

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

## **Computer-Forensik und Vorfallsbehandlung**

Carl Schünemann

Christoph Sell

29.08.2025

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
<b>2. Theoretischer Hintergrund</b>	<b>2</b>
2.1. Private Browsing . . . . .	2
2.2. Angreifermodell . . . . .	2
2.3. Private Browsing Artefakte . . . . .	2
<b>3. Ziel der Arbeit</b>	<b>3</b>
<b>4. Methodik</b>	<b>4</b>
4.1. Vorbereitung . . . . .	4
4.1.1. Browserauswahl . . . . .	4
4.1.2. Browsing Szenario . . . . .	6
4.2. Datensammlung . . . . .	10
4.3. Datenanalyse . . . . .	12
4.3.1. Common Locations . . . . .	12
4.3.2. Uncommon Locations . . . . .	15
4.3.3. Registry . . . . .	17
<b>5. Ergebnisse</b>	<b>20</b>
5.1. Firefox . . . . .	20
5.2. Tor . . . . .	27
5.3. Chrome . . . . .	32
5.4. Brave . . . . .	32
<b>6. Vergleich der Browser</b>	<b>33</b>
<b>7. Diskussion</b>	<b>37</b>
<b>8. Fazit</b>	<b>38</b>
<b>Anhänge</b>	<b>39</b>
A. Yara-Regeln . . . . .	43
B. Ausführliche Analyse: Firefox . . . . .	44
B.1. Common Locations . . . . .	44
B.2. Uncommon Locations . . . . .	49

C.	Ausführliche Analyse: Tor . . . . .	53
C.1.	Common Locations . . . . .	53
C.2.	Uncommon Locations . . . . .	57
C.3.	Registry . . . . .	58
<b>Literaturverzeichnis</b>		<b>60</b>
<b>Literatur</b>		<b>60</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. [37] Um die Privatsphäre im gesamten Internet zu schützen, werden zusätzliche Maßnahmen, wie beispielsweise VPNs empfohlen. [35]

Es gibt unterschiedliche Nutzermodelle für private Browsing-Modi. Einerseits verwenden Privatpersonen diese Technologien, um ihre Privatsphäre zu schützen und ihren lokalen digitalen Fußabdruck im Internet zu regulieren [13]. Darüber hinaus nutzen einige 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 [24, 37]. 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 [36]. 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. [45] Jedoch erschweren seine Funktionalitäten forensische Ermittlungen zu kriminellen Nutzern [29, 36].

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" [40]. Im Rahmen dieser Seminararbeit werden die privaten Browser die privaten Browsing-Modi von vier Webbrowsern untersucht: Mozilla Firefox, Tor-Browser, Google Chrome und Brave [24]. Es wird analysiert, ob und welche Spuren von diesen Browsern in ihren privaten Modi auf den lokalen Rechnern hinterlassen werden.

## **2. Theoretischer Hintergrund**

\*\*\* TODO: Christoph \*\*\*

### **2.1. Private Browsing**

### **2.2. Angreifermodell**

### **2.3. Private Browsing Artefakte**

### 3. Ziel der Arbeit

Die vorliegende Seminararbeit hat das Ziel, die Auswirkungen privater Browsingmodi auf potentiell 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 potentiellen Artefakte zu identifizieren. Dies verbessert die Effizienz zukünftiger Untersuchungen und verhindert, dass wichtige Inhalte übersehen werden. [13]

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 Kapitel X (TODO!) 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 Browsingmodi untersucht.

## 4. Methodik

In der Browserforensik 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, 13, 15] Izzati et al. empfehlen als Vorgehensmodell für die Browser-Forensik das *Generic Model Computer Forensics Investigations*, kurz *GCFIM*. [46] In Ihrer Anwendung auf die Browser-Forensik besteht das Modell aus vier Phasen: [15]

- 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. [15] 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 [42] (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, 5, 6, 13, 15, 17, 21, 24, 29, 30, 32, 33, 37, 38, 40]

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 Kapitel X (TODO!) 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. [25]

## 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. [45]

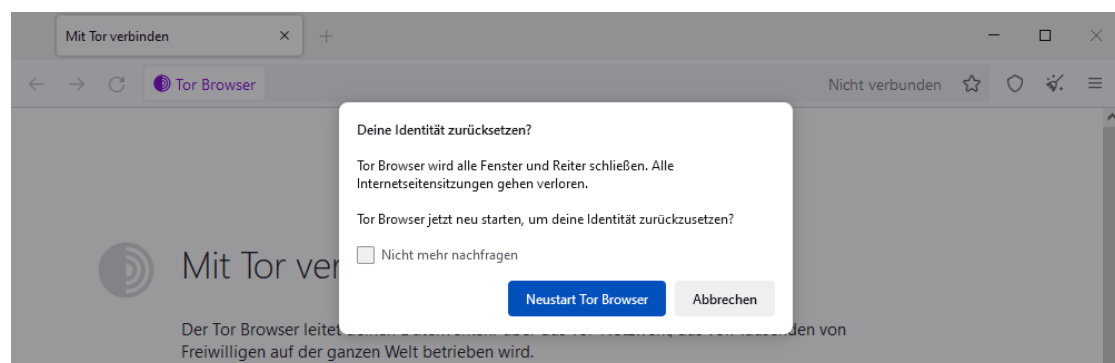


Abbildung 4.1.: Funktion "Neue Identität" des Tor-Browsers

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. [45]



## Chrome

## Brave

### 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
  - 1.2. Google-Suche nach "pfaffenhofen"
2. [www.google.com](http://www.google.com) aufrufen
  - 2.1. Cookies alle akzeptieren
  - 2.2. Google-Suche nach "nanoradar"
3. [www.google.com](http://www.google.com) aufrufen
  - 3.1. Cookies alle akzeptieren
  - 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
  - 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.

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

## 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. [29] Als Virtualisierungssoftware wird für diesen Versuch die kostenlose Software *Oracle VirtualBox VM* verwendet.

Alle VMs besitzen die gleiche, in Tabelle 4.2 dargestellte Basiskonfiguration. Zum Datenaustausch zwischen VM und Analyse-Rechner wird ein *gemeinsamer Ordner* eingerichtet. Somit werden beispielsweise ohne Kontaminierung der VM benötigte Programme auf dem Analyse-Rechner heruntergeladen, in den gemeinsamen Ordner gelegt und offline auf der VM installiert. Auf der VM werden zwei Werkzeuge der Sysinternal-Abteilung von Microsoft 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. [19, 20] 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:

**Firefox** C:\Program Files\Mozilla Firefox\firefox.exe

**Tor** C:\Program Files\Tor Browser\Browser\firefox.exe

**Chrome** \*\*\*TODO Christoph\*\*\*

**Brave** \*\*\*TODO Christoph\*\*\*

## 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. [9] Für diesen Versuch werden folgende Plugins verwendet:

- pslist
- yarascan
- memmap
- filescan
- svcscan

Die genaue Beschreibung der Plugins sowie deren Zusammenhang ist in der Analysephase in Kapitel 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
Oracle VirtualBox	Virtualisierung	7.0.8 r156879
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 [15]. 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. [5] 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. [5, 37] Die PML-Dateien werden mithilfe des gemeinsamen Ordners auf den Analyserechner transportiert.

### Speicherabbilder

Eine der Hauptaufgaben eines Computer-Forensischen-Ermittlers ist die Erstellung und Analyse von direkten Kopien der Speichermedien des untersuchten Rechners. [12] Im Falle der Browserforensik 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. [34] VM-Snapshots können *aufgetaut* werden, wodurch der Zustand des Betriebssystems zum Zeitpunkt der Momentaufnahme wiederhergestellt wird. Bei Oracle 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 [44]. 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> dumpvmmcore -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, 24, 30, 38–40]. Dieses Problem haben Muir, Leimich und Buchanan erkannt und spezifische Zeitpunkte zur Datensammlung vorgeschlagen. Diese ermögliche eine vollständige Analyse des Browserverhaltens vor, während und nach dem Browsing-Szenario. [29]. Wie in Abbildung 4.2 dargestellt, wurde sich an diesen Zeitpunkten für diesen Versuch orientiert.

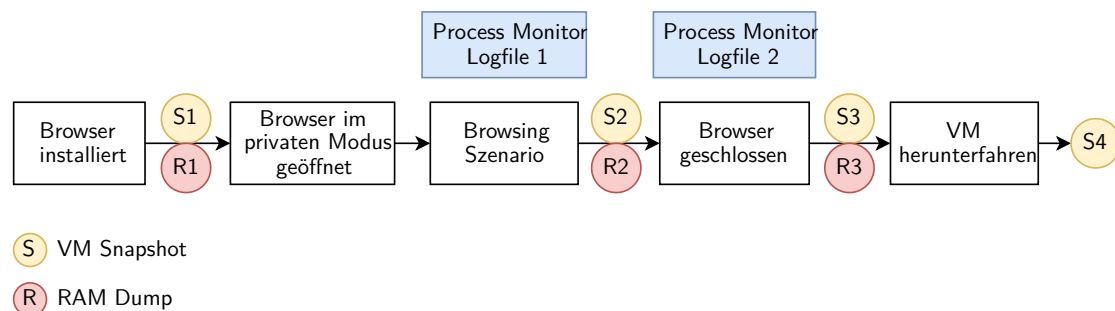


Abbildung 4.2.: Zeitpunkte zur Datensammlung während der Versuchsdurchführung nach [29]

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 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, der für die für Post-Mortem Analyse relevant ist.

**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.

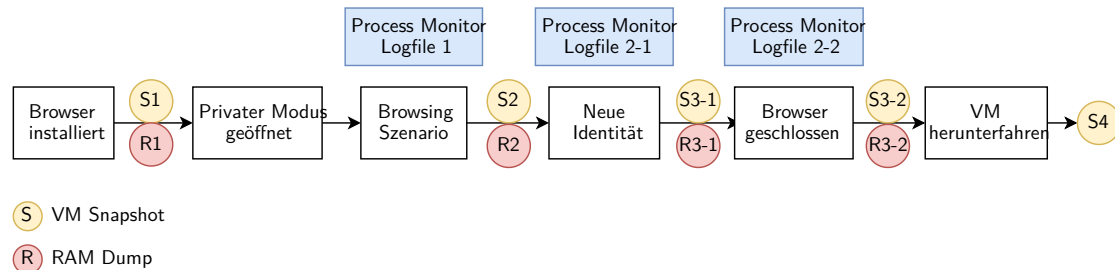


Abbildung 4.3.: Zeitpunkte zur Datensammlung während der Versuchsdurchführung für den Tor-Browser

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.

## 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 Browserforensik 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.

