

Vergleich und Analyse des privaten Modus verschiedener Browser

Computer-Forensik und Vorfallsbehandlung

Carl Schünemann

Christoph Sell

29.08.2025

Inhaltsverzeichnis

1	Einleitung	1
2	Theoretischer Hintergrund	4
2.1	Private Browsing	4
2.2	Angreifermodell	4
2.3	Private Browsing Artefakte	4
3	Ziel der Arbeit	5
4	Methodik	6
4.1	Vorbereitung	6
4.1.1	Browserauswahl	6
4.1.2	Browsing Szenario	7
4.2	Datensammlung	11
4.3	Datenanalyse	13
4.3.1	Common Locations	13
4.3.2	Uncommon Locations	15
4.3.3	Registry	16
5	Ergebnisse	18
5.1	Firefox	18
5.2	Tor	25
5.3	Chrome	30
5.4	Brave	30
6	Vergleich der Browser	31
7	Diskussion	32
8	Fazit	34
	Appendices	35
	Literaturverzeichnis	49
	Literatur	49

1 Einleitung

Steigende Beliebtheit private Browsing: [10] ■ Die Verwendung von PB wurde als die beliebteste Form der Online-Privatsphäre weltweit identifiziert. ■ Aufgrund der gestiegenen Sensibilität und Öffentlichkeit für den Schutz der Privatsphäre und die Regulierung des eigenen digitalen Fußabdrucks im Internet werden PB-Technologien wahrscheinlich häufiger auf den Geräten der Nutzer eingesetzt. ■ Auch wenn es schwierig ist, endgültige Nutzungsstatistiken für solche Aktionen zu erstellen, bietet der Konsens über den Online-Datenschutz einen Einblick. Im Jahr 2016 wurde die Verwendung eines PB-Fensters als die weltweit beliebteste Form der Online-Datenschutzmaßnahme identifiziert [1]. Allein in den USA nutzen Berichten zufolge rund 33 % der Nutzer ein PB-Fenster, wobei über 70 % zugeben, ihren Internetverlauf zu löschen [2]. - Eine umfassende Studie von Montasari und Peltola (2015) ergab, dass der Erfolg des privaten Modus bei verschiedenen Browsern sehr unterschiedlich ist

Vermeintliche Privatheit beim Browsen: [19] > Verschlüsselung ■ Datenschutz und Datenverwendung sind Hauptbedenken der Internetnutzer geworden [5]. ■ Fragen wie welche Daten von Unternehmen genutzt werden, mit wem sie geteilt werden und wie wertvoll sie sind, sind heute wichtige Themen. ■ Daher versuchen Benutzer, sich so weit wie möglich zu schützen, insbesondere durch Begrenzung der Datenweitergabe. ■ Lösungen wie Verschlüsselung auf HTTP-Ebene [6] und auf DNS-Ebene [7,8] sind Standard geworden und werden den Großteil des Datenverkehrs in den nächsten Jahren abdecken. ■ Sie können jedoch nur End-to-End-Konversationen verschlüsseln, d.h. IP- und TCP- oder UDP-Informationen sind immer noch verfügbar. > VPNs ■ Eine weitere beliebte Methode zum Schutz der Privatsphäre und zur Vermeidung von Datenverwendung ist die Verwendung von Virtual Private Networks (VPNs). ■ Obwohl VPNs immer beliebter geworden sind und die meisten von ihnen den IP-Verkehr verschlüsseln und tunneln können, kann der Datenverkehr tatsächlich am Endpunkt des VPNs überwacht werden. ■ Dies bedeutet, dass Akteure zwischen dem VPN-Servernetzwerk und dem Website-Server die Daten sehen und nutzen können. ■ Der VPN-Anbieter kann sogar noch weiter gehen, da er auch die Identität des Clients kennt. > Tor und Brave: 1. Die Endpunkte der verschlüsselten Verbindungen, die von Tor und Brave hergestellt werden, nicht vollständig verschlüsselt sind. Daher können einige Informationen, wie z.B. die IP-Adresse des Benutzers, an den letzten Servern in der Kette sichtbar sein. 2. Einige Tor-Ausgangsknoten haben in der Vergangenheit die Aktivität ihrer Benutzer ausspioniert, um Daten zu sammeln und möglicherweise zu verkaufen. 3. Obwohl die Verwendung von Brave und Tor dazu beitragen kann, dass Benutzer online nicht nachverfolgt werden, werden sie nicht vor Verfolgung durch andere Methoden wie Standortverfolgung oder Geräte-Fingerprinting geschützt. 4. Schließlich können auch andere Schwachstellen in der Implementierung oder Konfiguration von Tor oder Brave dazu führen, dass Daten durchsickern und somit die Privatsphäre der Benutzer kompromittiert wird.

Immer mehr Kriminelle im Internet [12]: > Das Internet und seine Nutzer wachsen ständig, aber auch die Anzahl organisierter Verbrechen und illegale Aktivitäten nehmen zu.

“Webbrowser immer beliebter bla bla ...“ [11] > Webbrowser sind heutzutage ein wichtiger Werkzeug für Online-Aktivitäten wie Online-Banking, Online-Shopping und soziale Netzwerke.

Immer mehr Internet-Nutzer:[11] ■ Im Jahr 2019 gab es laut [13] fast 4,5 Milliarden Internetnutzer.

