FernUniversität in Hagen
Fakultät für Mathematik und Informatik

Paper
im Fachpraktikum IT-Sicherheit, IT-Forensik und Datenschutz
WS 2024/2025

Polyglotte Dateien mit TrueCrypt Volumes

von
Matthias Ferstl
Fabian Kozlowski
Stefan Leippe
Malte Muthesius

Abgabe: 10.01.2025
Vortrag: 25.01.2025

Betreuer: Prof. Dr. Tobias Eggendorfer

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Problemstellung

Lange Zeit galt die Verschlüsselungssoftware TrueCrypt als unüberwindbar, siehe [1] und im Original [2]. Mittlerweile wurden Schwachstellen des Programms aufgezeigt (auch wenn das Bundesamt für Sicherheit in der Informationstechnik zu teilweise anderen Schlüssen kommt, siehe [3]), sodass eine Weiterentwicklung innerhalb der Abspaltung VeraCrypt erfolgte, die teilweise noch mit TrueCrypt kompatibel war.

Die Sicherheit eines verschlüsselten TrueCrypt oder VeraCrypt Volumens wird noch weiter erhöht, wenn es innerhalb einer Datei völlig anderen Formats so versteckt wird, dass die entstehende Datei sowohl als Volume in das entsprechende Verschlüsselungsprogramm eingehängt als auch mit der für die Hostdatei geeigneten Software geöffnet werden kann. Das Ergebnis wird als polyglotte Datei bezeichnet, die Technik der Zusammenführung der Dateiformate als Steganographie.

In diesem Paper werden die Grundlagen diskutiert, auf denen aufbauend verschiendene Dateitypen daraufhin analysiert werden, ob sie als Hostmedium einer polyglotten Datei mit einem TrueCrypt-Volume geeignet sind. Dazu wird wie folgt vorgegangen:
Im Kapitel 2 wird zunächst die Struktur des TrueCrypt Headers dargestellt und daran aufgezeigt, welche Teile innerhalb der polyglotten Datei unverändert vorliegen müssen und wie die Anpassung an die Struktur der Hostdatei vorgenommen werden kann. Anschließend werden unterschiedliche Dateiformate daraufhin untersucht, ob bzw. wie ein TrueCrypt Volume eingefügt werden kann. Die Struktur der Hostdatei muss so erhalten bleiben, dass die Datei noch im ursprünglichen Format geöffnet und verarbeitet werden kann.

Kapitel 3 enthält den Entwurf eines Tools, mit dem Dateien des PNG-Formats automatisiert mit einem TrueCrypt-Volume zu einer polyglotten Datei zusammengefügt werden können. Es wird generisch dargestellt, wie

mit Hilfe des Tools sowohl die Hostdatei als auch das TrueCrypt-Volumes unabhängig von der konkreten Struktur angepasst werden können, damit beide Dateiformate und ihre ursprünglichen Funktionen erhalten bleiben. Außerdem wird gezeigt, welche Schnittstellen das Tool bietet und welche konkreten Anpassungen notwendig sind, damit auch andere Dateiformate mit eingebunden und automatisiert mit TrueCrypt Volumes steganographisch kombiniert werden können.

In Kapitel 4 werden die erfolgten Tooltests sowie die Tests der erzeugten Dateien mit verschiedenen forensischen Werkzeugen dokumentiert. Darauf aufbauend folgen Verbesserungsvorschläge sowie Zusammenfassung und Ausblick.

2 Headerstrukturen und Steganographie

Der TrueCrypt Header besteht aus den ersten 512 Bytes eines Volumes. In den ersten 64 Byte wird ein bei der Verschlüsselung des Volumes zufällig erzeugtes „Salt “gespeichert, mit dem die Hashwerte der Masterkeys erzeugt werden.

Das Salt kann beliebig sein, insofern auch angepasst werden, um der Headerstruktur der Hostdatei im polyglotten Format zu genügen. Umgekehrt dürfen die Bytes nach Offset 63 nicht verändert werden. Da der restliche Header des TrueCrypt Volumes von Offset 64 bis 511 wichtige Strukturen enthält, die zur Entschlüsselung des Volumes entweder geprüft oder verwendet werden, kann bei Veränderung dieser Werte das Volume nicht mehr entschlüsselt werden. Die genaue Verteilung der Einträge und deren Bedeutung für die Entschlüsselung kann Abbildung 2.1 entnommen werden.

Als Hostdatei können Dateiformate dienen, bei denen innerhalb der ersten 64 Bytes eine Einfügemöglichkeit besteht, ohne dass das Dateiformat korrupt wird. Die einzufügenden Daten müssen eine beliebige Länge haben dürfen. Sollten noch Anpassungen in Form eines Präfixes vor den einzufügenden Daten notwendig sein, so müssen diese ebenfalls innerhalb der ersten 64 Bytes der Hostdatei untergebracht werden können.

Das polyglotte File wird nach folgendem Ablauf aus der Hostdatei und dem TrueCrypt Volume zusammengesetzt: Die ersten Bytes werden mit den Daten aus der Hostdatei belegt. Dadurch wird sichergestellt, dass die Struktur und Lesbarkeit der Hostdatei enthalten bleibt. Innerhalb der ersten 64 Byte wird die optimale Einfügeposition für das TrueCrypt Volume ermittelt. Dies kann z. B. nach einem Block obligatorisch vorhandener Daten aus der Hostdatei sein. Anschließend werden, beginnend mit

Offset (bytes)	Size (bytes)	Encryption Status†	Description
0	64	Unencrypted§	Salt
64	4	Encrypted	ASCII string "TRUE"
68	2	Encrypted	Volume header format version (5)
70	2	Encrypted	Minimum program version required to open the volume
72	4	Encrypted	CRC-32 checksum of the (decrypted) bytes 256-511
76	16	Encrypted	Reserved (must contain zeroes)
92	8	Encrypted	Size of hidden volume (set to zero in non-hidden volumes)
100	8	Encrypted	Size of volume
108	8	Encrypted	Byte offset of the start of the master key scope
116	8	Encrypted	Size of the encrypted area within the master key scope
124	4	Encrypted	Flag bits (bit 0 set: system encryption; bit 1 set: non-system in-place-encrypted/decrypted volume; bits 2–31 are reserved)
128	4	Encrypted	Sector size (in bytes)
132	120	Encrypted	Reserved (must contain zeroes)
252	4	Encrypted	CRC-32 checksum of the (decrypted) bytes 64-251
256	*Var.*	Encrypted	Concatenated primary and secondary master keys**
512	65024	Encrypted	Reserved (for system encryption, this item is omitted‡‡)