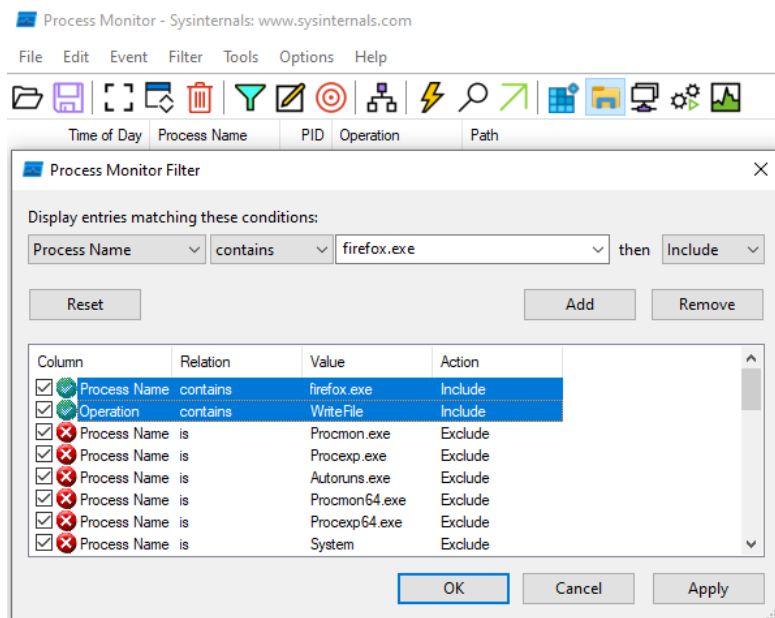


Abbildung 4.4.: Process Monitor Filter für Datei-Schreiboperationen

Anschließend wird als Prozessname der Browserprozess gesetzt:

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



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 Dateieextraktion 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.img windows.filesfan > filesfan.txt`) befindet, wird diese mit dem Volatility Plugin *dumpfiles* (`vol.py -f ram_dump.img -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.

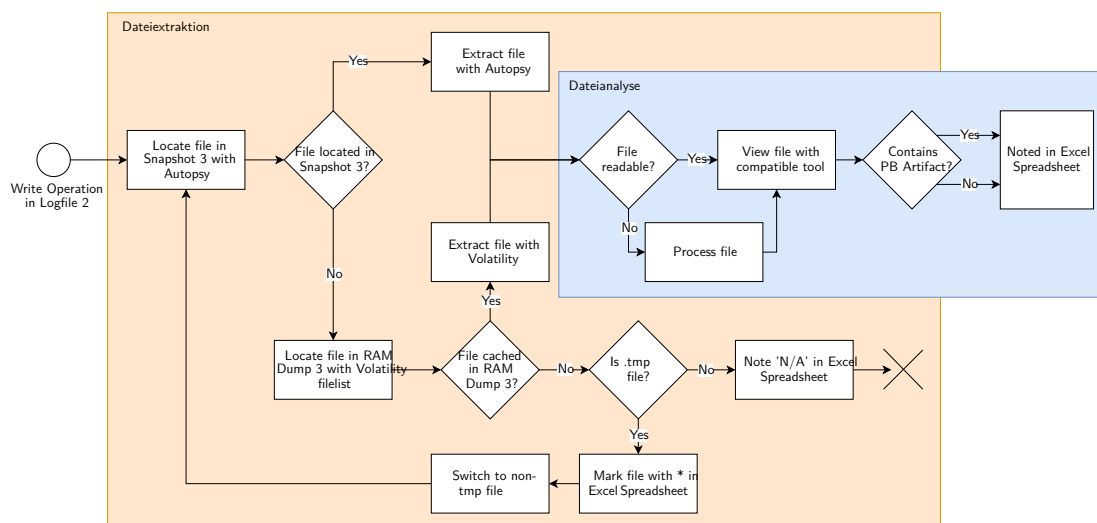


Abbildung 4.5.: Vorgehen zur Dateieextraktion und -analyse

**SQLite-Datenbanken** Eine besondere Rolle unter den Common Locations bei Browsern nehmen SQLite Datenbanken ein. Sie speichern und verwalten Nutzerinformationen, wie Lesezeichen, Browserverlauf, Caches, Cookies in Datenbankdateien, ohne einen separaten Datenbankserver zu benötigen.

Wie in Abbildung 4.6 dargestellt, erfolgt die Dateixtraktion 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 potentielle PB Artefakte zu berücksichtigen, wird der WAL mithilfe der *sqlite3* Befehlszeile (*sqlite3> PRAGMA wal\_checkpoint;*) in die SQLite Hauptdatenbank geschrieben.

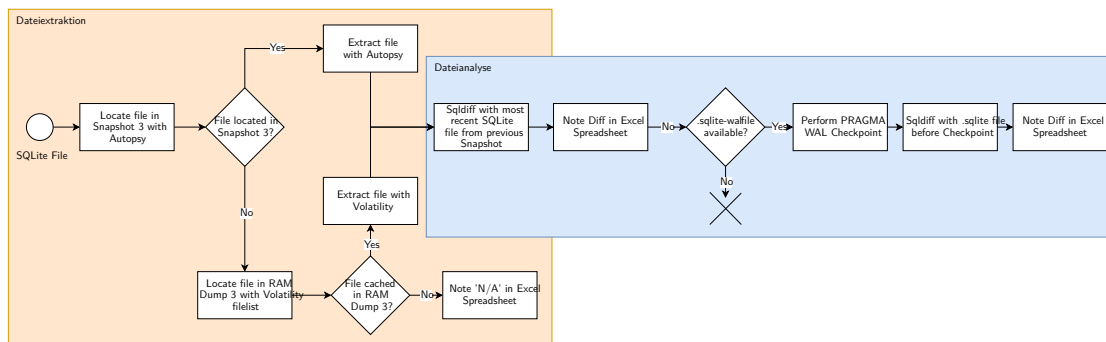


Abbildung 4.6.: Vorgehen zur Dateixtraktion und -analyse von SQLite-Datenbanken

### 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 oder andere Festplattenbereiche, wie beispielsweise unallokierte Speicherbereiche oder der Arbeitsspeicher. Uncommon Locations werden ohne Vorwissen über das Browserverhalten sowie ohne Vorverarbeitung der Dateien mithilfe der *Blackbox-Analyse* untersucht [3]: 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. 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 eingeleseene 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.

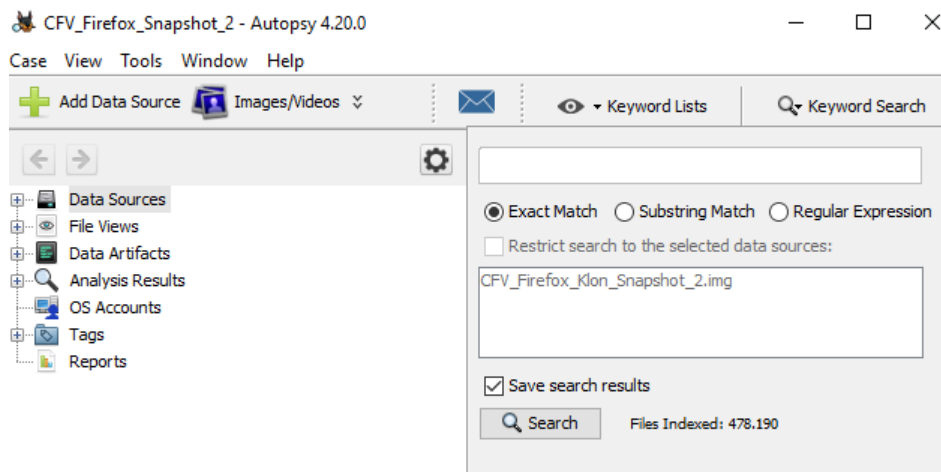


Abbildung 4.7.: Autopsy Funktion zur Stichwortsuche

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

### 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. [21, 24, 37] 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.

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. [38] Alle verwendeten Yara-Regeln

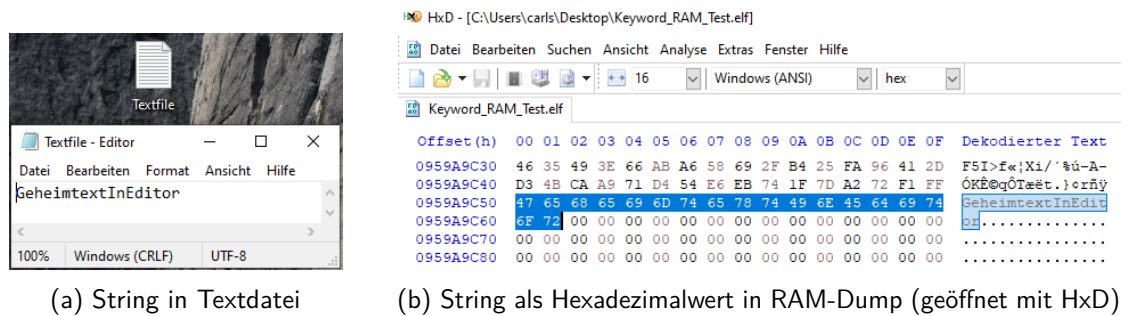


Abbildung 4.8.: Beispiel für ein RAM-Artefakt ohne vorheriges Browsing-Szenario

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.vadyscan -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.

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 Hexadezimaleditor, 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 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,

