HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

BACHELORARBEIT

Herr

Lukas Riedt

Erstellung eines Autopsy-Moduls zum Erkennen und Carven von Dateien von Kryptowährungs-Wallets

Mittweida, 2022

Fakultät Angewandte Computer- und
Biowissenschaften

BACHELORARBEIT

Erstellung eines Autopsy-Moduls zum Erkennen und Carven von Dateien von Kryptowährungs-Wallets

Autor:
Herr

Lukas Riedt

Studiengang:
Allgemeine und Digitale Forensik

Seminargruppe:
FO19w4-B

Erstprüfer:
Prof. Ronny Bodach

Zweitprüfer:
Stefan Schildbach, M.Sc.

Einreichung:
Mittweida, 27.09.2022

Verteidigung/Bewertung:
Mittweida, 2022

Faculty Angewandte Computer- und
Biowissenschaften

BACHELOR THESIS

Creation of an autopsy module to detect and carve files from cryptocurrency wallets

author:
Mr.

Lukas Riedt

course of studies:
Allgemeine und Digitale Forensik

seminar group:
FO19w4-B

first examiner:
Prof. Ronny Bodach

second examiner:
Stefan Schildbach, M.Sc.

submission:
Mittweida, 27.09.2022

defence/ evaluation:
Mittweida, 2022

Bibliografische Beschreibung:

Riedt, Lukas:

Erstellung eines Autopsy-Moduls zum Erkennen und Carven von Dateien von Kryptowährungs-Wallets. - 2022. – 7 S. Verzeichnisse, 58 S. Inhalt, 39 S. Anhänge

Mittweida, Hochschule Mittweida, Fakultät Angewandte Computer- und Biowissenschaften, Bachelorarbeit, 2022

Referat:

Zum Erkennen und Carven von Dateien von Kryptowährungs-Wallets wird ein Autopsy-Modul erstellt und evaluiert. Dieses soll bei Electrum, Exodus, Firefly, Wasabi, Monero, Ledger Live, Guarda und den Browser-Erweiterungen Coinbase, Binance und MetaMask auf Untersuchungsdatenträgers nach relevanten Dateien suchen, auch wenn diese gelöscht sind oder das Dateisystem defekt ist. Dazu wird das in Autopsy verwendete PhotoRec um Signaturen erweitert.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADB	Android Debug Bridge
APFS	Apple File System
AVD	Android Virtual Device
ext	Extended File System
HFS+	Hierarchical File System Plus
MFT	Master File Table
NTFS	New Technology File System
VDI	VirtualBox Disk Image
VM	Virtuelle Maschine
VMDK	Virtual Machine Disk
VMM	Virtual Machine Monitor

1 Einleitung

Als erste Kryptowährung wurde der Bitcoin 2008 von einer Person oder Gruppe unter dem Pseudonym Satoshi Nakamoto in der Veröffentlichung „Bitcoin: A Peer-to-Peer Electronic Cash System“[1] als Zahlungsmittel vorgestellt.[2] Ein dezentrales Transaktionssystem soll sicher und unabhängig von Finanzinstitutionen und staatlichen Eingriffen sein. [2]

Ab dem Jahr 2012 nimmt die Bedeutung des Bitcoin stetig zu, wobei sein Wert im Laufe der Jahre starken Kursschwankungen unterliegt.[3] Mittlerweile akzeptieren viele Unternehmen wie z.B. Lieferando, Steam oder Expedia den Bitcoin zum Begleichen ihrer Leistungen.[4] Neben El Salvador (2021) führt auch die Zentralafrikanische Republik (2022) den Bitcoin als zusätzliches gesetzliches Zahlungsmittel ein.[5] Die Bedeutung als Zahlungsmittel ist aber, zumindest derzeit, eher als gering einzuschätzen. Vielmehr ist Kryptogeld beliebtes Anlage- und Spekulationsobjekt.[6]

Der Erfolg des Bitcoins führt dazu, dass ab dem Jahr 2013 vermehrt weitere Kryptowährungen entstehen. [3] Nach Angaben des Finanzportals „investing.com“ sind es im September 2022 fast 10.000 unterschiedliche Kryptowährungen. [7]

Neben der Zunahme der allgemeinen Nutzung gewinnen Kryptowährungen gerade im Bereich der Kriminalität an Bedeutung. Sie werden bei vielen illegalen Aktivitäten wegen der möglichen Anonymität als Zahlungsmittel verwendet. So wird nach vielen Ransomware-Attacken das Lösegeld in Kryptowährungen gefordert oder illegale Dienstleistungen im Darknet damit bezahlt. Damit steigt der Analysebedarf bei Polizeibehörden.

Ziel dieser Arbeit ist es, Hinweise auf die Benutzung von Wallet-Softwares von Kryptowährungen auf einem Untersuchungsdatenträger zu finden und auch gelöschte relevante Dateien wiederherzustellen. Analysiert werden dabei Desktop Wallets, Browser-Erweiterungen und Mobile Wallets. Wallets werden zum Verwahren, Empfangen und Versenden von Kryptowährungen genutzt.

Angeregt durch das Landeskriminalamt Rheinland-Pfalz, Dezernat Cybercrime, wurde im Rahmen eines Praktikums die Erstellung eines Moduls für die forensische Software Autopsy und die Erweiterung des Quellcodes des File Carving Tools PhotoRec um neue Signaturen eingeleitet. Mit dem Modul soll den Ermittlern geholfen werden, indem Dateien der Wallet-Softwares automatisiert gesucht und aufbereitet werden. Darüber hinaus sollen gelöschte Wallet-Dateien oder Dateien defekter Dateisysteme gecarvt werden.

In Vorbereitung auf diese Arbeit wurde eine erste Version des Moduls mit den Wallet-Softwares Electrum, Exodus, Firefly, Wasabi, Monero und Ledger Live erstellt. Diese Version wird in Bezug auf verschiedene Betriebssysteme und verschiedene Versionen der

Wallet-Softwares evaluiert. Die Erkenntnisse aus der Evaluation führen zusammen mit der Erweiterung des Moduls um die Guarda Wallet und die Browser-Erweiterungen Binance, Coinbase und MetaMask zur zweiten Version des Moduls.

In den folgenden Kapiteln werden zunächst Grundlagen zu Kryptowährungen, File Carving, Virtualisierung und den genutzten Programmen erklärt. Darüber hinaus werden andere Arbeiten betrachtet, die sich mit ähnlichen Fragestellungen beschäftigen.

Danach wird das Vorgehen bei der Suche nach Dateien der Wallet-Softwares, beim Erstellen von Signaturen für PhotoRec, beim Erstellen des Autopsy-Moduls und beim Vorbereiten der virtuellen Maschinen für die Evaluation beschrieben.

Anschließend werden die Ergebnisse der Evaluation und die daraus folgenden Anpassungen des Moduls dargestellt. Mit den vom Modul gefundenen Dateien werden daraufhin die Zugriffsmöglichkeiten auf die Wallets überprüft.

Zum Abschluss werden die Ergebnisse der Arbeit diskutiert und zusammengefasst.

In den Anlagen befinden sich die Signaturen und gesuchten Dateien beider Versionen des Moduls und die Ergebnisse der Evaluation tabellarisch dargestellt.

2 Grundlagen

In diesem Kapitel werden zunächst Grundlagen zu Kryptowährungen, File Carving, Virtualisierung und den verwendeten Programmen erklärt. Darüber hinaus wird die vorgestellte Arbeit mit ähnlichen Ansätzen verglichen.

2.1 Kryptowährungen

Kryptowährungen wie Bitcoin, Ethereum oder Monero basieren auf Dezentralisierung und asymmetrischer Verschlüsselung und kommen ohne eine zentrale Instanz aus, die die Währung kontrolliert. Wichtige Komponente ist das Peer-to-Peer-Netzwerk, in dem alle Teilnehmer zugleich Konsument und Bereitsteller der Dienste sind. Mit diesem Netzwerk wird verhindert, dass die Kryptowährung mehrfach ausgegeben werden kann. [1] [8, S. 2]

Ausgetauscht werden die Währungen über Transaktionen. Diese sind dabei öffentlich für jeden einsehbar und erfolgen direkt zwischen Sender und Empfänger. Die Transaktionen werden in Blöcken gespeichert. Im Prozess des Minings werden die Blöcke verifiziert und danach zu einer Kette aneinander gehängt, der Blockchain. Der Schutz eines Blocks vor Veränderungen geschieht mittels einer Hashfunktion. Dabei wird der Block mit dem Hashwert des vorherigen Blocks verbunden. Eine Manipulation eines Blocks führt daher zu einer Inkonsistenz des Blocks selbst und aller folgenden Blöcke. [8, S. 15ff., S. 27ff, S. 233ff.] [1]

Die Kontrolle über die Korrektheit der in Blöcken stehenden Transaktionen basiert auf einem Mehrheitssystem. Die Blockchain wird bei allen Nutzern des Netzwerks gespeichert. Eine abweichende Blockchain im Netzwerk stellt eine Konsistenzverletzung dar und wird von der Mehrheit der Netzwerkteilnehmer als ungültig deklariert. Für einen Angriff auf die Blockchain ist somit die Kontrolle über mehr als die Hälfte der Netzwerkteilnehmer nötig. [1] [8, S. 229ff.]

Bei der Abwicklung einer Transaktion kommt asymmetrische Kryptographie zum Einsatz. Der private Schlüssel ist notwendig, um Zugriff auf die Kryptowährung und Adressen zu erhalten. Der öffentliche Schlüssel wird aus dem privaten Schlüssel berechnet. Der öffentliche Schlüssel muss nicht geheim gehalten werden, da dessen Berechnung aus dem privaten Schlüssel irreversibel ist. [8, S. 55ff.]

Die Transaktion einer Kryptowährung wird mit dem privaten Schlüssel vom Sender signiert. Die Verifizierung vom Empfänger der Zahlung erfolgt mit dem öffentlichen Schlüssel des Senders. [1]

Eine Adresse wird mit einer Hashfunktion aus dem öffentlichen Schlüssel generiert. [8, S. 56]

Für die Durchführung der Transaktion muss vom Absender eine Gebühr abgegeben werden. Ein Miner stellt Ressourcen zur Verfügung, die gebraucht werden, um die Transaktionen abzuwickeln. Im Gegenzug erhält er die Transaktionsgebühren und, solange die maximale Anzahl (im Falle von z.B. Bitcoin) Bitcoins noch nicht im Netzwerk ist, neue Coins. [8, S. 26ff.]

Bei der Transaktionsabwicklung werden die Transaktionen in Blöcken zusammengefasst, welche von den Minern validiert werden müssen. Damit wird ein Konsens im Netzwerk gefunden. Für die Verifizierung der Transaktionen muss der Miner eine komplexe Aufgabe lösen, für die viel Rechenleistung benötigt wird. Mehr als die Hälfte der Miner müssen die Daten als korrekt anerkennen, damit der Block als gültiger Block in die Blockchain integriert werden kann. [8, S. 26ff.]

Wallets (deutsch: Brieftasche) sind Software- oder Hardwarelösungen zum Verwahren, Senden und Empfangen von Kryptowährungen. In der Wallet befindet sich dabei nicht die Kryptowährung an sich, sondern nur die öffentlichen und privaten Schlüssel des Benutzers und die Adressen auf der Blockchain, die den Zugriff auf die Kryptowährungen ermöglichen. Die Kryptowährung selbst befindet sich also nicht in der Wallet, sondern in der Blockchain. [9]

Von Wallets gibt es verschiedene Arten mit unterschiedlichen Sicherheitsniveaus. Bei Software-Wallets werden die Informationen dabei in einer App oder Software lokal auf dem Gerät gespeichert. Diese gibt es als eigenständige Programme oder als Browser-Erweiterungen. Bei Online-Wallets werden die Daten auf den Servern des Wallet-Anbieters gespeichert. Die Sicherheit dieser beiden Wallet-Typen ist generell geringer. Bei Online-Wallets ist sie von den Vorkehrungen des Anbieters abhängig. Hardware-Wallets verwahren die Schlüssel auf externen, portablen USB-Stick-ähnlichen Geräten. Diese müssen nur zur Nutzung angeschlossen werden und bieten mit der offline-Speicherung der Schlüssel einen höheren Schutz, haben dafür aber in Anschaffungskosten und Funktionsumfang Nachteile. [10]

Paper Wallets setzen auf ein hohes Maß an Sicherheit, indem die privaten Schlüssel auf einem Blatt Papier gespeichert werden. Erstellen kann man sie auf Webseiten wie „Bitaddress.org“ [11]. Auf diesen kann man zufällige öffentliche und private Schlüssel erzeugen, die ausgedruckt werden können. [12]

Bei der Erstellung des Wallets werden die privaten Schlüssel aus einem Seed berechnet. Ein Seed ist eine Folge von zwölf oder 24 Wörtern, die dem Mnemonic Code BIP-39 Standard [13] entsprechen. Der Seed wird bei Erstellung eines Wallets generiert und dient als Backup des Wallets. [8, S. 95]

Mit Hilfe der Erfahrungen der Ermittler des LKA Rheinland-Pfalz wurden folgende Wallet-Softwares als besonders relevant erachtet und daher in der ersten Version des Moduls genauer betrachtet:

- Electrum [14]: ein Wallet für Bitcoin.
- Exodus [15]: eine Wallet-Software, mit der verschiedene Kryptowährungen verwaltet werden können.
- Firefly [16]: ein Wallet für IOTA.
- Wasabi [17]: ein Bitcoin-Wallet mit dem Fokus auf Privatsphäre, nutzt dafür den Onion Router (TOR).
- Monero [18]: hiermit wird die Kryptowährung Monero verwaltet.
- Ledger Live [19]: Software zur Verbindung der Hardware-Wallets von Ledger mit dem Netzwerk. Die Wallet-Software wurde nur installiert und ohne die dazugehörige Hardware betrachtet. Mit Ledger lassen sich mehrere Kryptowährungen verwalten.

Erweitert wurde das Modul in der zweiten Version um:

- Guarda Desktop Wallet [20]: ein Wallet zur Verwaltung verschiedener Kryptowährungen
- Guarda Online Wallet [21]: Browser App der Guarda Wallet
- Guarda Mobile App [20]: Mobilversion der Guarda Wallet

und die Browser Erweiterungen

- Coinbase [22]
- Binance [23]
- MetaMask [24]

Ermittlungsansätze bei einer gefundenen Adresse einer Kryptowährung ergeben sich bei Transaktionen. Nur diese hinterlassen Spuren, die von Ermittlern verfolgt werden können. Besonders wertvoll sind hierbei Transaktionen an eine Adresse einer Exchange-Plattform, da der Beschuldigte hier seine Kryptowährung in Fiatgeld, worunter man staatlich herausgegebene Währungen versteht, oder in andere Kryptowährungen umtauscht [25]. Mit einer Anfrage an eine Exchange-Plattform können bei der Ermittlung Informationen über die Identität des Senders der Transaktion erlangt werden.

2.2 File Carving

File Carving ist die Wiederherstellung von Dateien, für die keine Dateisysteminformationen mehr verfügbar sind. Diese forensische Methode wird hauptsächlich bei gelöschten Dateien oder einem defekten Dateisystem angewendet. Dabei werden die Rohdaten nach bestimmten Merkmalen, die eine Datei ausmachen, durchsucht. Wenn eine Datei gefunden wurde, wird diese extrahiert und separat abgespeichert. Da beim File Carving keine Dateisysteminformationen betrachtet werden, können Metadaten und der Dateiname nicht wiederhergestellt werden. [26, S. 153f.]

Das bekannteste Konzept ist hierbei das Auslesen der Magic Bytes. Dies sind mehrere Bytes einer Datei, die immer am Anfang und am Ende der Datei stehen und deren Beginn bzw. Ende anzeigen. Sie werden auch Header und Footer genannt. Diese Magic Bytes sind einzigartig für den jeweiligen Dateityp. [26, S. 153]

Darüber hinaus ist es wichtig zu verstehen, wie Dateien in einem Dateisystem strukturiert sind. Ein Dateisystem ist in Cluster aufgeteilt. Diese bestehen aus einem oder mehreren Sektoren. Sektoren sind die kleinste adressierbare Speichereinheit auf Ebene der Festplattenfirmware. Dateien werden im Dateisystem über die Cluster adressiert. Die Belegung der Cluster wird vom Dateisystem verwaltet. [27, S. 126] [27, S. 117]

Wenn eine Datei auf dem Datenträger angelegt wird, wird für die Datei neben dem Anlegen von Metadaten und Verzeichniseinträgen Speicherplatz auf dem Datenträger reserviert, in den die Rohdaten der Datei geschrieben werden. Der reservierte Speicherplatz ist damit allokiert. Speicherplatz, der für keine Datei reserviert ist, ist unallokiert. [27, S. 173]

Wird eine Datei gelöscht, gilt der Bereich auf dem Datenträger, auf dem sich die Datei befindet, wieder als unallokiert. Die Verweise des Dateisystems auf den Speicherort der Rohdaten der Datei werden dabei nicht unbedingt komplett entfernt, sondern nur als unbenutzt markiert, können aber durch neue Einträge überschrieben werden. Dabei werden die Dateiinhalte meist nicht gelöscht. In diesen unallokierten, nicht mehr vom Dateisystem belegten Bereichen, findet das File Carving statt. [27, S. 178, S. 203] [26, S. 153]

Dateisysteme, die häufig auftreten sind bei Linux Versionen das Extended File System (Ext, Ext2, Ext3, Ext4) und bei Mac HFS+ oder APFS. [28]

Bei aktuellen Windows-Versionen wird in der Regel NTFS (New Technology File System) als Dateisystem genutzt. Für jede Datei wird ein MFT(Master File Table)-Eintrag angelegt. Jeder MFT-Eintrag hat mehrere Attribute. Dazu gehört auch das Data-Attribut. Dieses kann resident oder non-resident sein. Bei einem residenten Data-Attribut befinden sich die Daten der Datei direkt im Data-Attribut. Wenn die Datei jedoch zu groß ist, wird das Data-Attribut non-resident. Das heißt, dass die Daten in externe Cluster geschrieben werden und im Data-Attribut nur auf den Speicherort der Daten verwiesen wird. [26, S. 203ff.]

Dies stellt beim Carven eine Herausforderung dar, da sich bei MFT-Einträgen mit residenten Data-Attributen die Rohdaten der Datei nicht am Clusteranfang befinden und viele Carving-Programme nur diese nach Header-Signaturen durchsuchen.

Beim File-Carving gibt es drei größere Konzepte. Das Header/Footer Carving oder Header/maximum-file-size Carving, das File Structure Based Carving und das Content-based Carving.

Beim Header/Footer-Carving wird nach den Magic Bytes von Dateien gesucht. Findet man eine Header-Signatur am Beginn eines Clusters, wird gecarvt, bis der entsprechende Footer oder ein neuer Header gefunden wurde. Die Daten zwischen dem Header und dem Footer werden als Datei angenommen und gespeichert. [29]

Das Header/maximum-file-size-Carving beginnt ebenfalls mit der Suche nach einer Header-Signatur am Dateianfang. Das Ende der Datei ist hierbei jedoch nicht durch eine Signatur definiert, sondern durch die Dateigröße. Diese ist für den Dateityp vorgegeben oder wird aus der Datei ausgelesen (auch Header/Embedded Length Carving). [29]

Beim File Structure based Carving werden neben den Signaturen Metadaten der internen Struktur der Datei ausgelesen und als zusätzliche Informationen für das Carven gewonnen. [29]

Beim Content-based Carving wird die Inhaltsstruktur oder andere Merkmale der Datei erfasst und zum Carven genutzt. Dabei spielen zum Beispiel Text- und Spracherkennung oder die Entropie eine Rolle. [28]

Ein Problem beim Carven von Dateien ist die Fragmentierung. Eine fragmentierte Datei liegt nicht komplett am Stück auf dem Datenträger, sondern ist in mehrere Stücke aufgeteilt, die an unterschiedlichen Stellen des Datenträgers liegen. Zwischen den einzelnen Fragmenten der Datei befinden sich häufig andere Dateien. Liegt zwischen dem ersten und dem zweiten Fragment einer Datei beispielsweise eine weitere Datei, wird die erste Datei bis zum Start der zweiten Datei gecarvt. Da sich am Start der zweiten Datei eine

neue Header-Signatur befindet, wird das Carven der ersten Datei abgeschlossen und mit dem Carven der zweiten Datei begonnen. Dies führt dazu, dass die erste Datei nicht komplett wiederhergestellt wird, sondern nur das erste Fragment. [30]

Ein weiteres Problem stellen eingebettete Dateien dar. Dies sind zum Beispiel Bilddateien, die in Word- oder PDF-Dateien eingebunden sind. Die eingebettete Bilddatei wird nicht als Teil der Word- oder PDF-Datei angesehen, sondern als eigenständige Datei gecarvt. Die Word- oder PDF-Datei wird dann wie eine fragmentierte Datei nur teilweise gecarvt. [31]

Besondere Bedeutung beim File Structure based Carving haben Json-Dateien. Json steht für für JavaScript Object Notation. Dieses Format wird von vielen Anwendungen zum Speichern von Daten genutzt. Die Daten sind dabei in Schlüssel-Wert-Paaren angeordnet. [32]

Gängige Programme, die zum Carven eingesetzt werden können, sind zum Beispiel Scalpel, Foremost oder PhotoRec. Letzteres soll hier genauer betrachtet werden. [28]

2.3 PhotoRec

PhotoRec ist ein in C geschriebenes, freiverfügbares open-source File Carving Tool, das von Christophe Grenier 2002 veröffentlicht wurde. Seitdem wird es stetig weiterentwickelt und umfasst heute über 480 Dateiendungen. Es wird als Begleitprogramm von TestDisk unter der GNU General Public License verbreitet. [33]

PhotoRec bietet zwei Möglichkeiten, neue Signaturen hinzuzufügen. Einerseits können neue Signaturen mit Hilfe einer Datei erstellt werden. Diese Datei muss „photorec.sig“ heißen und sollte im Nutzerverzeichnis („C:\Users\<User>\“) oder im selben Verzeichnis wie die PhotoRec-Executable abgelegt werden. In dieser Datei können Dateiendung, Offset der Signatur und die Signatur/der Header angegeben werden. Ein Beispiel eines Eintrags einer „photorec.sig“-Datei ist in Abbildung 1 dargestellt. [33, S. 43ff.]

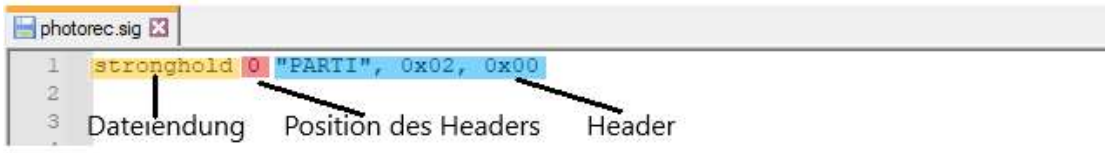

Abbildung 1: Beispiel einer photorec.sig mit Dateiendung, Position des Headers und Header

Um auch weitere Aspekte des Carvings wie zum Beispiel die Erkennung des Footers und das Definieren der Dateigröße bestimmen zu können, muss PhotoRec auf Quellcodeebene editiert werden. Dafür stellt PhotoRec eine Anleitung zur Verfügung [34].

