaagt 8 bits op. Elke bit representeert een cluster. Hierdoor is de eerste bit in de bitmap de eerste cluster in de datazone voorstelt. Omdat de nummering van de clusters, om technische redenen, start bij 2, moeten deze in mindering gebracht worden in de onderstaande formule. Met de formule kan de byte bekomen worden waarin de cluster beschreven wordt. In de bestandsentry kan het clusternummer

teruggevonden worden. Op basis van dit nummer kan het gewenste bytenummer (Bn) in de bitmap teruggevonden worden.

Formule:

$$\frac{(\text{Clusternummer} - 2)}{8\ (\#\ bits/byte)} = Bn$$

Voorbeeld 1.a

```
001231A0  C0 03 00 0C 0A 64 00 00  38 1C 04 00 00 00 00 00
001231B0  00 00 00 00 0B 00 00 00  38 1C 04 00 00 00 00 00
001231C0  C1 00 55 00 6E 00 74 00  69 00 74 00 6C 00 65 00
001231D0  64 00 2E 00 70 00 6E 00  67 00 00 00 00 00 00 00
```

Dit is een voorbeeld van een bestandsentry. De geselecteerde waarden beschrijven de startcluster van het bestand.

Aan de hand van het clusternummer kan de formule als volgt worden gebruikt (0B 00 00 00 = 11) :

(11 – 2)/8 = 1 = Bytenummer.

b) De gewenste bit terugvinden in de byte;

Eén van de 8 verkregen bits die in de bitmapbyte, bevat de waarde van de in vraag gestelde cluster. De bitmap beschrijft in zijn byte de clusters van laag naar hoog. Dit zorgt dat de byte van rechts naar links gelezen dient te worden.

Hoewel de bit snel uitgelezen kan worden, kan ook volgende formule gebruikt worden om het bitnummer (bn) terug te vinden. Omwille van technische redenen starten de clusternummers bij 2, dit getal dient dan ook in mindering gebracht te worden in de formule.

Formule:

$$(Clustern° - 2) - (Bn * 8) = bn$$

Voorbeeld 1.b:

Aan de hand van het verkregen bytenummer kan de formule als volgt worden gebruikt:

(11 - 2) – (1 * 8) = 1 = Bitnummer.

Bytenummer 1 uit bitmap

Clustern°	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**
Waarde	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

Bitnummer 1

Deze illustratie toont de betrokken bit uit het voorbeeld. De tweede cluster heeft de waarde 1 (= toegewezen).

8.2.4 Chaining

Het lijkt vreemd dat de File Allocation Table niet meer gebruikt wordt. Deze tabel stond in het verleden in voor het toewijzen van clusters aan bestanden. Bij niet-gefragmenteerde bestanden zal exFAT echter een nieuwe aanpak voorzien. Aangezien het bestandsysteem nu over een bitmap beschikt, zal het snel kunnen uitmaken welke ruimtes vrij zijn op de harde schijf. Hierdoor kan het door middel van chaining (ketenen) bestanden aan vrije clusters toewijzen. Chaining houdt in een bestand over vrije clusters wordt verspreid die elkaar chronologisch volgen. Door het hanteren van deze werkwijze, dienen enkel nog de startpositie van de eerste cluster en de laatste cluster gekend te zijn. Het bestand bevindt zich immers tussen deze posities.

Wanneer een gefragmenteerd bestand na verkleining aaneensluitend op de drager geplaatst kan worden (niet-gefragmenteerd), zal de bestandsentry aangeven dat het bestand niet gefragmenteerd is. Toch zal de FAT de verwijzingen naar oude clusters nog tonen tot deze overschreven wordt. Op deze manier kan het keteningmechanisme bij het verkleinen van bestanden een forensisch voordeel bieden bij het inkijken van de oude inhoud van het bestand. De Uppecasetable, de root directory en de bitmap, zijn altijd in de FAT beschreven.

Clusters	**…**	**5**	**6**	**7**		**…**
Data	00 … 00	xx … xx	xx … xx	xx ..	00	00 … 00

Voorbeeld van de chronologische ketening van een bestand dat verspreid over 3 clusters geschreven is. Voorstelling in hexadecimale weergave.

8.2.5 Bestand aanmaken

Het aanmaken van een bestand kent meerdere fases en verloopt niet altijd op dezelfde manier. Een bestand kan zich immers gefragmenteerd op de drager bevinden of kan opgebouwd worden doormiddel van de keteningmethode (zie hoofdstuk ketening/chaining). ExFAT opteert om zo weinig mogelijk gebruik te maken van fragmentatie en de FAT. Enkel wanneer er op de drager niet genoeg vrije, aaneensluitende, clusters zijn, zal voor fragmentatie gekozen worden.

De onderstaande methodes worden beschreven aan de hand van een voorbeeld waarbij de drager een clustergrootte omvat van 4096 en waarbij het bestand 6000 bytes groot is. De naam van het bestand zal in dit voorbeeld door 2 x 20 bytes beschreven worden en bevat 20 unicode-karakters.

8.2.5.1 Optie 1: Er is genoeg ruimte op de drager om een bestand aaneensluitend op de drager te plaatsen.

a. De bestandsentry wordt in de gewenste map aangemaakt;

Deze entry start met 0x85 (zie tabel bij hoofdstuk (root)directory en wordt gevolgd door het aantal subentry's die de naam zal bevatten. In dit voorbeeld wordt 0x85 gevolgd door 0x03, deze waarde geeft aan dat de entry door 3 subentry's gevolgd wordt.

85	**03**	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx
xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx

Deze illustratie geeft de main entry in hexadecimale weergave van het voorbeeldbestand weer. De xx stelt de waarde van het bestand voor en zal uit unicode-karakters bestaan.

De volgende entry (32-byte-waarde) start met 0xC0 en zal vertellen dat de FAT niet gebruikt moet worden (omdat het bestand aaneensluitend op de drager geplaatst kan worden). Dit wordt in dit voorbeeld aangegeven met de waarde 0x02 die direct na de startwaarde volgt. In deze entry staan onder andere ook de startcluster en de bestandsgrootte van het bestand genoteerd.

C0	**02**	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx
xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx

Deze illustratie geeft de bestandslocatieëntry in hexadecimale weergave van het voorbeeldbestand weer.

De laatste entry's beschrijven de bestandnaam van het bestand. Ze worden aangegeven met de waarden C1 en kunnen maximum 15 unicode-karakters bevatten aangezien de eerste byte voorzien is voor het aangeven van de startwaarde van de entry. Elk karakter bestaat uit 2 bytes. Wanneer, zoals in dit voorbeeld, een lange bestandsnaam gebruikt wordt, zullen meerdere bestandsnaamentry's aangemaakt moeten worden.

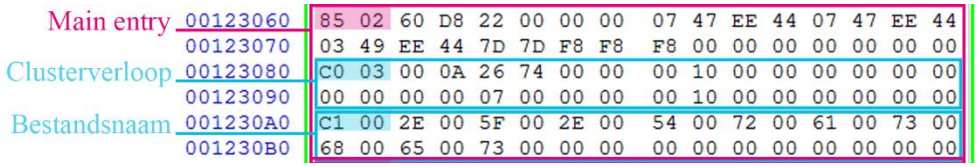
C1	**00**	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx
xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx

Deze illustratie geeft de eerste bestandsnaamentry in hexadecimale weergave van het voorbeeldbestand weer. De xx bevatten unicode-karakters.

C1	**00**	xx	xx	xx	xx	xx	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**
00	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

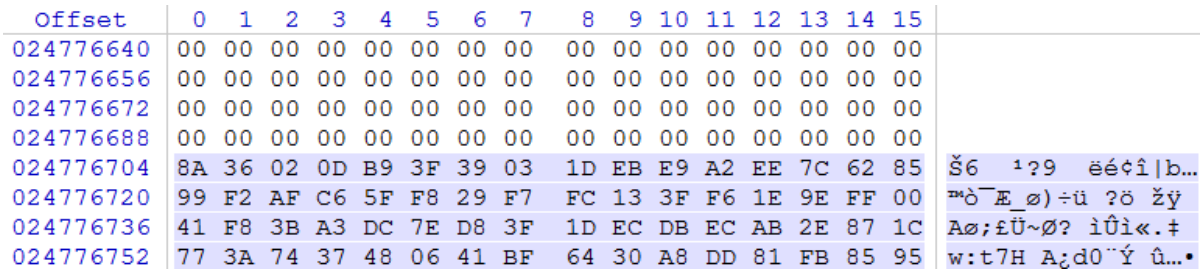
Deze illustratie geeft de tweede bestandsnaamentry in hexadecimale weergave van het voorbeeldbestand weer. De xx bevatten unicode-karakters. De 00 staan voor lege bytes.

Onderstaande afbeelding toont een samenvattend voorbeeld van een aangemaakte main entry.



Deze afbeelding illustreert een aangemaakte main entry in een directory in hexadecimale weergave in winhex.

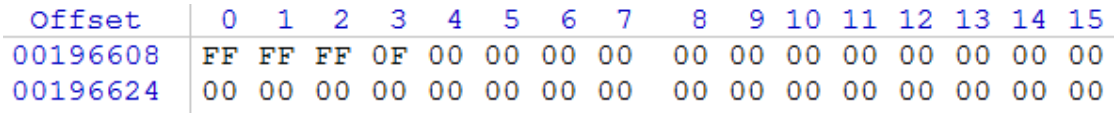
b. ***Het bestand wordt aaneensluitend, op een vrije positie op de drager geplaatst. Omdat het bestand op chronologisch volgende clusters geplaatst wordt, dient enkel de bestandsgrootte en startcluster gekend te zijn in de root directory. Op basis van deze informatie kan immers van start- tot eindcluster, de data continu uitgelezen worden;***



Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
024776640	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
024776656	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
024776672	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
024776688	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
024776704	8A	36	02	0D	B9	3F	39	03	1D	EB	E9	A2	EE	7C	62	85	Š6 ¹?9 ëé¢î|b…
024776720	99	F2	AF	C6	5F	F8	29	F7	FC	13	3F	F6	1E	9E	FF	00	™ò¯Æ_ø)÷ü ?ö žÿ
024776736	41	F8	3B	A3	DC	7E	D8	3F	1D	EC	DB	EC	AB	2E	87	1C	Aø;£Ü~Ø? ìÛì«.‡
024776752	77	3A	74	37	48	06	41	BF	64	30	A8	DD	81	FB	85	95	w:t7H A¿d0¨Ý ûˆ…

Deze afbeelding illustreert de inhoud van een bestand, geplaatst in clusters op een harde schijf. Een niet gefragmenteerd bestand wordt, zonder onderbrekingen, aaneensluitend op de drager geplaatst. Dit bestand wordt in hexadecimale weergave getoond in een directory in hexadecimale weergave in win-hex. De cluster start op offset 24776704.

c. ***De door het bestand bezette clusters worden in de bitmap op 'bezet' geplaatst.***



Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00196608	FF	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00
00196624	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Deze afbeelding illustreert een voorbeeld van een bitmap waarin bestandsgerelateerde clusters de status 'bezet' gekregen hebben (FF). 00 houdt in dat de respectievelijke clusters niet toegewezen zijn en dus beschreven kunnen worden.

Een bijkomend element is dat de driver van een MAC-computer bij het aanmaken van een nieuw bestand eerst een map creëert met de naam 'nieuwe map', vervolgens de naam entry van deze map 'wist' (op normale wijze, en dus nog steeds zichtbaar voor forensische onderzoekers), om tot slot een nieuwe naam entry aan te maken met de naam die door de gebruiker wordt meegegeven. Bij het aantreffen van deze bevinding kan ervanuit gegaan worden dat de map vanaf een MAC-computer werd aangemaakt. Ook de taal kan hieruit afgeleid worden, "new folder" zal bijvoorbeeld gebruikt worden bij Engelstalige instellingen, "nieuwe map" bij Nederlandstalige instellingen.

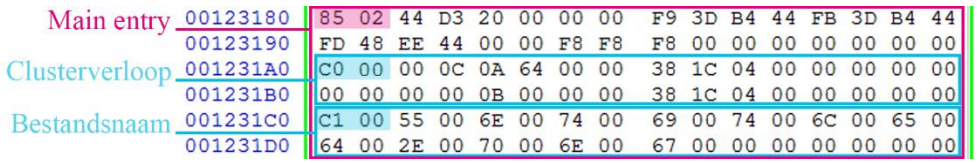
8.2.5.2 Optie 2: Er is niet genoeg ruimte op de drager om een bestand aaneensluitend op de drager te plaatsen.

Aangezien de drager niet genoeg ruimte bevat om het bestand er aaneensluitend op te plaatsen zal gebruikt gemaakt moeten worden van de File Allocation Tabel (FAT).

a. Idem aan optie 1 met 1 groot verschil; de eerste byte van de bestandslocatieëntry zal niet de waarde C0 02 bevatten. Wanneer tweede bit de waarde 0 heeft, geeft deze aan dat voor dit bestand de FAT gebruikt dient te wordt;

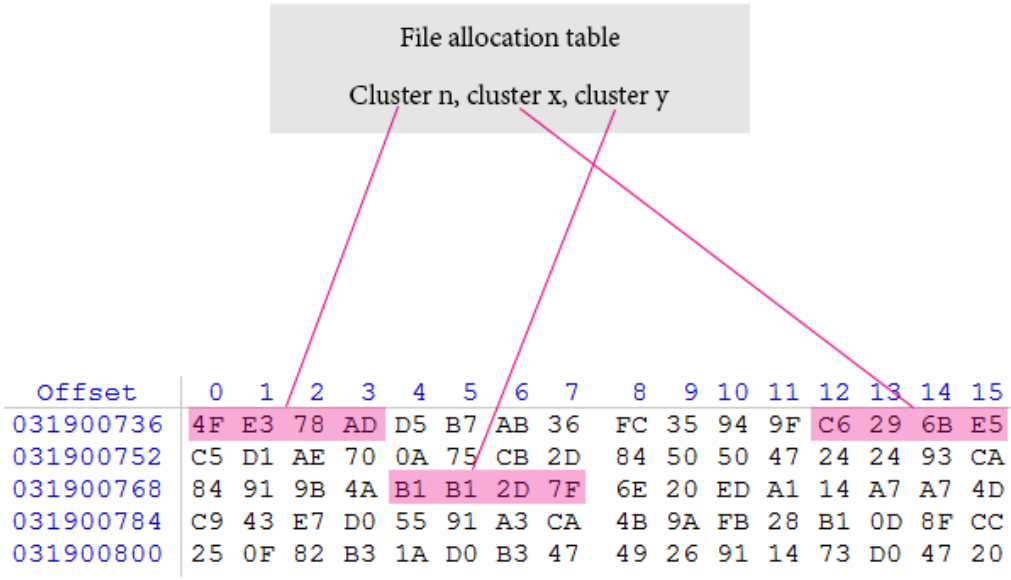
C0	**00**	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx
xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx

Deze illustratie geeft de bestandslocatieëntry in hexadecimale weergave van het voorbeeldbestand weer. Het verschil met optie 1 is dat de eerste byte van de entry een andere waarde bevat (C0 00 i.p.v. C0 02) .