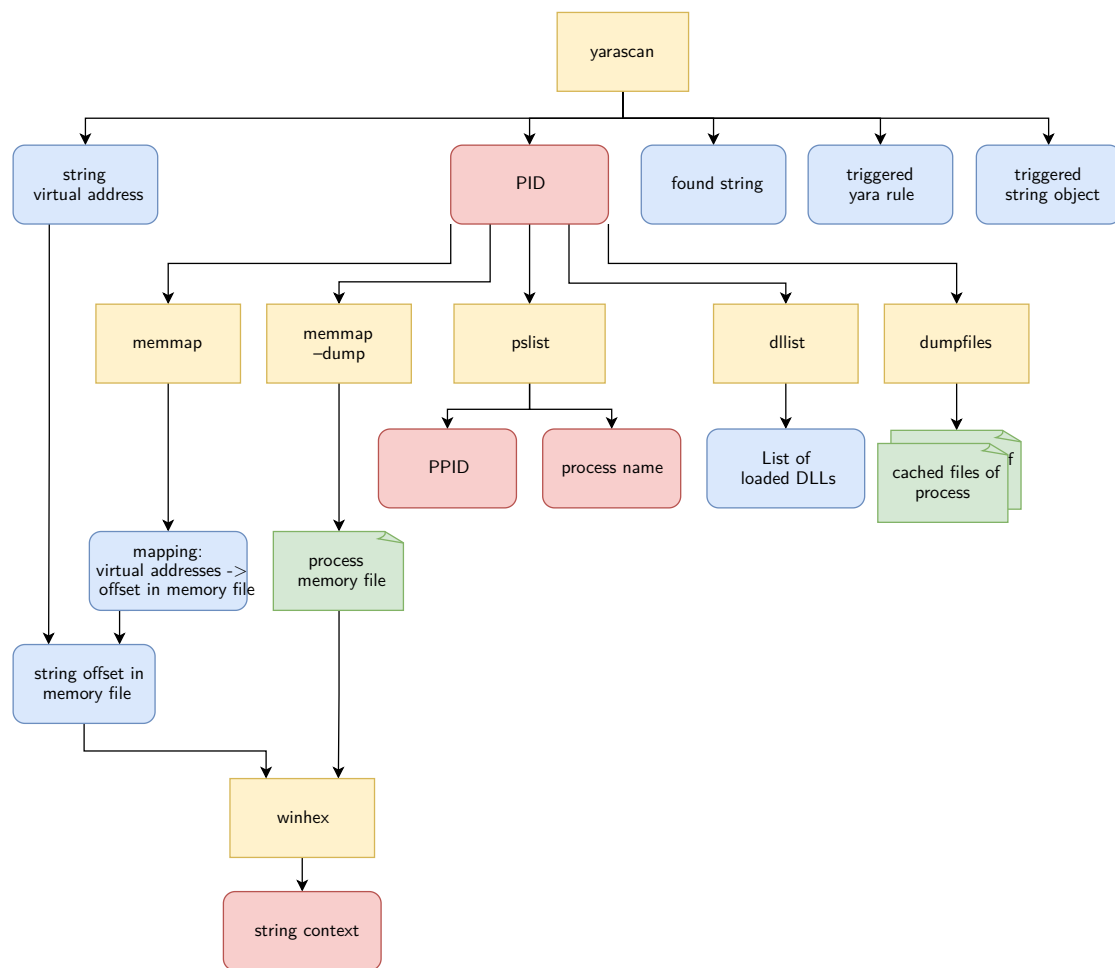


Abbildung 4.9.: Abhängigkeiten der verwendeten Volatility-Plugins yarascan, pslist und memmap

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. [11] Zur Analyse wird jeder Hive aus den Festplatten-Images extrahiert und in eine Registry-Explorer-Sitzung geladen, um eine Stringsuche nach PB-Artefakten durchzuführen.

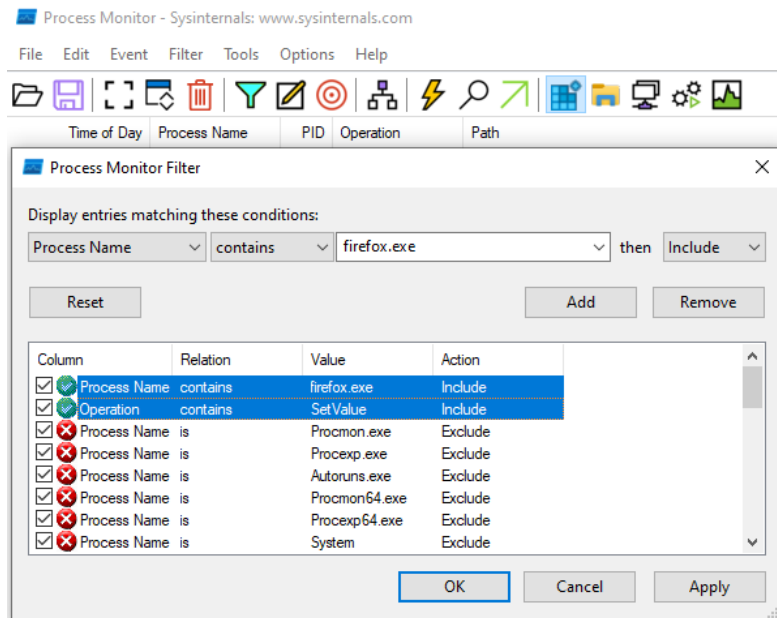


Abbildung 4.10.: Process Monitor Filter für Registry-Schreiboperationen

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. Diese Analyse wird an entsprechenden Stellen durch ausführliche Untersuchungen im Anhang ergänzt.

### 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 potentiellen privaten Browsing Artefakten untersucht. Diese standardmäßigen Speicherorte für Browserartefakte beziehen sich ausschließlich auf die Festplatte geschriebene Dateien. In diesem Versuch wurde gemäß Methodik in Kapitel 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 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 Kapitel 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 Kapitel 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 Kapitel 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 alle 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 gesuchten 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.



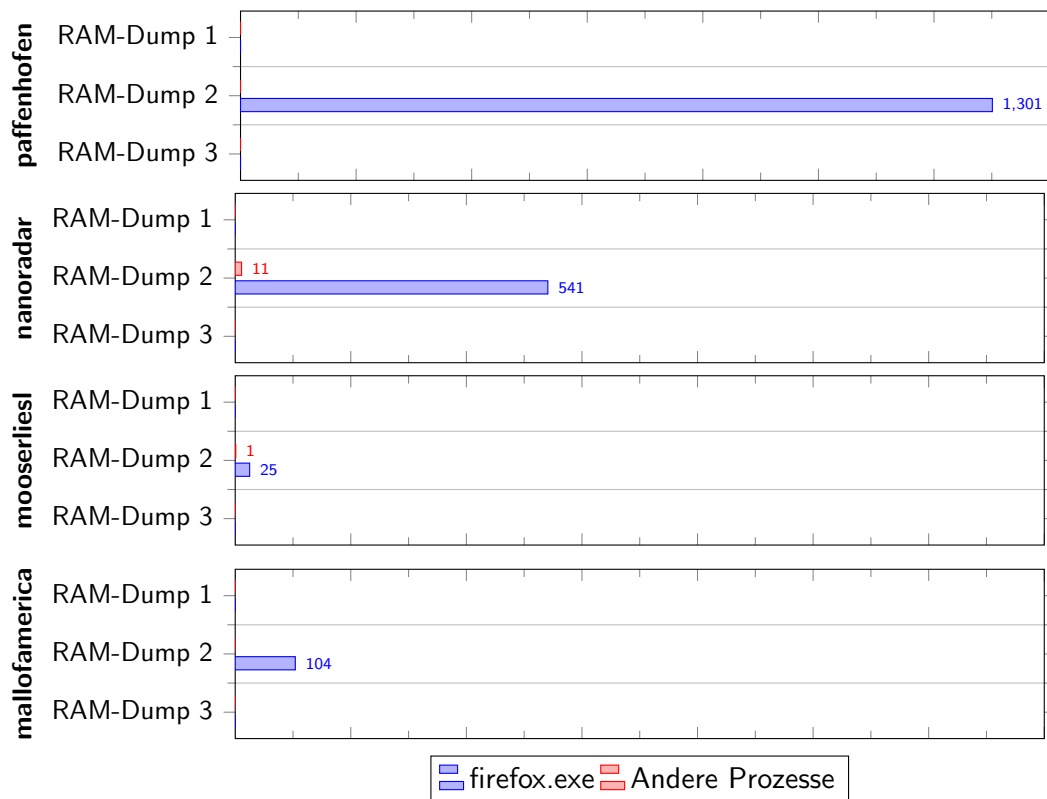


Tabelle 5.1.: Anzahl gefundener Suchbegriffe im Firefox RAM

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 *svcscan* 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. [31] Stattdessen wurde der dritte Snapshot aufgetaut, um im laufenden Windowsbetrieb den Dienst mithilfe des Process Explorers zu identifizieren. Wie in Abbildung 5.1 gezeigt, wurde anhand der PID 2252 der Dienst *DNSCache* ermittelt. Der *DNSCache*-Dienst unter Windows ist ein Teil des Betriebssystems, der für die Übersetzung von Domainnamen in IP-Adressen verantwortlich ist. Der *DNSCache*-Dienst speichert DNS-Anfragen und Antworten temporär, um wiederholte DNS-Anfragen zu beschleunigen. [22] Nach Löschen des *DNSCache*s mit dem Kommandozeilenbefehl `ipconfig /flushdns`, dem Schließen aller Process Monitor Instanzen sowie Beenden des *DNSCache*s Services wurde erneut ein RAM-Dump durchgeführt. Dort wurden keine URL Artefakte mehr gefunden.

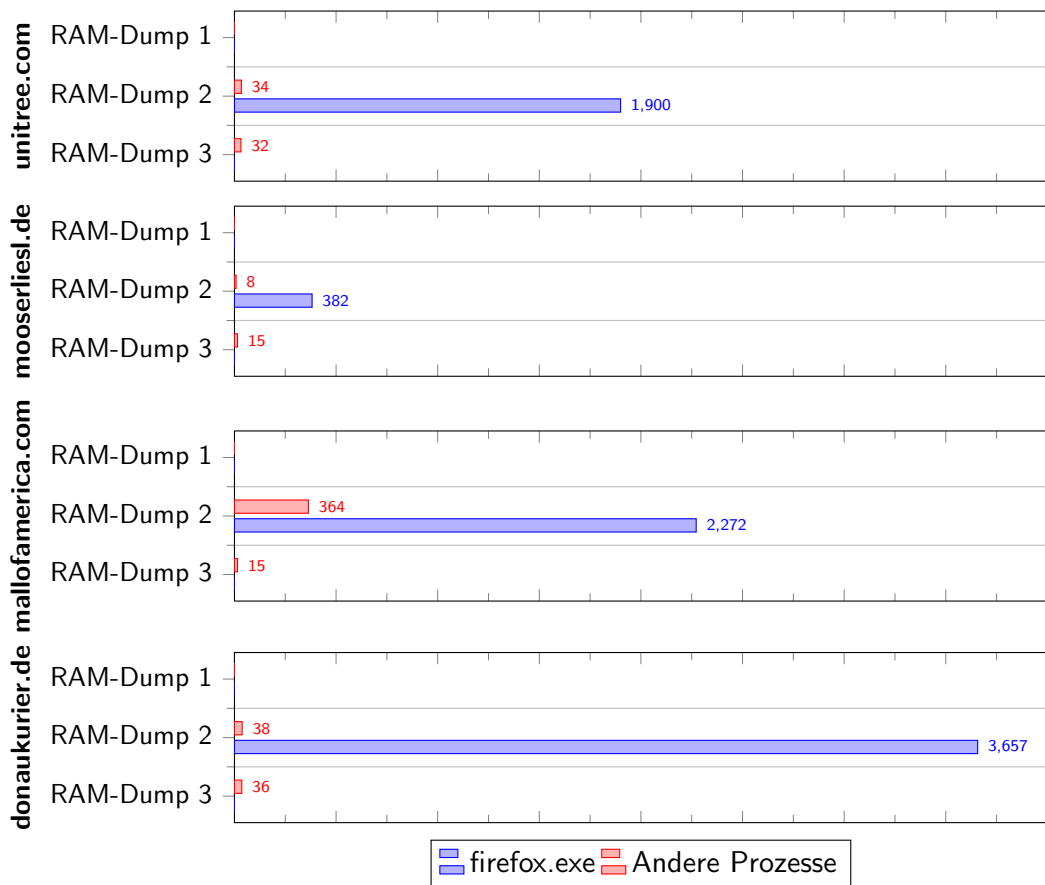


Tabelle 5.2.: Anzahl gefundener URLs im Firefox RAM

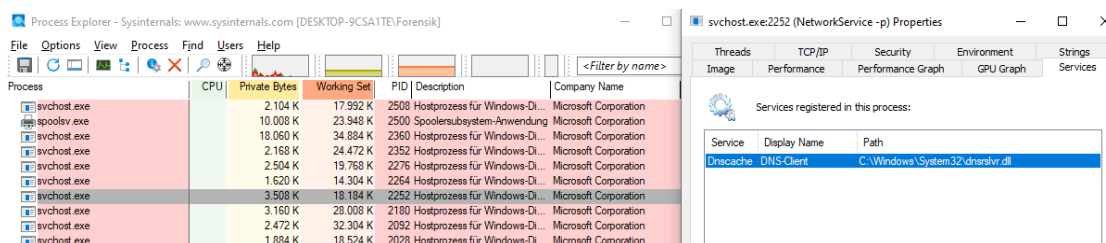


Abbildung 5.1.: Unter dem SVChost-Prozess PID 2252 läuft der DNSCache-Dienst.