Zunehmende Bestrebungen nach Privatheit erschwert forensische Ermittlungen [15] > Zunehmende Verwendung von verschlüsselten Daten in der Dateispeicherung und Netzwerkkommunikation erschwert Ermittlungen. > Besonders schwierig ist das Tor-Protokoll, das sich auf den Schutz der Privatsphäre des Nutzers konzentriert. > Tor-Browser hinterlässt digitale Artefakte, die von Ermittlern genutzt werden können.

Motivation Portable Browser [7] ■ Die Beliebtheit von tragbaren Webbrowsern nimmt aufgrund ihrer bequemen und kompakten Natur sowie des Vorteils, dass Daten einfach über einen USB-Stick gespeichert und übertragen werden können, zu. ■ Entwickler arbeiten an Webbrowsern, die tragbar sind und zusätzliche Sicherheitsfunktionen wie den privaten Modus-Browsing, eingebaute Werbeblocker usw. bieten. ■ Die erhöhte Wahrscheinlichkeit, tragbare Webbrowser für schädliche Aktivitäten zu nutzen, ist das Ergebnis von Cyberkriminellen, die der Ansicht sind, dass bei der Verwendung von tragbaren Webbrowsern im privaten Modus keine digitalen Fußabdrücke hinterlassen werden. ■ Das Forschungspapier zielt darauf ab, eine vergleichende Studie von vier tragbaren Webbrowsern, nämlich Brave, TOR, Vivaldi und Maxthon, zusammen mit verschiedenen Speichererfassungstools durchzuführen, um die Menge und Qualität der aus dem Speicherauszug wiederhergestellten Daten in zwei verschiedenen Bedingungen zu verstehen, nämlich wenn die Browser-Tabs geöffnet und geschlossen waren, um forensische Ermittler zu unterstützen.

Private Browsing Motivation und Ausnutzen von Kriminellen: [14] ■ Webbrowser werden täglich genutzt, um verschiedene Online-Aktivitäten durchzuführen. ■ Webbrowser speichern eine große Menge an Daten über Benutzeraktivitäten, einschließlich besuchter URLs, Suchbegriffen und Cookies. ■ Private Browsing-Modi wurden entwickelt, um Benutzern das Surfen im Internet zu ermöglichen, ohne Spuren zu hinterlassen. ■ Dies kann von Kriminellen ausgenutzt werden, um ihre Aktivitäten zu verschleiern. ■ Experimente werden auf jeder Browser-Modus durchgeführt, um zu untersuchen, ob sie Spuren auf der Festplatte oder im Arbeitsspeicher hinterlassen.

Motivation Private Browsing mit Portablen Browsern: [18] ■ Das Internet ist ein unverzichtbares Werkzeug für alltägliche Aufgaben. ■ Neben der üblichen Nutzung wünschen sich Benutzer die Möglichkeit, das Internet auf private Weise zu durchsuchen. ■ Dies kann zu einem Problem führen, wenn private Internetsitzungen vor Computerermittlern verborgen bleiben müssen, die Beweise benötigen. ■ Der Schwerpunkt dieser Forschung liegt darauf, verbleibende Artefakte aus privaten und portablen Browsing-Sitzungen zu entdecken. ■ Diese Artefakte müssen mehr als nur Dateifragmente enthalten und ausreichend sein, um eine positive Verbindung zwischen Benutzer und Sitzung herzustellen. ■ In den letzten 20 Jahren ist das Internet für alltägliche Aufgaben, die mit stationären und mobilen Computergeräten verbunden sind, drastisch unverzichtbar geworden. ■ Benutzer wünschen sich neben der üblichen Internetnutzung auch Privatsphäre und die Möglichkeit, das Internet auf private Weise zu durchsuchen. ■ Aus diesem Grund wurden neue Funktionen für das private Browsen entwickelt, die von allen gängigen Webbrowsern unterstützt werden. ■ Unsere Forschung konzentriert sich auf die Entdeckung von Informationen von lokalen Maschinen, da die meisten Computeruntersuchungen auf der Suche und Beschlagnahme von lokalen Speichergeräten beruhen. ■ Artefakte aus privaten und portablen Browsing-Sitzungen wie Benutzernamen, elektronische Kommunikation, Browsing-Verlauf, Bilder und Videos können für einen Computerermittler signifikante Beweise enthalten. ■ Wir

werden auch flüchtige Daten analysieren, die in einer gängigen Incident-Response-Umgebung verfügbar wären.

Schwachstellen in Browsern, durch die Daten "lecken" [24] ■ Private browsing ist seit 2005 eine beliebte Datenschutzfunktion in allen gängigen Browsern. ■ Laut einer Studie (-> TODO: welche?) leiden alle Browser unter einer Vielzahl von Schwachstellen, von denen viele zuvor nicht bekannt waren. ■ Die Probleme werden hauptsächlich durch eine laxere Kontrolle von Berechtigungen, inkonsistente Implementierungen der zugrunde liegenden SQLite-Datenbank, die Vernachlässigung von Cross-Mode-Interferenzen und eine fehlende Beachtung von Timing-Angriffen verursacht. ■ Alle Angriffe wurden experimentell verifiziert und Gegenmaßnahmen vorgeschlagen.