Deze afbeelding illustreert een aangemaakt main entry in een directory in hexadecimale weergave in winhex. OPGELET: de bestandlocatieëntry bedraagt C0 00, het bestand is dus gefragmenteerd.

b. ***Het bestand wordt door middel van de FAT gefragmenteerd op de drager geplaatst, de File allocation table wordt gebruikt om bij te houden welk deel van het bestand zich in welke cluster bevindt. De eerste cluster zal in de File Allocation Table een pointer bevatten die verwijst naar de positie van de tweede cluster van het bestand. De derde bevat de positie van de 4^{e} cluster, enz. De laatste cluster van het bestand krijgt de waarde FF FF FF FF in de FAT;***



Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
031900736	4F	E3	78	AD	D5	B7	AB	36	FC	35	94	9F	C6	29	6B	E5
031900752	C5	D1	AE	70	0A	75	CB	2D	84	50	50	47	24	24	93	CA
031900768	84	91	9B	4A	B1	B1	2D	7F	6E	20	ED	A1	14	A7	A7	4D
031900784	C9	43	E7	D0	55	91	A3	CA	4B	9A	FB	28	B1	0D	8F	CC
031900800	25	0F	82	B3	1A	D0	B3	47	49	26	91	14	73	D0	47	20

Deze afbeelding illustreert de plaatsing van een gefragmenteerd bestand op een harde schijf met behulp van de File Allocation Table.

c. De door het bestand bezette clusters worden in de bitmap op 'bezet' geplaatst.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00196608	FF	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00
00196624	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Deze afbeelding illustreert een voorbeeld van een bitmap waarin bestandsgerelateerde clusters de status 'bezet' gekregen hebben (FF). 00 houdt in dat de respectievelijke clusters niet toegewezen zijn en dus beschreven kunnen worden.

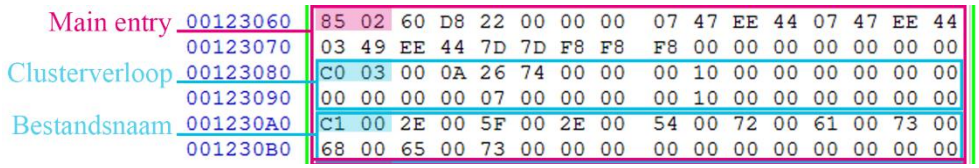
8.2.6 Bestand wissen (status niet-gebruikt)

Het wissen van een bestand, wordt net zoals het aanmaken en andere bewerkingen, uitgevoerd door een driver. Zoals in dit project reeds werd aangehaald, kunnen bewerkingen door drivers verschillen van elkaar. Dit kan zowel hetzelfde als een verschillend resultaat tot gevolg hebben. Hieronder wordt de meest logische en meest gebruikte methode beschreven. Deze methode wordt het meest gebruikt omdat ze het schrijven op de harde schijf zo beperkt mogelijk houdt.

Aangezien het aanmaken van een bestand met of zonder fragmentatie kan gebeuren, zal het wissen van een bestand vanuit hetzelfde standpunt bekeken worden.

8.2.6.1 Optie 1: Het bestand is niet gefragmenteerd.

a. *Er wordt naar de main entry gezocht (waarde 0x85);*

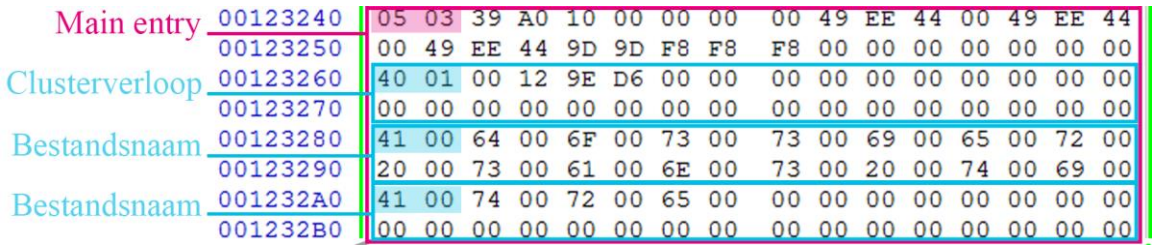


Deze afbeelding illustreert een aangemaakte main entry in een directory in hexadecimale weergave in winhex.

b. *De eerste bit van de main entry en diens subentry's worden op 0 geplaatst. Voor de waarde 85 wordt dit 05, voor C0 wordt dit 40 en voor C1 wordt dit 41. De daaropvolgende informatie wordt niet aangeraakt! Dit geeft een opvallend en zeer interessant voordeel vanuit forensisch oogpunt. De informatie die erin geschreven staat kan namelijk volledig terug gehaald worden. Het is pas op het moment dat een ander bestand aan de clusters van deze ongebruikte entry's wordt toegewezen, dat de informatie effectief 'gewist' wordt. Tot op dat moment wordt in dit hoofdstuk, van een 'ongebruikte' cluster en niet van een gewiste/lege cluster, gesproken. Bovendien kan men zelfs wanneer nieuwe informatie over een ongebruikte cluster geschreven wordt, delen uit diens slack space recupereren;*

Voorbeeldentry uit vorig hoofdstuk		**Gewiste voorbeeldentry uit vorig hoofdstuk**
85 03 xx xx xx xx xx xx ...		**05** 03 xx xx xx xx xx xx ...
xx xx xx xx xx xx xx xx ...		xx xx xx xx xx xx xx xx ...
C0 02 xx xx xx xx xx xx ...		**40** 02 xx xx xx xx xx xx ...
xx xx xx xx xx xx xx xx ...	=>	xx xx xx xx xx xx xx xx ...
C1 00 xx xx xx xx xx xx ...		**41** 00 xx xx xx xx xx xx ...
xx xx xx xx xx xx xx xx ...		xx xx xx xx xx xx xx xx ...
C1 00 xx xx xx xx xx 00 ...		**41** 00 xx xx xx xx xx 00 ...
00 00 00 00 00 00 00 00 ...		00 00 00 00 00 00 00 00 ...

Deze illustratie toont de gewiste entry's in hexadecimale weergave van het voorbeeldbestand uit vorig hoofdstuk.



Deze afbeelding illustreert een gewiste main entry in een directory in hexadecimale weergave in win-hex. De waarden 85 03, C0 01 en C1 00 werden gewijzigd naar 05 03, 40 01 en 41 00.

c. De ongebruikte bits die de ongebruikte clusters representeren krijgen in de bitmap de waarde 0.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00196608	FF	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00
00196624	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Deze afbeelding illustreert een voorbeeld van een bitmap waarin sommige bits in de bestandsgerelateerde clusters de status 'niet-toegewezen/niet-bezet' gekregen hebben (0). (0F = 0000 1111).

8.2.6.2 Optie 2: Het bestand is gefragmenteerd.

Een ander forensisch voordeel bij deze methode van wissen, is dat er geen verschil is tussen het verwijderen van een gefragmenteerd of niet-gefragmenteerd bestand. Bij het verwijderen van een gefragmenteerd bestand, worden stap a, b en c van optie uitgevoerd. De FAT blijft onaangeroerd! Hierdoor blijft de informatie gewoon aanwezig in de FAT. Als rechtstreeks gevolg kunnen gefragmenteerde bestanden theoretisch gezien dus probleemloos teruggehaald worden. (onder de voorwaarde dat de clusters niet hergebruikt door een nieuw bestand, dit kan gecheckt worden bij de analyse van de bitmap).

Theoretisch gezien zou een driver een al deze waarden op 0 kunnen plaatsen. Wanneer dit gebeurt, is de informatie volledig gewist. In praktijk kon geen voorbeeld gevonden van dergelijke werking. Het doel van exFAT is tevens ook om de schrijfbewerkingen te beperken en zou teniet gedaan worden bij deze werkwijze.

8.2.7 Bestand aanpassen

Zoals bij bovengenoemde bewerkingen, worden bestanden aangepast door middel van drivers. Ook bij dit hoofdstuk zal de gebruikelijke methode van verplaatsen besproken worden. Volgende handelingen kunnen worden genoteerd bij het aanpasbewerkingen:

8.2.7.1 Bestand verplaatsen

Het verplaatsen van een bestand is een erg eenvoudig proces. Main entry wordt aangemaakt in de gewenste map, sub-entries worden naderhand gekopieerd. De oude main entry en relatieve sub-entries worden als “unallocated” aangeduid. De plaats van het bestand op de drager wijzigt niet, bitmap wordt niet aangepast. Hierdoor vereist de bewerking weinig schrijflast.

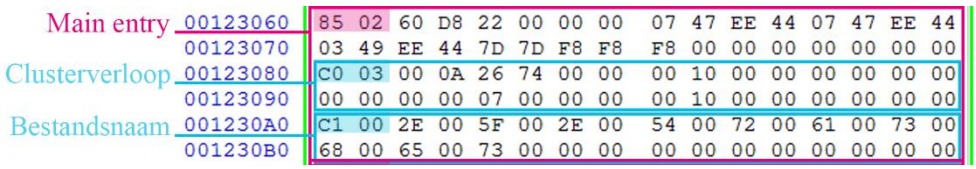
Er dient omzichtig omgesprongen te worden met wissen en verplaatsen. Een onderzoeker kan bijvoorbeeld opmerken dat bepaalde entry’s gewist werden. Het kan zijn dat deze entry’s verplaatst werden en niet gewist. Hierdoor zou een verkeerde conclusie getrokken kunnen worden.

8.2.7.2 Bestandsnaam aanpassen

Wanneer een bestand wordt aangemaakt krijgt de main entry van het bestand de bestandsnaam mee. Wanneer een nieuw bestand aangemaakt wordt, zal diens main entry vaak in de chronologisch volgende cluster weggeschreven worden. Het zou immers onlogisch zijn om lege plaatsen te laten op de drager. Wanneer een bestandsnaam echter aangepast wordt, kan het zijn dat deze uit minder of juist meer entry’s bestaat.

8.2.7.2.1 Optie 1: De bestandsnaam verkleint

a. Er wordt naar de main entry gezocht (waarde 0x85);

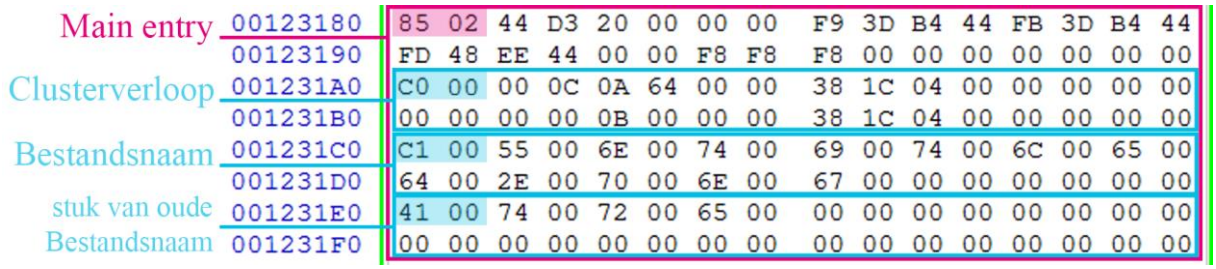


Deze afbeelding illustreert een aangemaakt main entry in een directory in hexadecimale weergave in winhex.

b. De niet-gebruikte entry's zullen de status 'niet-gebruikt' krijgen, het aantal gebruikte subentry's bij C0 wordt eveneens aangepast;

Voorbeeldentry uit vorig hoofdstuk		**Gewiste voorbeeldentry uit vorig hoofdstuk**
85 03 xx xx xx xx xx xx ...		85 02 xx xx xx xx xx xx ...
xx xx xx xx xx xx xx xx ...		xx xx xx xx xx xx xx xx ...
C0 02 xx xx xx xx xx xx ...		C0 02 xx xx xx xx xx xx ...
xx xx xx xx xx xx xx xx ...	=>	xx xx xx xx xx xx xx xx ...
C1 00 xx xx xx xx xx xx ...		C1 00 xx xx xx xx xx xx ...
xx xx xx xx xx xx xx xx ...		xx xx xx xx xx xx xx xx ...
C1 00 xx xx xx xx xx 00 ...		**41** 00 **xx xx xx xx xx** 00 ...
00 00 00 00 00 00 00 00 ...		00 00 00 00 00 00 00 00 ...

Deze illustratie toon de aangepaste bestandsnaam in de entry's in hexadecimale weergave. Er is sprake over hetzelfde voorbeeldbestand uit vorig hoofdstuk. De laatste bestandsnaamentry bevat een deel van de oude naam van het voorbeeldbestand.

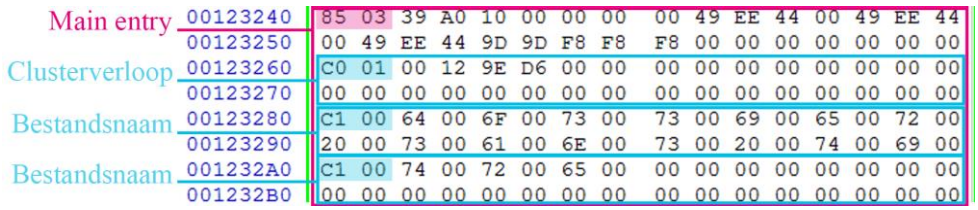


Deze afbeelding illustreert een het kleiner maken van een bestandsnaam. Hexadecimale weergave met winhex. 74 00 72 00 65 zijn karakters die afkomstig zijn uit de oude bestandsnaam.

8.2.7.2.2 Optie 2: De bestandsnaam vergroot, er is niet genoeg plaats na de cluster van de main entry

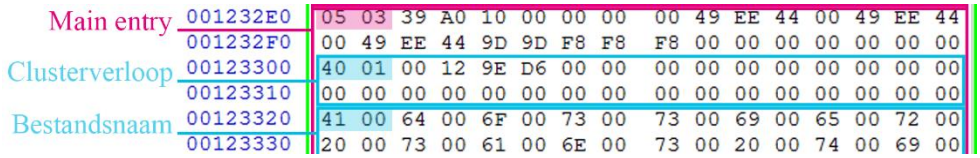
Het is vanzelfsprekend dat, wanneer er wel plaats na de cluster van de main entry zou zijn, de main entry gewoon uitgebreid zou worden.

a. De main entry wordt op een lege plaats op de drager geplaatst;



Deze afbeelding illustreert de nieuwe main entry op een nieuwe plaats in de directory. In hexadecimale weergave in winhex.

b. De oude main entry krijgt de status 'niet-gebruikt';



Deze afbeelding illustreert de oude, 'gewiste', main entry op de oude plaats in de directory.