**Yara-Regel “E-Mail”** Wie in Abbildung 5.3 gezeigt, wurden ausschließlich nach dem Browsing Szenario mit geöffnetem Firefox Browser (RAM Dump 2) alle E-Mail Artefakte 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

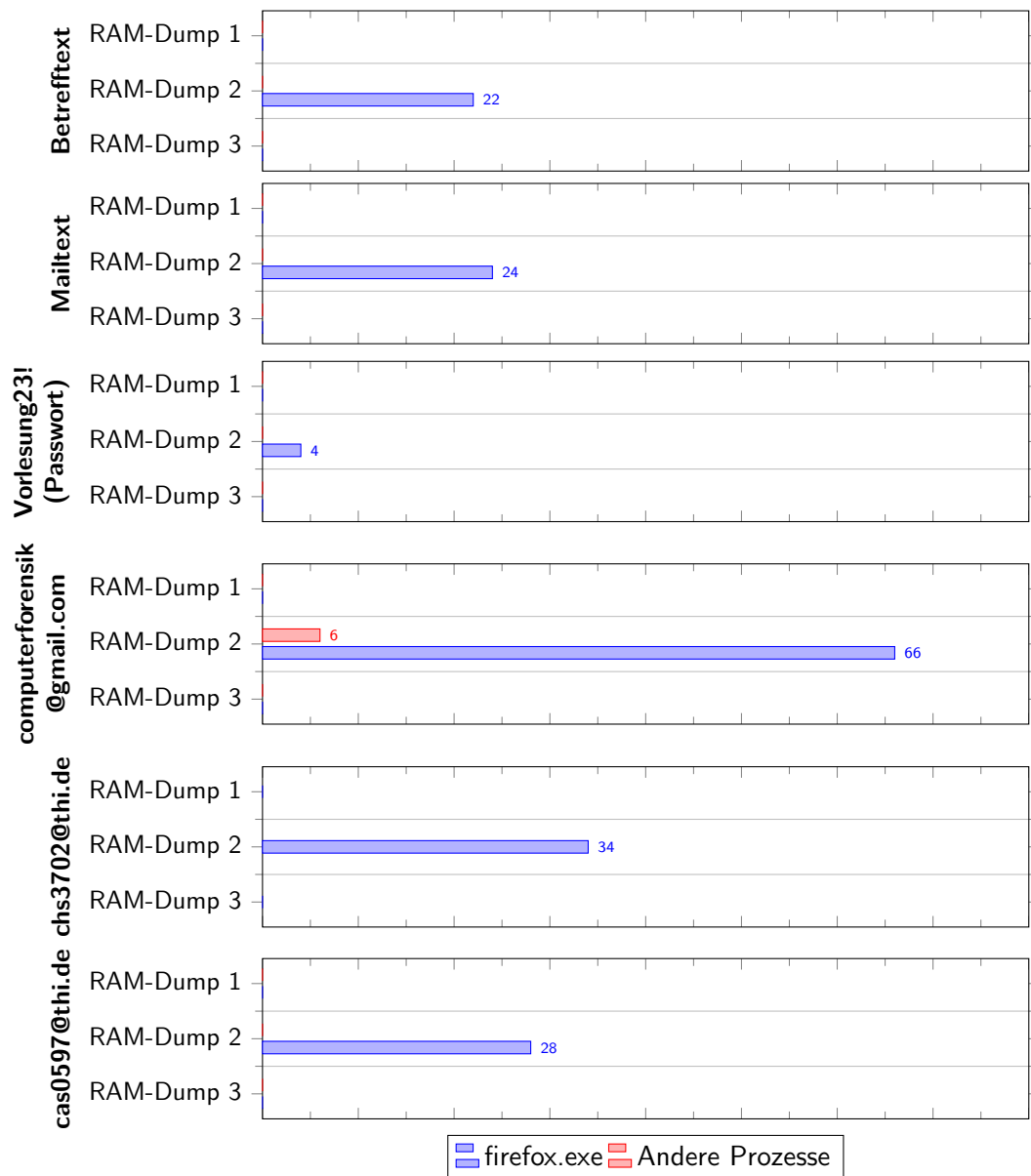


Tabelle 5.3.: Anzahl gefundener E-Mail Artefakte im Firefox RAM

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 5.4 zeigt die virtuellen Speicheradressen der Artefakte aus der Yarascan Ausgabe.

Zu diesen Artefakten wurde gemäß Methodik in Kapitel 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

Virtuelle Speicheradresse	PID	Byte-Offset in extrahierter Speicherseite
0xb9ce29180c8	7420	0x11dd40c8
0x2859f4ffd4e0	7420	0x12e234e0
0x24083b41858	8424	0x583858
0x240840e5b08	8424	0x96bb08

Tabelle 5.4.: Abbildung der virtuellen Speicheradressen im Firefox-RAM der gefundenen Strings auf Byte-Offsets der entsprechenden Speicherseiten

Byte-Offset in der extrahierten Speicherseite des Prozesses ermittelt.

11DD4040	58 02 00 00 08 00 00 00 67 6D 70 41 64 64 6F 6E	X.....gmpAddon	12E23470	00 00 00 00 00 00 00 00 10 02 00 00 34 00 00 00	.....4...
11DD4050	4B 4B 4B 4B 4B 4B 4B 4B DC F9 0E 50 4B 4B 4B 4B	XXXXXXXXXX.PKXXXX	12E23480	00 C8 CC E5 29 02 00 00 00 00 00 00 00 00 00 00	..Eiä).....
11DD4060	58 02 00 00 0D 00 00 00 67 6D 70 44 6F 77 6E 6C	X.....gmpDownl	12E23490	10 02 00 00 27 00 00 00 40 D3 CC E5 29 02 00 00	.....@0iä)...
11DD4070	6F 61 64 65 72 4B 4B 4B 50 C3 FB EA 4B 4B 4B 4B	oaderKKKKFÄÄÄKKKK	12E234A0	00 00 00 00 00 00 00 00 10 02 00 00 2A 00 00 00	.....7.....
11DD4080	58 02 00 00 0D 00 00 00 47 4D 50 44 6F 77 6E 6C	X.....GMPDownl	12E234B0	70 D3 CC E5 29 02 00 00 00 00 00 00 00 00 00 00	p0iä).....
11DD4090	6F 61 64 65 72 4B 4B 4B 0D 6F AE 8A 4B 4B 4B 4B	oaderKKKKDöeSKKKK	12E234C0	00 02 00 00 45 00 00 00 A0 47 7C 1A 55 23 00 00 00	...E...Gj.Uh...
11DD40A0	58 02 00 00 0D 00 00 00 5F 69 73 45 4D 45 45 6E	X....._lseMEEn	12E234D0	F8 DE FF F4 59 28 00 00 50 02 00 00 0C 00 00 00	eByöY(...P.....
11DD40B0	61 62 6C 65 64 4B 4B 4B 5F 4F 0E 73 4B 4B 4B 4B	abledKKKf.O.sKKKK	12E234E0	56 6F 72 6C 65 73 75 6E 67 32 33 21 A2 1D FB FF	Worlesung33(e.üü
11DD40C0	58 02 00 00 0C 00 00 00 56 6F 72 6C 65 73 75 6E	X.....Vorlesun	12E234F0	50 02 00 00 0A 00 00 00 69 64 65 6E 74 69 66 69	F.....identifi
11DD40D0	67 32 33 21 4B 4B 4B 4B F8 35 7D 4B 4B 4B 4B 4B	g234.KKKKs)HKKKK	12E23500	65 72 F0 B8 FA 7F 00 00 50 02 00 00 06 00 00 00	er8,ü...P.....
11DD40E0	58 02 00 00 0F 00 00 00 5F 69 73 41 64 64 6F 6E	X....._isAddon	12E23510	50 61 73 73 77 64 F9 FF 18 96 73 E5 29 02 00 00	Passwdüü...sü)...
11DD40F0	45 6E 61 62 6C 65 64 4B 42 1D 99 C2 4B 4B 4B 4B	EnabledKB."ÄKKKK	12E23520	50 02 00 00 0E 00 00 00 73 65 73 73 69 6F 6E 72	P.....sessionK
11DD4100	58 02 00 00 0F 00 00 00 6D 61 69 6C 2E 67 6F 6F 6F	X.....mail.goo	12E23530	65 73 74 6F 72 65 00 00 10 02 00 00 2E 00 00 00	estore.....
11DD4110	67 6C 65 2E 63 6F 6D 4B 44 47 D9 2D 4B 4B 4B 4B	gle.comKDGÜ-KKKK	12E23540	80 D2 CC E5 29 02 00 00 00 00 00 00 00 00 00 00	e0iä).....
11DD4120	58 02 00 00 10 00 00 00 5F 75 70 64 61 74 65 4C	X.....updateL	12E23550	10 02 00 00 1F 00 00 00 B8 CC E5 29 02 00 00	.....,iä)...
11DD4130	61 73 74 43 68 65 63 6B 43 1F 7D 4B 4B 4B 4B 4B	astCheckC.)KKKKK			
11DD4140	58 02 00 00 10 00 00 00 73 65 63 6F 6E 64 73 53	X.....secondsS			

(a) Byte-Offset 0x11dd40c8

(b) Byte-Offset 0x12e234e0

Abbildung 5.2.: Passwort-Klartext in Firefox Speicherseiten von PID 7420

Wie in Abbildung 5.2 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 Webinhalten verantwortlich, einschließlich HTML, CSS, JavaScript und anderen Medienformaten wie Bildern, Audio und Video. [27] In der gleichen Datei konnten nach dem gefundenen Passwort am Byte-Offset 0x12e234e0 die Strings "Passwd" sowie "sessionrestore" (siehe Common Location *Sessionstore* in Anhang B.1) identifiziert werden.