Abbildung 2.1: Headerstruktur TrueCrypt Volume, Quelle: [4]

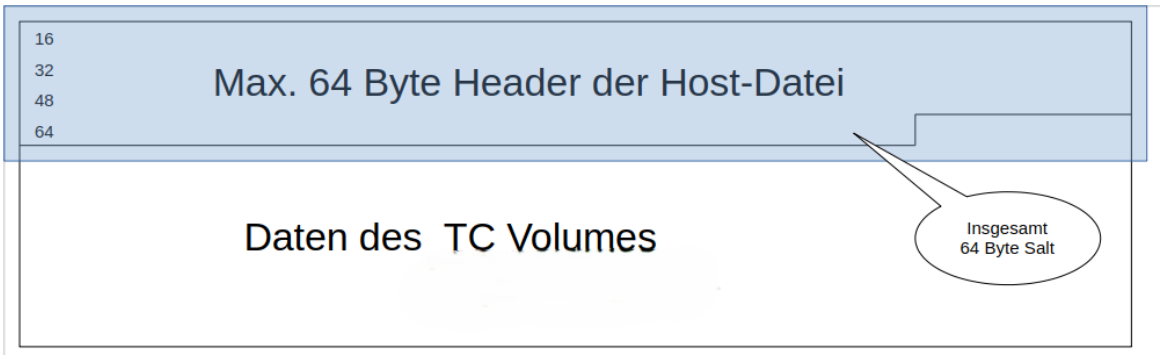

Abbildung 2.2: Header des polyglotten Files

dem Offset der Einfügeposition, die Daten aus dem TrueCrypt Volume übernommen. Spätestens ab Byte 64 entsprechen die Daten dem unbedingt und unverändert benötigten Teil des TrueCrypt Volumes, gefolgt von den verschlüsselten Daten. So wird von dem TrueCrypt Volume nur das Salt verändert, die restlichen Daten aber beibehalten, siehe Abbildung 2.2.
Im folgenden Abschnitt werden verschiedene Dateiformate dargestellt, die sich als Hostdatei zur Erstellung von polyglotten Dateien mit TrueCrypt Volumes eignen. Hierbei wird auch darauf eingegangen, wie die Hostdateien anzupassen sind und welche Einschränkungen berücksichtigt werden müssen.

1. Bilddateiformate

   PNG-Dateien Der Header von PNG-Dateien [5] startet mit den Magic Numbers *89 50 4e 47 0d 0a 1a 0a*. Dafür werden 8 Bytes reserviert. Im weiteren Verlauf sind PNG-Dateien in so genannte Chunks aufgeteilt, in denen unterschiedliche Arten von Daten gespeichert werden. Der erste Chunk nach den Magic Numbers ist der IHDR-Chunk, der Image Header. Hier sind Daten wie Dimensionen des Bildes, Bit-Tiefe, Farb-Typ und Kompression gespeichert. Der Bereich bis dahin darf nicht verändert werden.

   Im Anschluss an den IHDR-Chunk folgt der ICC-Chunk, in dem Farbprofile gespeichert werden. Da der ICC-Chunk 2733 Bytes umfasst, muss das TrueCrypt-Volume davor eingefügt werden. da ab Offset 64 der originale TrueCrypt Header zu übernehmen ist. Da jeder Chunk

mit vier Bytes beginnt, in denen die Länge des Chunks codiert wird, werden weitere 4 Bytes benötigt.

Zusätzlich werden weitere 4 Bytes benötigt, in denen der Typ des Chunks angegeben wird. Bei der Bezeichnung bestehen keine Beschränkungen. Allerdings ist auf die Verteilung von Klein- und Großbuchstaben zu achten. Wenn das fünfte Bit des ersten Bytes der Typbezeichnung, das so genannte Ancillary Bit, mit 1 belegt ist, wird der Chunk als Hilfschunk interpretiert und somit ignoriert. Das Ancillary Bit ist dann mit 1 belegt, wenn das entsprechende ASCII-Zeichen ein Kleinbuchstabe ist. Damit ist die Einfügeposition aber immer noch innerhalb der ersten 64 Bytes. [1] Da die Länge des Chunks in 4 Bytes kodiert wird, ist die Größe des einzufügenden Volumes entsprechend beschränkt.

An das Ende des Chunks ist noch die CRC-Prüfsumme anzufügen. Diese ist zu invertieren, indem Sie mit 0xFFFFFFFF logisch UND-verknüpft wird. [2]

TIFF-Dateien TIFF-Dateien [6] beginnen mit einem 8 Byte großen Header; die ersten beiden Bytes zeigen an, ob die Byte-Reihenfolge little endian ist (Belegung mit 0x4949) oder big endian (0x4D4D). Dementsprechend sind die Belegungen der folgenden Bytes. Die nächsten Bytes mit Offset 2 und 3 enthalten die Dezimalzahl 42, entsprechend der in Bytes 0 und 1 angegebenen Bytestruktur little oder big endian. In den nächsten vier Bytes wird der Offset zum ersten Image File Directory (IFD) eingetragen.

Im weiteren Verlauf bestehen TIFF-Dateien aus weiteren IFD-Einträgen, die in den ersten zwei Bytes die Nummer des Tags enthalten, gefolgt von zwei Bytes für den Feldtyp. Anschließend sind vier Bytes mit der Größe des Datenfelds belegt, schließlich stehen in den nächsten

[1] Tools wie beispielsweise unter https://www.nayuki.io/page/png-file-chunk-inspector können helfen, das Ende des IHDR-Chunks zu finden.

[2] Für die codemäßige Umsetzung sei auf http://www.libpng.org/pub/png/spec/1.2/PNG-CRCAppendix.html verwiesen.

vier Bytes entweder die Daten selbst oder der Offset, an dem das Datenfeld beginnt. Dies schränkt die Größe des einfügbaren Volumes auf Größen ein, die sich mit vier Bytes darstellen lassen.