Der Quellcode von TestDisk & PhotoRec wurde in der Betaversion 7.2 am 06.04.2022 von der GitHub-Seite von TestDisk heruntergeladen. [35]

PhotoRec ist das in Autopsy verwendete Carving Tool. Dazu gibt es in Autopsy ein Modul, das PhotoRec auf die unallokierten Bereiche des Datenträgers anwendet und die daraus gecarvten Dateien in einem Ordner ablegt. [36]

2.4 Autopsy ®

The Sleuth Kit® (TSK) ist eine in C/C++ geschriebene Bibliothek und Sammlung von Kommandozeilentools, die auf die Analyse von Dateisystemdaten spezialisiert ist. [37]

Autopsy® ist eine Open-Source Software, die eine grafische Benutzeroberfläche bietet, die im Hintergrund The Sleuth Kit® und weitere Tools nutzt [38]. Damit stellt Autopsy eine nutzerfreundliche Erweiterung zu The Sleuth Kit® dar, die mit Modulen in Java oder Jython erweitert werden kann [39]. Im Rahmen der Arbeit wurde Autopsy in der Version 4.19.3 verwendet.

Als Schnittstelle zwischen Autopsy und The Sleuth Kit® dient das Java Native Interface (JNI). Damit kann Autopsy auf die Bibliotheken von The Sleuth Kit® zugreifen. [40]

Jython ist eine Java-Implementation von Python, die es ermöglicht, für das in Java geschriebene Autopsy ein in Python geschriebenes Modul zu erstellen. Jython sieht von der Syntax her aus wie Python, konvertiert aber das Programm in Java Bytecode. Jython wurde in der Version 2.7.0 verwendet. [41]

In Autopsy lassen sich vier verschiedene Modultypen erstellen. Zum einen Ingest-Module, welche auf einen Datenträger angewendet werden, um spezielle Aufgaben auszuführen. Die verbleibenden drei Modultypen sind Report-Module, Content-Viewer und Result-Viewer.
Report-Module erstellen in der Regel einen Ergebnisbericht anderer Module, können aber auch zur Analyse genutzt werden. Content-Viewer sind grafische Module, die die Dateien in definierter Weise darstellen (s. Abb. 2, unten rechts: grün markiert). Result-Viewer zeigen Informationen über eine Reihe von Dateien oder über den Inhalt eines Ordners (s. Abb. 2, oben rechts: gelb markiert). [42]

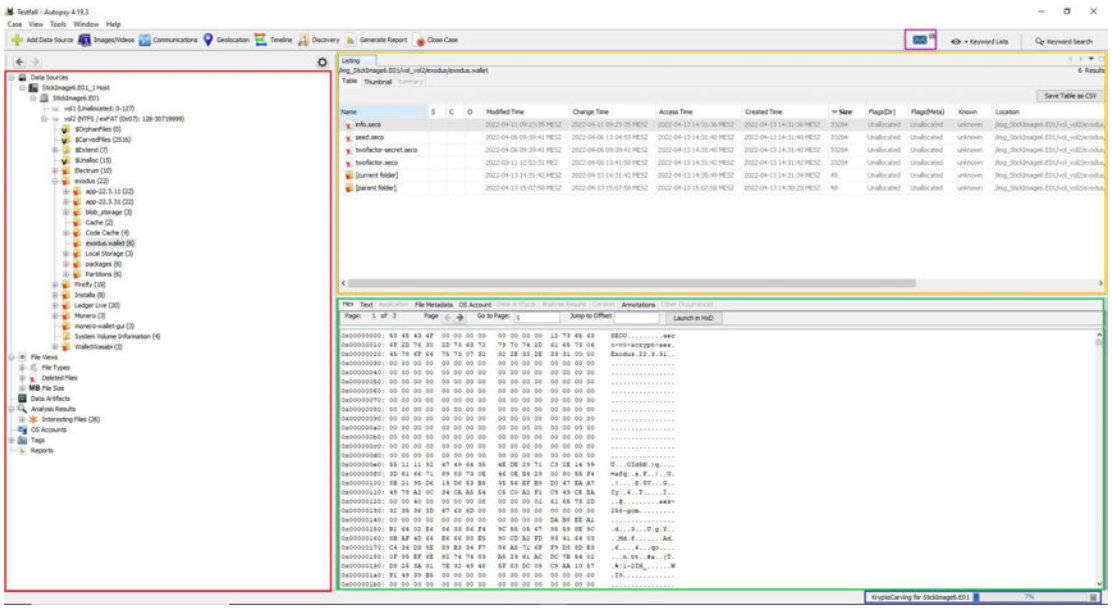

Abbildung 2: Grafische Oberfläche von Autopsy

Ingest-Module übergeben ihre Ergebnisse dem Blackboard, damit sie im Ergebnisbaum dem Nutzer zur Verfügung gestellt werden (s. Abb. 2, links: rot markiert). Dabei wird zwischen File-Ingest-Module und Data-Source-Ingest-Module unterschieden. File-Ingest-Module werden auf alle Dateien des Datenträgers angewendet. Data-Source-Ingest-Module lesen einen Datenträger ein und können die Falldatenbank nutzen, um spezielle Dateien des Datenträgers auszuwählen. [43]

Beim vorgestellten Modul handelt es sich um ein Data-Source-Ingest-Modul. Dieses benötigt eine Factory-Klasse und eine Ingest-Modul-Klasse. In Ersterer werden grundlegende Informationen wie der Modulname, die Version und eine Beschreibung des Moduls angegeben. Diese Klasse erlaubt Autopsy ebenfalls, Instanzen der Modulklasse anzulegen, die die Analyse durchführt. In der Ingest-Modul-Klasse befinden sich die Methoden *startUp()* und *process()*. In der *startUp()*-Methode werden Setup und Konfigurationen durchgeführt. In der *process()*-Methode findet die eigentliche Analyse statt. [43]

Ein eigenes Modul kann in Autopsy über das Menü (Tools → Python Plugins) hinzugefügt werden. An diese Stelle („C:\Users\<User>\AppData\Roaming\autopsy\pyhton_modules\“) muss für das Modul ein Ordner erstellt werden, in dem sich das Jython-Skript befindet. [41]

2.5 Virtualisierung

Unter Virtualisierung versteht man „Methoden zur Abstraktion von Ressourcen mit Hilfe von Software“. [44, S. 270]

Virtualisierung wird in dieser Arbeit für die Erstellung von Testumgebungen für die Evaluation des Moduls genutzt. Bei der Evaluation wird getestet, wie gut das Modul mit Untersuchungsdatenträgern verschiedener Betriebssysteme und bei unterschiedlichen Versionen der Wallet-Softwares funktioniert. Als Testumgebungen werden dazu sogenannte Virtuelle Maschinen (kurz VM) erstellt, auf denen die Wallet-Softwares installiert werden. Für die Betriebssysteme Windows, Linux und Mac wird als Virtualisierungsumgebung VirtualBox verwendet und für Android der Android Virtual Device (AVD) Manager von Android Studio.

Eine virtuelle Maschine verhält sich wie ein vollwertiger Computer in einer abgeschotteten Umgebung [44, S. 270] [45, S. 231]. Die eigentliche Kontrolle über die reale Maschine und seine Hardwareressourcen liegt jedoch beim Hostsystem [44, S. 270]. Die VM bekommt als Gastsystem die Hardwareressourcen (Prozessorleistung, Arbeitsspeicher, Grafikleistung) des Hostsystems vom Virtual Machine Monitor (VMM), auch Hypervisor genannt, zugewiesen [45, S. 235ff.]. Damit kann auf einem Hostbetriebssystem ein anderes bzw. mehrere Gastbetriebssysteme gleichzeitig ausgeführt werden. [46]

Von der virtualisierenden Software, dem Hypervisor, gibt es zwei Typen.

Der Typ-1-Hypervisor sitzt direkt über der Hardware und entspricht dem Betriebssystem [47, S. 581]. Diese Variante benötigt Hardware, die die Virtualisierung unterstützt [44, S. 278]. Dieser Ansatz wird auch Bare-metal-Ansatz genannt [48].

Ein Typ-2-Hypervisor läuft als normale Software auf einem Hostbetriebssystem [47, S. 581]. Diese Variante hat im Vergleich zu Typ-1 Performancenachteile [48]. Diese Variante wird auch Hosted-Ansatz genannt [44, S. 278].

In dieser Arbeit wird der zweite Typ verwendet.

Das Modul soll mit Windows-, Linux- und Mac-Systemen getestet werden. Dazu wird Oracle VM VirtualBox (kurz VirtualBox) in der Version 6.1.34 verwendet.

VirtualBox ist eine von Oracle unter der GNU General Public License v2 [49] verbreitete Open-Source-Virtualisierungs-Software, die für die Hostbetriebssysteme Windows, macOS, Linux und Oracle Solaris verfügbar ist [50] [51]. Als Gastsystem wird eine große Zahl verschiedener Betriebssysteme unterstützt. [52]. VirtualBox ist eine Virtualisierungs-Software vom Typ-2-Hypervisor. [53]

Zwei der gebräuchlichen Dateiformate für virtuelle Festplatten in VirtualBox sind VDI (VirtualBox Disk Image) und VMDK (Virtual Machine Disk). VirtualBox bietet mit Sicherungspunkten die Möglichkeit, Systemzustände zu speichern und die virtuelle Maschine später wieder in diesen Zustand zurückzusetzen. [54]

VBoxManage ist die Kommandozeilenschnittstelle von VirtualBox, deren Funktionen über die der grafischen Benutzeroberfläche von VirtualBox hinausgehen. Mit VBoxManage können virtuelle Maschinen erstellt und Änderungen an diesen vorgenommen werden und virtuelle Festplatten in andere Formate umgewandelt werden. [55]

Die Erstellung der für die Evaluation verwendeten VMs wird in Kapitel 3.4 beschrieben.

Zur Evaluation der Android Apps der Wallet-Softwares werden virtuelle Android Geräte genutzt. Hier kommt der Android Virtual Device (AVD) Manager zum Einsatz, mit dem zum Beispiel Apps getestet werden können.

Der AVD Manager ist Teil von Android Studio. Das ist die offizielle Entwicklungsumgebung für Android Apps. Android Studio wird in der Version 4.0.1 genutzt. [56]

Die Android Debug Bridge (adb) ist ein Kommandozeilentool zur Kommunikation mit Android-Geräten. Mit dieser können Apps installiert oder eine Shell des Geräts geöffnet werden. [57]

Als Host- und Untersuchungs-PC dient in der Evaluation ein Windows 10 Home PC der Version 21H1 mit 16 GB RAM, einem Intel® Core™ i7-6700K Prozessor und einer NVIDIA GeForce GTX 1080 Grafikkarte.

2.6 Themenverwandte Arbeiten

Der Fokus vieler Ansätze auf diesem Gebiet liegt auf der Identifizierung von Schlüsseln und Adressen und auf der Verfolgung von Transaktionen und der Ermittlung der Nutzer. Mit der Identifikation der Nutzung von Kryptowährungs-Wallets beschäftigt sich Alexander Gehrig in seiner Masterarbeit „Strategie zum Nachweis der Nutzung von Kryptowährungen“ [58]. Diese betrachtet Artefakte in Windows und Linux für Wallets der Kryptowährungen Bitcoin und Monero. Zum Nachweis der Wallets werden zum Beispiel Registry, Jump Lists und Shimcache betrachtet und die ausgewählten Artefakte bewertet.

C. Cartes erweitert in seiner Masterarbeit „Digitale Währungen – Identifikation, Evaluation und automatisierte Suche digitaler Spuren in forensischen Sicherungen“ [59] die forensische Software Bulk_extractor um Plugins, die automatisiert Hinweise auf die Nutzung digitaler Währungen finden.

Im Unterschied zum vorgestellten Ansatz werden andere Artefakte bei anderen Wallet-Softwares betrachtet. Auch wenn C. Cartes mittels regulärer Ausdrücke nach Dateien der Wallet-Softwares sucht, befassen sich die vorgenannten Autoren im Gegensatz zur vorliegenden Arbeit nicht mit dem Carven von Dateien von Kryptowährungs-Wallets.

Autopsy ermöglicht in seinem mitgelieferten Modul „Interesting Files Identifier“ eine Suche nach Kryptowährungs-Wallets. Das Modul umfasst deutlich mehr Wallet-Softwares, sucht aber lediglich nach den exe-Dateien der Wallet-Softwares über deren Namen. Das Modul beinhaltet keine Möglichkeit, gecarvte Dateien zu untersuchen und bereitet neben den exe-Dateien keine weiteren Dateien der Wallet-Softwares auf. [60]

Mark McKinnon hat ein Autopsy-Modul entwickelt, das Informationen der Atomic Wallet Crypto Currency App ausliest. Dabei werden Information der Connection-Logs und der History-Files der Wallet-App ausgelesen. [61]

3 Methoden

In diesem Abschnitt wird dargestellt, wie nach relevanten Dateien gesucht wird, die Hinweise auf Wallet-Softwares liefern und wie neue Signaturen in PhotoRec hinzugefügt werden. Darüber hinaus wird die Erstellung des Autopsy-Moduls und die Vorbereitung der virtuellen Maschinen für die Evaluation beschrieben.

3.1 Relevante Dateien

Um sich einen Überblick der Dateistruktur der einzelnen Wallet-Softwares zu verschaffen und darin relevante Dateien zu finden, werden die zwölf bereits erwähnten Wallet-Softwares auf einem Windows Betriebssystem bzw. in den Browsern Google Chrome und Firefox installiert und die entstandene Dateistruktur betrachtet. Die Android App wird auf einem virtuellen Android-Gerät installiert. Tabelle 1 listet die Versionen der betrachteten Wallet-Softwares auf.

Tabelle 1: Versionen der betrachteten Wallet-Softwares

Wallet-Software	**Version**
Electrum	4.1.5 (Windows Installer)
Exodus	22.3.11
Firefly	1.3.3
Wasabi	1.1.13
Monero	0.17.3.1
Ledger Live	2.40.2
Guarda Desktop Wallet	1.0.20
Guarda Online Wallet	Keine Angabe
Guarda App	2.62.0

Binance Browser Extension	2.13.7
Cionbase Browser Extension	2.29.1
MetaMask Browser Extension	10.18.3

Damit der Einfluss von Transaktionen auf Änderungen in den Dateien der Wallet-Softwares nachvollzogen werden kann, werden Adressen in den Wallets hinzugefügt und in den Testnetzwerken der Software-Wallets Transaktionen durchgeführt.

Bei der Suche liegt der Fokus auf Wallet-Dateien und Dateien, die Information wie Adressen enthalten. Dabei fällt auf, dass Wallets in keiner einheitlichen Form vorliegen. Sie haben teilweise eigene Dateiformate, sind Json-Dateien oder sind in mehreren Dateien abgelegt. Andere Dateien wie Config- oder Installationsdateien können genutzt werden, um Hinweise auf eine Installation der Wallet-Software zu finden.

Electrum, Firefly, Wasabi, Exodus, Ledger Live und Guarda legen viele ihrer Dateien im Verzeichnis „C:\Users\<User>\AppData\Roaming\“ ab. Bei Monero ist der Speicherort der Wallet-Dateien frei wählbar.

Im Folgenden werden die relevanten Dateien der einzelnen Wallet-Softwares dargestellt. Die Auflistung enthält Dateien, die von PhotoRec gecarvt werden und Dateien, für die keine Signaturen in PhotoRec hinzugefügt werden können und somit nicht gecarvt werden.

3.1.1 Electrum

Bei Electrum-Wallet-Dateien muss zwischen verschlüsselt und unverschlüsselt unterschieden werden. Ein unverschlüsseltes Electrum-Wallet liegt in einer Json-Struktur vor. Es ist jedoch nur unverschlüsselt, wenn bei der Wallet-Erstellung kein Passwort angegeben wird. Dies ist in den seltensten Fällen zu erwarten. Mit der Json-Struktur weist die Wallet-Datei einige Merkmale auf, die ein Carven ermöglichen.

Das Carven der verschlüsselten Electrum-Wallet-Datei ist nicht möglich. Eine verschlüsselte Wallet kann über den Pfad „C:\Users\<User>\AppData\Roaming\Electrum\wallets\“ dennoch gefunden werden, wenn sie noch im Dateisystem enthalten ist.

Ein weiterer Typ von Dateien, der bei Electrum zu finden ist, sind Zertifikate. Diese befinden sich im Ordner „C:\Users\<User>\AppData\Roaming\Electrum\certs\“. Sie weisen eine

gut auffindbare Struktur auf, welche immer mit „-----*BEGIN CERTIFICATE*-----“ beginnt und immer mit „-----*END CERTIFICATE*-----“ endet.

Auch die Electrum-Config-Datei weist eine Json-Struktur auf, die eine Erkennung der Datei ermöglicht.

Ebenfalls interessant ist die Datei „recent_servers“. Diese beinhaltet eine Liste der von Electrum zuletzt genutzten Servern, jedoch keine Struktur, die sich zum Carven eignet.

3.1.2 Exodus

Das Wallet von Exodus liegt in Form von vier „seco“-Dateien vor. „seco“ steht für „Secure Container“. Aufgrund der Signatur „SECO“ am Anfang der Datei sind Secure Container für ein Carving geeignet. Die von Exodus erstellten Secure Container beinhalten zusätzliche Informationen in der Headersignatur, die ein Schluss auf die Anwendung der Seco-Datei als Exodus-Wallet nahelegen (Signatur: „*seco-v0-scrypt-aes Exodus <version>*“, <version> steht dabei für die Exodus-Version).

Zusätzlich befindet sich im Wallet-Verzeichnis die Datei „passphrase.json“. Es handelt sich um eine Json-Datei mit den Schlüsseln „*passphrase*“ und „*system*“. Sie ist mit Hilfe der Struktur und den Schlüsselwörtern für ein Carving geeignet.

Darüber hinaus eignet sich „exodus.conf.json“ als Json-Datei mit der Dateistruktur und Schlüsselwörtern zum Carven.

Bei Asar handelt es sich um Archiv-Dateien, die zum Verpacken von Quellcode genutzt werden. Die im Installationsverzeichnis („C:\Users\<User>\\AppData\Local\exodus\“) von Exodus befindliche Datei „app.asar“ beinhaltet am Dateianfang eine Json-Struktur mit Informationen wie dem Produktnamen von Exodus oder die Homepage, die die Datei für ein Carving interessant macht.

Des Weiteren besteht die Möglichkeit, im Dateisystem nach der „Exodus.exe“ zu suchen.

Viele weitere Informationen sind in Level-DB-Datenbanken abgelegt. Diese eignen sich wegen ihrer Organisationsstruktur nicht für ein zielgerichtetes Carving.

3.1.3 Firefly

Firefly-Wallets liegen in Form von Dateien mit der Endung „stronghold“ vor. Diese Dateien haben den Header „*PARTI*“ (vollständig in hexadezimaler Schreibweise: 0x50 41 52 54 49 02 00). Mit dieser Signatur ist die Datei gut für ein Carving geeignet.

Weitere Dateien im Installationsverzeichnis („C:\Users\<User>\AppData\Local\Pograms\Firefly\“) sind die „app-update.yml“, die sich aufgrund von Schlüsselwörtern zum Carven eignet, und die „app.asar“, die am Dateianfang Informationen zu Firefly enthält. Auch die „Firefly.exe“ kann über die Dateisystemstrukturen gesucht werden.

3.1.4 Wasabi

Die Wallet-Dateien von Wasabi liegen im Json-Format vor. Diese sind aufgrund von Struktur und Schlüsselwörtern gut für das Carving geeignet. Dasselbe gilt für die Config-Datei.

Über das Dateisystem lassen sich ebenfalls die „Transactions.dat“-Dateien in den Ordnern „BitcoinStore/Mempool/“ und in „ConfirmedTransactions/2/“ finden. Diese beinhalten Informationen zu den durchgeführten Transaktionen.

3.1.5 Monero

Im Monero Wallet-Verzeichnis befinden sich drei Dateien. Eine Datei ohne besonderen Dateityp, wobei es sich um eine Wallet-Datei handelt, eine „.address.txt“, in der Adressen abgelegt sind, und eine „.keys“-Datei. Diese drei Dateien eignen sich nicht zum Carven, da sie keine Signaturen oder andere Strukturen beinhalten, die bei allen Dateien diesen Typs gleich sind.

Dateien im Installationsverzeichnis von Monero, die sich zum Carven eignen, sind die „unins000.dat“ und die „ReadMe.htm“. Die „monero-wallet-gui.exe“ kann mit Hilfe des Dateisystems gefunden werden.

3.1.6 Ledger Live

Ledger Live wird nur installiert und nicht in Verbindung mit der dazugehörigen Hardware genutzt. Wallet-Dateien konnten daher nicht analysiert werden, da sie sich nur auf der Hardware befinden.

Dateien im Installationsverzeichnis von Ledger Live, die sich aufgrund von Schlüsselwörtern zum Carven eignen, sind die „app-update.yml“ und die „app.asar“. Die „Ledger Live.exe“ kann mit Hilfe des Dateisystems gefunden werden.

3.1.7 Guarda

Von Guarda wird die Guarda Desktop-Version, die Android App und die Online-Wallet betrachtet.

Die Guarda Desktop-Version nutzt keine speziellen Dateiformate oder Json-Dateien als Wallet-Datei. Vielmehr empfiehlt das Programm dem Nutzer, regelmäßig Backup-Dateien zu erstellen. Diese sind verschlüsselte Textdateien, deren Name standardmäßig mit „-guarda-backup.txt“ endet. Der Inhalt, der in diese Backup-Dateien geschrieben wird, befindet sich auch im Verzeichnis „AppData\Roaming\Guarda\IndexedDB\https_guarda.co_0.indexeddb.leveldb“ in einer Level-DB-Datenbank. Beide sind für das Carven nicht geeignet, weil sie keine Signaturen enthalten, die für die Dateitypen einzigartig sind. Die Backup-Datei kann aber im Dateisystem gesucht werden.

Dateien im Installationsverzeichnis der Guarda Wallet, die sich zum Carven eignen, sind die Datei „app.asar“ und die Datei „app-update.yml“.

Auch die Guarda Online Wallet nutzt die „-guarda-backup.txt“-Dateien. Diese Wallet-Informationen wurden auch im Verzeichnis „AppData\Local\Google\Chrome\User Data\Default\IndexedDB\https_guarda.com_0.indexeddb.leveldb“ in einer Level-DB-Datenbank gefunden.

Bei der Android App werden die Backup-Dateien ebenfalls verwendet. Hier enden sie mit „-guarda-multiwallet-backup.txt“. Die Dateien der Guarda Android App befinden sich im Verzeichnis „/data/data/com.crypto.multiwallet“.

3.1.8 Browser-Erweiterungen

Die Browser-Erweiterungen Binance, Coinbase und MetaMask werden mit den Browsern Google Chrome und Firefox betrachtet.

Beim Chrome Browser befinden sich Dateien der Erweiterung im Verzeichnis „AppData\Local\Google\Chrome\User Data\Default\Extensions“.

Wallet-Informationen der Chrome Erweiterung sind in Level-DB-Datenbanken abgelegt. Diese befinden sich im Verzeichnis „AppData\Local\Google\Chrome\User Data\Default\Local Extension Settings“ und eignen sich nicht zum Carven.

Die Dateien der Erweiterungen sind beim Mozilla Firefox Browser in xpi-Container gepackt. Diese befinden sich im Verzeichnis „AppData\Roaming\Mozilla\Firefox\Profiles\<Profile>\extensions\“. Die im Verzeichnis „AppData\Roaming\Mozilla\Firefox\Profiles\

<Profile>\“ enthaltene Datei „extensions.json“ enthält Informationen zu den installierten Erweiterungen und dient damit hauptsächlich zur Detektion dieser beim Firefox Browser.

3.1.8.1 Binance

Dateien, nach denen bei der Binance Chrome Erweiterung gesucht werden kann, sind die „background.html“ und die Datei „manifest.json“. Diese eignen sich aufgrund von Schlüsselwörtern zum Erkennen im Modul.

In Firefox kann nach der Erwähnung der Binance Wallet in der Datei „extensions.json“ gesucht werden.

