

# **Vergleich und Analyse des privaten Modus verschiedener Browser**

Studienarbeit im Modul

**Computer-Forensik und Vorfallsbehandlung**

**Prüfer**

Prof. Dr. Stefan Hahndel

**Bearbeiter**

Carl Schünemann (00107827) – BISE  
Christoph Sell (00108676) – MAPR

**Abgabedatum:** 20.06.2023

# Erklärung

Wir erklären hiermit ehrenwörtlich, dass wir die Arbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet haben.

Ingolstadt, den 20.06.2023

Ingolstadt, den 20.06.2023

.....  
Christoph Sell

.....  
Carl Schünemann

# Arbeitsteilung der Autoren

Verantwortung für die Analyseergebnisse:

**Mozilla Firefox:** Carl Schünemann  
**Tor-Browser:** Carl Schünemann  
**Google Chrome:** Christoph Sell  
**Brave:** Christoph Sell

Bearbeiter der Kapitel:

**Kapitel 1:** Carl Schünemann  
**Kapitel 2:** Carl Schünemann  
**Kapitel 3:** Christoph Sell  
**Kapitel 4:** Carl Schünemann  
**Kapitel 5:** Carl Schünemann: Abschnitte 5.1 (Firefox) und 5.2 (Tor),  
Christoph Sell: Abschnitte 5.3 (Chrome) und 5.4 (Brave)  
**Kapitel 6:** Carl Schünemann und Christoph Sell (gemeinsam erarbeitet)  
**Kapitel 7:** Christoph Sell  
**Kapitel 8:** Christoph Sell  
**Anhänge:** Carl Schünemann: Anhänge A (Yara-Regeln), B (Firefox) und C (Tor),  
Christoph Sell: Anhänge D (Chrome) und E (Brave)

Jeder Autor hat eigenständig und in eigener Verantwortung die Inhalte sowie die Ergebnisse seines jeweiligen Kapitels erarbeitet.

# Inhaltsverzeichnis

<b>Erklärung</b>	<b>II</b>
<b>Arbeitsteilung der Autoren</b>	<b>III</b>
<b>1. Einleitung</b>	<b>1</b>
<b>2. Ziel der Arbeit</b>	<b>2</b>
<b>3. Theoretischer Hintergrund</b>	<b>3</b>
3.1. Private Browsing . . . . .	3
3.2. Browser-Forensik und Browsing-Artefakte . . . . .	4
3.3. Angreifermodell . . . . .	4
<b>4. Methodik</b>	<b>6</b>
4.1. Vorbereitung . . . . .	6
4.1.1. Browserauswahl . . . . .	6
4.1.2. Browsing-Szenario . . . . .	8
4.2. Datensammlung . . . . .	12
4.3. Datenanalyse . . . . .	15
4.3.1. Common Locations . . . . .	15
4.3.2. Uncommon Locations . . . . .	17
4.3.3. Registry . . . . .	21
<b>5. Ergebnisse</b>	<b>23</b>
5.1. Firefox . . . . .	23
5.2. Tor-Browser . . . . .	30
5.3. Chrome . . . . .	36
5.4. Brave . . . . .	43
<b>6. Vergleich der Browser</b>	<b>48</b>
<b>7. Zusammenfassung und Diskussion</b>	<b>52</b>
<b>8. Fazit und Ausblick</b>	<b>53</b>
<b>Anhänge</b>	<b>54</b>
A. Yara-Regeln . . . . .	55

B.	Ausführliche Analyse: Firefox . . . . .	56
B.1.	Common Locations . . . . .	56
B.2.	Uncommon Locations . . . . .	61
B.3.	Registry . . . . .	63
C.	Ausführliche Analyse: Tor . . . . .	65
C.1.	Common Locations . . . . .	65
C.2.	Uncommon Locations . . . . .	69
C.3.	Registry . . . . .	70
D.	Ausführliche Analyse: Chrome . . . . .	72
D.1.	Common Locations . . . . .	72
D.2.	Uncommon Locations . . . . .	77
D.3.	Registry . . . . .	78
E.	Ausführliche Analyse: Brave . . . . .	80
E.1.	Common Locations . . . . .	80
E.2.	Uncommon Locations . . . . .	84
E.3.	Registry . . . . .	86
<b>Abbildungsverzeichnis</b>		<b>87</b>
<b>Tabellenverzeichnis</b>		<b>89</b>
<b>Literaturverzeichnis</b>		<b>90</b>
<b>Literatur</b>		<b>90</b>

# 1. Einleitung

Webbrowser speichern Informationen wie den Verlauf von besuchten Websites, Suchbegriffe, Passwörter, Cookies und andere Nutzeraktivitäten. Um die Privatsphäre von Benutzern zu schützen, wurde der sogenannte *private Modus* für Browser entwickelt. Bei den meisten Browsern ist dieser Schutz auf den lokalen Rechner beschränkt. [48] Um die Privatsphäre im gesamten Internet zu schützen, werden zusätzliche Maßnahmen, wie beispielsweise VPNs empfohlen. [45]

Es gibt unterschiedliche Nutzermodelle für private Browsing-Modi. Einerseits verwenden Privatpersonen diese Technologien, um ihre Privatsphäre zu schützen und ihren lokalen digitalen Fußabdruck zu regulieren [20]. Darüber hinaus nutzen Personen private Browsing-Modi, um persönliche Informationen vor Betrügern im Internet zu schützen oder spezifische Websites, wie beispielsweise Erwachsenen- oder Geschenk-Websites, diskret zu besuchen [1]. Auf der anderen Seite nutzen kriminelle Nutzer private Browsing-Modi, um Online-Straftaten zu verschleiern und digitale Beweise in kriminellen Fällen zu minimieren oder zu verhindern [33, 48]. Des Weiteren gibt es staatlich unterdrückte Nutzer, wie beispielsweise Journalisten in autokratischen Staaten, die private Browsing-Modi nutzen, um einer freien Pressearbeit ohne Repressionen nachzugehen [47]. Jedes dieser Nutzermodelle hat seine eigenen Motivationen und Gegenspieler.

Entwickler von privaten Browsing-Modi stehen deshalb vor einem Dilemma, da sie entscheiden müssen, wer zu welchem Grad geschützt werden soll. Beispielsweise strebt der Tor-Browser an, Menschenrechte und die Freiheiten des Individuums zu fördern. [59] Jedoch erschweren seine Funktionalitäten forensische Ermittlungen zu kriminellen Nutzern [38, 47].

Unabhängig davon, wer private Browsing-Modi nutzt, haben alle Stakeholder Interesse daran zu erfahren, ob und welche Spuren hinterlassen werden. In der Literatur werden stets neue Schwachstellen identifiziert, durch die private Browsing-Daten „lecken“ [51]. Im Rahmen dieser Seminararbeit werden die privaten Modi von vier Webbrowsern untersucht: Mozilla Firefox, Tor-Browser, Google Chrome und Brave [33]. Es wird analysiert, ob und welche Spuren von diesen Browsern in ihren privaten Modi auf den lokalen Rechnern hinterlassen werden.

## 2. Ziel der Arbeit

Die vorliegende Seminararbeit hat das Ziel, die Auswirkungen privater Browsingmodi auf potenziell hinterlassene Dateien einer Internetsitzung, die *Browsing-Artefakte*, auf dem lokalen Rechner zu untersuchen. Konkret werden die Browser Firefox, Tor-Browser, Chrome und Brave analysiert, um festzustellen, welche dieser Browser die geringsten Spuren nach einer privaten Browsing-Sitzung hinterlassen.

Zentral für diese Arbeit ist eine *transparente Versuchsdurchführung*. Dies umfasst sowohl die Kontaminierung als auch die Analyse der Browsing-Artefakte durch die gleichen Akteure. Dadurch ist bereits vor der Analyse bekannt, nach welchen spezifischen Browsing-Artefakten gesucht wird. Dies entspricht keinem realistischen forensischen Analyseszenario von Strafverfolgungsbehörden. Dort ist in der Regel nicht bekannt, welche Webseiten besucht wurden. Stattdessen wird meist nach verdächtigen Browsing-Artefakten gesucht. Die transparente Versuchsdurchführung zielt darauf ab, das Verhalten des privaten Browsing-Modus umfassend zu analysieren und dabei alle potenziellen Artefakte zu identifizieren. Dies verbessert die Effizienz zukünftiger Untersuchungen und verhindert, dass wichtige Inhalte übersehen werden. [20]

Das oberste Ziel dieser Arbeit besteht darin, gefundene Browsing-Artefakte eindeutig dem entsprechenden Browsing-Szenario oder Browser-Prozess zuzuordnen. Dies ist nötig da digitale Beweise bei Gerichtsverfahren eine *Beweisauthentifizierung* erfordern, wodurch der Beweis eindeutig einer Straftat zugeordnet werden muss.

Diese Arbeit grenzt sich von bestimmten Themengebieten ab, die nicht im Fokus der Untersuchung liegen. Diese Arbeit beschränkt sich auf den lokalen Angreifer, wie er in Abschnitt 3.3 definiert wird und betrachtet nicht den Webangreifer. Eine Zuordnung gefundener Artefakte zu bestimmten Zeitstempeln wird nicht berücksichtigt. Weiterhin werden keine Indikatoren untersucht, die anzeigen, ob und wann ein Browser gestartet, geschlossen oder im privaten Modus verwendet wurde. Schließlich werden nicht die Auswirkungen von Browser-Erweiterungen auf die privaten Modi untersucht.

## 3. Theoretischer Hintergrund

Nachfolgend werden die für das Verständnis dieser Arbeit relevanten Begriffe *Private Browsing*, *Angreifermodell* sowie *Private-Browsing-Artefakte* erläutert.

### 3.1. Private Browsing

Ein *Web Browser*, kurz *Browser*, ist eine Softwareanwendung zum Abrufen und Durchsuchen von Informationsquellen im Internet [48]. Izuati und Ab Rahman [23] bezeichnen ihn als eine Software, die es Benutzern ermöglicht, das Internet über den von ihrem Dienstanbieter bereitgestellten Zugang zu nutzen. Sie werden für alltägliche Aktivitäten wie das Anschauen von Videos, das Durchsuchen von Websites, das Posten von Bildern oder Videos in sozialen Medien und das Herunterladen von Dateien genutzt. [23] Beim *normalen* Browsen speichert der Browser alle entstehenden Dateien wie Cache, Cookies, sowie den Suchverlauf auf dem Computer [23]. Um das zu verhindern wurde der sogenannte *private Modus* bei Webbrowsern eingeführt. Diese Funktion ermöglicht das *Private Browsing*, was den Internetnutzern eine größere Kontrolle über ihre Privatsphäre gibt, ohne Rückstände von Datenspuren auf dem Computer zu hinterlassen [49]. Der private Modus wurde erstmals 2005 mit Apple Safari 2.0 eingeführt [49]. Drei Jahre später folgte in Google Chrome der *Incognito-Modus* sowie *InPrivate Browsing Modus* für den Internet Explorer. Im Jahr 2009 führte Mozilla Firefox 3.5 mit dem *Privaten Modus* seine Version des privaten Modus ein [33].

Das genaue Ziel des privaten Modus unterscheidet sich je nach Browser. Beispielsweise sollen bei Mozilla Firefox und Google Chrome besuchte Webseiten ausschließlich auf dem lokalen Computer des Benutzers keine Spuren hinterlassen. [15, 36] Manche Browser erweitern diesen Schutz, indem sie verhindern, dass beispielsweise Webseitenbetreiber auf Informationen des Private-Browsing-Nutzers zugreifen. [59]

Die Nutzer des privaten Modus lassen sich in unterschiedliche Interessensgruppen aufteilen. Forensische Ermittler versuchen über Rückstände durchgeführter Browsing-Sessions, Kriminelle mithilfe forensischer Tools und Techniken gezielt zu überführen. [33]. Kriminelle versuchen hingegen, ihre illegalen Aktivitäten mithilfe des privaten Modus zu verbergen [25]. Herkömmliche Nutzer verwenden den privaten Modus zum Schutz der Privatsphäre. Dieser stellt zusammen mit dem Löschen des Verlaufes die beliebteste Online-Datenschutzmaßnahme dar [20].



### 3.2. Browser-Forensik und Browsing-Artefakte

Die *digitale Forensik* konzentriert sich auf die Anwendung forensischer Techniken und Methoden zur Untersuchung von digitalen Geräten, Netzwerken und elektronischen Daten, um digitale Beweise für die strafrechtliche Verfolgung von kriminellen Onlineaktivitäten zu sammeln. Dabei gilt es, Beweise vollständig und in ihrem Originalzustand zu sichern, um vor Gericht zulässig zu sein. Dazu muss insbesondere der Prozess der Erwerbung, Untersuchung, Analyse sowie Berichterstattung digitaler Beweise forensisch einwandfrei durchgeführt werden. [23].

Die *Browser-Forensik* sammelt und identifiziert Beweise und Informationen im Zusammenhang mit einem Verbrechen aus wiederhergestellten Spuren von Browser-Sitzungen. Dabei werden nach Informationen über die durchgeführten Browsing-Aktivitäten gesucht, die sogenannten *Browsing-Artefakte*. Dies umfasst beispielsweise lokale Dateien, die Informationen wie den Suchverlauf, Cookies, Caches und andere sensiblen Daten enthält. Auch Webseitenbetreiber können Browsing-Artefakte speichern, indem Informationen über die Eigenschaften und das Verhalten der Webseitenbesucher gespeichert werden. Im Rahmen dieser Arbeit werden ausschließlich lokale Browsing-Artefakte untersucht. [23] Browsing-Artefakte, die sensible Informationen während einer Private-Browsing-Sitzung speichern, werden als *Private-Browsing-Artefakte*, kurz *PB-Artefakte* bezeichnet.

Aufgrund der *Beweisauthentifizierung* muss ein gefundenes Browsing-Artefakt eindeutig einer Browsing-Aktivität zugeordnet werden, um einen Verdächtigen aufgrund seiner Online-Aktivitäten zu überführen [25].

### 3.3. Angreifermodell

In der Browser-Forensik bezieht sich das *Angreifermodell* auf die spezifischen Annahmen und Eigenschaften eines potenziellen Angreifers, der auf Browsing-Artefakte abzielt. Aggrawal et al. [1] definierten zwei anzunehmende Angreifer in der Browser-Forensik.

Der sogenannte *Web Attacker* versucht Onlineaktivitäten des Benutzers außerhalb des lokalen Computers zu verfolgen und zu identifizieren. Zum Beispiel kann mittels Tracking-Tools oder durch das Sammeln von Informationen über die IP-Adressen versucht werden, einzelne Benutzer sowie deren Aktivitäten zu identifizieren und nachzuverfolgen. So kann der Internetdienstanbieter den Datenverkehr der Kunden verfolgen, um die Daten unter Zustimmung der Kunden für Marketingzwecke zu monetarisieren [1].

Im Gegensatz dazu hat der *Local Attacker* physikalischen Zugriff auf den Computer, von welchem aus das Browsing durchgeführt wurde. Dies kann ein forensischer Prüfer, ein Familienmitglied oder Freund sein, der beispielsweise versucht, auf den Browserverlauf zuzugreifen. Wie im Ziel dieser Arbeit definiert, wird für dieses forensische Experiment der Local Attacker als Angreifermodell angenommen. [1]

In einem realistischen Forensischen Szenario hätte der Local Attacker erst Zugriff auf den Computer, nachdem der Benutzer den privaten Modus verlässt, was beispielsweise die Aufzeichnung von Browsing-Aktivitäten ausschließt. [1] In der Literatur der Browsing-Forensik wird bei forensischen Versuchen zur Untersuchung des Browser-Verhaltens an dieser Stelle von der Definition des Local Attackers abgewichen. Wie im Ziel der Arbeit beschrieben, hat der Local Attacker im Kontext der *transparenten Versuchsdurchführung* vor, während und nach der Browsing-Sitzung vollständigen Zugriff auf den lokalen Computer. [8, 48]

## 4. Methodik

In der Browser-Forensik ist eine definierte Methodik notwendig, um die Komplexität moderner Browser zu bewältigen. Sie bildet die wissenschaftliche Basis für den durchgeführten Versuch sowie einen Leitfaden für Ermittler bei zukünftigen Untersuchungen. [1, 20, 23] Izzati et al. empfehlen als Vorgehensmodell für die Browser-Forensik das *Generic Model Computer Forensics Investigations*, kurz *GCFIM*. [61] In Ihrer Anwendung auf die Browser-Forensik besteht das Modell aus vier Phasen: [23]

- **Vorbereitung:** Versuchsplanung und Konfiguration der Versuchsumgebung.
- **Datensammlung:** Speicherabbilder identifizieren und während des Browsing-Szenarios erstellen.
- **Datenanalyse:** Suche nach Browsing-Artefakten in gesammelten Daten.
- **Dokumentation:** Vorgehensweise und gefundene Artefakte dokumentieren.

Die Dokumentationsphase entspricht in dieser Arbeit dem Kapitel 6, „Vergleich der Browser“. Die Methodik der anderen Phasen wird nachfolgend beschrieben.

### 4.1. Vorbereitung

In der Vorbereitungsphase wird der durchgeführte Versuch geplant sowie die Versuchsumgebung konfiguriert. [23] Die Versuchsplanung umfasst die Auswahl von Browsern und Tools sowie die Definition der durchzuführenden Schritte zur Kontaminierung des Rechners. Die Konfiguration der Versuchsumgebung umfasst die Installation und Konfiguration der notwendigen Software und Hardware.

#### 4.1.1. Browserauswahl

Diese Arbeit widmet sich den zwei weit verbreiteten<sup>1</sup> Browsern *Google Chrome* und *Mozilla Firefox*. Weiterhin werden zwei Browser mit verstärktem Schutz der Privatsphäre ausge-

---

<sup>1</sup>Laut Statista [54] (Stand 23. Mai 2023) sind Chrome (62,82%), Safari (20,86%), Edge (5,28%) und Firefox (2,77%) die weltweit meistgenutzten Browser. Für Safari sind nur ältere Versionen für Windows verfügbar. Microsoft Edge wurde in nur 3 von 23 untersuchten Papern untersucht, während sowohl Firefox als auch Chrome in 15 von 23 Papern analysiert wurden. [1, 8, 9, 20, 23, 25, 29, 33, 38, 39, 41, 42, 48, 49, 51]

wählt: *Brave*, basierend auf Chromium, sowie der *Tor-Browser*, eine modifizierte Version von Firefox.

### Mozilla Firefox

Der Browser *Mozilla Firefox*, kurz *Firefox*, ist ein open-source Webbrowser der gemeinnützigen Organisation Mozilla. Firefox hat die Funktion des *Privaten Modus*. Diese ermöglicht es, ohne Speicherung von Verlaufsdaten und Cookies im Internet zu browsen. Laut Firefox wird mit dem Privaten Modus vor dem lokalen Angreifer geschützt, wie er in Abschnitt 3.3 definiert ist, jedoch nicht vor dem Webangreifer. Es wird ausdrücklich darauf hingewiesen, dass die besuchten Webseiten und Internetanbieter (ISP) weiterhin anhand der IP-Adresse Informationen sammeln können. [34]

### Tor-Browser

Der *Tor Browser*, kurz *Tor* genannt, ist ein auf Firefox basierender Webbrowser, der das Tor-Netzwerk nutzt. Im Gegensatz zu Firefox wird zudem mit Schutzmaßnahmen gegen den Webangreifer geworben. Der Schutz vor dem Webangreifer ist durch das Tor-Netzwerk gegeben. Der Tor Browser wirbt mit verstärkten Schutzmaßnahmen gegen den lokalen Angreifer. [59]

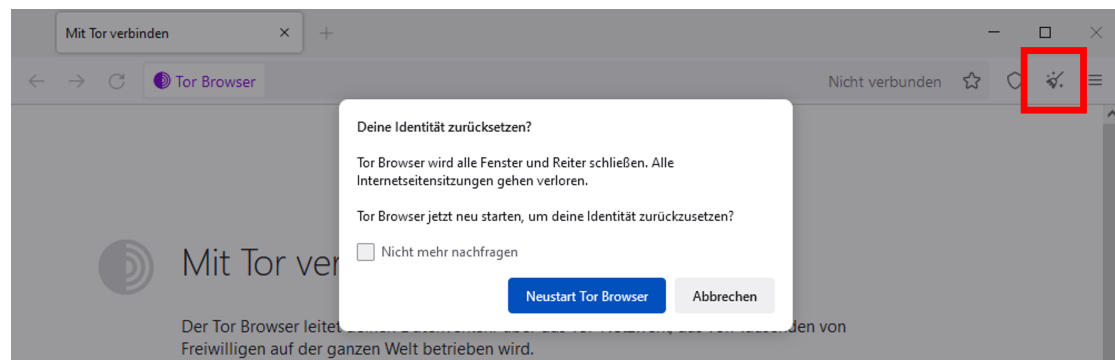


Abbildung 4.1.: Funktion „Neue Identität“ des Tor-Browsers

Die in Abbildung 4.1 gezeigte Funktion „*Neue Identität*“ ermöglicht es, alle aktuellen Tabs und Fenster zu schließen, sämtliche private Informationen wie Cookies und Verlauf zu löschen sowie die Verbindung mit dem Tor-Netzwerk neu zu konfigurieren. [59]

### Chrome

Der von Google entwickelte *Chrome* Browser bietet den *Inkognitomodus* als privaten Browsing-Modus an. Laut Google werden dadurch keine Verlaufsdaten, Cookies und Websitedaten oder in Formulare eingetragene Informationen gespeichert. Bezüglich des Schutzes vor Webangreifern

macht Google keine angaben. Chrome basiert auf dem Open-Source-Projekt *Chromium*, einem freien und quelloffenen Webbrowser, der von The Chromium Projects entwickelt wird. Dabei handelt es sich um eine von Google initiierte quelloffene Plattform für die Entwicklung von Webbrowsers. Der Chrome Browser selbst enthält proprietäre Elemente. [15]

### Brave

Der Open-Source-Browser *Brave* wurde von Brendan Eich, dem Mitbegründer von Mozilla, mit Fokus auf Privatsphäre, Sicherheit sowie Nutzerfreundlichkeit auf Basis des Chromium-Projekts entwickelt. Brave bietet Schutz vor dem lokalen Angreifer durch den *Private-Window*-Modus. In diesem Modus werden laut Herstellerangaben der Browserverlauf sowie Cookies nicht lokal auf dem Gerät gespeichert. Um vor dem Webangreifer zu schützen, bietet Brave mit dem *Private-Window-with-Tor-Connectivity*-Modus zusätzlich die Möglichkeit, eine Verbindung zum Tor-Netzwerk herzustellen. [4] Im Rahmen dieser Seminararbeit wird ausschließlich der Private-Window-Modus untersucht.

#### 4.1.2. Browsing-Szenario

Im Falle der transparenten Versuchsdurchführung der Browser-Forensik werden eine Reihe von Aktivitäten definiert, die für jeden untersuchten Browser durchgeführt werden – das sogenannte *Browsing-Szenario*. In diesem Protokoll wird definiert, mit welchen Daten der Rechner kontaminiert werden soll.

Für diesen Versuch wurden ausschließlich Daten definiert, die nicht bereits vor Durchführung des Browsing-Szenarios auf dem Rechner zu finden sind. Beispielsweise sind die Zeichenketten „twitter“ oder „facebook“ bereits in vielen Windows-Standardanwendungen enthalten.

Folgende Schritte werden für diesen Versuch mit jedem Browser durchgeführt:

1. [www.google.com](http://www.google.com) aufrufen
  - 1.1. Alle Cookies akzeptieren (wenn gefordert)
  - 1.2. Google-Suche nach „pfaffenhofen“
2. [www.google.com](http://www.google.com) aufrufen
  - 2.1. Cookies alle akzeptieren (wenn gefordert)
  - 2.2. Google-Suche nach „nanoradar“
3. [www.google.com](http://www.google.com) aufrufen
  - 3.1. Cookies alle akzeptieren (wenn gefordert)
  - 3.2. Google-Suche nach „mallofamerica“
  - 3.3. Auf Suchergebnis „mallofamerica.com“ klicken

4. [www.google.com](http://www.google.com) aufrufen
  - 4.1. Cookies alle akzeptieren (wenn gefordert)
  - 4.2. Google-Suche nach „mooserliesl“
  - 4.3. Auf Suchergebnis „mooserliesl.de“ klicken
5. „unitree.com“ über URL-Leiste öffnen
6. „donaukurier.de“ über URL-Leiste öffnen
  - 6.1. Donaukurier Logo in neuem Tab öffnen
7. „mail.google.com“ über URL-Leiste öffnen
  - 7.1. Mit google Account anmelden:
    - 7.1.1. E-Mail = „computerforensikvl@gmail.com“
    - 7.1.2. Passwort = „Vorlesung23!“
  - 7.2. Neue E-Mail schreiben:
    - 7.2.1. Empfänger: „cas0597@thi.de“ und „chs3702@thi.de“
    - 7.2.2. Betreff: „Betrefftext“
    - 7.2.3. Mailinhalt: „Mailinhalt“

Aus diesem Browsing-Szenario lassen sich die in Tabelle 4.1 dargestellten *Private-Browsing-Artefakte*, kurz *PB-Artefakte* ableiten. Dabei handelt es sich um Zeichenketten, die eindeutig einem Schritt im Browsing-Szenario zugeordnet werden können. Diese sind von zentraler Bedeutung in der Analysephase: Nur nach diesen Strings wird gesucht.

### Konfiguration virtueller Maschinen

Eine empfohlene Herangehensweise bei Versuchen in der Browser Forensik ist die Versuchsdurchführung in einer virtualisierten Umgebung. Dieser ermöglicht eine Reproduzierbarkeit und Transportierbarkeit der Ergebnisse. Pro Browser existiert eine virtuelle Maschine, kurz *VM*, auf der das Browsing-Szenario durchgeführt wird. Somit werden die Versuchsumgebungen der einzelnen Browser voneinander sowie von der Analyseumgebung getrennt. [38] Als Virtualisierungssoftware wird für diesen Versuch die kostenlose Software *Oracle VM VirtualBox* verwendet.

Alle VMs besitzen die gleiche, in Tabelle 4.2 dargestellte Basiskonfiguration. Zum Datenaustausch zwischen VM und Analysecomputer wird ein *gemeinsamer Ordner* eingerichtet. Somit werden beispielsweise ohne Kontaminierung der VM benötigte Programme auf dem Analysecomputer heruntergeladen, in den gemeinsamen Ordner gelegt und offline auf der VM installiert. Auf der VM werden zwei Werkzeuge der Sysinternal-Abteilung von Microsoft

Tabelle 4.1.: Private-Browsing-Artefakte des Browsing-Szenarios

Kategorie	Private-Browsing-Artefakt	Schritt im Browsing-Szenario
Suchbegriff	„pfaffenhofen“	1.2
	„nanoradar“	2.2
	„mallofamerica“	3.2
	„mooserliesl“	4.2
URL	„mooserliesl.de“	3.3
	„mallofamerica.com“	4.3
	„unitree.com“	5.
	„donaukurier.de“	6.
Bild	0x89 0x50 0x4E 0x47 ... (PNG als Hexadezimalwerte)	6.1
E-Mail	„computerforensikvl@gmail.com“	7.1.1
	„Vorlesung23! “	7.1.2
	„cas0597@thi.de“	7.2.1
	„chs3702@thi.de“	7.2.1

installiert, um in der Analysephase das Browserverhalten vollständig untersuchen zu können: *Process Monitor* ermöglicht die Aufzeichnung aller Prozessaktivitäten und *Process Explorer* erweitert die Funktionen des Windows Task Managers. [27, 28] Nachdem eine VM mit der

Tabelle 4.2.: Basiskonfiguration jeder VM des Versuchs

<b>Betriebssystem</b>	Windows 10 Pro, 64 Bit, Build: 19045.2006
<b>Festplatte</b>	30 GB, VDI-Format, kein SSD Laufwerk
<b>RAM</b>	6 GB
<b>Netzwerk</b>	Netzwerkbrücke
<b>Verbindung zu Host-PC</b>	Gemeinsamer Ordner
<b>Installierte Programme</b>	Process Monitor (Version 3.93) Process Explorer (Version 17.04)

Standardkonfiguration erstellt wurde, wird diese für jeden Browser dupliziert. Anschließend werden die Browser über den gemeinsamen Ordner in der entsprechenden VM installiert. Dazu wurden folgende Installationsverzeichnis verwendet:

1. **Firefox:** C:\Program Files\Mozilla Firefox\firefox.exe
2. **Tor:** C:\Program Files\Tor Browser\Browser\firefox.exe
3. **Chrome:** C:\Program Files\BraveSoftware\Brave-Browser\Application\brave.exe
4. **Brave:** C:\Program Files\Google\Chrome\Application\chrome.exe

### Verwendete Software

Neben der VM Konfiguration muss die Analyseumgebung vorbereitet werden. Als Analyseumgebung dient für diesen Versuch der Rechner, auf dem die VM läuft. (Windows 10 Home, 64 Bit, Build 19045.2965) Zur Analyse der Browser werden diverse Tools benötigt.

### Autopsy

Um erstellte Festplattenabbilder zu untersuchen wird das Tool *Autopsy* verwendet. Dabei handelt es sich um ein open-source Tool, das auf der Sleuthkit-Bibliothek für die forensische Analyse von Dateisystemen basiert, diese mit zusätzlichen Funktionen erweitert und eine grafische Benutzeroberfläche für die forensische Analyse bietet. [2]

### Volatility

Um Abbilder des Arbeitsspeichers zu untersuchen wird das open-source Framework *Volatility* verwendet, das speziell darauf ausgerichtet ist, Informationen und Artefakte aus dem physischen oder virtuellen Arbeitsspeicher eines Computers zu extrahieren. Für diesen Versuch wird *Volatility3* verwendet, eine 2020 veröffentlichte vollständige Neuschreibung des Volatility Frameworks. Volatility basiert auf Plugins, welche spezifische Funktionen und Analysen für verschiedene Aspekte des Systems bereitstellen. [12] Für diesen Versuch werden folgende Plugins verwendet:

- pslist
- yarascan
- memmap
- filesan
- svcscan

Die genaue Beschreibung der Plugins sowie deren Zusammenhang ist in der Analysephase in Abschnitt 4.3.2 beschrieben.

### Sonstige Tools

Tabelle 4.3 listet zusammenfassend alle in diesem Versuch verwendeten Software-Programme, deren Verwendungszweck sowie Version auf. Darunter befinden sich diverse zusätzliche unterstützende Tools, welche zur vollständigen Analyse benötigt werden.



Tabelle 4.3.: Vollständige Liste der verwendeten Software dieses Versuchs

Software	Verwendungszweck	Version
Mozilla Firefox	Webbrowser	114.0.1 (64-bit)
Tor-Browser	Webbrowser	12.0.4 (64-bit)
Google Chrome	Webbrowser	112.0.5615.138 (64-bit)
Brave	Webbrowser	1.50.121 (64-bit)
Windows 10 Pro	VM Betriebssystem	Build: 19045.2006
Process Monitor	Aufzeichnung Prozessaktivitäten	3.93
Process Explorer	Darstellung der Eigenschaften aktueller Prozesse	17.04
Autopsy	Analyse Festplattenabbilder	4.20.0
Volatility	Analyse RAM-Abbilder	Volatility3 Version 2.4.1
HxD	Analyse Binärdateien in hexadezimaler und ASCII-Darstellung	2.5.0.0
Notepad++	Analyse strukturierter Dateiformate, z.B. JSON, XML	8.4.5
Registry Explorer	Grafische Oberfläche zur Untersuchung von Windows-Registry Hives	2.0.0.0
DB Browser for SQLite	Grafische Oberfläche zur Verwaltung und Untersuchung von SQLite-Datenbanken	3.12.2
sqldiff.exe	Befehlszeilen-Programm zur Anzeige von Unterschieden zwischen SQLite-Datenbanken	3.42.0
ChromeCacheView	Einlesen von Chrome Cache-Dateien und visuelle Aufbereitung des Inhalts	2.46
MZCacheView	Einlesen von Firefox Cache-Dateien und visuelle Aufbereitung des Inhalts	2.21
FirefoxCache2	Erweitert MZCacheView, um Firefox „index“-Cachedatei zu analysieren	Commit b50ab4f
dejsonlz4	Dekomprimierung von .jsonlz4-Dateien	Commit c4305b8

## 4.2. Datensammlung

In der Phase der Datensammlung werden alle potenziellen Beweismittel identifiziert und in einem forensisch analysierbaren Format gesichert [23]. Im Rahmen dieser Arbeit umfasst dies die Durchführung des Browsing-Szenarios sowie das Sammeln potenzieller privater Browsing-Artefakte.

### Process Monitor Logfiles

Um das Verhalten von privaten Browsingmodi möglichst vollständig zu untersuchen, schlagen Fayyad-Kazan et al. [8] vor, alle Aktivitäten des Browsers während Browsing-Szenarios aufzuzeichnen. Dazu werden mit dem Tool Process Monitor alle Prozess-Aktivitäten zwischen zwei Zeitpunkten als *Process Monitor Logfile* (PML) oder CSV-Datei gespeichert. [8, 48] Die PML-Dateien werden mithilfe des gemeinsamen Ordners auf den Analysecomputer transportiert.

### Speicherabbilder

Eine der Hauptaufgaben eines Computer-Forensischen-Ermittlers ist die Erstellung und Analyse von direkten Kopien der Speichermedien des untersuchten Rechners. [19] Im Falle der Browser-Forensik werden Abbilder der Festplatten und des Arbeitsspeichers erstellt und analysiert.

**Festplatten-Image** Da in diesem Versuch die Festplatten virtualisiert werden, wird ein Abbild aus einem sogenannten *VM-Snapshot* gewonnen, eine Momentaufnahme der virtuellen Maschine. [43] VM-Snapshots können *aufgetaut* werden, wodurch der Zustand des

Betriebssystems zum Zeitpunkt der Momentaufnahme wiederhergestellt wird. Bei Oracle VM VirtualBox kann ein VM Snapshot über die grafische Oberfläche erstellt werden. Durch den Snapshot wird ein *Virtual Disk Image*, eine VDI-Datei, im Snapshot-Ordner der VM erzeugt. Diese Laufwerksdatei enthält nur differentielle Daten zum vorherigen Snapshot. Um aus den differentiellen Daten ein vollständiges Festplatten-Image zu erzeugen, muss ein *vollständiger Klon* des Snapshots erstellt werden. Die VDI-Datei der geklonten VM entspricht einem vollständigem Abbild der Festplatte zum Zeitpunkt des durchgeführten Snapshots.

Da Autopsy nicht das VDI-Format unterstützt, müssen die Laufwerksdateien der geklonten Snapshots in das generische *Image*-Format (.img) umgewandelt werden. Durch Nutzung des VirtualBox Befehlszeilen-Tool *vboxmanage* wird mit dem Befehl `vboxmanage clonehd <VDI_File>.vdi <IMG_File>.img -format raw` die VDI-Datei in eine IMG-Datei umgewandelt. Um ein Festplatten-Image in Autopsy einzulesen, wird ein neuer *Fall* (engl. Case) erstellt. Das Einlesen eines ca. 30 GB großen Festplatten-Images dauerte mit allen aktivierten Autopsy-Plugins zwischen 5 und 7 Stunden.