Bestimmte Feldtypen, wie beispielsweise der Typ INFO, werden von den meisten Programmen, mit denen TIFF-Dateien angezeigt werden können, ignoriert. Dementsprechend könnte das einzubettende TrueCrypt-Volume mit einer laufenden Nummer und einem Feldtyp INFO versehen werden. Mit der Größenangabe und dem Offset wäre dann ein IFD komplett, mit dem auf das Volume verwiesen würde. Entsprechend könnte das TrueCrypt Volume an verschiedenen Stellen in eine TIFF-Datei eingebunden werden, so lange der Header und mindestens ein IFD davor eingebaut wurden. Es würden aber innerhalb der 64 Byte, die für das Salt notwendig sind, auch mehrere IFD passen.

Angepasst werden müssen alle Werte, die als Offset angegeben werden. Zum einen ist das der Verweis auf das erste IFD. Sollten weitere Directories vorhanden sein - dies ist z. B. der Fall, wenn in einem File mehrere Bilder oder eine Miniaturansicht des Bildes etc. gespeichert sind -, müssen auch die Offsets angepasst werden, die auf diese verweisen. Weiterhin müssen zum anderen die Offsets der Tags angepasst werden, bei denen die Werte zu groß sind für vier Bytes. Es muss jeweils die Größe des eingefügten TrueCrypt Volumes abzüglich 8 Bytes für den übernommenen Header addiert und in die Byteadressen zurückgeschrieben werden.

JPEG-Dateien JPEG-Dateien beginnen mit den Magic Numbers, dem so genannten Start-of-Image Signal (SOI) *FF D8*. Anschließend sind zwei Segmente vorhanden, die Metadaten des Bildes enthalten, wie zum einen JFIF als ASCII-String und die JFIF-Version, sowie zum anderen die Bildauflösung und die Farbprofildaten. Diese Segmente beginnen mit *FF E0* bzw. *FF E2*. Dazwischen ließe sich noch das APP1 Segment platzieren, das mit *FF E1* beginnt und freie Metadaten wie Kameratyp, Standort, Belichtung etc. enthalten kann. Die Positionierung erfolgt grundsätzlich innerhalb der ersten 64 Bytes, sodass ein Einfügen eines Crypt Volumes möglich ist.

Beginnend mit dem APP1 Marker *FF E1* folgen dann zwei Bytes, die die Länge des so genannten EXIF-Segments einschließlich der beiden Bytes mit der Längenangabe beinhalten. Die meisten Viewer ignorieren die APP1-Segmente. Allerdings ist durch die Längenangabe innerhalb von zwei Bytes die größe des einfügbaren Volumes stark eingeschränkt.

BMP Der Header von BMP-Dateien hat eine feste Struktur und beginnt stets mit den 2-Byte großen Magic-Bytes BM in ASCII, die den Dateityp kennzeichnen, siehe [7] und [8]. Die Magic-Bytes sind Bestandteil des so genannten File-Headers, welcher eine Gesamtgröße von 14 Bytes hat. Neben den Magic-Bytes enthält der Header Informationen über die BMP-Dateigröße (4 Bytes), zwei Reserve-Felder mit einer Größe von jeweils 2 Bytes, welche einen von der BMP-erstellenden Software abhängigen Wert beinhalten, und einen 4 Byte großer Eintrag, welcher den Offset zu dem Beginn der Bilddaten angibt.

Es wäre zu erwarten, dass die Dateigröße in Byteoffset 2-5 um die Größe des eingefügten Volumes zu erhöhen ist. Eigene Experimente haben gezeigt, dass das File auch ohne Anpassung der Dateigröße weiter lesbar ist.

Das Offset-Feld muss jedoch so angepasst werden, dass das einzubindende TrueCrypt-Volume vom Parser übersprungen wird. Der Wert des Feldes ist auszulesen und um die Größe des TrueCrypt-Volumes abzüglich des Offsets der Einfügeposition erhöht wieder zurückzuschreiben. Da das Offset-Feld eine fixe Größe von 4 Bytes hat, kann maximal ein TrueCrypt Volume von 4 Gigabyte eingebunden werden.

Im Anschluss an den File-Header folgt der DIB-Header Dieser Header ist 40 Byte groß und beinhaltet Details über das eigentliche Bild wie Breite, Höhe, Farbtiefe und Komprimierungsart. Diese Einträge müssen für das Einfügen eines TrueCrypt-Volumens nicht verändert

werden. Ab Byteoffset 54 kann das TrueCrypt-Volume eingefügt werden.

SVG-Dateien SVG-Dateien sind Bildformate, bei denen die Informationen über Parameter und Inhalt des Bildes als Vekrografik mit xml-basierten Anweisungen codiert sind [9]. Aufgrund dessen können die Sprachregeln von xml herangezogen werden. Dort können beliebig große Kommentar eingefügt werden, deren Anfang mit der Zeichenfolge <!– markiert wird und deren Ende mit der Zeichenfolge //–>, wenn es sich um einen mehrzeiligen Kommentar handelt. Es kann also nach dem ersten Vorkommen des Zeichens > gesucht werden, sodass sichergestellt ist, dass das Crypt Volume nicht innerhalb eines Befehls eingefügt wird, sondern an dessen Ende. Die Position kann als Offset für das einzufügende Volume verwendet werden.

Ab der Position des ermittelten Offsets wird dann die Zeichenfolge <!– eingefügt, gefolgt von den Daten des Volumes ab der Position des Offsets + 4 Bytes für den Kommentarmarker. Anschließend folgen die Zeichenkette //–> und der Rest des SVG-Files. In der Struktur können beide Dateiformate verarbeitet werden.

2. Audio- und Videoformate

WAV-Dateien Im Header von WAV Dateien [10] steht in den ersten vier Bytes das Wort "RIFF". Die nächsten vier Bytes sind mit der Dateigröße belegt. Nach Einfügen des Volumes sind die Werte in diesen Bytes entsprechend um die Größe der eingefügten Daten zu erhöhen. Hier ist wieder zu berücksichtigen, dass die polyglotte Datei einschließlich der eingefügten Daten nicht größer sein darf als Werte, die in vier Bytes codiert werden können. Es folgen weitere 28 Bytes mit Header-Daten, die an dieser Stelle belassen werden müssen, sodass ab Offset 35 das Crypt Volume eingefügt werden kann.