3.1.8.2 Coinbase

Bei der Coinbase Chrome Erweiterung kann nach den Dateien „index.html“ und „manifest.json“ gesucht werden, welche Schlüsselwörter enthalten, die sich für die Erkennung eignen.

Die Coinbase Browser-Erweiterung ist für Firefox nicht verfügbar.

3.1.8.3 MetaMask

Dateien der MetaMask Chrome Erweiterung, die gesucht werden können, sind die Dateien „home.html“ und „manifest.json“. Auch diese eignen sich aufgrund von Schlüsselwörtern.

In Firefox kann neben dem Durchsuchen der Datei „extensions.json“ noch nach Dateien gesucht werden, deren Name mit „@metamask.io.xpi“ endet.

3.2 Erstellen neuer Signaturen für PhotoRec

Um neben den im Dateisystem vorhandenen relevanten Dateien auch Dateien ohne Dateisysteminformationen zu finden, kommt Carving zum Einsatz. Vieler dieser Dateien werden von der aktuellen PhotoRec-Version nicht gecarvt oder wie z.B. viele Json-Dateien nicht zuverlässig gecarvt. Um dieses Problem zu beheben, wird der Quellcode des in Autopsy verwendeten PhotoRec um 22 neue Signaturen erweitert. Zur Erstellung einer Signatur für PhotoRec muss eine Datei mit dem Namen *file_<Dateiendung>.c* (<Dateiendung> als Platzhalter für die Dateiendung) im „*src*“-Verzeichnis von PhotoRec erstellt werden und Hinweise zu dieser in den Dateien *src/file_list.c* und *Makefile.am* hinzugefügt werden. Eine genauere Anleitung und ein Beispiel dazu ist im Kapitel 3.2.10 zu finden.

Die verwendeten Signaturen sind für die gesuchten Dateien teilweise nicht eindeutig. Viele falsch positive Ergebnisse, die beim Carving von PhotoRec entstehen, werden später im Autopsy-Modul nicht als Dateien von Kryptowährungs-Wallets erkannt, da im Autopsy-Modul detailliertere Merkmale der Dateien überprüft werden.

Die Header-Signaturen werden ebenfalls in die Datei „photorec.sig“ geschrieben. Diese Datei kann verwendet werden, wenn die Benutzung des editierten PhotoRec nicht möglich ist. Dazu muss die Datei im Benutzerverzeichnis abgelegt werden („C:\Users\<User>\“). Bei den Signaturen der „photorec.sig“ sind jedoch keine Footer für die Dateitypen angegeben und es sind nicht alle Dateitypen verfügbar. Daher wird die Verwendung zu schlechteren Ergebnissen führen.

Ein Problem, das bei der „photorec.sig“ und anderen Signaturen ohne definierten Footer auftritt, ist, dass häufig bis zum nächsten Header gecarvt wird und die entstehende Datei deutlich größer als die gesuchte Datei wird.

Nachfolgend werden die erstellten Signaturen für jede Wallet-Software beschrieben. Die gewählten Dateiendungen sind dabei in Klammern angegeben. Eine Auflistung aller erstellten Signaturen mit Header, Footer und weiteren Informationen ist in „Anlagen, PhotoRec Modulversion 1 und Anlagen, Signaturen PhotoRec Modulversion 2“ zu finden.

3.2.1 Electrum

Für Electrum werden Signaturen für unverschlüsselte Wallets (mit der Endung .electrumWallet), Zertifikate (.cert) und für die Config-Datei (.electrumConfig) hinzugefügt.

Da eine unverschlüsselte Electrum-Wallet als Json-Datei vorliegt, können zum Carven die ersten 26 Zeichen der Datei genutzt werden („*{\r\n “addr_history“: {\r\n*“). Diese enthal-

ten noch keine variablen Felder, sondern nur Zeichen der Json-Struktur und Schlüssel und sind damit zum Carven geeignet.

Für verschlüsselte Wallet-Dateien kann keine Signatur für PhotoRec erstellt werden, da die Datei keine Header oder Footer-Informationen beinhaltet und die Json-Strukturen durch die Verschlüsselung nicht mehr auslesbar sind.

Für die Zertifikate werden „-----*BEGIN CERTIFICATE*-----“ als Header und „-----*END CERTIFICATE*-----“ als Footer verwendet. Neben Header und Footer wird das komplette Zertifikate, das von Electrum verwendet wird, später im Autopsy-Modul abgeglichen. Ein Problem dieser Dateien ist, dass sich die Zertifikate häufig ändern.

Bei der Electrum-Config wird ebenfalls der Beginn der Datei („*{\r\n \“auto_connect\“:* “) als Header verwendet und das Ende der Datei („*\r\n}*“) als Footer.

3.2.2 Exodus

Für Dateien der Exodus-Wallet-Software werden Signaturen für Secure Container (mit der Dateiendung .seco), die Exodus passphrase.json (.exodusPassphrase) und die Exodus config-Datei (.exodusConfig) erstellt.

Secure Container, die für die Exodus-Wallets genutzt werden, beginnen mit der Signatur „*SECO*“. Die Dateigröße für diese Dateien wurde auf 33284 Bytes festgelegt, da die beobachteten Secure Container von Exodus immer diese Größe haben. In PhotoRec wird lediglich „*SECO*“ als Header definiert. Die Überprüfung, ob es sich um einen Secure Container handelt, der von Exodus genutzt wird, erfolgt später im Autopsy-Modul.

Für die im Wallet-Verzeichnis von Exodus befindliche passphrase.json wird der Dateianfang als Header („*{\n \“passphrase\“: \“*“) und das Dateiende als Footer („*\n}\n*“) verwendet.

Für die Exodus-Config-Datei wird ebenfalls der Dateianfang als Header („*{\n “meta“: {\n “configVersion“: “*“) und das Dateiende als Footer („*\n}\n*“) verwendet.

3.2.3 Firefly

Dateien von Firefly, für die in PhotoRec Signaturen hinzugefügt werden, sind die Wallet-Dateien und die „app-update.yml“.

Wallet-Dateien von Firefly haben die Dateiendung „.stronghold“. Diese beginnen mit der Signatur („*PARTI*“, 0x02, 0x00), welche in PhotoRec als Header-Signatur angegeben wird.

Die Datei „app-update.yml“ aus dem Installationsverzeichnis von Firefly wird mit dem Anfang der Datei („*provider: generic\nurl: 'https://dl.firefly.iota.org/'\npublishAutoUpdate:* “) als Header und dem Dateiende („*publisherName:\n - IOTA Stiftung* “) als Footer als Dateityp mit der Endung „.fireflyAppupdate“ in PhotoRec hinzugefügt.

3.2.4 Wasabi

Die Wasabi Wallet-Datei (Endung .wasabiWallet) und die Config-Datei (Endung .wasabiConfig) sind Json-Dateien. Sie beginnen mit der Signatur „0xEF BB BF“. Das sind die Magic Bytes für die Byte-Reihenfolge-Markierung von UTF-8. Diese und der jeweilige erste Json-Schlüssel werden in PhotoRec für beide als Header definiert. Als Footer dient jeweils das Dateiende („*\n}*“).

Für die Transactions.dat kann keine Signatur erstellt werden. Sie enthält zwar nützliche Informationen wie Bitcoin-Adressen, verfügt aber über keine Strukturen, die sich für ein Carving eignen.

3.2.5 Monero

Für Monero werden Signaturen für die Dateien „unins000.dat“ (mit der Endung .moneroUninsDat) und ReadMe.htm (.html) aus dem Installationsverzeichnis erstellt.

Die „unins000.dat“ wird mit dem Dateianfang als Header gecarvt („*Inno Setup Uninstall Log (b) 64-bit*“, *0x00, 0x00,* „*Monero GUI Wallet*“).

Html sollte eigentlich als Teil von „file_txt.c“ gecarvt werden. Da dies in den Tests mit der Datei „ReadMe.htm“ aber nicht zuverlässig funktioniert, wird „html“ als Dateityp in PhotoRec hinzugefügt. Als Header wird „*<html>*“ verwendet, als Footer „*</html>*“.

Für die Wallet-Datei, die .keys-Datei und die .address.txt können keine Signaturen erstellt werden, da die Dateien weder Header oder Footer, noch Strukturen beinhalten, die als Signatur für das Carving geeignet sind.

3.2.6 Ledger Live

Als Datei von Ledger Live wird neben der nachfolgend erwähnten Asar-Datei lediglich die Datei „app-update.yml“ mit der Dateiendung „.ledgerAppupdate“ mit dem Dateistart als Header („*owner: LedgerHQ\nrepo: ledger-live-desktop\nprovider:* “) und dem Dateiende als Footer hinzugefügt („*updaterCacheDirName: ledger-live-desktop-updater* “).

3.2.7 Guarda

Für Guarda wird lediglich eine Signatur für die Datei „app-update.yml“ (mit der Endung .guardaAppupdate) erstellt. Als Header wird der Beginn der Datei verwendet („*owner: guardaco\nrepo: guarda-electron*“) und als Footer das Ende der Datei („*publisherName:\n - Guardarian OÜ\n*“).

Für die Wallet-Backup-Dateien können keine Signatur erstellt werden, da diese keine Strukturen beinhalten, die sich für das Carven eignen.

3.2.8 Browser Erweiterungen

Für die Chrome-Erweiterung der Binance Wallet und MetaMask wird eine Signatur für die Dateien „background.html“ und „home.html“ (als .html) erstellt. Der Header dieser Datei ist „*<!doctype html>*“ und der Footer „*</html>*“.

Die Datei „index.html“ der Coinbase Erweiterung wird mit der für Monero erstellten html-Signatur gecarvt.

Die Datei „manifest.json“ wird in Chrome von Binance, Coinbase und MetaMask verwendet. Für diese Datei wird in PhotoRec ein Dateityp mit den folgenden sechs möglichen Header erstellt:

- *{\r\n “background“: {*
- *{\n “background“: {*
- *{\r\n “author“:*
- *{\n “author“:*
- *{\r\n “app“: {*
- *{\n “app“: {*

Als Footer dient das Ende der Datei („*\n}*“).

Für die Browser Erweiterungen in Firefox wird eine Signatur für die Datei „extensions.json“ (.firefoxExtensions) erstellt. Als Header wird der Beginn der Datei verwendet („*{“schemaVersion“:*“) und als Footer das Ende der Datei („*]}*“).

3.2.9 Weitere Signaturen

Asar-Archive (mit der Dateiendung .asar) werden bei Firefly, Exodus, Ledger Live und Guarda im Installationsverzeichnis gefunden. Als Header wird die Bytefolge „0x04 00 00 00“, die bei allen betrachteten Asar-Archiven identisch ist, verwendet. Darüber hinaus wird der Header um die Zeichenfolge („*{“files“:{“*“) erweitert, die an Offset 0x10 der Datei den Anfang des dateiinternen Headers darstellt.

Darüber hinaus werden Signaturen für das „Bitcoin Core Wallet“, „MultiBit Bitcoin Wallet“ und das „MultiBit Bitcoin Wallet information file“ auf Basis der von „garykessler.net“ bereitgestellten Signaturliste [62] hinzufügt (s. Anlagen, Signaturen PhotoRec Modulversion 1).

3.2.10 Erstellung einer Signatur in PhotoRec am Beispiel der Electrum-Zertifikate

Für die Erstellung neuer Dateiendungen stellt PhotoRec eine Anleitung zur Verfügung [34]. Der Quellcode von PhotoRec ist auf GitHub verfügbar [35].

Im Folgenden wird beispielhaft die Erstellung einer Signatur in PhotoRec, anhand der Zertifikate von Electrum mit der Dateiendung „cert“, dargestellt.

Zunächst wird die Zeile „*extern const file_hint_t _hile_hint_cert;*“ in der Datei „*src/file_list.c*“ hinzugefügt (s. Abb. 3) und „{ .enable=0, .file_hint=&file_hint_cert },“ in „*file_enable_t array_file_enable[]*“ (s. Abb. 4) ergänzt. Anschließend muss mit der Zeile „*file_cert.c*“ das C-File der neuen Dateiendung in der Datei „*src/Makefile.am*“ hinzugefügt werden.

```
374  extern const file_hint_t file_hint_zcode;
375  extern const file_hint_t file_hint_zip;
376  extern const file_hint_t file_hint_zpr;
377  extern const file_hint_t file_hint_cert;
378  extern const file_hint_t file_hint_stronghold;
379  extern const file_hint_t file_hint_seco;
380  extern const file_hint_t file_hint_exodusPassphrase;
```

Abbildung 3: Ausschnitt der Datei "src/file_list.c"

```
1424  #if !defined(SINGLE_FORMAT) || defined(SINGLE_FORMAT_zcode)
1425    { .enable=0, .file_hint=&file_hint_zcode  },
1426  #endif
1427  #if !defined(SINGLE_FORMAT) || defined(SINGLE_FORMAT_zip)
1428    { .enable=0, .file_hint=&file_hint_zip  },
1429  #endif
1430  #if !defined(SINGLE_FORMAT) || defined(SINGLE_FORMAT_cert)
1431    { .enable=0, .file_hint=&file_hint_cert  },
1432  #endif
1433  #if !defined(SINGLE_FORMAT) || defined(SINGLE_FORMAT_stronghold)
1434    { .enable=0, .file_hint=&file_hint_stronghold  },
1435  #endif
1436  #if !defined(SINGLE_FORMAT) || defined(SINGLE_FORMAT_seco)
1437    { .enable=0, .file_hint=&file_hint_seco  },
1438  #endif
```

Abbildung 4: Ausschnitt von „*file_enable_t array_file_enable[]*“ der Datei „src/file_list.c“

Daraufhin kann die Datei „*file_cert.c*“ erstellt werden. Als Vorlage kann eine Beispieldatei von PhotoRec [63] oder die Datei einer anderen Dateiendung dienen. In der Methode „*register_header_check_cert(file_stat_t *file_stat)*“ (s. Abb. 5 Zeile 42ff.) kann der Header („-----BEGIN CERTIFICATE-----“) mit seiner Länge und seiner Position in der Datei angegeben werden und an die Methode *register_header_check()* übergeben werden (s. Abb. 5 Zeile 47f.). In der Methode „*file_check_cert(file_recovery_t *file_recovery)“* (s. Abb. 5 Zeile 24ff.) kann der Footer (hier „-----END CERTIFICATE-----“) definiert werden und an die Methode „*file_search_footer(file_recovery, cert_footer, sizeof(cert_footer), '\0')*“ (s. Abb. 5 Zeile 29) übergeben werden.

In „const file_hint_t hile_hint_cert={…}“ (s. Abb. 5 Zeile 15ff.) können Dateiendung, Beschreibung und Standardeinstellung zur Wiederherstellung eingestellt werden und die Funktion „*register_header_check_cert“* angegeben werden. In „*header_check_cert()*“ (s. Abb. 5 Zeile 32ff.) wird die „*file_hint_cert.extension*“ und die „*file_check_cert*“, in der der Footer definiert wurde, an „*file_recovery_new*“ übergeben.
In der ersten Zeile „*#if !defined(SINGLE_FORMAT) || defined(SINGLE_FORMAT_cert)*“ muss noch die richtige Dateiendung eingesetzt werden.

Daraufhin kann der Quellcode kompiliert werden. Dazu wird Cygwin genutzt. [64]

```
#if !defined(SINGLE_FORMAT) || defined(SINGLE_FORMAT_cert)
#ifdef HAVE_CONFIG_H3
#include <config.h>
#endif
#ifdef HAVE_STRING_H
#include <string.h>
#endif
#include <stdio.h>
#include "types.h"
#include "filegen.h"

/*@ requires valid_register_header_check(file_stat); */
static void register_header_check_cert(file_stat_t *file_stat);

const file_hint_t file_hint_cert= {
  .extension="cert",
  .description="Certificate",
  .max_filesize=PHOTOREC_MAX_FILE_SIZE,
  .recover=1,
  .enable_by_default=1,
  .register_header_check=&register_header_check_cert
};

static void file_check_cert(file_recovery_t *file_recovery)
{
  const unsigned char cert_footer[25]= {
    "-----END CERTIFICATE-----"
  };
  file_search_footer(file_recovery, cert_footer, sizeof(cert_footer), '\0');
}

static int header_check_cert(const unsigned char *buffer, const unsigned int
buffer_size, const unsigned int safe_header_only, const file_recovery_t
*file_recovery, file_recovery_t *file_recovery_new)
{
  reset_file_recovery(file_recovery_new);
  file_recovery_new->extension=file_hint_cert.extension;
  file_recovery_new->file_check=&file_check_cert;
  return 1;
}

static void register_header_check_cert(file_stat_t *file_stat)
{
  static const unsigned char cert_header[27]=  {
    "-----BEGIN CERTIFICATE-----"
  };
  register_header_check(0, cert_header, sizeof(cert_header), &header_check_cert,
  file_stat);
}
#endif
```

Abbildung 5: Beispiel einer Signatur-Datei von PhotoRec (file_cert.c)

3.3 Erstellen des Autopsy-Moduls

Wie bereits im Grundlagenteil beschrieben, wird für das Modul eine Modul-Factory-Klasse und eine Ingest-Modul-Klasse erstellt.

In der Modul-Factory-Klasse werden allgemeine Informationen zum Modul angegeben (s. Abb. 6).

```
class KryptoCarvingIngestModuleFactory(IngestModuleFactoryAdapter):

    moduleName = "KryptoCarving"

    def getModuleDisplayName(self):
        return self.moduleName

    def getModuleDescription(self):
        return "Module for detection and carving of files from wallet
softwares of cryptocurrencies."

    def getModuleVersionNumber(self):
        return "1.0"

    def isDataSourceIngestModuleFactory(self):
        return True

    def createDataSourceIngestModule(self, ingestOptions):
        return KryptoCarvingIngestModule()
```

Abbildung 6: Beispiel einer Modul-Factory-Klasse eines Autopsy-Moduls

Die Ingest-Modul-Klasse benötigt eine *startUp()*-Methode und eine *process()*-Methode. Darüber hinaus werden Methoden zum Loggen, zum Einlesen und Verarbeiten der Dateien und zum Hinzufügen von Dateien als „Interesting File“ erstellt.

In der *startUp()*-Methode werden Setup und Konfiguration durchgeführt.

In der *process()*-Methode wird die eigentliche Analyse durchgeführt. Hier werden die Dateien eingelesen, mit den Merkmalen der gesuchten Dateien aus dem Abschnitt 3.1 abgeglichen und im Erfolgsfall in Autopsy als „Interesting File“ hinzugefügt.

Beim Einlesen der Images wird der Fokus auf das E01-Format gelegt. Hier ist das direkte Lesen auf der Image-Datei wie bei Raw-Images nicht möglich. In Autopsy sind die Dateien über die Falldatenbank erreichbar. Dies geschieht über unterschiedliche Dateiarten („Filetype“). So können der Inhalt des Dateisystems („Filetype“ 0) und von PhotoRec gecarvte Dateien („Filetype“ 1) erreicht werden. Zugriff auf die Datenbank des Falls und Informationen zu Dateien erhält man mit der Funktion *„findAllFilesWhere(„type =<type> and data_source_obj_id=“+str(dataSource.getId()))*“. Damit werden die Dateiobjekte in eine Liste geschrieben.

Über die „unallocated Blocks“ (Filetype 4) können auch vom Dateisystem unallokierte Bereiche erreicht werden, die zum selbstständigen Carven im Autopsy-Modul genutzt werden und mit der Funktion *AddCarvedFile()* hinzugefügt werden können. Dies wird im erstellten Modul nicht durchgeführt.

Für das Einlesen der zu untersuchenden Dateien gibt es zwei Möglichkeiten. Bei der ersten Variante werden die von der Autopsy-API zur Verfügung gestellten Funktionen genutzt. Mittels *ReadContentInputStream(file)* wird ein Inputstream erstellt und mittels der Funktion *read(buffer)* werden die Daten in den Buffer, ein Byte-Array, geschrieben. Die Daten liegen im Buffer als dezimale Werte von -128 bis 127 (als signed int 8) vor. Mit Hilfe der erstellten Funktionen *bufferToHexBufferarray()*, *bufferToHexBufferlist()*, *bufferToStrBufferarray()* und *bufferToStrBufferlist()* wird der Buffer in hexadezimale Werte oder die Zeichenkettenrepräsentation umgewandelt.

Alternativ kann die Datei auch in eine temporäre Datei geschrieben werden und diese daraufhin mit Standard-Python-Funktionen eingelesen werden.

Im erstellten Modul wird aufgrund dessen, dass für viele Dateien nur der Dateianfang eingelesen werden muss, die erste Variante genutzt.

Die eingelesenen Dateien werden im Folgenden mit den gesuchten Dateien verglichen. Dies wird für Dateien des Dateisystems und von PhotoRec gecarvte Dateien getrennt voneinander durchgeführt, da sich die Suchmuster für die beiden Dateitypen unterscheiden.

Zum Vergleich der Dateien werden verschiedene Methoden genutzt. Dabei wird bei Dateien, für die noch Dateisysteminformationen verfügbar sind, zum Beispiel der Dateiname und der Pfad benutzt. Für Dateien des Dateisystems und gecarvte Dateien werden die Magic Bytes und die Dateiendung überprüft. Darüber hinaus wird nach Schlüsselwörtern in den Dateien und nach der Struktur von Json-Dateien gesucht.

Wenn eine Datei den Vergleichsparametern einer gesuchten Datei entspricht, wird sie in Autopsy als Interesting File hinzugefügt (s. Abb. 2, links: rot markiert: unter „Analysis Results“). Dazu wird die Funktion *AddAsInterestingItem()* erstellt. Ihr werden ein „*fileSetName*“ als Name des Unterordners bei den „Interesting Files“, die IDs der Datei und des Datenträgers, ein Score sowie Beschreibungen zur Datei übergeben.

Die Dateiabschnitte der gesuchten Dateien zum Abgleich mit den eingelesenen Dateien sind in der Klasse *Patterns* in der Datei *Patterns.py* ausgelagert. Jede eingelesene Datei wird mit den Patterns jeder gesuchten Datei verglichen und im Erfolgsfall in Autopsy als „Interesting File“ hinzugefügt.

Das Modul durchsucht ebenfalls alle Textdateien nach Seed-Wörtern. Hier werden alle Textdateien mit einer Seed-Wortliste [65] abgeglichen und in Abhängigkeit von Anzahl und Anteil an Seed-Wörtern als „Interesting File“ hinzugefügt.

Die ProgressBar befindet sich in der grafischen Oberfläche von Autopsy in der unteren rechten Ecke (s. Abb. 2) und gibt den Fortschritt des Moduls an. Sie wird der *process()*-Methode übergeben. Am Anfang des Modul wird sie auf „*indeterminate*“ (= unbestimmt) gesetzt. Nachdem bekannt ist, wie viele Dateien bearbeitet werden, wird die *ProgressBar* auf „*determinate*“ gesetzt. Damit schreitet die *ProgressBar* für jede bearbeitete Datei voran.

In Data-Source-Ingest-Modulen wird empfohlen, eine Möglichkeit einzubauen, das Modul vorzeitig zu beenden. Dies geschieht mit den Zeilen (s. Abb. 7):

```
if self.context.dataSourceIngestIsCancelled():
    return IngestModule.ProcessResult.OK
```

Abbildung 7: Überprüfung, ob der Nutzer das Modul vorzeitig beenden möchte

Diese werden in alle länger andauernden Schleifen eingebaut, damit das Programm sich beendet, wenn der Nutzer es mit dem „x“ in der rechten unteren Ecke beenden möchte.