Private Browsing Motivation und Ausnutzen von Kriminellen [21] ■ Fast alle Aspekte des Lebens nutzen bereits das Internet, um auf das Internet zugreifen zu können, wird ein Webbrowser verwendet. ■ Die Einführung des Internets hat das Leben der Menschen in vielen Bereichen verändert, darunter auch im Bereich der Kriminalität, insbesondere in der Verwendung von Webbrowser-Software für Transaktionen und Prozesse im Internet. ■ Webbrowser speichern normalerweise Informationen wie URL-Verlauf, Suchbegriffe, Passwörter und andere Nutzeraktivitäten. ■ Aus Sicherheitsgründen wurden einige Funktionen von Webbrowsern entwickelt, um den privaten Modus zu ermöglichen. ■ Leider wird diese Funktion von einigen skrupellosen Menschen für kriminelle Aktivitäten durch die Anti-Forensik genutzt, um digitale Beweise in kriminellen Fällen zu minimieren oder zu verhindern.

Auswirkung von Darknet und Tor auf Forensiker [20] ■ Personen, die Inhalte aus dem Darknet abrufen möchten, müssen nicht nur in einem regulären Browser Schlüsselwörter eingeben, sondern müssen es anonym über den TOR-Browser zugreifen, um ihre Identität wie IP-Adresse oder physische Lage zu verbergen. ■ Aufgrund dieser Tatsachen ist es für Strafverfolgungsbehörden oder digitale forensische Experten schwierig, den Ursprung des Datenverkehrs, den Standort oder die Eigentümerschaft eines Computers oder einer Person im Darknet zu lokalisieren. ■ Die Auswirkungen des Darknets traten auf, als das Federal Bureau of Investigation (FBI) im Oktober 2013 die Website Silk Road abschaltete, die ein Online-Schwarzmarkt und der erste moderne Darknet-Markt für den Verkauf illegaler Drogen war. ■ Silk Road war nur über das TOR-Netzwerk zugänglich und vom Mainstream-Web verborgen. ■ Da die meisten Darknet-Sites Transaktionen über anonyme digitale Währungen wie Bitcoin durchführen, die auf kryptografischen Prinzipien basieren, ist es für digitale forensische Experten sehr schwierig, solche Transaktionen zu verfolgen, da Benutzer und Dienste anonym sind. ■ Das Ziel dieser Arbeit besteht darin, digitale forensische Techniken zu diskutieren, um solche Darknet-Verbrechen zu behandeln.

2 Theoretischer Hintergrund

2.1 Private Browsing

2.2 Angreifermodell

2.3 Private Browsing Artefakte

3 Ziel der Arbeit

4 Methodik

In der Browserforensik ist eine definierte Methodik notwendig, um die Komplexität moderner Browser zu bewältigen. Sie bildet eine wissenschaftliche Basis für den durchgeführten Versuch sowie einen Leitfaden für Ermittler bei zukünftigen Untersuchungen. [1, 10, 11] Ein oft verwendetes Vorgehensmodell in der Computer Forensik ist das "Generic Model Computer Forensics Investigations", kurz GCFIM. [25]

Izzati et al. haben diese Phasen auf Browserforensik abgebildet: [11]

- 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 "Vergleich der Browser"(TODO!). Die Methodik der anderen Phasen wird nachfolgend beschrieben.

4.1 Vorbereitung

In der Vorbereitungsphase wird der durchgeführte Versuch geplant sowie die Versuchsumgebung konfiguriert. [11] 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

Für diese Arbeit dazu entschieden: zwei weit verbreitete Brower verwenden + zwei Browser mit verstärktem Schutz der Privatsphäre.

Dazu: Google Chrome und Mozilla Firefox ausgewählt.

Weiterhin werden zwei Browser mit verstärktem Schutz der Privatsphäre ausgewählt. Basierend auf Chromium wird der Browser "Brave"gewählt. Für Firefox wird der Tor-Browser gewählt, eine modifizierte Version von Firefox.

Firefox

Der Browser Mozilla Firefox, kurz Firefox, ist ein open-source Webbrowser der gemeinnützigen Organisation Mozilla. *** TODO: Hier Verweis auf Literatur ** Firefox hat die Funktion des "privaten Modus". Diese ermöglicht es, ohne Speicherung von Verlaufsdaten und Cookies zu 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 Ihr Internetanbieter (ISP) weiterhin anhand Ihrer IP-Adresse Informationen über die von Ihnen besuchten Seiten sammeln.

Tor

Der Tor Browser, ist ein auf Firefox basierender Webbrowser, der das Tor-Netzwerk nutzt. Im Gegensatz zu Firefox wird hier mit Schutzmaßnahmen gegen den Webangreifer geworben. Internetaktivitäten aufgrund des Tor-Netzwerks nicht verfolgt werden. Der Tor Browser wirbt mit folgendem Schutzmaßnahmen gegen lokalen Angreifer: - Tor Browser does not keep any browsing history. Cookies are only valid for a single Browsing session. Zusätzliche Funktion: "Neue Identität" Die Funktion "Neue Identität" im Tor Browser 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.

Chrome

Brave

4.1.2 Browsing Szenario

Im Falle der Browser-Forensik werden eine Reihe von Aktivitäten definiert, die für jeden zu untersuchenden Browser durchgeführt werden, das sog. "Browsing Szenario". Es wird somit definiert, mit welchen Daten der Rechner kontaminiert werden soll.

- Ziel: PB Artefakte, die ausschließlich in Browsing-Szenario vorkommen, bspw. "twitteröder "facebook" bereits in vielen Windows-Standardanwendungen enthalten.