Die Einfügung erfolgt als Chunk. Ein Chunk beginnt mit einer ID. Die meisten Programme, mit denen WAV-Dateien geöffnet werden können, ignorieren Chunks mit der ID INFO. Anschließend folgen vier

Bytes, in denen die Größe des Chunks als little endian 32-bit Integer angegeben wird. Dadurch wird die Größe des einfügbaren Volumes entsprechend beschränkt. Nach dem Ende des Chunks wird der Rest der Hostdatei eingefügt.

MP4-Dateien MP4-Dateien [11] bestehen aus Boxen, die mit vier Bytes beginnen, in denen die Größe der Box codiert ist als big ending 32-bit Integer. Die erste Box muss vom Typ ftyp sein, gefolgt von der Angabe eines Subtypen wie mmp4, mp41, mp42 oder weiteren, siehe [12]. Die restlichen Daten können frei verteilt sein und auch auf anderen Systemen abgelegt werden. Nach der ftyp-Box lässt sich also eine benutzerdefinierte Box mit dem TrueCrypt Volume als Inhalt einfügen.

Die Box beginnt mit der Länge der Daten. Hier ist in vier Bytes die Größe des Volumes abzüglich der Größe der ftyp-Box einzutragen. Dadurch wird die Länge der einfügbaren Daten entsprechend eingeschränkt. Die Bytes des Volumes sind dann ab Offset (Größe der ftyp-Box + 8) einzufügen. Die 8 Bytes werden mit der Größe der Box und dem Boxtypen befüllt. Eine Übersicht der möglichen Boxtypen kann [13] entnommen werden. Die Boxtypen udta (User-Data) sowie free und skip werden für gewöhnlich von Videoprogrammen ignoriert, sodass diese gewählt werden können.

MOV-Dateien MOV-Dateien [14] bestehen aus so genannten Atomen, die hierarchisch angeordnet sind und folgende Struktur haben: eine 4-Byte große Angabe der Atomgröße, gefolgt von einer 4 Byte großen Angabe des Atomtyps sowie den eigentlichen Daten. Durch die Angabe der Größe innerhalb von 4 Bytes wird die Länge der einfügbaren Daten entsprechend eingeschränkt.

Um ein TrueCrypt-Volume in eine MOV-Datei einzufügen und somit ein Polyglott zu erhalten, ist das TrueCrypt-Volume als erstes Atom zu definieren. Als Typ ist skip anzugeben. Dieser Typ stellt sicher, dass ein Parser das Atom überspringt und nicht versucht, die Daten des TrueCrypt-Volumes wiederzugeben. Im Anschluss daran muss das Atom folgen, welches die Mediendaten (Typ mdat) enthält, gefolgt

von einem Atom des Typen stco, das die Offsets zu den Videodatenblöcken im mdat Atom enthält.

Ein Atom vom Typ stco ist wie folgt aufgebaut: nach Angabe der Atomgröße (4 Bytes) und des Atomtyps (4 Bytes) folgen 4 Bytes zur Angabe der Version, 4 Bytes die angeben, wie viele Offsets vorliegen und dann die eigentliche Angabe der Offsets, die auf die Positionen im ‚mdat' Atom verweisen (variable Größe von 4*n Bytes, wobei n die Anzahl Offsets ist). Diese Offsets sind um die Größe des eingefügten TrueCrypt Volumes zu erhöhen, damit die Positionen weiter gefunden werden können.

3. Sonstige Formate

ZIP-Dateien ZIP-Parser interpretieren ZIP-Dateien typischerweise, indem sie am Ende der Datei beginnen und nach der Signatur des End of Central Directory Record (EOCD) suchen [15]. Die Signatur ist *50 4B 05 06*. Sobald der EOCD gefunden ist, liest der Parser die Einträge im Central Directory, das die Offsets zu den Local File Headers enthält, in denen die eigentlichen Datei-Daten referenziert sind. Da der ZIP-Parser nur das Central Directory und die Local File Headers verarbeitet und alles ignoriert, was vor dem EOCD liegt, kann ein TrueCrypt-Volumen am Anfang der Datei einfügen, ohne dass dies die Funktionalität des ZIP-Archivs beeinträchtigt. Allerdings ist der Offset zum Central Directory anzupassen, da der entsprechende Eintrag um die Länge des TrueCrypt Volumes nach hinten verschoben wurde. Der Offset ist vier Bytes groß ab Byte 16 im EOCD. Aufgrund der Codierung in vier Bytes ist die Größe des einfügbaren TrueCrypt dahingehend eingeschränkt, dass sich der neue Offset in vier Bytes darstellen lässt.

ICO-Dateien Das ICO-Dateiformat [16] besitzt eine einfache Header-Struktur, die aus zwei Hauptkomponenten besteht. Die erste Komponente ist das Icon-Directory, das 6 Bytes groß ist und grundlegende Informationen wie den Dateityp und die Anzahl der im Icon enthaltenen Bildressourcen enthält. Die zweite Komponente ist das Icon-Directory-Entry mit einer Größe von 16 Bytes. Dieses Entry speichert

Metadaten für jedes Bild, darunter die Höhe, Breite, Farbanzahl, die Bildgröße und das Image-Offset. Dieses Image-Offset, bestehend aus den letzten 4 Bytes des Icon-Directory-Entry, zeigt auf den Startpunkt der Bilddaten.