Zum Abschluss des Moduls wird in Autopsy eine Ingest-Nachricht in die Ingest-Message-Box (s. Abb. 2, oben rechts: pink markiert) ausgegeben. Diese gibt an, ob und wie viele Dateien gefunden wurden. Eine Ingest-Nachricht wird mit der Funktion *createMessage* erstellt. Dabei müssen *messageType*, *source*, *subject* und Details der Nachricht in HTML, ohne <html>-Tags, angegeben werden. Mit „*IngestService.getInstance(). postMessage(message)*“ wird die Nachricht der Ingest-Message-Box übergeben.

Es gibt einige Dateien, für die keine Signaturen erstellt werden konnten. Diese werden folglich nicht gecarvt und nur als „Interesting File“ in Autopsy hinzugefügt, wenn die Dateisystemstrukturen der Dateien noch vorhanden sind. Dabei handelt es sich um diese Dateien:

- Electrum: verschlüsseltes Wallet
- Exodus.exe
- Wasabi: Transactions.dat
- Firefly.exe
- Monero-wallet-gui.exe
- Monero Wallet: Dateien im Wallet-Verzeichnis
- Ledger Live.exe
- Guarda: guarda-backup.txt / guarda-multiwallet-backup.txt
- MetaMask: @Metamask.io.xpi
- Exodus-App: RKStorage

Alle vom Modul gesuchten Dateien sind in „Anlagen, gesuchte Dateien Modulversion 1“ und „Anlagen, gesuchte Dateien Modulversion 2“ zu finden.

Die erste Version des Moduls umfasst die Wallet-Softwares Electrum, Exodus, Firefly, Wasabi, Monero und Ledger Live.

Die zweite Version des Moduls besteht aus den in Kapitel 4.2 aufgeführten Anpassungen nach der Evaluation und der Erweiterung des Moduls um Dateien der Guarda Wallet, Browser Erweiterungen und der Apps von Electrum, Exodus und Guarda.

3.4 Evaluation des Autopsy-Moduls

In diesem Kapitel wird die Erstellung der Testumgebungen und die Umwandlung der Images der Testumgebungen in für Autopsy lesbare Formate behandelt.

Zum Testen der Wallet-Softwares werden virtuelle Maschinen verwendet. Dazu werden für die Betriebssysteme Windows, Linux und macOS jeweils Maschinen in VirtualBox genutzt.

Von den virtuellen Maschinen wird jeweils für die Suche nach Dateien im Dateisystem und für das Carven von Dateien Sicherungspunkte erstellt. Diese Systemzustände werden geklont. Die VDI- bzw. VMDK-Datei der geklonten virtuellen Maschine wird anschließend mit VBoxManage in ein Raw-Image umgewandelt (s. Abb. 8) und in Autopsy eingelesen.

```
vboxmanage.exe clonehd <Speicherort der virtuellen Festplatte> <Speicherort der Ausgabedatei>
--format raw
```

Abbildung 8: Befehl zum Unwandeln einer VDI- oder VMDK-Datei in eine Raw-Datei

Für virtuelle Android-Geräte wurde der AVD Manager von Android Studio genutzt.

3.4.1 Linux

Als Ausgangsbasis zum Testen der Wallet-Softwares auf einem Linux-Betriebssystem wird Debian 11.3.0 Cinnamon [66] verwendet.

Dazu wird eine neue VM in VirtualBox erstellt. Dies geschieht über das Menü (Maschine → Neu) des Oracle VM VirtualBox Managers. Hier können nun Name, Speicherort der virtuellen Maschine auf dem Hostsystem, Betriebssystem der virtuellen Maschine und die Version des Betriebssystems festgelegt werden. Anschließend kann die Größe des Hauptspeichers bestimmt und mit dem Erstellen einer virtuellen Festplatte die virtuelle Maschine erzeugt werden.

Nun muss noch über das Menü (erstellte VM auswählen → Maschine → Ändern) im Bereich „Massenspeicher“ die Iso-Datei der oben erwähnten Linux-Version in ein virtuelles Laufwerk eingelegt werden. Nach dem Start der virtuellen Maschine („Starten“) wird die Iso-Datei aus dem virtuellen Laufwerk gestartet. Das Betriebssystem kann jetzt auf der virtuellen Festplatte installiert werden.

Für eine komfortablere Nutzung der virtuellen Maschine werden die VirtualBox GuestAdditions installiert [67].

Auf der fertiggestellten Testumgebung werden alle oben erwähnten Wallet-Softwares installiert. An dieser Stelle wird ein Sicherungspunkt der virtuellen Maschine erstellt. Ein zweiter Sicherungspunkt wird erstellt, nachdem Dateien der Wallet-Softwares gelöscht sind. Dieser dient zum Testen, welche Dateien gecarvt und vom Modul erkannt werden. Die VDI-Dateien der beiden Systemzustände werden mit VBoxManage in Raw-Dateien umgewandelt und in Autopsy eingelesen.

3.4.2 Mac

Für die Evaluation des Moduls mit Dateien eines macOS wurde eine mac-VM von Geekrar in der Version macOS Mojave 10.14.3 als VMDK-Datei heruntergeladen [68]. Die Einrichtung der VM ist an diese Anleitung [69] angelehnt.

Zunächst wird eine neue VM erstellt und dieser die VMDK-Datei als virtuelle Festpatte übergeben. In den Einstellungen der VM werden die Anzahl der Prozessoren und der Grafikspeicher erhöht. Zusätzlich wird in VirtualBox das Oracle VM VirtualBox Extension Pack [67] installiert, um in den Einstellungen der VM den USB-Controller auf „USB-3.0-Controller (xHCI)“ umstellen zu können.

Anschließend müssen mit VBoxManage in der Kommandozeile noch Anpassungen an der virtuellen Maschine vorgenommen werden. [70]

Die VM kann nun gestartet, das mac-Betriebssystem eingerichtet und die virtuelle Maschine genutzt werden.

Anschließend werden auf dem Mac-System alle Wallet-Softwares installiert. Auch hier werden Sicherungspunkte für die Dateien im Dateisystem und zum Testen der Carving-Funktionen erstellt, die VMDK-Datei mit VBoxManage in eine Raw-Datei umgewandelt und in Autopsy eingelesen.

3.4.3 Windows

Für das Testen verschiedener Versionen der einzelnen Wallet-Softwares wird eine Windows 10 VM („MSEdge on Win10“) genutzt [71]. Diese wird von Microsoft fertig bereitgestellt und muss in VirtualBox über das Menü (Werkzeuge → Importieren) nur importiert werden.

Auf der VM werden die oben aufgeführten Versionen der Wallet-Softwares installiert und mittels Sicherungspunkten für die einzelnen Versionen für Dateien des Dateisystems und zum Carven festgehalten.

3.4.4 Android

Die mobilen Wallet-Softwares werden auf Android getestet. Dazu werden mit dem AVD Manager von Android Studio mehrere Virtual Devices erstellt.

Ein virtuelles Android-Gerät kann im AVD Manager über die Schaltfläche „Create Virtual Device“ erstellt werden. Hier wählt man ein Hardware-Profil und eine Android-Version aus. Anschließend kann das AVD benannt und weitere Einstellungen vorgenommen werden.

Für die Electrum und Guarda Apps wird ein Pixel 3a mit Android 10.0+ (Google Play) (API Level 30) verwendet. Dateien der Exodus App werden auf einem Pixel 2 mit Android 8.1 (Google Play) (API Level 27) gesucht.

Zum Erstellen der Images der Android-Geräte wird nach dieser Anleitung [72] vorgegangen, wobei ein anderes Programm zum Rooten des AVD genutzt wird. Dafür müssen auf dem Host-PC ADB und netcat installiert sein.

Zunächst muss das AVD gestartet werden. Anschließend werden die virtuellen Smartphones mit rootAVD [73] und Magisk gerootet. Danach kann per „Drag and Drop“ die Installation von BusyBox auf dem AVD erfolgen. Nach dem Öffnen der BusyBox-App muss

dieser Root-Zugriff gewährt werden und BusyBox über die Schaltfläche „Install“ installiert werden.

Nun kann auf dem Hostsystem über die Kommandozeile mittels ADB eine Shell des AVD geöffnet werden (s. Tab. 2). Mit dem Befehl „su“ kann man in dieser zum Nutzer „root“ wechseln. Dies muss auf dem Android-Gerät erlaubt werden. Besonders interessant ist das “/data“-Verzeichnis, da die zu untersuchenden Apps dort ihre Daten ablegen. Mit „mount | grep data“ kann festgestellt werden, auf welchem Blockgerät sich die User-Daten befinden. In diesem Beispiel ist das „/dev/block/dm-4“. In einer zweiten Kommandozeile auf dem Hostsystem wird mit dem Befehl „adb forward tcp:8888 tcp:8888“ eine Weiterleitung von Anfragen zwischen dem Host und dem AVD an Port 8888 eingerichtet. In der ersten Kommandozeile kann in der Shell mit „dd“ ein Image des Blockgerät erstellt (s. Tab. 2) und dieses über das Netzwerk versendet werden. Mit netcat wird dazu an Port 8888 auf eine Verbindung gewartet (s. Tab. 2). Diese wird in der zweiten Kommandozeile am Hostsystem mit netcat hergestellt (s. Tab. 2). Daraufhin kann das Image hier empfangen und in eine dd-Datei geschrieben werden (s. Tab. 2), welche in Autopsy eingelesen werden kann. Die Abfolge der Befehle in den beiden Kommandozeilen ist in Tabelle 2 dargestellt.

Tabelle 2: Kommandoabfolge bei der Image-Erstellung eines AVD

Kommandozeile 1	**Kommandozeile 2**
adb devices	
adb -s <device-name> shell	
su	
mount | grep /data	
	adb foward tcp:8888 tcp:8888
dd if=/dev/block/dm-4 | busybox nc -l -p 8888	
	ncat 127.0.0.1 8888 > <file-name>.dd

4 Ergebnisse

In diesem Kapitel werden die Ergebnisse der Evaluation, die daraus folgenden Anpassungen am Modul und die Zugriffsmöglichkeiten auf die Wallets mit den vom Modul gefunden Dateien erläutert.

4.1 Evaluation

Im Folgenden werden die Ergebnisse der Evaluation für die verschiedenen Betriebssysteme dargestellt. In Linux, macOS und Android wird jeweils eine Version der Wallet-Softwares getestet, um die Unterschiede zu den Windows-Versionen zu erkennen. Auf der Basis letzterer wird die erste Version des Moduls erstellt. In Windows werden verschiedene Versionen der einzelnen Wallet-Softwares getestet, um die Veränderungen der Wallet-Softwares im Verlauf der Zeit zu betrachten. Das Veröffentlichungsdatum der jeweiligen Version der Wallet-Softwares ist nach der Versionsbezeichnung in Klammern angegeben.

Die Evaluation des Moduls wird mit der ersten Version des Moduls durchgeführt. Diese enthält die Wallet-Softwares Electrum, Exodus, Firefly, Wasabi, Monero und Ledger Live.

Schon vor der Evaluation der Wallet-Softwares auf verschiedenen Betriebssystemen und unterschiedliche Versionen der Wallet-Softwares sollen zu erwartende Testergebnisse eruiert werden.

Bei den für die Wallet-Dateien genutzten Dateitypen werden im Verlauf der Versionen der Wallet-Softwares keine häufigen Änderungen erwartet. Auch unterschiedliche Dateitypen für die Wallet-Dateien bei unterschiedlichen Betriebssystemen sind nicht zu erwarten. Dass sich Dateien im Laufe der Zeit verändern, wie zum Beispiel Config-Dateien in einzelnen Json-Schlüsseln, ist durchaus möglich und könnte dazu führen, dass das Modul die Dateien nicht erkennt. Bei unterschiedlichen Betriebssystemen ist zu erwarten, dass sich die Dateien im Installationsverzeichnis unterscheiden und Dateien wie die exe-Dateien der Wallet-Softwares bei Linux und macOS nicht vorhanden sind.

Die Ergebnisse der Evaluation sind auch in „Anlagen, Ergebnisse der Evaluation (tabellarisch)“ dargestellt.

Die Wallet-Versionen der einzelnen Wallet-Softwares werden auf den folgenden Seiten heruntergeladen:

Electrum

- Aktuelle Version: https://electrum.org/#download [74]
- Alte Versionen: https://download.electrum.org/ [75]

Exodus

- Aktuelle Version: https://www.exodus.com/download/ [76]
- Alte Versionen: https://downloads.exodus.io/releases/exodus-windows-x64-<version>.exe (gewünschte Version für <version> einsetzen) [77]

Firefly

- Aktuelle Version: https://firefly.iota.org/ [16]
- Alte Versionen: https://github.com/iotaledger/firefly/releases [78]

Wasabi

- Aktuelle Version: https://wasabiwallet.io/index.html#download [79]
- Alte Versionen: https://github.com/zkSNACKs/WalletWasabi/releases [80]

Monero

- Aktuelle Version: https://www.getmonero.org/downloads/ [81]
- Alte Versionen: https://github.com/monero-project/monero-gui/releases [82]

Ledger Live

- Aktuelle Version: https://www.ledger.com/de/ledger-live [19]
- Alte Versionen: https://github.com/LedgerHQ/ledger-live-desktop/releases [83]

4.1.1 Linux

Nachfolgend werden die Ergebnisse der Evaluation für das Linux-System beschrieben. Da hier keine Windows-exe-Dateien zu erwarten sind, werden diese im Folgenden ausgelassen.

4.1.1.1 Electrum

Electrum wird auf der Linux-VM in der Version 4.2.2 (vom 28.5.2022) installiert. Die Dateistruktur der sich im Verzeichnis „/home/<user>/.electrum“ befindlichen Dateien unterscheidet sich nicht von der aus Windows bereits bekannten Struktur des Ordners „AppData\Roaming\Electrum\“.

Mit Dateistruktur ist gemeint, welche Dateien die Wallet-Softwares zur Verwaltung des Wallets nutzen. Dazu gehören z.B. Anzahl und Dateitypen der Wallet-Dateien oder das Vorhandensein von Config-Dateien.

Das Modul sucht nach Wallet-Dateien von Electrum, der Config-Datei, der Datei „recent_servers“ und nach Zertifikaten. Die Datei „recent_servers“ wird nur im Dateisystem gesucht. Diese Dateien sind zwar im Linux-System vorhanden, aber das Modul konnte keine davon finden.

Auch bei den von PhotoRec gecarvten Dateien kann das Modul keine Datei von Electrum finden.

Grund dafür sind die unterschiedlichen Zeilenumbrüche bei Windows und Linux. In Windows werden dafür die Zeichen „\r\n“ (hexadezimal: 0x0D0A) genutzt. In Linux nur „\n“ (hexadezimal: 0x0A). Das Modul wird mit den Windows-Versionen der Wallet-Softwares erstellt und ist somit auf die Windows-Dateien von Electrum angepasst und sucht nach der in Windows verwendeten Zeichenfolge. Das Problem tritt bei der Suche nach unverschlüsselten Wallet-Dateien, bei der Config-Datei, bei der Datei „recent_servers“ und bei Zertifikaten auf. In Abbildung 9 ist der Unterschied einer unverschlüsselten Wallet-Datei in Windows und Linux beispielhaft dargestellt.

Auch die Suche nach Electrum Wallets über den Pfad ist bei Linux nicht erfolgreich, da der Pfad, nach dem gesucht wird, auf die Verzeichnisstruktur in Windows angepasst ist.

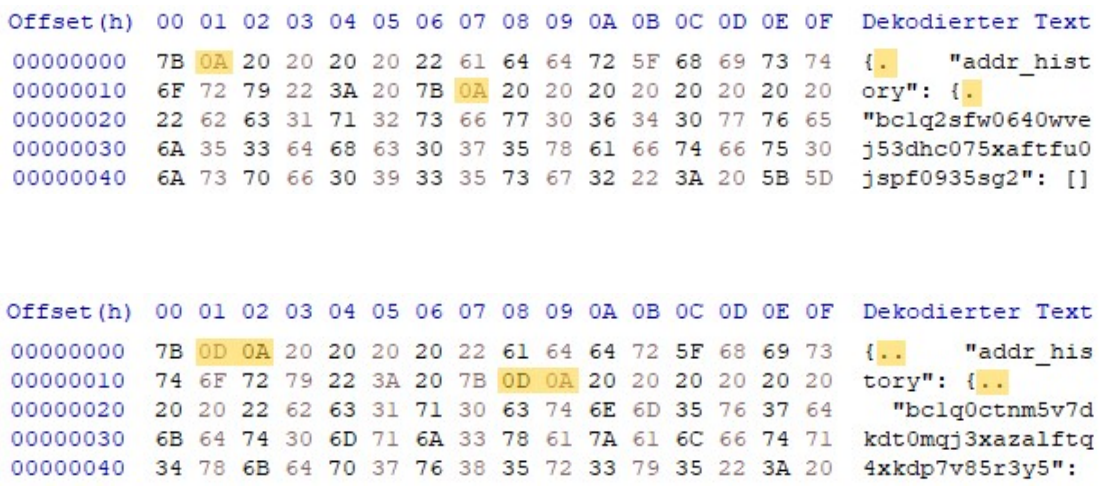

Abbildung 9: Vergleich der Electrum Wallet-Datei von Linux (oben) und Windows (unten) in HxD

4.1.1.2 Exodus

Exodus wird in der Version 22.7.1 (30.6.2022) betrachtet. Die meisten interessanten Dateien befinden sich im Verzeichnis „/home/<user>/.config/Exodus“. Auch Exodus verwendet zur Verwaltung des Wallets in der Linux-Version die gleichen Dateien wie in Windows.

Bei der Suche nach Dateien im Dateisystem erkennt das Modul die Exodus Config-Datei und die Datei „app.asar“, welche sich bei Linux im Verzeichnis „/usr/lib/exodus/resources/“ befindet. Im Wallet-Verzeichnis findet das Modul alle Secure Container und die Datei „passphrase.json“. Damit findet das Modul im Dateisystem alle erwarteten Dateien von Exodus.

Von den gelöschten Dateien kann das Modul mit Hilfe von PhotoRec die Exodus Config-Datei, die Datei „app.asar“, Secure Container und die Datei „passphrase.json“ wiederherstellen.

Bei den Json-Dateien von Exodus treten die Zeilenumbruchprobleme nicht auf, da die Dateien von Exodus sowohl in Windows als auch in Linux „\n“ (0x0A) für den Zeilenumbruch nutzen und das Modul somit darauf ausgerichtet ist.

4.1.1.3 Firefly

Das Firefly Wallet wird in der Version 1.6.2 (3.6.2022) als AppImage-Datei von der offiziellen Firefly-Seite heruntergeladen. Die Dateien befinden sich hier, ähnlich wie bei Exodus, im Verzeichnis „/home/<user>/.config/Firefly“. Auch hier entspricht die Dateistruktur der der Windows-Version.

Das Modul sucht nach der Wallet-Datei (.stronghold). Dieser Dateityp wird auch in der Linux-Version von Firefly verwendet. Außerdem sucht das Modul nach den Dateien „app.asar“ und „app-update.yml“.

Von den Dateien im Dateisystem findet das Modul lediglich die .stronghold-Wallet-Dateien. Die anderen beiden Dateien sind auf dem Linux-System nicht vorhanden bzw. in der AppImage-Datei gepackt und können somit vom Modul nicht gefunden werden.

Auch beim Carven kann nur die Wallet-Datei von Firefly gefunden werden.

4.1.1.4 Wasabi

Wasabi wird in der Version 2.0.1.1 (4.7.2022) betrachtet. Die relevanten Dateien von Wasabi befinden sich beim Linux-Betriebssystem im Verzeichnis „/home/<user>/.walletwasabi/“, welches die gleichen Dateien enthält wie das Verzeichnis „AppData\Roaming\WalletWasabi\“ in Windows.

Das Modul sucht nach der Json-Wallet, der Config-Datei und der Datei „Transactions.dat“. Letztere nur im Dateisystem.

Diese sind zwar auch im Verzeichnis der Wasabi-Wallet unter Linux vorhanden, werden jedoch vom Modul sowohl im Dateisystem als auch beim Carven nicht gefunden. Bei der Wallet-Datei und der Config-Datei liegt das an den unterschiedlichen Zeichen, die in Windows und Linux für den Zeilenumbruch verwendet werden. Die Datei „Transactions.dat“ wird vom Modul über den Pfad gesucht, der an den Windows-Pfad angepasst ist. Daher kann diese Datei auch nicht gefunden werden.

4.1.1.5 Monero

Monero wird in der Version 0.17.3.2 (29.4.2022) installiert. Die Wallet-Dateien befinden sich im Verzeichnis „/home/<user>/Monero/wallets“. Die Dateistruktur des Wallet-Verzeichnisses entspricht der bereits aus Windows bekannten Dateistruktur.

Neben den Dateien im Wallet-Verzeichnis sucht das Modul nach einer Uninstall-Datei und einer Readme-Datei. Nur die Datei „readme.htm“ wird dabei gecarvt.

Bei der Suche im Dateisystem kann lediglich die „.keys“-Datei im Wallet-Verzeichnis gefunden werden. Diese sucht das Modul über den relativen Pfad „Monero/wallets/*/“. Die Dateien aus dem Windows-Installationsverzeichnis, also die Uninstall-Datei und die „readme.htm“, sind im Linux-System nicht vorhanden.

Beim Carving kann keine Datei gefunden werden, da die einzige Datei, die von Monero gecarvt wird, die Datei „readme.htm“, nicht im Linux-System vorhanden ist.

Die Datei „.address.txt“ ist bei neueren Versionen nicht direkt nach der Installation vorhanden, sondern entsteht erst, nachdem Transaktionen durchgeführt werden. In der aktuellen Versuchsanordnung ist die Datei nicht vorhanden.

4.1.1.6 Ledger Live

Ledger Live wird in der Version 2.44.0 (5.7.2022) als AppImage-Datei betrachtet.

Das Modul sucht nach den Dateien „app.asar“ und „app-update.yml“.

Von diesen kann keine gefunden werden, da keine dieser Dateien im Linux-System vorhanden sind bzw. in der AppImage-Datei verpackt sind.

4.1.2 Mac

Die Auswertung der Ergebnisse der Mac-VM wird in diesem Kapitel behandelt. Auch hier werden die exe-Dateien ausgelassen.

4.1.2.1 Electrum

Electrum wird in der Version 4.2.2 (28.5.2022) installiert. Electrum legt seine Dateien im Verzeichnis /Users/<user>/.electrum ab.

Das Modul sucht nach Wallet-Dateien, der Config-Datei, der Datei „recent_servers“ und nach Zertifikaten.

Wie auch bei Linux kann das Modul sowohl im Dateisystem als auch beim Carven keine der gesuchten Dateien finden. Dies liegt auch hier an den Zeilenumbruchsunterschieden zu Windows. Der Zeilenumbruch der Electrum-Dateien ist bei Mac wie bei Linux „\n“ (hexadezimal: 0x0A).

Die Suche nach Wallet-Dateien über den Pfad bleibt ebenso erfolglos, da der gesuchte Pfad an Windows angepasst ist. Das Modul sucht Dateien im Verzeichnis „AppData/Roaming/Electrum/wallets“. Bei Mac befindet sich das Wallet jedoch im Verzeichnis „.electrum/wallets“.

4.1.2.2 Exodus

Exodus wird in der Version 22.7.15 (14.7.2022) betrachtet. Bei macOS speichert Exodus seine Dateien im Verzeichnis „/Users/<user>/Library/Application Support/Exodus“.

Das Modul findet sowohl im Dateisystem als auch beim Carven Seco-Dateien, die Exodus Config-Datei und die Datei „passphrase.json“.

Die Datei „app.asar“ von Exodus, die sich bei macOS im Verzeichnis „/Applications/Exodus.app/Contents/Resources“ befindet, kann das Modul nicht finden. Dies liegt daran, dass die Datei beim Test eine Größe von null Bytes hat und das Modul diese Dateien nicht einliest.