Folgende Schritte werden in jedem Browser durchgeführt:

1. www.google.com aufrufen
 - 1.1. Alle Cookies akzeptieren
 - 1.2. Google-Suche nach "pfaffenhofen"
2. www.google.com aufrufen
 - 2.1. Cookies alle akzeptieren
 - 2.2. Google-Suche nach "nanoradar"

3. 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 aufrufen
 - 4.1. Cookies alle akzeptieren
 - 4.2. Google-Suche nach "mooserliesl"
 - 4.3. Auf Suchergebnis "mooserliesl.de" klicken
5. "www.unitree.com" über URL-Leiste öffnen
6. "www.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 X dargestellten "private Browsing Artefakte", kurz PB Artefakte ableiten. Dabei handelt es sich dabei um Strings oder reguläre Ausdrücke, die eindeutig einem Schritt im Browsing-Szenario zugeordnet werden können. Diese sind von zentraler Bedeutung in der Analysephase: Nur nach diesen Strings wird gesucht.

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

VM Konfiguration

Best Practice in Browser Forensik: Versuche sowie Analysen in virtualisierter Umgebung durchführen. (TODO: Quellen) - Dadurch Reproduzierbarkeit der Ergebnisse sichergestellt - Keine Vermischung der PB Artefakte, wenn gleicher Rechner zur Analyse verwendet - Trennung der Versuchsumgebung von der Analyseumgebung - Ergebnisse sind transportabel -> Zustände von Virtuellen Maschinen exportierbar

Als Virtualisierungssoftware für Versuch verwendet: Kostenlose Oracle VirtualBox VM.

Pro Browser wird eine VM erstellt, auf der Browsing Szenario durchgeführt wird. Alle VMs basieren auf gleicher OVA, deren Konfiguration in Tabelle X dargestellt ist.

Um Programme auf VM zu installieren: Gemeinsamer Ordner zwischen VM und Rechner eingerichtet, auf dem VM läuft. Ordner wird in VM als Netzwerklaufwerk angezeigt. Programme auf Analyse-Rechner heruntergeladen, dann in VM offline installiert, um VM nicht vor Browsing-Szenario bereits mit Browsing Artefakten zu kontaminieren.

Auf VM zwei Werkzeuge der Sysinternal-Abteilung von Microsoft installiert, um in Analysephase Browserverhalten vollständig zu untersuchen: "Process Monitor", zur Aufzeichnung aller Prozessaktivitäten sowie "Process Explorer", um die Funktionen des Windows Task Managers zu erweitern.

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

Browserinstallation

Nachdem OVA mit Standardkonfiguration erstellt: Für jeden Browser dupliziert und Browser über gemeinsamen Ordner installiert. Folgende Installationsverzeichnisse verwendet:

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

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

Chrome ***TODO!***

Brave ***TODO!***

Verwendete Software

Neben Konfiguration der VM muss Analyseumgebung vorbereitet werden. Als Analyseumgebung dient der Rechner, auf dem die VM läuft. Spezifikationen: *** TODO *** Dazu: Diverse Tools zur Analyse installieren

Autopsy

Um erstellte Festplattenabbilder zu untersuchen: Tool "Autopsy" verwendet. = Source-Digital-Forensik-Tool, das auf der Sleuthkit-Bibliothek basiert, diese mit zusätzlichen Funktionen erweitert und eine grafische Benutzeroberfläche für die forensische Analyse bietet. = Sammlung Befehlszeilen-Tools für die forensische Analyse von Dateisystemen.

Volatility

Um Abbilder des Arbeitsspeichers zu untersuchen: "VolatilityFramework verwendet. = Open-Source-Tool, das speziell darauf ausgerichtet ist, Informationen und Artefakte aus dem physischen oder virtuellen Arbeitsspeicher eines Computers zu extrahieren. Geschrieben in Python, frei auf GitHub verfügbar: Für diesen Versuch verwendet: "Volatility3- vollständige Neuschreibung des Volatility Memory Forensics Frameworks, wird seit 2020 entwickelt. Volatility basiert auf Plugins, welche spezifische Funktionen und Analysen für verschiedene Aspekte des Systems bereitstellen. Für diesen Versuch werden folgende Plugins verwendet.

- pslist
- yarascan
- memmap
- filesca
- svcscan

Genaue Beschreibung der Plugins und deren Zusammenhang: Siehe Analysephase in Kapitel X (TODO!).

Software	Zweck	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 indexCachedatei zu analysieren	Commit b50ab4f
dejsonlz4	Dekomprimierung von .jsonlz4-Dateien	Commit c4305b8

Sonstige Tools

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

4.2 Datensammlung

In der Phase der Datensammlung werden alle potenziellen Beweismittel identifiziert und in einem analysierbaren Format gesichert.[11]

Für diesen Versuch umfasst dies die Durchführung des Browsing Szenarios sowie die Sammlung von Ressourcen, die potentielle private Browsing Artefakte enthalten.

Process Monitor Logfiles

Um das Verhalten von privaten Browsingmodi möglichst vollständig zu untersuchen, schlagen Fayyad-Kazan et al. [4] vor, alle Aktivitäten des Browsers während Browsing-Szenarios aufzeichnen. Dazu werden mit dem Tool "Process Monitor" alle Prozess-Aktivitäten zwischen zwei Zeitpunkten als "Process Monitor Logfile" (PML) oder CSV-Datei gespeichert. [4, 21] Die PML-Dateien werden mithilfe des gemeinsamen Ordners auf den Analyse-Rechner transportiert.