**RAM-Dump** Ein *RAM-Dump* erfasst den Zustand des Arbeitsspeichers, einschließlich der im Speicher befindlichen Daten, Programme und Prozesse zu einem bestimmten Zeitpunkt [57]. VirtualBox empfiehlt, Abbilder des RAMs ebenfalls über das *vboxmanage* Befehlszeilen-Tool durchzuführen. Im Unterschied zu Festplatten-Images können RAM-Dumps nur im angeschalteten Zustand der virtuellen Maschine mithilfe des Befehls `vboxmanage debugvm <VM Name> dumpvmcore -filename <RAM Dump Dateiname>.elf` durchgeführt werden. RAM-Dumps im .elf Format können direkt vom Analysetool Volatility innerhalb weniger Minuten eingelesen werden.

**Zeitpunkte zur Datensammlung** Wichtig für die Qualität der Versuchsergebnisse sind die Zeitpunkte während des Browsing-Szenarios zum Sammeln der Daten. In der Literatur wählen die Autoren meist ohne Begründung Zeitpunkte zur Datensammlung [1, 33, 39, 49–51]. Dieses Problem haben Muir, Leimich und Buchanan erkannt und spezifische Zeitpunkte zur Datensammlung vorgeschlagen. Diese ermöglichen eine vollständige Analyse des Browserverhaltens vor, während und nach dem Browsing-Szenario. [38]. Wie in Abbildung 4.2 dargestellt, wurde sich an diesen Zeitpunkten für diesen Versuch orientiert.

Der erste RAM-Dump sowie der erste VM-Snapshot nach der Browser-Installation, vor Beginn des Browsing-Szenarios, dienen als Baseline für die Analyse, da in diesen Speicherabbildern kein PB-Artefakt gefunden werden darf. Nachdem der private Modus im Browser geöffnet wird und bevor das Browsing-Szenario beginnt, wird die Aufnahme des ersten Process Monitor Logfiles gestartet. Um ausschließlich Schreiboperationen aufzuzeichnen, die auf das private Browsing zurückzuführen sind, wird die Aufzeichnung erst nach dem erstmaligen Öffnen des Browsers im privaten Modus gestartet. Nach Durchführung des Browsing-Szenarios, während der Browser noch geöffnet ist, wird die Aufnahme des ersten Process Monitor Logfiles gestoppt. Weiterhin wird ein zweiter RAM-Dump sowie VM-Snapshot erstellt. Anschließend wird eine zweite Process Monitor Aufzeichnung gestartet. Nachdem der Browser geschlossen

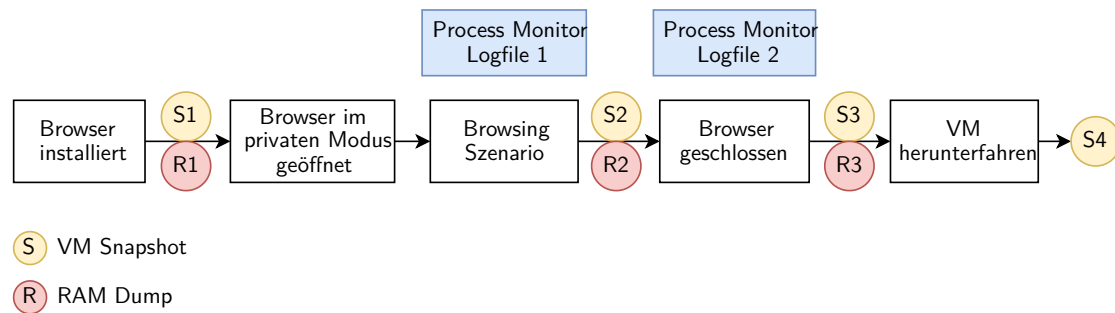


Abbildung 4.2.: Zeitpunkte zur Datensammlung während der Versuchsdurchführung nach [38]

wurde, wird die Aufzeichnung des zweiten Process Monitor Logfiles beendet. Somit enthält das zweite Logfile alle Prozessaktivitäten vom Schließen der Browser. Zusätzlich wird ein dritter RAM-Dump sowie VM-Snapshot erstellt. Nach Herunterfahren der VM wird ein vierter VM-Snapshot erstellt.

**Sonderfälle** Dieses Vorgehen zur Datensammlung wird bei allen Browsern durchgeführt. Einzig der Tor-Browser weicht davon ab. Um die „Neue Identität“-Funktion des Tor-Browsers zu berücksichtigen, werden zusätzlich Daten vor und nach der Erstellung einer „Neuen Identität“ gesammelt. Wie in Abbildung 4.3 dargestellt, umfasst dies einen zusätzlichen RAM-Dump sowie VM-Snapshot und ein weiteres Process Monitor Logfile.

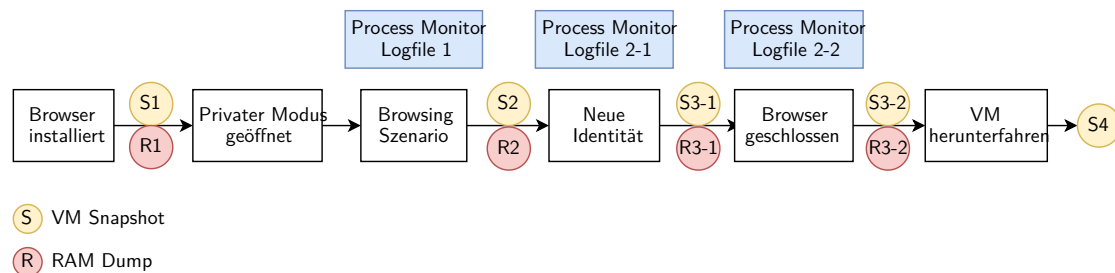


Abbildung 4.3.: Zeitpunkte zur Datensammlung während der Versuchsdurchführung für den Tor-Browser

Bei Durchführung des Browsing-Szenarios für den Firefox-Browser wurde nach erstmaligem Öffnen des Browsers automatisch die Firefox Datenschutz-Webseite <https://www.mozilla.org/de/privacy/firefox/> im nicht-privaten Modus durch den Browser geöffnet.

Sowohl Google Chrome als auch der Brave Browser öffneten sich automatisch nach der abgeschlossenen Installation im nicht-privaten Modus.

### 4.3. Datenanalyse

Nachdem die Daten in Form von Process Monitor Logfiles und Festplatten- sowie RAM-Speicherabbildern gesammelt wurden, wird nach den PB-Artefakten aus Tabelle 4.1 in Abschnitt 4.1.2 gesucht. Die gesammelten Daten des Versuchs werden zur Vereinfachung der Analyse in drei Kategorien aufgeteilt: *Common Locations*, *Uncommon Locations* sowie *Registry*.

#### 4.3.1. Common Locations

Die sogenannten *Common Locations* beziehen sich im Zusammenhang der Browser-Forensik auf die standardmäßigen Verzeichnisse eines Browsers auf der Festplatte, beispielsweise Ordner von Browsern zur Verwaltung von Nutzerdaten. Untersucht werden Common Locations mittels *Whitebox-Analyse*. Dabei besitzt der Forensiker Fachwissen über das Browserverhalten. Anhand dieses Wissens können gezielt potenzielle Beweise gesammelt werden. [3]

Bei diesem Versuch werden die Speicherorte über die Schreiboperationen der Process Monitor Logfiles identifiziert. Anschließend wird für jede Datei in den Speicherorten geprüft, ob PB-Artefakte enthalten sind. Dazu sind zwei Schritte notwendig:

1. **Dateiextraktion:** Extraktion der Datei aus dem Speicherabbild. Wenn die Datei nicht mehr vorhanden ist, werden dazu ggf. Tools zur Dateiwiederherstellung benötigt.
2. **Dateianalyse:** Um zu überprüfen ob die Datei PB-Artefakte enthält, werden ggf. Tools für spezielle Dateiformate benötigt, beispielsweise Dekomprimierungstools.

**Identifikation der Common Locations** Um die gängigen Browserpfade und -dateien zu identifizieren, werden die in den Process Monitor Logfiles aufgezeichneten Datei-Schreibaktivitäten der Browserprozesse ausgewertet.

Dazu wird jedes Logfile mit dem Process Monitor eingelesen. Anschließend werden die Aktivitäten gefiltert. Wie in Abbildung 4.4 dargestellt, wird dazu ausschließlich die Option „File System Activity“ ausgewählt.

Anschließend wird als Prozessname der Browserprozess gesetzt:

- **Firefox:** firefox.exe
- **Tor-Browser:** firefox.exe und tor.exe
- **Chrome:** chrome.exe
- **Brave:** brave.exe

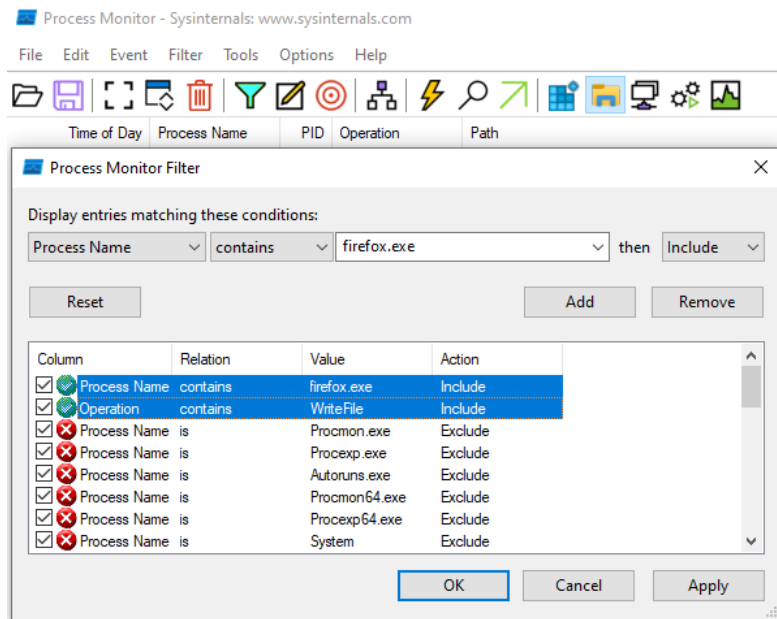


Abbildung 4.4.: Process Monitor Filter für Datei-Schreiboperationen

Da PB-Artefakte nur über Schreiboperationen entstehen können, wird als Prozessoperation „WriteFile“ gesetzt. Die gefilterte Logfile wird als CSV exportiert, um sie dann in Excel zu öffnen und irrelevante Spalten sowie Duplikate zu löschen. Die geschriebenen Dateien werden anschließend browserspezifisch gruppiert.

**Prüfung auf PB-Artefakte** Nachdem die geschriebenen Browserdateien identifiziert und gruppiert wurden, wird für jede Datei geprüft, ob PB-Artefakte enthalten sind. Folgende, in Abbildung 4.5 dargestellte Schritte sind zur Dateixtraktion und Dateianalyse notwendig: Die Datei befindet sich entweder im entsprechenden Festplatten-Image oder ist im RAM-Dump. Wenn die Datei nicht mit Autopsy aus dem Festplatten-Image extrahiert werden kann und sich der Dateiname in der Ausgabe des Volatility Plugins *filesfan* (`vol.py -f ram_dump.elf windows.filesfan > filesfan.txt`) befindet, wird diese mit dem Volatility Plugin *dumpfiles* (`vol.py -f ram_dump.elf -o \some_folder\ windows.dumpfiles -virtaddr <virtual file address>`) aus dem RAM extrahiert. Wenn auch dies nicht möglich ist und es sich um eine temporäre Datei (.tmp) handelt, wird versucht die entsprechende nicht-temporäre Datei zu extrahieren. Im Falle der Datei „some-file.json.tmp“ wird beispielsweise geprüft, ob die Datei „some-file.json“ existiert. Nachdem die Datei extrahiert wurde und ggf. mit einem Tool zu Analyse vorbereitet wurde, wird geprüft, ob die Datei PB-Artefakte enthält.

**SQLite-Datenbanken** Eine besondere Rolle unter den Common Locations bei Browsern nehmen SQLite-Datenbanken ein. Sie speichern und verwalten Nutzerinformationen, wie

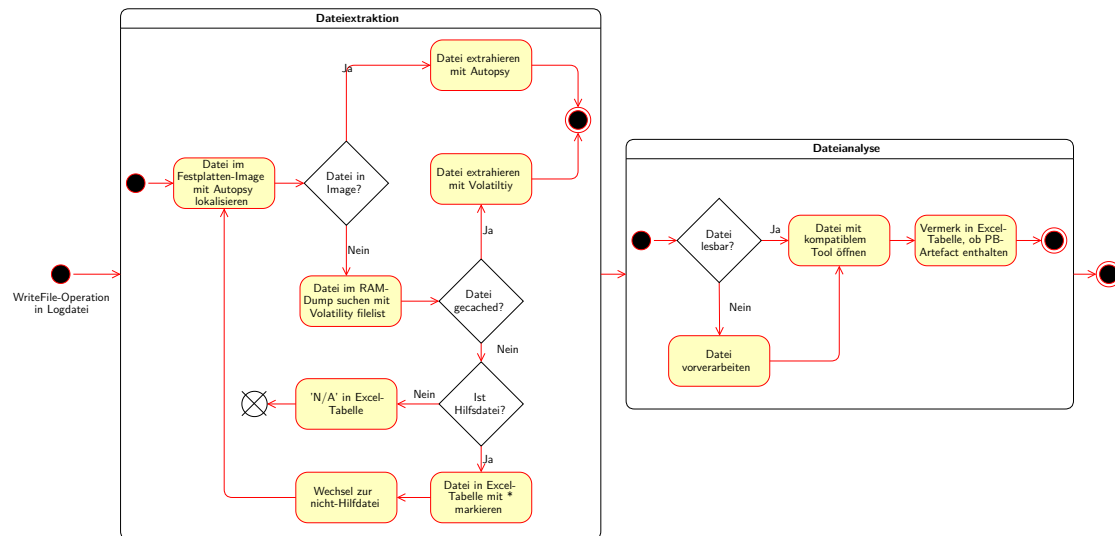


Abbildung 4.5.: Vorgehen zur Dateiextraktion und -analyse

Lesezeichen, Browserverlauf, Caches, Cookies in Datenbankdateien, ohne einen separaten Datenbankserver zu benötigen.

Wie in Abbildung 4.6 dargestellt, erfolgt die Dateiextraktion analog zur Vorgehensweise bei den Datei-Schreiboperationen der Process Monitor Logfiles. Um die SQLite-Datenbanken zu analysieren wird jede Datenbank mit der gleichen Datenbank aus dem vorherigen Snapshot mithilfe des Befehlszeilentools *sqldiff.exe* (*sqldiff.exe database1.sqlite database2.sqlite*) verglichen. Die Inhaltsunterschiede werden für jede SQLite-Datei in jedem Snapshot untersucht und in einer Excel Tabelle festgehalten. Datenbankänderungen einer SQLite-Datei werden zuerst im *Write-Ahead Log*, kurz *WAL*, vorübergehend protokolliert. Um potenzielle PB-Artefakte zu berücksichtigen, wird der WAL mithilfe der *sqlite3* Befehlszeile (*sqlite3> PRAGMA wal\_checkpoint;*) in die SQLite-Hauptdatenbank geschrieben.

### 4.3.2. Uncommon Locations

*Uncommon Locations* beziehen sich auf Verzeichnisse, die nicht zu den gängigen Speicherorten gehören. Bei Festplatten-Images handelt es sich dabei meist um Dateien des Betriebssystems<sup>2</sup> oder andere Festplattenbereiche, wie beispielsweise unallokierte Speicherbereiche oder der Arbeitsspeicher. Uncommon Locations werden ohne Vorwissen über das Browserverhalten

<sup>2</sup>In der betrachteten Literatur wird die sogenannte *pagefile.sys* als Teil der Uncommon Locations unter Windows untersucht. Diese Datei wird verwendet, um Daten aus dem Hauptspeicher auf die Festplatte auszulagern, wenn der verfügbare physische Speicher begrenzt ist. [30] In der betrachteten Literatur wurde nur die Stringsuche im Hexadezimaeditor als Analysemethode identifiziert. Wie in diesem Abschnitt beschrieben, ist dies aufgrund der fehlenden Zuordnung zu einem spezifischen Browser unzureichend. Um eine Beweisauthentifizierung gemäß dem Ziel der Arbeit zu garantieren, wird der Inhalt der *pagefile.sys* nicht untersucht. Gleiches gilt für die Speicherverwaltungsdatei *hiberfile.sys*

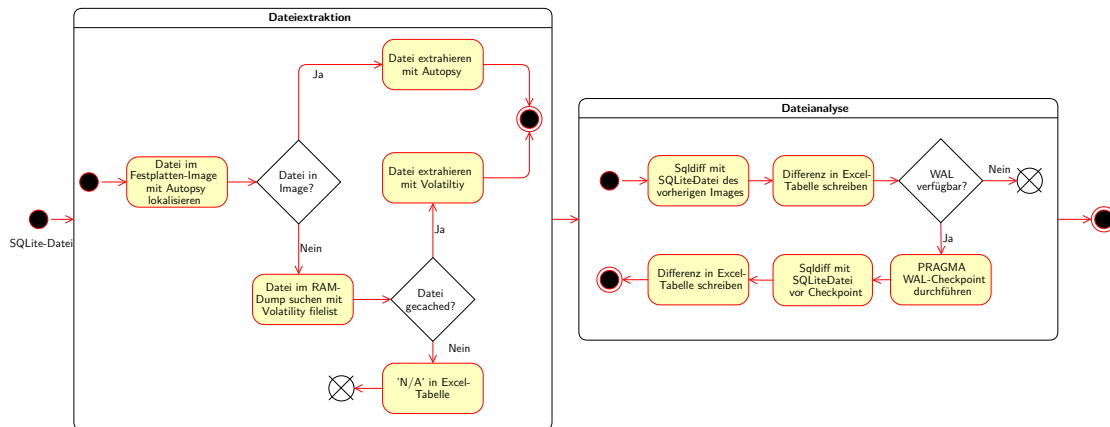


Abbildung 4.6.: Vorgehen zur Dateiextraktion und -analyse von SQLite-Datenbanken

sowie ohne Vorverarbeitung der Dateien mithilfe der *Blackbox-Analyse* untersucht: Im Kontext der Browser Forensik werden dazu Stringsuchen nach PB-Artefakten über die gesamten Speicherabbilder durchgeführt. Dies ist nur mit Unterstützung durch Forensik-Tools möglich. [3] Somit wird bei der Analyse der Uncommon Locations auf die Vollständigkeit der Tools vertraut.

### Analyse mit Autopsy

Bei den Uncommon Locations wird Autopsy als forensisches Werkzeug zur Analyse der Festplatten-Images verwendet. Dazu wird eine Stichwortsuche mit den in Tabelle 4.1 definierten PB-Artefakten über das gesamte eingelesene Festplatten-Image durchgeführt. Autopsy bietet dazu die in Abbildung 4.7 dargestellte Funktion zur Suche nach Strings, Teilstrings oder regulären Ausdrücken in Dateinamen und Dateiinhalten an.

Zusätzlich kategorisiert Autopsy automatisch die Dateien eines Festplatten-Images. Für diesen Versuch sind folgende Dateikategorien von Interesse:

- Web Bookmarks
- Web Cookies
- Web History
- Web Categories

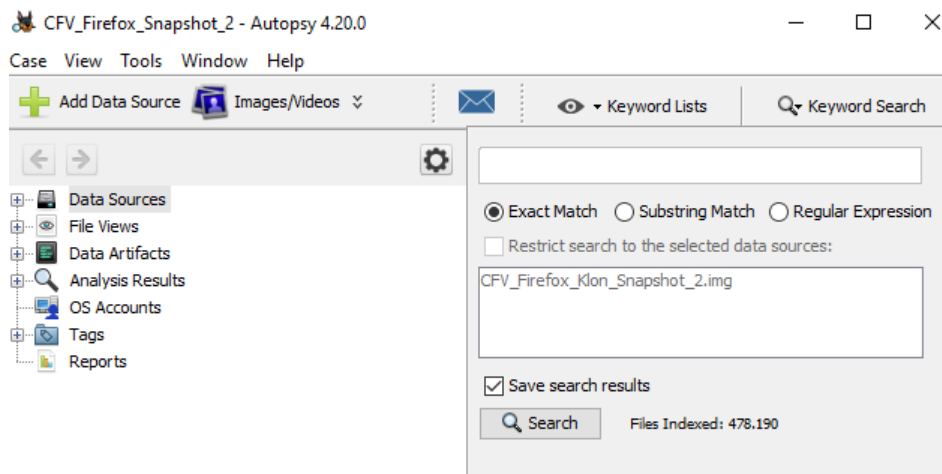


Abbildung 4.7.: Autopsy Funktion zur Stichwortsuche

### Analyse mit Volatility

Bei der Analyse des Arbeitsspeichers als Uncommon Location ist es von entscheidender Bedeutung, dass ein gefundener String eindeutig einem Browserprozess zugeordnet werden kann.

In der Literatur werden Arbeitsspeicherabbilder oft unzureichend durch eine Stichwortsuche im Hexadezimaeditor analysiert. [29, 33, 48] Dies kann aber nach Auffassung der Autoren dieser Arbeit zu Fehlschlüssen führen. Beispielhaft wird dies in Abbildung 4.8 gezeigt: Ein String, der in einer Textdatei auf dem Desktop gespeichert ist, wird im Hexadezimaeditor HxD angezeigt, obwohl kein Browsing-Szenario durchgeführt wurde.

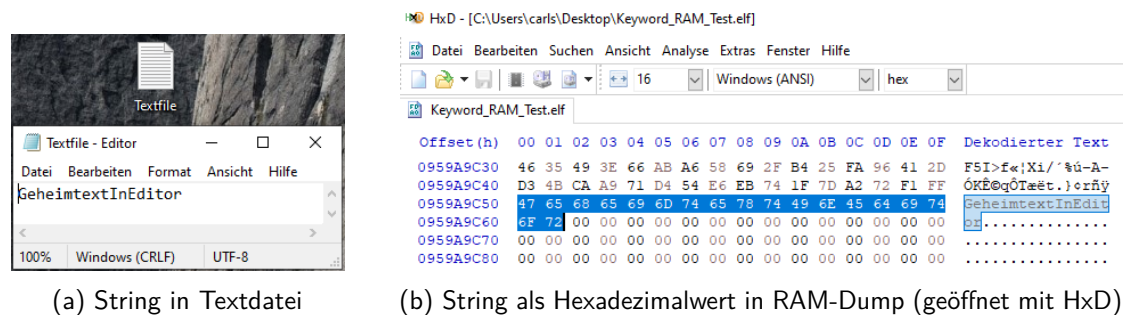


Abbildung 4.8.: Beispiel für ein RAM-Artefakt ohne vorheriges Browsing-Szenario

Um einen im RAM gefundenen String einem Browserprozess zuordnen zu können, wird deshalb das forensische Analysetool Volatility mit dem Plugin *Yarascan* verwendet. Mithilfe sogenannter *Yara-Regeln* (engl. Yararules) wird nach bestimmten Mustern im Arbeitsspeicher gesucht. Die für diesen Versuch verwendeten Yara-Regeln entsprechen den Strings der PB-Artefakte



in Tabelle 4.1. Zusätzlich sucht eine Regel nach HTML-Fragmenten, die eindeutig einer besuchten Seite des Browsing-Szenarios zuzuordnen sind. [49] Alle verwendeten Yara-Regeln sind im Anhang A aufgelistet. Um den RAM-Dump nach den Yara-Regeln zu durchsuchen, wird folgender Befehl ausgeführt: `vol.py -f ram_dump.img windows.vadyarascan -yara-file yara_rules.yara > yarascan.txt` Nachdem der RAM-Dump nach den Regeln durchsucht wurde, gibt die Yarascan-Ausgabe für jeden gefundenen String die PID des Prozesses, in dem der String gefunden wurde, sowie die virtuelle Speicheradresse des gefundenen Strings an.

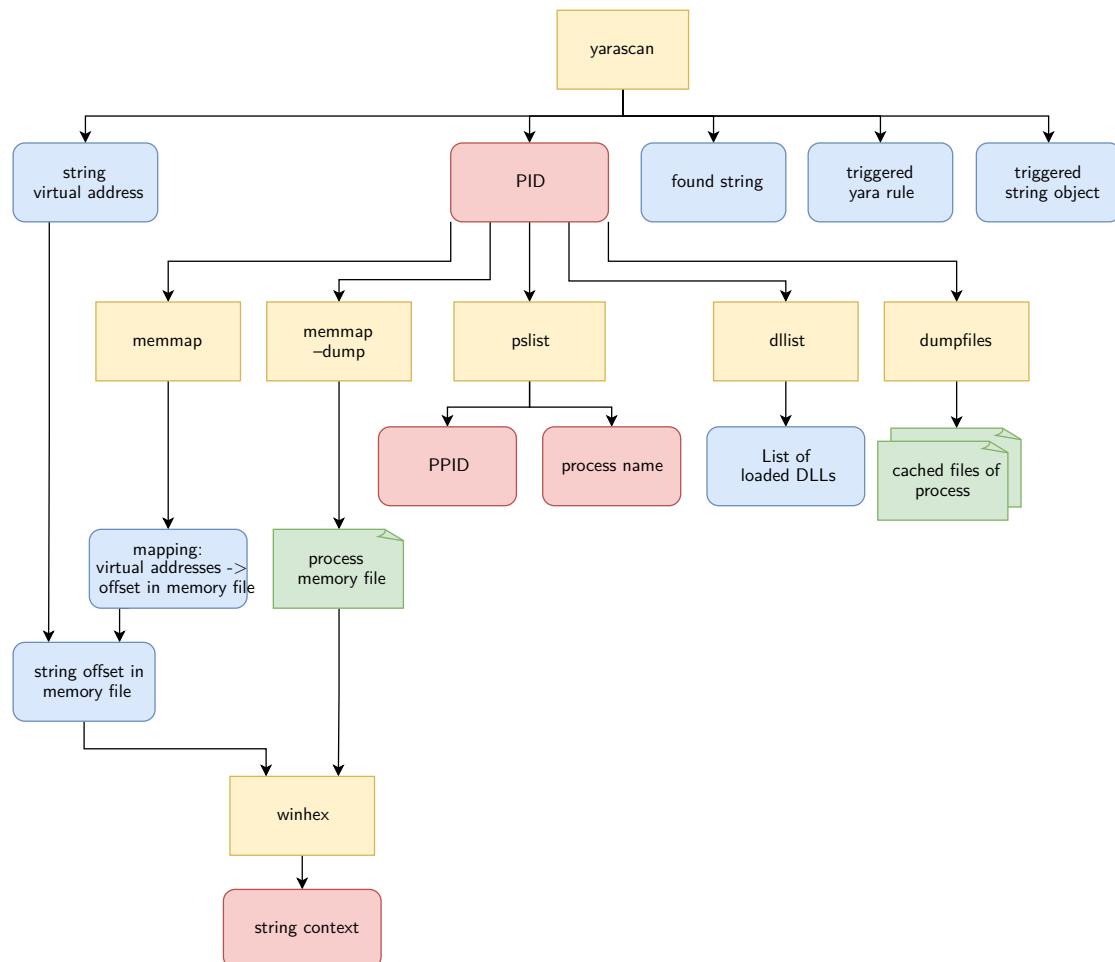


Abbildung 4.9.: Abhängigkeiten der verwendeten Volatility-Plugins yarascan, pslist und memmap

Wie in Abbildung 4.9 dargestellt, wird davon ausgehend mit dem Plugin *pslist* (`vol.py -f ram_dump.img windows.pslist -pid <PID> > pslist.txt`) der Prozessname der PID ermittelt, in dem der String gefunden wurde.

Oft ist bei einem gefundenen String von Interesse, ob in den Speicheradressen vor und nach

dem Treffer weitere Zusammenhänge erkennbar sind. Mithilfe des Plugins *memmap* (`vol.py -f ram_dump.img windows.memmap -pid <PID> > memmap.txt`) wird die Abbildung der virtuellen Speicheradressen eines Prozesses auf die Byte-Offsets der extrahierten Speicherseite des Prozesses ermittelt. Diese Seite kann mithilfe des „dump“-Flags extrahiert werden: `vol.py -f ram_dump.img -o \dump_dir\ windows.memmap -pid <PID> --dump`. In einem Hexadezimaeditor, wie HxD, kann die Umgebung des String-Treffers anhand des ermittelten Byte-Offsets in der Speicherseite untersucht werden.

### 4.3.3. Registry

Die letzte Kategorie analysierter Daten umfasst die Artefakte der Registry. Diese zählen sowohl zu den Common als auch Uncommon Locations und werden deshalb als eigene Kategorie aufgeführt.

**Common Locations** Als Teil der Common Locations werden die Registry-Aktivitäten in den Process Monitor Logfiles analysiert. Wie in Abbildung 4.10 gezeigt, wird zunächst das

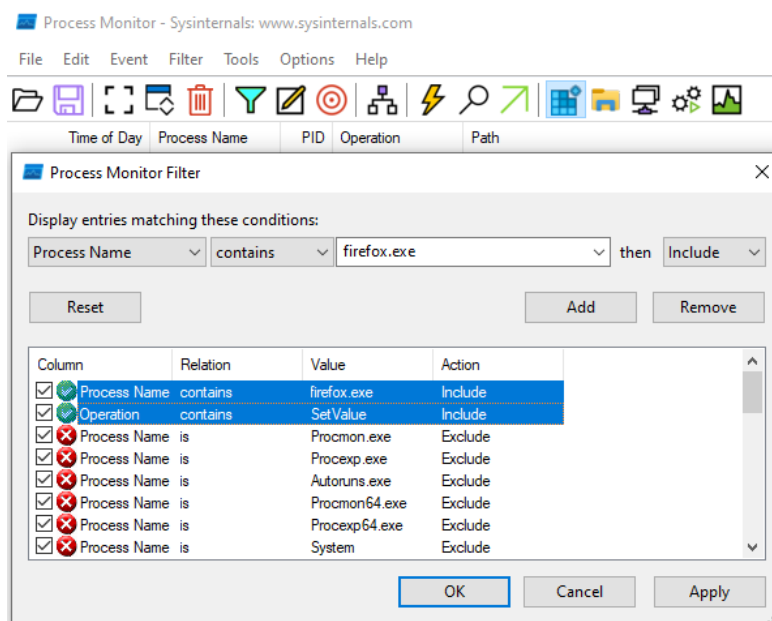


Abbildung 4.10.: Process Monitor Filter für Registry-Schreiboperationen

Logfile nach „Registry Activity“ sowie Einträgen mit der Operation „SetValue“ sowie dem Browser-Prozessnamen gefiltert. Als CSV-Datei wird das Logfile in Excel weiter verarbeitet, indem Duplikate gelöscht werden und die geschriebenen Registry-Keys browserspezifisch gruppiert werden.

**Uncommon Locations** Unter Betrachtung als Uncommon Location werden alle Windows Registry Hives in jedem Festplatten-Image mit dem Registry Explorer untersucht. Dabei wird zwischen Hives zur Speicherung von Systemeinstellungen (System-Hives) und individuellen Benutzerkonfigurationen (User-Hives) unterschieden. Diese in Tabelle 4.4 dargestellten Hives werden von Windows beim Start geladen und dienen Systemkomponenten und Anwendungen als Quelle für Einstellungen und Informationen. [17] Zur Analyse wird jeder Hive aus den Festplatten-Images extrahiert und in eine Registry-Explorer-Sitzung geladen, um eine Stringsuche nach PB-Artefakten durchzuführen.

Tabelle 4.4.: Windows Registry Hives

System-Hives (C:\Windows\System32\Config)	
Dateiname	Inhalt
DEFAULT	Standardkonfigurationseinstellungen für neue Benutzerprofile
SAM	Sicherheitskontendaten, einschließlich der Benutzerkonten und deren Kennwörter
SECURITY	Sicherheitsinformationen für die Zugriffssteuerung und Authentifizierung
SOFTWARE	Konfigurationsdaten für installierte Software und Anwendungen
SYSTEM	Systemkonfigurationseinstellungen und Gerätetreiberinformationen

User-Hives (C:\Users\<username>)	
Dateiname	Inhalt
NTUSER.DAT	Individuelle Einstellungen und Konfigurationen für den angemeldeten Benutzer
USRCLASS.DAT	Dateizuordnungen und Registrierungseinstellungen für den angemeldeten Benutzer

## 5. Ergebnisse

In diesem Kapitel werden die vier Browser Mozilla Firefox, Tor-Browser, Google Chrome sowie Brave gemäß definierter Methodik in Kapitel 4 analysiert. Dabei wird für jeden Browser untersucht, ob Private-Browsing-Artefakte in den Common Locations, Uncommon Locations und der Registry hinterlassen wurden.

### 5.1. Firefox

Im nachfolgenden Abschnitt werden die Ergebnisse der Datenanalyse für den Webbrowser Firefox beschrieben. Die Analyse ist in drei Hauptkategorien unterteilt: Common Locations, Uncommon Locations und Registry.

#### Common Locations

Zunächst werden die Common Locations nach potenziellen privaten Browsing-Artefakten untersucht. Diese standardmäßigen Speicherorte für Browsing-Artefakte umfassen ausschließlich auf die Festplatte geschriebene Dateien. In diesem Versuch wurde gemäß Methodik in Abschnitt 4.3.1 zwischen Datei-Schreiboperationen aus den Process Monitor Logfiles und SQLite-Datenbanken zur Verwaltung von Nutzerdaten unterschieden. Weder in den Schreiboperationen der Process Monitor Logfiles noch in den SQLite-Datenbanken konnten PB-Artefakte gefunden werden.

Eine detaillierte Analyse der untersuchten Dateien ist im Anhang B.1 beschrieben.

#### Uncommon Locations

Nachfolgend werden die Analyseergebnisse der Firefox Uncommon Locations beschrieben. Im Gegensatz zu den Common Locations benötigt ein Forensiker dabei kein Wissen über das Browserverhalten. Stattdessen wird sich auf die Vollständigkeit der Funktionen von Forensik-Tools verlassen. Im Rahmen dieses Versuchs werden die Tools Autopsy und Volatility verwendet.

### Analyse mit Autopsy

Zur Analyse der Common Locations in Abschnitt 5.1 wird Autopsy zur Dateixtraktion genutzt. Im Falle der Uncommon Locations dient Autopsy zusätzlich als forensisches Werkzeug zur Datenanalyse.

Eine Autopsy Stichwortsuche gemäß Methodik in Abschnitt 4.3.2 lieferte in allen Snapshots keine Treffer. Dabei wurde zusätzlich das \$Carved Verzeichnis durchsucht, in dem Autopsy alle wiederhergestellten Dateien speichert.

Ebenso wurden in den von Autopsy automatisch kategorisierten Dateien keine PB-Artefakte gefunden. Eine detaillierte Analyse der Kategorien „Web Bookmarks“, „Web Cookies“, „Web History“ sowie „Web Categories“ ist im Anhang B.2 beschrieben.

### Analyse mit Volatility

Zur Untersuchung des RAM als Uncommon Location wurde eine Stringsuche in den gesamten Arbeitsspeicherabbildern nach PB-Artefakten durchgeführt. Wie in Abschnitt 4.3.2 ausführlich beschrieben, muss ein gefundener String eindeutig einem Browser zugeordnet werden können. Dazu wird mit dem Volatility PlugIn *Yarascan* nach den in Anhang A aufgeführten Yara-Regeln im RAM gesucht. Davon ausgehend wird das PlugIn *pslist* verwendet, um den Prozessnamen anhand PID zu identifizieren. Die Ergebnisse dieser Stringsuche sind nachfolgend nach den Yara-Regeln geordnet.

**Yara-Regel „HTML“** In keinem der Firefox RAM Dumps wurden HTML Fragmente der besuchten Seiten gefunden. Somit wird diese Yara-Regel nicht weiter betrachtet.