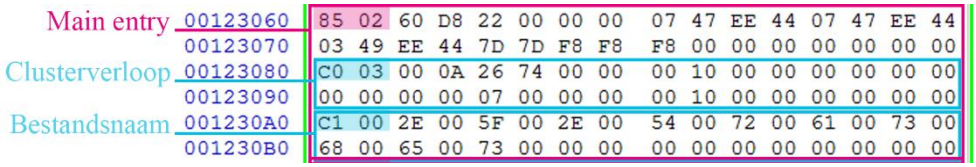
8.2.8 Bestand terughalen

De nieuwe werking van exFAT brengt een groot forensisch voordeel met zich mee; Waar het vroeger bijna onmogelijk was om een 'gewist' gefragmenteerd bestand terug te halen kan nu gewoon beroep gedaan worden op de File Allocation Table. Het wisproces van FAT32 wiste in het verleden immers de posities van de clusters in de FAT. Zoals uit, het in vorige hoofdstukken beschreven onderzoek, gebleken is, wordt de File Allocation Table bij exFAT echter niet meer aangeraakt wanneer een bestand 'gewist' wordt.

Twee verschillende methoden worden van elkaar onderscheden:

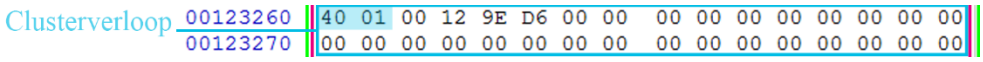
8.2.8.1 Optie 1: Een 'gewist', niet-gefragmenteerd, bestand terughalen

a. ***Op zoek gaan naar de waarde 0x05 (unallocated main entry (0x85)) in de map waar het bestand zich bevindt;***



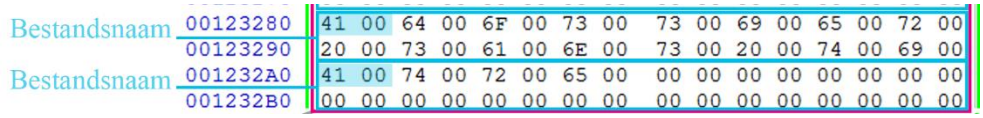
Deze afbeelding illustreert een main entry in een directory in hexadecimale weergave in winhex.

b. ***Op zoek gaan naar de sub entry met waarde 0x40 (is de 'gewiste' positietoewijzingsentry (0xC0)). In deze entry bevinden zich de startpositie van de eerste cluster en de bestandsgrootte van het bestand. Op basis van deze waarden kan het aantal clusters (clustersegment) berekend worden;***



Deze afbeelding illustreert een gewiste clusterverloopentry/positietoewijzingsentry in een directory in hexadecimale weergave in winhex.

c. ***Bij de subentry's met waarden 0x41 (is de 'gewiste' bestandsnaamentry (0xC1)) op zoek gaan naar de bestandsnaam. Dit om te controleren of het juiste bestand teruggehaald wordt;***



Deze afbeelding illustreert een gewiste bestandsnaamentry in een directory in hexadecimale weergave in winhex.

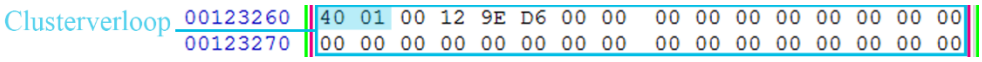
d. ***De data van start- tot en met eindcluster van de harde schijf halen.***

Een uitgebreid van het terughalen van een niet-gefragmenteerd bestand staat beschreven in het hoofdstuk 'casus'.

8.2.8.2 Optie 2: Een 'gewist', gefragmenteerd bestand, terughalen

a. Idem aan optie 1;

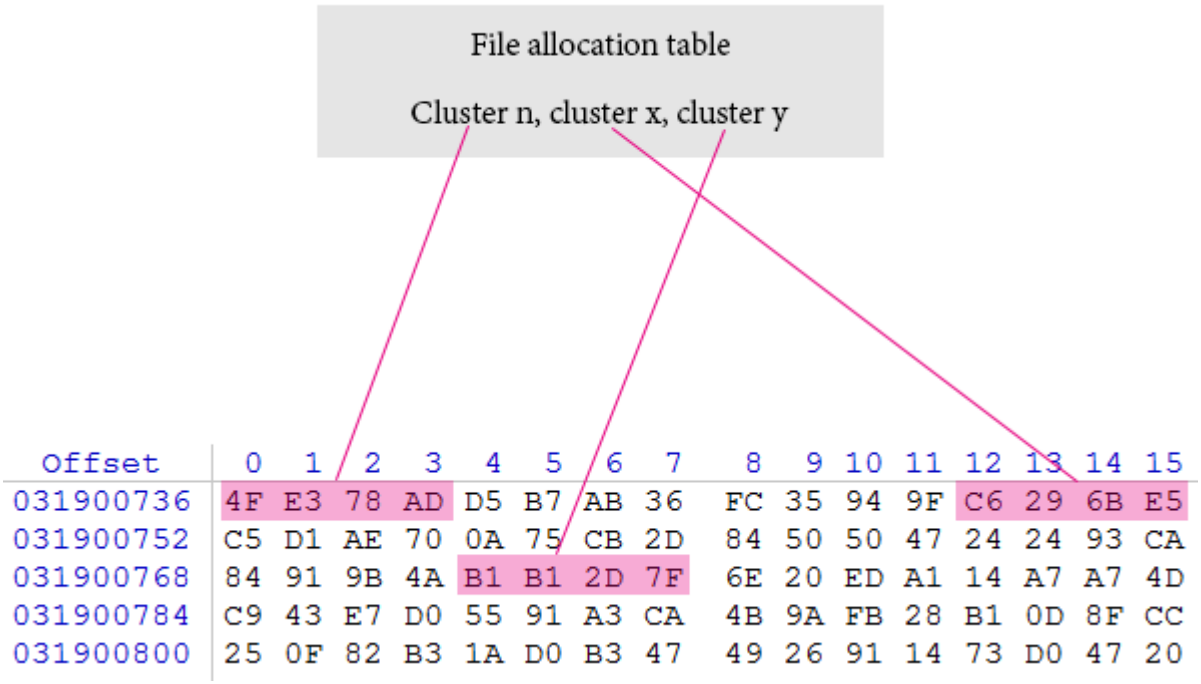
b. Op zoek gaan naar de sub entry met waarde 0x40 (is de 'gewiste' positietoewijzingsentry (0xC0)). In deze entry bevinden zich de startpositie van de eerste cluster en de bestandsgrootte van het bestand;



Deze afbeelding illustreert een gewiste clusterverloopentry/positietoewijzingsentry in een directory in hexadecimale weergave in winhex.

c. Idem aan optie 1;

d. ***Op zoek gaan naar de waarde van de startpositie van bovengenoemde sub entry. Deze startpositie dient vervolgens opgezocht te worden in de File Allocation Table. Vervolgens dient de File Allocation Table gebruikt te worden om de posities van de andere clusters terug te vinden. Elke cluster zal telkens naar de volgende cluster verwijzen tot de waarde FF FF FF FF getoond wordt. Het is mogelijk dat de oude cluster reeds door een nieuw bestand overschreven werd. Dit kan gemakkelijk opgezocht worden in de bitmap. Wanneer de oude cluster in de bitmap de waarde 'gebruikt' heeft, dan wil dit zeggen dat de oude cluster overschreven werd. Enkel de slack space zal dan nog gerecupereerd kunnen worden.***



Deze afbeelding illustreert de plaatsing van een gefragmenteerd bestand op een harde schijf met behulp van de File Allocation Table. Aan de hand van de File Allocation Table kunnen gefragmenteerde bestanden hersteld worden.

Een uitgebreid van het terughalen van een gefragmenteerd bestand staat beschreven in het hoofdstuk ‘casus’.

9 CASUS

Na een inval in een woning worden enkele personen gearresteerd in het kader van terrorisme. Enkele bewijsstukken zorgden ervoor dat speurders de verdachten in de woning op het spoor kwamen. Na ondervragingen blijkt dat de verdachten het brein zijn van een terroristische vereniging. Via anonieme bronnen is gebleken dat deze verdachten voor hun arrestatie een nieuwe aanslag beraamd hebben die zeer binnenkort zal plaatsvinden. De anonieme bron kan de verdachten beschrijven maar heeft geen idee waar de aanslag plaats zou kunnen vinden. In het huis van de organisatoren van de aanslag werd een usb-stick gevonden. Deze usb-stick bevat op het eerste zicht nutteloze informatie. De onderzoekers krijgen de stick in handen voor verdere analyse. Op vraag van de cel terrorismebestrijding heeft het vinden van mogelijke aanwijzingen momenteel prioriteit ten aanzien van bewijslast. Het doel is immers eerst om de aanslag te voorkomen. Het verzamelen van bewijslast zal pas in een verder stadium van het onderzoek noodzakelijk zijn.

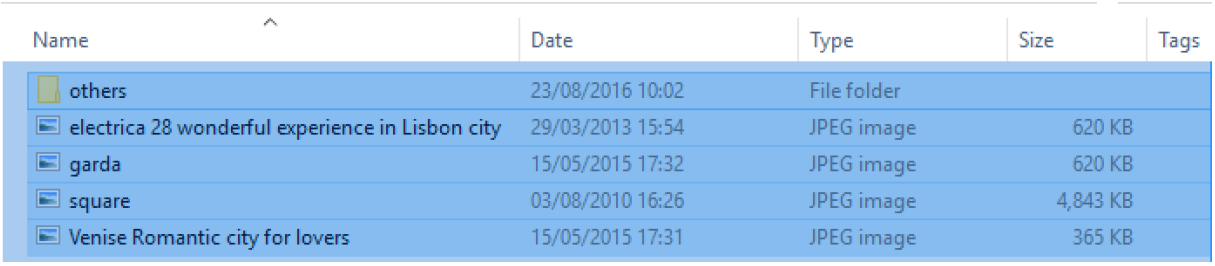
Om aan te tonen dat alle informatie handmatig verkregen kan worden, zonder gebruik van een betalend programma, zullen alle processen uitgevoerd door linux-commando's. Voor het uitvoeren van bepaalde commando's wordt gebruik gemaakt van het vrij verkrijgbare programma 'the sleuthkit' dat draait op het besturingssysteem linux.

9.1 Analyse van de gegevensdrager

Bij de start van een onderzoek dient een analyse gemaakt te worden van de gegevensdrager waarop elektronische media zich bevinden. Het resultaat van deze analyse verwacht voornamelijk een beeld te scheppen over het aantal partities op de gegevensdrager en het gebruikte bestandssysteem.

Aan de hand van bepaalde onderzoekselementen, zoals de start van de datazone, kunnen dan weer andere elementen teruggevonden worden. Voor het onderzoek wordt het MBR en het VBR geanalyseerd met het oog op het terugvinden van het gebruikte besturingssysteem, het aantal partities, en het gebruikte bestandssysteem.

Wanneer de schijf geopend wordt zijn op het eerste zicht enkel volgende bestanden waar te nemen:



Deze afbeelding toont de inhoud van usb-stick in Windows.

9.2 Technische onderzoekselementen

Om het bewijsmateriaal te kunnen verzamelen en om de bewerkingen van/met bestanden te kunnen begrijpen dienen enkele technische onderzoekselementen teruggevonden te worden. Deze elementen zullen belangrijke informatie bevatten over de bestanden en waar zij terug te vinden zijn. Over volgende elementen dient men zeker te beschikken:

9.2.1 MBR

In dit werk wordt bewust gebruik gemaakt van een usb-stick met één partitie met daarop het bestandssysteem exFAT. Hierdoor bevat het casusvoorbeeld geen MBR. De werking van het MBR is in andere werken reeds uitvoerig besproken, en zou reeds gekend moeten zijn voor alle andere bestandssystemen. Het wordt daarom ook niet in de scope van dit werk besproken.

9.2.2 VBR

Elke partitie bevat een VBR. Dit record bevat namelijk informatie over onder andere het aantal partities en de gebruikte bestandssystemen.

```
00000000  EB 76 90 45 58 46 41 54  20 20 20 00 00 00 00 00
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000040  80 00 00 00 00 00 00 00  00 E8 07 00 00 00 00 00
00000050  80 00 00 00 00 02 00 00  80 02 00 00 B0 FC 00 00
00000060  06 00 00 00 79 F4 37 CE  00 01 00 00 09 03 01 80
00000070  04 00 00 00 00 00 00 00  33 C9 8E D1 8E C1 8E D9
00000080  BC D0 7B BD 00 7C 88 16  6F 7C B4 41 BB AA 55 CD
00000090  13 72 69 81 FB 55 AA 75  63 F6 C1 01 74 5E FE 06
000000A0  02 7C 66 50 B0 65 E8 A6  00 66 58 66 B8 01 00 00
000000B0  00 8A 0E 6D 7C 66 D3 E0  66 89 46 E8 66 B8 01 00
000000C0  00 00 8A 0E 6C 7C 66 D3  E0 66 89 46 D8 66 A1 40
000000D0  7C 66 40 BB 00 7E B9 01  00 66 50 E8 41 00 66 58
000000E0  66 40 BB 00 80 B9 01 00  E8 34 00 66 50 B0 78 E8
000000F0  5D 00 66 58 E9 09 01 A0  FC 7D EB 05 A0 FB 7D EB
00000100  00 B4 7D 8B F0 AC 98 40  74 0C 48 74 0E B4 0E BB
00000110  07 00 CD 10 EB EF A0 FD  7D EB E6 CD 16 CD 19 66
00000120  60 66 6A 00 66 50 06 53  66 68 10 00 01 00 B4 42
00000130  B2 80 8A 16 6F 7C 8B F4  CD 13 66 58 66 58 66 58
00000140  66 58 66 61 72 B1 03 5E  D8 66 40 49 75 D1 C3 66
00000150  60 B4 0E BB 07 00 B9 01  00 CD 10 66 61 C3 42 00
00000160  4F 00 4F 00 54 00 4D 00  47 00 52 00 0D 0A 52 65
00000170  6D 6F 76 65 20 64 69 73  6B 73 20 6F 72 20 6F 74
00000180  68 65 72 20 6D 65 64 69  61 2E FF 0D 0A 44 69 73
```

Deze afbeelding illustreert het VBR uit het voorbeeldbestand in hexadecimale weergave met winhex..