Erstellen der Speicherabbilder

Eine der Hauptaufgaben eines Computer-Forensischen-Ermittlers ist die Erstellung und Analyse von direkten Kopien der Speichermedien des untersuchten Rechners. [8] Im Falle der Browser Forensik werden Abbilder der Festplatten und des Arbeitsspeicher erstellt und analysiert.

Festplatten-Image Da in diesem Versuch die Festplatten virtualisiert wurden, wird ein Abbild aus einem sogenannten "VM-Snapshot" gewonnen, eine Momentaufnahme der virtuellen Maschine. VM-Snapshots können "aufgetaut" werden, wodurch der Zeitpunkt der Momentaufnahme des Betriebssystems 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 ein 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.

*** TODO: Hier Einlesen der Festplatten-Images => Zeitaufwändig! ***

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

Zeitpunkte zur Datensammlung Wichtig für die Qualität der Versuchsergebnisse sind die Zeitpunkte während des Browsing Szenarios zum Sammeln der Daten. Dieses Thema wird in der Literatur kaum thematisiert. Die Autoren wählen die Zeitpunkte meist ohne Begründung [1, 14, 16, 22–24].

Dieses Problem haben Muir, Leimich und Buchanan erkannt und Zeitpunkte zur Datensammlung vorgeschlagen, um das Browserverhalten während des Browsing-Szenarios vollständig analysieren zu können [15]. Wie in Abbildung X (TODO!) dargestellt, wurde sich an diesen Zeitpunkten für diesen Versuch orientiert. *** TODO: Kürzen *** Nach der Browser-Installation, vor Beginn des Browsing-Szenarios wird der erste RAM-Dump sowie der erste VM-Snapshot erstellt. Diese Speicherabbilder dienen als Benchmark für die Analyse, da in diesen Speicherabbildern kein PB Artefakt gefunden werden darf.

Nachdem der Private Modus im Browser geöffnet wird, bevor das Browsing Szenario beginnt wird die Aufnahme des ersten Process Monitor Logfiles gestartet. Die Aufzeichnung beginnt erst zu diesem Zeitpunkt, da beim erstmaligem Öffnen der Browser einige Dateien initial angelegt werden. 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 beendet. Weiterhin wird ein zweiter RAM-Dump sowie VM-Snapshot erstellt. Anschließend wird das eine zweite Process Monitor Aufzeichnung gestartet. Somit enthält das erste Logfile zu diesem Zeitpunkt die Prozessaktivitäten während des Browsing Szenarios.

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

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 X (TODO!) dargestellt, umfasst dies einen zusätzlichen RAM-Dump sowie VM-Snapshot und ein weiteres Process Monitor Logfile. 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 geöffnet.

4.3 Datenanalyse

Nachdem Daten in Form von Process Monitor Logfiles und Festplatten- sowie RAM-Speicherabbildern gesammelt: in gesammelten Daten nach Strings in Tabelle X aus Kapitel X (TODO!) suchen.

Um die Analyse zu vereinfachen: die gesammelten Daten des Versuchs lassen sich in drei Kategorien aufteilen: > Common Locations > Uncommon Locations > Registry

4.3.1 Common Locations

*** TODO: Wichtig: NUR FESTPLATTE, NICHT RAM ***

Die sogenannten "Common Locations", (dt. "gängige Speicherorte") beziehen im Zusammenhang der Browserforensik auf die standardmäßigen Verzeichnisse eines Browsers, beispielsweise Ordner von Browsern zur Verwaltung von Nutzerdaten. Untersucht werden Common Locations mittels "Whitebox-Analyse"[2] Der Fokus liegt darauf, das System vollständig zu verstehen und alle relevanten Beweise zu sammeln.

Bei diesem Versuch werden die Browser 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. Dateixtraktion: 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.

Process Monitor Logfiles

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

Dazu wird jede Process Logfile mit dem Process Explorer eingelesen. Anschließend werden die Aktivitäten gefiltert. Wie in Abbildung X dargestellt werden dazu ausschließlich die Option "File System Activity" ausgewählt. Anschließend wird als Prozessname der Browserprozess gesetzt:

Firefox firefox.exe

Tor-Browser firefox.exe und tor.exe

Chrome chrome.exe

Brave brave.exe

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 kategorisiert wurden, wird für jede Datei geprüft, ob PB Artefakte enthalten sind. Folgende, in Abbildung X (TODO!) dargestellte Schritte sind zur Dateixtraktion und Dateianalyse notwendig: Die Datei befindet sich entweder im entsprechenden Festplatten-Image oder ist im RAM-Dump gespeichert. Wenn die Datei nicht mit Autopsy aus dem Festplatten-Image extrahiert werden kann und sich der Dateiname in der Ausgabe des Volatility Plugins "filescaan"(vol.py -f ram_dump.img windows.filescaan > filescaan.txt) befindet, wird diese mit dem Volatility Plugin "dumpfiles" 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 "bome-file.json.tmp" wird beispielsweise geprüft, ob die Datei "bome-file.json" existiert. Nachdem die Datei extrahiert wurde und ggf. mit einem Tool zu Analyse vorbearbeitet wurde, wird geprüft, ob die Datei PB Artefakte enthält.

SQLite-Datenbanken

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

Wie in Abbildung X dargestellt erfolgt die Dateixtraktion analog zur Vorgehensweise bei den Schreiboperationen der Process Monitor Logfiles.