**Yara-Regel „Suchbegriffe“** Wie in Tabelle 5.1 gezeigt, wurden die Suchbegriffe „pfaffenhofen“, „nanoradar“, „mooserliesl“ sowie „mallofamerica“ ausschließlich nach dem Browsing-Szenario mit geöffnetem Browser (RAM Dump 2) identifiziert. Die Suchbegriffe wurden größtenteils in den Speicherbereichen von Firefox-Prozessen gefunden. Nur in elf Fällen wurden Suchbegriffe in anderen Prozessen identifiziert. Am häufigsten wurde der Suchbegriff „pfaffenhofen“ mit 1301 Artefakten gefunden. Dies ist vermutlich auf den visuellen Google Maps Kartenausschnitt zurückzuführen, welcher bei der Google-Suche relevante Informationen über die gesuchte Stadt zeigt.

**Yara-Regel „URLs“** Es konnten in den Arbeitsspeicherabbildern alle besuchten URLs „unitree.com“, „mooserliesl.de“, „mallofamerica.com“ sowie „donaukurier.de“ identifiziert werden. Wie in Tabelle 5.2 gezeigt, wurden die meisten Artefakte nach dem Browsing-Szenario mit geöffnetem Browser (RAM Dump 2) gefunden. Die besuchten URLs wurden hauptsächlich in Firefox-Prozessen gefunden. Die URL „mooserliesl.de“ wurde mit insgesamt 390 Artefakten am wenigsten gefunden, „donaukurier.de“ mit über 3600 Artefakten am häufigsten.

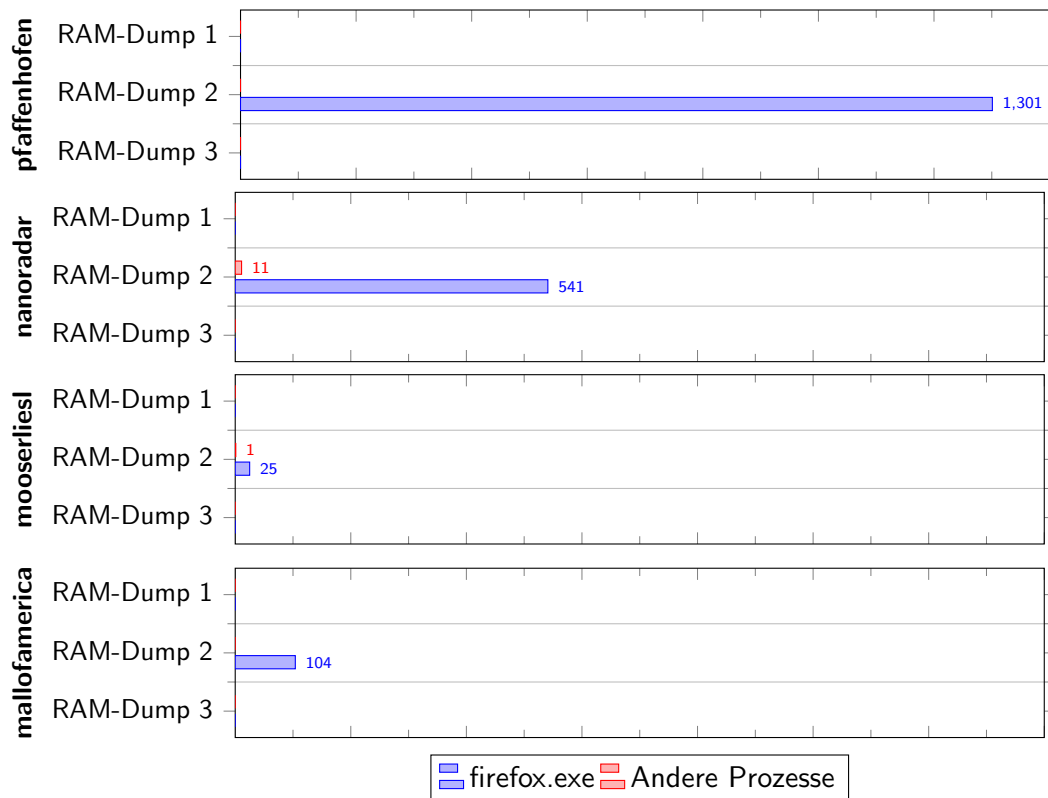


Abbildung 5.1.: Firefox: Anzahl gefundener Suchbegriffe im RAM

Bemerkenswert ist, dass URL-Artefakte gefunden wurden, selbst nachdem der Browser geschlossen wurde (RAM Dump 3). Dabei wurde kein URL-Artefakt in einem Firefox Prozess gefunden. Anhand der PID 2252 wurde festgestellt, dass sich alle URL-Artefakte nach Schließen des Browsers (RAM-Dump 3) in einem *svchost.exe* Prozess mit der gleichen PID befinden. Unter dem *Service Host* Prozess laufen gruppierte Windows-Dienste, um Ressourcen zu sparen und die Systemleistung zu verbessern. Volatility bietet das Plugin *svcsan* an, mit dem alle laufenden Dienste ausgegeben werden können. Bei Anwendung auf den dritten RAM Dump wurde jedoch zu keinem Dienst eine PID angegeben, wodurch der Dienst mit den URL Artefakten nicht im RAM identifiziert werden konnte. [40] Stattdessen wurde der dritte Snapshot aufgetaut, um im laufenden Windowsbetrieb den Dienst mithilfe des Process Explorers zu identifizieren. Wie in Abbildung 5.3 gezeigt, wurde anhand der PID 2252 der Dienst *DNSSCache* ermittelt. Der *DNSSCache*-Dienst unter Windows ist ein Teil des Betriebssystems, der für die Übersetzung von Domainnamen in IP-Adressen verantwortlich ist. Der *DNSSCache*-Dienst speichert DNS-Anfragen und Antworten temporär, um wiederholte DNS-Anfragen zu beschleunigen. [31] Nach Löschen des *DNSSCaches* mit dem Kommandozeilenbefehl `ipconfig /flushdns`, dem Schließen aller Process Monitor Instanzen sowie Beenden des *DNSSCaches* Services wurde erneut ein RAM-Dump durchgeführt. Dort wurden keine URL Artefakte mehr gefunden.

## 5. Ergebnisse

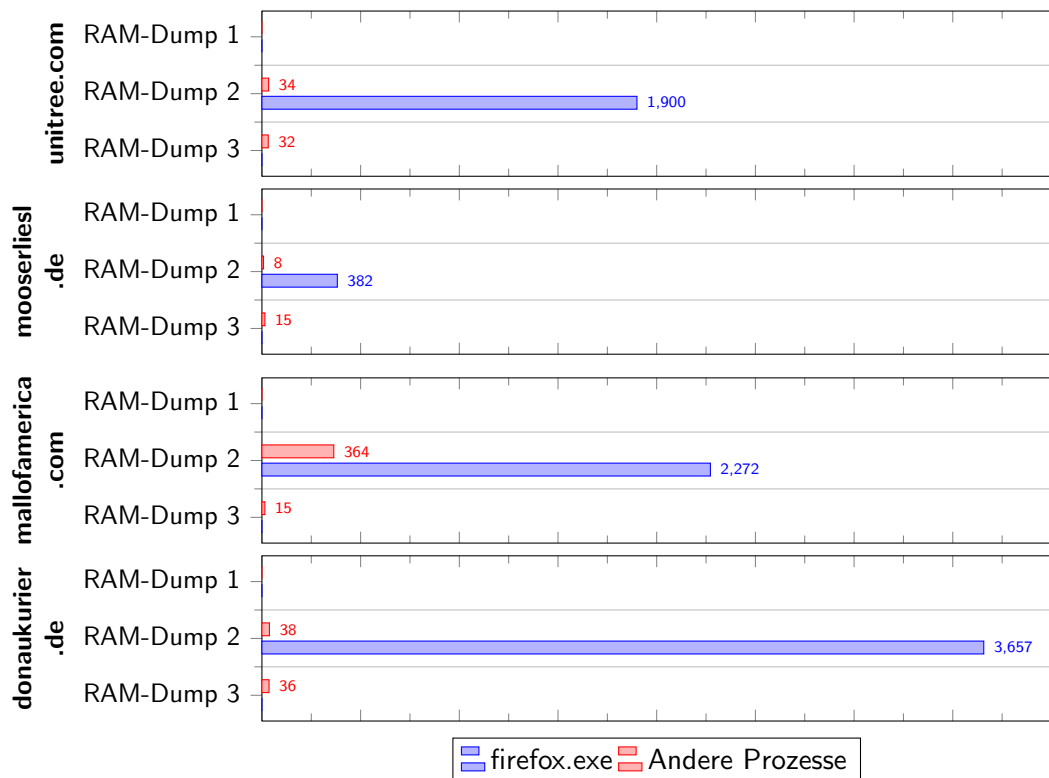


Abbildung 5.2.: Firefox: Anzahl gefundener URL-Artefakte im RAM

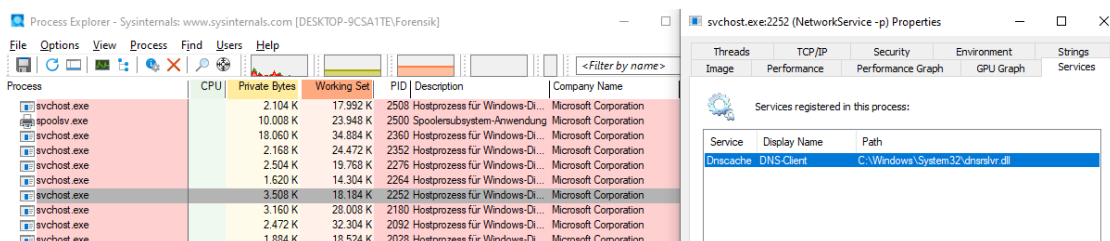


Abbildung 5.3.: Firefox: Unter dem SVChost-Prozess PID 2252 läuft der DNSCache-Dienst.

**Yara-Regel „E-Mail“** Wie in Abbildung 5.4 gezeigt, wurden E-Mail-Artefakte ausschließlich nach dem Browsing-Szenario mit geöffnetem Firefox Browser (RAM Dump 2) gefunden. Dabei wurden Artefakte jeder Kategorie gefunden. Unter den gefundenen Artefakten befindet sich am häufigsten die Absenderadresse „computerforensikvl@gmail.com“. Dieses Artefakt wurde als einziges E-Mail-Artefakt sechsmal in anderen Prozessen als Firefox gefunden.

Bemerkenswert ist, dass das Passwort des Google-Accounts, mit dem die E-Mails verschickt wurden, viermal als Klartext im RAM gefunden wurden. Das Passwort wurde je zweimal in zwei Firefox Prozessen mit den PIDs 7420 und 8424 gefunden. Tabelle 5.1 zeigt die virtuellen Speicheradressen der Artefakte aus der Yarascan Ausgabe.

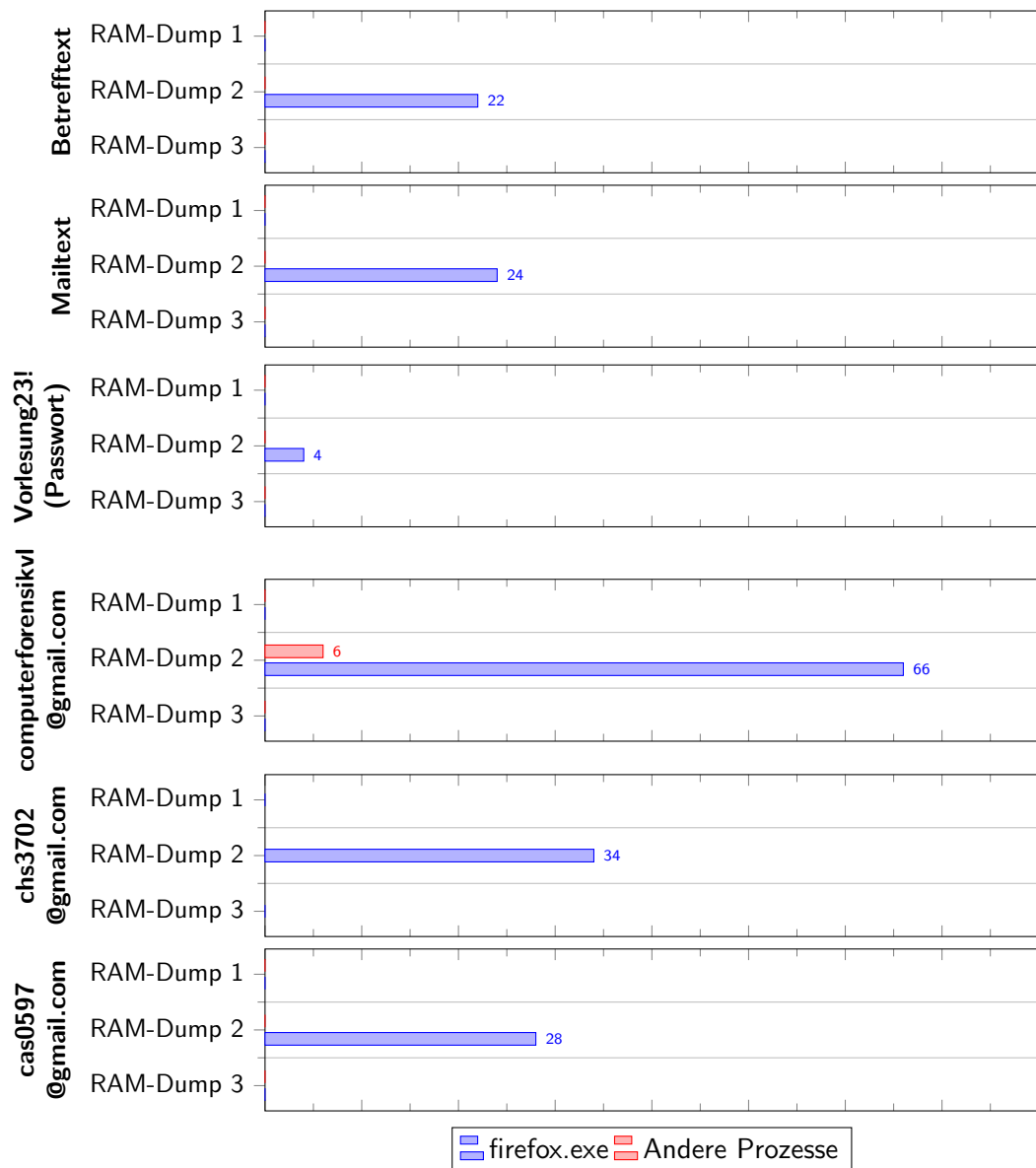


Abbildung 5.4.: Firefox: Anzahl gefundener E-Mail Artefakte im RAM

Zu diesen Artefakten wurde gemäß Methodik in Abschnitt 5.1 der String Kontext – also die Zeichen vor und nach dem gefundenen Artefakt im Speicherbereich – ermittelt. Dazu wurde mithilfe des Volatility memmap Plugins die Abbildung der virtuellen Speicheradressen auf den Byte-Offset in der extrahierten Speicherseite des Prozesses ermittelt.

Wie in Abbildung 5.5 gezeigt, sind in der Speicherseite des Prozesses mit PID 7420 in unmittelbarer Umgebung des gefundenen Passworts am Byte-Offset 0xb9ce29180c8 Code-Fragmente der *Gecko-Engine* zu finden. Dieser Teil des Firefox Browsers ist für das Rendering von We-





gefunden.

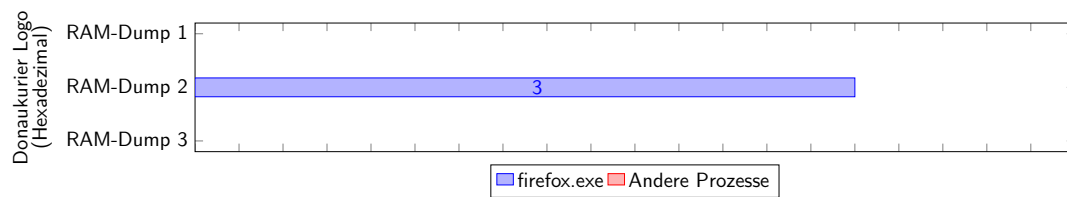


Abbildung 5.7.: Firefox: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im RAM

### Registry

Die Analyse der Registry zählt gemäß Methodik in Abschnitt 4.3.3 sowohl zu den Common als auch Uncommon Locations. Weder in den Process Monitor „SetValue“ Operations noch in den System- und User-Hives konnten PB-Artefakte gefunden werden. Eine detaillierte Analyse dieser Common- und Uncommon Locations der Registry ist im Anhang B.3 beschrieben.

## 5.2. Tor-Browser

In diesem Abschnitt werden die Ergebnisse der Datenanalyse der Common Locations, Uncommon Locations sowie der Registry für den Tor-Browser präsentiert.

### Common Locations

Zunächst werden die Common Locations analysiert, um potenzielle Hinweise auf Internetaktivitäten des Browsing-Szenarios zu finden. Bei der Untersuchung der gängigen Speicherorte wurde gemäß der im Abschnitt 4.3.1 beschriebenen Methodik zwischen Schreibvorgängen in den Logdateien des Process Monitors und den SQLite-Datenbanken zur Verwaltung von Benutzerdaten unterschieden. Dabei konnten in keiner Datei PB-Artefakte gefunden werden. Eine detaillierte Analyse der Process Monitor „WriteFile“ Operations sowie der SQLite-Datenbanken ist im Anhang C.1 beschrieben.

### Uncommon Locations

Nachfolgend werden die Analyseergebnisse der Tor Uncommon Locations beschrieben. Dazu werden die vollständigen Speicherabbilder nach PB-Artefakten untersucht, ohne das genaue Browserverhalten zu berücksichtigen. Stattdessen wird sich auf die Vollständigkeit der Funktionen der Forensik-Tools Autopsy und Volatility verlassen.

### Analyse mit Autopsy

Autopsy wird bei den Uncommon Locations als konkretes forensisches Werkzeug verwendet, statt nur zur Dateieextraktion, wie es bei den Common Locations der Fall war.

Eine Stichwortsuche nach PB-Artefakten in Autopsy in allen fünf Tor Festplatten-Images ergab keine Treffer.

Ebenso wurden in den automatisch von Autopsy kategorisierten Dateien keine PB-Artefakte gefunden. Im Anhang C.2 ist eine detaillierte Analyse der kategorisierten Dateien beschrieben.

### Analyse mit Volatility

Bei der Untersuchung der Tor RAM-Dumps mithilfe des Volatility Plugins Yarascan, konnten PB-Artefakte identifiziert werden. Nachfolgend werden die Ergebnisse geordnet nach Yara-Regeln beschrieben. Die vollständigen Yara-Regeln sind im Anhang A aufgelistet.

**Yara-Regel „HTML“** Wie bei Firefox, konnten keine HTML-Artefakte im RAM gefunden werden. Deshalb wird diese Kategorie nicht weiter aufgeführt.

**Yara-Regel „Suchbegriffe“** Wie in Abbildung 5.8 gezeigt, wurden nach dem Browsing-Szenarios sowie vor (RAM-Dump 2) als auch nach Erstellen einer „Neuen Identität“ (RAM Dump 3-1) Suchbegriffe des Browsing-Szenarios gefunden. Nachdem eine „Neue Identität“ erstellt wurde, reduzierten sich die gefundenen Artefakte deutlich. Die Suchbegriffe wurden hauptsächlich in Firefox-Prozessen gefunden. Kein Artefakt war im Tor-Prozess zu finden. Mit 4833 Artefakten wurde am häufigsten der Suchbegriff „pfaffenhofen“ nach dem Browsing-Szenario im zweiten RAM-Dump gefunden. Nach dem Schließen des Tor-Browsers wurden keine Suchbegriffe im RAM identifiziert.

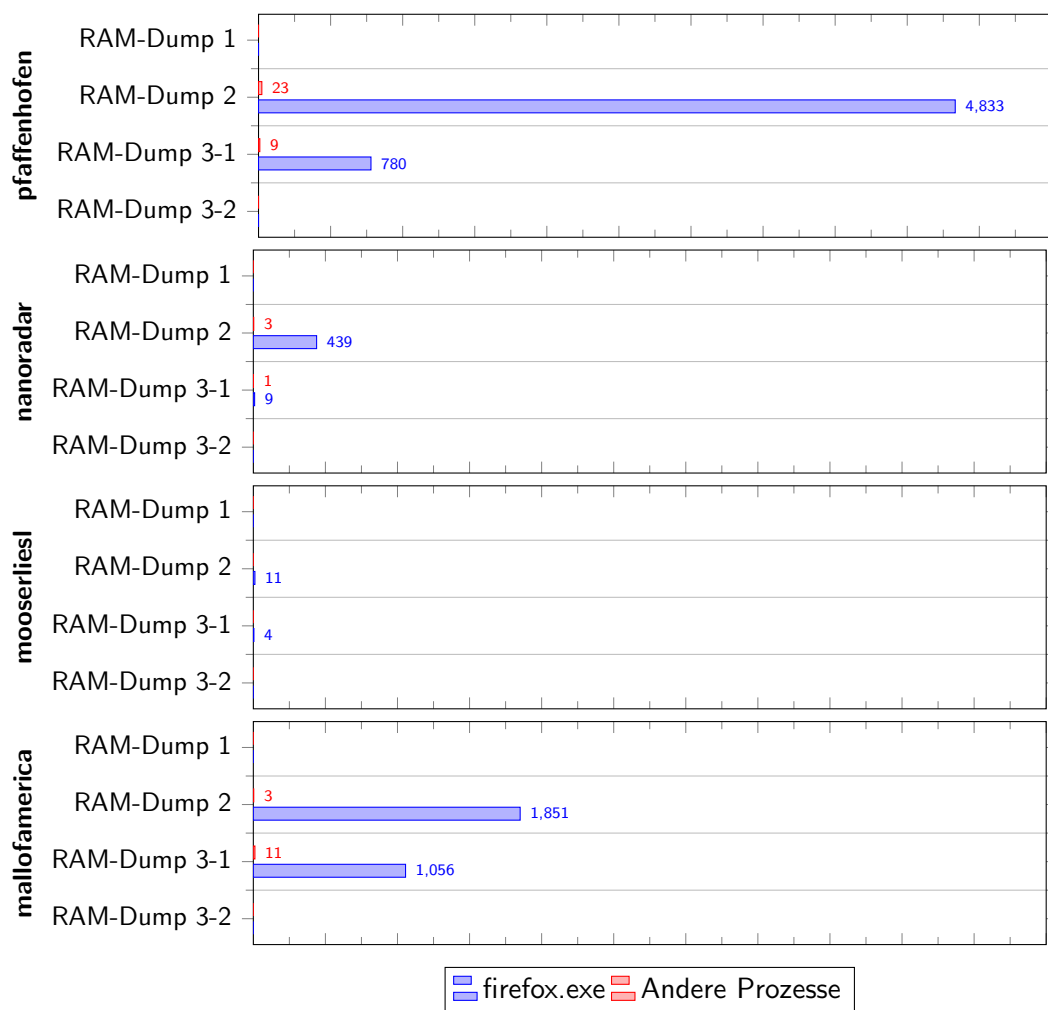


Abbildung 5.8.: Tor-Browser: Anzahl gefundener Suchbegriffe im RAM

**Yara-Regel „URLs“** Ähnlich zur Yara-Regel „Suchbegriffe“ wurden, wie in Abbildung 5.9 gezeigt, ausschließlich nach dem Browsing-Szenario vor (RAM-Dump 2) und nach Zuweisung einer neuen Identität (RAM-Dump 3-1) URL Artefakte gefunden. Dabei wurden im RAM Dump 3-1 deutlich weniger URL-Artefakte als in RAM Dump 2 gefunden. Artefakte dieser Yara-Regel wurden nach Firefox-Prozessen hauptsächlich in Tor-Prozessen gefunden. Am wenigsten Artefakte wurden in anderen Prozessen identifiziert. Auffällig ist, dass die URL „mallofamerica.com“ 26505 Mal in RAM-Dump 2 gefunden wurde. Im Gegensatz dazu wurde „mooserliesl.de“ nur 518 Mal gefunden. Nach Schließen des Tor-Browsers wurden keine URL-Artefakte mehr im RAM gefunden.

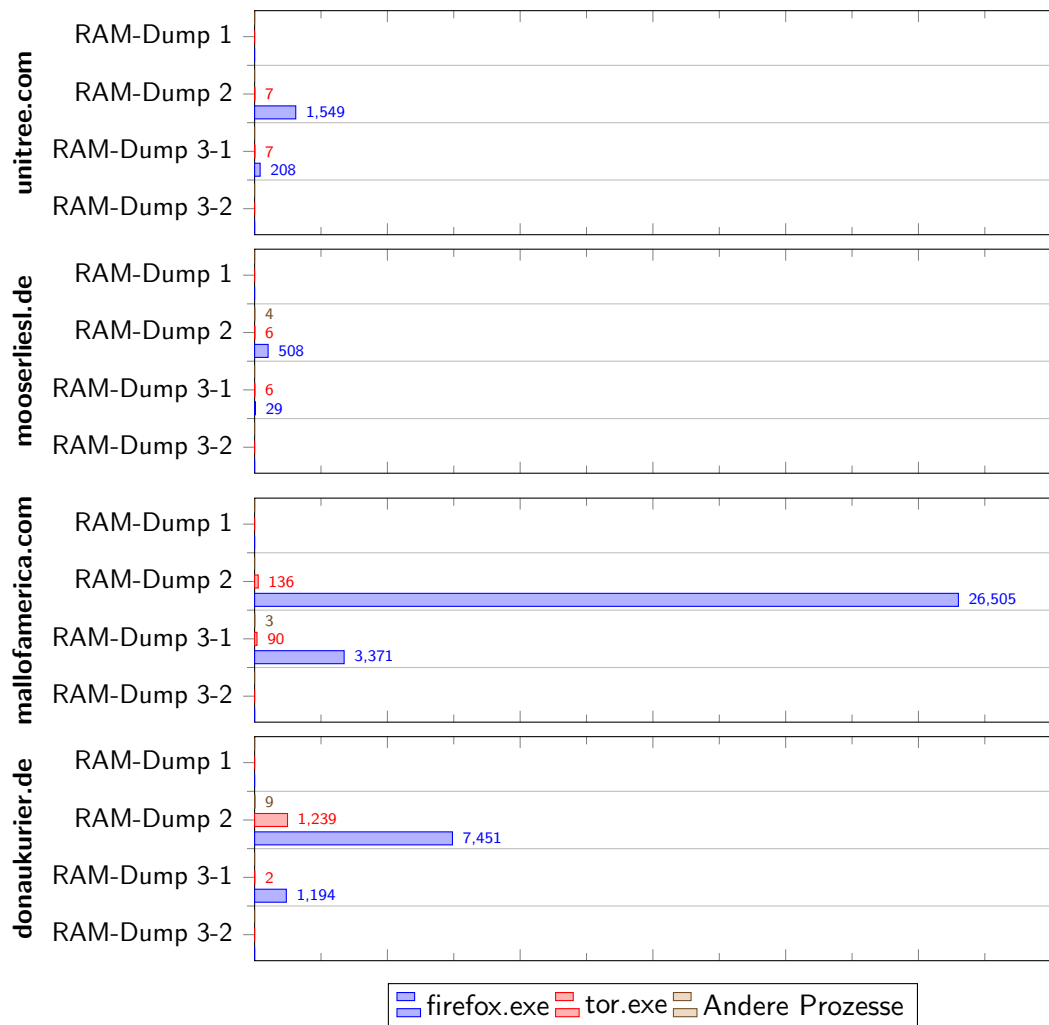


Abbildung 5.9.: Tor-Browser: Anzahl gefundener URL-Artefakte im RAM

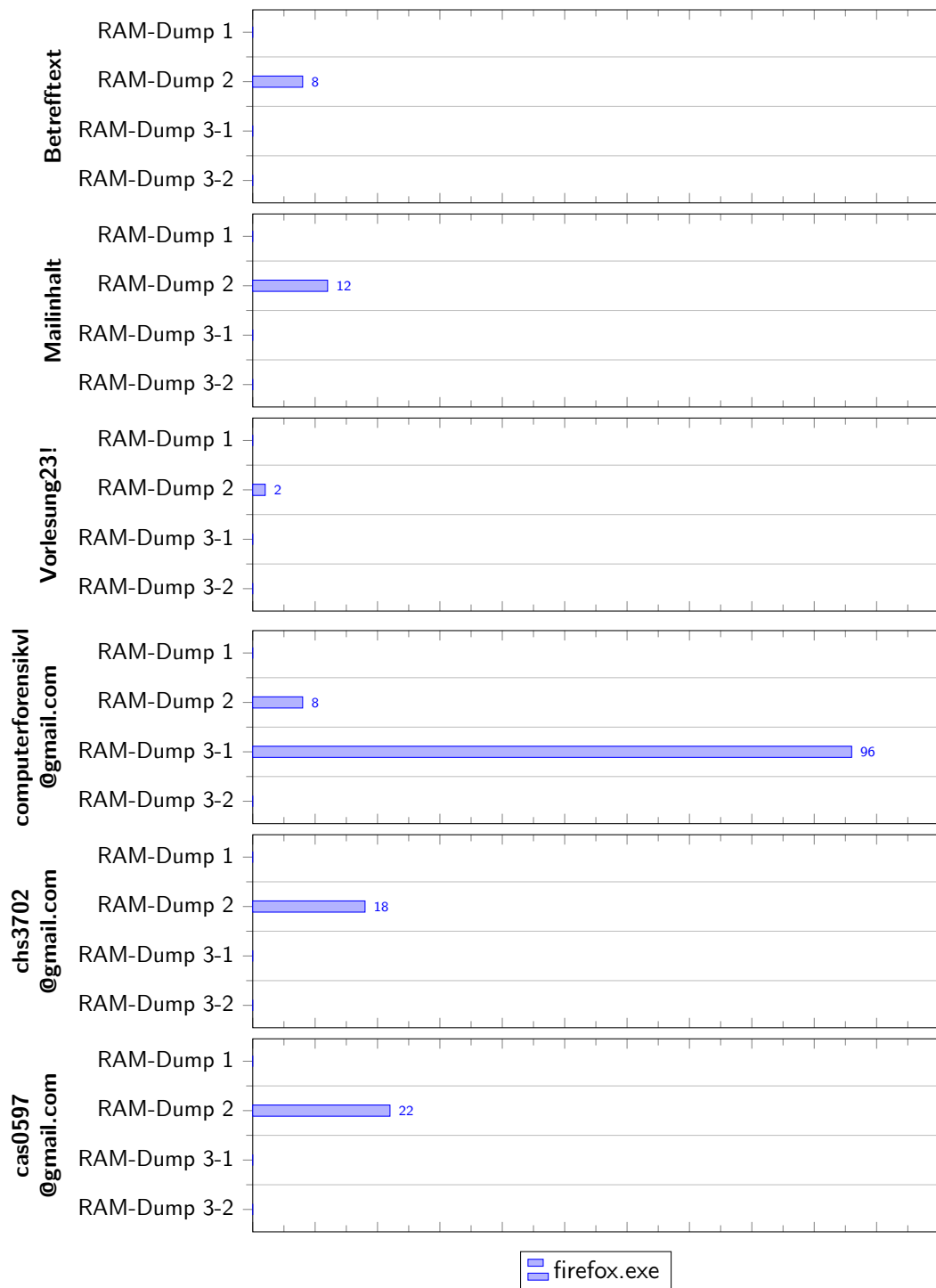


Abbildung 5.10.: Tor-Browser: Anzahl gefundener E-Mail Artefakte im RAM

**Yara-Regel „E-Mail“** Nach dem Browsing-Szenario, vor Zuweisung einer „Neuen Identität“ (RAM-Dump 2), konnten alle Kategorien von E-Mail-Artefakten gefunden werden. Nur die Absenderadresse „computerforensikvl@gmail.com“ wurde noch nach Erstellen der „Neuen Identität“ (RAM-Dump 3-1) gefunden. Die Absenderadresse ist ebenso das am häufigsten gefundene E-Mail Artefakt. Wie in Abbildung 5.10 dargestellt, wurden die Artefakte ausschließlich in Firefox-Prozessen gefunden. Wie bei der Analyse der Firefox RAM-Dumps in Abschnitt 5.1, wurde das Passwort als Klartext nach dem Browsing-Szenario, vor Zuweisung einer „Neuen Identität“ (RAM-Dump 2) gefunden.

Tabelle 5.2.: Tor-Browser: Abbildung der virtuellen Speicheradressen der gefundenen Strings im RAM auf Byte-Offsets der entsprechenden Speicherseiten

Virtuelle Speicheradresse	PID	Byte-Offset in extrahierter Speicherseite
0x2b1e2c22318	708	0xea0318
0x2b1e2ecb748	708	0x10f7748

Das Passwort wurde zweimal im Firefox Prozess mit der PID 708 gefunden. Tabelle 5.2 zeigt die virtuellen Speicheradressen der Artefakte aus der Yarascan Ausgabe sowie deren Abbildung auf die mithilfe des Volatility Plugins *memmap* identifizierten Byte-Offsets der extrahierten Speicherseiten.

```

00EA02D0 00 00 00 00 E5 E5 E5 E5 01 00 E5 E5 E5 E5 E5 E5 ....AAAA.AAAAAA
00EA02E0 C0 94 63 77 FC 7F 00 00 E8 22 C2 E2 B1 02 00 00 A"cwü...e"Ää±...
00EA02F0 E8 22 C2 E2 B1 02 00 00 01 E5 E5 E5 E5 E5 E5 e"Ää±...AAAAAA
00EA0300 00 69 49 EC B1 02 00 00 E5 E5 E5 E5 E5 E5 E5 E5 .iI±...AAAAAA
00EA0310 02 00 00 00 1A 00 00 00 56 00 6F 00 72 00 6C 00 .....V.o.r.t.
00EA0320 56 00 73 00 73 00 6E 00 67 00 32 00 33 00 21 00 ...Küßch&2014.
00EA0330 00 00 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 .AAAA.AAAAAA
00EA0340 00 00 00 00 40 D8 F6 FF 60 EA 00 00 F8 2B 09 00 ...@00y"ë...e+...
00EA0350 00 00 00 00 40 D8 F6 FF 60 EA 00 00 F8 2B 09 00 ...@00y"ë...e+...
00EA0360 01 00 00 C0 01 00 00 C0 50 3C 4D EC B1 02 00 00 ...Ä...ÄP<Hi±...
010F76C0 01 00 00 00 38 00 00 00 43 53 50 5F 69 67 6E 6F ....8...CSF_igno
010F76D0 72 69 6E 67 63 72 63 46 6F 72 53 74 72 69 63 74 ringSrcForStrict
010F76E0 44 79 6E 61 6D 69 63 00 E5 E5 E5 E5 E5 E5 E5 E5 Dynamic.AAAAAA
010F76F0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 .AAAA.AAAAAA
010F7700 A0 07 18 77 FC 7F 00 00 E8 07 18 77 FC 7F 00 00 ...wü...ë...wü...
010F7710 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010F7720 04 00 00 00 00 00 00 00 00 00 4D CF E4 B1 02 00 00 .....Mie±...
010F7730 98 2C E9 76 FC 7F 00 00 00 00 00 00 00 00 00 00 ",ëvü.....
010F7740 01 00 00 00 38 00 00 00 56 00 6F 00 72 00 6C 00 .....8...V.o.r.t.
010F7750 56 00 73 00 73 00 6E 00 67 00 32 00 33 00 21 00 ...Küßch&2014.
010F7760 00 00 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 .AAAA.AAAAAA
010F7770 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 .AAAA.AAAAAA
010F7780 02 00 00 00 03 00 00 00 C8 06 0F 77 FC 7F 00 00 .....E...wü...
010F7790 60 A3 07 EB B1 02 00 00 50 7B 64 77 FC 7F 00 00 "E.e±...P(dwü...
010F77A0 10 2C D8 E2 B1 02 00 00 10 E7 67 77 FC 7F 00 00 ...@ä±...gwü...
010F77B0 A0 9F 0B E4 B1 02 00 00 E5 E5 E5 E5 E5 E5 E5 E5 1.ä±...AAAAAA
010F77C0 01 00 00 00 38 00 00 00 2F 00 69 00 6E 00 76 00 ...8.../i.n.v.
010F77D0 61 00 6C 00 69 00 64 00 61 00 74 00 69 00 6F 00 a.i.i.d.a.t.i.o.
010F77E0 6E 00 2F 00 6C 00 63 00 73 00 2F 00 63 00 6C 00 n./l.c.s./c.l.
010F77F0 69 00 65 00 6E 00 74 00 00 00 E5 E5 E5 E5 E5 E5 i.e.n.t...AAAAAA

```

(a) Byte-Offset 0xea0318

(b) Byte-Offset 0x10f7748

Abbildung 5.11.: Tor-Browser: Passwort-Klartext in Speicherseiten von PID 708

Bei Untersuchung des String-Kontexts in Abbildung 5.11, wurden für das Passwort am Byte-Offset 0xea0318 keine auffälligen Artefakte entdeckt. Im Bereich des gefundenen Passworts am Byte-Offset 0x10f7748 befindet sich der String „CSP\_ignoringSrcForStrictDynamic“, dessen Bedeutung nicht näher bestimmt werden konnte. Weiterhin wurde die Zeichenkette „invalidation/lcs/client“ in der Nähe des Passworts gefunden. Auf diesen String wird in einem Firefox Bug-Ticket verwiesen, welches bereits 2017 geschlossen wurde. Der Bug betraf ein Speicher-Leck. [5] Der genaue Zusammenhang mit dem gefundenen Passwort konnte nicht ermittelt werden.