Um ein TrueCrypt-Volumen in eine ICO-Datei einzubetten und ein Polyglott zu erstellen, wird das Image-Offset im Icon-Directory-Entry so angepasst, dass es die Größe des TrueCrypt-Volumens berücksichtigt. Dadurch wird sichergestellt, dass der Parser das TrueCrypt-Volumen überspringt und korrekt auf den Anfang der Bilddaten verweist. Die übrigen Felder im Icon-Directory und Icon-Directory-Entry können unverändert bleiben, da sie die Einbettung des Volumens nicht beeinflussen. Ab Byteoffset 22 kann das Volume eingefügt werden. Anschließend folgt der Rest der Hostdatei. Durch das geänderte Offset bleibt die Struktur beider Dateitypen erhalten. Aufgrund des 4 Byte großen Offsets ist die Größe des einfügbaren Volumes aber entsprechend beschränkt.

html-Dateien Bei HTML-Dateien handelt es sich um ein textbasiertes Dateiformat. Befehle werden in spitze Klammern <...> eingefügt. Nach einem Befehl kann ein beliebig großer Kommentar eingefügt werden, dessen Anfang mit der Zeichenfolge <!– markiert wird und dessen Ende mit der Zeichenfolge //–>, wenn es sich um einen mehrzeiligen Kommentar handelt [17]. Es kann also nach dem ersten Vorkommen des Zeichens > gesucht werden, sodass sichergestellt ist, dass das Crypt Volume nicht innerhalb eines Befehls eingefügt wird, sondern an dessen Ende. Die Position kann als Offset für das einzufügende Volume verwendet werden.

Ab der Position des ermittelten Offsets wird dann die Zeichenfolge <!– eingefügt, gefolgt von den Daten des Volumes ab der Position des Offsets + 4 Bytes für den Kommentarmarker. Anschließend folgen die Zeichenkette //–> und der Rest des HTML-Files. In der Struktur können beide Dateiformate verarbeitet werden.

dll-Dateien DLL-Dateien sind Dateien, die Bibliotheken mit dynamischen Verknüpfungen enthalten [18]. Dadurch können Ressourcen zur

Laufzeit von mehreren Programmen aus gleichzeitig geöffnet und verwendet werden. Dies führt zu einer hohen Flexibilität und effizienten Speichernutzung.

Der DOS-Header von DLL-Dateien besteht aus 64 Bytes. Ab Byte 0x3C ist der Offset des PE-Headers angegeben, der damit auf die nächste Komponente verweist. Der Platz zwischen DOS-Header und PE-Header wird in der Regel nicht verwendet, sodass hier ein Crypt Volume eingefügt werden kann. Der PE-Header Offset ist um die Größe der eingefügten Daten abzüglich 64 Bytes zu erhöhen, da der Header der DLL-Datei vollständig verwendet wird und somit das Crypt Volume erst nach dem Salt eingefügt wird. Dabei ist zu beachten, dass der Offset in einem Wort im Big Endian Format codiert wird. Anschließend an das Crypt Volume folgen die weiteren Bytes der DLL-Datei.

3 Entwurf eines Tools zur Erstellung von polyglotten Dateien mit TrueCrypt Volumen

Um ein Tool zu entwerfen, das automatisiert TrueCrypt Volumes mit anderen Dateitypen zu polyglotten Dateien zusammenfügt, wird zunächst zusammengefasst, welche Schritte dazu vorzunehmen sind. Die Zielstruktur ist die, in der das Salt des entstandenen Polyglotts der Struktur des Headers der Hostdatei entspricht. Ab dem Byte mit Offset 64 müssen die Daten dem eingefügten Volume entsprechen. Der Rest der Hostdatei folgt dann nach dem Ende der Daten des TrueCrypt Volumes.

Zusammenfügen lassen sich die Daten nur im entschlüsselten Zustand. Da das Salt unverschlüsselt in der Datei vorliegt, sind die ersten 64 Bytes des zusammengefügten Files bereits bekannt. Würde das unbearbeitete zusammengefügte File, bei dem nur das Volume formatspezifisch in die Hostdatei eingefügt wurde, bereits als TrueCrypt-Volume interpretiert, so würden Salt und verschlüsselter Header nicht zusammenpassen. Dann könnte das TrueCrypt Volume trotz bekannten Passworts nicht entschlüsselt werden. Der verschlüsselte Header entstammt dem eingefügten TrurCrypt Volume, während das Salt dem Beginn der Host-Datei entspricht.

Zunächst müssen also zwei Entschlüsselungen vorgenommen werden: zum einen ist das TrueCrypt Volume zu entschlüsseln mit dem dateispezifischen Salt zur Ermittlung des Schlüssels. Der Erfolg des Entschlüsselns kann erkannt werden, wenn ab Byteoffset 64 der ASCII-String "TRUE" zu lesen ist. Im weiteren Verlauf des entschlüsselten Headers folgen weitere Angaben, die zum ordnungsgemäßen Einhängen des Volumes in TrueCrypt notwendig sind [19].

Zum anderen ist die Datei zu entschlüsseln, bei der das TrueCrypt Volume in die Hostdatei eingefügt wurde. Hierzu ist für die Schlüsselableitungsfunktion das Salt zu nutzen, das in der kombinierten Datei als Header der Hostdatei, ggf. unter Ergänzung durch einige Bytes aus dem Crypt Volume, in den ersten 64 Bytes vorliegt. Die genaue Zusammensetzung erklärt sich durch die dateitypspezifische Einfügeposition bzw. evtl. geänderten und / oder eingefügten Bytes, siehe Abschnitt 2.

Da in der verschlüsselten Datei ab Byteoffset 64 bis 511 der Header des TrueCrypt Volumes vorliegen muss, sind diese Bytes aus dem entschlüsselten Crypt Volume in die kombinierte Datei an die entsprechende Position zu kopieren. Zur Ableitung des Schlüssels für die Verschlüsselung des Polyglotts wird wieder das Salt aus der kombinierten Datei verwendet. Nach der Verschlüsselung wird das Salt in die ersten 64 Bytes vor die entstandene verschlüsselte Datei gehängt. Damit ist die gewünschte Dateistruktur erreicht:

- das Salt, mit dem die Verschlüsselung vorgenommen wurde, liegt unverschlüsselt in den ersten 64 Bytes vor. Es handelt sich hier hauptsächlich um den Header der Hostdatei.
- der Header des Volume sowie die Nutzlast sind ab Byteoffset 64 vorhenden. Unter Verwendung des Salt lässt sich die Datei in TrueCrypt einhängen und öffnen. Es ist unter gewissen Umständen sogar möglich, Dateien aus dem virtuellen Laufwerk zu entnehmen oder neue Dateien einzufügen, ohne die polyglotte Struktur zu zerstören. Nicht möglich ist das in den Fällen, in denen Offsets, File- oder Chunkgrößen im Header der Hostdatei codiert sind, da diese nach Veränderungen in der Datenstruktur des Crypt Volumes nicht mehr korrekt sind.
- das Programm zur Verarbeitung der Hostdatei ignoriert die Daten des Volumes und fährt dahinter mit der Datenverarbeitung fort. Die Hostdatei wird ihrem Format entsprechend korrekt geöffnet.