Um die SQLite Datenbanken zu analysieren wird jede Datenbank mit der gleichen Datenbank aus dem vorherigem Snapshot mithilfe des Befehlszeilentools "sqldiff.exe" (sqldiff.exe database1.sqlite

database2.sqlite) verglichen. Die Inhaltsunterschiede werden für jede 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.

4.3.2 Uncommon Locations

Ungewöhnliche Speicherorte 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 [2]: Im Kontext der Browser Forensik werden dazu Stringsuchen nach PB Artefakten über die gesamten Speicherabbilder durchgeführt. Dies ist nur durch Unterstützung mit Forensik-Tools möglich. Somit wird bei der Analyse der Uncommon Locations in 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 X definierten PB Artefakten über das gesamte eingelesene Festplatten-Image durchgeführt.

Autopsy bietet dazu die in Abbildung X dargestellte Funktion an, zur Suche nach Strings, Teilstrings oder regulären Ausdrücken in Dateinamen und Dateiinhalten.

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 kritisch, dass ein gefundener String eindeutig einem Browserprozess zugeordnet werden kann.

In der Literatur wird der Arbeitsspeicher oft unzureichend analysiert, indem eine Stringsuche im RAM-Dump durchgeführt ist, der als Binärdatei in einem Hexadezimaeditor geöffnet ist. [13, 14, 21] Wie beispielhaft in Abbildung X und Y gezeigt, wird ein String, der in einer Textdatei auf dem Desktop gespeichert ist ebenfalls 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" wird bestimmten Mustern im Arbeitsspeicher gesucht. Die für diesen Versuch verwendeten Yara-Regeln entsprechen den Strings der PB Artefakte in Tabelle X in Kapitel Y. Zusätzlich sucht eine Regel nach HTML-Fragmenten, die eindeutig einer besuchten Seite des Browsing-Szenarios zuzuordnen sind. [22] Alle verwendeten Yara-Regeln sind im Anhang X (TODO!) aufgelistet.

Um den RAM-Dump nach den Yara-Regeln zu durchsuchen wird folgender Befehl ausgeführt: `vol.py -f ram_dump.img windows.vadyarascan -yara-file yara_rules.yara > yarascan.txt` Nachdem der RAM-Dump nach den Regeln durchsucht wurde, gibt die Yarascan-Ausgabe für jeden gefundenen String die PID des Prozesses, in dem der String gefunden wurde sowie die virtuelle Speicheradresse des gefundenen Strings an.

Wie in Abbildung X 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 für 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-Offset der extrahierten Speicherseite des Prozesses ermittelt. Diese Seite kann mithilfe des `-dump` Flags extrahiert werden: `vol.py -f ram_dump.img -o \dump_dir\ windows.memmap -pid <PID> -dump`. In einem Hexadezimaeditor, wie HxD, kann der String-Treffer 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 eigene Kategorie aufgeführt.

Common Locations *** TODO: Common location: Shellactivities Key *** existiert nicht mehr -> Nicht mehr vorhanden in aktueller Version (Verweis auf E-Mail)

Als Teil der Common Locations werden die Registry-Aktivitäten in den Process Monitor Logfiles analysiert. Wie in Abbildung X gezeigt, wird zunächst die Logfile ausschließlich nach "Registry Activity" sowie Einträgen mit der Operation `SetValue` sowie dem Browser-Prozessnamen gefiltert. Als CSV Datei wird das Logfile in Excel weiter verarbeitet, indem Duplikate gelöscht werden und die geschriebenen Registry Keys browserspezifisch gruppiert werden.

Uncommon Locations Als Uncommon Location werden alle 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 X dargestellten Hives werden von Windows beim Start geladen und dienen Systemkomponenten und Anwendungen als Quelle für Einstellungen und Informationen. Zur Analyse wird

jeder Hive aus den Festplatten-Images extrahiert und in eine Registry Explorer Sitzung geladen und eine Stringsuche nach PB Artefakten durchgeführt.

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

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

5 Ergebnisse

*** TODO ***

5.1 Firefox

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

Common Locations

Zunächst werden die standardmäßigen Speicherorte für Browserartefakte nach potentiellen privaten Browsing Artefakten untersucht. Diese Common Locations beziehen sich ausschließlich auf Dateien, die auf die Festplatte geschrieben werden. In diesem Versuch wird gemäß Methodik in Kapitel X (TODO!) zwischen 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 X beschrieben.

Uncommon Locations

Nachfolgend werden die Analyseergebnisse der Firefox Uncommon Locations beschrieben. Wie in Kapitel X erläutert, wird im Gegensatz zu Common Locations die Suchrichtung umgekehrt und es werden alle gesammelten Daten nach einem spezifischen PB Artefakt durchsucht. Somit benötigt ein Forensiker 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

Bei den Common Locations in Kapitel X wird Autopsy nur zur Dateiextraktion genutzt. Im Falle der Uncommon Locations dient Autopsy als forensisches Werkzeug zur Datenanalyse.

Eine Autopsy Stichwortsuche gemäß Methodik in Kapitel X (TODO!) lieferte in allen Snapshots keine Treffer. Es 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 X beschrieben.

Analyse mit Volatility

Nachdem die Firefox Festplattenabbilder als Uncommon Location mit Autopsy untersucht wurden, werden nachfolgend die Analyseergebnisse des RAMs als Uncommon Location beschrieben. Dazu wurde eine Stringsuche im gesamten RAM nach PB Artefakten durchgeführt. Wie in Kapitel X ausführlich beschrieben muss ein gefundener String eindeutig einem Browser zugeordnet werden können.