**Yara-Regel „DK-Logo“** Wie in Abbildung 5.12 dargestellt, wurde der Hexadezimal-Wert des Donaukurier-Logos ein einziges Mal nach dem Browsing-Szenario, vor Erstellen der „Neuen

Identität“ (RAM-Dump 2) in einem Firefox Prozess gefunden.

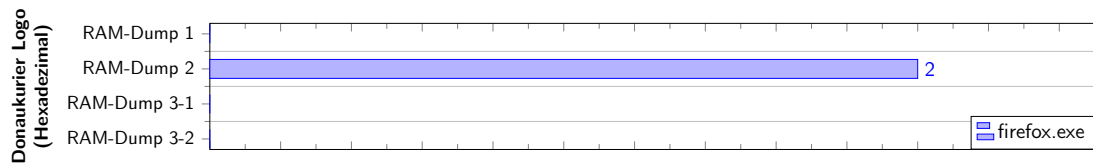


Abbildung 5.12.: Tor-Browser: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im RAM

### Registry

Wie in der Methodik in Abschnitt 4.3.3 beschrieben, teilt sich die Analyse der Registry sowohl in Common als auch Uncommon Locations. Weder in den Process Monitor „SetValue“ Operations noch über die Stringsuche in den System- und User-Hives konnten PB-Artefakte gefunden werden. Eine detaillierte Analyse der Registry ist im Anhang B.3 beschrieben.



### 5.3. Chrome

Anschließend folgt die Analyse des Webbrowsers Chrome. Die Ergebnisse werden dabei wie in den beiden vorherigen Kapitel präsentiert.

#### Common Locations

Zuerst werden die Common Locations nach Artefakten des privaten Browsing-Szenarios untersucht. Dabei wird wieder zwischen Datei-Schreiboperationen aus den Process Monitor Logfiles und den SQLite-Datenbanken unterschieden, wie im Unterabschnitt 4.3.1 erläutert. Es konnten darin jedoch keine Artefakte gefunden werden.

Eine weitergehende Analyse aller Schreiboperationen sowie den Datenbanken ist im Unterabschnitt D.1 detailliert aufgeführt.

#### Uncommon Locations

Nachfolgend werden die Ergebnisse der Uncommon Location Analyse des Chrome Browsers dargelegt. Dafür werden die beiden Forensik-Programme Autopsy und Volatility verwendet.

#### Analyse mit Autopsy

Auch in Autopsy konnten keine Artefakte der durchgeführten Browsing-Session gefunden werden, weder durch die Substringsuche, noch durch die Analyse der von Autopsy kategorisierten Dateien. Eine weitergehende Analyse einiger kategorisierter Dateien ist im Abschnitt D.2 zu finden.

#### Analyse mit Volatility

Nachdem bis jetzt noch keine Artefakte gefunden werden konnten, folgen nun die Analyseergebnisse des Hauptspeichers (RAM) mittels des Forensik-Tools Volatility. Im RAM konnten schließlich Artefakte gefunden werden. Diese wurden mittels des Volatility Plugins Yarascan identifiziert, wobei die Yara-Regeln aus Abschnitt A verwendet wurden. Zusätzlich zu den erkannten Strings spielt auch noch die PID eine entscheidende Rolle, da damit der Treffer entweder dem Browser oder einem anderen Prozess eindeutig zugeordnet werden kann. Die Ergebnisse dieser String-Suche werden nachfolgend dargelegt.

**Yara-Regel „HTML“** Im Gegensatz zu den Ergebnissen der Analyse der beiden vorherigen Browser, konnte ein HTML-Fragment bei Chrome im Arbeitsspeicher gefunden werden. Einzig der der String „>Themen:“ wurde im zweiten RAM-Dump in einem Chrome-Prozess gefunden.

Das Volatility-Plugin lieferte hierbei neben dem Treffer noch die PID (7012 in diesem Fall) des Prozesses, in welchem der String gefunden wurde, sowie die virtuelle Adresse, an welchem das Fragment gefunden wurde (0xa6c018187e1). Mit dieser Information wurde dann zunächst der Befehl `python3 vol.py -f DUMP_FILE windows.memmap -pid 7012` ausgeführt, um das Memory-Mapping des Prozesses zu erhalten, also die Zuordnung der virtuellen zu den physikalischen Adressen des Prozesses, sowie deren Offset im Memory-Dump-File. In Tabelle 5.3 ist ein Ausschnitt der Ausgabe des Befehls zu sehen.

Tabelle 5.3.: Chrome: Abbildung virtueller Adressen auf Byte-Offsets der Speicherseite

Virtual	Physical	Size	Offset in File	File Output
0xa6c01817000	0xc4efc000	0x1000	0x105d000	Disabled
0xa6c01818000	0xd2dfb000	0x1000	0x105e000	Disabled
0xa6c01819000	0xd50fa000	0x1000	0x105f000	Disabled

In der Farbe rot ist hervorgehoben, dass die virtuelle Adresse 0xa6c01818000 auf die physikalische Adresse 0xd2dfb000 gemapped ist und der Offset im memory-file 0x105e000 entspricht. Mithilfe dieser Adressen konnte dann die tatsächliche Position des String im memory-file errechnet werden. Diese betrug somit 0x105e7e1. Anschließend wurde der Speicherbereich des Prozesses als Memory-Dump-File gedumpt und in den Hex-Editor HxD geladen. Dort wurde dann genau an dem errechneten Offset nach dem String gesucht und konnte genau dort auch gefunden werden, was auch in Abbildung 5.13 zu sehen ist.

```

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Dekodierter Text
0105E7B0 63 65 6E 74 65 72 22 3E 3C 64 69 76 20 63 6C 61 center"><div cla
0105E7C0 73 73 3D 22 6D 61 72 67 69 6E 2D 72 69 67 68 74 ss="margin-right
0105E7D0 2D 31 35 20 66 6F 6E 74 2D 73 69 7A 65 2D 31 36 -15 font-size-16
0105E7E0 22 3E 54 68 65 6D 65 6E 3A 0A 09 09 09 09 3C 2F ">Themen:....</
0105E7F0 64 69 76 3E 3C 64 69 76 3E 3C 75 6C 20 63 6C 61 div><div><ul cla
0105E800 73 73 3D 22 6C 69 73 74 2D 69 6E 6C 69 6E 65 20 ss="list-inline
0105E810 6D 62 2D 30 22 3E 3C 6C 69 20 63 6C 61 73 73 3D mb-0"><li class=
0105E820 22 6C 69 73 74 2D 69 6E 6C 69 6E 65 2D 69 74 65 "list-inline-ite
0105E830 6D 22 3E 3C 61 20 68 72 65 66 3D 22 68 74 74 70 m"><a href="http

```

Abbildung 5.13.: Chrome: Ausschnitt aus dem Memory-Dump-File an der Stelle des gefundenen HTML-Strings

Daraufhin wurde weitergehend versucht zu analysieren, ob nur ein Teil der Website im RAM verfügbar ist, oder ob die komplette Donaukurier-Website als HTML Artefakt im RAM vorhanden ist. Dafür wurde die Internetseite donaukurier.de nochmals aufgerufen, heruntergeladen und der Beginn sowie das Ende der HTML-Datei analysiert. In Abbildung 5.14 sind die ersten sowie letzten Strings der HTML-Datei zu sehen:

Abbildung 5.14.: Chrome: Beginn (links) und Ende (rechts) des HTML-Dokuments der Donaukurier Webseite

[illegible]

Gleiches wurde mit dem Ende der HTML-Datei durchgeführt. Abbildung 5.15 zeigt rechts einen Screenshot mit dem relevanten String-Treffer, welcher das Ende der Website donaukurier.de darstellt.

Zusammenfassend konnte somit die Website „donaukurier.de“ im zweiten Memory-Dump

komplett aus dem RAM extrahiert und rekonstruiert werden.

**Yara-Regel „Suchbegriffe“** Wie in Abbildung 5.16 dargestellt, wurden die Suchbegriffe „pfaffenhofen“, „nanoradar“, „mooserliesl“ und „mallofamerica“ ausschließlich bei dem noch geöffnetem Chrome-Browser nach dem Browsing-Szenario (zweiter RAM Dump) gefunden. Es wurden dabei alle Suchbegriffe bis auf einen einzigen in Speicherbereichen des Chrome-Browsers gefunden. Am häufigsten konnte der Suchbegriff „pfaffenhofen“ mit 1922 Treffern identifiziert werden. Am seltensten war das der Fall bei dem Begriff „mooserliesl“ mit nur 281 Suchtreffern.

Der eine Treffer bei der Suchphrase „mallofamerica“ war im Speicher eines Prozesses mit dem Namen *MemCompression*, was anhand der PID 1828 festgestellt wurde. Dieser Prozess ist dafür zuständig, dass Windows einen Teil des Arbeitsspeichers komprimiert speichert, was zwar zusätzliche CPU-Ressourcen beansprucht, dafür jedoch deutlich schneller ist, als den Hauptspeicher in das pagefile auszulagern [21]. Daher kann es sein, dass ein Teil des RAMs, in welchem der Suchbegriff vorhanden war, komprimiert wurde und folglich dieser Prozess diesen Begriff beinhaltete.

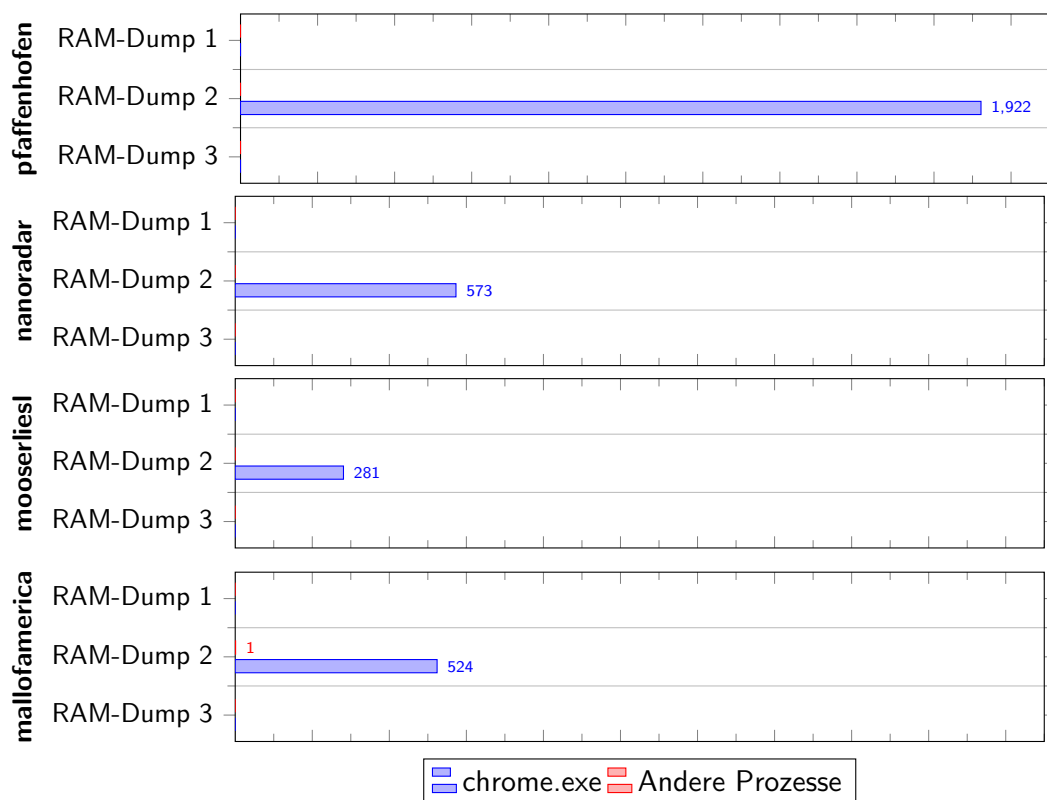


Abbildung 5.16.: Chrome: Anzahl gefundener Suchbegriffe im RAM

**Yara-Regel „URLs“** Abbildung 5.17 zeigt, dass in den RAM-Dumps alle Suchbegriffe identifiziert werden konnten. Im ersten Speicherabbild wurden dabei keine URLs gefunden, im zweiten am meisten und im dritten, also nach Beenden des Browsers, konnten auch noch fünf URLs identifiziert werden. Dabei waren es bei der URL „donaukurier.com“ am meisten Suchtreffer mit insgesamt 8157 Treffern, zehn davon waren nicht im Speicher von Chrome-Prozessen zu finden. Zwei davon waren im Prozess *MemCompression*, die restlichen acht wurden im Prozess mit der PID 3760 identifiziert, was in diesem Fall der *sihost.exe* war. Dieses Programm entspricht dem *Shell Infrastructure Host*, welcher die Grafikbenutzeroberfläche erstellt und verwaltet, wie beispielsweise Desktop-Hintergründe, Popup-Benachrichtigungen und Taskleisten [55]. Im Prozessbereich dieses Programms befanden sich auch die fünf Treffer aus dem dritten RAM-Dump.

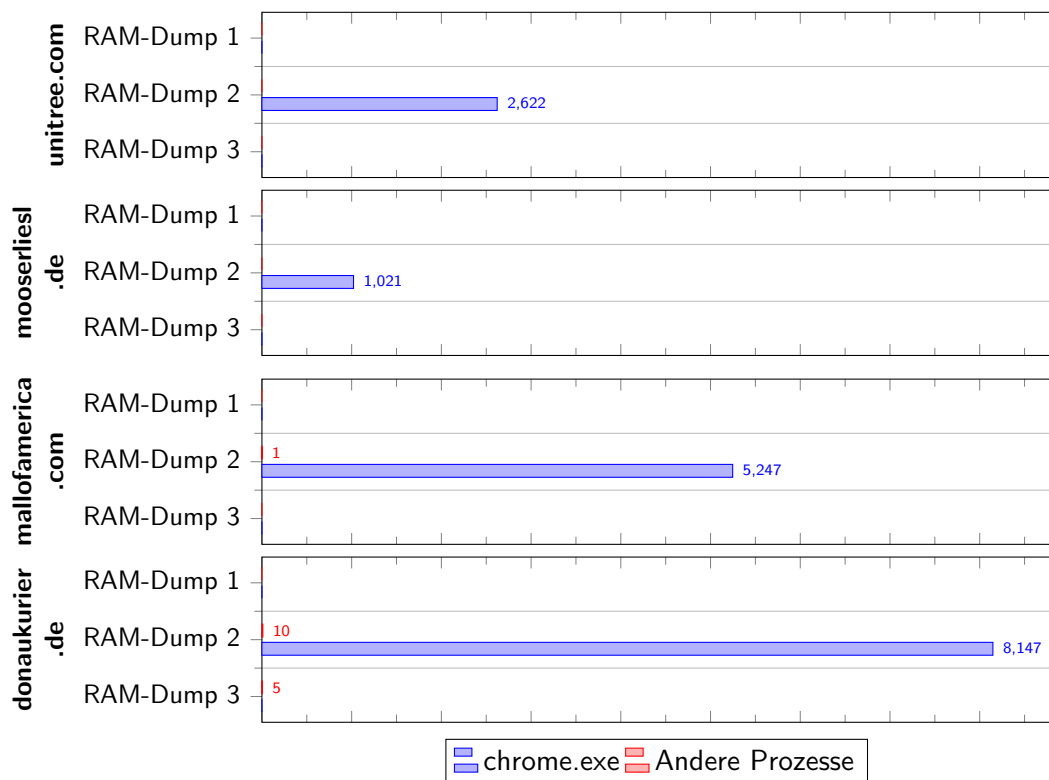


Abbildung 5.17.: Chrome: Anzahl gefundener URLs im RAM

**Yara-Regel „E-Mail“** Wie in Abbildung 5.18 gezeigt, wurden sowohl nach dem Browsing-Szenario bei geöffnetem Browser, als auch nach Schließen desselben E-Mail Artefakte gefunden. Darunter war die E-Mail-Adresse mit 172 Treffern, acht davon außerhalb von Chrome-Prozessen, das häufigste Suchresultat. unter den acht externen Prozessen waren der *Desktop Window Manager*, welcher für visuelle Effekte wie Desktop-Animationen und halbtransparente Fenster verantwortlich ist [44], *explorer.exe*, *sihost.exe* und weiteren Windows-Prozessen. Da

dies aber meist Window-Manager Prozesse waren, wurden diese nicht weiter analysiert. Die beiden Studenten-Mailadressen, an welche die Mail versendet wurden, waren mit 121 bzw. 97 Suchtreffern in Chrome-Prozessen vertreten. Der Mailtext war mit 136 Artefakten mehr als doppelt so oft im RAM zu finden als der Betrefftext. Im dritten RAM-Dump wurde dann noch die E-Mail-Adresse im Prozess *explorer.exe* gefunden. Grundsätzlich wurden bei dieser Yara-Regel wenige Artefakte identifiziert im Vergleich zu den URLs beispielsweise.

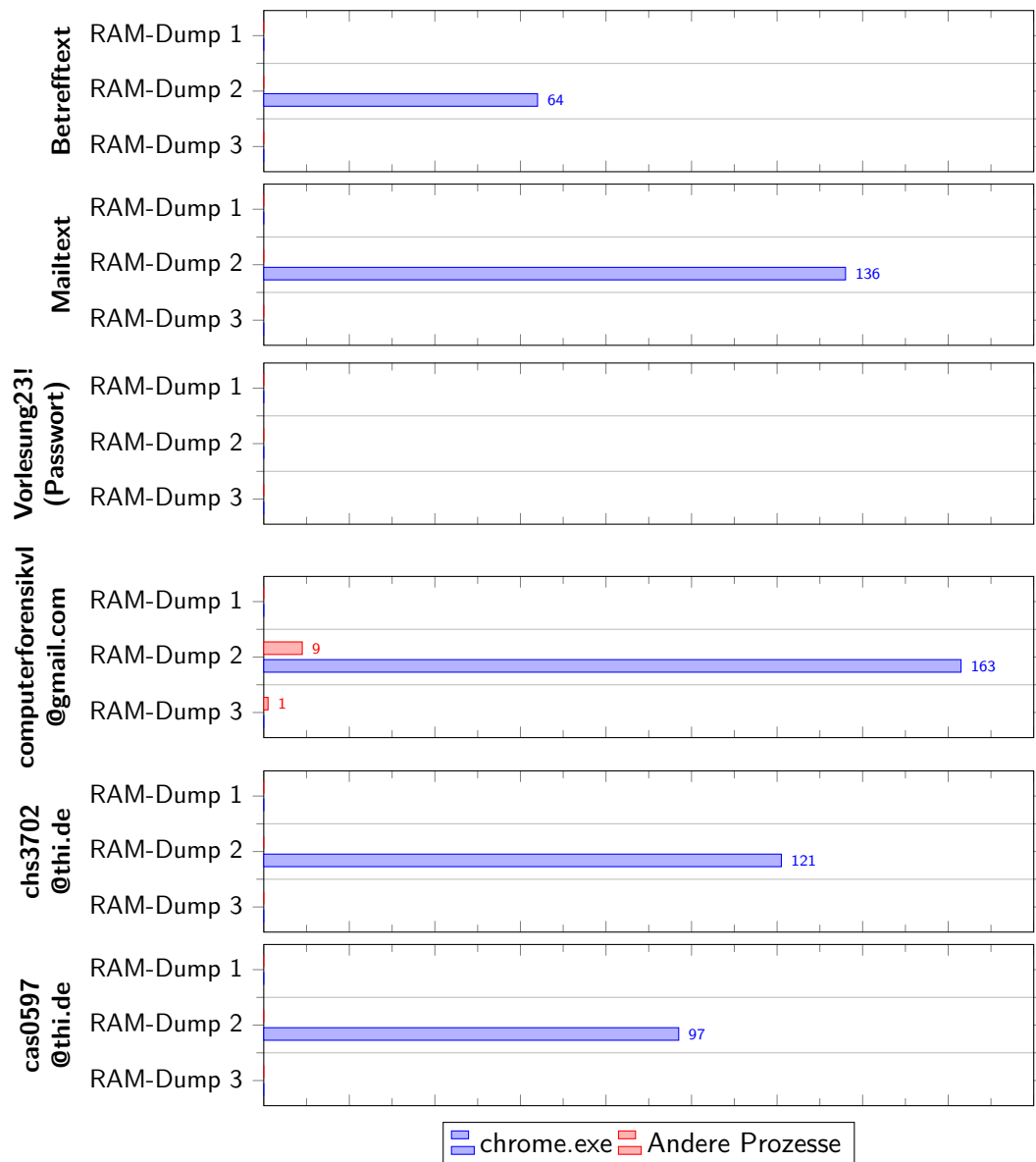


Abbildung 5.18.: Chrome: Anzahl gefundener E-Mail Artefakte im RAM

**Yara-Regel „DK-Logo“** Bei der letzten Yara-Regel zeigt Abbildung 5.19 die Ergebnisse, in welchem RAM-Dump und wie oft das Donaukurier Logo im RAM identifiziert werden konnte. Zu sehen ist, dass dies nur im zweiten Speicherabbild der Fall war und insgesamt dreimal dort in einem Chrome-Prozess zu finden war.

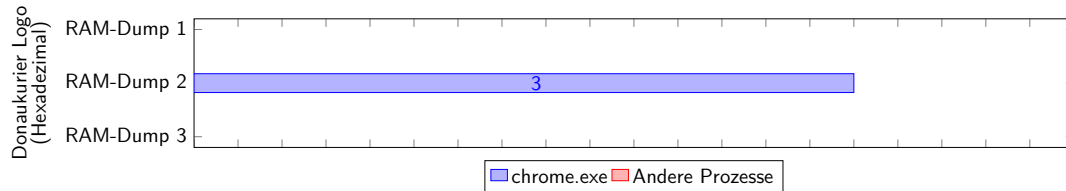


Abbildung 5.19.: Chrome: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im RAM

### Registry

Wie in Unterabschnitt 4.3.3 beschrieben, zählt die Analyse der Registry sowohl zu den Common als auch zu den Uncommon Locations. Es können weder in den „SetValue“-Operationen in den Process Monitor Logs noch durch die Analyse der System- und User-Hives durch den RegistryExplorer Artefakte gefunden werden. Eine weitergehende Analyse der Registry ist im Anhang D.3 beschrieben.

## 5.4. Brave

Abschließend werden in diesem Kapitel die Analyseergebnisse des Browsers Brave dargelegt, wobei diese wieder, wie in den vorherigen Kapiteln, in Common Locations, Uncommon Locations und der Registry unterschieden werden.

### Common Locations

Zu Beginn erfolgt die Untersuchung der Common Locations auf potenzielle private Browsing-Artefakte. Bei der Analyse der Schreiboperationen aus den Process Monitor Logfiles konnten keine Artefakte befunden werden, wie es auch bei der Untersuchung der SQLite-Datenbanken der Fall war.

Eine umfangreiche Auswertung der Daten sowie den Datenbankenn befindet sich im Unterabschnitt E.1.

### Uncommon Locations

Anschließend an die Common Locations folgt nun die Untersuchung der Uncommon Locations. Dafür werden vollständige Speicherabbilder nach Artefakten der privaten Browsing-Session untersucht. Für diesen Zweck werden die beiden Forensik-Programme Autopsy und Volatility verwendet.

#### Analyse mit Autopsy

Autopsy wird bei den Uncommon Locations zusätzlich als forensisches Werkzeug verwendet im Gegensatz zur Analyse der Common Locations, bei welchen es eingesetzt wurde, um Dateien aus den Snapshots zu extrahieren.

Zunächst wurde die Stringsuche gemäß Abschnitt 4.3.2 eingesetzt, wobei es keine Suchtreffer gab.

Zusätzlich wurden automatisch kategorisierte Dateien untersucht, wobei hier auch keine Artefakte zu finden waren. Abschnitt E.2 geht auf diese Dateien ausführlicher ein.

#### Analyse mit Volatility

Für die Analyse des Arbeitsspeichers wurde das Forensik-Tool Volatility verwendet, womit eine Stringsuche mittels des Plugins Yarascan durchgeführt wird. Damit durchsucht man ein Arbeitsspeicherabbild nach gewissen Strings, welche zuvor in einer Yara-Regel festgelegt werden können. Die für diese Arbeit verwendete Datei inklusive der Regeln ist in Anhang A aufgeführt.



**Yara-Regel „HTML“** Beim Brave-Browser konnte ausschließlich ein HTML-Fragment im zweiten RAM-Dump wiederhergestellt werden. Dabei handelt es sich wie bei Chrome um den String „>Themen:“.

Die Extraktion der kompletten Webseite war dabei wie in Abschnitt 5.3 möglich. Auch das Vorgehen war identisch, außer, wie zu erwarten war, die virtuellen und physikalischen Adressen der verschiedenen Strings. Daher wird hier nicht nochmal ausführlich auf das Vorgehen der Extraktion der Webseite aufgeführt.

Anschließend an die Wiederherstellung der HTML-Datei wurden die beiden extrahierten Dateien nochmals gegeneinander verglichen. Abbildung 5.20 zeigt dies anhand der *compare*-Funktionalität von Visual Studio Code. An der rechten Seite in der Bildlaufleiste sind die Sektionen farbig dargestellt, welche Unterschiede aufweisen. Dabei wurde durch Analyse derer deutlich, dass der Aufbau und die Struktur beider extrahierter Webseiten übereinstimmt, es nur geringe Unterschiede bei einigen Namen der Artikel und Meldungen gibt. Die Dateien sind nahezu identisch groß (290kB).

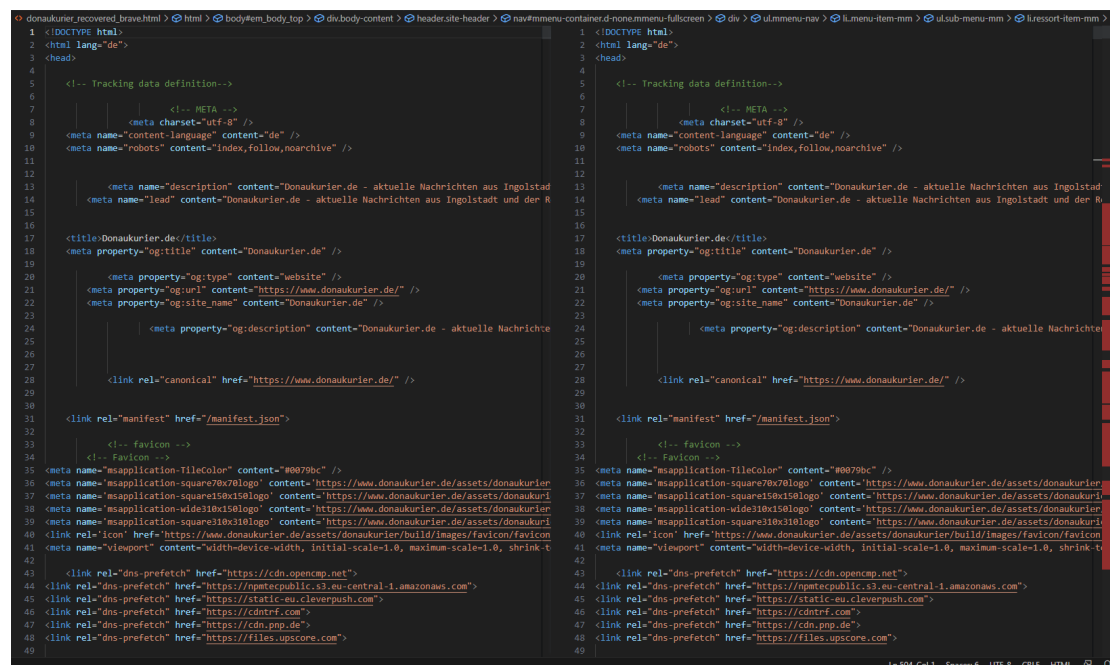


Abbildung 5.20.: Brave: Ausschnitt aus VS Code mit dem Vergleich der beiden Donaukurier Webseiten

**Yara-Regel „Suchbegriffe“** Abbildung 5.21 zeigt, dass alle der vier Suchbegriffe jeweils gefunden wurden, jedoch nur im zweiten RAM-Dump, also nach der Durchführung des Browsing-Szenarios bei noch geöffnetem Browser. Der String „pfaenhofen“ wurde dabei am häufigsten mit 1447 Treffern, „nanoradar“ am seltensten mit nur 51 Suchergebnissen gefunden. Dabei wurden auch alle Suchbegriffe rein in Brave-Prozessen ausfindig gemacht.

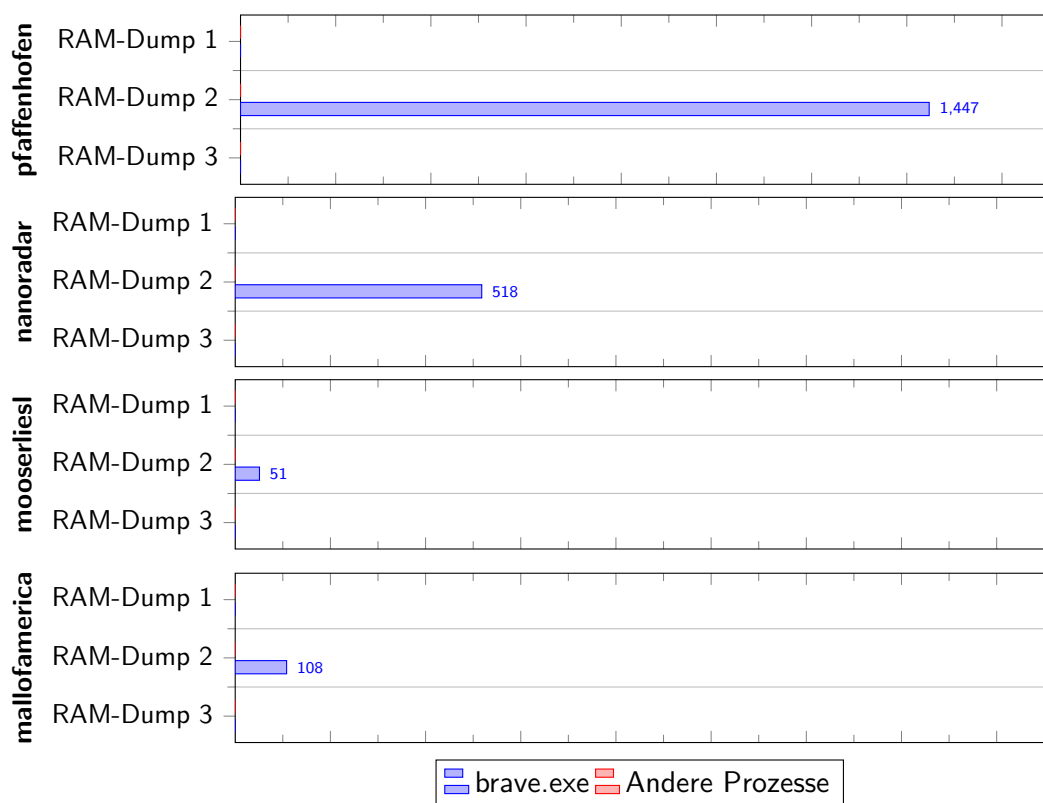


Abbildung 5.21.: Brave: Anzahl gefundener Suchbegriffe im RAM

**Yara-Regel „URLs“** Wie in Abbildung 5.22 gezeigt, konnten alle URLs im RAM nachgewiesen werden. Bei „donaukurier.de“ gab es sogar einen Treffer im dritten RAM-Dump, alle anderen waren nur im zweiten Speicherabbild auffindbar. Darunter waren mit insgesamt 3514 Artefakten am meisten Donaukurier-URLs, wobei sechs davon in anderen Prozessen neben Brave gefunden wurden. Sehr präsent war hier der Prozess „NisSrv.exe“. Dieser ist ein Teil des Microsoft Defenders [46]. Dies lässt sich evtl. dadurch erklären, dass dieser im Hintergrund den Datenverkehr mitliest und prüft, ob bösartige Software unsicheren Netzwerkverkehr tätigt.

**Yara-Regel „E-Mail“** Wie in Abbildung 5.24 dargestellt, wurden im zweiten als auch im dritten Arbeitsspeicherabbild Artefakte bzgl. der Yara-Regel „E-Mail“ gefunden. Davon waren die meisten bei der E-Mail-Adresse „computerforensikvl@gmail.com“ mit insgesamt 134 Suchtreffern vorhanden. Neun dieser waren aus anderen Prozessen, die restlichen 125 waren direkt in einem Speicherbereich des Chrome-Browsers zu finden. Bei dieser wurde auch noch ein Artefakt im dritten RAM-Dump gefunden im Speicherbereich des dwm.exe Prozesses, welcher zuvor bereits angesprochen wurde. Am wenigsten Suchtreffer hab es bei dem Betrefftext.