Bij de analyse van het VBR kunnen onmiddellijk de FAT, de startcluster van de datazone, de startcluster van de root directory en met wat rekenwerk ook de bitmap teruggevonden worden. Deze elementen zullen in de onderstaande hoofdstukken aan de hand van een voorbeeldbestand onderzocht worden.

Het VBR kan ook uitgelezen worden met een linux-commando door middel van het programma ‘the sleuthkit’. Volgende commando geeft onder andere het aantal sectoren per cluster weer.

fsstat *naamvanhetvolume*.dd

Voor het gegeven voorbeeld geeft dit volgend resultaat:

```
tdg@tdg-VirtualBox:~/Downloads$ fsstat exFAT.dd
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: exFAT

Volume Serial Number: ce37-f479
Volume Label (from root directory): demoexFAT
File System Name (from MBR): EXFAT
File System Revision: 1.0
Partition Offset: 128
Number of FATs: 1

File System Layout (in sectors):
Range: 0 - 518143
* Reserved: 0 - 127
** Volume Boot Record (VBR): 0 - 11
*** Boot Sector (MBR): 0
** Backup Volume Boot Record (VBR): 12 - 23
*** Backup Boot Sector (MBR): 12
** FAT alignment space: 24 - 127
* FAT 1: 128 - 639
* Data Area: 640 - 518143
** Cluster Heap: 640 - 518143
*** Root Directory: 672 - 679

METADATA INFORMATION
--------------------------------------------
Metadata Layout (in virtual inodes):
Range: 2 - 8280069
* Root Directory: 2

CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 4096
Cluster Range: 2 - 64689
```

9.2.3 Sectoren per cluster

In het VBR kan teruggevonden worden hoeveel sectoren elke cluster op de drager telt. Deze informatie is cruciaal voor het terugvinden van bepaalde onderzoekselementen en kan teruggevonden worden op positie 0x6D in het VBR.

00000060	06	00	00	00	79	F4	37	CE	00	01	00	00	09	03	01	80

Deze afbeelding toont de waarde van offset 0x6D in het VBR van het casusvoorbeeld in hexadecimale weergave met winhex

Deze effectieve waarde wordt getoond als macht van opgegeven waarde. 2 tot de derdemacht is gelijk aan 8. Het aantal sectoren op per cluster op de gegevensdrager bedraagt dus 8.

De byte net voor offset 0x6D beschrijft het aantal bytes per sector. Uitgerekend bedraagt dit 512. Aangezien elke cluster 8 sectoren bevat, en elke sector uit 512 bytes bestaat, bestaat elke cluster op deze gegevensdrager uit 4096 bytes.

9.2.4 FAT

De File Allocation Table zal belangrijke informatie bevatten over de clusterlocaties van gefragmenteerde bestanden.

De positie van de File Allocation Table staat beschreven op offset 0x50 en is 4 bytes groot. Offset 0x54 beschrijft de grootte van de FAT in sectoren. In dit voorbeeld geeft dit volgend resultaat:

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	EB	76	90	45	58	46	41	54	20	20	20	00	00	00	00	00
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	80	00	00	00	00	00	00	00	00	E8	07	00	00	00	00	00
00000050	80	00	00	00	00	02	00	00	80	02	00	00	B0	FC	00	00
00000060	06	00	00	00	79	F4	37	CE	00	01	00	00	09	03	01	80

Deze afbeelding toont de waarde van offset 0x50 en 0x54 in het VBR van het casusvoorbeeld in hexadecimale weergave met winhex

Uit het resultaat blijkt dat de FAT zich op sectorpositie 0x80 (= 128) bevindt. De eerste 4 bytes op positie 0x50 beschrijven namelijk de startpositie van de FAT in sectoren, relatief tot de start

van de partitie. Deze startpositie is het clusternummer en niet de eigenlijke offset van de FAT. De offset kan teruggevonden worden door de startpositie te vermenigvuldigen met het aantal sectoren. Aangezien de FAT zich niet in de datazone bevindt zal dit getal steeds 512 bedragen.

In dit geval geeft dit volgend resultaat:
128 (sector) * 512 (bytes per sector) = 65 536 = de eigenlijke startpositie van de FAT in bytes.

Zoals op onderstaande afbeelding te zien is start de FAT in dit voorbeeld effectief op offset 65 536.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
000065536	F8	FF	FF	FF	FF	FF	FF	FF	03	00	00	00	FF	FF	FF	FF
000065552	05	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	00	00	00	00
000065568	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Deze afbeelding toont de FAT van het casusvoorbeeld in hexadecimale weergave met winhex.

9.2.5 Datazone

Ook de start van de datazone wordt aangegeven in het VBR. De startcluster van dit element dient geweten te zijn om andere onderzoekselementen te kunnen berekenen en kan teruggevonden worden op offset 0x58.

00000050	80	00	00	00	00	02	00	00	80	02	00	00	B0	FC	00	00

Deze afbeelding toont de waarde van offset 0x58 in het VBR van het casusvoorbeeld in hexadecimale weergave met winhex

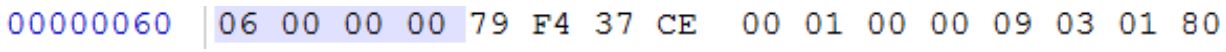
Uit het resultaat blijkt dat de start op positie 00 00 02 80 (= 640) bevindt. De 4 bytes op positie 0x58 beschrijven namelijk de startpositie van de datazone in sectoren. Ook hier wordt de waarde van het clusternummer getoond waardoor, na omrekening pas de eigenlijke offset verkregen wordt.

Omrekening:
640 * 512 = 327 680 = de eigenlijke startpositie van de datazone in bytes.

9.2.6 Root directory

In de root directory is tal van informatie terug te vinden. Niet alleen bestandsnamen, startlocaties, en bestandsgroottes, enz. maar ook verplaatste en ‘gewiste’ bestandsnamen staan erin genoteerd. Het is vooral de laatste informatie die in deze casus van grote rol zal spelen in het onderzoek.

De eerste **cluster** van de root directory, het belangrijkste onderzoekselement in exFAT, kan eveneens in het VBR teruggevonden worden. Offset 0x60 beschrijft de start van de eerste cluster die typisch zal starten met de systeementry’s om vervolgens de mappen en bestanden te beschrijven.



Deze afbeelding toont de waarde van offset 0x60 in het VBR van het casusvoorbeeld in hexadecimale weergave met winhex

Uit het resultaat blijkt dat de root directory zich op positie 0x06 (= 6) bevindt. Met behulp van onderstaande formule kan de start van de datazone teruggevonden worden.

*(Start datazone * 512) + ((1e cluster root directory - 2) * bytes per cluster)*

Omrekening:

(640 * 512) + ((6 – 2) * 4096) = 344 064 = eigenlijke offset van de root directory in bytes.

000344064	83	09	64	00	65	00	6D	00	6F	00	65	00	78	00	46	00
000344080	41	00	54	00	00	00	00	00	00	00	00	00	00	00	00	00
000344096	81	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000344112	00	00	00	00	02	00	00	00	96	1F	00	00	00	00	00	00
000344128	82	00	00	00	0D	D3	19	E6	00	00	00	00	00	00	00	00
000344144	00	00	00	00	04	00	00	00	CC	16	00	00	00	00	00	00
000344160	85	03	29	ED	16	00	00	00	80	4E	17	49	80	4E	17	49
000344176	80	4E	17	49	37	37	88	88	88	00	00	00	00	00	00	00
000344192	C0	03	00	19	B8	FF	00	00	00	10	00	00	00	00	00	00
000344208	00	00	00	00	07	00	00	00	00	10	00	00	00	00	00	00
000344224	C1	00	53	00	79	00	73	00	74	00	65	00	6D	00	20	00
000344240	56	00	6F	00	6C	00	75	00	6D	00	65	00	20	00	49	00
000344256	C1	00	6E	00	66	00	6F	00	72	00	6D	00	61	00	74	00
000344272	69	00	6F	00	6E	00	00	00	00	00	00	00	00	00	00	00
000344288	85	02	2D	94	16	00	00	00	84	4E	17	49	84	4E	17	49
000344304	84	4E	17	49	B4	B4	88	88	88	00	00	00	00	00	00	00
000344320	C0	03	00	0C	29	B3	00	00	00	10	00	00	00	00	00	00
000344336	00	00	00	00	09	00	00	00	00	10	00	00	00	00	00	00
000344352	C1	00	24	00	52	00	45	00	43	00	59	00	43	00	4C	00
000344368	45	00	2E	00	42	00	49	00	4E	00	00	00	00	00	00	00
000344384	05	02	B7	82	20	00	00	00	1C	50	17	49	D3	86	7D	42
000344400	1C	50	17	49	32	00	88	88	88	00	00	00	00	00	00	00
000344416	40	03	00	0A	8D	D1	00	00	5A	AE	09	00	00	00	00	00
000344432	00	00	00	00	0B	00	00	00	5A	AE	09	00	00	00	00	00
000344448	41	00	74	00	72	00	61	00	6D	00	32	00	38	00	2E	00
000344464	6A	00	70	00	67	00	00	00	00	00	00	00	00	00	00	00

Deze afbeelding toont de root directory van het casusvoorbeeld in hexadecimale weergave met winhex.

Aan de hand van de entry's kan ook worden opgemaakt dat meerdere bestanden op harde schijf aanwezig zijn. Met betrekking tot dit onderzoek gaat de aandacht vooral uit naar de entry's die beginnen met 0x05, 0x40 en 0x41 . Deze entry's bevatten immers gegevens over de gewiste of verplaatste bestanden. Ook blijkt uit het onderzoek, dat op het eerste zicht niet meer aanwezige bestanden, toch nog zichtbaar zijn wat betreft de bestandsnaam en startlocaties.

Met volgend linux-commando kan informatie getoond worden over bestanden en mappen in de root directory:

```
fls naamvanhetvolume.dd
```

Deze afbeelding toont hoe "the sleuthkit" bestanden en mappen weergeeft na uitvoering van het "fls"-commando.

Het uitvoeren van dit commando geeft volgend resultaat:

```
tdg@tdg-VirtualBox:~/Downloads$ fls exFAT.dd
r/r 515:        demoexFAT (Volume Label Entry)
r/r 516:        $ALLOC_BITMAP
r/r 517:        $UPCASE_TABLE
d/d 518:        System Volume Information
d/d 522:        $RECYCLE.BIN
r/r * 525:      tram28.jpg
r/r 528:        square.jpg
r/r * 531:      dns.png
d/d 534:        others
r/r 537:        garda.jpg
r/r * 540:      italy.jpg
r/r * 543:      secret.mov
r/r * 546:      lake.jpg
r/r * 549:      empty.dd
r/r 552:        electrica 28 wonderful experience in Lisbon city.jpg
r/r 558:        Venise Romantic city for lovers.jpg
r/r * 563:      trainticket.pdf
r/r * 566:      The Hague.png
r/r * 569:      target_earth.png
v/v 8280067:    $MBR
v/v 8280068:    $FAT1
d/d 8280069:    $OrphanFiles
```

Deze afbeelding illustreert het resultaat van het "fls"-commando.

Bestand- en mapentry', aangegeven met een "*" hebben de status "niet-toegewezen" (het equivalent van het zichtbare 0x05, 0x40, 0x41 van de hierboven getoonde afbeelding). Het nummer is een uniek nummer, dat door The Sleuthkit gegenereerd wordt.

9.2.7 Bitmap

Een ander belangrijk onderzoekselement is de bitmap. De bitmap geeft namelijk aan of een cluster al dan niet in gebruik is. Een niet-toegewezen cluster kan informatie bevatten van 'gewiste' of verplaatste bestanden.

In de bitmapentry kan de start van de bitmap teruggevonden worden.

```
000344096  81 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
000344112  00 00 00 00 02 00 00 00   96 1F 00 00 00 00 00 00
```

Deze afbeelding toont de bitmapentry in de root directory van het casusvoorbeeld in hexadecimale weergave met winhex

Positie 20 geeft in 4 bytes de startcluster (0x02 = 2) aan van de bitmap. De genoteerde waarde is echter het clusternummer. Om te weten te komen op welke offset deze cluster start, kan volgende formule gebruikt worden:

*(Start datazone * bytes/sector) + ((1e cluster bitmap – 2) * bytes/cluster)*

Bij het toepassen van deze formule kan opgemaakt worden dat de bitmap start op positie:

(640 * 512) + ((2 – 2) * 4096) = 327 680 (in bytes).

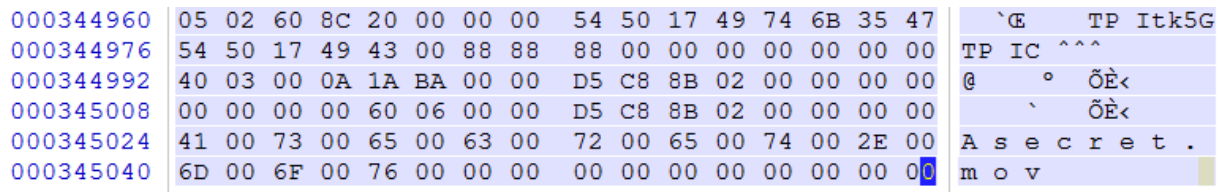
000327680	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327696	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327712	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327728	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327744	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327760	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327776	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327792	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327808	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327824	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327840	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	7F	C0	FF	FF	FF
000327856	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000327872	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	3F	00	00	00	00
000327888	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Deze afbeelding toont de bitmap van het casusvoorbeeld in decimale weergave met winhex De offsets zijn in decimale weergave.

9.3 Recuperatie van inhoud

9.3.1 Niet-gefragmenteerde recuperatie

Zoals in vorige hoofdstukken beschreven, is het terughalen van een niet-gefragmenteerd bestand niet zo moeilijk. Omdat, bij het wissen of verplaatsen van een bestand, enkel de clusters vrijgegeven worden in de bitmap en de entrywaarden aangepast worden (naar 0x05, 0x40, 0x41) in de directory, hoeven enkel de startlocatie en eindlocatie van het bestand bepaald te worden. In de analyse van gegevensdrager werden de startlocatie en grootte van het bestand reeds bepaald. Door deze aan volgend commando mee te geven, kan een bestand teruggehaald worden.



Deze afbeelding toont de file entry (bestandsentry) van het gewiste bestand "secret.mov" in de root directory. Presentatie in hexadecimale weergave met winhex. De Offsets zijn in decimale weergave.