Während die Vorgehensweise zum Entschlüsseln des Crypt Volumes und der kombinierten Datei sowie zum Verschlüsseln der bearbeiteten polyglotten Datei unabhängig davon ist, in welchen Dateitypen das TrueCrypt Volume eingefügt wurde, muss zum Einfügen des TrueCrypt Volumes je nach Dateityp der Hostdatei unterschiedlich vorgegangen werden. Deshalb

wurde bei Entwurf der Programmstruktur die Ver- und Entschlüsselungs Routine vor die Klammer gezogen und zentralisiert.

Als Sprache zur Realisierung des Tools wurde Python gewählt. Zum einen ist Python laut dem PYPL-Ranking 2024 mit einem Marktanteil von fast 30% die meistgenutzte Programmiersprache [20]. Sie ist intuitiv und leicht erlernbar und weist eine geringe Komplexität auf, was die Erweiterbarkeit im Sinne der geplanten Aufgabenstellung vereinfacht. Zudem ist Python plattformübergreifend einsetzbar.
Durch Bibliotheken wie subprocess lässt sich problemlos externe Software wie beispielsweise tchunt einbinden, um den Nutzer direkt darüber zu informieren, ob das erstellte Polyglott von bekannten Tools erkannt wird. Auch die Arbeit auf Byte-Ebene wird von Python nativ gut unterstützt, und weitere Bibliotheken ermöglichen eine einfache Erstellung eines Tools mit GUIs und einem CLI.
Die Architektur des Tools orientiert sich an den grundlegenden Prinzipien der objektorientierten Programmierung und einer an MVC angelehnten Struktur, wobei das „Model“ durch die Beschaffenheit der nötigen, verwendeten Daten in diesem Tool keine prominente Rolle erhält. Zur visuellen Darstellung der folgenden Erläuterungen siehe Abbildung 3.1.

Die Rolle des Controllers übernimmt der Programmteil Engine, welcher alle anderen Programmteile verknüpft und steuert. Das beinhaltet das Auswählen des User-Interfaces, die Entgegennahme der Dateipfade, das Umwandeln in einen Bytestream, das Laden der entsprechenden Plugins und das Weiterreichen der Bytestreams von den entsprechenden bearbeitenden Teilen des Tools. Final wird hier auch der Bytestream der polyglotten Datei in das Format der Hostdatei gebracht und gespeichert.

Für die Interaktion abseits eines Aufrufs über ein einfaches CLI Kommando stehen auch ein geführtes Text User Interface und ein grafisches Interface zur Verfügung. Die Auswahl wird durch eine Änderung in den Optionen und einen Programmneustart oder durch das Ändern des Parameters in der config Datei getroffen. Um Redundanzen zu vermeiden und allen anderen Programmteilen einheitliche Schnittstellen zum Interface zur Verfügung zu stellen, wurde eine abstrakte Master-UI Klasse eingeführt. Diese beinhaltet die Methoden, die zur Kommunikation nötig sind. Die Implementierungsde-

tails werden den abgeleiteten Klassen überlassen. Das Text Interface führt anhand von Auswahlmöglichkeiten durch den Prozess und Untermenüs. Das grafische Interface bietet eine Drag and Drop-Lösung zur Eingabe der Dateien und einen ein- und ausblendbaren Log, um den User über den Programmablauf zu Informieren.

Der Umfang der Informationen bezüglich des Programmablaufs ist davon abhängig, ob ein User den Verbose Modus aktiviert hat oder nicht. Zur Verarbeitung der Interaktionen wurde ein Eventmanager und entsprechende Handler verwendet. Diese können ebenfalls von allen implementierten UI verwendet werden.

Erweiterbar ist das Tool durch die Verwendung von Plugins. Der Plugin-Loader ist als Programmteil für die Zusammenarbeit mit der Schnittstelle verantwortlich. Die Entscheidung zur Nutzung eines Plugin-Loaders beruht darauf, dass eine „installierte“ oder gepackte Version des Programms nun durch das Erstellen eines Datei-Typ-Plugins und dem Speichern im entsprechenden Ordner erweitert werden kann, ohne dass Änderungen im Programmcode selbst nötig sind.

Hierzu wird von einer eingelesenen Datei der Typ anhand der Dateiendung erkannt und ein entsprechendes Plugin gesucht. Um eine Erweiterbarkeit auch für die Art der Plugins zu ermöglichen, wird von neuen Plugins das Präfix in der Form „btp_“ erwartet. Nach dem Unterstrich folgt der Dateityp, wobei mindestens einer angegeben sein muss. Dateitypen, welche mit verschiedenen Endungen arbeiten, wie beispielsweise .jpg/ .jpeg wird damit begegnet, dass auch die btp_ Datei entsprechend benannt wird. So würde für das .jpg Plugin eine Namenserweiterung auf „btp_jpg_jpeg“ ermöglichen, dass dasselbe Plugin verwendet wird, unabhängig von der verwendeten Erweiterung der Hostdatei.