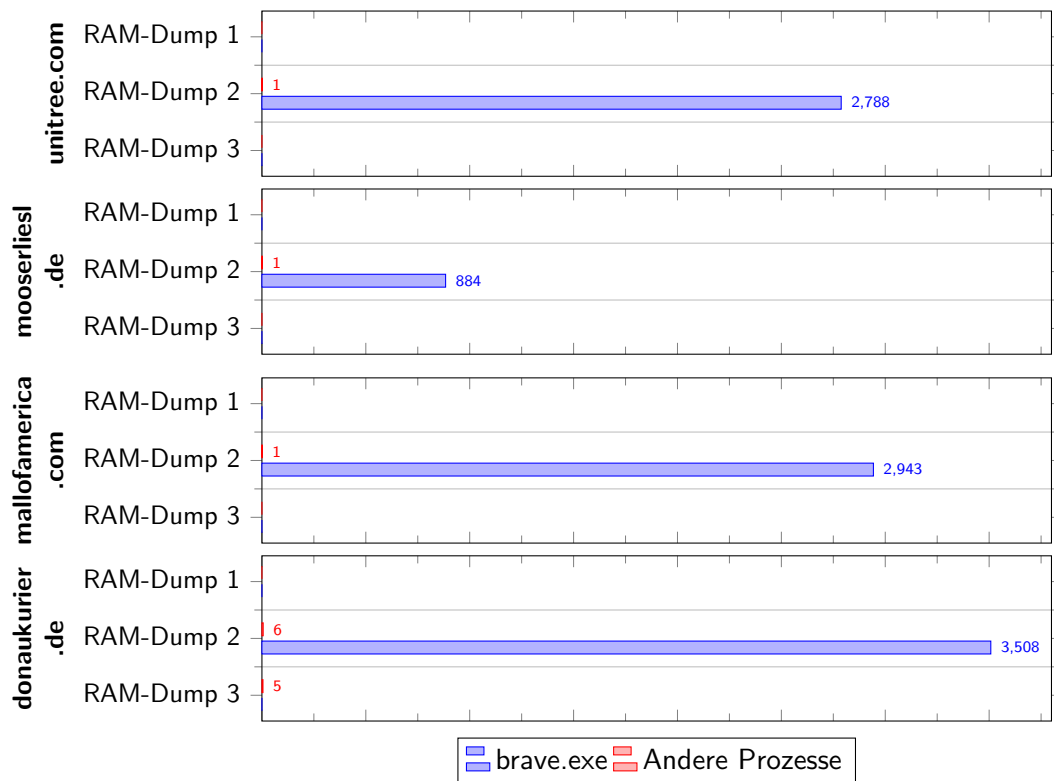


Abbildung 5.22.: Brave: Anzahl gefundener URLs im RAM

**Yara-Regel „DK-Logo“** Auch bei Brave wurde wieder in den Arbeitsspeicherabbildern nach den Bytes des Donaukurier-Logos gesucht. Hier kam es zu drei Treffern im zweiten RAM-Dump, was in der Abbildung 5.23 zu erkennen ist.

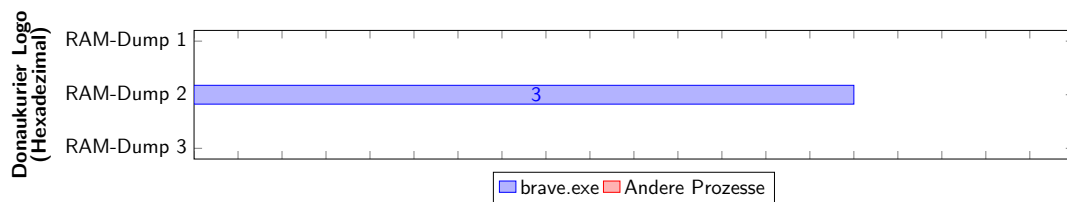


Abbildung 5.23.: Brave: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im RAM

## Registry

Bei der Registry konnten weder in den Process Monitor Logs noch durch die Analyse der verschiedenen Hives Artefakte gefunden werden. Weitergehende Informationen zu den

Schreiboperationen und der Stringsuche in den verschiedenen Hives ist in Anhang E.3 zu finden.

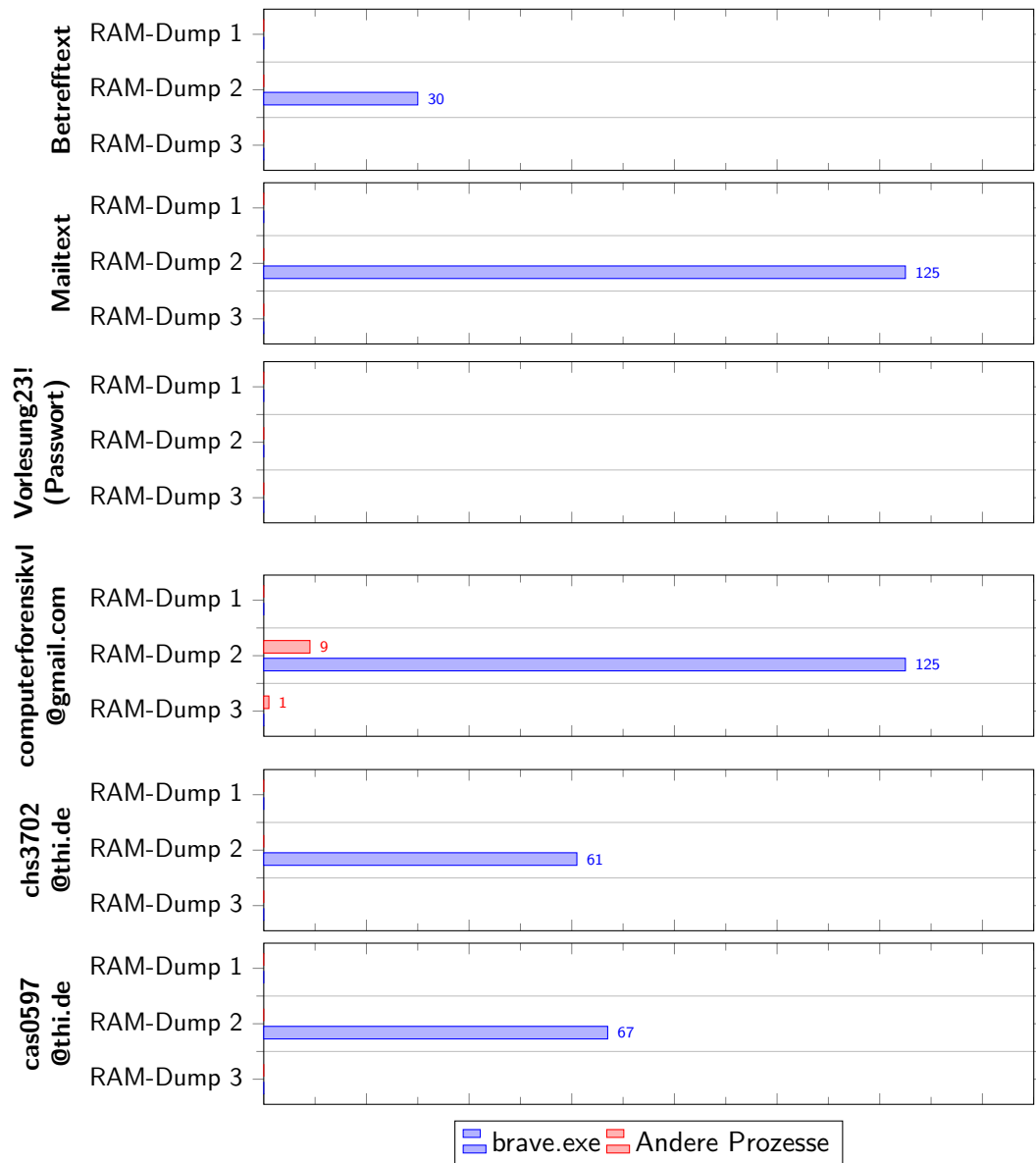


Abbildung 5.24.: Brave: Anzahl gefundener E-Mail Artefakte im RAM

## 6. Vergleich der Browser

In diesem Kapitel werden die untersuchten Browser hinsichtlich ihrer hinterlassenen PB-Artefakte verglichen.

### Common Locations

Bei keinem Browser konnten PB-Artefakte über die Analyse der Datei-Schreiboperationen in den Process Monitor Logfiles gefunden werden. Ebenso konnten durch die genaue Untersuchung der Entwicklung der SQLite-Datenbanken aller Browser keine PB-Artefakte identifiziert werden. Somit sind die Common Locations aller Browser während der gesamten Versuchsdurchführung frei von PB-Artefakten.

### Registry

Bei Betrachtung der Registry als Common Locations wurden keine PB-Artefakte in den Registry-Key bzw. -Values der Registry-Schreiboperationen der Process Monitor Logfiles gefunden. Unter Betrachtung der Registry als Uncommon Locations konnten weder in System- noch User-Hives PB-Artefakte gefunden werden. Somit befinden sich bei jedem Browser auch in der Registry zu keinem Zeitpunkt PB-Artefakte.

### Uncommon Locations

Weder über Autopsy Stichwortsuchen noch in den automatisch von Autopsy kategorisierten Dateien konnten keine PB-Artefakte identifiziert werden.

Einzig über die Untersuchung der Arbeitsspeicherabbilder mit Volatility konnten PB-Artefakte zu unterschiedlichen Zeitpunkten der Versuchsdurchführung identifiziert werden. Wie in Abbildung 6.1 dargestellt, konnten in keinem der Browser vor Durchführung des Browsing-Szenarios (RAM-Dump 1) PB-Artefakte im Arbeitsspeicher gefunden werden.

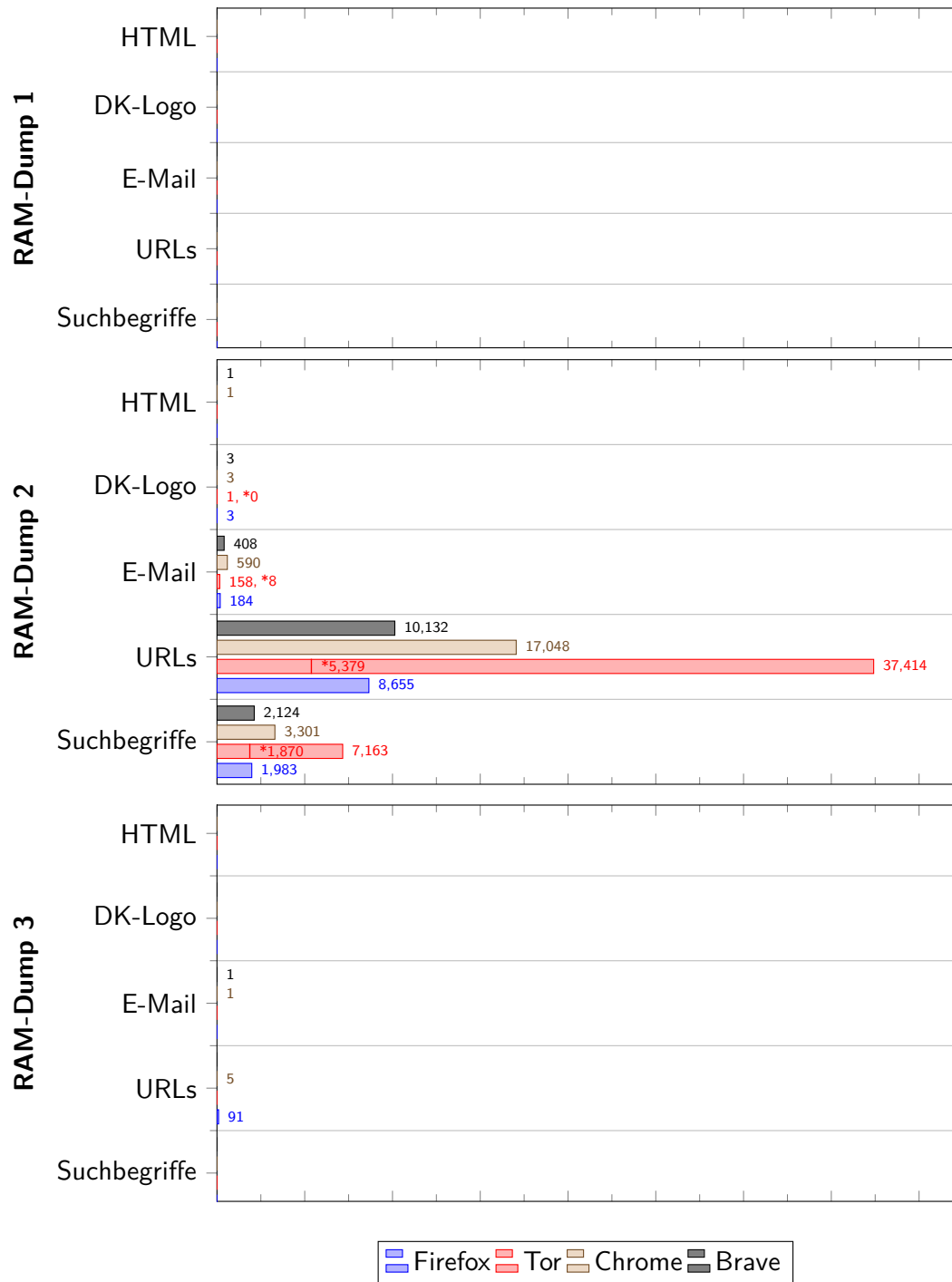


Abbildung 6.1.: Zusammenfassung gefundener Artefakte im RAM der Browser

**Firefox vs Tor** Tor hinterlässt nach dem Browsing-Szenario, vor Zuweisung einer „Neuen Identität“ mehr URL-Artefakte und Suchbegriffe im RAM als Firefox (RAM-Dump 2). Sowohl bei Tor als auch bei Firefox ist zu diesem Zeitpunkt das Gmail-Passwort als Klartext im RAM identifizierbar. Wie anhand der roten, mit \* gekennzeichneten Werte in Abbildung 6.1 zu erkennen ist, reduzieren sich die hinterlassenen Artefakte nach Zuweisung einer „Neuen Identität“ beim Tor-Browser deutlich, sodass danach weniger Artefakte als bei Firefox vorhanden sind. Wie in Abschnitt 4.1.1 erklärt, ermöglicht die „Neue Identität“ das Schließen von Tabs und Fenstern, das Löschen von privaten Informationen und die Neukonfiguration der Tor-Netzwerkverbindung. Mit der „Neuen Identität“ ist bei Tor das Passwort nicht mehr als Klartext im Arbeitsspeicher zu identifizieren. Nach Schließen des Browsers (RAM-Dump 3) hinterlässt der Tor Browser keine Artefakte, bei Firefox konnten noch 91 URLs im DNSCache gefunden werden. Keiner der beiden Browser hinterließ zu irgendeinem Zeitpunkt HTML-Artefakte.

**Chrome vs Brave** Chrome und Brave hinterlassen jeweils nur im zweiten und dritten RAM-Dump PB-Artefakte. Beide Browser hinterlassen in zweiten RAM-Dump gleich viele Artefakte in den Kategorien HTML (eins), DK-Logo (drei) sowie im dritten RAM-Dump in der Kategorie E-Mail (eins). Bei den Kategorien E-Mail, URLs sowie bei den Suchbegriffen hinterließ Brave dabei jeweils weniger Artefakte als der Browser Chrome. Bei keinem der beiden Browser konnte zu irgendeinem Zeitpunkt ein Passwort-Artefakt im RAM festgestellt werden.

**Firefox vs Chrome** Firefox hinterlässt nach dem Browsing-Szenario mit geöffnetem Browser (RAM-Dump 2) in jeder Kategorie gleich viele (DK-Logo) oder weniger (E-Mail, URLs, Suchbegriffe, HTML) Artefakte als Chrome. Dabei werden bei Firefox im zweiten RAM-Dump durchschnittlich 2.024 Artefakte weniger gefunden als bei Chrome. Nach Schließen des Browsers (RAM Dump 3) hinterlässt Firefox mehr URLs im Arbeitsspeicher als Chrome. Im RAM-Dump von Chrome wurde zusätzlich die Absender-Adresse gefunden.

**Tor vs Brave** Vor Zuweisung der „Neuen Identität“ hinterlässt der Tor-Browser nach dem Browsing-Szenario deutlich mehr URL- und Suchbegriff-Artefakte als Brave (RAM-Dump 2). Insbesondere hinterlässt Tor zweimal das Passwort des Google-Accounts als Klartext im RAM. Nach Zuweisung der „Neuen Identität“ des Tor-Browsers hinterlässt dieser durchschnittlich 1315 Artefakte weniger als Brave. Nach Schließen des Browsers (RAM-Dump 3) hinterlässt Tor kein Artefakt im RAM, Brave einmal die Absender-Mail.

**Quantitativer Vergleich aller Browser** Um unter den vier untersuchten Browsern denjenigen mit den wenigsten PB-Artefakten zu ermitteln, wird eine in Tabelle 6.1 gezeigte, sogenannte *Gewinner-Tabelle* erstellt [20].

Tabelle 6.1.: Gewinner-Tabelle der vier untersuchten Browser

	Suchbegriffe	URLs	E-Mail	DK-Logo	HTML	Gewinner pro RAM-Dump
RAM-Dump 1	Unentschieden	Unentschieden	Unentschieden	Unentschieden	Unentschieden	Unentschieden
RAM-Dump 2 (ohne "Neuer ID")	Firefox	Firefox	Brave*	Tor	Firefox, Tor	Firefox
RAM-Dump 2 (mit "Neuer ID")	Tor	Tor	Tor	Tor	Firefox, Tor	Tor
RAM-Dump 3	Unentschieden	Tor, Brave	Firefox, Tor	Unentschieden	Unentschieden	Tor
Gewinner pro Kategorie (ohne "Neuer ID")	Firefox	Firefox, Tor, Brave	Firefox, Tor, Brave*	Tor	Firefox, Tor	
Gewinner pro Kategorie (mit "Neuer ID")	Tor	Tor	Tor	Tor	Firefox, Tor	

Dabei liegt der Fokus auf den Kategorien der PB-Artefakte: Es wird für jede RAM-Dump/Kategorie Kombination derjenige Browser mit den wenigsten PB-Artefakten ermittelt. Somit wird die Anzahl gefundener Artefakte nur innerhalb einer Kategorie verglichen. Die Differenz gefundener PB-Artefakte wird dabei nicht berücksichtigt.

Daraus ergeben sich zwei Arten von *Gewinnern*:

- **Gewinner pro RAM-Dump:** Browser, der innerhalb eines RAM-Dumps am häufigsten die wenigsten PB-Artefakte einer Kategorie hinterlässt. (Zeilenweiser Mehrheitsentscheid)
- **Gewinner pro Kategorie:** Browser, der innerhalb einer Kategorie am häufigsten die wenigsten PB-Artefakte in den RAM-Dumps hinterlässt. (Spaltenweiser Mehrheitsentscheid)

Um ein differenziertes Ergebnis zu ermitteln, wird bei RAM-Dump 2 zusätzlich unterschieden, ob der Tor-Browser vor- oder nach der Zuweisung der „Neuen Identität“, kurz „Neue ID“, bewertet wird. Diese Unterscheidung findet ebenfalls bei den Gewinnern statt. Die in Tabelle 6.2 ermittelten vier Gewinner ergeben sich per Mehrheitsentscheid innerhalb der jeweiligen Gewinner-Kategorie.

Tabelle 6.2.: Ermittelte Gewinner gemäß Gewinner-Tabelle

	Mit „Neuer Identität“	Ohne „Neuer Identität“
Gewinner pro RAM-Dump	Tor	Firefox, Tor
Gewinner pro Kategorie	Tor	Firefox, Tor

Unter Berücksichtigung der „Neuen Identität“ ist der Tor-Browser sowohl der Gewinner pro RAM-Dump als auch pro Kategorie. Ohne Berücksichtigung ist neben dem Tor-Browser auch Firefox Gewinner pro RAM-Dump und pro Kategorie. Somit ist gemäß Auswertung der Gewinner-Tabelle der Tor-Browser derjenige Browser mit den wenigsten PB-Artefakten, gefolgt von Mozilla Firefox.



## 7. Zusammenfassung und Diskussion

Zusammenfassend liefert jeder der vier betrachteten Browser ein - für den Nutzer des privaten Modus - gutes Ergebnis, da keine Private-Browsing-Artefakte in den Festplatten-Abbildern identifiziert werden konnten. Ausschließlich im Arbeitsspeicher konnten PB-Artefakte gefunden werden.

Solange der Browser noch mit allen besuchten Seiten geöffnet ist, sind in einem realistischen Szenario keine forensischen Werkzeuge nötig, um den Besuch dieser Seiten nachzuweisen. Somit ist für Forensiker insbesondere der RAM-Dump nach dem Schließen des Browsers von besonderer Relevanz. Wie in Kapitel 6 gezeigt, wurden dort kaum PB-Artefakte gefunden. Teilweise konnten die zu diesem Zeitpunkt gefundenen Artefakte beseitigt werden. Beispielsweise wurden bei Firefox aufgerufene URLs im DNS-Cache gefunden. DNS-Anfragen werden vom Betriebssystem gespeichert, unabhängig davon, ob der private Modus aktiviert ist oder nicht [51]. Diese Einträge können per Kommandozeilenbefehl geleert werden. Es existieren bereits Browser-Erweiterungen, um das Speichern von DNS-Anfragen zu verhindern [51].

Wie in Kapitel 6 übersichtlich dargelegt wurde, hinterlässt der Tor-Browser am wenigsten Artefakte auf dem lokalen Computer. Zusätzlich zum Schutz vor einem Local Attacker bietet Tor den Vorteil, dass der Datenverkehr durch das Tor-Netzwerks vor dem Web Attacker geschützt ist.

Hier muss jedoch beachtet werden, dass Tor in dieser Arbeit nur quantitativ gesehen der beste Browser ist. Bei der Ermittlung des besten Browsers müssen stets individuelle qualitative Präferenzen berücksichtigt werden. Beispielsweise hinterlässt der Tor-Browser zwar weniger E-Mail-Artefakte, speichert jedoch im Gegensatz zu Google Chrome und Brave das Passwort als Klartext im RAM. Umgekehrt hinterlässt der Tor-Browser keine HTML-Artefakte. Ist für einen Nutzer das Auffinden des Passworts schwerwiegender als hinterlassene HTML-Artefakte, wären Chrome und Brave „besser“ beziehungsweise „sicherer“.

Weiterhin unterscheidet sich die Nutzererfahrung der Browser deutlich. Zwar hinterlässt der Tor-Browser am wenigsten PB-Artefakte, ist jedoch aufgrund des Tor-Netzwerks oft langsam und erfordert häufig das Lösen von Captchas, was sich negativ auf die Nutzererfahrung auswirkt. Im Gegensatz dazu legt beispielsweise der Brave-Browser besonderen Wert auf die Nutzerfreundlichkeit, beispielsweise durch einen integrierten Ad-Blocker. Weiterhin kann entschieden werden, ob der private Modus mit oder ohne Verbindung zum Tor-Netzwerk aktiviert werden soll. [4]

## 8. Fazit und Ausblick

Im Rahmen dieser Arbeit wurden vier verschiedene Browser im privaten Modus auf hinterlassene Browsing-Artefakte untersucht. Ein transparenter Ansatz wurde verfolgt, bei dem ein Browsing-Szenario definiert und erwartete Artefakte festgelegt wurden. Vor, während und nach dem Szenario wurden die Aktivitäten der Browser aufgezeichnet sowie Festplatten- und Arbeitsspeicherabbilder erstellt. Die Artefakte wurden an gängigen sowie ungewöhnlichen Speicherorten, wie dem Arbeitsspeicher analysiert. Ein quantitativer Vergleich der Browser wurde durchgeführt, wobei der Tor-Browser die geringste Menge an Artefakten hinterließ. Qualitative Kriterien wie die Nutzererfahrung wurden nicht berücksichtigt.

Ausschließlich im Arbeitsspeicher wurden Spuren des Browsers im privaten Modus gefunden. Somit verhindert der private Modus zwar das Speichern von Browser-Cache und -Verlauf, aber bei einer forensischen Analyse des laufenden Systems können dennoch eindeutig zuordenbare Artefakte gefunden werden.

Die identifizierten Schwachstellen stellen Browser-Entwickler vor einem Dilemma. Von diesen Ergebnissen profitieren einerseits Browser-Nutzer, da Browser-Entwickler auf Basis der Ergebnisse die aufgedeckten Schwächen beseitigen können. Allerdings besteht auch das Risiko des Missbrauchs des privaten Browsers für illegale Aktivitäten. Das Schließen von Sicherheitslücken erschwert es somit Forensikern, kriminelle Aktivitäten aufzudecken und nachzuweisen. Andererseits profitieren Forensiker selbst von solchen Analysen, um bei identifizierten Schwachstellen gezielt nach Artefakten zu suchen.

Zukünftige Arbeiten könnten den Umfang dieser Arbeit um folgende Punkte erweitern, welche aufgrund von Zeitbeschränkungen und begrenztem Umfang nicht möglich war. Statt nach Prozessnamen zu filtern, könnten Process Monitor Logfiles nach den bekannten Speicherorten durchsucht werden, um festzustellen, ob andere Prozesse dort Schreibaktivitäten ausführen. Eine detailliertere Untersuchung weiterer Betriebssystem-Speicherorte wie des DNS-Caches könnte vorgenommen werden. Es wäre wichtig, eine Analyse von Browsern für Mac oder Linux durchzuführen, da bisherige Literatur nur Windows betrachtet. Außerdem könnte der private Modus direkt mit dem normalen Browsing-Modus verglichen werden, um Unterschiede festzustellen. Schließlich wäre es interessant, weitere gängige Browser wie Microsoft Edge oder Safari einer forensischen Analyse zu unterziehen.

Diese Arbeit hat den persönlichen Wissensstand der Autoren erweitert, indem sie sich intensiv mit forensischer Analyse, Tools, Browsern, virtuellen Maschinen und dem Verhalten des Windows-Betriebssystems auseinandergesetzt haben. Die Arbeit mit Speicherabbildern verdeutlichte den umfangreichen und zeitintensiven Aufwand solcher Untersuchungen. Insgesamt war es eine spannende und anspruchsvolle Erfahrung.

# Anhänge

## A. Yara-Regeln

```
rule keyword {
  strings:
    $pfaffenhofen_keyword="pfaffenhofen" wide ascii nocase
    $nanoradar_keyword="nanoradar" wide ascii nocase
  condition:
    $pfaffenhofen_keyword or $nanoradar_keyword
}

rule keyword_mooserliesl {
  strings:
    $mooserliesl1_keyword="mooserliesl" wide ascii nocase
    $mooserliesl1_keyword2="mooserliesl.de" wide ascii nocase
  condition:
    $mooserliesl1_keyword and not $mooserliesl1_keyword2
}

rule keyword_mallofamerica {
  strings:
    $mallofamerica1_keyword="mallofamerica" wide ascii nocase
    $mallofamerica1_keyword2="mallofamerica.com" wide ascii nocase
  condition:
    $mallofamerica1_keyword and not $mallofamerica1_keyword2
}

rule url {
  strings:
    $mallofamerica_url="mallofamerica.com" wide ascii nocase
    $mooserliesl_url="mooserliesl.de" wide ascii nocase
    $unitree_url="unitree.com" wide ascii nocase
    $donaukurier_url="donaukurier.de" wide ascii nocase
  condition:
    $mallofamerica_url or $mooserliesl_url or $unitree_url or $donaukurier_url
}

rule html {
  strings:
    $mallofamerica_html="Insiders</span>" wide ascii nocase
```

```
    $mooserliesl_html="Ja</span>" wide ascii nocase
    $unitree_html="L1</div>" wide ascii nocase
    $donaukurier_html=">Themen:" wide ascii nocase
  condition:
    $mallofamerica_html or $mooserliesl_html or $unitree_html
    or $donaukurier_html
}

rule image {
  strings:
    $image_hex = {
      89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 00 00 01 2C 00 00
      00 32 08 03 00 00 00 D1 08 16 18 00 00 00 19 74 45 58 74 53 6F 66
      74 77 61 72 65 00 41 64 6F 62 65 20 49 6D 61 67 65 52 65 61 64 79
      71 C9 65 3C 00 00 03 84 69 54 58 74 58 4D 4C 3A 63 6F 6D 2E 61 64
      6F 62 65 2E 78 6D 70 00 00 00 00 00 00 3C 3F 78 70 61 63 6B 65 74 20
      ...
      4E AD 38 61 03 55 6A AB 5E BF F6 40 4E 9D BA 20 FE 43 FE 99 81 2C
      0A 8F B2 F1 D8 7B DE E5 75 7E 45 E3 FC E4 C8 81 E5 C0 72 60 39 B0
      1C 71 60 39 B0 1C 58 0E 2C 07 D6 FF A9 FC 57 80 01 00 D9 B2 CD 5E
      42 B8 37 25 00 00 00 00 49 45 4E 44 AE 42 60 82}
  condition:
    $image_hex
}

rule mail {
  strings:
    $computerforensik_address="computerforensikvl@gmail.com" wide ascii nocase
    $computerforensik_password="Vorlesung23!" wide ascii nocase
    $cas0597_address="cas0597@thi.de" wide ascii nocase
    $chs3702_address="chs3702@thi.de" wide ascii nocase
    $subject="Betrefftext" wide ascii nocase
    $mail_body="Mailinhalt" wide ascii nocase
  condition:
    $computerforensik_address or $computerforensik_password
    or $cas0597_address or $chs3702_address or $subject
    or $mail_body
}
```

---

## B. Ausführliche Analyse: Firefox

### B.1. Common Locations

#### Process Monitor WriteFile Operations

Gemäß Versuchsdurchführung wurden für Firefox mit dem Process Monitor Tool zwei Logfiles erstellt. Diese Dateien enthalten alle aufgezeichneten Prozessaktivitäten während und nach dem Browsing-Szenario. Zunächst wurden beide Logfiles gemäß Methodik in Abschnitt 4.2 in Excel aufbereitet. Tabelle B.1 listet alle in den gefilterten Logfiles identifizierten Dateien auf. Dabei wurde für jede Datei vermerkt, ob und wie sie wiederherstellbar war, mit welchem Tool die Datei analysiert wurde und ob in der Datei PB-Artefakte enthalten sind. Die wiederherstellbaren Dateien wurden in die Kategorien *Cache*, *Datareporting*, *Sessionstore-Backup* und *Sonstige Dateien* eingeordnet. In keiner der Dateien wurden PB-Artefakte identifiziert.

Bei detaillierter Untersuchung der wiederherstellbaren Dateien konnten zwei Pfade identifiziert werden, in die Firefox während des Versuchs Dateien schreibt:

- **Local:** C:\Users\<User>\AppData\Local\Mozilla\Firefox\Profiles\  
<Profile>.default-release\
- **Roaming:** C:\Users\<User>\AppData\Roaming\Mozilla\Firefox\Profiles\  
<Profile>.default-release\

In Tabelle B.1 sind die Dateien je nach Speicherort *Local* (Hellblau) oder *Roaming* (Dunkelblau) entsprechend eingefärbt. Im Speicherort Local sind ausschließlich Dateien der Kategorie „Cache“ gespeichert.

**Cache** Firefox verwendet den Cache, um Webseiten und deren Ressourcen temporär lokal zu speichern. Dadurch können wiederholte Anfragen an den Server vermieden und die Ladezeiten verringert werden. Die Inhalte dieser Dateien sind binär. Die Cache-Dateien im Format \cache2\entries\<ID> werden im Local Pfad gespeichert.

Diese Dateien können mit dem Tool MZCacheView eingelesen werden. Wie in Abbildung B.1 gezeigt, konnten im Firefox Cache-Ordner des Festplatten-Images vom zweiten Snapshot drei JSON Dateien identifiziert werden. Dabei handelt es sich um Zertifikatsdateien, die von der *One Certificate Revocation List* stammen, einem Mechanismus von Firefox zur Überprüfung von Zertifikaten. In keinem der Zertifikate konnten mit HxD Private-Browsing-Artefakte oder besuchte Seiten gefunden werden. [56]



Tabelle B.1.: Firefox alle „WriteFile“-Operationen der Logfiles 1 und 2

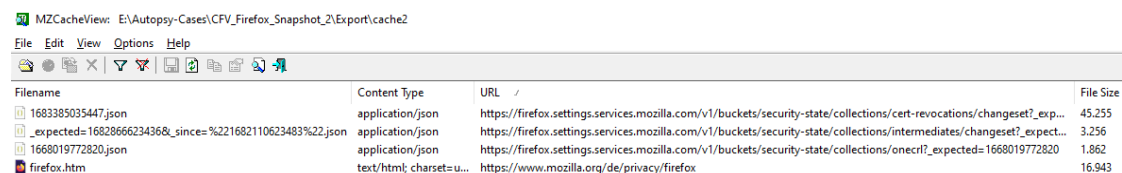
## LOGFILE 1:

Kategorie	Dateiname	Dateistatus	Tool für Analyse	Enthaltene Artefakte
Cache	\cache2\entries\037778A55E1B7E9BED3390289866D09402D6C913	Datei vorhanden	MZCacheView	Keine PB-Artefakte
	\cache2\entries\1223A0378B8971FA4CD25EA1731C80B2B1676B42	Datei vorhanden	MZCacheView	Keine PB-Artefakte
	\cache2\entries\250EE2BC03AFF526F1A1C3DB212A79DE3EB60D5E	Datei vorhanden	MZCacheView	Keine PB-Artefakte
	\jumpListCache\ZKJGVJPzPe7w4w0KwEY0jw==.ico	Datei vorhanden	Windows Foto App	Keine PB-Artefakte
	\cache2\entries\D16E4E5DFB15B4C8DE8842C05A47A07C611E01D	Datei vorhanden	MZCacheView	Keine PB-Artefakte
	\cache2\entries\2F040683A85A4372A73572713C6C52B510854566	Datei vorhanden	MZCacheView	Keine PB-Artefakte
Datareporting	\datareporting\glean\events\pageload	Datei vorhanden	HxD	Keine PB-Artefakte
	\datareporting\glean\db\data.safe.tmp	Nicht-temp-Datei verwendet	HxD	Keine PB-Artefakte
	\datareporting\glean\tmp\95ea3e10-e732-4642-8e92-515f4c4e090c	Datei nicht wiederherstellbar	N/A	N/A
	\datareporting\glean\tmp\16ab2ae2-c7b8-4390-a26e-7dcd95f5ff24	Datei nicht wiederherstellbar	N/A	N/A
Sessionstore	\sessionstore-backups\recovery.jsonlz4.tmp	Nicht-temp-Datei verwendet	dejsonlz4 + Notepad++	Keine PB-Artefakte
Sonstige Dateien	\prefs-1.js	Datei vorhanden	HxD	Keine PB-Artefakte
	\xulstore.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB-Artefakte

## LOGFILE 2:

Kategorie	Dateiname	Dateistatus	Tool für Analyse	Enthaltene Artefakte
Cache	\cache2\index.log	Datei vorhanden	MZCacheView	Keine PB-Artefakte
	\cache2\index	Datei vorhanden	MZCacheView	Keine PB-Artefakte
Datareporting	\datareporting\glean\db\data.safe.tmp	Nicht-temp-Datei verwendet	HxD	Keine PB-Artefakte
	\datareporting\archived\2023-05\1683405837882.9102466b-e465-4ecb-810f-74ae90c64c63.new-profile.jsonlz4.tmp	Nicht-temp-Datei verwendet	Session History Scrounger	Keine PB-Artefakte
	\datareporting\archived\2023-05\1683405837905.86f4c992-6329-415b-8c29-911a2d4b7f9d.event.jsonlz4.tmp	Nicht-temp-Datei verwendet	Session History Scrounger	N/A
	\datareporting\archived\2023-05\1683405837939.abf8b065-41a4-4e94-a044-1cead61e396a.main.jsonlz4.tmp	Nicht-temp-Datei verwendet	Session History Scrounger	N/A
Sessionstore	\sessionstore.jsonlz4.tmp	Nicht-temp-Datei verwendet	dejsonlz4 + Notepad++	Keine PB-Artefakte
Sonstige Dateien	\sessionCheckpoints.json.tmp	Nicht-temp-Datei verwendet	HxD	Keine PB-Artefakte
	\prefs-1.js	Datei vorhanden	HxD	Keine PB-Artefakte
	\xulstore.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB-Artefakte
	\saved-telemetry-pings\9102466b-e465-4ecb-810f-74ae90c64c63.tmp	Datei nicht wiederherstellbar	N/A	N/A
	\saved-telemetry-pings\86f4c992-6329-415b-8c29-911a2d4b7f9d.tmp	Datei nicht wiederherstellbar	N/A	N/A
	\saved-telemetry-pings\abf8b065-41a4-4e94-a044-1cead61e396a.tmp	Datei nicht wiederherstellbar	N/A	N/A
	\saved-telemetry-pings\a35decee-d7c6-4820-a381-2dc89ff33c76.tmp	Datei nicht wiederherstellbar	N/A	N/A

Weiterhin befindet sich im Cache das HTML-Dokument der Firefox Datenschutzseite, welche sich beim ersten Start des Browsers automatisch öffnete, siehe Abschnitt 4.1.2. Weitere Cache Dateien konnten in keinem Festplatten-Image gefunden werden. Die Indexdatei



Filename	Content Type	URL	File Size
1683385035447.json	application/json	https://firefox.settings.services.mozilla.com/v1/buckets/security-state/collections/cert-revocations/changeset?_exp...	45.255
_expected=1682866623436&_since=%221682110623483%22.json	application/json	https://firefox.settings.services.mozilla.com/v1/buckets/security-state/collections/intermediates/changeset?_expect...	3.256
1668019772820.json	application/json	https://firefox.settings.services.mozilla.com/v1/buckets/security-state/collections/onecl?_expected=1668019772820	1.862
firefox.htm	text/html; charset=u...	https://www.mozilla.org/de/privacy/firefox	16.943

Abbildung B.1.: Firefox: Mit MZCacheView eingelesene Cache-Dateien

\cache2\index dient als Datenbank im Cache. Sie ermöglicht es dem Firefox-Browser, schnell auf die zwischengespeicherten Ressourcen zuzugreifen und diese effizient zu verwalten. In diese Datei wurde beim Schließen des Browsers geschrieben. Sowohl mit HxD als auch dem Tool FirefoxCache2 konnten keine PB-Artefakte identifiziert werden.