4.1.2.3 Firefly

Firefly wird in der Version Version 1.6.3 (14.7.2022) installiert.

In der Mac-VM gibt es Probleme, ein Wallet in der Firefly Wallet-Software zu erstellen. Der Erstellungsprozess einer Wallet kann nicht abgeschlossen werden, da Firefly in dessen Verlauf abstürzt. Daher kann für die Mac-VM nicht überprüft werden, ob das Modul die Stronghold-Dateien hier findet.

Die Dateien der Mac-Version von Firefly werden im Verzeichnis „/Users/<user>/Library/Application Support/Firefly“ abgelegt.

Das Modul erkennt die Datei „app.asar“ aus dem Verzeichnis „/Applications/Firefly.app/Contents/Resources“. Diese wird sowohl im Dateisystem als auch beim Carven erkannt. Die Datei „app-update.yml“ aus demselben Verzeichnis wird nicht erkannt, da sich die Felder der Datei von den Suchmustern unterscheiden.

4.1.2.4 Wasabi

In der Mac-VM wird Wasabi in der Version 2.0.1.2 (13.7.2022) betrachtet. Die Dateien der Wasabi-Wallet befinden sich im Verzeichnis „/Users/<user>/.walletwasabi“.

Sowohl im Dateisystem als auch beim Carven findet das Modul keine Dateien.

Die Dateien „Wallet.json“ und „Config.json“ werden nicht gefunden, weil für Mac und Windows unterschiedliche Zeichen für den Zeilenumbruch genutzt werden. Bei den Mac-Dateien von Wasabi wird wie bei Linux „\n“ (0x0A) für den Zeilenumbruch verwendet. In Windows benutzen die Wasabi-Dateien „\r\n“ (0x0D0A) für den Zeilenumbruch. Da das Modul auf Windows ausgerichtet ist, werden die Mac-Dateien nicht gefunden.

Die Datei „Transactions.dat“, die über den Pfad gesucht wird, kann ebenfalls nicht gefunden werden. Grund dafür ist, dass der Pfad an Windows angepasst ist und sich die Ordnerstruktur bei Mac von der von Windows unterscheidet.

4.1.2.5 Monero

Monero wird in der Version 0.17.3.2 (29.4.2022) betrachtet. Die Dateien von Monero befinden sich im Verzeichnis „/Users/<user>/Monero/“.

Im Dateisystem wird die „.keys“-Datei aus dem Wallet-Verzeichnis gefunden. Nach dieser Datei wird über den relativen Pfad gesucht.

Beim Carven kann keine Datei erkannt werden.

Die Datei Uninstall-Dat und die Datei „readme.htm“ werden vom Modul nicht gefunden, da sie bei der Mac-Version von Monero nicht vorhanden sind.

Auch hier ist die Datei „address.txt“ nicht vorhanden.

4.1.2.6 Ledger Live

Ledger Live wird in der Version 2.44.0 (5.7.2022) installiert.

Die Dateien „app.asar“ und „app-update.yml“ befinden sich bei Ledger Live im Verzeichnis „/Users/<user>/Downloads/Ledger Live.app/Contents/Resources“. Diese beiden Dateien werden jedoch im Dateisystem und beim Carven nicht erkannt, da sich einige Felder der Dateien seit der Erstellung des Moduls verändert haben und nicht mehr mit den Suchmustern übereinstimmen.

4.1.3 Windows

Im Folgenden werden die Ergebnisse der Evaluation der auf der Windows-VM installierten Versionen der Wallet-Softwares erläutert.

4.1.3.1 Electrum

Electrum wird in den Versionen 1.8 (portable, setup und exe-standalone, vom 27.7.2013), 2.5 (17.10.2015), 3.2.0 (30.6.2018) und 4.2.2 (28.5.2022) installiert.

Die Dateistruktur von Electrum der Version 1.8 unterscheidet sich zu der von späteren Versionen. So liegt z.B. die Wallet-Datei in Form einer .dat-Datei vor. Diese ähnelt zwar der aus späteren Versionen bekannten json-Datei, unterscheidet sich aber so sehr, dass das Modul die Datei nicht erkennt. Neben dieser Datei befindet sich nur die Datei „blockchain_headers“ im Verzeichnis der Wallet. Die Dateien, nach denen das Modul sucht, also Zertifikate, die Datei „recent_servers“ und die Config-Datei sind nicht vorhanden. Daher kann das Modul keine Dateien von Electrum in der Version 1.8 finden.

Im Vergleich zu Version 1.8 nutzt die Wallet-Software von Electrum bei den Versionen 2.5 (setup) und 3.2.0 (setup) die bereits bekannten Dateien.

Im Dateisystem kann das Modul hier bei den Versionen 2.5 und 3.2.0 jeweils zwei Zertifikate und eine unverschlüsselte Wallet-Dateien finden. Die Datei „recent_servers“ wird nicht gefunden, weil keiner der gesuchten Server in der Datei enthalten ist.

Beim Carven werden jeweils Zertifikate gefunden und zwar bei Version 2.5 zwei Zertifikate und bei Version 3.2.0 ein Zertifikat. Weitere Dateien können beim Carven nicht gefunden werden.

Die Config-Datei wird sowohl im Dateisystem als auch beim Carven nicht gefunden, da sich einige Felder der Json-Datei von den vom Modul gesuchten Feldern unterscheiden. Aus demselben Grund kann die unverschlüsselte Wallet-Datei nicht gecarvt werden. Im Dateisystem wird sie nur über den Pfad erkannt.

Bei der Version 4.2.2 (setup) kann ebenfalls die bekannte Dateistruktur festgestellt werden.

Im Dateisystem findet das Modul ein Zertifikat, die Config-Datei und eine unverschlüsselte Wallet-Datei. Die Datei „recent_servers“ wird nicht gefunden.

Beim Carven kann das Modul ein Zertifikat, die Config-Datei und eine unverschlüsselte Wallet-Datei finden.

Außerdem gibt es bei den Electrum Wallets jeweils falsch positive Ergebnisse bei der Suche über den Pfad. So werden das aktuelle Verzeichnis „.“ und das übergeordnete Verzeichnis „..“ aus dem Verzeichnis „AppData\Roaming\Electrum\wallets\“ ebenfalls als Interesting File hinzugefügt.

4.1.3.2 Exodus

Exodus wird in den Versionen 19.1.3 (3.1.2019), 20.1.3 (3.1.2020)), 21.1.7 (7.1.2021) und 22.7.1 (30.6.2022) betrachtet.

Die Dateistruktur unterscheidet sich bei den verschiedenen Versionen nur geringfügig.

Das Modul sucht nach Secure Containern und der Datei „passphrase.json“ aus dem Wallet-Verzeichnis. Zusätzlich sucht es nach der Config-Datei, der Datei app.asar und der exe-Datei. Letztere wird nur im Dateisystem gesucht.

Bei den Versionen 19.1.3 und 20.1.3 findet das Modul im Dateisystem je zwei Secure Container und die Datei „passphrase.json“ und damit alle Dateien im Wallet-Verzeichnis.

Die exe-Datei kann aufgrund eines Fehlers im Programmcode des Moduls nicht gefunden werden.

Beim Carven werden jeweils zwei Seco-Dateien gefunden.

Die Datei „passphrase.json“ wird nicht gecarvt. Dies könnte daran liegen, dass die Datei sehr klein ist und somit im NTFS-Dateisystem das Data-Attribut des MFT-Eintrags der Datei resident ist. Bei residenten Einträgen befindet sich der Beginn der Daten der Datei nicht an der Stelle, an der PhotoRec nach dem Dateianfang sucht, also nicht am Beginn des Clusters. PhotoRec carvt eigentlich residente Data-Attribute [84]. Da in Autopsy aber nicht der komplette Datenträger in PhotoRec eingelesen wird, sondern nur die einzelnen unallokierten Bereiche, kann das dazu führen, dass PhotoRec das Dateisystem des Datenträgers nicht erkennt (hier NTFS) und somit die residenten Einträge nicht carvt.

In Abbildung 10 ist der residente Eintrag einer Exodus Passphrase-Datei in HxD dargestellt. Hier ist zu sehen, dass die eigentlichen Daten der Datei, die zum Carven dienen, nicht am Beginn eines Clusters stehen, sondern am Ende des MFT-Eintrags, der mit „FILE“ beginnt (an Offset 0x9F3603800). PhotoRec sucht die Signaturen zu Beginn eines

Clusters. Der Beginn der Daten der Datei ist aber an Offset 0x9F36039A0 und damit mitten in einem Cluster.

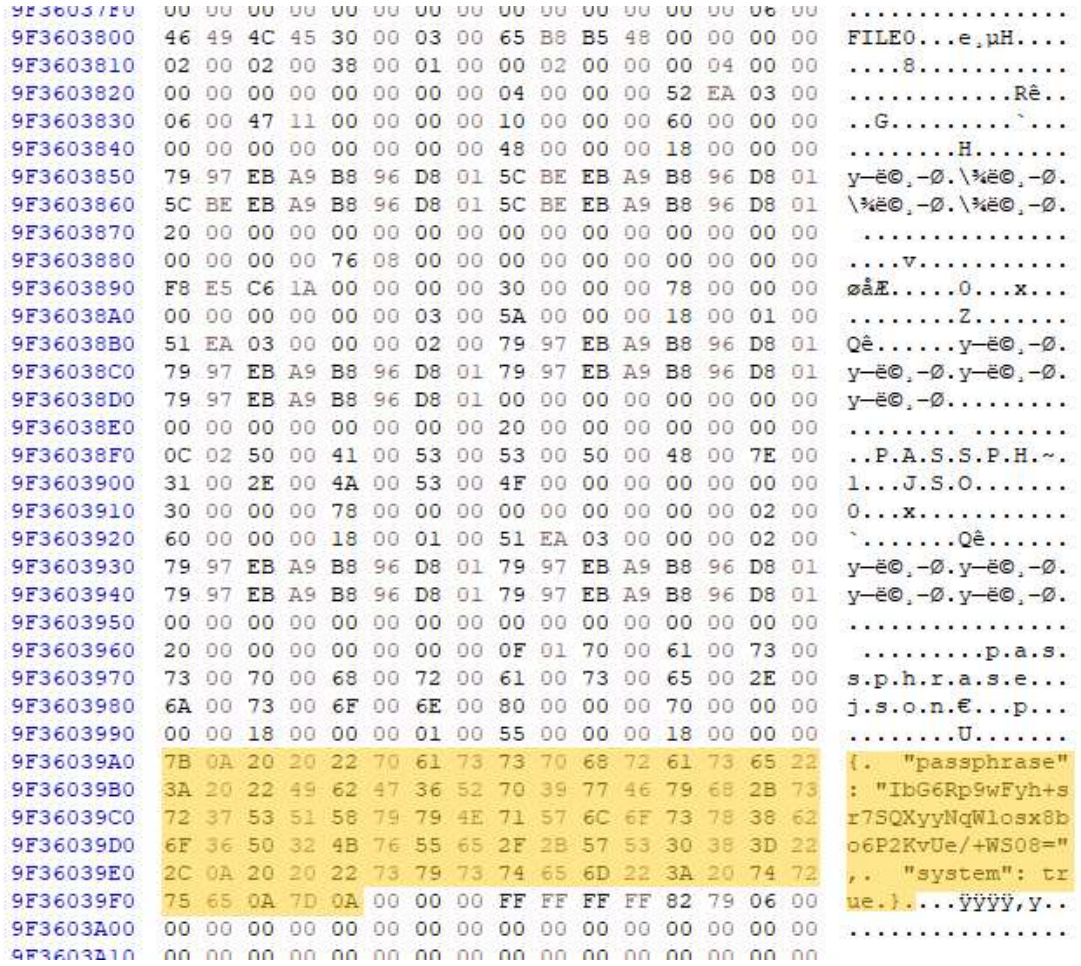

Abbildung 10: MFT-Eintrag der Exodus Passphrase-Datei in HxD mit einem residenten Data-Attribut. Markiert sind die eigentlichen Daten der Datei.

Die Datei „app.asar“ wird sowohl beim Carven als auch im Dateisystem nicht gefunden, weil sich Json-Felder der Datei dieser Versionen von den Suchmustern des Moduls für diese Datei unterscheiden.

Die Config-Datei von Exodus ist bei diesen Versionen noch nicht vorhanden, weshalb die Datei nicht erkannt werden kann.

Bei Version 21.1.7 kann das Modul im Dateisystem alle vier Secure Container des Wallet-Verzeichnisses und die Datei „passphrase.json“ finden.

Die Exodus-Config-Datei, die exe-Datei und die Datei „app.asar“ werden bei dieser Version aus denselben Gründen nicht erkannt wie bei den beiden vorherigen Versionen.

Beim Carven können vier Seco-Dateien wiederhergestellt werden. Die Datei „passphrase.json“ wird aus denselben Gründen wie bei den vorherigen Versionen nicht gecarvt.

Bei Version 22.7.1 findet das Modul im Dateisystem alle fünf Seco-Dateien im Wallet-Verzeichnis und die Datei „passphrase.json“. Des Weiteren findet das Modul die Dateien „app.asar“ und die Config-Datei. Die exe-Datei kann auch hier aus den bereits oben erwähnten Gründen nicht gefunden werden.

Beim Carven werden vier Seco-Dateien und die Exodus-Config-Datei wiederhergestellt. Die Datei „passphrase.json“ kann auch hier aus denselben Gründen nicht gecarvt werden wie bei den vorherigen Versionen. Auch die Datei „app.asar“ kann nicht komplett gecarvt werden. Dadurch sind Suchmuster des Moduls nicht in der Datei enthalten und die Datei wird nicht erkannt.

Die verschiedenen Versionen von Exodus unterscheiden sich hauptsächlich in der Anzahl der verwendeten Secure Container. Im Gegensatz zu früheren Versionen wird in späteren Versionen eine Config-Datei genutzt.

4.1.3.3 Firefly

Firefly wird in den Versionen 0.0.2 (20.3.2021), 1.0.0 (21.4.2021), 1.3.3 (24.1.2022) und 1.6.2 (3.6.2022) installiert. Um Firefly in der Windows-VM nutzen zu können, musste zusätzlich „VC++ runtime“ installiert werden. Die Dateistruktur unterscheidet sich bei den verschiedenen Versionen nicht. Alle Dateien, nach denen gesucht wird, sind bei allen Versionen vorhanden.

Das Modul sucht im Dateisystem nach den Dateien „Firefly.exe“, „app-asar“, „.stronghold“-Wallet-Dateien und „app-update.yml“. Von diesen können die ersten drei Dateien bei allen betrachteten Versionen im Dateisystem gefunden werden.

Bei den Versionen 1.3.3 und 1.6.2 kann noch zusätzlich die Datei „app-update.yml“ gefunden werden. Bei vorherigen Versionen wird diese nicht erkannt, da sich einzelne Felder in der Datei von den Suchmustern unterscheiden.

Beim Carven sucht das Modul Stronghold-Dateien, die .asar-Datei und die Datei „app-update.yml“. Von diesen findet das Modul bei allen Versionen die ersten beiden Dateien. Die Datei „app-update.yml“ kann bei keiner Version gecarvt werden. Der Grund dafür

könnte sein, dass die Datei sehr klein ist und wie die Exodus Passphrase-Datei in einem residenten Data-Attribut des MFT-Eintrags abgelegt und das Carven damit verhindert wird.

4.1.3.4 Wasabi

Wasabi wird in den Versionen 0.9.3 (9.8.2018), 1.0 (31.8.2018), 1.1.7 (11.9.2019) und 2.0.1.1 (4.7.2022) installiert.

Das Modul sucht nach der Json-Wallet-Datei, nach der Config-Datei und der Datei „Transactions.dat“. Letztere nur über den Pfad im Dateisystem.

Die Json-Wallet-Datei und die Config-Datei sind bei allen Versionen vorhanden.

Die Datei „Transactions.dat“ ist bei den betrachteten Versionen erst ab Version 2.0.1.1 vorhanden. Bei den Versionen 0.9.3 und 1.0 gibt es eine Json-Transactions-Datei.

Das Modul findet bei den Versionen 0.9.3, 1.0 und 1.1.7 lediglich die Wallet-Datei im Dateisystem über den Pfad.

Die Config-Datei wird im Dateisystem und beim Carven nicht gefunden, weil sich einzelne Json-Felder der Datei von den Suchmustern unterscheiden.

Die Wallet-Datei wird zwar von PhotoRec gecarvt, aber nicht vom Modul als „Interesting File“ hinzugefügt, weil sich einzelne Json-Felder der Datei von den Suchmustern unterscheiden.

Bei Version 2.0.1.1 können die Wallet-Datei und die Config-Datei sowohl im Dateisystem als auch beim Carven gefunden werden.

Die Datei „Transactions.dat“, die über den Pfad gesucht wird, wird im Dateisystem nicht gefunden, da sie bei den Tests keinen Inhalt hatte und das Modul nur Dateien markiert, die größer als null Bytes sind.

4.1.3.5 Monero

Monero wird in den Versionen 0.11.0.0 (24.9.2017), 0.14.0.0 (1.3.2019), 0.16.0.0 (2.6.2020), 0.17.1.0 (14.10.2020) und 0.17.3.2 (29.4.2022) betrachtet.

Das Modul sucht nach zwei Dateien im Wallet-Verzeichnis und zusätzlich nach einer Uninstall-, einer Readme- und der exe-Datei. Die Uninstall-Datei und die exe-Datei von Monero werden nur im Dateisystem gesucht. Die „readme.htm“ wird zusätzlich gecarvt. Im Wallet-Verzeichnis sucht das Modul nach zwei Dateien. Einer „.keys“-Datei und einer Datei mit der Endung „.address.txt“. Beide werden nur im Dateisystem gesucht.

Bei Version 0.11.0.0 findet das Modul im Dateisystem die exe-Datei, die .keys-Datei und die address.txt-Datei. Beim Carven kann keine Datei gefunden werden. Die Uninstall-Dat und die „readme.htm“ werden nicht gefunden, da sie bei dieser Version nicht im Installationsverzeichnis von Monero enthalten sind.

Die „.address.txt“-Datei ist bei späteren Versionen erst nach dem Ausführen von Transaktionen vorhanden.

Bei den Versionen 0.14.0.0 und 0.17.1.0 werden die exe-Datei, die Readme-Datei und die Uninstall-Datei im Dateisystem gefunden. Im Wallet-Verzeichnis findet das Modul die „.keys“-Datei.

Beim Carven kann bei Version 0.14.0.0 und Version 0.17.1.0 die Datei „readme.htm“ wiederhergestellt werden.

Bei den Versionen 0.16.0.0 und 0.17.3.2 werden ebenfalls im Dateisystem die Dateien „readme.htm“, die Uninstall-Datei, die exe-Datei und die .keys-Datei gefunden.

Beim Carven findet das Modul keine Datei, auch nicht die „readme.htm“. Bei der manuellen Suche nach der Datei auf dem Datenträger kann diese nicht gefunden werden. Eventuell wurde sie nach dem Löschen überschrieben.

4.1.3.6 Ledger Live

Ledger Live wird in den Versionen 1.0.0-beta (22.8.2018), 1.6.0 (15.3.2019), 2.0.0 (28.2.2020) und 2.44.0 (5.7.2022) installiert.

Das Modul sucht nach der exe-Datei von Ledger, nach der Datei „app.asar“ und nach der Datei „app-update.yml“. Erstere nur im Dateisystem, die anderen beiden werden auch gecarvt.

Bei den Versionen 1.0.0-beta und 1.6.0 kann das Modul im Dateisystem die exe-Datei und die Asar-Datei finden. Die Asar-Datei wird auch bei beiden Versionen gecarvt. Die Datei App-update.yml kann jeweils nicht gefunden werden, weil einzelne Felder der Datei nicht mit den Suchmustern des Moduls übereinstimmen.

Bei Version 2.0.0 werden die Dateien „App.asar“, „app-update.yml“ und „Ledger Live.exe“ im Dateisystem gefunden. Also alle Dateien, nach denen das Modul sucht. Beim Carven kann die Asar-Datei wiederhergestellt werden.

Bei Version 2.44.0 findet das Modul die exe-Datei und die Asar-Datei im Dateisystem. Die Asar-Datei kann auch gecarvt werden.

Die Datei „app-update.yml“ kann im Dateisystem nicht gefunden werden, weil sich die Suchmuster des Moduls von den Dateiinhalten unterscheiden.

Die Datei „app-update.yml“ wird bei keiner Version gecarvt. Grund dafür könnte sein, dass es sich um eine sehr kleine Datei handelt, bei der das Data-Attribut des MFT-Eintrags resident ist. Dies könnte wie bei der Exodus Passphrase-Datei zu Problemen beim Carven führen.

4.1.4 Android

Nachfolgend werden die Ergebnisse der Betrachtung der Electrum und Exodus Apps auf einem Android Virtual Device dargestellt.

4.1.4.1 Electrum

Bei der Electrum Android-App (Version 4.2.2) befinden sich im Verzeichnis „/data/data/org.electrum.electrum/files/data/“ die bereits von der Electrum Desktopversion bekannten Dateien.

Da der Inhalt des „/data“-Verzeichnisses nach dem Erstellen des Images verschlüsselt ist, konnte das Modul zunächst keine Dateien finden.

Um zu überprüfen, welche Dateien das Modul findet, wird das Verzeichnis „/data/data/org.electrum.electrum/files/data/“ in einen unverschlüsselten Ordner kopiert und erneut ein Image erstellt.

Es werden keine der gesuchten Dateien gefunden. Dies liegt an den bereits aus Linux bekannten Zeilenumbruchsunterschieden.

4.1.4.2 Exodus

Das Modul kann auf dem Image des untersuchten virtuellen Android-Geräts keine Dateien finden. Die Exodus App (Version 22.6.30) speichert Informationen im Verzeichnis „/data/data/exodusmovement.exodus/“. Hier sind weder die gesuchte Config-Datei noch Secure Container oder die Datei „passphrase.json“ zu finden. Im Verzeichnis „/data/data/exodusmovement.exodus/databases/“ befindet sich eine Datenbank, bei deren Inhalt es sich um Wallet-Informationen handelt. Dies lässt vermuten, dass die Exodus App keine Secure Container zum Speichern der Wallet-Informationen nutzt, sondern diese Datenbank.

4.2 Erweiterung und Anpassung des Moduls

Mit den aus der Evaluation gewonnen Erkenntnissen können die Signaturen für PhotoRec und die Suchmuster des Autopsy-Moduls ergänzt und erweitert werden. Dies soll zu besseren Ergebnissen des Moduls bei der Suche nach Dateien von Linux-Versionen, Mac-Versionen und älteren oder neueren Versionen der Wallet-Softwares führen.

Evaluiert wurde dabei die erste Version des Moduls. Diese umfasst die Wallet-Softwares Electrum, Exodus, Firefly, Wasabi, Monero und Ledger Live.

Die zweite Version des Moduls besteht aus den in diesem Kapitel aufgeführten Anpassungen nach der Evaluation und der Erweiterung des Moduls um Dateien der Guarda Wallet, Browser Erweiterungen und der Apps von Electrum, Exodus und Guarda.

In „Anlagen, Ergebnisse der Evaluation (tabellarisch)“ sind die Ergebnisse der zweiten Modulversion im Vergleich zur ersten Version dargestellt.

4.2.1 PhotoRec

Der bedeutendste Unterschied zwischen den Windows-Versionen der Wallet-Softwares und den Versionen bei Linux und Mac ist der Zeilenumbruch. Bei Windows geschieht der Zeilenumbruch mit den beiden Zeichen „\r\n“ (hexadezimal „0x0D0A“). Bei Linux und Mac nur mit dem Zeichen „\n“ (hexadezimal „0x0A“). Dieses Problem tritt besonders bei Dateien von Electrum und Wasabi auf. Daher wird bei allen Signaturen dieser Wallet-Softwares, bei denen der Zeilenumbruch „\r\n“ enthalten ist, ein alternativer Header bzw. Footer mit „\n“ statt „\r\n“ hinzugefügt.