Aan de hand van de waarden in de root directory kan opgemaakt worden dat het bestand met naam secret.mov (zie 0x41 voor bestandsnaam) een gewist bestand is. Uit de waarden in allocation entry (0x40) kan vastgesteld worden dat het om een niet-gefragmenteerd bestand gaat waarvan de startpositie zich bevindt op de 20ste positie, bestaande uit 4 bytes. De waarde uit het voorbeeld (little endian) bedraagt 0x00000660 (= 1632). Op de 24ste positie is de bestandsgrootte in bytes beschreven in 8 bytes (little endian). Het voorbeeld heeft volgens deze beschrijving een grootte van 42715349 bytes.

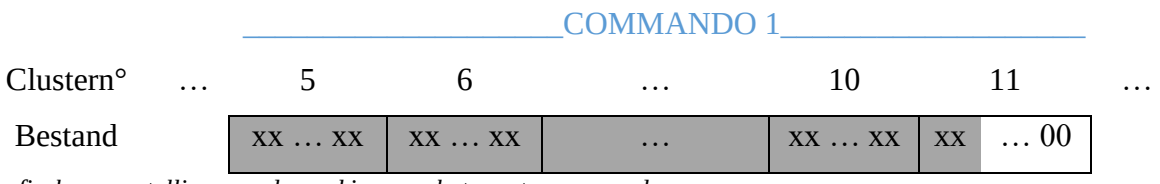
Volgend linux-commando maakt een bruikbare kopie van het 'gewiste/verplaatste' bestand naar opgegeven locatie:

```
dd if=teonderzoekenschijf.dd bs=sectorgrootte skip=$(((offset-2)*#sectoren/cluster) + dataheap)) count=$ ((Bestandsgrootte/sectorgrootte)+1)
| dd bs=1 count=bestandsgrootte of=outputfile.extensie
```

Met het commando "dd" kunnen bestanden geconverteerd of gekopieerd worden. Het eerste deel van de formule (tot voor het pipe-teken "|") gaat op de schijf op zoek naar een bepaald

bestand. Met het commando “skip”, wordt de exacte startpositie meegegeven (de eerdere data wordt genegeerd). Vervolgens wordt met count het aantal te kopiëren clusters meegegeven (minimum 1). Door de uitvoering van het commando zullen alle clusters die data van het bestand bevatten gekopieerd worden. Aangezien een bestand zeer zelden ook de laatste cluster volledig zal opvullen (zie hoofdstuk slack space), zal vaak een teveel aan data gekopieerd worden.

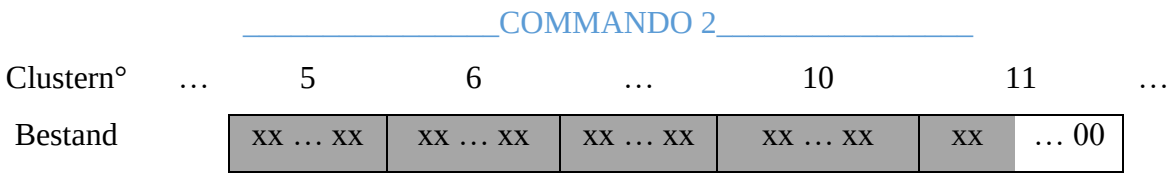
Grafisch voorbeeld commando 1:



Grafische voorstelling van de werking van het eerste commando.

Het tweede deel van het commando, het deel na het pipe-teken, zorgt ervoor dat dit teveel buiten rekening gehouden wordt door niet per cluster te kopiëren maar per byte en vervolgens ook de exacte grootte van het bestand op te geven.

Grafisch voorbeeld commando 2:



Grafische voorstelling van de werking van het tweede commando.

Wanneer de formule toegepast wordt op het niet-gefragmenteerde bestand, wordt het volgende bekomen:

```
$ dd if=exFAT.dd bs=512 skip=$((((1632-2)*8)+640)) count=$(((42715349/512)+1))
| dd bs=1 count=42715349 of=recup1.mov
```

Deze afbeeldingen tonen de ingevoerde waarden om het niet-gefragmenteerde bestand handmatig met "The Sleuthkit" te recupereren.

Dankzij het commando kan een 'gewist' filmpje teruggevonden worden op de gegevensdrager. Op dit filmpje is te zien hoe 4 verdachten een bom in elkaar plaatsen. Deze beelden tonen duidelijk de gezichten van de 4 mannen. Op basis van deze elementen kan onder andere een signalering verspreidt worden.

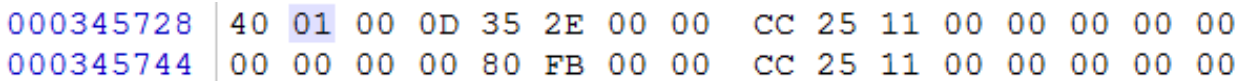
Afbeelding van de niet-gefragmenteerde; gewiste, video die gerecupereerd kon worden aan de hand van het linux-commando.

Tot slot kan in de bitmap nog nagegaan worden of clusters van het oude bestand intussen reeds aan nieuwe bestanden zijn toegewezen. Dit zou extra bewijswaarde kunnen opleveren maar is voor deze casus niet noodzakelijk.

9.3.2 Gefragmenteerde recuperatie

Net zoals bij een niet-gefragmenteerd bestand kan informatie gehaald worden uit de main entry. De startlocatie zal ook in dit geval ook van groot belang zijn. Er kan echter geen gebruik meer gemaakt worden van de bestandsgrootte voor het bepalen van de eindlocatie van het bestand. Aangezien het bestand gefragmenteerd is zullen de clusters die de data van het bestand bevatten niet meer chronologisch volgen. In dit geval zal dus beroep gedaan moeten worden op de File Allocation table die de locaties van de clusters bijhoudt.

In de allocation entry (in de root directory) kan de startcluster van het bestand gevonden worden. Op basis van deze entry kan tevens ook opgemaakt worden dat het bestand gefragmenteerd is (zie waarde 40 ***01***). Aan de hand van de startcluster van het bestand kan ook de beschrijvende 4-byte-waarde in de FAT teruggevonden worden.



Deze afbeelding toont de allocation entry van het gewiste bestand "The Hague.png" in de root directory. Presentatie in hexadecimale weergave met winhex.

Formule voor het terugvinden van de beschrijving van een gewenste cluster in FAT:

*(Start van FAT * sectorgrootte) + (Clustern° * 4)*

Ingevuld met de verkregen waarden geeft dit volgend resultaat:
126 * 512 + 64 384 * 4 = 323072 (byte offset).

Er dient handmatig opgezocht te worden welke clusters niet continu na elkaar geplaatst werden. Deze handeling is erg tijdrovend. Het doel is om te vinden welke clusters elkaar in de FAT-ketening niet opvolgen. De laatste 4 bytes voor "de sprong" zullen het clusternummer van het einde van het eerste fragment bevatten, de eerste 4 bytes na de sprong bevatten de eerste clusternummer van het tweede fragment. Vervolgens dient naar de volgende sprongen gezocht te worden tot het einde van het bestand wordt aangegeven met de waarde "FF FF FF FF".

In dit voorbeeld is er sprake van 2 fragmenten. Het eerste fragment start met clusternummer 64 385, het einde van het eerste fragment is op clusternummer 64 640 (waarde 00 00 FC 7F). De start van het tweede fragment is de waarde die in de laatste clusterverwijzing meegegeven wordt, de waarde 00 00 2F 8B (waarde geselecteerd in het blauw). Deze waarde verwijst naar de startcluster van het tweede fragment (12 171). Na opzoeking, door middel van de vorige formule (cf. “Formule voor het terugvinden van de beschrijving van een gewenste cluster in FAT”), wordt de waarde 114 220 bekomen. In de FAT kan vervolgens vanaf deze offset de FAT-ketening opnieuw gevolgd worden tot eindcluster 12 189, aangegeven met de waarde FF FF FF FF.

Eerste fragment:

```
000323056 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000323072 81 FB 00 00 82 FB 00 00  83 FB 00 00 84 FB 00 00
000323088 85 FB 00 00 86 FB 00 00  87 FB 00 00 88 FB 00 00
                                …
000324016 6D FC 00 00 6E FC 00 00  6F FC 00 00 70 FC 00 00
000324032 71 FC 00 00 72 FC 00 00  73 FC 00 00 74 FC 00 00
000324048 75 FC 00 00 76 FC 00 00  77 FC 00 00 78 FC 00 00
000324064 79 FC 00 00 7A FC 00 00  7B FC 00 00 7C FC 00 00
000324080 7D FC 00 00 7E FC 00 00  7F FC 00 00 8B 2F 00 00
000324096 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Deze afbeelding toont het eerste fragment van de clusterverwijzing van het “The Hague.png” in de FAT. Hexadecimale weergave in winhex.

Tweede fragment:

```
000114208 00 00 00 00 00 00 00 00  00 00 00 00 8C 2F 00 00
000114224 8D 2F 00 00 8E 2F 00 00  8F 2F 00 00 90 2F 00 00
000114240 91 2F 00 00 92 2F 00 00  93 2F 00 00 94 2F 00 00
000114256 95 2F 00 00 96 2F 00 00  97 2F 00 00 98 2F 00 00
000114272 99 2F 00 00 9A 2F 00 00  9B 2F 00 00 9C 2F 00 00
000114288 9D 2F 00 00 FF FF FF FF  00 00 00 00 00 00 00 00
```

Deze afbeelding toont het tweede fragment van de clusterverwijzing van het “The Hague.png” in de FAT. Hexadecimale weergave in winhex.

Met volgende linux commando's kunnen de twee fragmenten aan elkaar geplaatst worden om zo handmatig het bestand te recupereren.

1. dd if=teonderzoekenschijf.dd bs=sectorgrootte skip=$(dataheap + (1e offset 1e fragment -2)*#clustergrootte in sectoren)) count=$ ((segmentgrootte in clusters * clustergrootte in sectoren)) of=deel1.dd

2. dd if=teonderzoekenschijf.dd bs=sectorgrootte skip=$(dataheap + (1e offset 2e fragment-2)*#clustergrootte in sectoren)) count=$ ((segmentgrootte in clusters * clustergrootte in sectoren)) of=deel2.dd

3. dd if=deel2.dd >> deel1.dd

4. dd if=deel1.dd bs=1 count=bestandsgrootte of=teruggehaaldbestand.extensie

De eerste 1e stap dient per fragment herhaald te worden. De derde stap in dit voorbeeld voegt de aangemaakte stukken data samen. De vierde en laatste stap verwijderd net zoals bij het voorbeeld van de niet-gefragmenteerde bestanden de overbodige data uit de slack space.

Dit commando zoekt doormiddel van de FAT naar de gefragmenteerde data van het bestand, plaats deze opnieuw in de juiste volgorde aan elkaar en bouwt een kopie op van het bestand.