005837F0	02 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00	.....	0096BA80	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583800	C0 9E EA FF 40 02 00 00 0E 00 00 00 00 00 00 00	ÄzeY@.....	0096BA90	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583810	02 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00	.....AAAAAA	0096BAA0	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583820	02 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00	.....	0096BAB0	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583830	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0096BAC0	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583840	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BAD0	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583850	02 00 00 00 1A 00 00 00 56 00 6F 00 72 00 6C 00	.....V.o.r.i.	0096BAE0	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583860	56 00 73 00 75 00 6E 00 67 00 32 00 33 00 21 00	e.v.u.n.g.2.3.1.	0096BAF0	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583870	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0096BB00	01 00 00 00 38 00 00 00 5F 00 6F 00 72 00 6C 00	...s...V.o.r.i.
00583880	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BB10	56 00 73 00 75 00 6E 00 67 00 32 00 33 00 21 00	e.v.u.n.g.2.3.1.
00583890	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BB20	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00583900	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BB30	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583910	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BB40	00 C8 93 B8 FA 7F 00 00 28 C8 93 B8 FA 7F 00 00	..E".ü... (E".ü...
00583920	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BB50	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00583930	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BB60	04 00 00 00 00 00 00 00 80 9B 47 9B 40 02 00 00	.....e)G"e...
00583940	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BB70	2C 41 7B B8 FA 7F 00 00 00 00 00 00 00 00 00 00	..Äi.ü.....
00583950	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BB80	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583960	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BB90	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA
00583970	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA	0096BBA0	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	AAAAAAAAAAAAAAAA

(a) Byte-Offset 0x583858

(b) Byte-Offset 0x96bb08

Abbildung 5.3.: Passwort-Klartext in Firefox Speicherseiten von PID 8424

Wie in Abbildung 5.3 gezeigt, kann in den Byte-Offsets der gefundenen Passwörter in der Speicherseite der PID 8424 kein sinnvoller Kontext ermittelt werden. Im Gegensatz zur Speicherseite der PID 7420 wird das Passwort dort mit 2 Bytes pro Zeichen enkodiert, was

eine Unicode-Zeichenenkodierung vermuten lässt.

**Yara-Regel “DK-Logo“** Wie in Abbildung 5.5 gezeigt, wurde das im Browsing Szenario geöffnete Donaukurier Logo ausschließlich im zweiten RAM Dump dreimal in Firefox Prozessen gefunden.

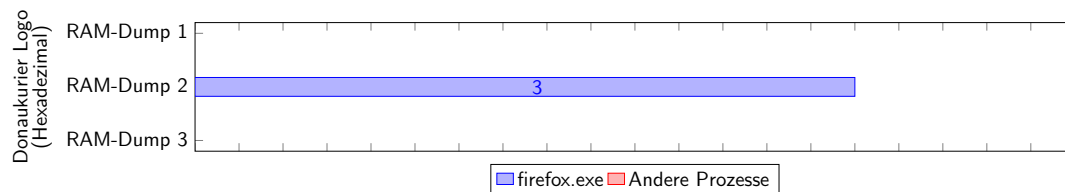


Tabelle 5.5.: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im Firefox RAM

## Registry

Die Analyse der Registry zählt gemäß Methodik in Kapitel 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.2 beschrieben.

## 5.2. Tor

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 Internet-Aktivitäten des Browsing Szenarios zu finden. Bei der Untersuchung der gängigen Speicherorte wurde gemäß der im Kapitel 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 der 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 5.6 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.

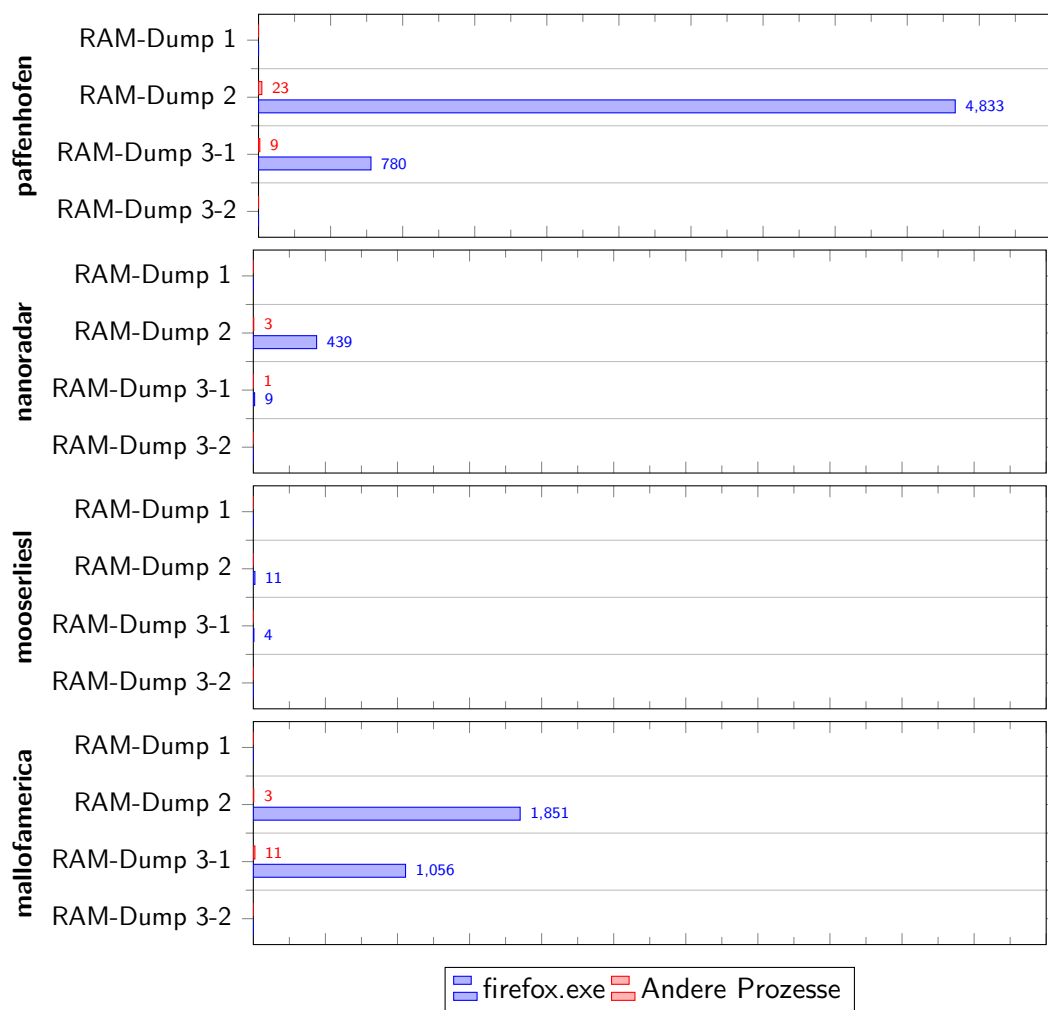


Tabelle 5.6.: Gefundene Suchbegriffe im Tor RAM

**Yara-Regel “URLs”** Ähnlich zur Yara-Regel “Suchbegriffe” wurden, wie in Abbildung 5.7 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.

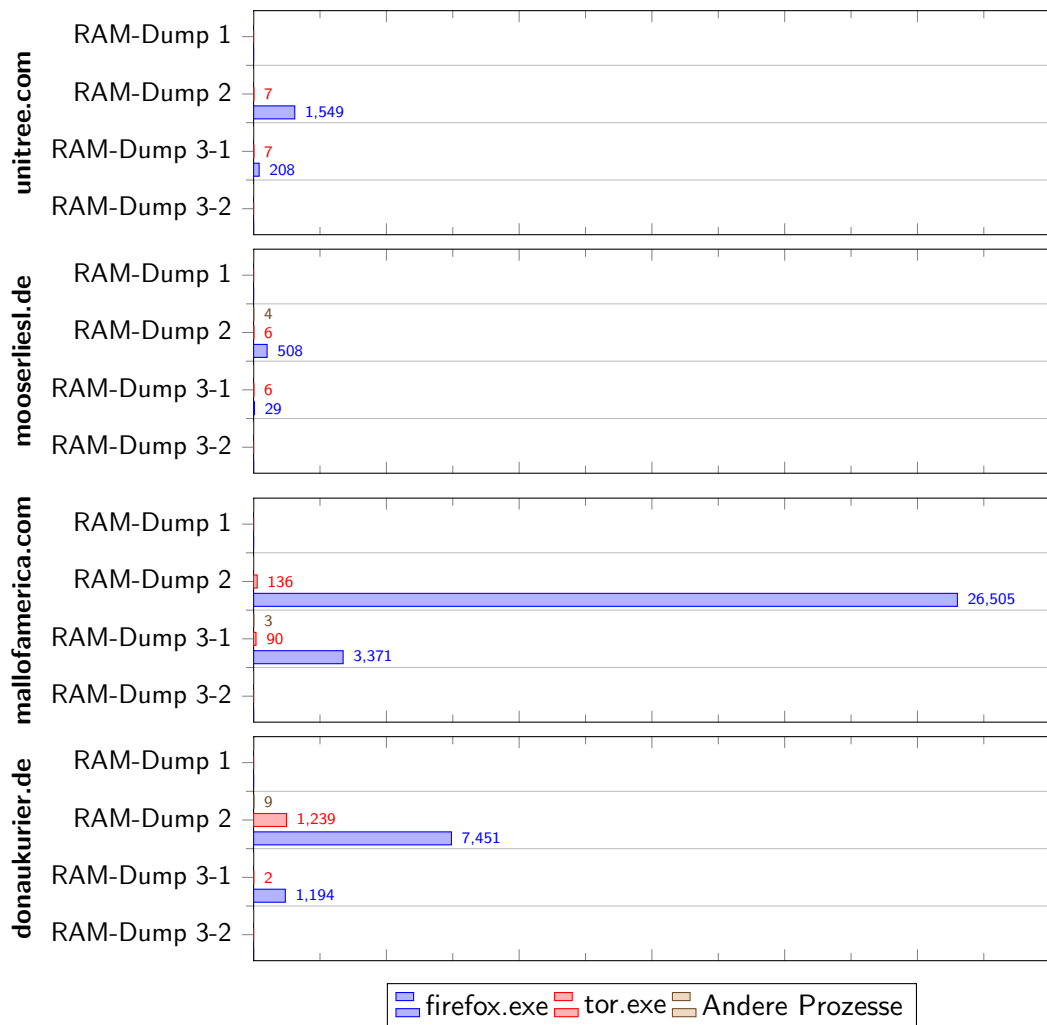


Tabelle 5.7.: Gefundene URLs im Tor RAM



**Yara-Regel “E-Mail”** Nach dem Browsing-Szenario, vor Zuweisung einer “Neuen Identität” (RAM-Dump 2), konnten alle E-Mail Artefakte 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 5.4 dargestellt, wurden die Artefakte ausschließlich in Firefox Prozess gefunden. Wie bei der Analyse der Firefox RAM-Dumps in Kapitel 5.1, wurde das Passwort als Klartext nach dem Browsing-Szenario, vor Zuweisung einer “Neuen Identität” (RAM-Dump 2) gefunden. Das Passwort wurde zweimal im Firefox Prozess mit der PID 708 gefunden. Tabelle 5.8 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.