Deshalb wurde dazu das Volatility Plugin "Yarascan" verwendet, ein Werkzeug um nach bestimmten Mustern im RAM zu suchen. Dazu wurden die in Tabelle X aufgeführten Yara-Regeln verwendet. Wie in Kapitel Methodik (TODO!) beschrieben, wird davon ausgehend das Plugin "pslist" verwendet, um den Prozessnamen anhand PID zu identifizieren. Die Ergebnisse dieser Stringsuche sind nachfolgend nach Kategorie geordnet.

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

Yararule Keyword Wie in Abbildung X (TODO!) gezeigt, wurden alle Suchbegriffe "pfaffenhofen", "nanoradar", "mooserliesl" sowie "mallofamerica" identifiziert im zweiten RAM Dump, nach dem Browsing Szenario mit geöffnetem Browser, identifiziert. Die Artefakte befinden sich ausschließlich im zweiten RAM Dump. Die Suchbegriffe wurden größtenteils in den Speicherbereichen von Firefox-Prozessen gefunden. Nur in zwölf Fällen wurden Suchbegriffe in anderen Prozessen identifiziert. Am häufigsten wurde der Suchbegriff "pfaffenhofen" mit 1301 gefundenen Artefakten im zweiten Firefox RAM Dump gefunden. Dies ist vermutlich auf den Google Maps Kartenbereich zurückzuführen, einen visuellen Ausschnitt der Karte, welcher bei der Google-Suche erscheint und Informationen über die geografische Lage, Straßen, Sehenswürdigkeiten und andere relevante Orte in der gesuchten Stadt zeigt. In den RAM Dumps 1 und 3 konnten Artefakte zu den Suchbegriffen identifiziert werden.

Yararule URL Es konnten in den Arbeitsspeicherabbildern alle besuchten URLs unitree.com, mooserliesl.de, mallofamerica.com sowie donaukurier.de identifiziert werden. Dabei wurden die meisten Artefakte nach dem Browsing Szenario mit geöffnetem Browser (RAM Dump 2) gefunden. Alle besuchten URLs wurden in diesem Dump sowohl in Firefox als auch anderen Prozessen gefunden, wobei die meisten Artefakte in Firefox Prozessen zu finden sind. Dabei wurde "mooserliesl.de" mit insgesamt 390 Artefakten am wenigsten gefunden, "donaukurier.de" mit über 3600 Artefakten am häufigsten.

Bemerkenswert ist, dass URL Artefakte gefunden wurden, nachdem der Browser geschlossen wurde (RAM Dump 3). Dabei wurde kein URL Artefakte in einem Firefox Prozess gefunden. Anhand