Bei Electrum und Wasabi wird dies jeweils für die Wallet-Datei und die Config-Datei durchgeführt.

Für Dateien der Electrum-Wallet werden noch zusätzlich Signaturen für die Datei „electrum.dat“ und die Wallet-Datei hinzugefügt bzw. erweitert.

Die Datei „electrum.dat“ ist bei der Evaluation bei älteren Versionen der Electrum-Wallet aufgefallen. Für diese Datei wird in PhotoRec ein Dateityp mit der Endung „.electrumDat“ erstellt. Dieser hat den Header „*{'addr_history':*“ und den Footer „*]}*“.

Der Electrum-Wallet-Dateityp wird um den Header „*{\r\n \"accounts\": {\r\n*“ bzw. „*{\n \"accounts\": {\n*“ ergänzt. Dieser tritt bei älteren Versionen der Electrum-Wallet auf.

Bei Firefly wird für die Datei „app-update.yml“ mit der PhotoRec-Endung „.fireflyAppupdate“ ein alternativer Header hinzugefügt: „*provider: generic\nurl: 'https://iotaledger-files*“. Dieser kommt bei einigen älteren Versionen der Datei vor.

Auch bei Ledger Live werden die Signaturen für die Datei „app-update.yml“ angepasst. Als alternativer Header wird „*provider: generic\nurl: https://download.live.ledger*“ hinzugefügt und als alternative Footer „*ledger-live-desktop\nprovider: github*“ und „*publisherName:\n - Ledger SAS\n*“.

4.2.2 Autopsy

Auch im Autopsy-Modul werden die Suchmuster angepasst. Ein alternatives Suchmuster mit „\n“ als Zeilenumbruch wird für das unverschlüsselte Electrum-Wallet, die Electrum-Config-Datei, die Datei „recent_servers“ und die Zertifikate hinzugefügt. Bei Wasabi werden die Suchmuster um Optionen mit dem Zeilenumbruch „\n“ für die Wallet-Datei und die Config-Datei erweitert.

4.2.2.1 Electrum

Für Electrum werden noch weitere Suchmuster angepasst. Neben Wallets im Verzeichnis „AppData/Roaming/Electrum/wallets/“ werden nun auch Wallets im Verzeichnis „.electrum/wallets/“ gesucht. In Letzterem befinden sich die Wallet-Dateien in Linux und Mac. Die Dateien aus diesen Verzeichnissen dürfen dabei nicht „.“ oder „..“ heißen. Diese Änderung wird hinzugefügt, da teilweise auch der übergeordnete Ordner („..“) als Interesting File markiert wird. Darüber hinaus werden für die Wallet-Datei von Electrum Suchmuster für ältere Versionen der Datei hinzugefügt.

Auch für ältere Versionen der Config-Datei werden alternative Suchmuster erstellt und die Suchmuster angepasst.

Die Datei „electrum.dat“ wird als neue Datei erstellt und für diese Suchmuster hinzugefügt.

Für die Electrum Android-App wird der Pfad der Wallet-Dateien („org.electrum.electrum/files/data/wallets/“) zur Suche im Dateisystem ergänzt.

4.2.2.2 Exodus

Bei Exodus wird der Pfad, in dem nach Wallet-Dateien gesucht wird, auf „Exodus/exodus.wallet/“ geändert. Dieser schließt neben Dateien der Windows-Version des Wallets auch Dateien der Linux- und der Mac-Version ein.

Die Datei „passphrase.json“ wird im Dateisystem teilweise doppelt als „Interesting File“ hinzugefügt. Dies passiert, da seco-Dateien und die Json-Datei gemeinsam über den Pfad gesucht werden. Dies wird geändert, indem seco-Dateien und die Datei „passphrase.json“ separat über den Pfad gesucht werden. Der Pfad der eingelesenen Datei wird jetzt nicht mehr mit dem Pfad des Exodus-Wallet-Verzeichnisses abgeglichen, wenn die Datei schon als „passphrase.json“ erkannt ist.

Bei der Suche nach der „Exodus.exe“ wird ein Fehler behoben. Die Datei wird über den relativen Pfad gesucht. Der Abgleich mit dem Pfad der eingelesenen Datei erfolgt mit dem Gleichheitsoperator „==“ statt wie eigentlich vorgesehen und jetzt durchgeführt mit regulären Ausdrücken.

Für die Datei „app.asar“ werden die Suchmuster so angepasst, dass auch ältere Versionen der Datei gefunden werden können.

Für die Exodus Android-App wird der Pfad „exodusmovement.exodus/databases/“ zur Datenbank „RKStorage“ hinzugefügt. Diese enthält Wallet-Informationen.

4.2.2.3 Firefly

Bei Firefly wird ein Suchmuster der Datei „app-update.yml“ entfernt, um auch ältere Versionen finden zu können.

4.2.2.4 Wasabi

Auch bei Wasabi werden die Suchmuster für ältere Versionen der Wallet-Datei und der Config-Datei angepasst.

Bei der Suche nach der Wallet-Datei und der Datei „Transactions.dat“ werden alternative Pfade für Linux- und Mac-Versionen der Wallet-Software hinzugefügt.

4.2.2.5 Monero

Die Monero-Uninstall-Datei fehlt bei der ersten Versionen des Moduls beim Durchsuchen der gecarvten Dateien. Dies wird korrigiert.

Von den Monero-Dateien im Wallet-Verzeichnis werden nicht mehr nur die „.keys“- und „.address.txt“-Dateien über den Pfad als „Interesting File“ hinzugefügt, sondern alle Dateien im Wallet-Verzeichnis.

4.2.2.6 Ledger Live

Bei Ledger Live werden die Suchmuster für die Datei „app-update.yml“ angepasst, damit auch ältere Versionen gefunden werden können.

4.3 Zugriffsmöglichkeiten auf Wallets mit den gefundenen Informationen

Dieses Kapitel beschäftigt sich mit der Frage, ob mit den gefundenen Dateien auf die Wallets zugegriffen werden kann. Dazu werden die vom Modul gefundenen und gecarvten Dateien genutzt und mit den Wallet-Wiederherstellungsfunktionen in die Wallet-Softwares eingelesen.

Das Modul sucht bei Browser-Erweiterungen lediglich Dateien, die die Installation der Wallet identifizieren. Nach Dateien mit Wallet-Informationen der Erweiterungen sucht das Modul nicht. Daher werden die Browser-Erweiterungen hier ausgelassen. Browser Erweiterungen von Chrome speichern ihre Daten in Level-DB-Datenbanken. Diese kann man z.B. mit einem Viewer wie dem „Leveldb-py“ [85] von Mark McKinnon betrachteten.

4.3.1 Electrum

Mit der in Abschnitt 4.1.3.1 gecarvten unverschlüsselten Wallet-Datei der Electrum-Version 4.2.2 wird getestet, ob mit den aus PhotoRec und Autopsy gewonnen Dateien Zugriff auf das Wallet erlangt werden kann. Das Einbinden der Wallet-Datei in Electrum ist auf verschiedene Arten möglich. Zum einen kann man die Wallet-Datei im Verzeichnis „AppData\Roaming\Electrum\wallets\“ ablegen. Eine andere Möglichkeit ist, über das Menü (Datei → Öffnen) die Wallet-Datei zu öffnen. Alternativ kann am Beginn des Installations-Assistenten eine Datei ausgewählt werden. Mit diesen Varianten kann die unverschlüsselte Wallet-Datei eingelesen werden und der Zugriff auf das Wallet erfolgen.

Bei verschlüsselten Wallet-Dateien ist die Kenntnis über das Passwort der Wallet notwendig, um Zugriff zu erhalten.

Des Weiteren ist die Wiederherstellung eines Wallets mit einem Seed oder mit dem privaten Schlüssel möglich. Dazu muss beim Installations-Assistent im Bereich „Schlüsselspeicher“ „Ich habe bereits eine Seed“ bzw. „Einen Generalschlüssel verwenden“ ausgewählt werden.

4.3.2 Exodus

Zur Wiederherstellung von Exodus-Wallets werden die im Dateisystem der Windows-VM gewonnen Dateien der Exodus-Version 21.1.7 und die im Mac-System (Exodus-Version 22.7.15) gecarvten Dateien verwendet.

Im Dateisystem der Windows-VM befinden sich vier Seco-Dateien und die Datei „passphrase.json“ im Wallet-Verzeichnis von Exodus. Indem diese fünf Dateien im Wallet-Verzeichnis („AppData\Roaming\Exodus\exodus.wallet“) abgelegt werden, kann das Wallet wiederhergestellt werden.

Das Modul kann im Mac-System vier Secure-Container und die Datei „passphrase.json“ carven. Die von PhotoRec mit der Endung „exodusPassphrase“ versehene Datei wird in „passphrase.json“ umbenannt und in das Wallet-Verzeichnis von Exodus kopiert. Im Wallet-Verzeichnis von Exodus befinden sich mehrere Seco-Dateien: „info.seco“, „seed.seco“, „twofactor.seco“ und „twofactor-secret.seco“. Nach dem Carven ist nicht mehr ersichtlich, welche der wiederhergestellten Dateien welcher Seco-Datei entspricht. Deshalb werden die vier gecarvten Dateien nacheinander in „seed.seco“ umbenannt und in das Wallet-Verzeichnis abgelegt. Da Exodus nur bei der richtigen Seed-Datei startet, kann diese so gefunden werden. Um zu überprüfen, ob das Wallet wiederhergestellt wurde, werden die privaten Schlüssel des gerade neu wiederhergestellten Wallets mit den des Wallets in der Mac-VM verglichen. Diese stimmen überein. Zur Wiederherstellung des Wallets reichen beim Test die Dateien „passphrase.json“ und „seed.seco“ aus.

Für den Fall, dass keine „passphrase.json“ gecarvt werden kann, muss für den Zugriff auf das Wallet ein Passwort eingegeben werden.

Auch bei Exodus kann eine Wallet mit der Eingabe des Seeds wiederhergestellt werden.

4.3.3 Firefly

Bei Firefly gibt es zwei Möglichkeiten, eine Wallet wiederherzustellen. Einerseits mit dem Seed, der einem bei der Wallet-Erstellung mitgeteilt wird, und andererseits, indem man eine Wallet-Backup-Datei einliest. Dafür kann eine im Dateisystem gefundene oder eine gecarvte Backup-Datei genutzt werden. Bei diesem Backup handelt es sich wie bei den

Wallet-Dateien um Stronghold-Dateien. Die Stronghold-Dateien aus dem Wallet-Verzeichnis können jedoch nicht zur Wiederherstellung wie eine Backup-Datei genutzt werden. Zur Wiederherstellung des Wallets mit der Backup-Datei wird das bei der Wallet-Erstellung angegebene Passwort benötigt.

Auch das Einsetzen einer gecarvten Stronghold-Datei aus dem Dateisystem mit dem im Wallet-Verzeichnis befindlichen DB-Ordner in das Wallet-Verzeichnis eines anderen Computers ist nicht möglich.

4.3.4 Wasabi

Auch bei Wasabi gibt es die Möglichkeit, ein Wallet sowohl mit dem Seed als auch mit der Json-Wallet-Datei wiederherzustellen. Die Besonderheit ist hier, dass in beiden Fällen das Passwort eingegeben werden muss.

Die Wallet-Datei kann entweder in Wasabi importiert oder ins Wallet-Verzeichnis „AppData\Roaming\WalletWasabi\Client\Wallets\“ kopiert werden. Dies funktioniert auch mit der im Dateisystem gefundenen Wallet-Datei der in Windows getesteten Wasabi-Version 2.0.1.1. Die gecarvte Wallet-Datei muss zunächst angepasst werden, da PhotoRec Daten in die Datei gecarvt hat, die nicht zur Datei gehören. Danach wird der Datei noch die Endung „.json“ gegeben. Anschließend kann das gecarvte Wallet, nach Eingabe des Passworts, wiederhergestellt werden.

4.3.5 Monero

Ein Monero-Wallet kann mit dem Seed oder der „.keys“-Datei aus dem Wallet-Verzeichnis wiederhergestellt werden. Die „.keys“-Datei kann im Monero-Menü unter („Öffne eine Wallet aus einer Datei“ → „Dateisystem durchsuchen“) ausgewählt werden. Zum Wiederherstellen der Wallet mit dieser Methode wird das bei der Wallet-Erstellung angelegte Passwort benötigt. Mit der von Autopsy gefundenen „keys“-Datei der Monero-Version 0.17.1.0 aus Abschnitt 4.1.3.5 kann so das Wallet wiederhergestellt werden.

Alternativ kann auch der gesamte Ordner der Wallet in das Verzeichnis für Monero-Wallets (Standardmäßig bei Windows „Dokumente\Monero\wallets“) abgelegt und wie jedes andere Wallet mit dem Passwort geöffnet werden.

4.3.6 Guarda

Die Wiederherstellung einer Guarda-Wallet ist mit einer Guarda-Backup-Datei oder den aus der Level-DB-Datenbank des Wallet-Verzeichnisses extrahierten Wallet-Datei möglich. Dabei wird das Passwort benötigt. Diese Textdatei kann im Menü unter („Settings“ → „Restore“) hochgeladen werden.

5 Diskussion

Mit dem vorgestellten Ansatz können Dateien der Wallet-Softwares Electrum, Exodus, Firefly, Wasabi, Monero, Ledger Live, Guarda und der Browser-Erweiterungen Coinbase, Binance und MetaMask identifiziert und gecarvt und damit die Ermittler bei der Arbeit unterstützt werden.

Die Evaluation der ersten Version des Moduls führt zu Erkenntnissen, durch die das Modul verbessert wird und Probleme behoben werden.

Bei Linux und Mac legen die Wallet-Softwares ihre Daten in ähnlichen Strukturen wie bei Windows ab und verwenden dieselben Dateitypen. Probleme, Dateien zu erkennen, hat das Modul besonders bei Json-Dateien. Dies liegt an den Zeilenumbruchunterschieden zwischen Windows und Linux.

Die Ergebnisse bei Linux und Mac sind sehr ähnlich, was bei der „Verwandtschaft“ der beiden Betriebssysteme nicht überrascht. Unterschiede gibt es hierbei hauptsächlich bei den Pfaden, an denen die Dateien zu finden sind.

Bei der Evaluation in Bezug auf verschiedene Versionen der Wallet-Softwares können keine großen Unterschiede festgestellt werden. Bei allen Wallet-Softwares außer Electrum gibt es keine Veränderungen bei den für die Wallet-Dateien genutzten Dateiformaten. Einige Dateien verändern im Laufe der Zeit jedoch einzelne Inhalte wie z.B. Json-Schlüssel, weshalb eine ständige Anpassung notwendig ist. Auch kann bei einigen Wallet-Softwares beobachtet werden, dass neue Dateien hinzukommen.

In Windows-Systemen können sehr kleine Dateien, wie z.B. die Datei „passphrase.json“ der Exodus Wallet, nicht gecarvt werden. Das liegt vermutlich daran, dass die in Autopsy eingelesenen unallokierten Bereiche nicht als NTFS-Dateisystem erkannt werden und damit keine MFT-Einträge mit residenten Data-Attributen gecarvt werden. Hier könnten eine manuelle Suche auf dem Datenträger oder das Durchsuchen des Datenträgers mit PhotoRec außerhalb von Autopsy zum Auffinden der Dateien führen.

Neben Anpassungen für Windows, Linux und Mac-Versionen, werden auch Android-Apps der Wallets betrachtet.

Beim Vergleich verschiedener Plattform-Typen (Desktop-Version, Online Wallet, Android App) eines Wallets werden keine einheitlichen Muster erkannt. Die Electrum App nutzt dieselben Dateien wie die Electrum Desktop-Version. Im Gegensatz dazu können bei der Exodus App keine Secure Container gefunden werden, sondern eine Datenbank. Die Guarda Desktop-Version und die Online Wallet nutzen beide Level-DB-Datenbanken zum

Ablegen der Wallet-Informationen. Bei allen drei Versionen (Desktop-Version, Online Wallet und App) können die gleichen Backup-Dateien erstellt werden. Bei der Android App werden diese anders benannt.

Browser Erweiterungen in Chrome und die Guarda Wallet speichern Wallet-Informationen in Level-DB-Datenbanken. Diese könnte man zum Beispiel mit dem Level-DB-Viewer „Leveldb-py“ [85] von Mark McKinnon betrachten, um relevante Informationen zu finden.

Wenn bei einer Signatur kein Footer angegeben ist, carvt PhotoRec meistens bis zum nächsten Header. Dies führt teilweise zu sehr großen Dateien und dazu, dass Wallet-Dateien nach dem Carven nicht die richtige Größe haben. Daraus ergeben sich Probleme bei der Wiederherstellung der Wallets.

Bei verschlüsselten Electrum-Wallets, Wallets von Monero und Guarda werden keine Signaturen gefunden. Hier werden nur andere Dateien gecarvt. Unter der Annahme, dass sich die Wallet-Datei in der Nähe der anderen gefundenen Dateien befindet, wäre hier nur eine aufwändige händische Suche auf dem Datenträger möglich, um die Wallet-Dateien zu erhalten.

Das hier entwickelte Modul verspricht, Ermittler bei ihrer Arbeit zu entlasten. Bisher notwendige händische Suche kann großteils automatisiert durchgeführt werden.

Zukünftig sollte überprüft werden, wie gut das Modul in der praktischen Anwendung funktioniert und auf das mögliche Auftreten falsch positiver Ergebnisse geachtet werden. Das Modul sollte bei Veröffentlichung neuer Wallet-Versionen angepasst werden. Auch kann es sinnvoll sein, weitere Wallets einzuarbeiten.

Literatur

[1] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. [Online] Verfügbar unter: https://bitcoin.org/bitcoin.pdf. Zugriff am 21.05.2022 um 12:27 Uhr.

[2] Bergmann, C.: Kleine Geschichte der Kryptowährungen (1/4): Die Anfänge. 2021. [Online] Verfügbar unter: https://bisonapp.com/blog/geschichte-kryptowaehrungen-anfaenge/. Zugriff am: 13.09.2022 um 14:25 Uhr.

[3] Bergmann, C.: Kleine Geschichte der Kryptowährungen (2/4): Vom Bitcoin zu den Altcoin. 2021. [Online] Verfügbar unter: https://bisonapp.com/blog/kleine-geschichte-kryptowaehrungen-bitcoin-altcoin/. Zugriff am: 13.09.2022 um 14:38 Uhr.

[4] Becker, M.: Mit Bitcoin einkaufen: Diese Läden akzeptieren die Kryptowährung. 2021. [Online] Verfügbar unter: https://praxistipps.chip.de/mit-bitcoin-einkaufen-diese-laeden-akzeptieren-die-kryptowaehrung_102476. Zugriff am: 13.09.2022 um 14:51 Uhr.

[5] Börse Express: Zentralafrikanische Republik: Eines der ärmsten Länder der Welt führt Kryptowährung als Zahlungsmittel ein. 2022. [Online] Verfügbar unter: https://www.boerse-express.com/news/articles/zentralafrikanische-republik-eines-der-aermsten-laender-der-welt-fuehrt-kryptowaehrung-als-zahlungsmittel-ein-478294. Zugriff am: 13.09.2022 um 15:04 Uhr.

[6] Thiele, C.-L. et al.: Kryptowährung Bitcoin: Währungswettbewerb oder Spekulationsobjekt: Welche Konsequenzen sind für das aktuelle Geldsystem zu erwarten?. ifo Schnelldienst, 2017, 70, Nr. 22, 03-20. [Online] Verfügbar unter: https://www.ifo.de/publikationen/2017/aufsatz-zeitschrift/kryptowaehrung-bitcoin-waehrungswettbewerb-oder. Zugriff am 13.09.2022 um 15:32 Uhr.

[7] Investing.com: Alle Kryptowährungen. [Online] Verfügbar unter: https://de.investing.com/crypto/currencies. Zugriff am: 14.09.2022 um 16:34 Uhr.

[8] Antonopoulos, A.: Mastering Bitcoin: Programming the Open Blockchain, Second Edition. O'Reilly Media. 2017.

[9] Bitpanda: Was ist eine Wallet und wo bekomme ich eine? [Online] Verfügbar unter: https://www.bitpanda.com/academy/de/lektionen/was-ist-eine-wallet-und-wo-bekomme-ich-eine/. Zugriff am: 04.05.2022 um 15:07 Uhr.

[10] Coinbase: Was ist eine Krypto-Wallet? [Online] Verfügbar unter: https://www.coinbase.com/de/learn/crypto-basics/what-is-a-crypto-wallet. Zugriff am: 04.05.2022 um 15:43 Uhr.

[11] Bitaddress.org: Offener, clientseitiger Bitcoin-Wallet-Generator in JavaScript. [Online] Verfügbar unter: https://www.bitaddress.org/. Zugriff am: 01.08.2022 um 12:47 Uhr.

[12] Krause, F.: Paper Wallet – Erstellen, Sichern, Aufbewahren. 2022. [Online] Verfügbar unter: https://blockchainwelt.de/paper-wallet-erstellen-sichern-aufbewahren/. Zugriff am: 25.08.2022 um 13:41 Uhr.

[13] Bitcoin: bip-0039.mediawiki. [Online] Verfügbar unter: https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki. Zugriff am: 27.5.2022 um 12:41 Uhr.

[14] Electrum Bitcoin Wallet. [Online] Verfügbar unter: https://electrum.org/#home. Zugriff am: 26.05.2022 um 16:57 Uhr.

[15] Exodus. [Online] Verfügbar unter: https://www.exodus.com/. Zugriff am: 26.05.2022 um 16:58 Uhr.

[16] Firefly. [Online] Verfügbar unter: https://firefly.iota.org/. Zugriff am: 26.05.2022 um 16:59 Uhr.

[17] Wasabi wallet. [Online] Verfügbar unter: https://wasabiwallet.io/. Zugriff am: 26.05.2022 um 16:59 Uhr.

[18] Monero. [Online] Verfügbar unter: https://www.getmonero.org/. Zugriff am: 26.05.2022 um 17:00 Uhr.

[19] Ledger Live. [Online] Verfügbar unter: https://www.ledger.com/de/ledger-live. Zugriff am: 26.05.2022 um 17:01 Uhr.

[20] Guarda. [Online] Verfügbar unter: https://guarda.com/. Zugriff am: 01.09.2022 um 13:27 Uhr.

[21] Web Multi-currency Wallet | Guarda. [Online] Verfügbar unter: https://guarda.com/app/. Zugriff am: 06.09.2022 um 11:57 Uhr.

[22] Coinbase Wallet. [Online] Verfügbar unter: https://www.coinbase.com/de/wallet. Zugriff am 12.09.22 um 16:32 Uhr.

[23] Binance Extension Wallet. [Online] Verfügbar unter: https://www.bnbchain.org/ru/blog/binance-extension-wallet/. Zugriff am 12.09.2022 um 16:35 Uhr.

[24] Download MetaMask. [Online] Verfügbar unter: https://metamask.io/download/. Zugriff am: 12.09.2022 um 16:37 Uhr.

[25] Moneyland: Fiatgeld [Online] Verfügbar unter: https://www.moneyland.ch/de/fiatgeld-definition, Zugriff am 28.05.2022 um 16:50 Uhr.

[26] Carrier, B.: File System Forensic Analysis. Upper Saddle River, NJ, USA. Addison Wesley Professional. 2005.

[27] Oettinger, W.: Learn Computer Forensics, A beginner’s guide to searching, analyzing, and securing digital evidence. Birmingham, UK. Packt Publishing. 2020.