```
tdg@tdg-VirtualBox:~/Downloads$ dd if=exFAT.dd bs=512 skip=$((640+(64382*8)
)) count=$((256*8)) of=part1.dd
2048+0 records gelezen
2048+0 records geschreven
1048576 bytes (1,0 MB, 1,0 MiB) copied, 0,00642216 s, 163 MB/s
tdg@tdg-VirtualBox:~/Downloads$ dd if=exFAT.dd bs=512 skip=$((640+(12169*8)
)) count=$((19*8)) of=part2.dd
152+0 records gelezen
152+0 records geschreven
77824 bytes (78 kB, 76 KiB) copied, 0,0003405 s, 229 MB/s
tdg@tdg-VirtualBox:~/Downloads$ dd if=part2.dd >> part1.dd
152+0 records gelezen
152+0 records geschreven
77824 bytes (78 kB, 76 KiB) copied, 0,000714448 s, 109 MB/s
tdg@tdg-VirtualBox:~/Downloads$ ls -l
totaal 771712
-rw-rw-r-- 1 tdg tdg 216371200 aug 24 21:38 deel1.dd
-rw-rw-r-- 1 tdg tdg 215117824 aug 24 21:37 deel2.dd
-rw-rw-r-- 1 tdg tdg 265289728 aug 24 11:59 exFAT.dd
-rw-rw-r-- 1 tdg tdg  42715648 aug 24 17:53 outputfile.mov
-rw-rw-r-- 1 tdg tdg   1126400 aug 25 21:05 part1.dd
-rw-rw-r-- 1 tdg tdg     77824 aug 25 21:05 part2.dd
-rw-rw-r-- 1 tdg tdg  42715349 aug 24 18:44 recup1.mov
-rw-rw-r-- 1 tdg tdg   5677683 aug 25 17:50 recupWithSleuthkit.jpg
-rw-rw-r-- 1 tdg tdg   1123788 aug 24 21:40 teruggehaaldbestand.png
tdg@tdg-VirtualBox:~/Downloads$ dd if=part1.dd bs=1 count=1123788 of=recove
red.png
1123788+0 records gelezen
1123788+0 records geschreven
1123788 bytes (1,1 MB, 1,1 MiB) copied, 1,28339 s, 876 kB/s
tdg@tdg-VirtualBox:~/Downloads$ display recovered.png
```

1e commando

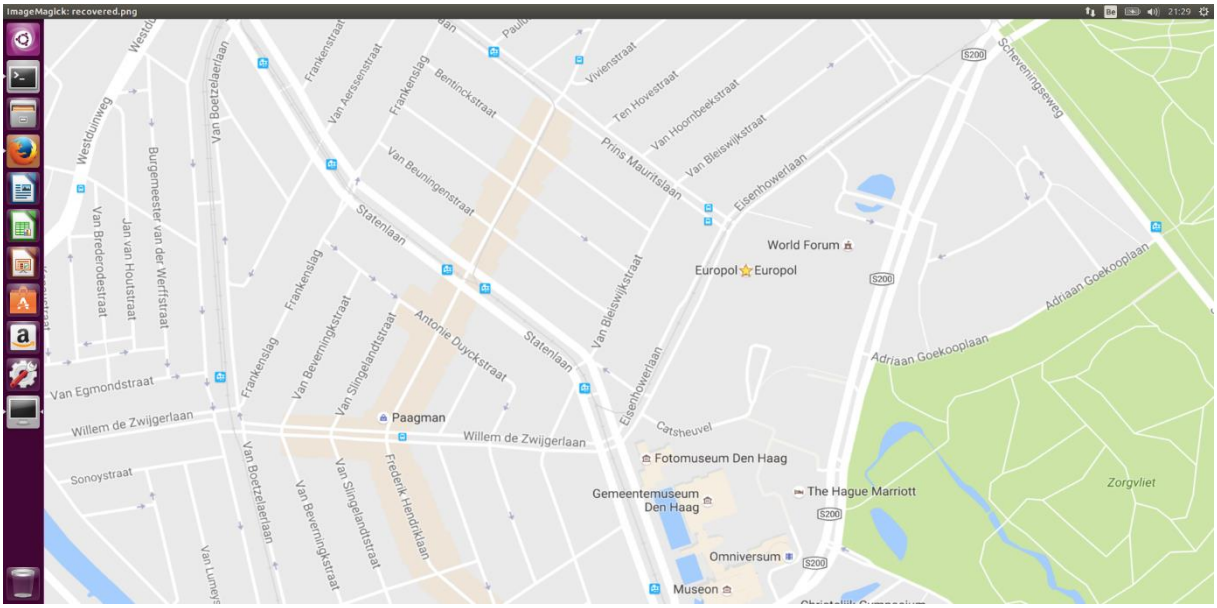
2e commando

3e commando

4e commando

Deze afbeelding toont de ingevoerde commando's en waarden om het niet-gefragmenteerde bestand handmatig met "The Sleuthkit" te recupereren.

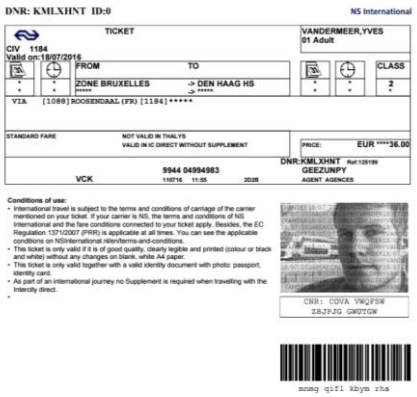
Na het opbouwen van het “gewiste” en gefragmenteerde bestand, wordt de inhoud van de foto duidelijk. Op de foto is een gebouw van Europol waar te nemen.



Afbeelding van de gefragmenteerde, gewiste, foto die gerecupereerd kon worden aan de hand van de linux-commando’s.

9.3.3 Caseconclusie

Op de gegevensdrager werd een niet-gewist treinticket teruggevonden. Dit bestand kon zonder problemen bekeken worden. Op het treinticket richting Den Haag, stond een datum en de naam van de tickethouder, die verdacht wordt de daad in de toekomst te gaan plegen.



Afbeelding van het teruggehaalde pdf-bestand (treinticket) waarvan sprake is in de conclusie van de casus.

Vervolgens werd na recuperatie van een niet-gefragmenteerd, gewist, filmpje duidelijk, dat de daad hoogstwaarschijnlijk door middel van het gebruik van een bom gepleegd zal worden. Er zijn op dit filmpje ook 4 verdachten waar te nemen.

Tot slot kon ook het doel geïdentificeerd worden aan de hand van de recuperatie van een gefragmenteerd, gewist, bestand, met name het Europolgebouw in Den Haag.

10 BESTAANDE PROGRAMMA'S/TOOLS

10.1 Algemeen gebruik

De dienst Federal Computer Crime Unit maakt voornamelijk gebruik van de programma's "The Sleuthkit", "Autopsy", "X-Ways" en "Tyrhex". Alle beschreven programma's zijn in staat om de meest courante bestandssystemen te lezen, analyseren, manipuleren en meestal ook te interpreteren. Een aantal functies in de programma's vereisen niet altijd voorkennis van het type bestandssysteem, andere functies vereisen dan weer een uitgebreide kennis. Het onderscheid wordt gemaakt in de grafische weergave, de ondersteuning op besturingssystemen, de prijs en de mogelijkheden.

In onderstaande voorbeelden wordt telkens hetzelfde volume met dezelfde inhoud gebruikt. Volgende weergave van de root directory van het volume beschrijft de bewerkingen die de entry's in de directory ondergaan hebben.

$Bitmap	file, existing
$UpCase	file, existing
dns.png	file, prev. existing, data not necessarily intact
electrica 28 wonderful experie...	file, existing
empty.dd	file, prev. existing, data not necessarily intact
garda.jpg	file, existing
italy.jpg	file, renamed/moved, data not necessarily intact
lake.jpg	file, prev. existing, data not necessarily intact
secret.mov	file, prev. existing, data not necessarily intact
square.jpg	file, existing
target_earth.png	file, prev. existing, data not necessarily intact
The Hague.png	file, prev. existing, data not necessarily intact
trainticket.pdf	file, prev. existing, data not necessarily intact
tram28.jpg	file, renamed/moved, data not necessarily intact
Venise Romantic city for lover...	file, existing
Boot sector	file, virtual (for examination purposes)
FAT	file, virtual (for examination purposes)

Deze afbeelding illustreert de handelingen die de bestanden op de root directory ondergingen. Deze afbeelding werd gegenereerd met het programma X-ways forensics.

10.1.1 The Sleuthkit

The sleuthkit is een gratis, open-source, programma dat draait op het besturingssysteem Linux en MAC OS (in samenwerking met het programma “homebrew”). The sleuthkit is een krachtig programma dat werkt via de terminal. Met linux-commando’s kunnen in de terminal gevraagde elementen weergegeven worden.

Het gebrek aan het gebruik van een interface kan soms erg onhandig zijn. Ook de interpretatie van verplaatste en hernoemde bestanden wordt verkeerd weergegeven in het programma. Het krijgt standaard de term “deletet”, ook al kan het bestand gewoon verplaatst zijn. Hier moet zeer omzichtig mee omgesprongen worden bij interpretaties in processen-verbaal.

Zoals eerder aangegeven werd, is het tevens ook mogelijk om met bepaalde commando’s bestanden terug te halen. Voor deze handelingen is echter kennis m.b.t Linux en bestandssystemen vereist. Sommige functies zijn in staat om bestanden terug te halen al blijken de functies nog moeilijkheden te hebben met het terughalen van gefragmenteerde gewiste bestanden in exFAT.

Een grafische interface voor de clusterlijst kan getoond worden voor niet-gefragmenteerde bestanden maar werkt opnieuw niet voor gefragmenteerde bestanden. Het is tevens ook jammer dat niet getoond wordt of een bestand al dan niet gefragmenteerd is.

Het grote voordeel aan het programma is dat de software open-source en gratis is. Voor onderzoekers die over de nodige kennis van bestandssytemen beschikken, is de tool erg handig, het toont enkel de nodige elementen en staat de onderzoeker toe stap voor stap zelf de nodige onderzoeksdaden te stellen. Ook voor educatieve doeleinden is het gebruik van het programma aan te raden. Voor Windows-gebruikers of gebruikers die het gebrek aan interface een nadeel vinden, is Autopsy een goed alternatief voor The Sleuthkit.

Onderstaande foto toont een overzicht van hoe The Sleuthkit inhoud op een gegevensdrager weergeeft. De nummer is een uniek nummer, dat door het programma gegenereerd wordt en dat gebruikt kan worden om bepaalde acties uit te voeren. Bestanden met een "*"-teken zijn "niet-toegewezen" bestanden maar volgens de uitleg van de makers van het programma zijn het gewiste bestanden.

```
r/r 515:        demoexFAT (Volume Label Entry)
r/r 516:        $ALLOC_BITMAP
r/r 517:        $UPCASE_TABLE
d/d 518:        System Volume Information
d/d 522:        $RECYCLE.BIN
r/r * 525:      tram28.jpg
r/r 528:        square.jpg
r/r * 531:      dns.png
d/d 534:        others
r/r 537:        garda.jpg
r/r * 540:      italy.jpg
r/r * 543:      secret.mov
r/r * 546:      lake.jpg
r/r * 549:      empty.dd
r/r 552:        electrica 28 wonderful experience in Lisbon city.jpg
r/r 558:        Venise Romantic city for lovers.jpg
r/r * 563:      trainticket.pdf
r/r * 566:      The Hague.png
r/r * 569:      target_earth.png
v/v 8280067:    $MBR
v/v 8280068:    $FAT1
d/d 8280069:    $OrphanFiles
```

Deze foto geeft de bestanden weer die zich in de root directory bevinden, getoond met het programma The Sleuthkit.

```
usage: fls [-adDFlpruvV] [-f fstype] [-i imgtype] [-b dev_sector_size] [-m dir/] [-o imgoffset] [-z ZONE] [-s seconds]
image [images] [inode]
        If [inode] is not given, the root directory is used
        -a: Display "." and ".." entries
        -d: Display deleted entries only
        -D: Display only directories
        -F: Display only files
        -l: Display long version (like ls -l)
        -i imgtype: Format of image file (use '-i list' for supported types)
        -b dev_sector_size: The size (in bytes) of the device sectors
        -f fstype: File system type (use '-f list' for supported types)
        -m: Display output in mactime input format with
            dir/ as the actual mount point of the image
        -o imgoffset: Offset into image file (in sectors)
        -p: Display full path for each file
        -r: Recurse on directory entries
        -u: Display undeleted entries only
        -v: verbose output to stderr
        -V: Print version
        -z: Time zone of original machine (i.e. EST5EDT or GMT) (only useful with -l)
        -s seconds: Time skew of original machine (in seconds) (only useful with -l & -m)
```

Deze uitleg, gegenereerd met de hulpfunctie in Linux, geeft meer info over het commando. En legt uit dat "fls -d" enkel gewiste bestanden toont.

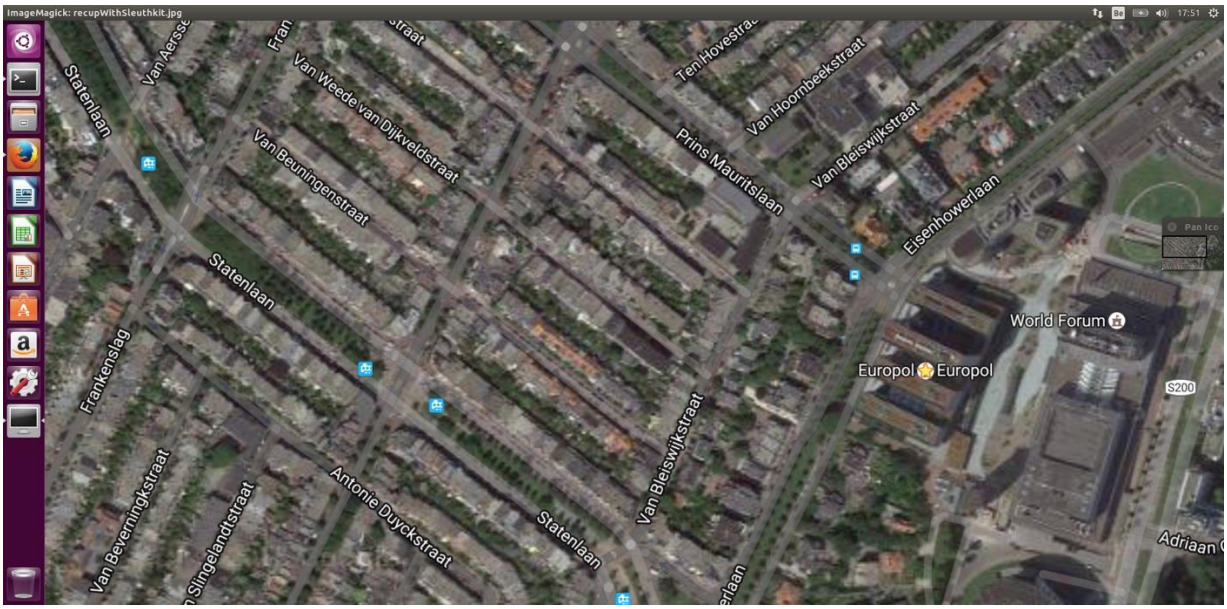
```
r/r * 525:    tram28.jpg
r/r * 531:    dns.png
r/r * 540:    italy.jpg
r/r * 543:    secret.mov
r/r * 546:    lake.jpg
r/r * 549:    empty.dd
r/r * 563:    trainticket.pdf
r/r * 566:    The Hague.png
r/r * 569:    target_earth.png
```

Deze uitkomst is het resultaat van het commando "fls -d" en toont alle niet-toegewezen bestanden als gewiste bestanden. Ook de bestanden tram28.jpg en italy.jpg, die hernoemt maar niet gewist werden.

Onderstaande foto geeft weer hoe een niet-gefragmenteerd bestand teruggehaald kan worden aan de hand van "The Sleuthkit".