Das vom System geladene Plugin enthält nun als Parameter die Bytestreams der Hostdatei und des TrueCrypt Volumes. Hier werden die Bytestreams so zusammengefügt, dass die kombinierte Datei dem Dateiformat der Hostdatei entspricht und so auch geöffnet werden kann. Ein Mounten des TrueCrypt Volumes ist nach diesem Schritt noch nicht möglich. Dieser Schritt ist spezifisch für den Dateitypen der Hostdatei. Deshalb wird das

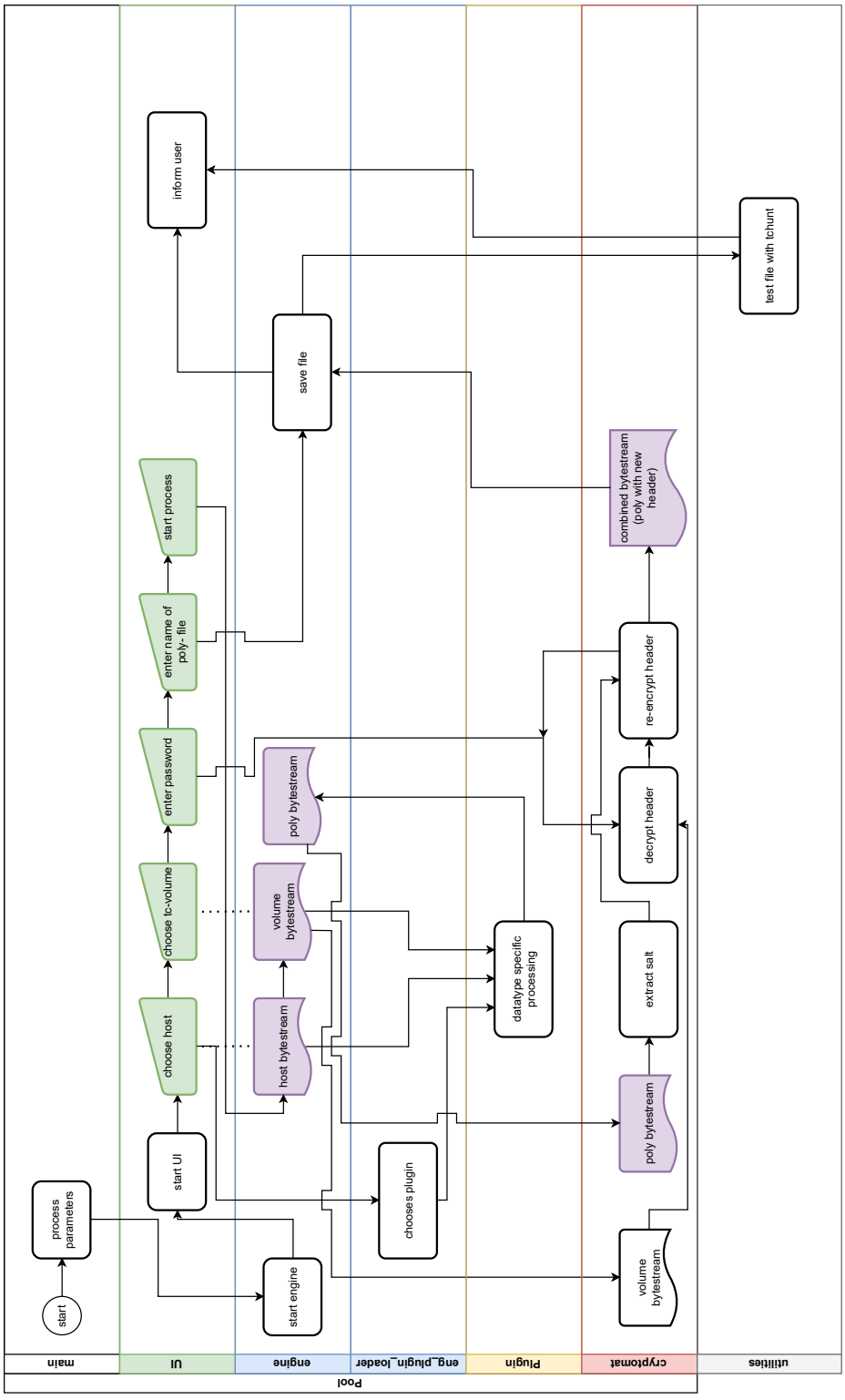

Abbildung 3.1: Swimlane Polyglott-Tool Buttertoast

in ein Plugin ausgelagert. Der nächste Schritt allerdings ist universell und findet daher wieder im Hauptprogramm statt.

Nachdem das Plugin der Engine den Bytestream der kombinierten Datei übergeben hat, ruft diese den Programmteil auf, welcher für die kryptologische Problemlösung verantwortlich ist. Der „Cryptomat“ genannte Programmteil nimmt das von User angegebene Passwort, die vom Plugin kombinierte Datei und die ursprüngliche verschlüsselte TrueCrypt-Datei entgegen und führt die notwendigen kryptographischen Operationen durch. Der Cryptomat stellt sicher, dass der Header der TrueCrypt-Datei entschlüsselt, manipuliert und erneut verschlüsselt wird, sodass eine konsistente und funktionale polyglotte Datei entsteht. Die zugrunde liegenden kryptographischen Prozesse nutzen den AES-XTS-Modus mit SHA-512 zur Ableitung von Schlüsseln und implementieren sowohl die Entschlüsselung als auch die erneute Verschlüsselung des Volumen-Headers.

Nach der kryptographischen Bearbeitung durch den Cryptomat übergibt dieser der Engine den generierten Bytestream. Dort wird der Bytestream in das spezifische Ursprungsdateiformat zurückkonvertiert und die endgültige polyglotte Datei erzeugt. Die Datei wird im vom User angegebenen Pfad gespeichert.

4 Test des Tools und Ausblick

Konkret implementiert wurde die Funktionalität, ein Polyglott aus einem TrueCrypt Volume und einer PNG-Datei herzustellen. Das dateiformatspezifische Skript führt dabei die Schritte aus, die in Abschnitt 2 im Absatz über PNG-Dateien dargestellt sind: Das Skript sucht das Ende des IHDR-Chunks des PNG und merkt sich die Position. Der neue Chunk wird zusammengestellt, indem die Länge der aus dem TrueCrypt Volume ab Byteoffset 64 einzufügenden Daten ermittelt wird. Dieser Wert wir in vier Bytes codiert, in den anschließenden vier Bytes der Name des Chunks eingefügt und die TrueCrypt Daten angehängt. Schließlich wird die invertierte CRC-Checksumme berechnet und ebenfalls an den Chunk angehängt.

Die gesamte polyglotte Datei wird dann erstellt, indem zunächst der Anfang des PNG bis zum Ende des IHDR-Chunks übernommen wird. Auf 64 Bytes wird mit Nullen aufgefüllt, um das Salt komplett zu haben. Schließlich wird der zusammengefügte Chunk und der Rest des PNG hinzugefügt.