[28] Warlock: File Carving. 2018. [Online] Verfügbar unter: https://resources.infosecinstitute.com/topic/file-carving/: Zugriff am: 17.05.2022 um 15:02.

[29] Povar, D., Bhadran, V.K.: Forensic Data Carving. In: Baggili, I. (eds) Digital Forensics and Cyber Crime. ICDF2C 2010. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 53. pp 137-148. Springer, Berlin, Heidelberg. 2011. https://doi.org/10.1007/978-3-642-19513-6_12

[30] Alherbawi, N., Shukur, Z., Sulaiman, R.: Systematic Literature Review on Data Carving in Digital Forensic. Procedia Technology. Volume 11. 2013. pp 86-92. ISSN 2212-0173. https://doi.org/10.1016/j.protcy.2013.12.165.

[31] Schuster, A.: File Carving – Grundlagen und neue Techniken. [Online] Verfügbar unter: https://www.dfn-cert.de/dokumente/workshop/2008/schuster.pdf. Zugriff am: 17.05.2022 um 15:15 Uhr.

[32] Einführung in JSON. [Online] Verfügbar unter: https://www.json.org/json-de.html. Zugriff am: 26.05.2022 um 13:21 Uhr.

[33] Grenier, C.: TestDisk Documentation Release 7.1. [Online] Verfügbar unter: https://www.cgsecurity.org/testdisk.pdf, Zugriff am: 05.05.2022 um 14:37 Uhr.

[34] Developers: Adding a new file format to PhotoRec. [Online] Verfügbar unter: https://www.cgsecurity.org/wiki/Developers#Adding_a_new_file_format_to_PhotoRec. Zugriff am: 06.05.2022 um 12:34 Uhr.

[35] Testdisk & PhotoRec Source code. [Online] Verfügbar unter: https://github.com/cgsecurity/testdisk, Zugriff am: 06.04.2022 um 14:37 Uhr.

[36] Autopsy User Documentation: PhotoRec Carver Module. [Online] Verfügbar unter: https://sleuthkit.org/autopsy/docs/user-docs/3.1/photorec_carver_page.html. Zugriff am: 03.05.2022 15:01 Uhr.

[37] The Sleuth Kit® [Online] Verfügbar unter: https://www.sleuthkit.org/. Zugriff am: 03.05.2022 um 14:08 Uhr.

[38] Autopsy® [Online] Verfügbar unter: https://www.sleuthkit.org/autopsy/. Zugriff am: 03.05.2022 um 14:20 Uhr.

[39] Autopsy Forensic Browser Developer's Guide and API Reference. [Online] Verfügbar unter: https://www.sleuthkit.org/autopsy/docs/api-docs/4.19.3/. Zugriff am: 04.05.2022 um 10:15 Uhr.

[40] The Sleuth Kit Java Bindings Developer's Guide and API Reference. [Online] Verfügbar unter: http://sleuthkit.org/sleuthkit/docs/jni-docs/4.6/index.html. Zugriff am 28.05.2022 um 17:25 Uhr.

[41] Autopsy: Python Development Setup. [Online] Verfügbar unter: https://www.sleuthkit.org/autopsy/docs/api-docs/4.19.3/mod_dev_py_page.html. Zugriff am: 04.05.2022 um 11:24 Uhr.

[42] Autopsy: Module Development Overview. [Online] Verfügbar unter: https://www.sleuthkit.org/autopsy/docs/api-docs/4.19.3/platform_page.html. Zugriff am: 03.05.2022 um 9:58 Uhr.

[43] Autopsy: Developing Ingest Modules. [Online] Verfügbar unter: https://www.sleuthkit.org/autopsy/docs/api-docs/4.19.3/mod_ingest_page.html. Zugriff am: 03.05.2022 um 11:29 Uhr.

[44] Mandl, P.: Grundkurs Betriebssysteme. Architekturen, Betriebsmittelverwaltung, Synchronisation, Prozesskommunikation, Virtualisierung. Wiesbaden: Springer Vieweg. 5., aktualisierte Auflage. 2020.

[45] Baun, C.: Betriebssysteme kompakt. Wiesbaden: Springer Vieweg. 2017.

[46] Luber, S., Karlstetter, F.: Was ist Virtualisierung?. 2018. [Online] Verfügbar unter: https://www.cloudcomputing-insider.de/was-ist-virtualisierung-a-756279/. Zugriff am: 09.09.2022 um 13:25 Uhr.

[47] Tanenbaum, A., Bos, H.: Moderne Betriebssysteme. Peason Education Deutschland GmbH. 4., aktualisierte Auflage. 2016.

[48] Ruest, N.: Vergleich zwischen Typ 1 und Typ 2: Den richtigen Hypervisor auswählen. 2014. [Online] Verfügbar unter: https://www.computerweekly.com/de/tipp/Vergleich-zwischen-Typ-1-und-Typ-2-Den-richtigen-Hypervisor-auswaehlen. Zugriff am: 08.09.2022 um 13:10 Uhr.

[49] GPL – Oracle VM VirtualBox. [Online] Verfügbar unter: https://www.virtualbox.org/wiki/GPL. Zugriff am: 29.08.2022 um 11:46 Uhr.

[50] VM VirtualBox | Virtualisierung | Oracle Deutschland. [Online] Verfügbar unter: https://www.oracle.com/de/virtualization/virtualbox/. Zugriff am: 29.08.2022 um 12:13 Uhr.

[51] Data Sheet | Oracle VM VirtualBox. Version 2.0. [Online] Verfügbar unter: https://www.oracle.com/de/a/ocom/docs/oracle-vm-virtualbox-ds-1655169.pdf. Zugriff am: 29.08.2022 um 12:36 Uhr.

[52] Guest_OSes – Oracle VM VirtualBox. [Online] Verfügbar unter: https://www.virtualbox.org/wiki/Guest_OSes. Zugriff am: 09.09.2022 um 14:36 Uhr.

[53] Oracle Docs: Introduction to Virtualization - Hypervisor. [Online] Verfügbar unter: https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1011.html. Zugriff am: 08.09.2022 um 13:56 Uhr.

[54] Oracle® VM VirtualBox User Manual for Release 6.0: Snapshots. [Online] Verfügbar unter: https://docs.oracle.com/en/virtualization/virtualbox/6.0/user/snapshots.html. Zugriff am: 08.09.2022 um 14:08 Uhr.

[55] Oracle® VM VirtualBox® User Manual: Chapter 8. VBoxManage. [Online] Verfügbar unter: https://www.virtualbox.org/manual/ch08.html. Zugriff am: 22.07.2022 um 10:38 Uhr.

[56] Android Studio User guide: Meet Android Studio. [Online] Verfügbar unter: https://developer.android.com/studio/intro. Zugriff am: 30.08.2022 um 14:49 Uhr.

[57] Android Studio User guide: Android Debug Bridge (adb). [Online] Verfügbar unter: https://developer.android.com/studio/command-line/adb. Zugriff am: 08.09.2022 um 14:15 Uhr.

[58] Gehrig, A.: Strategie zum Nachweis der Nutzung von Kryptowährungen (Masterthesis). 2021. [Online] Verfügbar unter: https://it-forensik.fiw.hs-wismar.de/images/0/03/MT_AGehrig1.pdf. Zugriff am: 08.09.2022 um 16:24 Uhr.

[59] Cartes, C.: Digitale Währungen – Identifikation, Evaluation und automatisierte Suche digitaler Spuren in forensischen Sicherungen (Masterthesis). 2020. [Online] Verfügbar unter: https://it-forensik.fiw.hs-wismar.de/images/5/5c/MT_CCartes.pdf. Zugriff am: 08.09.2022 um 16:55 Uhr.

[60] Autopsy 4.16 Release Highlights. 2020. [Online] Verfügbar unter: https://www.autopsy.com/autopsy-4-16-release-highlights/. Zugriff am: 09.09.2022 um 16:30 Uhr.

[61] McKinnon, M.: Autopsy-Plugins: Atomic-Wallet. [Online] Verfügbar unter: https://github.com/markmckinnon/Autopsy-Plugins/tree/master/Atomic_Wallet. Zugriff am: 12.09.2022 um 10:39 Uhr.

[62] Kessler, G.C.: GCK'S FILE SIGNATURES TABLE. [Online] Verfügbar unter: https://www.garykessler.net/library/file_sigs.html. Zugriff am: 25.05.2022 um 17:35 Uhr.

[63] File_template.c. [Online] Verfügbar unter: https://git.cgsecurity.org/cgit/testdisk/tree/src/file_template.c. Zugriff am: 25.05.2022 um 18:02 Uhr.

[64] Compile Win - CGSecurity. [Online] Verfügbar unter: https://www.cgsecurity.org/wiki/Compile_Win. Zugriff am: 29.07.2022 um 14:57 Uhr.

[65] BIP39 Wortliste. [Online] Verfügbar unter: https://github.com/bitcoin/bips/blob/master/bip-0039/english.txt. Zugriff am: 28.05.2022 um 17:45 Uhr.

[66] Download debian-live-11.3.0-amd64-cinnamon.iso. [Online] Verfügbar unter: https://cdimage.debian.org/debian-cd/current-live/amd64/iso-hybrid/. Zugriff am: 06.07.2022 um 18:28 Uhr.

[67] Downloads – Oracle VM VirtualBox. [Online] Verfügbar unter: https://www.virtualbox.org/wiki/Downloads. Zugriff am: 15.08.2022 um 13:56 Uhr.

[68] Gupta, V.: Download macOS Mojave VMDK – Latest Version. 2022. [Online] Verfügbar unter: https://techrechard.com/download-macos-mojave-vmdk-file-latest-version/. Zugriff am: 22.08.2022 um 15:38 Uhr.

[69] Gupta, V.: Install MacOS Mojave on VirtualBox on Windows PC: 5 Easy Steps. 2021. [Online] Verfügbar unter: https://techrechard.com/install-macos-mojave-on-virtualbox-on-windows-pc/. Zugriff am: 20.07.2022 um 12:13 Uhr.

[70] Gupta, V.: Install MacOS Mojave on VirtualBox on Windows PC: 5 Easy Steps: 3. Apply the Code on VirtualBox. 2021. [Online] Verfügbar unter: https://techrechard.com/install-macos-mojave-on-virtualbox-on-windows-pc/#3_Apply_the_Code_on_VirtualBox. Zugriff am: 20.07.2022 um 12:13 Uhr.

[71] Virtuelle Computer – Microsoft Edge Developer. [Online] Verfügbar unter: https://developer.microsoft.com/de-de/microsoft-edge/tools/vms/. Zugriff am: 19.07.2022 um 11:41 Uhr.

[72] Jamas, J.: Imaging Android with ADB, Root, Netcat and DD. [Online] Verfügbar unter: https://dfir.science/2017/04/Imaging-Android-with-root-netcat-and-dd.html. Zugriff am: 19.07.2022 um 11:41 Uhr.

[73] Newbit1: rootAVD. [Online] Verfügbar unter: https://github.com/newbit1/rootAVD. Zugriff am: 16.07.2022 um 15:49 Uhr.

[74] Electrum Bitcoin Wallet: download latest release. [Online] Verfügbar unter: https://electrum.org/#download. Zugriff am: 02.07.2022 um 12:50 Uhr.

[75] Download Electrum. [Online] Verfügbar unter: https://download.electrum.org/. Zugriff am: 23.06.2022 um 14:37 Uhr.

[76] Crypto Wallet App Download for Desktop & Mobile | Exodus Download. [Online] Verfügbar unter: https://www.exodus.com/download/. Zugriff am: 23.06.2022 um 14:52 Uhr.

[77] Download Exodus. [Online] Verfügbar unter: https://downloads.exodus.io/releases/exodus-windows-x64-19.1.3.exe. Zugriff am: 23.06.2022 um 15:24 Uhr.

[78] Firefly Releases. [Online] Verfügbar unter: https://github.com/iotaledger/firefly/releases. Zugriff am: 25.06.2022 um 10:48 Uhr.

[79] Download Wasabi Wallet. [Online] Verfügbar unter: https://wasabiwallet.io/index.html#download. Zugriff am: 26.06.2022 um 15:21 Uhr.

[80] Wasabi Wallet Releases. [Online] Verfügbar unter: https://github.com/zkSNACKs/WalletWasabi/releases. Zugriff am: 26.06.2022 um 15:46 Uhr.

[81] Downloads | Monero – secure, private, untraceable. [Online] Verfügbar unter: https://www.getmonero.org/downloads/. Zugriff am: 27.06.2022 um 11:27 Uhr.

[82] Monero GUI Wallet Releases. [Online] Verfügbar unter: https://github.com/monero-project/monero-gui/releases. Zugriff am: 27.06.2022 um 12:06 Uhr.

[83] Ledger Live Releases. [Online] Verfügbar unter: https://github.com/LedgerHQ/ledger-live-desktop/releases. Zugriff am: 02.07.2022 um 14:51 Uhr.

[84] TestDisk and PhotoRec in various digital forensics testcase. [Online] Verfügbar unter: https://www.cgsecurity.org/wiki/TestDisk_and_PhotoRec_in_various_digital_forensics_testcase. Zugriff am: 30.08.2022 um 10:28 Uhr.

[85] McKinnon, M.: Leveldb GUI Viewer and Dumper. [Online] Verfügbar unter: https://github.com/markmckinnon/Leveldb-py. Zugriff am: 05.09.2022 um 15:07 Uhr.

Anlagen

Anlagen, Anwendung des Moduls

1. Editiertes PhotoRec in Autopsy einbinden

Um das Modul anzuwenden, muss die aktuelle von Autopsy genutzte PhotoRec-Version mit der editierten Version ersetzt werden:

- Dazu Ordner „bin“ aus „C:\Programme\Autopsy-<Version>\autopsy\photorec_exe\“ z.B. auf den Desktop verschieben
- Und in „C:\Programme\Autopsy-<Version>\autopsy\photorec_exe\“ den Ordner „bin“ aus dem PhotoRec-Verzeichnis schieben
- Wenn PhotoRec auf dem PC bereits genutzt wurde, sollte PhotoRec ausgeführt werden und sichergestellt sein, dass alle Dateitypen ausgewählt sind

Wenn die Standardversion von PhotoRec wieder gebraucht wird: „bin“-Ordner wieder tauschen.

2. Autopsy-Modul im Autopsy-Modul-Ordner einfügen

Der Python-Modulordner lässt sich über das Autopsy-Menü (Tools → Python Plugins) öffnen oder über den Pfad „C:\Users\<User>\AppData\Roaming\autopsy\python_modules\“. Hier muss der Ordner „KryptoCarving“ eingefügt werden.

3. Ausführen des PhotoRec Carver-Moduls

Zuerst das PhotoRec Carver-Modul alleine ausführen.

- Autopsy-Menü: Tools → Run Ingest Modules → Datenträger auswählen → Modul „PhotoRec Carver“ auswählen → Finish

Das PhotoRec Carver Modul sollte beendet sein, bevor das Autopsy-Modul „KryptoCarving“ ausgeführt wird.

4. Modul „KryptoCarving“ ausführen

Anschließend kann das Modul „KryptoCarving“ ausgeführt werden.

- Autopsy-Menü: Tools → Run Ingest Modules → Datenträger auswählen → Modul „KryptoCarving“ auswählen → Finish

Gefundene Dateien werden als „Interesting File“ hinzugefügt.

Hinweise zu den Dateien stehen in der „Configuration“ der Metadaten der Datei.

Hinweis: Autopsy hat die Suche nach exe-Dateien von Walletsoftwares im Modul „Interesting Files Identifier“ integriert.

Das Modul wurde mit der Autopsy-Version 4.19.3 getestet.

Anlagen, Signaturen PhotoRec Modulversion 1

Hinzugefügte Dateien:

Folgende Dateien werden zum Hinzufügen neuer Signaturen in PhotoRec erstellt (diese befinden sich im Verzeichnis „PhotoRec\PhotoRec – Dateien\“):

- file_asar.c
- file_cert.c
- file_electrumConfig.c
- file_electrumWallet.c
- file_exodusConfig.c
- file_exodusPassphrase.c
- file_fireflyAppupdate.c
- file_html.c
- file_ledgerAppupdate.c
- file_moneroUninsDat.c
- file_seco.c
- file_stronghold.c
- file_wasabiConfig.c
- file_wasabiWallet.c

- file_bitcoinCoreWallet.c
- file_multibitWallet.c
- file_multibitWalletInfo.c

Editierte Dateien:

- file_list.c
- Makefile.am

Auflistung aller Signaturen:

Electrum:

Zertifikat (.cert):

Header: ‚-----*BEGIN CERTIFICATE*-----'

Footer: ‚-----*END CERTIFICATE*-----‘

Wallet (unverschlüsselt) (.electrumWallet):

Header: ‚*{\r\n “addr_history“: {\r\n*‘

Footer: ‚*n}*‘

Config (.electrumConfig):

Header: ‚*{\r\n “auto_connect“:* ‘

Footer: ‚*r\n}*‘

Exodus:

Config (.exodusConfig):

Header: ‚*{\n “meta“: {\n “configVersion“: “*

Footer: ‚*n}\n*‘

Passphrase (.exodusPassphrase):

Header: ‚*{\n “passphrase“: “*

Footer: ‚*n}\n*‘

Secure Container (.seco):

Header: ‚*SECO*‘

maximale Dateigröße: 33284 Bytes (dies war die Größe aller betrachteten Seco-Dateien)

Firefly:

App-update.yml (.fireflyAppupdate)

Header: ‚*provider: generic\nurl: 'https://dl.firefly.iota.org/'\npublishAutoUpdate:* ‘

Footer: ‚*publisherName:\n - IOTA Stiftung* ‘

Wallet (.stronghold)

Header: '*PARTI*', 0x02, 0x00

Wasabi:

Wasabi (.wasabiWallet)

Header: ‘*{\r\n “EncryptedSecret“: “*‘

Footer: ‚*\n}*‘

Config (.wasabiConfig)

Header: 0xef, 0xbb, 0xbf, ‚*{\r\n "Network": "*‘

Footer: ‚*\n}*‘

Monero:

Unins000.dat (.moneroUninsDat)

Header: ‚*Inno Setup Uninstall Log (b) 64-bit*‘, 0x00, ‚*Monero GUI Wallet*‘

Readme.htm (.html)

Header: ‚*<html>*‘

Footer: ‚*</html>*‘

Ledger Live:

App-update.yml (.ledgerAppupdate)

Header: ‚*owner: LedgerHQ\nrepo: ledger-live-desktop\nprovider:* ‘

Footer: ‚*updaterCacheDirName: ledger-live-desktop-updater* ‘

Asar-Archiv (.asar) (Bei Ledger, Firefly und Exodus)

1. Header: 0x04, 0x00, 0x00, 0x00

2. Header ab Stelle 16 der Datei: ‚*{“files“:{*“‘

Signaturen von garykessler.net: [https://www.garykessler.net/library/file_sigs.html]

BitcoinCore Wallet (.bitcoinCoreWallet):

Header (ab Stelle 9 der Datei): 0x00, 0x00, 0x00, 0x00, 0x62, 0x31, 0x05, 0x00

multiBit Wallet (.multibitWallet):

Header: 0x0A, 0x16, 0x6F, 0x72, 0x67, 0x2E, 0x62, 0x69, 0x74, 0x63, 0x6F, 0x69, 0x6E, 0x2E, 0x70, 0x72

Multibit Wallet information file (.multibitWalletInfo):

Header: 0x6D, 0x75, 0x6C, 0x74, 0x69, 0x74, 0x2E, 0x69, 0x6E, 0x66, 0x6F

Anlagen, Signaturen PhotoRec Modulversion 2

Hinzugefügte Dateien:

Folgende Dateien werden zum Hinzufügen neuer Signaturen in PhotoRec erstellt:

- file_asar.c
- file_cert.c
- file_chromeExtManifest.c
- file_doctypeHtml.c
- file_electrumConfig.c
- file_electrumDat.c
- file_electrumWallet.c
- file_exodusConfig.c
- file_exodusPassphrase.c
- file_fireflyAppupdate.c
- file_firefoxExtensions.c
- file_guardaAppupdate.c
- file_html.c
- file_ledgerAppupdate.c
- file_moneroUninsDat.c
- file_seco.c
- file_stronghold.c
- file_wasabiConfig.c
- file_wasabiWallet.c

- file_bitcoinCoreWallet.c
- file_multibitWallet.c
- file_multibitWalletInfo.c

Editierte Dateien:

- file_list.c
- Makefile.am

Electrum:

Zertifikat (.cert):

Header: ‚*-----BEGIN CERTIFICATE-----*'

Footer: ‚*-----END CERTIFICATE-----*‘

Wallet (unverschlüsselt) (.electrumWallet):

Header:

1. Variante: ‚*{\r\n “addr_history“: {\r\n*‘
2. Variante: ‚*{\n “addr_history“: {\n*‘
3. Variante: ‚*{\r\n "accounts": {\r\n*‘
4. Variante: ‚*{\n "accounts": {\n*‘

Footer: ‚*\n}*‘

Config (.electrumConfig):

Header:

1. Variante: ‚*{\r\n “auto_connect“:* ‘
2. Variante: ‚*{\n "auto_connect":* ‘

Footer:

1. Variante: ‚*\r\n}*‘
2. Variante: ‚*\n}*‘

Electrum.dat (.electrumDat):

Header: ‚*{'addr_history':*‘

Footer: ‚*]}*‘

Exodus:

Config (.exodusConfig):

Header: ‚*{\n “meta“: {\n   “configVersion“: “*‘

Footer: ‚*\n}\n*‘

Passphrase (.exodusPassphrase):

Header: ‚*{\n “passphrase“: “*‘

Footer: ‚*\n}\n*‘

Secure Container (.seco):

Header: ‚*SECO*‘

maximale Dateigröße: 33284 Bytes (dies war die Größe aller betrachteten Seco-Dateien)

Firefly:

App-update.yml (.fireflyAppupdate)

Header:

1. Variante: ‚*provider: generic\nurl: 'https://dl.firefly.iota.org/'\npublishAutoUpdate:* ‘
2. Variante: ‚*provider: generic\nurl: 'https://iotaledger-files*‘

Footer: ‚*publisherName:\n - IOTA Stiftung* ‘

Wallet (.stronghold)

Header: '*PARTI*', 0x02, 0x00

Wasabi:

Wasabi (.wasabiWallet)

Header:

1. Variante: 0xef, 0xbb, 0xbf, ‚*{\r\n "EncryptedSecret": "*‘
2. Variante: 0xef, 0xbb, 0xbf, ‚*{\n "EncryptedSecret": "*‘

Footer: ‚*\n}*‘

Config (.wasabiConfig)

Header:

1. Variante: 0xef, 0xbb, 0xbf, ‚*{\r\n "Network": "*‘
2. Variante: 0xef, 0xbb, 0xbf, ‚*{\n "Network": "*‘

Footer: ‚*\n}*‘

Monero:

Unins000.dat (.moneroUninsDat)

Header: ‚*Inno Setup Uninstall Log (b) 64-bit*‘, 0x00, ‚*Monero GUI Wallet*‘

Readme.htm (.html)

Header: ‚*<html>*‘

Footer: ‚*</html>*‘

Ledger Live:

App-update.yml (.ledgerAppupdate)

Header:

1. Variante: ‚*owner: LedgerHQ\nrepo: ledger-live-desktop\nprovider:* ‘
2. Variante: ‚*provider: generic\nurl: https://download.live.ledger*‘
3. Variante: ‚*owner: LedgerHQ\nrepo*‘

Footer:

1. Variante: ‚*updaterCacheDirName: ledger-live-desktop-updater* ‘
2. Variante: ‚*ledger-live-desktop\nprovider: github*‘
3. Variante: ‚*publisherName:\n - Ledger SAS\n*‘

Guarda:

App-update.yml (.guardaAppupdate)

Header: ‚*owner: guardaco\nrepo: guarda-electron*‘

Footer: ‚*publisherName:\n - Guardarian OÜ\n*‘

Asar-Archiv (.asar) (Bei Ledger Live, Firefly, Exodus und Guarda)

1. Header: 0x04, 0x00, 0x00, 0x00

2. Header ab Stelle 16 (0x10) der Datei: ‚*{“files“:{“*

Browser Erweiterungen:

Manifest.json (.chromeExtManifest):

Header:

1. Variante: ‚*{\r\n "background": {*‘
2. Variante: ‚*{\n "background": {*‘
3. Variante: ‚*{\r\n "author":* ‘
4. Variante: ‚*{\n "author":* ‘
5. Variante: ‚*{\r\n "app": {*‘
6. Variante: ‚*{\n "app": {*‘

Footer: ‚*\n}*‘

Background.html, home.html (.html):

Header: ‚*<!doctype html>*‘

Footer: ‚*</html>*‘

Extensions.json (.firefoxExtensions):

Header: ‚*{"schemaVersion":*‘

Footer: ‚*]}*‘

Signaturen von garykessler.net: [https://www.garykessler.net/library/file_sigs.html]

BitcoinCore Wallet (.bitcoinCoreWallet):

Header (ab Stelle 9 der Datei): 0x00, 0x00, 0x00, 0x00, 0x62, 0x31, 0x05, 0x00

multiBit Wallet (.multibitWallet):

Header: 0x0A, 0x16, 0x6F, 0x72, 0x67, 0x2E, 0x62, 0x69, 0x74, 0x63, 0x6F, 0x69, 0x6E, 0x2E, 0x70, 0x72

Multibit Wallet information file (.multibitWalletInfo):

Header: 0x6D, 0x75, 0x6C, 0x74, 0x69, 0x74, 0x2E, 0x69, 0x6E, 0x66, 0x6F

Anlagen, gesuchte Dateien Modulversion 1

Electrum

- Zertifikate (Dateisystem + Carven)
- Recent_servers (Dateisystem)
- Config-Datei (Dateisystem + Carven)
- Wallet-Datei (unverschlüsselt) (Dateisystem + Carven)
- Wallet-Datei (verschlüsselt) (Dateisystem (über Pfad))

Exodus

- Wallet: Secure Container (Dateisystem + Carven)
- Wallet: passphrase.json (Dateisystem + Carven)
- Config-Datei (Dateisystem + Carven)
- Exodus.exe (Dateisystem)
- Asar (Dateisystem + Carven)

Wasabi

- Wallet-Datei (Dateisystem + Carven)
- Config (Dateisystem + Carven)
- Transactions.dat (Dateisystem (über Pfad))

Firefly

- Wallet: Stronghold (Dateisystem + Carven)
- App-update.yml (Dateisystem + Carven)
- Firefly.exe (Dateisystem)
- Asar (Dateisystem + Carven)

Monero

- Unins000.dat (Dateisystem)
- Exe-Datei (Dateisystem)
- Readme.htm (Dateisystem + Carven)
- Wallet: .keys (Dateisystem (über Pfad))
- Wallet: .address.txt (Dateisystem (über Pfad))

Ledger Live

- App-update.yml (Dateisystem + Carven)
- Ledger Live.exe (Dateisystem)
- Asar (Dateisystem + Carven)

BitcoinCore

- Wallet (Carven)

multiBit

- Wallet (Carven)
- Wallet Info (Carven)

Anlagen, gesuchte Dateien Modulversion 2

Electrum

- Zertifikate (Dateisystem + Carven)
- Recent_servers (Dateisystem)
- Config-Datei (Dateisystem + Carven)
- Wallet-Datei (unverschlüsselt) (Dateisystem + Carven)
- Wallet-Datei (verschlüsselt) (Dateisystem (über Pfad))
- Electrum.dat (Dateisystem + Carven)

Exodus

- Wallet: Secure Container (Dateisystem + Carven)
- Wallet: passphrase.json (Dateisystem + Carven)
- Config-Datei (Dateisystem + Carven)
- Exodus.exe (Dateisystem)
- Asar (Dateisystem + Carven)
- Android: RKStorage (Dateisystem)

Wasabi

- Wallet-Datei (Dateisystem + Carven)
- Config (Dateisystem + Carven)
- Transactions.dat (Dateisystem (über Pfad))

Firefly

- Wallet: Stronghold (Dateisystem + Carven)
- App-update.yml (Dateisystem + Carven)
- Firefly.exe (Dateisystem)
- Asar (Dateisystem + Carven)

Monero

- Unins000.dat (Dateisystem + Carven)
- Exe-Datei (Dateisystem)
- Readme.htm (Dateisystem + Carven)
- Wallet: .keys (Dateisystem (über Pfad))
- Wallet: .address.txt (Dateisystem (über Pfad))
- Wallet: Wallet-Datei (Dateisystem (über Pfad))

Ledger Live

- App-update.yml (Dateisystem + Carven)
- Ledger Live.exe (Dateisystem)
- Asar (Dateisystem + Carven)

Guarda

- Asar (Dateisystem + Carven)
- App-update.yml (Dateisystem + Carven)
- Wallet: guarda-backup.txt / guarda-multiwallet-backup.txt

Browser Erweiterungen

Binance

- Chrome: manifest.json (Dateisystem + Carven)
- Chrome: Background.html (Dateisystem + Carven)
- Firefox: extensions.json (Dateisystem + Carven)

Coinbase

- Chrome: manifest.json (Dateisystem + Carven)
- Chrome: index.html (Dateisystem + Carven)

MetaMask

- Chrome: manifest.json (Dateisystem + Carven)
- Chrome: home.html (Dateisystem + Carven)
- Firefox: @metamask.io.xpi (Dateisystem)
- Firefox: extensions.json (Dateisystem + Carven)

BitcoinCore

- Wallet (Carven)

multiBit

- Wallet (Carven)
- Wallet Info (Carven)

Anlagen, Ergebnisse der Evaluation (tabellarisch)

Legende:

✗ - Datei wurde vom Modul nicht gefunden

✓ - Datei wurde vom Modul gefunden

(nicht vorhanden) – Datei ist im System nicht vorhanden

(wird nicht gesucht) – Datei wird nicht gesucht

(wird nicht gecarvt) – Datei wird nicht gecarvt

Linux

Tabelle 3: Ergebnisse der Evaluation des Linux-Systems der ersten Modulversion (links) im Vergleich zur zweiten Modulversion (rechts)

Datei	**Erste Modulversion**		**Zweite Modulversion**	
	Dateisystem	**Carven**	**Dateisystem**	**Carven**
Electrum – Wallet-Datei (unverschlüsselt)	✗	✗	✓	✓
Electrum – Wallet-Datei (verschlüsselt)	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Electrum – Electrum.dat	(wird nicht gesucht)	(wird nicht gecarvt)	(nicht vorhanden)	(nicht vorhanden)
Electrum – Config-Datei	✗	✗	✓	✓

Electrum – „recent_servers“	✗	(wird nicht gecarvt)	✗	(wird nicht gecarvt)
Electrum – Zertifikate	✗	✗	✓	✓
Exodus – Secure Container	✓	✓	✓	✓
Exodus – „passphrase.json“	✓	✓	✓	✓
Exodus – Config-Datei	✓	✓	✓	✓
Exodus – Exodus.exe	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Exodus – „app.asar“	✓	✓	✓	✓
Firefly – Stronghold	✓	✓	✓	✓
Firefly – „app-update.yml“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Firefly – Firefly.exe	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Firefly – „app.asar“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Wasabi – Wallet-Datei	✗	✗	✓	✓

Wasabi – Config-Datei	✗	✗	✓	✓
Wasabi – Transac-tions.dat	✗	(wird nicht gecarvt)	✗	(wird nicht gecarvt)
Monero – „unins000.dat“	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(nicht vorhanden)
Monero – exe-Datei	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Monero – „read-me.htm“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Monero – „.keys“-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero – „.address.txt“-Datei	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Monero – Wallet-Datei	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Ledger Live – „app-update.yml“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Ledger Live – Ledger Live.exe	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Ledger Live – „app.asar“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)

Mac

Tabelle 4: Ergebnisse der Evaluation des Mac-Systems der ersten Modulversion (links) im Vergleich zur zweiten Modulversion (rechts)

Datei	Erste Modulversion		Zweite Modulversion	
	Dateisystem	**Carven**	**Dateisystem**	**Carven**
Electrum – Wallet-Datei (unverschlüsselt)	✗	✗	✓	✓
Electrum – Wallet-Datei (verschlüsselt)	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Electrum – Electrum.dat	(wird nicht gesucht)	(wird nicht gecarvt)	(nicht vorhanden)	(nicht vorhanden)
Electrum – Config-Datei	✗	✗	✓	✓
Electrum – „recent_servers“	✗	(wird nicht gecarvt)	✗	(wird nicht gecarvt)
Electrum – Zertifikate	✗	✗	✓	✓
Exodus – Secure Container	✓	✓	✓	✓
Exodus – „passphrase.json“	✓	✓	✓	✓
Exodus – Config-Datei	✓	✓	✓	✓

Exodus – Exodus.exe	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Exodus – „app.asar“	✗	✗	✗	✗
Firefly – Stronghold	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Firefly – „app-update.yml“	✗	✗	✗	✗
Firefly – Firefly.exe	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Firefly – „app.asar“	✓	✓	✓	✓
Wasabi – Wallet-Datei	✗	✗	✓	✓
Wasabi – Config-Datei	✗	✗	✓	✓
Wasabi – Transac-tions.dat	✗	(wird nicht gecarvt)	✗	(wird nicht gecarvt)
Monero – „unins000.dat“	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(nicht vorhanden)
Monero – exe-Datei	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Monero – „read-me.htm“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)

Monero – „.keys“-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero – „.address.txt“-Datei	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Monero – Wallet-Datei	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Ledger Live – „app-update.yml“	×	×	✓	✓
Ledger Live – Ledger Live.exe	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Ledger Live – „app.asar“	×	×	×	×

Windows – Electrum

Tabelle 5: Ergebnisse der Evaluation der Electrum-Versionen des Windows-Systems. Links sind die Ergebnisse der ersten, rechts die Ergebnisse der zweiten Modulversion dargestellt.

Datei	**Erste Modulversion**		**Zweite Modulversion**	
	Dateisystem	**Carven**	**Dateisystem**	**Carven**
Electrum v1.8 – Wallet-Datei (unverschlüsselt)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Electrum v1.8 – Wallet-Datei (verschlüsselt)	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)

Electrum v1.8 – Electrum.dat	(wird nicht gesucht)	(wird nicht gecarvt)	✓	✓
Electrum v1.8 – Config-Datei	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Electrum v1.8 – „recent_servers“	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Electrum v1.8 – Zertifikate	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Electrum v2.5 – Wallet-Datei (unverschlüsselt)	✓	✗	✓	✓
Electrum v2.5 – Wallet-Datei (verschlüsselt)	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Electrum v2.5 – Electrum.dat	(wird nicht gesucht)	(wird nicht gecarvt)	(nicht vorhanden)	(nicht vorhanden)
Electrum v2.5 – Config-Datei	✗	✗	✓	✗
Electrum v2.5 – „recent_servers“	✗	(wird nicht gecarvt)	✗	(wird nicht gecarvt)
Electrum v2.5 – Zertifikate	✓	✓	✓	✓

Electrum v3.2.0 – Wallet-Datei (unverschlüsselt)	✓	×	✓	✓
Electrum v3.2.0 – Wallet-Datei (verschlüsselt)	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Electrum v3.2.0 – Electrum.dat	(wird nicht gesucht)	(wird nicht gecarvt)	(nicht vorhanden)	(nicht vorhanden)
Electrum v3.2.0 – Config-Datei	×	×	✓	×
Electrum v3.2.0 – „recent_servers“	×	(wird nicht gecarvt)	×	(wird nicht gecarvt)
Electrum v3.2.0 – Zertifikate	✓	✓	✓	✓
Electrum v4.2.2 – Wallet-Datei (unverschlüsselt)	✓	✓	✓	✓
Electrum v4.2.2 – Wallet-Datei (verschlüsselt)	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Electrum v4.2.2 – Electrum.dat	(wird nicht gesucht)	(wird nicht gecarvt)	(nicht vorhanden)	(nicht vorhanden)
Electrum v4.2.2 – Config-Datei	✓	✓	✓	✓

Electrum v4.2.2 – „recent_servers“	✗	(wird nicht gecarvt)	✗	(wird nicht gecarvt)
Electrum v4.2.2 – Zertifikate	✓	✓	✓	✓

Windows – Exodus

Tabelle 6: Ergebnisse der Evaluation der Exodus-Versionen des Windows-Systems. Links sind die Ergebnisse der ersten Modulversion dargestellt, rechts die Ergebnisse der zweiten Modulversion.

Datei	**Erste Modulversion**		**Zweite Modulversion**	
	Dateisystem	**Carven**	**Dateisystem**	**Carven**
Exodus v19.1.3 – Secure Container	✓	✓	✓	✓
Exodus v19.1.3 – „passphrase.json“	✓	✗	✓	✗
Exodus v19.1.3 – Config-Datei	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Exodus v19.1.3 – Exodus.exe	✗	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Exodus v19.1.3 – „app.asar“	✗	✗	✗	✗

Exodus v20.1.3 – Secure Container	✓	✓	✓	✓
Exodus v20.1.3 – „passphrase.json“	✓	✗	✓	✗
Exodus v20.1.3 – Config-Datei	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Exodus v20.1.3 – Exodus.exe	✗	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Exodus v20.1.3 – „app.asar“	✗	✗	✗	✗
Exodus v21.1.7 – Secure Container	✓	✓	✓	✓
Exodus v21.1.7 – „passphrase.json“	✓	✗	✓	✗
Exodus v21.1.7 – Config-Datei	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Exodus v21.1.7 – Exodus.exe	✗	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Exodus v21.1.7 – „app.asar“	✗	✗	✓	✓

Exodus v22.7.1 – Secure Container	✓	✓	✓	✓
Exodus v22.7.1 – „passphrase.json“	✓	✗	✓	✗
Exodus v22.7.1 – Config-Datei	✓	✓	✓	✓
Exodus v22.7.1 – Exodus.exe	✗	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Exodus v22.7.1 – „app.asar“	✓	✗	✓	✗

Windows – Firefly

Tabelle 7: Ergebnisse der Evaluation der Firefly-Versionen des Windows-Systems. Links sind die Ergebnisse der ersten Modulversion dargestellt, rechts die Ergebnisse der zweiten Modulversion.

Datei	**Erste Modulversion**		**Zweite Modulversion**	
	Dateisystem	**Carven**	**Dateisystem**	**Carven**
Firefly v0.0.2 – Stronghold	✓	✓	✓	✓
Firefly v0.0.2 – „app-update.yml“	✗	✗	✓	✗
Firefly v0.0.2 – Firefly.exe	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)

Firefly v0.0.2 – „app.asar“	✓	✓	✓	✓
Firefly v1.0.0 – Stronghold	✓	✓	✓	✓
Firefly v1.0.0 – „app-update.yml“	✗	✗	✓	✗
Firefly v1.0.0 – Firefly.exe	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Firefly v1.0.0 – „app.asar“	✓	✓	✓	✓
Firefly v1.3.3 – Stronghold	✓	✓	✓	✓
Firefly v1.3.3 – „app-update.yml“	✓	✗	✓	✗
Firefly v1.3.3 – Firefly.exe	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Firefly v1.3.3 – „app.asar“	✓	✓	✓	✓
Firefly v1.6.2 – Stronghold	✓	✓	✓	✓
Firefly v1.6.2 – „app-update.yml“	✓	✗	✓	✗

Firefly v1.6.2 – Firefly.exe	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Firefly v1.6.2 – „app.asar“	✓	✓	✓	✓

Windows – Wasabi

Tabelle 8: Ergebnisse der Evaluation der Wasabi-Versionen des Windows-Systems. Links sind die Ergebnisse der ersten Modulversion dargestellt, rechts die Ergebnisse der zweiten Modulversion.

Datei	**Erste Modulversion**		**Zweite Modulversion**	
	Dateisystem	**Carven**	**Dateisystem**	**Carven**
Wasabi v0.9.3 – Wallet-Datei	✓	×	✓	✓
Wasabi v0.9.3 – Config-Datei	×	×	✓	✓
Wasabi v0.9.3 – Transactions.dat	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Wasabi v1.0 – Wallet-Datei	✓	×	✓	✓
Wasabi v1.0 – Config-Datei	×	×	✓	✓
Wasabi v1.0 – Transactions.dat	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)

Wasabi v1.1.7 – Wallet-Datei	✓	✗	✓	✓
Wasabi v1.1.7 – Config-Datei	✗	✗	✓	✓
Wasabi v1.1.7 – Transactions.dat	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Wasabi v2.0.1.1 – Wallet-Datei	✓	✓	✓	✓
Wasabi v2.0.1.1 – Config-Datei	✓	✓	✓	✓
Wasabi v2.0.1.1 – Transactions.dat	✗	(wird nicht gecarvt)	✗	(wird nicht gecarvt)

Windows – Monero

Tabelle 9: Ergebnisse der Evaluation der Monero-Versionen des Windows-Systems. Links sind die Ergebnisse der ersten Modulversion dargestellt, rechts die Ergebnisse der zweiten Modulversion.

Datei	**Erste Modulversion**		**Zweite Modulversion**	
	Dateisystem	**Carven**	**Dateisystem**	**Carven**
Monero v0.11.0.0 – „unins000.dat“	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(nicht vorhanden)
Monero v0.11.0.0 – exe-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)

Monero v0.11.0.0 – „readme.htm“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Monero v0.11.0.0 – „.keys“-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.11.0.0 – „.address.txt“-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.11.0.0 – Wallet-Datei	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.14.0.0 – „unins000.dat“	✓	(wird nicht gecarvt)	✓	✓
Monero v0.14.0.0 – exe-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.14.0.0 – „readme.htm“	✓	✓	✓	✓
Monero v0.14.0.0 – „.keys“-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.14.0.0 – „.address.txt“-Datei	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Monero v0.14.0.0 – Wallet-Datei	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)

Monero v0.16.0.0 – „unins000.dat“	✓	(wird nicht gecarvt)	✓	✓
Monero v0.16.0.0 – exe-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.16.0.0 – „readme.htm“	✓	×	✓	×
Monero v0.16.0.0 – „.keys“-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.16.0.0 – „.address.txt“-Datei	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Monero v0.16.0.0 – Wallet-Datei	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.17.1.0 – „unins000.dat“	✓	(wird nicht gecarvt)	✓	✓
Monero v0.17.1.0 – exe-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.17.1.0 – „readme.htm“	✓	✓	✓	✓
Monero v0.17.1.0 – „.keys“-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.17.1.0 – „.address.txt“-Datei	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)

Monero v0.17.1.0 – Wallet-Datei	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.17.3.2 – „unins000.dat“	✓	(wird nicht gecarvt)	✓	✓
Monero v0.17.3.2 – exe-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.17.3.2 – „readme.htm“	✓	×	✓	✓
Monero v0.17.3.2 – „.keys“-Datei	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Monero v0.17.3.2 – „.address.txt“-Datei	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Monero v0.17.3.2 – Wallet-Datei	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)

Windows – Ledger Live

Tabelle 10: Ergebnisse der Evaluation der Ledger Live-Versionen des Windows-Systems. Links sind die Ergebnisse der ersten Modulversion dargestellt, rechts die Ergebnisse der zweiten Modulversion.

Datei	Erste Modulversion		Zweite Modulversion	
	Dateisystem	Carven	Dateisystem	Carven
Ledger Live v1.0.0-beta – „app-update.yml“	✗	✗	✓	✗
Ledger Live v1.0.0-beta – Ledger Live.exe	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Ledger Live v1.0.0-beta – „app.asar“	✓	✓	✓	✓
Ledger Live v1.6.0 – „app-update.yml“	✗	✗	✓	✗
Ledger Live v1.6.0 – Ledger Live.exe	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Ledger Live v1.6.0 – „app.asar“	✓	✓	✓	✓
Ledger Live v2.0.0 – „app-update.yml“	✓	✗	✓	✗
Ledger Live v2.0.0 – Ledger Live.exe	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)

Ledger Live v2.0.0 – „app.asar“	✓	✓	✓	✓
Ledger Live v2.44.0 – „app-update.yml“	×	×	✓	×
Ledger Live v2.44.0 – Ledger Live.exe	✓	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Ledger Live v2.44.0 – „app.asar“	✓	✓	✓	✓

Windows – Guarda

Tabelle 11: Ergebnisse der Evaluation der Guarda Desktop Wallet. Links sind die Ergebnisse der ersten Modulversion dargestellt, rechts die Ergebnisse der zweiten Modulversion.

Datei	**Erste Modulversion**		**Zweite Modulversion**	
	Dateisystem	**Carven**	**Dateisystem**	**Carven**
Guarda v1.0.20 – „app.asar“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	✓
Guarda v1.0.20 – „app-update.yml“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	×
Guarda v1.0.20 – „guarda-backup.txt“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)

Windows – Browser-Erweiterungen

Tabelle 12: Ergebnisse der Evaluation der Browser-Erweiterungen. Links sind die Ergebnisse der ersten Modulversion dargestellt, rechts die Ergebnisse der zweiten Modulversion.

Datei	Erste Modulversion		Zweite Modulversion	
	Dateisystem	Carven	Dateisystem	Carven
Binance – Chrome – „manifest.json“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	✓
Binance – Chrome – „Background.html“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	✗
Binance – Firefox – „extensions.json“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	✓
Coinbase – Chrome – „manifest.json“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	✓
Coinbase – Chrome – „index.html“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	✓
MetaMask – Chrome – „manifest.json“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	✓
MetaMask – Chrome – „home.html“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	✓
MetaMask – Firefox – „extensions.json“	(wird nicht gesucht)	(wird nicht gecarvt)	✓	✓
MetaMask – Firefox - @metamask.io.xpi	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)

Android

Tabelle 13: Ergebnisse der Evaluation der Android Apps. Links sind die Ergebnisse der ersten Modulversion dargestellt, rechts die Ergebnisse der zweiten Modulversion.

Datei	Erste Modulversion		Zweite Modulversion	
	Dateisystem	**Carven**	**Dateisystem**	**Carven**
Electrum – Wallet-Datei (unverschlüsselt)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Electrum – Wallet-Datei (verschlüsselt)	✗	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Electrum – Electrum.dat	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Electrum – Config-Datei	✗	✗	✓	✓
Electrum – „recent_servers“	✗	(wird nicht gecarvt)	✗	(wird nicht gecarvt)
Electrum – Zertifikate	✗	✗	✓	✗
Exodus – Secure Container	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Exodus – „passphrase.json“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Exodus – Config-Datei	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)

Exodus – Exodus.exe	(nicht vorhanden)	(wird nicht gecarvt)	(nicht vorhanden)	(wird nicht gecarvt)
Exodus – „app.asar“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Exodus – RKStorage	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)
Guarda – „app.asar“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Guarda – „app-update.yml“	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)	(nicht vorhanden)
Guarda – guarda-backup.txt	(wird nicht gesucht)	(wird nicht gecarvt)	✓	(wird nicht gecarvt)

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.

[redacted]

_____	_____	_____
Ort, Datum	Vollständiger Name	Unterschrift

Nutzungs- und Verwertungsrechte

Ich übertrage zusätzliche Nutzungs- und Verwertungsrechte für die vorliegende Arbeit und allen damit in Zusammenhang stehenden Daten auf Grundlage *der Creative Commons Lizenz* "*CC0*" an alle genannten Betreuer dieser Arbeit.

[redacted]

_____	
Ort, Datum	Unterschrift