Schließlich enthält die Datei \jumpListCache\ZKJGVJPzPe7w4w0KwEY0jw==.ico ein 64x64 Pixel großes Mozilla Logo. Dieses Logo ist keinem Schritt des Browsing-Szenarios zuzuordnen und ist vermutlich auf die automatisch geöffnete Datenschutzhinweiseite zurückzuführen.

**Datareporting** Dateien im Ordner \datareporting\glean\db sind Teil des Glean-Systems, das für die Sammlung von Telemetriedaten und deren Übermittlung an Mozilla verwendet wird. [12] Die Datei data.safe.bin enthält verschlüsselte und anonyme Informationen über die Nutzung des Browsers. In HxD konnten keine PB-Artefakte in den Dateien gefunden werden.

Dateien im Format \datareporting\glean\db\<Profilname>.new-profile.jsonlz4 speichern Informationen über das Firefox-Profil, das von Glean verwendet wird. In diese Dateien wurde erst nach dem Browsing-Szenario, beim Schließen des Browsers geschrieben. Diese Dateien im proprietären *jsonlz4*-Format lassen sich mit dem Tool *dejsonlz4* dekomprimieren. Die entstandene JSON Datei wurde mit dem Notepad++ JSON Plugin untersucht. Dabei konnten keine PB-Artefakte gefunden werden.

**Sessionstore** Die Datei \sessionstore-backups\recovery.jsonlz4 enthält eine Sicherungskopie der vorherigen Sitzung. Sie wird erstellt, wenn der Firefox-Browser nach einem Absturz oder einem unerwarteten Beenden neu gestartet wird. Jefferson Scher entwickelte das Online-Tool *Session History Scrounger for Firefox* zur Analyse dieser „Sessionstore-Backup“ Dateien. [24] Wie in Abbildung B.2 gezeigt, enthielt die Datei sowohl im Festplatten-Image 2 (Logfile 1) und 3 (Logfile 2) nur die automatisch geöffnete Seite der Firefox Datenschutzhinweise.



## Closed Window 1

### Tab 1

Firefox Datenschutzhinweis — Mozilla [5/6/2023, 10:24:59 PM]  
<https://www.mozilla.org/de/privacy/firefox/>

Copyright © 2020 Jefferson Schier (BSD-3-Clause License). lz4.js © 2016 Pierre Curto (MIT License; Sept. 1, 2016). FileSaver.js © 2016 Eli Grey (MIT License; v1.3.2).

Abbildung B.2.: Firefox: Sitzungsdatei recovery.jsonlz4, geöffnet mit dem „Session History Scrounger for Firefox“

**Sonstige Dateien** In der Datei prefs-1.js werden benutzerspezifische Einstellungen und Konfigurationen für den Firefox-Browser gespeichert. Die Datei enthält Präferenzen des Benutzers in Form von JavaScript-Objekten. Es konnten in den Dateien beider Logfiles mit HxD keine PB-Artefakte gefunden werden. Schließlich speichert die Datei xulstore.json benutzerspezifische Anpassungen und Konfigurationen des Firefox-Browsers. In der Datei konnten in den Festplatten-Images beider Logfiles mit Notepad++ keine PB-Artefakte gefunden werden. [37]

## SQLite-Datenbanken

Wie in Abschnitt 4.3.1 erwähnt, werden SQLite-Datenbanken als Datenstrukturen für Nutzerdaten detailliert und getrennt von den Schreiboperationen der Process Logfiles untersucht. Mithilfe der Process Monitor Logfiles wurden zunächst die in Tabelle B.2 dargestellten SQLite-Datenbanken für Firefox identifiziert:

Tabelle B.2.: Firefox: Veränderte SQLite-Datenbanken und deren Verwendungszwecke

Datenbank	Gespeicherte Daten [10]
places.sqlite	Informationen über Lesezeichen und Verlauf Zu jeder besuchten Webseite: URL, Seitentitel, Zeitstempel des Besuchs etc.
cookies.sqlite	Von besuchten Webseiten verwendete Cookies.
storage.sqlite	Diverse Webdaten, z. B. Indexed-Datenbanken, Offline-Cache-Daten und andere lokale Speicherinformationen
favicons.sqlite	Enthält Favicons (kleine Symbole in der Adressleiste) um besuchte Webseiten visuell zu identifizieren
webappsstore.sqlite	Speichert Informationen über installierte Webanwendungen im Firefox-Browser, z.B. Berechtigungen und Einstellungen
1657114595AmcateirvtiSty.sqlite	Datenspeicher für Activity Stream, eine personalisierte Übersicht über Browser-Aktivitäten beim Öffnen eines neuen Tabs
3870112724rsegmnoittet-es.sqlite	Datenspeicher für Remote Settings, eine zentrale Verwaltung von benutzerspezifischen Browsereinstellungen

Entsprechend der Methodik in Abschnitt 4.3.1 wurde jede SQLite-Datenbank aus den Festplatten-Images aller vier Snapshots extrahiert und verglichen. Die Ergebnisse sind in Tabelle B.3 dargestellt.

Tabelle B.3.: Veränderung der Firefox SQLite-Datenbanken während der Versuchsdurchführung

Dateiname	Vor Browsing Szenario (S1)	Nach Browsing Szenario, Browser geöffnet (S2)		Nach Browsing Szenario, Browser geschlossen (S3)		VM heruntergefahren (S4)	
		Vor WAL	Nach WAL	Vor WAL	Nach WAL	Vor WAL	Nach WAL
places.sqlite	N/A	Initialisiert (Datenschutz-Seite)	keine Veränderung	Indizes aktualisiert	keine Veränderung	keine Veränderung	keine Veränderung
cookies.sqlite	N/A	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung			
storage.sqlite	N/A	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung			
favicons.sqlite	N/A	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung			
webappsstore.sqlite	N/A	N/A	N/A	Initialisiert (Nur Spaltennamen)			
formhistory.sqlite	N/A	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung			
1657114595AmcateirvtiSty.sqlite	N/A	Initialisiert (origin: "chrome")	keine Veränderung	Binärdaten, keine PB Artefakte			
3870112724rsegmnoittet-es.sqlite	N/A	Initialisiert (origin: "chrome")	keine Veränderung	keine Veränderung			

Unmittelbar nach der Installation von Firefox (Snapshot 1) existierte noch keine der SQLite-Dateien.

Nach dem Browsing-Szenario (Snapshot 2) wurden alle SQLite-Datenbanken außer `webappsstore.sqlite` initialisiert. Dabei wurden in `places.sqlite` die automatisch im normalen Modus geöffnete Firefoxseite der Datenschutzhinweise eingetragen. Die restlichen Datenbanken wurden leer initialisiert, nur die Spaltennamen wurden definiert. Der Inhalt aller initialisierten Datenbanken blieb nach Durchführung von PRAGMA WAL Checkpoints unverändert.

Nach Schließen des Browsers (Snapshot 3) wurden in `places.sqlite` die Indizes der eingetragenen Seiten aktualisiert. Die SQLite-Datenbank `1657114595AmcateirvtiSty.sqlite` erhielt ein binäres Datenobjekt als Eintrag. Bei der Untersuchung mit HxD konnten keine Artefakte gefunden werden. Weiterhin wurde `webappsstore.sqlite` leer initialisiert. Die restlichen Daten blieben im Vergleich mit Snapshot 2 unverändert. Ebenfalls veränderte sich nicht der Inhalt nach Durchführung von PRAGMA WAL Checkpoints.

Weder nach dem Herunterfahren der VM (Snapshot 4), noch nach Durchführung der PRAGMA WAL Checkpoints, entstanden Änderungen in den SQLite-Datenbanken. Somit wurden in den SQLite-Datenbanken von Firefox keine zurückverfolgbaren PB-Artefakte im privaten Modus hinterlassen.

## Zusammenfassung Firefox Common Locations

Mithilfe der Process Monitor Logfiles wurde festgestellt, dass sowohl während des Browsing-Szenarios (Logfile 1) als auch danach (Logfile 2) Inhalte in Dateien geschrieben wurden. Wie in Abbildung B.3 dargestellt, gab es mit Ausnahme der *Datareporting* Dateien in Logfile 1 stets mehr oder gleich viele Schreiboperationen wie in Logfile 2. Keine der Schreiboperation hinterließ Private-Browsing-Artefakte.

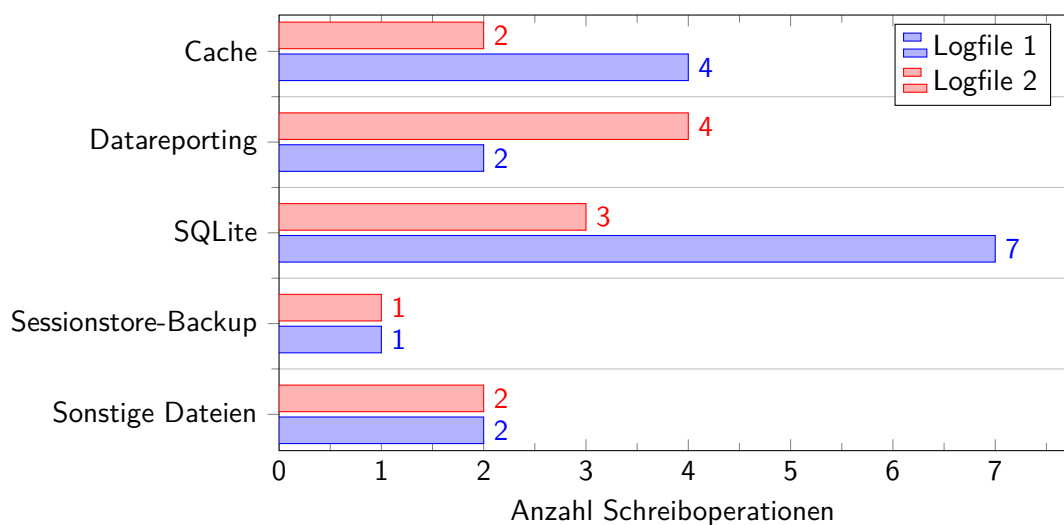


Abbildung B.3.: Firefox: Anzahl Schreiboperationen Logfile 1 vs Logfile 2, geordnet nach Kategorie

## B.2. Uncommon Locations

### Analyse mit Autopsy - Kategorisierte Dateien

Beim Vergleich der Festplattenabbilder wurde festgestellt, dass ein Festplatten-Image stets die kategorisierten Dateien des Festplatten-Images des vorherigen Snapshots enthält. Somit enthält das Festplatten-Image von Snapshot 4 alle kategorisierten Dateien der vorherigen Snapshots.

**Web Bookmarks** Bereits vor Durchführung des Browsing-Szenarios enthielt Firefox im ersten Snapshot die in Abbildung B.4 dargestellte Bing Startseite als gespeichertes Lesezeichen. In den restlichen Snapshots 2 bis 4 blieb diese Kategorie unverändert.

Source Name	S	C	O	URL	Title	Date Created	Program Name	Domain
Bing.url			19	http://go.microsoft.com/fwlink/p/?LinkId=255142	Bing.url	2023-04-25 16:09:28 MESZ	Internet Explorer Analyzer	microsoft.com

Abbildung B.4.: Firefox: Von Autopsy als „Web Bookmarks“ kategorisierte Dateien

**Web Cookies** Die Kategorie „Web Cookies“ enthält bereits vor Beginn des Browsing-Szenarios zehn in Abbildung B.5 gezeigte Cookie-Einträge in der Datei WebCacheV01.dat. Dabei handelt es sich um eine Datenbank des Microsoft Edge Browsers zur Speicherung von Nutzerdaten. Diese Datei verhält sich ähnlich wie die in diesem Versuch relevanten SQLite-Dateien. Bei den Einträgen handelt es sich um Cookies für Bing und die Outlook

Webseite, obwohl diese Seiten nie in Microsoft Edge geöffnet wurden. In den Snapshots 2 bis 4 kamen keine weiteren Einträge in dieser Kategorie hinzu.

Source Name	S	C	O	URL	Date Accessed	Name	Value	Program Name	Domain	Data Source
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:17 MESZ	SUID	M	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	www.bing.com	2023-05-06 19:51:24 MESZ	MUID8	31708C5FC3CF47068AFADC1CB47D0111	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:17 MESZ	SRCHD	AF=NOFORM	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:17 MESZ	SRCHUID	V=28&UID=B2C50ADBCE984234A9FE14DB81DC891D&dm...	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:17 MESZ	SRCHUSR	DOB=20230506	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:20 MESZ	SRCHPGUSR	SRCHLANG=de&LUT=16834026192238JPMH=des20405&...	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:19 MESZ	CortanaAppUID	C164AA3A4D47E127DDC66AD915CFD04C	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:55:22 MESZ	AIKON	A=A3B5B679A14D59B0AA02763FFFFFFFFFF	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	live.com	2023-05-06 19:50:30 MESZ	MUID	118A534093A9626528C5404997A96688	Microsoft Edge Analyzer	live.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	login.live.com	2023-05-06 19:51:06 MESZ	Host-MSAATHP		Microsoft Edge Analyzer	live.com	CPV_Firefox_Klon_Snapshot_3.img

Abbildung B.5.: Firefox: In Autopsy als „Web Cookies“ kategorisierte Dateien

**Web History** Die Kategorie „Web History“ listet alle Dateien mit gespeichertem Suchverlauf auf. Vor Beginn des Browsing-Szenarios (Snapshot 1) enthält die Kategorie zwei Einträge zur Outlook Webseite in der Datei WebCacheV01.dat. Nach Durchführung des Browsing-Szenarios (Snapshot 2) wurde ein Eintrag in der places.sqlite Datenbank hinzugefügt. Dabei handelt es sich um die automatisch im normalen Browsingmodus geöffnete Firefox-Standardseite über Datenschutzhinweise. Dies deckt sich mit den Beobachtungen der Common Locations in Anhang B.1. Darüber hinaus enthält dieser Snapshot für die Datei WebCacheV01.dat den Eintrag file:///Z:/Logfile\_1. Dabei handelt es sich um das Process Monitor Logfile, das gemäß Methodik in Abschnitt 4.1 über den gemeinsamen VM-Ordner zum Analyse-Rechner transportiert wurde. Ergänzt wird die Kategorie nach Schließen des Browsers (Snapshot 3) durch den Eintrag file:///Z:/Logfile\_2, dem zweiten Process Monitor Logfile. Nach Herunterfahren der virtuellen Maschine (Snapshot 4) werden in dieser Kategorie keine neuen Dateien erfasst. Die kategorisierten Dateien sind in Abbildung B.6 dargestellt.

Source Name	S	C	O	URL	Date Accessed	Referrer URL	Title	Program Name	Domain	Data Source	Username
places.sqlite			6	https://www.mozilla.org/de/privacy/ffrefox/	2023-05-06 22:25:00 MESZ	https://www.mozilla.org/privacy/ffrefox/	Firefox Datenschutzhinweis — Mozilla	Firefox Analyzer	mozilla.org	CPV_Firefox_Klon_Snapshot_3.img	
WebCacheV01.dat			15	https://login.live.com/oauth20_desktop.aspx?ik=1031	2023-05-06 19:51:06 MESZ			Microsoft Edge Analyzer	live.com	CPV_Firefox_Klon_Snapshot_3.img	Forensik
WebCacheV01.dat			15	https://login.live.com/oauth20_authorize.srf?client_j...	2023-05-06 19:51:08 MESZ			Microsoft Edge Analyzer	live.com	CPV_Firefox_Klon_Snapshot_3.img	Forensik
WebCacheV01.dat				file:///Z:/Logfile_1	2023-05-06 20:29:36 MESZ			Microsoft Edge Analyzer		CPV_Firefox_Klon_Snapshot_3.img	Forensik
WebCacheV01.dat				file:///Z:/Logfile_2	2023-05-06 20:44:19 MESZ			Microsoft Edge Analyzer		CPV_Firefox_Klon_Snapshot_3.img	Forensik

Abbildung B.6.: Firefox: In Autopsy als „Web History“ kategorisierte Dateien

**Web Categories** Diese Kategorie klassifiziert im Speicherabbild gefundene Browsing-Artefakte nach Inhalt. Vor Beginn des Browsing-Szenarios (Snapshot 1) werden hier bereits zwei in Abbildung B.7 dargestellte Einträge in der Datei WebCacheV01.dat aufgelistet. Der Eintrag bing.com wird als „Suchmaschine“ klassifiziert und live.com als „Web-Email“. Wie oben erwähnt, wurden beide Seiten nie aufgerufen. Es gab keine zusätzlichen Einträge in dieser Kategorie während des restlichen Browsing-Szenarios (Snapshots 2 bis 4).

Somit wurden in allen Kategorien ausschließlich Browsing-Artefakte des Edge Browsers in der Datei WebCacheV01.dat gefunden, sowie ein Eintrag in der Firefox SQLite-Datenbank

Source Name	△ S	C	O	Source Type	Score	...	Domain	Host	Name	File Path
WebCacheV01.dat			0	File	Unknown		bing.com	bing.com	Search Engine	/img_CPV_Firefox_klon_Snapshot_3.img/vol_vo3/Users/Forensik/AppData/Local/Microsoft/Windows/WebCache/WebCacheV01.dat
WebCacheV01.dat			0	File	Unknown		live.com	login.live.com	Web Email	/img_CPV_Firefox_klon_Snapshot_3.img/vol_vo3/Users/Forensik/AppData/Local/Microsoft/Windows/WebCache/WebCacheV01.dat

Abbildung B.7.: Firefox: In Autopsy als „Web Categories“ kategorisierte Dateien

places.sqlite. In keiner der Kategorien konnten Private-Browsing-Artefakte identifiziert werden. Die von Autopsy erkannte Firefox-Standardseite deckt sich mit den Ergebnissen der Common Locations. Die aufgelisteten Einträge in der Datei WebCacheV01.dat sind nicht auf Schritte des Browsing-Szenarios zurückzuführen. Die Einträge sind bereits im ersten Snapshot enthalten, obwohl vor Beginn des Browsing-Szenarios keine Browseraktivitäten durchgeführt wurden. Weiterhin enthält diese Datei Einträge über die Process Monitor Logfiles, welche über einen gemeinsamen VM-Ordner zum Rechner transportiert wurde, auf dem die virtuelle Maschine läuft.

### B.3. Registry

#### Process Monitor SetValue Operations

Als Teil der Common Locations werden für Firefox alle Registry „SetValue“ Schreiboperationen der beiden Process Monitor Logfiles untersucht.

In beiden Logfiles wurden zwei Kategorien von Registry Keys geschrieben: *PreXULSkeletonUISettings* und *Business Activity Monitoring*. In Abbildung B.8 ist der Anteil der Schreiboperationen je Kategorie für beide Logfiles gezeigt.

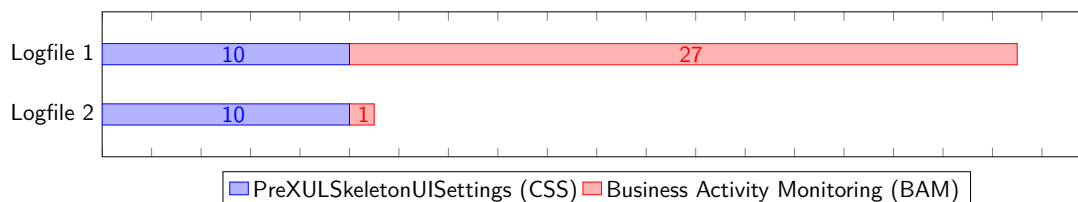


Abbildung B.8.: Firefox: Registry „SetValue“ Operationen in den Process Monitor Logfiles 1 und 2

**PreXULSkeletonUISettings** Der *PreXULSkeletonUISettings* Registry Key enthält Einstellungen für die Benutzeroberfläche (UI) des Firefox-Browsers, insbesondere für das sogenannte *Skeleton UI*, eine vereinfachte Benutzeroberfläche, die während des Ladens des Browsers angezeigt wird, bevor die vollständige Benutzeroberfläche geladen ist. PreXULSkeletonUISettings Registry Keys haben das Format HKCU\SOFTWARE\Mozilla\Firefox\PreXULSkeletonUISettings\<Absoluter Firefox Installationspfad>\firefox.exe

---

|<Skeleton UI Setting>. Somit enthält der Key den absoluten Installationspfad von Firefox gefolgt von einer Skeleton UI Einstellung. Nachfolgend sind alle möglichen UI Einstellungen aufgelistet, gefolgt vom Datentyp des Keys. [32]

- ScreenX (DWORD)
- ScreenY (DWORD)
- Width (DWORD)
- Height (DWORD)
- Maximized (DWORD)
- Flags (DWORD)
- CssToDevPixelScaling (REG\_BINARY)
- UrlbarCSSSpan (REG\_BINARY)
- SearchbarCSSSpan (REG\_BINARY)
- SpringsCSSSpan (REG\_BINARY)

Somit enthalten die Keys nur Daten zur Formatierung und Struktur der grafischen Oberfläche. Es wurden keine PB-Artefakte geschrieben.

**Business Activity Monitoring** *Business Activity Monitoring*, kurz *BAM* ist eine weitgehend undokumentierte Windows Funktion, die im Hintergrund ausgeführte Programme steuert. Der Registry Key hat das Format HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\<SID>\Device\HarddiskVolume2\<Absoluter Firefox Installationspfad>\firefox.exe und den Datentyp REG\_BINARY. Jeder Schlüssel wird durch die Sicherheits-ID (SID) des Benutzers identifiziert. Ein BAM Registry Key schreibt für alle ausgeführten Programme – hier Firefox – den Zeitstempel der letzten Ausführung. PB-Artefakte sind dabei nicht enthalten. [22, 26]

### Stringsuche in Registry Hives

Gemäß Methodik in Abschnitt 4.3.3 wird die Firefox Registry als Uncommon Location behandelt, indem über alle auf der Festplatte vorhandenen Registry Datenbanken, den Registry Hives, eine Stringsuche durchgeführt wird, ohne die Struktur der Hives zu beachten. Dazu wurden sowohl die System-Hives als auch die User-Hives aus Tabelle 4.4 aus jedem Festplatten-Image extrahiert und mithilfe des Registry Explorers nach PB-Artefakten durchsucht. Dabei wurde in keinem Snapshot in keinem Hive ein PB-Artefakt gefunden.

---

## C. Ausführliche Analyse: Tor

### C.1. Common Locations

#### Process Monitor WriteFile Operations

Bei der Versuchsdurchführung für den Tor-Browser gemäß Abschnitt 4.2 wurden drei Process Monitor Logfiles erstellt. Diese Dateien enthalten alle aufgezeichneten Prozessaktivitäten während des Browsing-Szenarios, dem Erzeugen einer „Neuen Identität“ sowie des Schließens des Browsers. Tabelle C.1 enthält alle in den Logfiles identifizierten Dateien. Für jede Datei wurde vermerkt ob und wie sie wiederherstellbar war, mit welchem Tool die Datei analysiert wurde und ob PB-Artefakte enthalten sind. Die Dateien wurden in die Kategorien *Cache*, *datareporting*, und *Sonstige Dateien* eingeordnet. In keiner der identifizierten Dateien konnten PB-Artefakte gefunden werden.

Bei Analyse der Schreiboperationen konnten zwei Datei-Pfade identifiziert werden:

- **Caches:** C:\Users\Forensik\Desktop\Tor Browser\Browser\TorBrowser\Data\Browser\Caches\profile.default\
- **Profile.default:** C:\Users\Forensik\Desktop\Tor Browser\Browser\TorBrowser\Data\Browser\profile.default\

In der Tabelle sind die Dateien je nach Speicherort *Caches* (Hellblau) oder *Profile.default* (Dunkelblau) eingefärbt.

Bei der Auswertung der Process Monitor Logfiles wurde festgestellt, dass alle Schreiboperationen von „firefox.exe“-Prozessen und nicht „tor.exe“-Prozessen durchgeführt wurden. Obwohl keine der Dateien PB-Artefakte enthält, werden zum vollständigen Verständnis des Browsers im Sinne der White-Box-Forensik die wichtigsten Dateien im Zusammenhang des Tor-Browsers genauer untersucht.

**Cache** Der Tor-Browser schreibt eine einzige Cache-Datei \Caches\profile.default\startupCache\startupCache.8.little im Caches-Pfad. Alle anderen geschriebenen Dateien befinden sich im Profile.default-Pfad. Die Datei „startupCache.8.little“ ist eine interne Datei, welche erstellt wird, um den Startvorgang des Browsers zu beschleunigen. Sie enthält Informationen über bereits geladene Browser-Komponenten wie JavaScript-Code, CSS-Dateien, Bilder und andere Ressourcen. [36] Bei Untersuchung mit HxD konnten keine PB-Artefakte gefunden werden.





Tabelle C.1.: Tor-Browser: Alle „WriteFile“-Operationen der Logfiles 1, 2-1 und 2-2

**LOGFILE 1:**

Kategorie	Dateiname	Dateistatus	Verwendetes Tool zur Analyse	Enthaltene Artefakte
Cache	\startupCache\startupCache.8.little	Datei vorhanden	HxD	Keine PB-Artefakte
Datareporting	\datareporting\glean\db\data.safe.bin	Datei vorhanden	HxD	Keine PB-Artefakte
	\datareporting\state.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB-Artefakte
Sonstige Dateien	\addonStartup.json.lz4.tmp	Nicht-temp-Datei verwendet	dejsonlz4, Notepad++	Keine PB-Artefakte
	\AlternateServices.txt	Datei vorhanden	Notepad++	Keine PB-Artefakte
	\broadcast-listeners.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB-Artefakte
	\extensions.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB-Artefakte
	\extensions\staged{73a6fe31-595d-460b-a920-fcc0f8843232}.xpi	Datei vorhanden	HxD	Keine PB-Artefakte
	\onion-aliases.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB-Artefakte
	\prefs-1.js	Datei vorhanden	Notepad++	Keine PB-Artefakte
	\security_state\data.safe.bin	Datei vorhanden	HxD	Keine PB-Artefakte
	\settings\data.safe.bin	Datei vorhanden	HxD	Keine PB-Artefakte
	\SiteSecurityServiceState.txt	Datei vorhanden	Notepad++	Keine PB-Artefakte
	\SiteSecurityServiceState-1.txt	Datei vorhanden	Notepad++	Keine PB-Artefakte
	\xulstore.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB-Artefakte

**LOGFILE 2-1**

Sonstige Dateien	\prefs-1.js	Datei vorhanden	dejsonlz4, Notepad++	Keine PB-Artefakte
	\cert_override.txt	Datei vorhanden	Notepad++	Keine PB-Artefakte
	\enumerate_devices.txt	Datei vorhanden	Notepad++	Keine PB-Artefakte

**LOGFILE 2-2**

Datareporting	\datareporting\glean\db\data.safe.bin	Datei vorhanden	HxD	Keine PB-Artefakte
Sonstige Dateien	\prefs-1.js	Datei vorhanden	Notepad++	Keine PB-Artefakte
	\sessionCheckpoints.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB-Artefakte
	\xulstore.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB-Artefakte

---

**Datareporting** Im „Datareporting“-Ordner wird die Datei `\datareporting\data.safe.bin` geschrieben. Sie enthält verschlüsselte und anonyme *Glean* Informationen über die Nutzung des Browsers. [11] In HxD konnten keine PB-Artefakte gefunden werden. Weiterhin wird die Datei `\datareporting\state.json` geschrieben. Sie enthält Informationen über den Zustand und die Konfiguration des Tor-Browsers, beispielsweise installierte Add-Ons, oder Browser-Einstellungen. Sie wird verwendet, um bei Bedarf den Zustand und die Einstellungen des Browsers wiederherzustellen. [10] Eine Analyse mit Notepad++ und dem JSON-Plugin brachte keine PB-Artefakte.

**Sonstige Dateien** Die im ersten Logfile geschriebene Datei `AlternateServices.txt` enthält .onion-URLs des HTTP Alternative Services. Dieser Mechanismus ermöglicht es Servern, Clients mitzuteilen, dass der Dienst, auf den sie zugreifen, an einem anderen Netzwerkstandort oder über ein anderes Protokoll verfügbar ist. Die Datei speichert diese Zuordnung. [35]

Weiterhin wird während des Browsing-Szenarios die Datei `\extensions\staged\73a6fe31-595d-460b-a920-fcc0f8843232.xpi` geschrieben. Dabei handelt es sich um das von Tor verwendete *NoScript*-AddOn zur selektiven Ausführung von JavaScript Webseiteninhalten. Nach Extraktion dieser Datei, kann diese per Drag-and-Drop in ein geöffnetes Firefox-Fenster gezogen werden und es ist möglich, die Erweiterung zu installieren.

Die geschriebene Datei `onion-aliases.json` enthält *SecureDrop* Adressen, beispielsweise für Medien wie die Süddeutsche Zeitung. *SecureDrop* ist ein Open-Source-Software-Tool, das von Journalisten und Nachrichtenorganisationen verwendet wird, um anonyme Informationen von Whistleblowern entgegenzunehmen. Es ermöglicht den sicheren Austausch von Informationen, ohne die Identität der Quelle preiszugeben. Whistleblower können über .onion-URLs auf die *SecureDrop*-Websites zugreifen und vertrauliche Dokumente oder Nachrichten sicher und anonym übermitteln. [52]

Schließlich wurde in die Datei `SiteSecurityServiceState.txt` geschrieben. Diese Datei speichert Daten wie Zertifikate, Verschlüsselungseinstellungen und andere Sicherheitsmerkmale, die von den besuchten Websites verwendet werden. Es ist anzumerken, dass diese Datei früher Private-Browsing-Artefakte enthielt. [13] In der aktuellen Tor-Browser-Version konnten keine PB-Artefakte gefunden werden.

## SQLite-Datenbanken

Anhand der Process Monitor Logfiles ist erkennbar, dass Tor die gleichen SQLite-Datenbanken wie Firefox verwaltet und schreibt (vgl. Anhang B.1).

Wie bei der Analyse der SQLite-Datenbanken bei Firefox wird die Entwicklung der Datenbankeninhalte aller fünf Festplatten-Images der Snapshots 1, 2, 3-1, 3-2 und 4 betrachtet. Die Ergebnisse sind in Tabelle C.2 dargestellt.

Tabelle C.2.: Tor-Browser: Veränderung der SQLite-Datenbanken während der Versuchsdurchführung

Dateiname	Vor Browsing Szenario (S1)	Nach Browsing Szenario, Browser geöffnet (S2)		Nach Browsing Szenario, Neue Identität (S3-1)		Nach Browsing Szenario, Browser geschlossen (S3-2)		VM heruntergefahren (S4)	
		Vor WAL	Nach WAL	Vor WAL	Nach WAL	Vor WAL	Nach WAL	Vor WAL	Nach WAL
places.sqlite	N/A	Initialisiert (Tor-Standardseiten)	keine Veränderung	Indizes aktualisiert	keine Veränderung	Indizes aktualisiert	keine Veränderung	keine Veränderung	keine Veränderung
cookies.sqlite	N/A	Initialisiert (Nur Spaltennamen)		keine Veränderung		keine Veränderung			
storage.sqlite	N/A	Initialisiert (Nur Spaltennamen)		keine Veränderung		keine Veränderung			
favicons.sqlite	N/A	Initialisiert (Tor-Standardseiten, Präfix: fake-favicon-uri*)		keine Veränderung		Indizes aktualisiert			
webappsstore.sqlite	N/A	Initialisiert (Nur Spaltennamen)		keine Veränderung		keine Veränderung			
formhistory.sqlite	N/A	Initialisiert (Nur Spaltennamen)		keine Veränderung		keine Veränderung			
1657114595AmcateirvtiSty.sqlite	N/A	Initialisiert (origin: "chrome")		keine Veränderung		keine Veränderung			
3870112724rsegmnoittet-es.sqlite	N/A	Initialisiert (origin: "chrome")		keine Veränderung		keine Veränderung			

Nach Browser-Installation wurde noch keine SQLite-Datei angelegt (Snapshot 1).

Während des Browsing-Szenarios wurden alle Datenbanken initialisiert. In places.sqlite wurden automatisch .onion-URLs geschrieben, die zu Tor Standardseiten führen. Beispielsweise Seiten wie „The Tor Blog“ oder „Tor Browser Manual“. Die gleichen Einträge wurden in der favicons.SQLite-Datenbank geschrieben, mit dem Präfix „Fake-favicon-uri“. Ein tatsächliches Icon wurde nicht in die Datenbank geschrieben. Weiterhin erhielt die „remote settings“ Datenbank den gleichen Eintrag wie es bereits bei Firefox der Fall war. Der Eintrag enthält keine PB-Artefakte. Die restliche SQLite-Dateien wurden ohne Inhalt angelegt, nur die Spaltennamen wurden definiert. Nach Durchführung der WAL Checkpoints bleiben Dateien unverändert.

Nachdem dem Tor-Browser eine „Neue Identität“ zugewiesen wurde (Snapshot 3-1), wurden in places.sqlite die Indizes bei den eingetragenen Seiten aktualisiert. Die restlichen Dateien blieben unverändert. Das Schreiben der WAL-Dateien in die Hauptdatenbanken veränderte den Inhalt nicht.

Nach Schließen des Browsers (Snapshot 3-2) wurden in places.sqlite sowie favicons.sqlite erneut Indizes bei eingetragenen Seiten aktualisiert. Die restliche Dateien blieben unverändert, ebenso ergaben die WAL Checkpoints keine Veränderungen.