```
tdg@tdg-VirtualBox:~/Downloads$ icat exFAT.dd 569 > recupWithSleuthkit.jpg
tdg@tdg-VirtualBox:~/Downloads$ display recupWithSleuthkit.jpg
```

Deze foto toont hoe (niet-gefragmenteerde) bestanden teruggehaald kunnen worden met het programma The Sleuthkit.



Deze foto geeft het resultaat weer van bovenstaande commando's.

The sleuthkit baseert aanpassingen in de broncode op basis van feedback van gebruikers. Dit probleem zal vermoedelijk in komende versies van het programma aangepakt worden en kan bij het lezen van dit werk reeds verholpen zijn.

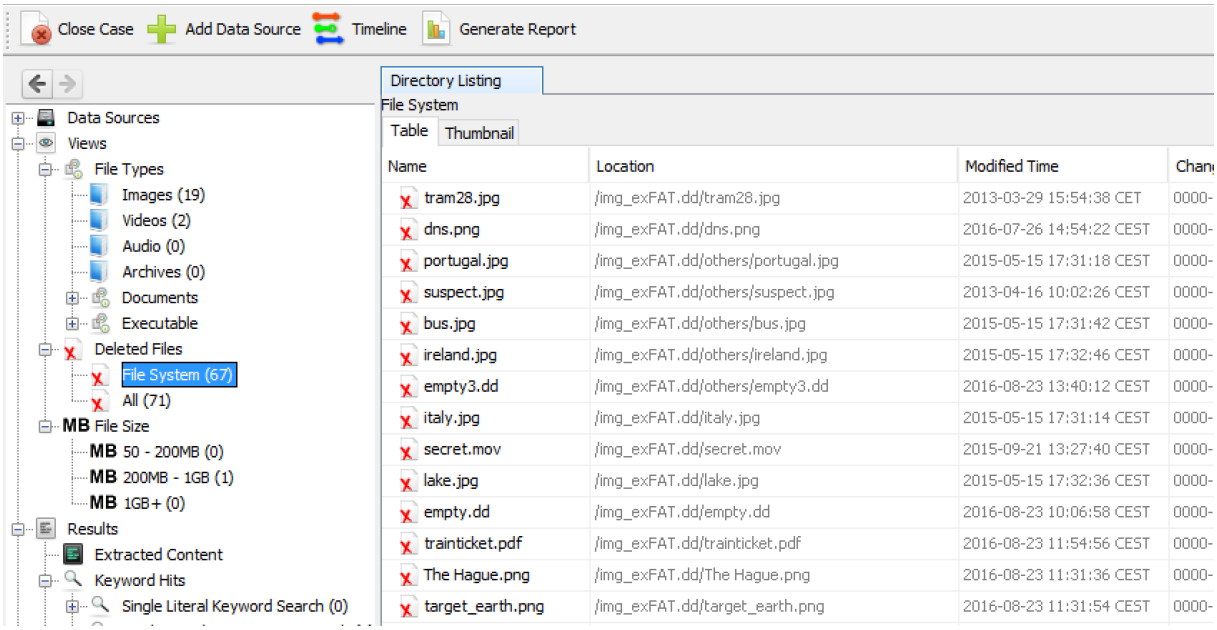
10.1.2 Autopsy

Autopsy werkt onder de motor met The Sleuthkit. Het programma draait op het besturingssysteem Windows en beschikt, in tegenstelling tot The Sleuthkit over een Grafische User Interface.

Omdat het achterliggend gebruikt maakt van The Sleuthkit adopteert het ook enkele nadelen. Ook bij Autopsy worden foutief verplaatste en hernoemde bestanden als gewist aangegeven. Het programma beschikt niet over een grafische interface voor de clusterlijst. Net zoals bij The Sleuthkit is het ook jammer dat niet getoond wordt of een bestand al dan niet gefragmenteerd is.

Het grote voordeel aan het programma is dat het gratis is. Omdat de tool minder details bevat, is hij erg handig voor simpele handelingen en zal hij voor de meeste gebruikers ruim voldoende zijn om bestanden te recupereren. Het is pas bij gefragmenteerde bestanden, of bij diepgaande problemen dat de beperkingen van dit programma zichtbaar zullen zijn.

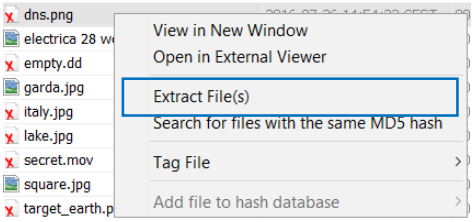
Onderstaande foto toont een overzicht van hoe Autopsy inhoud op een gegevensdrager weergeeft. De niet-toegewezen bestanden krijgen een afbeelding van een blad met een rood kruis voor de bestandsnaam. Ook hier wordt nergens het onderscheidt gemaakt tussen gewiste en verplaatste bestanden.



Deze afbeelding geeft de bestanden weer die zich in de root directory bevinden, getoond met het programma The Autopsy.

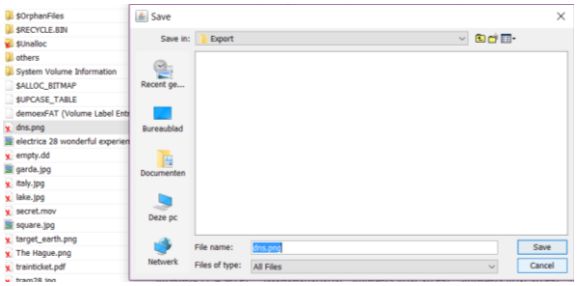
Via onderstaande werkwijze kunnen “gewiste”, niet-gefragmenteerde, bestanden teruggehaald worden:

I. Selecteer “extract file(s)” door met de rechtermuisknop op het terug te halen bestand te klikken:



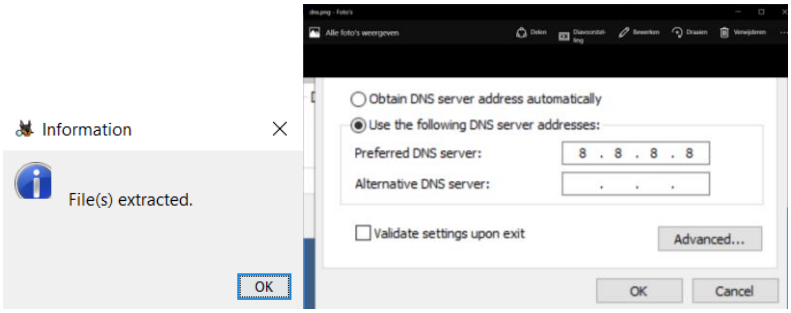
Deze afbeelding geeft weer hoe de optie “extract file(s)” gekozen kan worden.

II. Geef de gewenste opslaglocatie op:



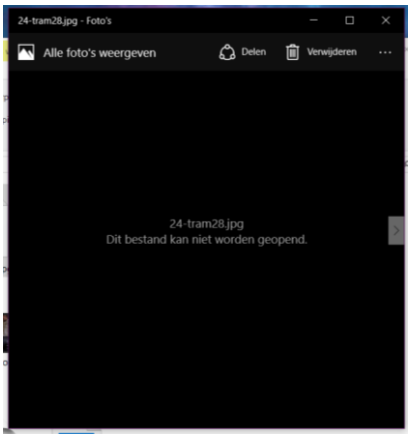
Deze afbeelding geeft weer hoe dat het bestand een opslaglocatie vraagt voor het te herstellen bestand.

III. Resultaat:



De linkse afbeelding toont dat de opdracht geslaagd is. De rechtse afbeelding toont de effectieve foto.

Wanneer gefragmenteerde afbeeldingen hersteld proberen worden, toont het programma dat de bestanden correct hersteld werden. De werkelijkheid bewijst het tegendeel:



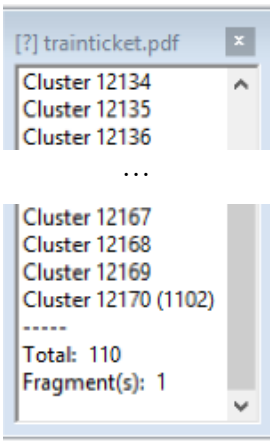
Deze afbeelding toont het resultaat van de herstelling van een gefragmenteerd bestand met het programma Autopsy.

10.1.3 X-Ways

X-ways is een onderneming die meerdere programma's aanbiedt. Het programma dat in dit werk besproken wordt, is het programma "X-ways forensics". Gemakkelijkheid halve zal in de werk de naam van het programma afgekort worden tot "X-ways". Het programma draait op het besturingssysteem Windows en beschikt over een uitgebreide gebruikersinterface.

In tegenstelling tot The Sleuthkit en Autopsy, wordt een duidelijk onderscheidt gemaakt tussen gewiste, verplaatste en hernoemde bestanden. Het uitgevoerde onderzoek naar de werking van het programma kon geen foutieve resultaten opleveren. Er mag aangenomen worden dat de interpretaties van het programma, omtrent deze termen, kloppen.

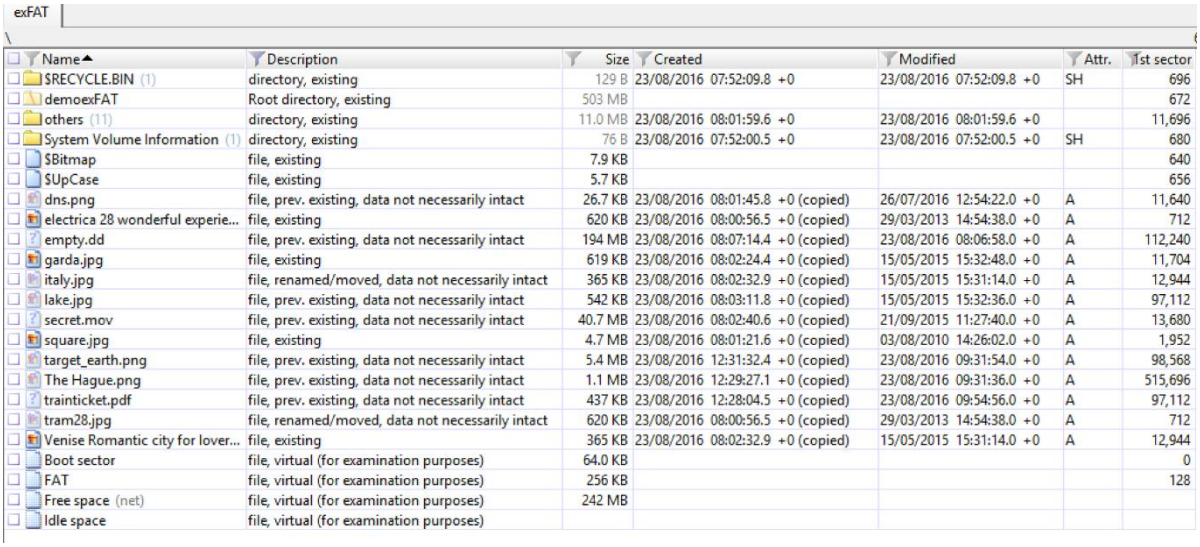
Het is met X-ways mogelijk om zowel gefragmenteerde als niet-gefragmenteerde bestanden terug te halen. Er wordt tevens ook getoond of een bestand al dan niet gefragmenteerd is, al is dit niet onmiddellijk zichtbaar in het overzicht. Ook de aanwezigheid van een clusterlijst is een belangrijk pluspunt voor het programma, al is het terughalen van gefragmenteerde bestanden niet evident aangezien de clusters niet gegroepeerd getoond worden.



Afbeelding van de clusterlijst in X-ways forensics.

Het programma biedt veel mogelijkheden voor een onderzoeker met diepgaande kennis. De vele opties maken tal van onderzoeksdaden mogelijk en zorgen voor een vlotter gebruik bij het analyseren en herstellen van bestanden. Voor leken zal dit programma echter veel te uitgebreid ogen. X-ways is niet gratis. Een jaarlijkse licentie per persoon kost 629€ (zonder groepskorting).

Onderstaande afbeelding toont de weergave van de inhoud van de root directory met het programma X-ways. Naast elk bestand staat, in Engelse termen, geschreven welke handelingen het bestand of de map ondergingen.



exFAT

Name	Description	Size	Created	Modified	Attr.	1st sector
$RECYCLE.BIN (1)	directory, existing	129 B	23/08/2016 07:52:09.8 +0	23/08/2016 07:52:09.8 +0	SH	696
demoexFAT	Root directory, existing	503 MB				672
others (11)	directory, existing	11.0 MB	23/08/2016 08:01:59.6 +0	23/08/2016 08:01:59.6 +0		11,696
System Volume Information (1)	directory, existing	76 B	23/08/2016 07:52:00.5 +0	23/08/2016 07:52:00.5 +0	SH	680
$Bitmap	file, existing	7.9 KB				640
$UpCase	file, existing	5.7 KB				656
dns.png	file, prev. existing, data not necessarily intact	26.7 KB	23/08/2016 08:01:45.8 +0 (copied)	26/07/2016 12:54:22.0 +0	A	11,640
electrica 28 wonderful experie...	file, existing	620 KB	23/08/2016 08:00:56.5 +0 (copied)	29/03/2013 14:54:38.0 +0	A	712
empty.dd	file, prev. existing, data not necessarily intact	194 MB	23/08/2016 08:07:14.4 +0 (copied)	23/08/2016 08:06:58.0 +0	A	112,240
garda.jpg	file, existing	619 KB	23/08/2016 08:02:24.4 +0 (copied)	15/05/2015 15:32:48.0 +0	A	11,704
italy.jpg	file, renamed/moved, data not necessarily intact	365 KB	23/08/2016 08:02:32.9 +0 (copied)	15/05/2015 15:31:14.0 +0	A	12,944
lake.jpg	file, prev. existing, data not necessarily intact	542 KB	23/08/2016 08:03:11.8 +0 (copied)	15/05/2015 15:32:36.0 +0	A	97,112
secret.mov	file, prev. existing, data not necessarily intact	40.7 MB	23/08/2016 08:02:40.6 +0 (copied)	21/09/2015 11:27:40.0 +0	A	13,680
square.jpg	file, existing	4.7 MB	23/08/2016 08:01:21.6 +0 (copied)	03/08/2010 14:26:02.0 +0	A	1,952
target_earth.png	file, prev. existing, data not necessarily intact	5.4 MB	23/08/2016 12:31:32.4 +0 (copied)	23/08/2016 09:31:54.0 +0	A	98,568
The Hague.png	file, prev. existing, data not necessarily intact	1.1 MB	23/08/2016 12:29:27.1 +0 (copied)	23/08/2016 09:31:36.0 +0	A	515,696
trainticket.pdf	file, prev. existing, data not necessarily intact	437 KB	23/08/2016 12:28:04.5 +0 (copied)	23/08/2016 09:54:56.0 +0	A	97,112
tram28.jpg	file, renamed/moved, data not necessarily intact	620 KB	23/08/2016 08:00:56.5 +0 (copied)	29/03/2013 14:54:38.0 +0	A	712
Venise Romantic city for lover...	file, existing	365 KB	23/08/2016 08:02:32.9 +0 (copied)	15/05/2015 15:31:14.0 +0	A	12,944
Boot sector	file, virtual (for examination purposes)	64.0 KB				0
FAT	file, virtual (for examination purposes)	256 KB				128
Free space (net)	file, virtual (for examination purposes)	242 MB				
Idle space	file, virtual (for examination purposes)					

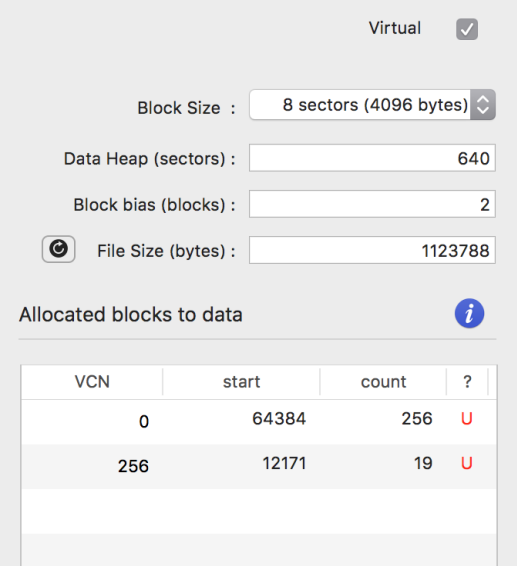
Afbeelding van de bestanden die zich in de root directory bevinden, weergegeven met X-ways forensics.

Een uitgebreide handleiding over het gebruik van het terughalen van gefragmenteerde en niet-gefragmenteerde bestanden is te verkrijgen bij de organisatie.

10.1.4 Tyrhex

Tyrhex is een gratis programma dat draait op het besturingssysteem MAC OS. Het is ontwikkeld door Yves Vandermeer, rechercheur bij de dienst ‘Federal Computer Crime Unit’ van de Federale Gerechtelijke Politie van België en docent aan de Universiteit van Dublin.

Tyrhex beschikt over een gebruikersinterface die de gebruiker van een aantal elementaire details voorziet, diepgaande details kunnen ook teruggevonden na het aanklikken van het bestand. Het toont “niet-toegewezen” bestanden als “U” (= unallocated) en “A” (=allocated), en laat zo de interpretatie over aan de onderzoeker. Tyrhex beschikt over een gegroepeerde clusterlijst waardoor het handmatig terughalen van gefragmenteerde bestanden veel overzichtelijker wordt.

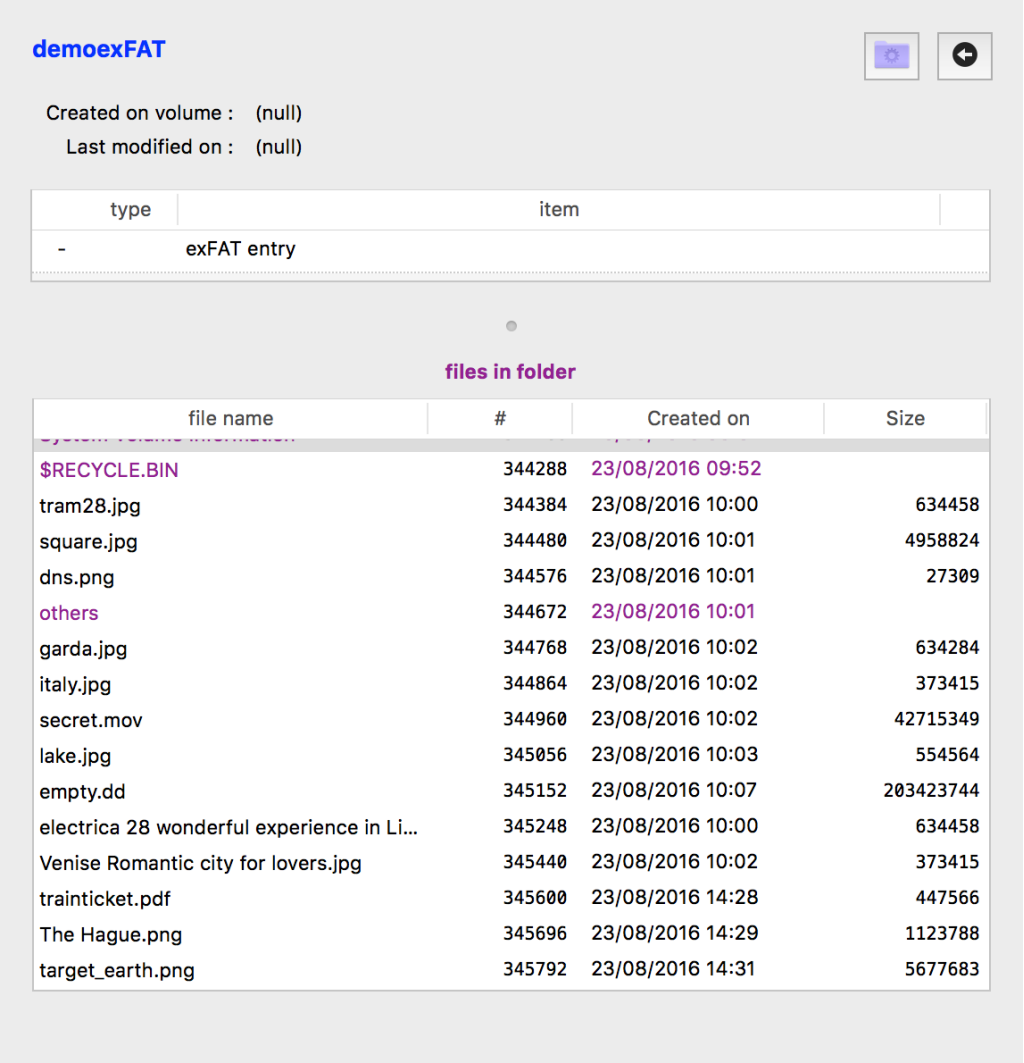


Afbeelding van de gegroepeerde clusterlijst in X-ways forensics.

Tyrhex is in staat om zowel gefragmenteerde bestanden als niet-gefragmenteerde bestanden, volledig autonoom terug te halen.

Het gratis programma kan door de eenvoudige interface gebruikt worden door gebruikers die weinig kennis hebben van bestandssystemen, maar biedt ook de mogelijkheid om diepgaandere details terug te vinden wanneer een bestand geselecteerd wordt. Op die manier komt het tegemoet aan de noden van beide gebruikers. Voor MAC OS is dit het enige programma dat de bewerkingen kan uitvoeren met een gebruikersinterface en het enige programma dat in staat is om gefragmenteerde bestanden zowel autonoom als automatisch terug te halen.