Zum Testen des Tools wurden von der Seite https://de.freepik.com/ 20 verschiedene PNG-Bilder mit einer Größe zwischen 500 KB und 4 MB heruntergeladen. Es wurden drei verschiedene TrueCrypt Volumes mit 300 KB, 10 MB und 100 MB Volumengröße erstellt und Testdateien in die TrueCrypt Volumes eingefügt. Die mit dem Tool erzeugten Dateien konnten alle sowohl als PNG-Bild geöffnet als auch als TrueCrypt Volume gemounted werden. Keine der entstandenen Dateien war korrupt.

Die entstandenen Polyglotte wurden daraufhin überprüft, ob TCHunt oder Binwalk die Struktur erkennt. [...]
Das Tool enthält neben dem ausführlich getesteten Dateiformat PNG auch Plugins zur Bearbeitung der folgenden Dateitypen:

- BMP
- HTML
- ICO
- WAV
- ZIP

Die Bearbeitung dieser Dateitypen funktioniert einwandfrei, die Ergebnisse wurden aber nicht so ausführlich getestet wie für den Hauptdateitypen PNG. Weitere Dateitypen können durch Erstellen eines Plugins, das zwei Bytestreams entgegen nimmt und daraus eine kombinierte Datei gemäß der in diesem Paper dargelegten Vorgaben erstellt und zurückgibt, implementiert werden. Erweiterbarkeit ist gewährleistet, wenn das entsprechende Skript unter der angegebenen Namenskonvention btp_ + Dateityp gespeichert wird.

Weitere Erweiterungsmöglichkeiten bestehen darin, wie bereits in Abbildung 3.1 skizziert, die entstandenen Polyglotte direkt durch TCHunt überprüfen zu lassen. Außerdem können im Cryptomat noch Erweiterungen vorgenommen werden, mit denen zusätzliche Hash- bzw. Verschlüsselungsalgorithmen ermöglicht werden.

Die Arbeitsaufteilung erfolgte so, dass durch Mitarbeit an den anderen Workstreams jeder der Teammitglieder einen weitestgehend gleichen Anteil am Endprodukt hat. Die Hauptverantwortlichkeiten für die Arbeitspakete werden in Tabelle 4.1 aufgeführt.

Matthias Ferstl Fabian Kozlowski Stefan Leippe Malte Muthesius

Tabelle 4.1: Aufteilung der Verantwortlichkeiten im Projektteam

Literatur

[1] "Heise: 31C3-Die Angriffe auf Verschluesselung durch NSA und GCHQ." (), Adresse: https://www.heise.de/news/31C3-Die-Angriffe-auf-Verschluesselung-durch-NSA-und-GCHQ-2507004.html. (abgerufen am 11.11.2024).

[2] "Jacob Applebaum, Laura Poitras: Cryptography." (), Adresse: https://media.ccc.de/v/31c3_-_6258_-_en_-_saal_1_-_201412282030_-_reconstructing_narratives_-_jacob_-_laura_poitras#t=1986. (abgerufen am 11.11.2024).

[3] "BSI: TrueCrypt Studie." (), Adresse: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Truecrypt/Truecrypt.pdf?__blob=publicationFile&v=1. (abgerufen am 11.11.2024).

[4] "TrueCrypt Doku." (), Adresse: https://archive.ph/QkbW1. (abgerufen am 11.11.2024).

[5] "PNG File Format." (), Adresse: http://www.libpng.org/pub/png/spec/1.2/PNG-Contents.html. (abgerufen am 11.11.2024).

[6] "TIFF - Image File Format." (), Adresse: https://docs.fileformat.com/image/tiff/. (abgerufen am 11.11.2024).

[7] "Dr. Bobbs: BMP-Format." (), Adresse: https://drdobbs.com/architecture-and-design/the-bmp-file-format-part-1/184409517. (abgerufen am 11.11.2024).

[8] "Microsoft: BMP-Header." (), Adresse: https://learn.microsoft.com/en-us/windows/win32/gdi/bitmap-header-types?redirectedfrom=MSDN. (abgerufen am 11.11.2024).

[9] "W3C: SVG-Grafik Format." (), Adresse: https://www.w3.org/TR/SVG11/. (abgerufen am 11.11.2024).

[10] "WAV - Waveform Audio File Format." (), Adresse: https://docs.fileformat.com/audio/wav/. (abgerufen am 11.11.2024).

[11] "MP4 File Format." (), Adresse: https://docs.fileformat.com/video/mp4/. (abgerufen am 11.11.2024).

[12] "Active @ file recovery: MP4 Signature Format." (), Adresse: https://www.file-recovery.com/mp4-signature-format.htm. (abgerufen am 11.11.2024).

[13] "MP4ra: Boxes." (), Adresse: https://mp4ra.org/registered-types/boxes. (abgerufen am 11.11.2024).

[14] "Apple: Quick-Time Atom." (), Adresse: https://developer.apple.com/documentation/quicktime-file-format/movie_data_atom. (abgerufen am 11.11.2024).

[15] "zziplib - zip format." (), Adresse: https://zziplib.sourceforge.net/zzip-parse.html. (abgerufen am 11.11.2024).

[16] "Microsoft: ICO-Files." (), Adresse: https://learn.microsoft.com/en-us/previous-versions/ms997538(v=msdn.10). (abgerufen am 11.11.2024).

[17] "HTML - Kommentare." (), Adresse: https://wiki.selfhtml.org/wiki/HTML/Tutorials/Element,_Tag_und_Attribut#Kommentare. (abgerufen am 11.11.2024).

[18] "Microsoft: dll-files." (), Adresse: https://learn.microsoft.com/de-de/troubleshoot/windows-client/setup-upgrade-and-drivers/dynamic-link-library. (abgerufen am 11.11.2024).

[19] "TrueCrypt Format Specification." (), Adresse: https://www.truecrypt71a.com/documentation/technical-details/truecrypt-volume-format-specification/. (abgerufen am 11.11.2024).

[20] "PYPL-Ranking." (), Adresse: https://pypl.github.io/PYPL.html. (abgerufen am 11.11.2024).