Nach Herunterfahren der VM (Snapshot 4) blieben alle Datenbanken unverändert. Auch nach Durchführung der WAL Checkpoints gab es keine neuen Inhalte.

## Zusammenfassung Tor Common Locations

Im Balkendiagramm C.1 ist zu erkennen, dass die meisten Schreiboperationen im ersten Logfile stattfinden. Dort werden Dateien jeder Kategorie beschrieben. Das Schließen des Tor-Browsers führt zu mehr oder genauso vielen Schreiboperationen wie das Zuweisen einer „Neuen Identität“. Keine der geschriebenen Dateien enthielt Private-Browsing-Artefakte.

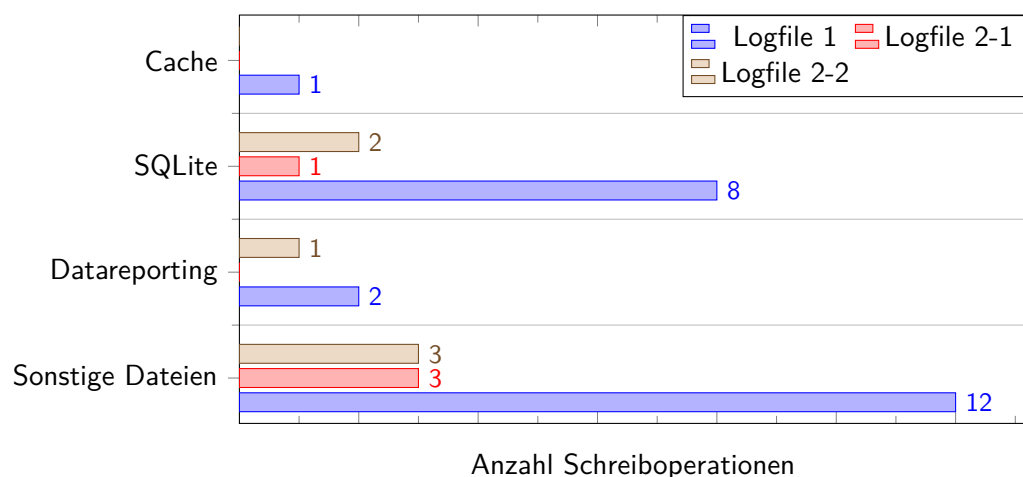


Abbildung C.1.: Tor-Browser: Anzahl Schreiboperationen Logfile 1 vs Logfile 2-1 vs Logfile 2-2, geordnet nach Kategorie

## C.2. Uncommon Locations

### Analyse mit Autopsy - Kategorisierte Dateien

Wie bei Firefox in Anhang B.2 wurde keine der kategorisierten Dateien gelöscht. Somit befanden sich im letzten Festplatten-Image des Snapshots 4 alle kategorisierten Dateien der vorherigen Festplatten-Images

**Web Bookmarks** Wie in Abbildung C.2 gezeigt, wurden nur in der Datei Bing.url ein Lesezeichen zur Bing-Startseite gefunden. Diese Datei wurde im Festplatten-Image des ersten Snapshots geschrieben.

Source Name	S	C	O	URL	Title	Date Created	Program Name	Domain
Bing.url			19	http://go.microsoft.com/fwlink/p/?LinkId=255142	Bing.url	2023-04-25 16:09:28 MESZ	Internet Explorer Analyzer	microsoft.com

Abbildung C.2.: Tor-Browser: In Autopsy als „Web Bookmarks“ kategorisierte Dateien

**Web Cookies** Im Festplatte-Image des ersten VM-Snapshots wurden die in Abbildung C.3 gezeigten neun Cookies-Einträge in die Datenbank des vorinstallierten Edge Browsers geschrieben. Dabei handelt es sich um Cookies für die Bing- und Outlook-Startseite.

**Web History** Zwei Einträge mit Browsingverläufen wurden im Festplatten-Image des ersten VM-Snapshots in der Datei WebCacheV01.dat geschrieben. Wie in Abbildung C.4 gezeigt, handelt es sich dabei zweimal um die Outlook-Startseite, obwohl diese nie bei der

Source Name	S	C	O	URL	▼ Date Accessed	Name	Value	Program Name	Domain
WebCacheV01.dat			15	bing.com	2023-05-07 13:56:11 MESZ	SUID	A	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat			15	bing.com	2023-05-07 13:47:01 MESZ	ANON	A=D8530A977A1F5DAD20B78D8CFFFFFFF	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat			15	login.live.com	2023-05-07 12:26:40 MESZ	_Host-MSAAUTHP		Microsoft Edge Analyzer	live.com
WebCacheV01.dat			15	live.com	2023-05-07 12:25:56 MESZ	MUID	0EDC58E500A76FCE1B0F4BEF04A76BD6	Microsoft Edge Analyzer	live.com
WebCacheV01.dat			15	www.bing.com	2023-05-07 12:25:44 MESZ	MUIDB	31708C5FC3CF47068AFADC1CB47D0111	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat			15	bing.com	2023-05-07 12:25:44 MESZ	SRCHD	AF=NOFORM	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat			15	bing.com	2023-05-07 12:25:44 MESZ	SRCHUID	Y=28GUID=62F5FD78E9D94446AFDF9DEC8188103&dm...	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat			15	bing.com	2023-05-07 12:25:44 MESZ	SRCHUSR	DOB=20230507	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat			15	bing.com	2023-05-07 12:25:44 MESZ	SRCHHPGUSR	SRCHLANG=de	Microsoft Edge Analyzer	bing.com

Abbildung C.3.: Tor-Browser: In Autopsy als „Web Cookies“ kategorisierte Dateien

Versuchsdurchführung geöffnet wurde. Wie bei Firefox wurden in der Datei ebenfalls die zum Analyserechner über den gemeinsamen Ordner transportierten Process Monitor Logfiles gespeichert.

Source Name	S	C	O	URL	▼ Date Accessed	Program Name	Domain	Username
WebCacheV01.dat				file:///Z:/Logfile2-2	2023-05-07 14:28:06 MESZ	Microsoft Edge Analyzer		Forensik
WebCacheV01.dat				file:///Z:/Logfile2-1	2023-05-07 14:17:05 MESZ	Microsoft Edge Analyzer		Forensik
WebCacheV01.dat				file:///Z:/Logfile1	2023-05-07 13:55:54 MESZ	Microsoft Edge Analyzer		Forensik
WebCacheV01.dat			15	https://login.live.com/oauth20_authorize.srf?client_id=000...	2023-05-07 12:26:43 MESZ	Microsoft Edge Analyzer	live.com	Forensik
WebCacheV01.dat			15	https://login.live.com/oauth20_desktop.srf?lc=1031	2023-05-07 12:26:40 MESZ	Microsoft Edge Analyzer	live.com	Forensik

Abbildung C.4.: Tor-Browser: In Autopsy als „Web History“ kategorisierte Dateien

**Web Categories** Autopsy klassifizierte im Festplatten-Image des ersten VM-Snapshots den Eintrag bing.com als „Suchmaschine“ und live.com als „Web-Email“, gezeigt in Abbildung C.5. Es gab keine zusätzlichen Einträge in dieser Kategorie in den Festplatten-Images der restlichen Snapshots.

Source Name	△ S	C	O	Source Type	Score	Conclusion	Configuration	Justification	Domain	Host	Name
WebCacheV01.dat			0	File	Unknown				bing.com	bing.com	Search Engine
WebCacheV01.dat			0	File	Unknown				live.com	login.live.com	Web Email

Abbildung C.5.: Tor-Browser: In Autopsy als „Web Categories“ kategorisierte Dateien

Somit wurden in den von Autopsy kategorisierten Dateien keine PB-Artefakte entdeckt. Weiterhin gab es verglichen mit der Analyse der Common Locations keine neuen Erkenntnisse. Autopsy erkannte nicht die in der places.sqlite-Datenbank geschriebenen .onion-URLs der Tor-Standardseiten.

### C.3. Registry

#### Process Monitor SetValue Operations

Bei Betrachtung als Common Locations werden gemäß Methodik in Abschnitt 4.3.3 alle „SetValue“ Schreiboperationen in den Process Monitor Logfiles für die Prozesse „tor.exe“ und

---

„firefox.exe“ untersucht.

Dabei wurden die gleichen beiden Registry Keys identifiziert, wie bei der Untersuchung der Firefox Registry in Anhang B.3: PreXULSkeletonUISettings und Business Activity Monitoring. In keinem Registry-Key befinden sich PB-Artefakte. Wie in Abbildung C.6 dargestellt, wurden beide Registry Keys annähernd gleich oft geschrieben. Bei Vergleich der drei Process Monitor Logfiles 1, 2 und 3 nimmt die Anzahl der Registry „SetValue“-Operationen bei Logfile 2 und 3 kontinuierlich ab.

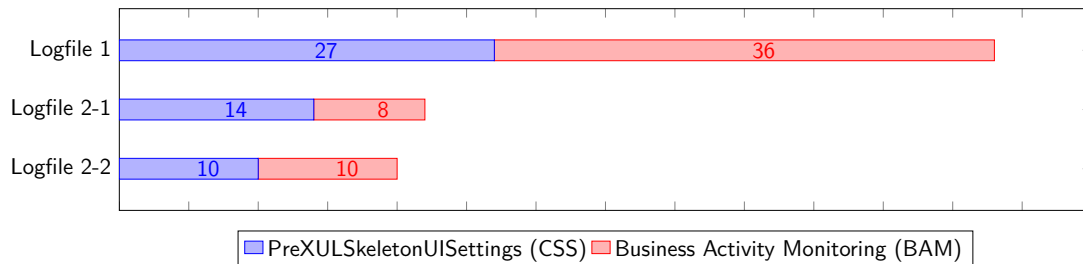


Abbildung C.6.: Tor-Browser: Registry „SetValue“ Operationen in den Process Monitor Logfiles 1, 2-1 und 2-2

### Stringsuche in Registry Hives

Bei Betrachtung der Registry als Uncommon Locations, wurden die in Tabelle 4.4 in Abschnitt 4.3.3 aufgelisteten Registry Hives mithilfe des Registry Explorers untersucht. Weder in den System-Hives noch in den User-Hives konnten PB-Artefakte identifiziert werden.

---

## D. Ausführliche Analyse: Chrome

### D.1. Common Locations

#### Process Monitor WriteFile Operations

Gemäß Versuchsdurchführung wurden auch für Chrome 2 Logfiles erstellt, welche die Prozessaktivitäten während (Logfile 1) und nach dem Browsing-Szenario während des Schließens des Browsers (Logfile 2) aufzeichnet. Beide Logfiles wurden gemäß Methodik in Unterabschnitt 4.3.1 in Excel aufbereitet. Tabelle D.1 listet dabei alle Dateien auf, welche während des Szenarios durchgeführt wurden, Tabelle D.2 hingegen alle Schreiboperationen beim Schließen des Browsers. Dabei wurde für jede Datei wieder vermerkt, ob und wie sie wiederherstellbar war, mit welchem Tool die Datei analysiert wurde und ob in der Datei PB-Artefakte enthalten sind. Die Schreiboperationen wurden dieses mal in Dateiendungen gegliedert. In keiner der Dateien wurden PB-Artefakte identifiziert.

Dabei wurde in Tabelle D.1 unterschieden zwischen den **Local**-Browser Pfaden (hellblau) und **andere Pfade** (dunkelblau).

Zu sehen ist hier, dass hauptsächlich Dateien in den Browser-Pfad geschrieben werden.

Alle *.tmp*-Dateien waren nicht wiederherstellbar, was eine Analyse unmöglich macht. Die Dateien mit dem Namen *data\_1* waren jeweils vorhanden, enthielten jedoch keine PB-Artefakte nach ausführlicher Analyse. Die *-journal*-Dateien Diese sind sogenannte „Rollback Journals“ und sind relevant für atomare Commit- und Rollback-Funktionen [53]. Da diese Dateien jedoch alle 0 Bytes groß waren, konnte hier kein Artefakt gefunden werden.

Auch wurden noch *000003.log*-Dateien geschrieben. Diese sind Teil eines LevelDB-Schlüsselwertespeichers [14, 18] . Mithilfe von HxD und einer Analyse mittels eines selbst erstellten Python-Scripts zur Ausgabe der Schlüssel-Werte-Paare derjenigen Ordner, welche die *.log* und weitere LevelDB spezifische Dateien enthalten, kann auch in diesen Ordner bzw. Dateien auch keine Artefakte gefunden werden.





Tabelle D.1.: Chrome alle "WriteFile"-Operationen des Logfiles 1

LOGFILE 1:				
Kategorie	Dateiname	Dateistatus	Verwendetes Tool zur Analyse	Enthaltene Artefakte
Temp files (.tmp)	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\1f7ba833-406a-40cf-b107-6e391f4bd1d3.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\2da074d0-9208-4026-b970-d7261bd389c3.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\44a1b7b5-40eb-4265-8d3e-b55d21084e65.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\6f9e2d84-9a77-41e3-8955-b0c836f8fd0c.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\8b2096fb-9e68-4a4c-9df5-3dd0949aa210.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\97615fa9-9081-43b0-af51-534da2fd8cb4.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\aoea17f1-38e8-48e0-a2f4-98e9be6a6dd3.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\b23e8f25-4bfb-4625-a9d5-836ff096b671.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\c029e5f2-88df-4271-bc24-2c50db41cc89.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\9a105eba-925a-4d38-994f-c59962a8a60c.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Network\184cc287-bc19-4faf-bd09-fdfc1ff1c6b8.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Network\3ab3887-9ed6-45e7-a1bc-e0a34974b332.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\e5e0606b-51a1-44ba-a8f9-80f1cf5c48a3.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\fb23442-8e9b-47cb-95e6-9da65df2c42e.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\fecad46f-9d32-40a2-aa3c-7b1cc275a5e2.tmp	Datei nicht wiederherstellbar	N/A	N/A
data_1 files	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Cache\Cache_Data\data_1	Datei vorhanden	ChromeCacheView	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\DawnCache\data_1	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\GPUCache\data_1	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\GrShaderCache\data_1	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\ShaderCache\data_1	Datei vorhanden	HxD	Keine PB-Artefakte
SQLite -journal Dateien	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\History-journal	Datei leer (0 Bytes groß)	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Network\Reporting and NEL-journal	Datei leer (0 Bytes groß)	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Web Data-journal	Datei leer (0 Bytes groß)	N/A	N/A
000003.log Dateien (LevelDB)	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\shared_proto_db\000003.log	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Sync Data\LevelDB\000003.log	Datei vorhanden	HxD	Keine PB-Artefakte
Temporary .png files	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Web Applications\Temp\scoped_dir764_530297989\icons\128.png	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Web Applications\Temp\scoped_dir764_530297989\icons\192.png	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Web Applications\Temp\scoped_dir764_530297989\icons\256.png	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Web Applications\Temp\scoped_dir764_530297989\icons\32.png	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Web Applications\Temp\scoped_dir764_530297989\icons\48.png	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Web Applications\Temp\scoped_dir764_530297989\icons\512.png	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Web Applications\Temp\scoped_dir764_530297989\icons\64.png	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Web Applications\Temp\scoped_dir764_530297989\icons\96.png	Datei nicht wiederherstellbar	N/A	N/A
Spelling default files	C:\Users\Forensik\AppData\Roaming\Microsoft\Spelling\de-DE\default.acl	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Roaming\Microsoft\Spelling\de-DE\default.dic	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Roaming\Microsoft\Spelling\de-DE\default.exc	Datei vorhanden	HxD	Keine PB-Artefakte
D3DS Cache	C:\Users\Forensik\AppData\Local\D3DSCache\cb00da9ba77862e\F4EB2D6C-ED2B-4BDD-AD9D-F913287E6768.idx	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\D3DSCache\cb00da9ba77862e\F4EB2D6C-ED2B-4BDD-AD9D-F913287E6768.lock	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\D3DSCache\cb00da9ba77862e\F4EB2D6C-ED2B-4BDD-AD9D-F913287E6768.val	Datei vorhanden	HxD	Keine PB-Artefakte

Tabelle D.2.: Chrome: alle "WriteFile"-Operationen des Logfiles 2

LOGFILE 2:					
Kategorie	Dateiname	Dateistatus	Verwendetes Tool zur Analyse	Enthaltene Artefakte	
<i>settings.dat</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Crashpad\settings.dat	Datei vorhanden	HxD	Keine PB-Artefakte	
<i>Temp files (.tmp)</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\35deb2e-9a97-4829-b0d1-2c6efb7246bc.tmp	Datei nicht wiederherstellbar	N/A	N/A	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\4dce7d9d-2753-424d-ad13-eb84e1ea9326.tmp	Datei nicht wiederherstellbar	N/A	N/A	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\51ff0ac1-e188-4c8a-8b3d-891f326bb890.tmp	Datei nicht wiederherstellbar	N/A	N/A	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\52837c44-01bc-43d1-b859-0fe50c823372.tmp	Datei nicht wiederherstellbar	N/A	N/A	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Storage\ext\nmmhkkccagldldgiimedpiccmgmieda\def\Network\c280cbe5-825f-482f-8c5f-e4b0f0e8d560.tmp	Datei vorhanden	HxD	Keine PB-Artefakte	
<i>SQLite -journal Dateien</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\History-journal	Datei leer (0 Bytes groß)	N/A	N/A	
<i>JSON-Datei</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Variations	Datei vorhanden	HxD	Keine PB-Artefakte	
<i>000003.log Datei (LevelDB)</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Session Storage\000003.log	Datei vorhanden	HxD	Keine PB-Artefakte	
<i>data_1 files</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Cache\Cache_Data\data_1	Datei vorhanden	HxD	Keine PB-Artefakte	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\GrShaderCache\data_1	Datei vorhanden	HxD	Keine PB-Artefakte	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\ShaderCache\data_1	Datei vorhanden	HxD	Keine PB-Artefakte	
<i>shutdown_ms.txt</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\chrome_shutdown_ms.txt	Datei vorhanden	HxD	Keine PB-Artefakte	

Zuletzt wurden auch noch Dateien geschrieben, welche dem Shader-Cache zuzuordnen sind. Diese enthielten jedoch nach genauerer Analyse auch keine Informationen über die PB Session.

Die Datei *settings.dat* im Crashpad Ordner ist Teil der Crashpad library, welche den Maschinen- und Programmzustand im Falle eines Prozessabsturzes erfasst und einen Absturzbericht an einen Backend-Server übermittelt [7]. Die (binäre) Datei an sich beinhaltet dabei die Einstellungen für die Bibliothek [7]. Diese Datei beinhaltet jedoch keine Browsing-Artefakte. Weiterhin wurden wieder temporäre Dateien geschrieben, welche bis auf eine Datei nicht rekonstruierbar sind. Diese Eine enthält jedoch keine Artefakte.

Die *-journal*-Datei zeigte auch wieder keine Artefakte. Die Datei *Variations* direkt im Google\Chrome\User Data\\_-Ordner ist der Dateistruktur nach eine JSON-Datei. Diese enthält aber auch keinerlei Informationen über das durchgeführte Browsing-Szenario.

Gleiches gilt für die geschriebenen *data 1*-Dateien.

Zuletzt wurde noch eine Textdatei namens *chrome\_shotdown\_ms.txt* geschrieben. Diese enthält die Zeit in Millisekunden, welche Chrome für das Schießen benötigt [6]. Dort fanden sich neben der Zeit in Millisekunden keine weiteren Artefakte.

Es wurden mittels des Tools ChromeCacheView der Cache-Ordner von Chrome analysiert. Exemplarisch dafür zeigt Abbildung D.1 die Analyse des dritten Snapshots.

[illegible]

75

Zu sehen ist hier, dass keine Cache-Dateien auf eine von uns besuchte Website zurückzuführen ist. Auch eine detaillierte Analyse mittels verschiedener Tools lieferte hier keine Artefakte. Dafür wurden alle Dateien extrahiert und nach Strings bzw. Byte-Sequenzen durchsucht. Abbildung D.2 zeigt einen Ausschnitt aus VS Code, worin eine css-Datei analysiert wird.



Abbildung D.2.: Chrome: Analyse eines Cache-files

## SQLite-Datenbanken

Wie in Unterabschnitt 4.3.1 beschrieben, werden SQLite-Datenbanken als Datenstrukturen für Nutzerdaten detailliert und getrennt von den Schreiboperationen der Process Logfiles untersucht. Bei Chrome gibt es im Vergleich zu Firefox und Tor den Unterschied, dass die Datenbanken nicht direkt als .sqlite Dateien erkennbar sind, sondern als Dateien ohne bestimmte Endung im Browser-Verzeichnis vorliegen.

Auch hier wurden alle relevanten Datenbanken aus den Festplatten-Speicherabbildern extrahiert (welche vorhanden waren) und diese anschließend im Hinblick auf deren Inhalte, Tabellen und Schreiboperationen verglichen und analysiert. Tabelle D.3 zeigt die Ergebnisse übersichtlich.

Erkennbar ist, dass im Gegensatz zu Firefox und Tor bereits einige Datenbanken bereits vor dem Browsing-Szenario initialisiert wurden. Das kam dadurch zustande, dass sich Chrome direkt nach der Installation geöffnet hat, was nicht verhindert werden konnte. Trotz einer offline executable, welche extra für den Zweck heruntergeladen wurde, dass Chrome keine Artefakte bereits vor dem Browsing-Szenario anlegt, konnte jenes Starten nicht verhindert werden.

Zu sehen ist außerdem, dass trotz einiger Schreiboperationen auf die Datenbanken keine

Tabelle D.3.: Chrome: Veränderung der SQLite-Datenbanken während der Versuchsdurchführung

Dateiname	Vor Browsing Szenario (S1)	Nach Browsing-Szenario, Browser geöffnet (S2)	Nach Browsing-Szenario, Browser geschlossen (S3)	VM heruntergefahren (S4)
History	Initialisiert (Nur Spaltennamen)	keine Veränderung	Datensätze hinzugekommen, keine Artefakte	Datensatz gelöscht
Web Data	Initialisiert (Nur Spaltennamen)	Inhalte verändert, keine PB-Artefakte	keine Veränderung	Datensätze hinzugekommen, keine Artefakte
Shortcuts	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung
Top Sites	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung
Login Data	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung
Network Action Predictor	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung
Reporting and NEL	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung
Cookies	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung

Artefakte gefunden werden konnten. Eine der wichtigsten Datenbank „History“ zeigte keine Datensätze, obwohl genau in diese Informationen zu besuchten Websites und Login-Zeitpunkte gespeichert werden. Dies wurde auch exemplarisch anhand einer durchgeführten Beispiels-public-Session nachgewiesen. Zusätzlich wurde das Tool „ChromeHistoryView“ verwendet, in der Hoffnung, dass dieses Artefakte findet. Da es aber, wie auch Autopsy und der DB-Browser für SQLite nichts anderes durchführt, als die Datenbank einzulesen, konnten auch damit keine weiteren Artefakte identifiziert werden.

Zusammenfassend, wurden somit in keiner Common Location PB-Artefakte hinterlassen.

## D.2. Uncommon Locations

### Analyse mit Autopsy - Kategorisierte Dateien

Wie in Abschnitt C.2 ausführlich dargestellt, wurden auch für Chrome alle von Autopsy kategorisierten Dateien betrachtet. Dabei wird immer das Image des vierten Snapshots aufgeführt, da darin alle kategorisierten Dateien aller vorherigen Snapshots enthalten sind.

**Web Bookmarks** Wie in Abbildung D.3 dargestellt, wurde nur ein einer Datei namens *Bing.url* ein Lesezeichen zur Bing-Startseite gefunden.


Source Name	S	C	O	URL	Title	Date Created	Program Name	Domain
 Bing.url			14	http://go.microsoft.com/fwlink/?LinkId=255142	Bing.url	2023-04-25 16:09:28 MESZ	Internet Explorer Analyzer	microsoft.com



Abbildung D.3.: Chrome: In Autopsy als „Web Bookmarks“ kategorisierte Dateien

Source Name	S	C	O	URL	Date Accessed	Name	Value	Program Name	Domain
WebCacheV01.dat	8	bing.com			2023-05-06 20:14:22 MESZ	SUID	M	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat	8	www.bing.com			2023-05-06 20:14:29 MESZ	MUID8	31708C5FC3C47068AFAD1CB47D0111	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat	8	bing.com			2023-05-06 20:14:22 MESZ	SRCHD	AF=NOFORM	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat	8	bing.com			2023-05-06 20:14:22 MESZ	SRCHUIJ	V=28GUDJ=-29D5FOF349244E8B2E9C57C13462A8dmnch=w=1	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat	8	bing.com			2023-05-06 20:14:22 MESZ	SRCHUSR	DOB=20230506	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat	8	bing.com			2023-05-06 20:14:27 MESZ	SRCHPGUSR	SRCHLANG=de&URL=1683404065126&JPMH=dee20405&JPMID=1683404065968	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat	8	bing.com			2023-05-06 20:14:24 MESZ	CertainAppIdUI	C164AA3A4C47E127DDC66A9D15CFD04C	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat	8	bing.com			2023-05-06 20:16:58 MESZ	ANON	A=A38F1B8C96CFS&B3F0C49DFFFFFFFFFF	Microsoft Edge Analyzer	bing.com
WebCacheV01.dat	8	live.com			2023-05-06 19:58:26 MESZ	MUID	20934DF6CAE768410BDC5EFFCE76FB0	Microsoft Edge Analyzer	live.com
WebCacheV01.dat	8	login.live.com			2023-05-06 20:14:44 MESZ	_Host-MSAUAUTHP		Microsoft Edge Analyzer	live.com

**Web History** Bei dieser Kategorie waren, wie in Abbildung D.5 dargestellt, nur zwei Dateien mit Browsingverläufen vorhanden. Dabei handelt es sich zweimal um die Outlook-Startseite, welche jedoch nie aufgerufen wurde.

Source Name	S	C	O	URL	Date Accessed	Program Name	Domain	Username
webCache01.dat				file:///ephotos	2023-05-06 19:59:20 MEIST	Microsoft Edge Analytics	Fornetrik	
webCache01.dat				file:///f	2023-05-06 19:59:20 MEIST	Microsoft Edge Analytics	Fornetrik	
webCache01.dat				file:///ephotos/vh_ch_ephoto1.htm	2023-05-06 19:59:20 MEIST	Microsoft Edge Analytics	Fornetrik	
webCache01.dat			B	https://pages.linuxswat20_dedim.de/?m=1001	2023-05-06 20:14:48 MEIST	Microsoft Edge Analytics	lwe.com	Fornetrik
webCache01.dat			B	https://pages.lwe.com/swat20_authcenter/?id=swat-ko000000070725&acq-source=swat.lwe.com_PWE_S&response_type=json&displaymode=embed&displaymode-width=1011&displaymode-height=1011&authcenter=swat_lwe.com_PWE_S&response_type=json&displaymode=embed&displaymode-width=1011&displaymode-height=1011&authcenter=swat_lwe.com_PWE_S	2023-05-06 20:14:48 MEIST	Microsoft Edge Analytics	lwe.com	Fornetrik
webCache01.dat				file:///ephotos/chrome_ephoto1.htm	2023-05-06 20:16:30 MEIST	Microsoft Edge Analytics	Fornetrik	
webCache01.dat				file:///c:/logfiles	2023-05-06 20:30:45 MEIST	Microsoft Edge Analytics	Fornetrik	
webCache01.dat				file:///ephotos/chrome_ephoto2.htm	2023-05-06 20:39:19 MEIST	Microsoft Edge Analytics	Fornetrik	
webCache01.dat				file:///c:/logfiles	2023-05-06 20:46:00 MEIST	Microsoft Edge Analytics	Fornetrik	

**Web Categories** Abbildung D.6 zeigt, dass in zwei Dateien nur Bing als Suchmaschine und live.com als „Web Email“ erkannt hat. Auch in den restlichen Images gab es hier keine weiteren Einträge.

Source Name	S	C	O	Source Type	Score	Conclusion	Configuration	Justification	Domain	Host	Name
 WebCacheV01.dat			1	File	Unknown				bing.com	bing.com	Search Engine
 WebCacheV01.dat			1	File	Unknown				live.com	login.live.com	Web Email

Somit wurden in allen Kategorien keine Browsing-Artefakte bei Chrome identifiziert.

Gemäß Methodik in Abschnitt 4.3 wurde die Registry bei Chrome sowohl als Common- als auch Uncommon Location untersucht.

## Process Monitor SetValue Operations

Als Teil der Common Locations wurden die Process Monitor Logfiles nach *RegSetValue*-Operationen gefiltert. Die darin enthaltenen Schlüssel wurden dann analysiert. Dabei war ein Großteil der vorhandenen Schlüssel direkt vom Datentyp String, welche man somit direkt ablesen konnte. Alle anderen Werte wurden zusätzlich zu Zeichenketten umgewandelt, um sicher keine Artefakte zu übersehen. Hier konnte aber auch kein Hinweis auf das PB Szenario gefunden werden.

## Stringsuche in Registry Hives

Als Teil der Uncommon Locations wurden alle in Tabelle 4.4 aufgelisteten Registry Hives in den Registry Explorer eingelesen und dann sowohl nach URLs, Keywords, HTML-Fragmente und E-Mail Artefakten gesucht. Abbildung D.7 zeigt einen Screenshot davon. Im Suchfenster unten ist zu sehen, dass hier keine Strings gefunden wurden. Exemplarisch ist hier das Registry-Explorer-Projekt zum vierten Snapshot gezeigt. Auch die Hives aller anderen Speicherabbilder zeigten hier keine Treffer. Somit sind keine Artefakte in der Registry auffindbar.

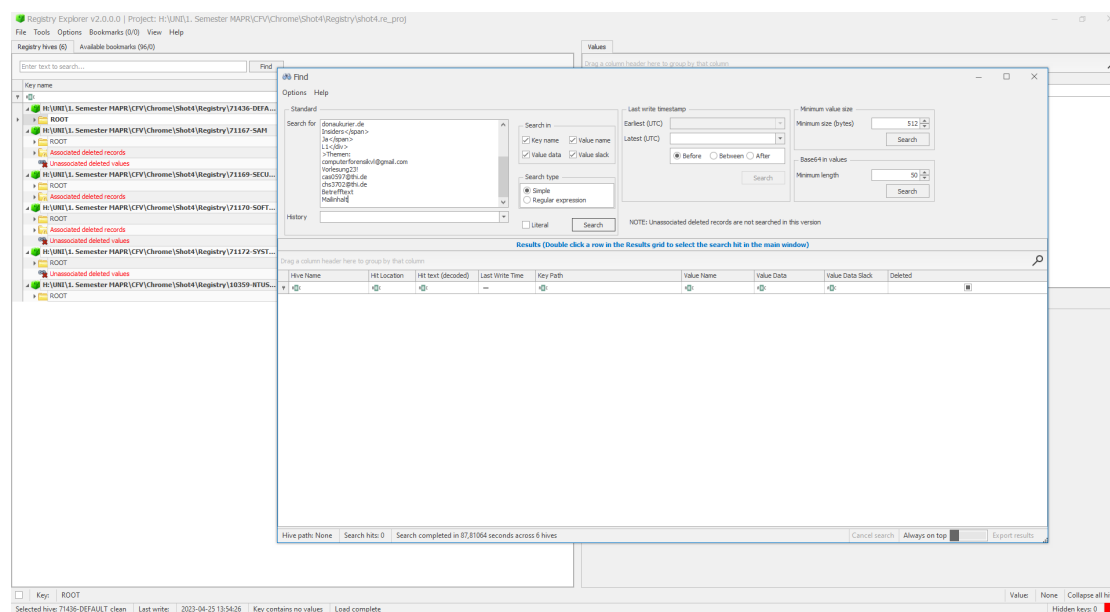


Abbildung D.7.: Chrome: Stringsuche in den Registry-Hives mithilfe des Registry Explorers

---

## E. Ausführliche Analyse: Brave

### E.1. Common Locations

#### Process Monitor WriteFile Operations

Wie in 4.3.1 beschrieben, wurden für den Browser Brave zwei Logfiles erstellt, welche die Prozessaktivitäten während (Erstes Logfile) und nach dem Browsing-Szenario beim Schließen des Browsers (Zweites Logfile) aufzeichnet. Beide Logfiles wurden anschließend so gefiltert, dass nur die relevanten Schreiboperationen übrig bleiben, welche hier analysiert werden sollen. Daraufhin wurden beide in Excel eingelesen und aufbereitet. Tabelle E.1 listet alle Dateien auf, welche während des Browsing-Szenarios beschrieben wurden. Die Kategorie „ComponentUnpacker“ umfasst deutlich mehr Dateien als in der Tabelle dargestellt. Um die Lesbarkeit zu bewahren, sind diese Dateien separat in Tabelle E.2 dargestellt. Keine der Dateien war wiederherstellbar. Bei Betrachtung der beiden Tabellen fällt sofort auf, dass während des private Browsers deutlich mehrere Dateien in browserunspezifische „uncommon“-Pfade beschrieben werden. Diese waren jedoch alle im *Local\Temp* oder im *Roaming* Ordner zu finden. Auch hier wurden wieder die Schreiboperationen in Dateiendungen bzw. Kategorien aufgeteilt. Die beiden größeren Dateikategorien, welche im Vergleich zu Chrome hinzugekommen sind, sind die „ComponentUnpacker“- und „chrome\_url\_fetcher“-Dateien. Da keiner der Dateien bis auf eine Einzige mehr vorhanden war, welche mit HxD untersucht wurde und dabei keine Artefakte feststellbar waren, wurde hier auch nicht weiter nachgeforscht.

Die restlichen Dateien verhielten sich dabei ähnlich zu Chrome, was den Grund haben könnte, dass beide Browser auf Chromium, dem open-source Browserprojekt, basieren bzw. darauf aufbauen.

Die .tmp Dateien waren wieder nicht wiederherstellbar, weder in den Festplatten-Images, noch im RAM-Dump. Die *-journal*-Dateien der SQLite-Datenbanken waren abermals leer. Die 000003.log-Datei wurde, genauso wie der ganze LevelDB-Ordner, analysiert, jedoch ohne einen Fund eines Artefakts. Die *store\_new*-Dateien enthielten ebenfalls keine relevanten Informationen. Die Wörterbuch-Dateien sowie die Dateien aus dem „D3DSCache“-Ordner waren ebenfalls ohne einen auf das PB zurückführbare Inhalt.

Anschließend folgt die Analyse des zweiten Logfiles.

Auch bei der Analyse der Dateien des zweiten Logfiles konnten keine Artefakte gefunden werden. Tabelle E.3 zeigt die Ergebnisse hierfür. Nähere Informationen zu den beschriebenen Dateien wurden im vorherigen Kapitel bereits für Chrome dargelegt und werden daher hier nicht nochmal aufgeführt.