<pre> 00EA02D0 00 00 00 00 E5 E5 E5 E5 01 00 E5 E5 E5 E5 E5 E5 ...AAAAA.AAAAAA 00EA02E0 C0 94 63 77 FC 7F 00 00 E8 22 C2 E2 B1 02 00 00 A"cwü...e"Äa±... 00EA02F0 E8 22 C2 E2 B1 02 00 00 01 E5 E5 E5 E5 E5 E5 E5 e"Äa±...AAAAA 00EA0300 00 69 49 EC B1 02 00 00 E5 E5 E5 E5 E5 E5 E5 .ill±...AAAAA 00EA0310 02 00 00 00 1A 00 00 00 E6 00 6F 00 72 00 6C 00 .....700x±... 00EA0320 E5 00 73 00 75 00 6E 00 67 00 32 00 33 00 24 00 ..s.u.n.g.2.3.1. 00EA0330 00 00 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ..AAAAA.AAAAAA 00EA0340 00 00 00 00 40 D8 F6 FF 60 EA 00 00 F8 2B 09 00 ...000y"ä...e+... 00EA0350 00 00 00 00 40 D8 F6 FF 60 EA 00 00 F8 2B 09 00 ...000y"ä...e+... 00EA0360 01 00 00 C0 01 00 00 C0 50 3C 4D EC B1 02 00 00 ...Ä...ÄP&lt;M1±... </pre>	<pre> 010F76C0 01 00 00 00 38 00 00 00 43 53 50 5F 69 67 6E 6F ....8....CSP_igno 010F76D0 72 69 6E 67 53 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 Dynamic.AAAAAA 010F76F0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 AAAAAA.AAAAAA 010F7700 A0 07 18 77 FC 7F 00 00 E8 07 18 77 FC 7F 00 00 ..wü...e...wü... 010F7710 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....M1±... 010F7720 04 00 00 00 00 00 00 00 00 40 CF E4 B1 02 00 00 .....700x±... 010F7730 98 2C E5 76 EC 7F 00 00 00 00 00 00 00 00 00 00 ..s.u.n.g.2.3.1. 010F7740 01 00 00 38 00 00 00 E5 00 6F 00 72 00 6C 00 ..s.u.n.g.2.3.1. 010F7750 E5 00 73 00 75 00 6E 00 67 00 32 00 33 00 24 00 ..s.u.n.g.2.3.1. 010F7760 00 00 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ..AAAAA.AAAAAA 010F7770 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ..AAAAA.AAAAAA 010F7780 02 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±...Fidwü... 010F77A0 10 2C D8 E2 B1 02 00 00 10 E7 67 77 FC 7F 00 00 ..,0ä±...0püwü... 010F77B0 A0 9F 0B E4 B1 02 00 00 E5 E5 E5 E5 E5 E5 E5 Y.ä±...AAAAA 010F77C0 01 00 00 38 00 00 00 2F 00 69 00 6E 00 76 00 ...s.../i.n.v. 010F77D0 61 00 6C 00 69 00 64 00 61 00 74 00 69 00 6F 00 a.l.i.d.a.t.i.o. 010F77E0 6E 00 2F 00 6C 00 63 00 73 00 2F 00 63 00 6C 00 n./i.c.s./c.l. 010F77F0 69 00 65 00 6E 00 74 00 00 00 E5 E5 E5 E5 E5 i.e.n.t...AAAAA </pre>
---	---

(a) Byte-Offset 0xea0318

(b) Byte-Offset 0x10f7748

Abbildung 5.5.: Passwort-Klartext in Firefox Speicherseiten von PID 708

Bei Untersuchung des String-Kontexts in Abbildung 5.5, 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. [4]

**Yara-Regel “DK-Logo”** Wie in Abbildung 5.6 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.

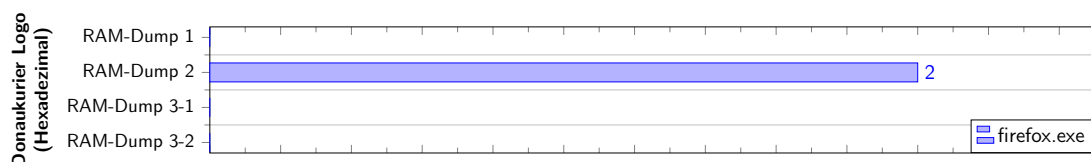


Abbildung 5.6.: Gefundener Hexadezimalwert des Donaukurier-Logos im Tor RAM

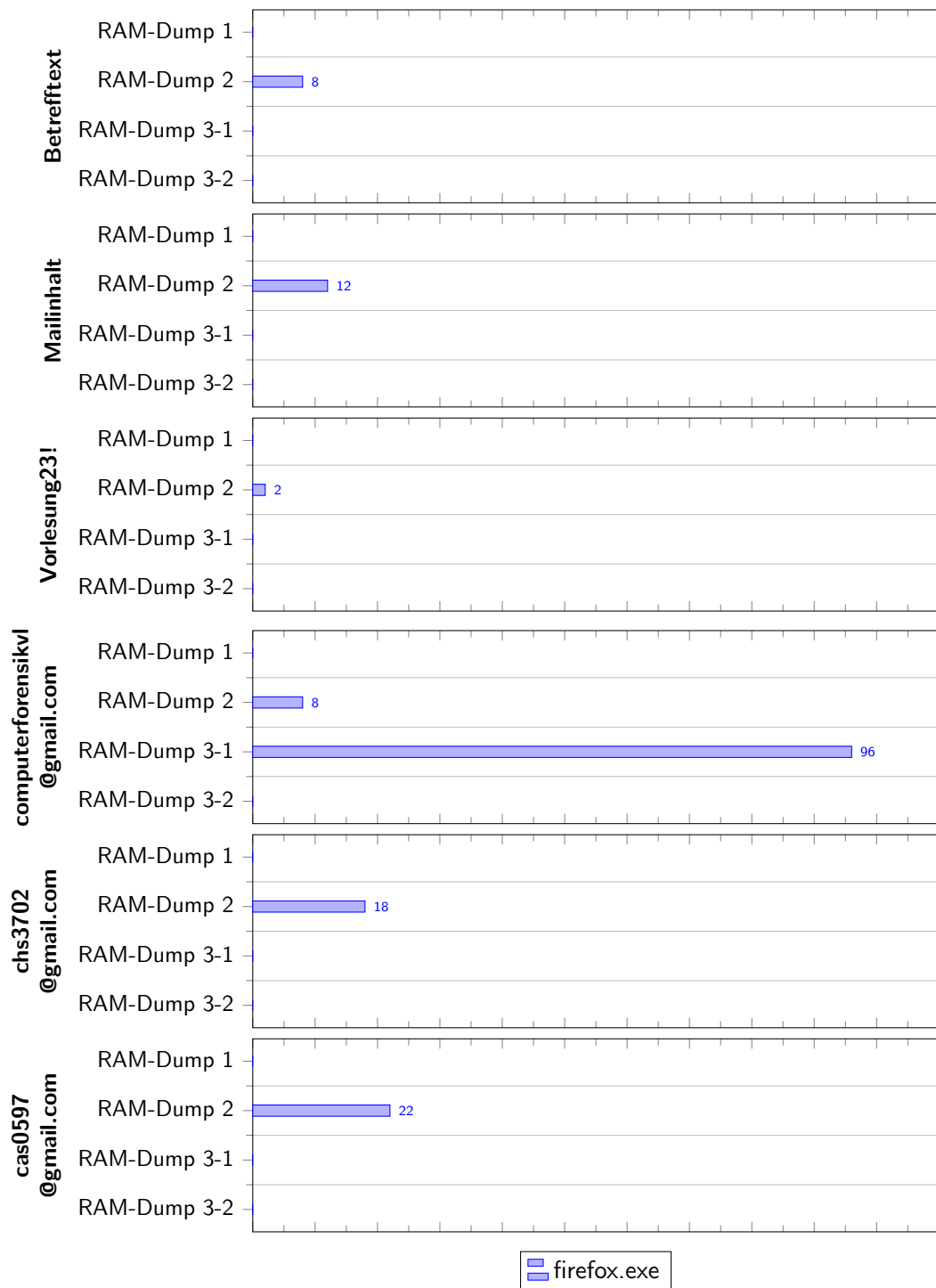


Abbildung 5.4.: Gefundene E-Mail Artefakte im Tor RAM

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

Tabelle 5.8.: Abbildung der virtuellen Speicheradressen der gefundenen Strings im Tor-RAM auf Byte-Offsets der entsprechenden Speicherseiten

## Registry

Wie in der Methodik in Kapitel 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.2 beschrieben.

## 5.3. Chrome

\*\*\* TODO: Christoph \*\*\*

## 5.4. Brave

\*\*\* TODO: Christoph \*\*\*

## 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 Stichwortsuche noch in den automatisch von Autopsy kategorisierten Dateien konnten für keinen Browser 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.