Onderstaande foto toont een overzicht van hoe Tyrhex de inhoud op een gegevensdrager weergeeft. Mappen worden in het paars weergegeven, ook de grootte en startcluster van elke map of van elk bestand worden in het overzicht getoond.



Afbeelding van de bestanden die zich in de root directory bevinden, weergegeven met Tyrhex.

Een uitgebreide handleiding over het gebruik van het terughalen van gefragmenteerde en niet-gefragmenteerde bestanden is te verkrijgen op de website van de ontwikkelaar.

10.1.5 Overzicht voor- en nadelen

Criteria	Sleuthkit (TSK)	Autopsy	X-Ways	Tyrhex
Correcte interpretatie Niet-toegewezen Bestanden	Neen	Neen	Ja	Ja(1)
Terughalen niet-gefragmenteerde bestanden	Ja	Ja	Ja	Ja
Automatisch terughalen gefragmenteerde bestanden	Neen	Neen	Ja	Ja
Grafische gebruikersinterface	Neen	Ja	Ja	Ja
Clusterlijst	Ja(2)	Neen	Ja(4)	Ja(3)
Operating system	Linux, MAC OS *(5)	Windows	Windows	MAC OS
Toont fragmentatie?	Neen	Neen	Ja (6)	Ja
Prijs	Gratis	Gratis	629€ /p./j.	Gratis

*(1) Laat interpretatie over aan onderzoeker, toont Unallocated of Allocated.
*(2) Toont geen clusterlijst voor gefragmenteerde bestanden.
*(3) Gegroepeerd, leesbaar.
*(4) Niet-gegroepeerd, minder leesbaar.
*(5) MAC O.S. d.m.v. het programma Home Brew.
*(6) Enkel in clusterlijst.

11 BESLUIT

Het gebrek aan kennis met betrekking tot exFAT, het bestandssysteem van Microsoft dat FAT32 op termijn zal moeten vervangen, heeft te maken met licenties en patenten. Microsoft kiest er immers bewust voor om de werking van exFAT niet publiek te maken en deelt betalende licenties uit aan diegenen die exFAT willen gebruiken. Het is echter van nefast belang voor forensische analisten om het bestandssysteem te kunnen doorgronden. Ook dienen programma's verbeterd/gemaakt te worden en is het noodzakelijk om gebruikte forensische methoden aan te kunnen tonen en te beschrijven zoals omschreven door de ACPO.

Een antwoord op deze noodzaak werd geboden doormiddel van dit onderzoek. Tot stand gekomen door reverse engineering, andere bronnen en het bestuderen van verouderde FAT-versies waarvan wel voldoende informatie voorhanden is. Hieruit kon opgemaakt worden dat exFAT een aantal elementen gemeen heeft met zijn voorgangers. Zo maakt exFAT bijvoorbeeld nog steeds gebruik van ketening en zal de File Allocation Table aangesproken worden voor gefragmenteerde bestanden. Oude nadelen als de te kleine bestandsgroottes en maximum volumes werden voorlopig opgelost, het blijkt zelfs dat exFAT hier in vergelijking met andere bestandssystemen veel beter scoort. De snelheid en schrijfbelasting werden verbeterd door het gebruik van een bitmap en de File Allocation Table zal niet meer benut worden voor niet-gefragmenteerde bestanden. Niet-gefragmenteerde, gewiste, bestanden kunnen nog steeds probleemloos teruggehaald worden. ExFAT biedt een extra forensisch voordeel, in tegenstelling tot de werking van FAT32, blijft de File Allocation Table ook na het wissen nog intact. Hierdoor kan bij exFAT ook een 'gewist' gefragmenteerd bestand teruggehaald worden, zolang de clusters die het bestand bevatten niet overschreven werden.

De huidige software is al tot een aantal bijzondere bewerkingen in staat. Het ene programma werkt gemakkelijker voor bepaalde bewerkingen dan het andere. Toch moet omzichtig omgesprongen worden met gebruikte methoden en bewoordingen in bepaalde programma's. Tot slot blijkt dat alle onderzochte programma's in staat zijn om niet-gefragmenteerde bestanden te recupereren maar dat enkel Tyrhex en X-ways gefragmenteerde bestanden bruikbaar kunnen terughalen.

Door de werking te bekijken vanuit het oogpunt van een forensisch analist komen bepaalde onderwerpen minder aan bod. Deze zouden in de toekomst nuttig kunnen blijken voor het ontwikkelen van software. Ook konden niet alle mogelijk werkingen van drivers onderzocht worden. Door zich te focussen op de meest gebruikte werkingen kon echter wel een algemeen beeld geschetst worden van de gebruikelijke werking. Andere drivers kunnen echter op een andere manier tewerk gaan. Dit fenomeen zou als uitbreiding op dit werk onderzocht kunnen worden.

12 BIJLAGEN

Startwaarde	Suboffset (byte #0)	Lengte (bytes)	Inhoud
0x81	**Bitmapinformatie entry**		
	0	1	Bevat de waarde 0x81
	20	4	Eerste cluster (Meestal 0x03)
	24	8	Grootte in bytes = ((Aantal clusters - 2)/8)+1
0x82	**Uppercase tabel entry**		
	0	1	Bevat de waarde 0x82
	20	4	Eerste cluster (Meestal 0x02)
	24	8	Grootte in bytes
0x83	**Volume label entry**		
	0	1	Bevat de waarde 0x83
	1	1	Lengte van de volumenaam, max. 11
	2	Naamlengte * 2	Volumenaam (Unicode karakters, max 11 karakters lang)
0x85	**Bestand/Map main entry**		
	0	1	Bevat de waarde 0x85 (= 0x05 wanneer niet meer in gebruik)
	1	1	Aantal subentry's voor het bestand
	4	1	DOS-attributen
	8	4	Datum en tijd van aanmaak (DOS-formaat)
	12	4	Datum en tijd van laatste keer geopend/bekeken (DOS-formaat)
	16	4	Datum en tijd van laatste aanpassing (DOS-formaat)
	20	1	Aanmaaktijd bias (1/100s)
	21	1	Aanpassingstijd bias (1/100s)
	22	1	Tijdzone van aanmaak
	23	1	Tijdzone van laatste keer geopend/bekeken
	24	1	Tijdzone van laatste aanpassing

0xC0	**Clusterverloop**		
	0	1	Bevat de waarde 0xC0 (= 0x40 wanneer niet gebruikt)
	1	1	Geeft in de byte, in bits aan, of FAT gebruikt wordt (0x02 = niet). De betekenis van andere waarden is niet gekend.
	3	1	Bestandsnaamlengte
	20	4	Nummer van de startcluster van het bestand
	24	8	Bestandsgrootte van het bestand in bytes
0xC1	**Bestandsnaam**		
	0	1	Bevat de waarde 0xC1 (= 0x41 wanneer niet gebruikt)
	2	30	Bestandsnaam in unicode (max. 15)

13 BIBLIOGRAFIE

Bhat, W. A., & Quadri, S. M. K. (2010). Review of FAT Data Structure of FAT32 file system. *Oriental Journal of Computer Science & Technology*, *3*(1).

Bhat, W. A., & Quadri, S. M. K. (2011). Design Considerations for Developing Disk File System. *Journal of Emerging Trends in Computing and Information Sciences*, *2*(12).

Chen R. (2008). Windows Confidential A Brief and Incomplete History of FAT32. Geraadpleegd op 12 februari, 2016, van https://technet.microsoft.com/nl-nl/magazine/8b7e756a-339d-43bd-8d01-bda67c099116

Compaq Computer Corporation, Phoenix Technologies LTD & Intel corporation. (1994). *Plug and Play BIOS Specification 1.0A.*

Ghania A. S. (2016). Analyzing Master Boot Record for Forensic Investigations. *International Journal of Applied Information Systems*, *10* (2249-0868, 1 - 18) .

Microsoft. (2005). FAT32 File System. Geraadpleegd op 20 februari, 2016 van http://web.archive.org/web/20050319235548/www.microsoft.com/resources/documentation/Windows/XP/all/reskit/en-us/prkc_fil_cycz.asp

Microsoft. (2005). Windows XP Professional Resource Kit: Working with File Systems. Geraadpleegd op 15 maart, 2016, van https://web.achive.org/web/20060307082555/http://www.microsoft.com/technet/prodtechnol/winxppro/reskit/c13621675.mspx

Microsoft. (2006). Beschrijving van het bestandssysteem FAT32. Geraadpleegd op 12 februari, 2016, van https://support.microsoft.com/nl-nl/kb/154997

Microsoft. (2007). Overview of FAT, HPFS, and NTFS File Systems. Geraadpleegd op 20 februari, 2016, van https://support.microsoft.com/nl-nl/kb/100108

Microsoft. (2015). MS-DOS: De map en submap beperkingen. Geraadpleegd op 12 februari, 2016, van https://support.microsoft.com/nl-nl/kb/39927

Microsoft. (2015). Vaste schijven groter dan 64 GB worden niet volledig herkend door Fdisk. Geraadpleegd op 12 april, 2016, van https://support.microsoft.com/nl-nl/kb/263044

Mitchell, I., & Hara, M. S. (2009). *Delivery of Forensic Computing Analysis using Effective Formative Feedback.*

Munegowda, K., Venkatraman, S., & Raju, D. G. (2011, October). *The Extend FAT file system: Differentiating with FAT32 file system*. In *Linux Conference, Prague, Czech Republic, Europe*.

Nabity, P., & Landry, B. J. (2013). Recovering deleted and wiped files: A digital forensic comparison of FAT32 and NTFS file systems using evidence eliminator.

Nikkel, B. J. (2009). Forensic analysis of GPT disks and GUID partition tables.*Digital Investigation*, *6*(1), 39-47. doi:10.1016/j.diin.2009.07.

Shullich R. (2009). *Reverse Engineering the Microsoft exFAT File* [master- of bachelorproef]. Plaats van uitgave: Boston, Verenigde Staten. Sans Institute.

Sleuthkitorg. (2016). Sleuthkitorg. Geraadpleegd op 12 augustus, 2016, van http://www.sleuthkit.org/

Williams J. (2012). *ACPO Good Practice Guide ACPO Good Practice Guide for Digital Evidence*. Groot-Brittanië: Association of Chief Police Officers .

X-waysnet. (2016). X-waysnet. Geraadpleegd op 12 augustus, 2016, van https://www.x-ways.net/

Yves vandermeer. (2016). Tyrhexcom. Geraadpleegd op 12 augustus, 2016, van http://www.tyrhex.com/