Tabelle E.1.: Brave: alle "WriteFile"-Operationen des Logfiles 1

LOGFILE 1:				
Kategorie	Dateiname	Dateistatus	Verwendetes Tool zur Analyse	Enthaltene Artefakte
Temp files (.tmp)	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\0db1b95e-0d5c-4abc-8578-43f411aef4f6.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\34c45fa1-bbae-49ea-be4a-09ece2ba4a06.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\425f59e2-c8e1-4ba1-9959-c014285f46ba.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\4b1304ae-da90-46cd-8692-e95b7d206559.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\549d3611-028b-4847-b46e-c42a7c80c7e7.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\608cc84a-2701-45ba-a644-beb71bdd3aaa.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\71270d22-2772-4f9e-90e0-5a95af365a5d.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\73a8a877-dbff-4c42-ae66-5161368ec2a3.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\74013239-2558-4678-aed6-d8c4ff488083.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\75d05c29-c75e-4a2b-a957-31e2231cc7cc.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\9c13cd2a-ef5d-4483-ab26-02233a8964d8.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\abe9059-29ce-409c-8a51-ec18bd6f53f.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\b5af0eca-4f54-43b4-907b-d30180a2e2eb.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\c417166b-a007-4b57-8472-f5739506510.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\da9d1eca-98b8-44fb-80e2-28503ff59489.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\33c120ad-fa65-43d9-9092-88172b5bdf5b.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\617c8b03-79a1-41b1-a2e8-40b0b6ca134.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\6fdd7c94-a0f7-42b3-b19b-556408f99e7b.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\742a45df-231e-4362-8e75-b9d7004a8675.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\9d19e692-6d20-4725-9343-d6d667206de1.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\ef3f0a969-4ea1-40d4-b9bc-9dd6b0bb86e3.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\f34e3fe2-52d1-403d-8239-705d8d7750d0.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\f8ce722d-f552-4f08-839b-17d771596191.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\f90073ed-04af-4d66-8391-2c4e0e592b31.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Temp\940bede6-3d51-499e-8297-b6fb8be8b878.tmp	Datei nicht wiederherstellbar	N/A	N/A
SQLite -journal Dateien	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\History-journal	Datei leer (0 Bytes groß)	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\Web Data-journal	Datei leer (0 Bytes groß)	N/A	N/A
000003.log Dateien (LevelDB)	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\shared_proto_db\000003.log	Datei vorhanden	HxD	Keine PB-Artefakte
store_new-Dateien	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Safe Browsing\ChromeExt\Malware.store_new	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Safe Browsing\Ip\Malware.store_new	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Safe Browsing\UrlBilling.store_new	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Safe Browsing\UrlMalBin.store_new	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Safe Browsing\Url\Malware.store_new	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Safe Browsing\UrlSoceng.store_new	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Safe Browsing\UrlUws.store_new	Datei nicht wiederherstellbar	N/A	N/A
ComponentUnpacker-Dateien	Beispielweise: C:\Users\Forensik\AppData\Local\Temp\chrome\ComponentUnpacker_BeginUnzipping1624_1632884734_metadata\verified_contents.json	Dateien nicht wiederherstellbar	N/A	N/A
chrome_url_fetcher	C:\Users\Forensik\AppData\Local\Temp\chrome_url_fetcher_1624_1068124828\ggkkehgbnfjpeggfpleeakpidbkibmnm_2022.12.16.779_all_adskxucitmf4cbdu5r455ysysj5a.crx3	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Temp\chrome_url_fetcher_1624_1487667590\jamhcnnkiihinmdllakkaopbjbbcgflc_115.0.5779.0_all_acyyu57xgykye5bm362qyffav3q.crx3	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Temp\chrome_url_fetcher_1624_1812863496\ALzUVHP-vRgKCzqwbtGugSE	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\Temp\chrome_url_fetcher_1624_944393755\obedbbhhbpmojnkancicoggnmelmoomoc_20230506.531812958.14_all_DE500000_jpllevjyopdnosylzkd7obney.crx3	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\Temp\chrome_url_fetcher_1624_961672251\efnioljnjndmcbiieegkicadnoecijef_590_all_cr6cdatfpdldxmd7wmjxxodem.crx3	Datei nicht wiederherstellbar	N/A	N/A
Spelling default files	C:\Users\Forensik\AppData\Roaming\Microsoft\Spelling\de-DE\default.acf	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Roaming\Microsoft\Spelling\de-DE\default.dic	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Roaming\Microsoft\Spelling\de-DE\default.exc	Datei vorhanden	HxD	Keine PB-Artefakte
D3DS Cache	C:\Users\Forensik\AppData\Local\D3DSCache\4575d9371844649b\F4EB2D6C-ED2B-4BDD-AD9D-F913287E6768.idx	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\D3DSCache\4575d9371844649b\F4EB2D6C-ED2B-4BDD-AD9D-F913287E6768.lock	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\D3DSCache\4575d9371844649b\F4EB2D6C-ED2B-4BDD-AD9D-F913287E6768.val	Datei vorhanden	HxD	Keine PB-Artefakte

Tabelle E.2.: Brave alle „ComponentUnpacker“-Dateien des Logfiles 1

[illegible]

Tabelle E.3.: Brave alle "WriteFile"-Operationen des Logfiles 2

LOGFILE 2:				
Kategorie	Dateiname	Dateistatus	Verwendetes Tool zur Analyse	Enthaltene Artefakte
<i>settings.dat</i>	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Crashpad\settings.dat	Datei vorhanden	HxD	Keine PB-Artefakte
<i>Temp files (.tmp)</i>	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\57402cf7-46a9-4d82-b61c-21dfbb66dec7.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\44060f54-e8f1-4c07-af6e-6d8d40b920ca.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\25309f67-40f2-42fa-9350-df5ca2dbc818.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\6cfe06c0-8f91-4b10-abfe-a448809e9204.tmp	Datei nicht wiederherstellbar	N/A	N/A
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\fa8bc398-2d9b-4d6a-b63a-432272755bae.tmp	Datei nicht wiederherstellbar	N/A	N/A
<i>SQLite -journal Dateien</i>	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\History-journal	Datei leer (0 Bytes groß)	N/A	N/A
<i>JSON-Datei</i>	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Variations	Datei vorhanden	HxD	Keine PB-Artefakte
<i>000003.log Datei (LevelDB)</i>	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\Session Storage\000003.log	Datei vorhanden	HxD	Keine PB-Artefakte
<i>data_1 files</i>	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\Cache\Cache_Data\data_1	Datei vorhanden	ChromeCacheView, HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\GrShaderCache\data_1	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\ShaderCache\data_1	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\DawnCache\data_1	Datei vorhanden	HxD	Keine PB-Artefakte
	C:\Users\Forensik\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\GPUCache\data_1	Datei vorhanden	HxD	Keine PB-Artefakte

---

## Cache

Auch für den Brave-Browser wurde der Cache mittels des Tools ChromeCacheView untersucht. Das Ergebnis hier war jedoch erneut, dass keine Cache-Dateien auf das durchgeführte PB Szenario zurückführbar sind. Dies wurde jeweils im zweiten, dritten und vierten Snapshot mit den Cache-Ordnern durchgeführt. Eine weitere Analyse durch Extraktion und Untersuchung der Dateien mittels HxD zeigte erneut, dass im Cache keine Artefakte vorhanden sind.

## SQLite-Datenbanken

Tabelle E.4 zeigt eine Übersicht über die Veränderungen der relevanten Brave-Datenbanken über das gesamte Browsing-Szenario. Auf den ersten Blick ist sofort erkennbar, dass bereits alle Datenbanken zu Beginn vorliegen und am Ende nicht gelöscht werden, wie das bei Chrome der Fall ist. Die Datenbanken *History* und *Web Data* verhielten sich dabei ähnlich wie bei Chrome, bei *Reporting and NEL* gab es beispielsweise weniger Veränderungen bei Brave. Zusammenfassend konnte hier bei Brave aber auch kein Artefakt identifiziert werden.

Tabelle E.4.: Brave: Veränderung der SQLite-Datenbanken während der Versuchsdurchführung

Dateiname	Vor Browsing Szenario (S1)	Nach Browsing-Szenario, Browser geöffnet (S2)	Nach Browsing-Szenario, Browser geschlossen (S3)	VM heruntergefahren (S4)
History	Initialisiert (Nur Spaltennamen)	keine Veränderung	Datensätze hinzugekommen, keine Artefakte	Datensatz gelöscht
Web Data	Initialisiert (Nur Spaltennamen)	Inhalte verändert, keine PB-Artefakte	keine Veränderung	Datensätze hinzugekommen, keine Artefakte
Shortcuts	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung
Top Sites	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung
Login Data	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung
Network Action Predictor	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung
Reporting and NEL	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung
Cookies	Initialisiert (Nur Spaltennamen)	keine Veränderung	keine Veränderung	keine Veränderung

## E.2. Uncommon Locations

### Analyse mit Autopsy - Kategorisierte Dateien

In Autopsy wurden neben der Stringsuche die kategorisierten Dateien analysiert. Die verschiedenen Kategorien *Web Bookmarks*, *Web Cookies*, *Web History* und *Web Categories* wurden erneut bei Brave untersucht.

**Web Bookmarks** Wie in Abbildung E.1 dargestellt, wurde nur ein einer Datei namens *Bing.url* ein Lesezeichen zur Micorsoft Bing-Startseite gefunden.


Source Name	S	C	O	URL	Title	Date Created	Program Name	Domain
 Bing.url			14	http://go.microsoft.com/fwlink/p/?LinkId=255142	Bing.url	2023-04-25 16:09:28 MESZ	Internet Explorer Analyzer	microsoft.com

Abbildung E.1.: In Autopsy als „Web Bookmarks“ kategorisierte Dateien

**Web Cookies** In Abbildung E.2 sind die zehn Cookies-Einträge zu sehen, welche Cookies der Bing- und Outlook-Startseite darstellen.











Source Name	S	C	O	URL	Date Accessed	Name	Value	Program Name	Domain
 WebCacheV01.dat			8	bing.com	2023-05-07 12:35:13 MESZ	SUID	A	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			8	www.bing.com	2023-05-07 12:35:15 MESZ	MUIDB	31708C5FC3CF47068AFAD1CB47D0111	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			8	bing.com	2023-05-07 12:26:08 MESZ	SRCHD	AF=NJOFORM	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			8	bing.com	2023-05-07 12:26:08 MESZ	SRCHUID	V=28&GUID=D238DA0C035DC405DAAE8825DCFE3F5F&dmnchg=1	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			8	bing.com	2023-05-07 12:26:08 MESZ	SRCHUSR	DOB=20230507	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			8	bing.com	2023-05-07 12:35:13 MESZ	SRCHHPGUSR	SRCHLANG=de&LUT=1683462371676&JPMH=dee20405&JPMID=1683462911877	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			8	bing.com	2023-05-07 12:26:11 MESZ	CortanaAppUID	C164AA3A4D47E127DC66AD915CFD04C	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			8	bing.com	2023-05-07 12:31:07 MESZ	ANON	A=7D1BCC32FE293AE5B52A1768FFFFFFF	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			8	live.com	2023-05-07 12:26:23 MESZ	MUID	0D4D788ED765609F051B68B1D36564D1	Microsoft Edge Analyzer	live.com
 WebCacheV01.dat			8	login.live.com	2023-05-07 12:35:31 MESZ	__Host-MSAATHP		Microsoft Edge Analyzer	live.com

Abbildung E.2.: Brave: In Autopsy als „Web Cookies“ kategorisierte Dateien

**Web History** Bei dieser Kategorie waren, wie in Abbildung E.3 dargestellt, nur zwei Dateien mit Browsingverläufen vorhanden. Dabei handelt es sich zweimal um die Outlook-Startseite, welche jedoch nie aufgerufen wurde.








Source Name	S	C	O	URL	Date Accessed	Program Name	Domain	Username
 WebCacheV01.dat			8	https://login.live.com/oauth20_desktop.aspx?fb=1031	2023-05-07 12:35:31 MESZ	Microsoft Edge Analyzer	live.com	Forensik
 WebCacheV01.dat			8	https://login.live.com/oauth20_authorize.srf?ident_id=0000000046073C53scope=service:us:live.com:IMEL_S3&response_type=token&display=windows&theme=win7&w=1033&redirect_uri=https://login.live.com/oauth20_desktop.aspx?fb=1031&w=452	2023-05-07 12:35:33 MESZ	Microsoft Edge Analyzer	live.com	Forensik
 WebCacheV01.dat				no-spring-relay://lgdhes/	2023-05-07 12:35:18 MESZ	Microsoft Edge Analyzer		Forensik
 WebCacheV01.dat				file:///IP:(msghst_brave_shot2.htm	2023-05-07 12:35:24 MESZ	Microsoft Edge Analyzer		Forensik
 WebCacheV01.dat				file:///IP:(lgdhes)	2023-05-07 13:29:43 MESZ	Microsoft Edge Analyzer		Forensik
 WebCacheV01.dat				file:///IP:(msghst_brave_shot2.htm	2023-05-07 13:33:31 MESZ	Microsoft Edge Analyzer		Forensik
 WebCacheV01.dat				file:///IP:(lgdhes)	2023-05-07 14:05:06 MESZ	Microsoft Edge Analyzer		Forensik

Abbildung E.3.: Brave: In Autopsy als „Web History“ kategorisierte Dateien

**Web Categories** Abbildung E.4 zeigt, dass in zwei Dateien nur Bing als Suchmaschine und live.com als „Web Email“ erkannt hat. Auch in den restlichen Images gab es hier keine weiteren Einträge.



Source Name	S	C	O	Source Type	Score	Conclusion	Configuration	Justification	Domain	Host	Name
 WebCacheV01.dat			1	File	Unknown				live.com	login.live.com	Web Email
 WebCacheV01.dat			1	File	Unknown				bing.com	www.bing.com	Search Engine

Abbildung E.4.: Brave: In Autopsy als „Web Categories“ kategorisierte Dateien

Somit wurden in allen Kategorien keine Browsing-Artefakte bei Brave identifiziert.

---

### **E.3. Registry**

Ebenso wurde für Brave gemäß Methodik in Abschnitt 4.3 die Registry sowohl als Common- als auch Uncommon Location untersucht.

#### **Process Monitor WriteFile Operations**

In den Registry-Aktivitäten der Process Monitor Logfiles konnten keine relevanten Informationen aus den RegSet-Operationen extrahiert werden, welche das Browsing-Szenario betreffen.

#### **Stringsuche in Registry Hives**

Eine detaillierte Stringsuche in den extrahierten Registry-Hives wurde ebenfalls durchgeführt. Dabei kam es jedoch in allen vier VM-Snapshots zu keinem Treffer mit den Suchbegriffen.

# Abbildungsverzeichnis

4.1. Funktion „Neue Identität“ des Tor-Browsers . . . . .	7
4.2. Zeitpunkte zur Datensammlung während der Versuchsdurchführung nach [38] . . . . .	14
4.3. Zeitpunkte zur Datensammlung während der Versuchsdurchführung für den Tor-Browser . . . . .	14
4.4. Process Monitor Filter für Datei-Schreiboperationen . . . . .	16
4.5. Vorgehen zur Dateieextraktion und -analyse . . . . .	17
4.6. Vorgehen zur Dateieextraktion und -analyse von SQLite-Datenbanken . . . . .	18
4.7. Autopsy Funktion zur Stichwortsuche . . . . .	19
4.8. Beispiel für ein RAM-Artefakt ohne vorheriges Browsing-Szenario . . . . .	19
4.9. Abhängigkeiten der verwendeten Volatility-Plugins yarascan, pslist und memmap . . . . .	20
4.10. Process Monitor Filter für Registry-Schreiboperationen . . . . .	21
5.1. Firefox: Anzahl gefundener Suchbegriffe im RAM . . . . .	25
5.2. Firefox: Anzahl gefundener URL-Artefakte im RAM . . . . .	26
5.3. Firefox: Unter dem SVChost-Prozess PID 2252 läuft der DNSCache-Dienst. . . . .	26
5.4. Firefox: Anzahl gefundener E-Mail Artefakte im RAM . . . . .	27
5.5. Firefox: Passwort-Klartext in Speicherseiten von PID 7420 . . . . .	28
5.6. Firefox: Passwort-Klartext in Speicherseiten von PID 8424 . . . . .	28
5.7. Firefox: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im RAM . . . . .	29
5.8. Tor-Browser: Anzahl gefundenener Suchbegriffe im RAM . . . . .	31
5.9. Tor-Browser: Anzahl gefundener URL-Artefakte im RAM . . . . .	32
5.10. Tor-Browser: Anzahl gefundener E-Mail Artefakte im RAM . . . . .	33
5.11. Tor-Browser: Passwort-Klartext in Speicherseiten von PID 708 . . . . .	34
5.12. Tor-Browser: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im RAM . . . . .	35
5.13. Chrome: Ausschnitt aus dem Memory-Dump-File an der Stelle des gefundenen HTML-Strings . . . . .	37
5.14. Chrome: Beginn (links) und Ende (rechts) des HTML-Dokuments der Donau- kurier Webseite . . . . .	38
5.15. Chrome: Beginn (links) und Ende (rechts) des HTML-Dokuments der Donau- kurier Webseite in HxD . . . . .	38
5.16. Chrome: Anzahl gefundener Suchbegriffe im RAM . . . . .	39
5.17. Chrome: Anzahl gefundener URLs im RAM . . . . .	40
5.18. Chrome: Anzahl gefundener E-Mail Artefakte im RAM . . . . .	41
5.19. Chrome: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im RAM . . . . .	42



5.20. Brave: Ausschnitt aus VS Code mit dem Vergleich der beiden Donaukurier Webseiten . . . . .	44
5.21. Brave: Anzahl gefundener Suchbegriffe im RAM . . . . .	45
5.22. Brave: Anzahl gefundener URLs im RAM . . . . .	46
5.23. Brave: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im RAM . . . . .	46
5.24. Brave: Anzahl gefundener E-Mail Artefakte im RAM . . . . .	47
6.1. Zusammenfassung gefundener Artefakte im RAM der Browser . . . . .	49
B.1. Firefox: Mit MZCacheView eingelesene Cache-Dateien . . . . .	58
B.2. Firefox: Sitzungsdatei recovery.jsonlz4, geöffnet mit dem „Session History Scrounger for Firefox“ . . . . .	59
B.3. Firefox: Anzahl Schreiboperationen Logfile 1 vs Logfile 2, geordnet nach Kategorie . . . . .	61
B.4. Firefox: Von Autopsy als „Web Bookmarks“ kategorisierte Dateien . . . . .	61
B.5. Firefox: In Autopsy als „Web Cookies“ kategorisierte Dateien . . . . .	62
B.6. Firefox: In Autopsy als „Web History“ kategorisierte Dateien . . . . .	62
B.7. Firefox: In Autopsy als „Web Categories“ kategorisierte Dateien . . . . .	63
B.8. Firefox: Registry „SetValue“ Operationen in den Process Monitor Logfiles 1 und 2 . . . . .	63
C.1. Tor-Browser: Anzahl Schreiboperationen Logfile 1 vs Logfile 2-1 vs Logfile 2-2, geordnet nach Kategorie . . . . .	69
C.2. Tor-Browser: In Autopsy als „Web Bookmarks“ kategorisierte Dateien . . . . .	69
C.3. Tor-Browser: In Autopsy als „Web Cookies“ kategorisierte Dateien . . . . .	70
C.4. Tor-Browser: In Autopsy als „Web History“ kategorisierte Dateien . . . . .	70
C.5. Tor-Browser: In Autopsy als „Web Categories“ kategorisierte Dateien . . . . .	70
C.6. Tor-Browser: Registry „SetValue“ Operationen in den Process Monitor Logfiles 1, 2-1 und 2-2 . . . . .	71
D.1. Chrome: Cache Analyse mit ChromeCacheView . . . . .	75
D.2. Chrome: Analyse eines Cache-files . . . . .	76
D.3. Chrome: In Autopsy als „Web Bookmarks“ kategorisierte Dateien . . . . .	77
D.4. Chrome: In Autopsy als „Web Cookies“ kategorisierte Dateien . . . . .	78
D.5. Chrome: In Autopsy als „Web History“ kategorisierte Dateien . . . . .	78
D.6. Chrome: In Autopsy als „Web Categories“ kategorisierte Dateien . . . . .	78
D.7. Chrome: Stringsuche in den Registry-Hives mithilfe des Registry Explorers . . . . .	79
E.1. In Autopsy als „Web Bookmarks“ kategorisierte Dateien . . . . .	85
E.2. Brave: In Autopsy als „Web Cookies“ kategorisierte Dateien . . . . .	85
E.3. Brave: In Autopsy als „Web History“ kategorisierte Dateien . . . . .	85
E.4. Brave: In Autopsy als „Web Categories“ kategorisierte Dateien . . . . .	85

# Tabellenverzeichnis

4.1. Private-Browsing-Artefakte des Browsing-Szenarions . . . . .	10
4.2. Basiskonfiguration jeder VM des Versuchs . . . . .	10
4.3. Vollständige Liste der verwendeten Software dieses Versuchs . . . . .	12
4.4. Windows Registry Hives . . . . .	22
5.1. Firefox: Abbildung der virtuellen Speicheradressen der gefundenen Strings auf Byte-Offsets der entsprechenden Speicherseiten . . . . .	28
5.2. Tor-Browser: Abbildung der virtuellen Speicheradressen der gefundenen Strings im RAM auf Byte-Offsets der entsprechenden Speicherseiten . . . . .	34
5.3. Chrome: Abbildung virtueller Adressen auf Byte-Offsets der Speicherseite . .	37
6.1. Gewinner-Tabelle der vier untersuchten Browser . . . . .	51
6.2. Ermittelte Gewinner gemäß Gewinner-Tabelle . . . . .	51
B.1. Firefox alle „WriteFile“-Operationen der Logfiles 1 und 2 . . . . .	57
B.2. Firefox: Veränderte SQLite-Datenbanken und deren Verwendungszwecke . . .	59
B.3. Veränderung der Firefox SQLite-Datenbanken während der Versuchsdurchführung	60
C.1. Tor-Browser: Alle „WriteFile“-Operationen der Logfiles 1, 2-1 und 2-2 . . . .	66
C.2. Tor-Browser: Veränderung der SQLite-Datenbanken während der Versuchs- durchführung . . . . .	68
D.1. Chrome alle „WriteFile“-Operationen des Logfiles 1 . . . . .	73
D.2. Chrome: alle „WriteFile“-Operationen des Logfiles 2 . . . . .	74
D.3. Chrome: Veränderung der SQLite-Datenbanken während der Versuchsdurch- führung . . . . .	77
E.1. Brave: alle „WriteFile“-Operationen des Logfiles 1 . . . . .	81
E.2. Brave alle „ComponentUnpacker“-Dateien des Logfiles 1 . . . . .	82
E.3. Brave alle „WriteFile“-Operationen des Logfiles 2 . . . . .	83
E.4. Brave: Veränderung der SQLite-Datenbanken während der Versuchsdurchführung	84

# Literatur

- [1] Gaurav Aggarwal u. a. "An Analysis of Private Browsing Modes in Modern Browsers." In: *USENIX security symposium*. 2010, S. 79–94.
- [2] Autopsy. *Autopsy | Digital Forensics*. 29.03.2023.  
URL: <https://www.autopsy.com/>.
- [3] Gabriele Bonetti u. a.  
"Black-box forensic and antifoensic characteristics of solid-state drives".  
In: *Journal of Computer Virology and Hacking Techniques* 10 (2014), S. 255–271.
- [4] Brave. *Die beste Privatsphäre im Internet*. URL: <https://brave.com/de/>.
- [5] Bugzilla. 1452114 - *Spurious report of OutOfMemory if a script fails to parse*. 5.06.2023. URL: [https://bugzilla.mozilla.org/show\\_bug.cgi?id=1452114](https://bugzilla.mozilla.org/show_bug.cgi?id=1452114).
- [6] Chromium. *Contents of /trunk/src/chrome/browser/browser\_shutdown.cc*.  
Zuletzt zugegriffen am 28.05.2023 um 16:34 Uhr. 2009.  
URL: [https://src.chromium.org/viewvc/chrome/trunk/src/chrome/browser/browser\\_shutdown.cc?pathrev=26058](https://src.chromium.org/viewvc/chrome/trunk/src/chrome/browser/browser_shutdown.cc?pathrev=26058).
- [7] Chromium. *Crashpad Overview Design*.  
Zuletzt zugegriffen am 28.05.2023 um 21:58 Uhr. URL: [https://chromium.googlesource.com/crashpad/crashpad/+HEAD/doc/overview\\_design.md](https://chromium.googlesource.com/crashpad/crashpad/+HEAD/doc/overview_design.md).
- [8] Hasan Fayyad-Kazan u. a.  
"Forensic analysis of private browsing mechanisms: Tracing internet activities".  
In: (2021).
- [9] Ryan M Gabet, Kathryn C Seigfried-Spellar und Marcus K Rogers.  
"A comparative forensic analysis of privacy enhanced web browsers and private browsing modes of common web browsers". In: *International Journal of Electronic Security and Digital Forensics* 10.4 (2018), S. 356–371.
- [10] GitHub. *Firefox Data Stores*. 8.04.2019.  
URL: <https://mozilla.github.io/firefox-browser-architecture/text/0010-firefox-data-stores.html>.
- [11] GitHub. *GitHub - mozilla/glean: Modern cross-platform telemetry*. 5.06.2023.  
URL: <https://github.com/mozilla/glean>.
- [12] GitHub. *GitHub - volatilityfoundation/volatility3: Volatility 3.0 development*. 5.06.2023. URL: <https://github.com/volatilityfoundation/volatility3>.

- 
- [13] Gitlab. *Tor browser writes SiteSecurityServiceState.txt with usage history (#18589)* · Issues · The Tor Project / Applications / Tor Browser · GitLab. 5.06.2023.  
URL: <https://gitlab.torproject.org/tpo/applications/tor-browser/-/issues/18589>.
- [14] Google. *leveldb*. Zuletzt zugegriffen am 28.05.2023 um 22:20 Uhr. 2023.  
URL: <https://github.com/google/leveldb>.
- [15] Google Chrome. *Startseite*. URL: <https://www.google.com/intl/de/chrome/>.
- [16] Ms Pooja Gupta. "Capturing Ephemeral Evidence Using Live Forensics".  
In: *IOSR J. Electron. Commun. Eng* (2013), S. 109–113.
- [17] Haircutfish. *TryHackMe Windows Forensics 1: Task 3 Accessing registry hives offline & Task 4 Data Acquisition*. 4.11.2022.  
URL: <https://medium.com/@haircutfish/tryhackme-windows-forensics-1-task-3-accessing-registry-hives-offline-task-4-data-acquisition-b440f5be2a13>.
- [18] *Hang on! That's not SQLite! Chrome, Electron and LevelDB*.  
Zuletzt zugegriffen am 28.05.2023 um 22:21 Uhr. Okt. 2020.  
URL: <https://www.cclsolutionsgroup.com/post/hang-on-thats-not-sqlite-chrome-electron-and-leveldb>.
- [19] Nihad A Hassan. *Digital forensics basics: A practical guide using Windows OS*.  
Apress, 2019.
- [20] Graeme Horsman u. a. "A forensic examination of web browser privacy-modes".  
In: *Forensic Science International: Reports* 1 (2019), S. 100036.
- [21] Windoes OS Hub.  
*Memory Compression Process: High Memory and CPU Usage in Windows 10 and 11*.  
Zuletzt zugegriffen am 08.06.2023 um 11:51 Uhr. Jan. 2022. URL: <https://woshub.com/memory-compression-process-high-usage-windows-10/>.
- [22] InfoSec Notes. *Artefacts overview*. 5.06.2023.  
URL: [https://notes.qazeer.io/dfir/windows/\\_artefacts\\_overview](https://notes.qazeer.io/dfir/windows/_artefacts_overview).
- [23] Aina Izzati und Nurul Hidayah Ab Rahman. "A Comparative Analysis of Residual Data Between Private Browsing and Normal Browsing Using Live Memory Acquisition".  
In: *Applied Information Technology And Computer Science* 3.2 (2022), S. 68–83.
- [24] Jefferson Scher.  
*Session History Scrounger for Firefox (with lz4 support) — Fx File Utilities*.  
29.11.2020. URL: <https://www.jeffersonscher.com/ffu/scrounger.html>.
- [25] Ahmed Redha Mahlous und Houssam Mahlous. "Private Browsing Forensic Analysis: A Case Study of Privacy Preservation in the Brave Browser".  
In: *International Journal of Intelligent Engineering Systems* 13.06 (2020), S. 294–306.
- [26] MandiOhlinger. *Business Activity Monitoring (BAM) - BizTalk Server*. 5.06.2023.  
URL: <https://learn.microsoft.com/de-de/biztalk/core/business-activity-monitoring-bam>.

- 
- [27] Markruss. *Prozess-Explorer - Sysinternals*. 5.06.2023. URL: <https://learn.microsoft.com/de-de/sysinternals/downloads/process-explorer>.
- [28] Markruss. *Prozessmonitor - Sysinternals*. 5.06.2023. URL: <https://learn.microsoft.com/de-de/sysinternals/downloads/procmon>.
- [29] Raihana Md Saidi u. a. "Analysis of Private Browsing Activities". In: *Regional Conference on Science, Technology and Social Sciences (RCSTSS 2016) Theoretical and Applied Sciences*. Springer. 2018, S. 217–228.
- [30] Microsoft Learn. *Einführung in Auslagerungsdateien*. 21.03.2023. URL: <https://learn.microsoft.com/de-de/troubleshoot/windows-client/performance/introduction-to-the-page-file>.
- [31] Microsoft Learn. *How to disable Windows 10 DNS Cache services - Microsoft Q&A*. 5.06.2023. URL: <https://learn.microsoft.com/en-us/answers/questions/47441/how-to-disable-windows-10-dns-cache-services>.
- [32] Matt Mills. *Skeleton UI, New Firefox Interface to Start up Much Faster | ITIGIC*. 2021. URL: [https://itigic.com/skeleton-ui-new-firefox-interface-to-start-up-much-faster/#google\\_vignette?utm\\_content=cmp=true](https://itigic.com/skeleton-ui-new-firefox-interface-to-start-up-much-faster/#google_vignette?utm_content=cmp=true).
- [33] Reza Montasari und Pekka Peltola. "Computer forensic analysis of private browsing modes". In: *Global Security, Safety and Sustainability: Tomorrow's Challenges of Cyber Security: 10th International Conference, ICGS3 2015, London, UK, September 15-17, 2015. Proceedings 10*. Springer. 2015, S. 96–109.
- [34] Mozilla. *Geschichte des Mozilla-Projekts*. 5.06.2023. URL: <https://www.mozilla.org/de/about/history/>.
- [35] Mozilla Support. *AlternateServices.txt: what does this file do?* 26.10.2020. URL: <https://support.mozilla.org/en-US/questions/1310302>.
- [36] MozillaWiki. *StartupCache - MozillaWiki*. 5.06.2023. URL: <https://wiki.mozilla.org/StartupCache>.
- [37] mozillazine. *Prefs.js file - MozillaZine Knowledge Base*. 29.12.2022. URL: [https://kb.mozillazine.org/Prefs.js\\_file](https://kb.mozillazine.org/Prefs.js_file).
- [38] Matt Muir, Petra Leimich und William J Buchanan. "A forensic audit of the tor browser bundle". In: *Digital Investigation* 29 (2019), S. 118–128.
- [39] Apurva Nalawade, Smita Bharne und Vanita Mane. "Forensic analysis and evidence collection for web browser activity". In: *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*. IEEE. 2016, S. 518–522.
- [40] Nicholasswhite. *Diensthostrefactoring in Windows 10 Version 1703 - Windows Application Management*. 5.06.2023. URL: <https://learn.microsoft.com/de-de/windows/application-management/svchost-service-refactoring>.

- 
- [41] Junghoon Oh, Seungbong Lee und Sangjin Lee.  
“Advanced evidence collection and analysis of web browser activity”.  
In: *Digital investigation* 8 (2011), S62–S70.
- [42] Donny Jacob Ohana und Narasimha Shashidhar.  
“Do private and portable web browsers leave incriminating evidence? a forensic analysis of residual artifacts from private and portable web browsing sessions”.  
In: *2013 IEEE Security and Privacy Workshops*. IEEE. 2013, S. 135–142.
- [43] Oracle. *1.10. Snapshots*. 2020. URL: <https://docs.oracle.com/en/virtualization/virtualbox/6.0/user/snapshots.html>.
- [44] PCTipps. *Dwm.exe: Was ist der Desktop Window Manager?*  
<https://pctipps.de/dwm-exe/>. Zuletzt zugegriffen am 08.06.2023 um 12:29 Uhr.
- [45] Daniel Perdices u. a.  
“Web browsing privacy in the deep learning era: Beyond VPNs and encryption”.  
In: *Computer Networks* 220 (2023), S. 109471.
- [46] Denis Pogonin und Igor Korkin.  
*Microsoft Defender Will Be Defended: MemoryRanger Prevents Blinding Windows AV*. 2022. DOI: <https://doi.org/10.48550/arXiv.2210.02821>.  
arXiv: 2210.02821 [cs.CR].
- [47] Digvijaysinh Rathod. “Darknet forensics”. In: *future* 11 (2017), S. 12.
- [48] Tri Rochmadi, Imam Riadi und Yudi Prayudi.  
“Live forensics for anti-forensics analysis on private portable web browser”.  
In: *Int. J. Comput. Appl* 164.8 (2017), S. 31–37.
- [49] Huwida Said u. a. “Forensic analysis of private browsing artifacts”.  
In: *2011 International Conference on Innovations in Information Technology*. IEEE. 2011, S. 197–202.
- [50] Priya P Sajan u. a. “Tor Browser Forensics”. In: *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12.11 (2021), S. 5599–5608.
- [51] Kiavash Satvat u. a. “On the privacy of private browsing—a forensic approach”.  
In: *Data Privacy Management and Autonomous Spontaneous Security: 8th International Workshop, DPM 2013, and 6th International Workshop, SETOP 2013, Egham, UK, September 12-13, 2013, Revised Selected Papers*. Springer. 2014, S. 380–389.
- [52] SecureDrop. *Share and accept documents securely*. 5.06.2023.  
URL: <https://securedrop.org/>.
- [53] SQLite. *Temporary Files Used By SQLite*.  
Zuletzt zugegriffen am 28.05.2023 um 22:10 Uhr. Jan. 2022.  
URL: <https://www.sqlite.org/tempfiles.html>.

- 
- [54] Statista. *Global market share held by leading internet browsers from January 2012 to May 2023*. 23.05.2023. URL: <https://www.statista.com/statistics/268254/market-share-of-internet-browsers-worldwide-since-2009/>.
- [55] Sylie. *Lösung: Shell Infrastructure Host hohe CPU-Auslastung [MiniTool]*. <https://de.minitool.com/nachrichten/shell-infrastructure-host-hohe-cpu-auslastung.html>. Zuletzt zugegriffen am 08.06.2023 um 12:20 Uhr. Feb. 2023.
- [56] Tech Support Guy. *What exactly is in Firefox's Cache2 folder?* 5.06.2023. URL: <https://www.techguy.org/threads/what-exactly-is-in-firefoxs-cache2-folder.1221567/>.
- [57] TILT. *Memory Forensics of a Virtualbox VM* · TILT. 25.03.2023. URL: <https://kollee.github.io/posts/memory-forensics-of-a-virtualbox-vm/>.
- [58] Sergey Tkachenko. *Add or Remove Words in Spell Checking Dictionary in Windows 10*. Zuletzt zugegriffen am 28.05.2023 um 21:59 Uhr. Jan. 2018. URL: <https://winaero.com/words-spell-checking-dictionary-windows-10/>.
- [59] Tor. *Das Tor Project | Privatsphäre & Freiheit Online*. 24.05.2023. URL: <https://www.torproject.org/de/download/>.
- [60] WinTotal. *Was ist hiberfil.sys? - So können Sie die Datei löschen oder verschieben*. <https://www.wintotal.de/tipp/hiberfil-sys/>. Zuletzt zugegriffen am 15.06.2023 um 23:02 Uhr.
- [61] Yunus Yusoff, Roslan Ismail und Zainuddin Hassan. "Common phases of computer forensics investigation models". In: *International Journal of Computer Science & Information Technology* 3.3 (2011), S. 17–31.