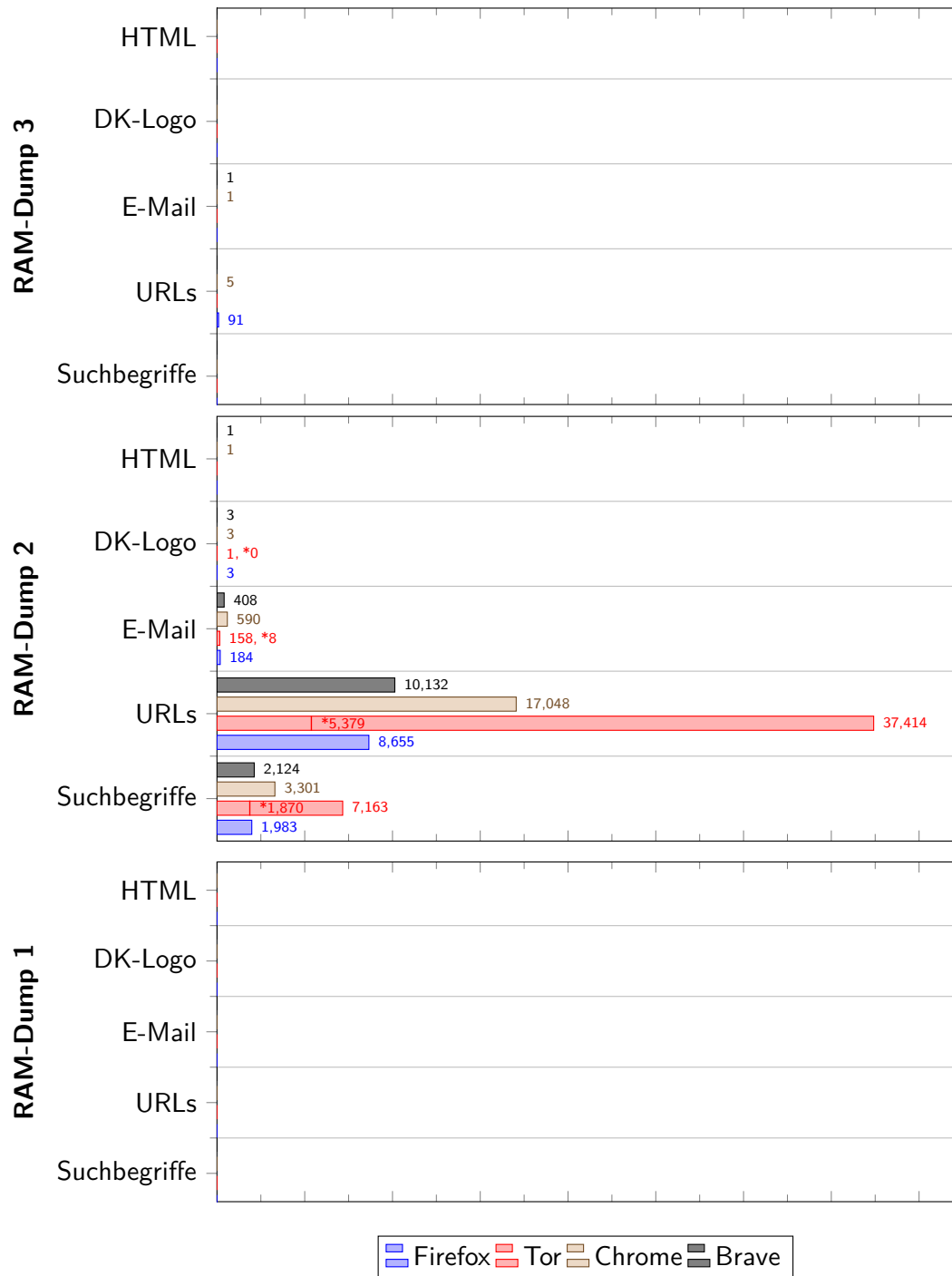


Tabelle 6.1.: Zusammenfassung gefundener Artefakte im RAM der Browser

**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 reduziert sich die hinterlassenen Artefakte nach Zuweisung einer "Neuer Identität" für den Tor-Browsers deutlich, sodass danach weniger Artefakte als bei Firefox vorhanden sind. Wie in Kapitel 4.1.1 erklärt, ermöglicht die "Neue Identität" ermöglicht das Schließen von Tabs und Fenstern, das Löschen von privaten Informationen und die Neukonfiguration der Tor-Netzwerkverbindung. Mit der "Neue 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 einem Zeitpunkt HTML-Artefakte.

**Chrome vs Brave** \*\*\* TODO: Christoph \*\*\*  
(Brave hinterlässt weniger Artefakte als Chrome)

**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 6.2 gezeigte, sogenannte *Gewinner-Tabelle* erstellt [13].

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. Per Mehrheitseinscheid ergeben sich daraus zwei Arten von *Gewinnern*:

Tabelle 6.2.: Gewinner-Tabelle der vier untersuchten Browser

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

- **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. Somit ergeben sich die in Tabelle 6.3 ermittelten vier Gewinner:

Tabelle 6.3.: Ermittelte Gewinner gemäß Gewinner-Tabelle

	Mit "Neuer Identität"	Ohne "Neuer Identität"
<b>Gewinner pro RAM-Dump</b>	Tor	Firefox, Tor
<b>Gewinner pro Kategorie</b>	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. Diskussion

\*\*\* TODO: Christoph \*\*\*



## 8. Fazit

\*\*\* TODO: Christoph \*\*\*

# 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 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 Kapitel 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. Ausschließlich Dateien der Kategorie "Cache" sind im Local Pfad 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.



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

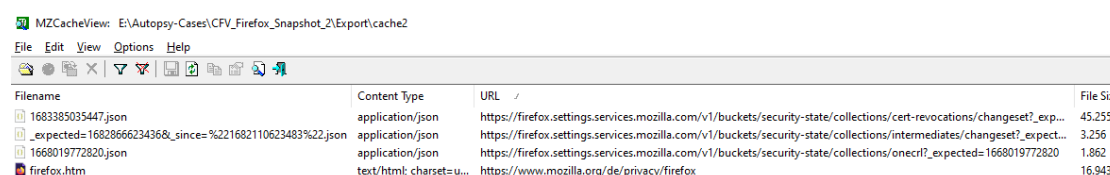
## LOGFILE 1:

Kategorie	Dateiname	Dateistatus	Tool für Analyse	Enthaltene Artefakte
Cache	\cache2\entries\037778A55E1B7E9BED3390289866D09402D6C913	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
	\cache2\entries\1223A0378B8971FA4CD25EA1731C80B2B1676B42	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
	\cache2\entries\250EE2BC03AFF526F1A1C3DB212A79DE3EB60D5E	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
	\jumpListCache\ZKJGVJPzPe7w4w0KwEY0jw==.ico	Datei vorhanden	Windows Foto App	Keine PB Artefakte
	\cache2\entries\D16E4E5DFB15B4C8DE88842C05A47A07C611E01D	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
	\cache2\entries\2F040683A85A4372A73572713C6C52B510854566	Datei vorhanden	MozillaCacheView	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	MozillaCacheView	Keine PB Artefakte
	\cache2\index	Datei vorhanden	MozillaCacheView	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\35decee-d7c6-4820-a381-2dc89ff33c76.tmp	Datei nicht wiederherstellbar	N/A	N/A

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, ein Mechanismus von Firefox zur Überprüfung von Zertifikaten. In keinem der Zertifikate konnten mit HxD private Browsing Artefakte oder besuchte Seiten gefunden werden. [43] Weiterhin befindet sich im Cache das HTML-Dokument der Firefox Datenschutzseite, welche sich beim ersten Start des Browsers automatisch öffnete, siehe Kapitel 4.1.2 Weitere Cache Dateien konnten in keinem Festplatten-Image gefunden werden. Die Indexdatei `\cache2\index` dient als Datenbank im Cache. Sie ermöglicht es



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=168286623436&_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/onecrl?_expected=1668019772820	1.862
firefox.htm	text/html; charset=u...	https://www.mozilla.org/de/privacy/firefox	16.943

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

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 Datenschutzhinweiseseite 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. [9] 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 Browser 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. [16] Wie 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

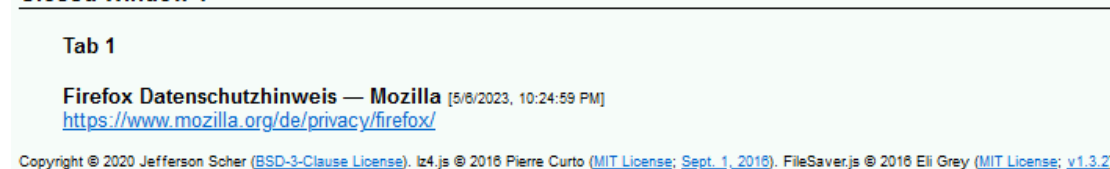


Abbildung B.2.: Firefox Sitzungsdatei recovery.jsonlz4 geöffnet mit dem "Session History Scrounger for Firefox"

**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. [28]

## SQLite Datenbanken

Wie in Kapitel 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.: Veränderte Firefox SQLite-Datenbanken und deren Verwendungszwecke

Datenbank	Gespeicherte Daten [7]
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.
1657114595AmcateinvtiSty.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 Kapitel 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	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	
cookies.sqlite	N/A	Initialisiert (Nur Spaltennamen)		keine Veränderung			
storage.sqlite	N/A						
favicons.sqlite	N/A						
webappsstore.sqlite	N/A	N/A	Initialisiert (Nur Spaltennamen)	keine Veränderung			
formhistory.sqlite	N/A	Initialisiert (Nur Spaltennamen)	keine Veränderung				
1657114595AmcateirvtiSty.sqlite	N/A	Initialisiert ("origin": "chrome")	Binärdaten, keine PB Artefakte				
3870112724rsegmnoittet-es.sqlite	N/A	Initialisiert ("origin": "chrome")	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.

Sowohl nachdem die VM heruntergefahren wurde, (Snapshot 4) als auch nach Durchführung der PRAGMA WAL Checkpoints, entstanden keine 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.

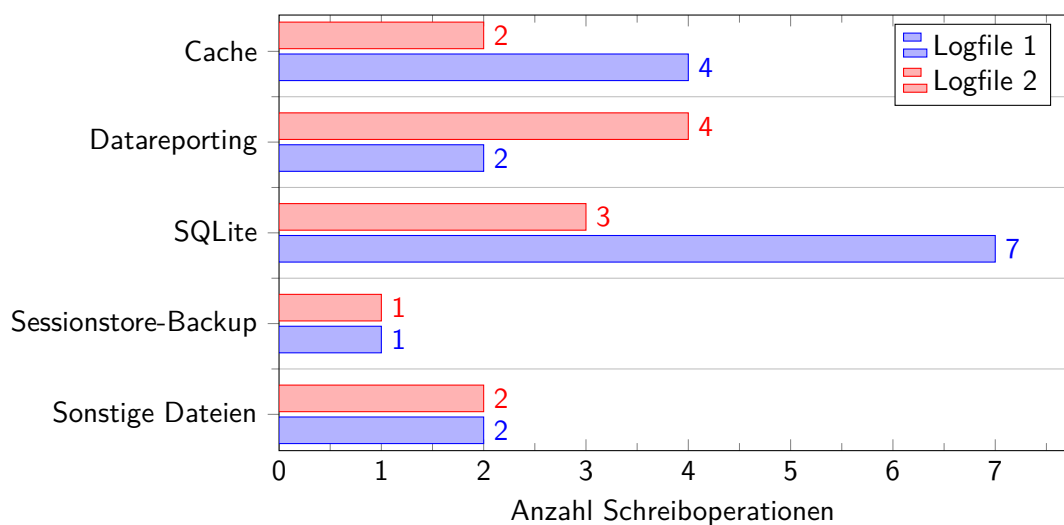


Abbildung B.3.: Firefox: Anzahl Schreiboperationen Logfile 1 vs Logfile 2, geordnet nach Kategorie

## 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 Leesezeichen. In den restlichen Snapshots 2 – 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 – 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=B2C50AD6CB984234A9FE14D681DCB91D&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=des204058d...	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	AI/ON	A=A3B5B679A14D59B0AA027635FFFFFFFF	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: Von 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 Kapitel 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/firefox/	2023-05-06 22:25:00 MESZ	https://www.mozilla.org/privacy/firefox/	Firefox Datenschutzhinweis — Mozilla	Firefox Analyzer	mozilla.org	CPV_Firefox_Klon_Snapshot_3.img	
WebCacheV01.dat			15	https://login.live.com/oauth20_desktop.aspx?lc=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_id=...	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: Von 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 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 - 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 places.sqlite. In keiner der Kategorien konnten private Browsing Artefakte identifiziert

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: Von Autopsy als “Web Categories“ kategorisierte Dateien

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.

## 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.

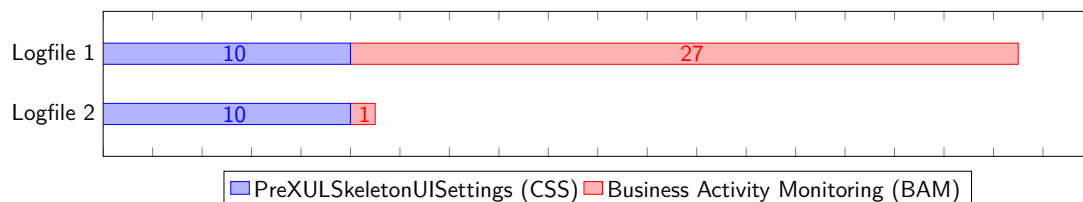


Abbildung B.8.: Firefox Registry “SetValue“ Operationen in den Process Monitor Logfiles 1 und 2

**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. [23]

- 
- 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. [14, 18]

### Stringsuche in Registry Hives

Gemäß Methodik in Kapitel 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 Kapitel 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 durchgeführt wurde, nicht "tor.exe". Obwohl keine der Dateien PB Artefakte enthält, werden zum vollständigen Browserverständnis 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. [27] Bei Untersuchung mit HxD konnten keine PB Artefakte gefunden werden.



Tabelle C.1.: Tor 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 vom Tor-Browser die Datei `\datareporting\data.safe.bin` geschrieben. Sie enthält verschlüsselte und anonyme *Glean* Informationen über die Nutzung des Browsers. [8] 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 dem Browser bei Bedarf den Zustand und die Einstellungen wiederherzustellen. [7] 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. [26]

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 die Süddeutsche Zeitung. SecureDrop ist ein Open-Source-Softwaretool, 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. [41]

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. [10] In der aktuellen Tor-Browser-Version konnten keine private Browsing Artefakte gefunden werden.

## SQLite Datenbanken

Anhand der Process Monitor Logfiles ist erkennbar, dass Tor die gleichen SQLite Datenbanken wie Firefox aus Kapitel B.1 verwaltet und schreibt.

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.: Veränderung der Tor-Browser 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		keine Veränderung					
storage.sqlite	N/A	(Nur Spaltennamen)							
favicons.sqlite	N/A	Initialisiert (Tor-Standardseiten, Präfix: fake-favicon-uri*)							
webappsstore.sqlite	N/A	Initialisiert							
formhistory.sqlite	N/A	(Nur Spaltennamen)							
1657114595AmcateirvtiSty.sqlite	N/A								
3870112724rsegmnoittet-es.sqlite	N/A	Initialisiert ("origin": "chrome")							

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.

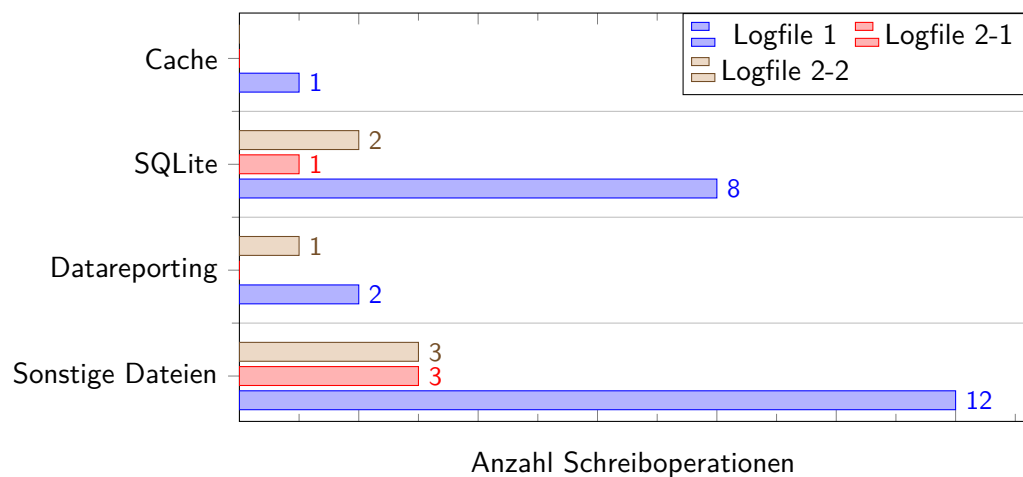


Abbildung C.1.: Tor: Anzahl Schreiboperationen Logfile 1 vs Logfile 2-1 vs Logfile 2-2, geordnet nach Kategorie

## C.2. Uncommon Locations

### Analyse mit Autopsy - Kategorisierte Dateien

Wie bei Firefox in Kapitel 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 Leesezeichen 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: Von 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: Von 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: Von 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: Von 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 Kapitel 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 Kapitel B.2: 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 Anzahl der Registry “SetValue“-Operationen bei Logfile 2 und 3 kontinuierlich ab.

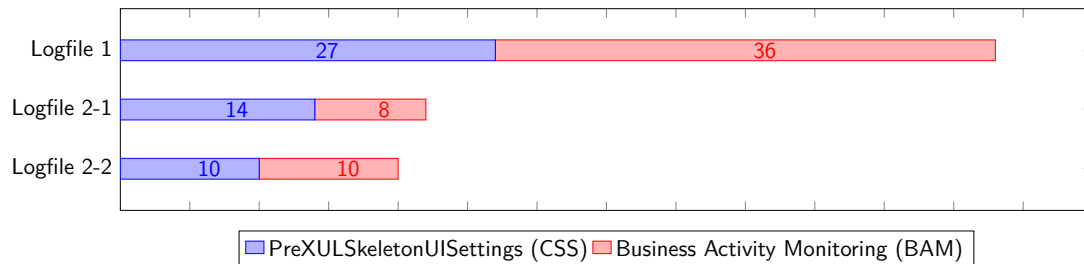


Abbildung C.6.: Tor Registry “SetValue“ Operationen in den Process Monitor Logfiles 1, 2-1 und 2-2

### Stringsuche in Registry Hives

Bei Betrachtung der Registry als Uncommon Locations, wurden die in Tabelle 4.4 im Kapitel 4.3.3 aufgelisteten Registry-Hives mithilfe des Registry Explorers untersucht. Weder in den System-Hives noch in den User-Hives konnte in keinem Festplatten-Image PB Artefakte identifiziert werden.

# 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] 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).
- [5] Hasan Fayyad-Kazan u. a.  
"Forensic analysis of private browsing mechanisms: Tracing internet activities".  
In: (2021).
- [6] 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.
- [7] GitHub. *Firefox Data Stores*. 8.04.2019.  
URL: <https://mozilla.github.io/firefox-browser-architecture/text/0010-firefox-data-stores.html>.
- [8] GitHub. *GitHub - mozilla/glean: Modern cross-platform telemetry*. 5.06.2023.  
URL: <https://github.com/mozilla/glean>.
- [9] GitHub. *GitHub - volatilityfoundation/volatility3: Volatility 3.0 development*.  
5.06.2023. URL: <https://github.com/volatilityfoundation/volatility3>.
- [10] 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>.
- [11] 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>.

- 
- [12] Nihad A Hassan. *Digital forensics basics: A practical guide using Windows OS*. Apress, 2019.
- [13] Graeme Horsman u. a. "A forensic examination of web browser privacy-modes". In: *Forensic Science International: Reports* 1 (2019), S. 100036.
- [14] InfoSec Notes. *Artefacts overview*. 5.06.2023. URL: [https://notes.qazeer.io/dfir/windows/\\_artefacts\\_overview](https://notes.qazeer.io/dfir/windows/_artefacts_overview).
- [15] 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.
- [16] Jefferson Scher. *Session History Scrounger for Firefox (with lz4 support) — Fx File Utilities*. 29.11.2020. URL: <https://www.jeffersonscher.com/ffu/scrounger.html>.
- [17] 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.
- [18] MandiOhlinger. *Business Activity Monitoring (BAM) - BizTalk Server*. 5.06.2023. URL: <https://learn.microsoft.com/de-de/biztalk/core/business-activity-monitoring-bam>.
- [19] Markruss. *Prozess-Explorer - Sysinternals*. 5.06.2023. URL: <https://learn.microsoft.com/de-de/sysinternals/downloads/process-explorer>.
- [20] Markruss. *Prozessmonitor - Sysinternals*. 5.06.2023. URL: <https://learn.microsoft.com/de-de/sysinternals/downloads/procmon>.
- [21] 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.
- [22] 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>.
- [23] 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).
- [24] 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.
- [25] Mozilla. *Geschichte des Mozilla-Projekts*. 5.06.2023. URL: <https://www.mozilla.org/de/about/history/>.

- 
- [26] Mozilla Support. *AlternateServices.txt: what does this file do?* 26.10.2020.  
URL: <https://support.mozilla.org/en-US/questions/1310302>.
- [27] MozillaWiki. *StartupCache - MozillaWiki*. 5.06.2023.  
URL: <https://wiki.mozilla.org/StartupCache>.
- [28] 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).
- [29] Matt Muir, Petra Leimich und William J Buchanan.  
"A forensic audit of the tor browser bundle".  
In: *Digital Investigation* 29 (2019), S. 118–128.
- [30] 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.
- [31] 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>.
- [32] Junghoon Oh, Seungbong Lee und Sangjin Lee.  
"Advanced evidence collection and analysis of web browser activity".  
In: *Digital investigation* 8 (2011), S62–S70.
- [33] 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.
- [34] Oracle. *1.10. Snapshots*. 2020. URL: <https://docs.oracle.com/en/virtualization/virtualbox/6.0/user/snapshots.html>.
- [35] Daniel Perdices u. a.  
"Web browsing privacy in the deep learning era: Beyond VPNs and encryption".  
In: *Computer Networks* 220 (2023), S. 109471.
- [36] Digvijaysinh Rathod. "Darknet forensics". In: *future* 11 (2017), S. 12.
- [37] 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.
- [38] Huwida Said u. a. "Forensic analysis of private browsing artifacts".  
In: *2011 International Conference on Innovations in Information Technology*. IEEE. 2011, S. 197–202.
- [39] Priya P Sajan u. a. "Tor Browser Forensics". In: *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12.11 (2021), S. 5599–5608.



- 
- [40] 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.
- [41] SecureDrop. *Share and accept documents securely*. 5.06.2023. URL: <https://securedrop.org/>.
- [42] 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/>.
- [43] 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/>.
- [44] TILT. *Memory Forensics of a Virtualbox VM* · TILT. 25.03.2023. URL: <https://kollee.github.io/posts/memory-forensics-of-a-virtualbox-vm/>.
- [45] Tor. *Das Tor Project | Privatsphäre & Freiheit Online*. 24.05.2023. URL: <https://www.torproject.org/de/download/>.
- [46] 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.