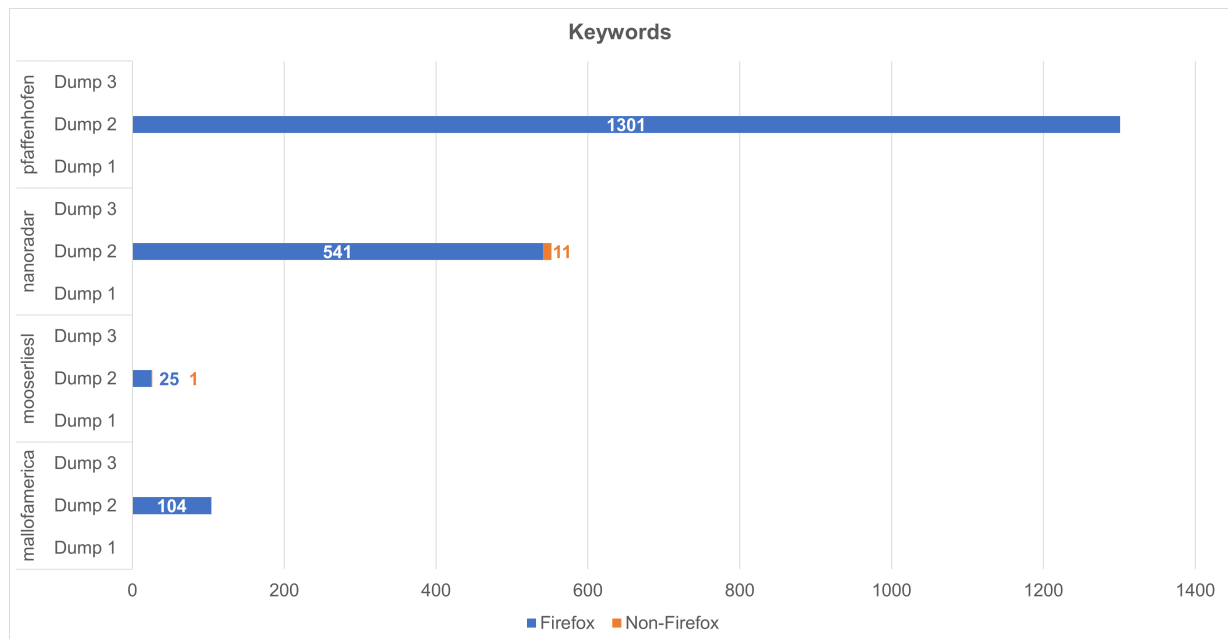


Abbildung 5.1: Keywords

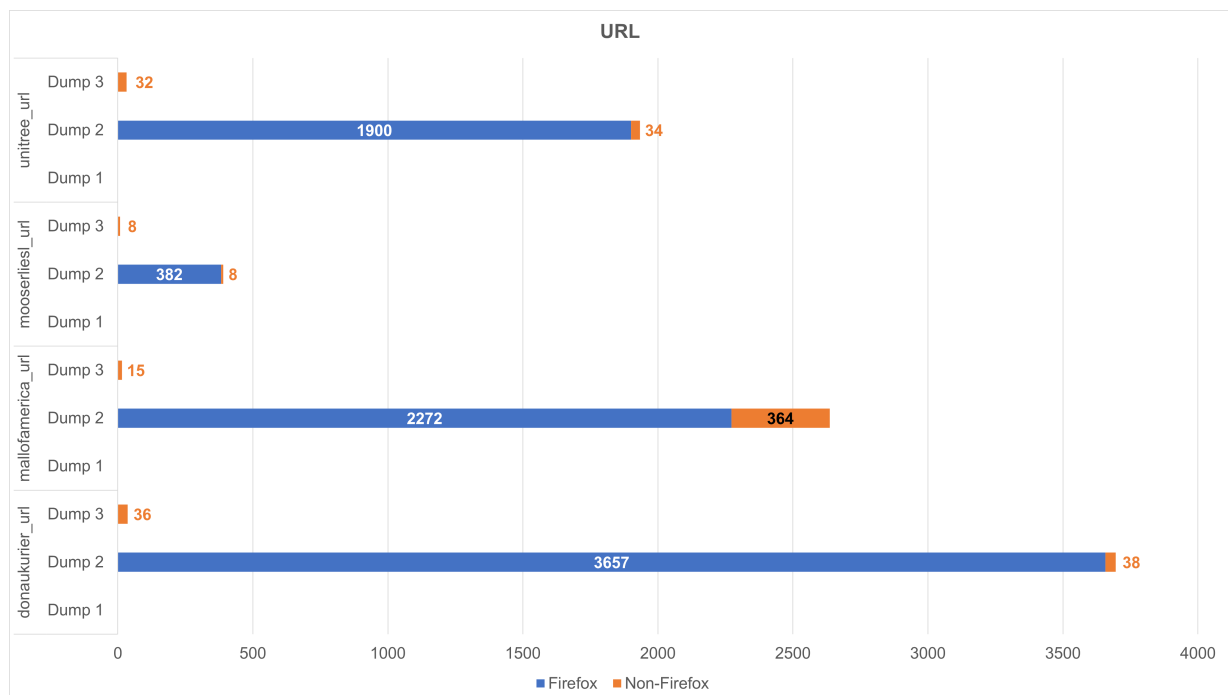


Abbildung 5.2: URL

der PID 2252 wurde festgestellt, dass sich alle URL Artefakte des dritten RAM Dumps in einem `lvchost.exe` Prozess mit der gleichen PID befinden. Unter dem `SService Host` Prozess laufen gruppierte Windows-Dienste, um Ressourcen zu sparen und die Systemleistung zu verbessern. Volatility bietet das

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

Plugin Bvscan, 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. Stattdessen wurde der dritte Snapshot aufgetaut, um im laufenden Windowsbetrieb den Dienst mithilfe des Process Explorers zu identifizieren. Wie in Abbildung X (TODO!) gezeigt, wurde anhand der PID 2252 der Dienst "DNSCache" ermittelt. Der DNSCache-Dienst unter Windows ist ein Teil des Betriebssystems, der für

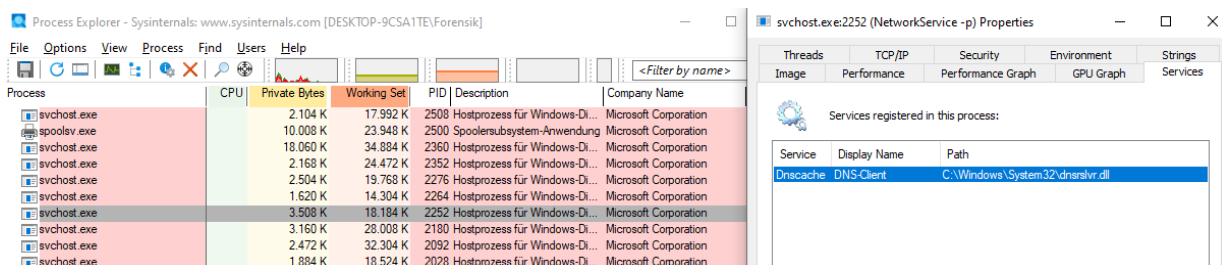


Abbildung 5.3: URL

die Übersetzung von Domainnamen in IP-Adressen verantwortlich ist. Der DNSCache-Dienst speichert DNS-Anfragen und Antworten temporär, um wiederholte DNS-Anfragen zu beschleunigen. Nach Löschen des DNSCaches mit dem Kommandozeilenbefehl `ipconfig /flushdns`, dem Schließen aller Process Monitor Instanzen sowie Beenden des DNSCaches Services wurde erneut ein RAM-Dump durchgeführt. Dort wurden keine URL Artefakte mehr gefunden.

Yararule Mail Es konnten alle E-Mail Artefakte des Browsing Szenarios gefunden werden. Die Artefakte befinden sich ausschließlich im zweiten Firefox RAM Dump, nach dem Browsing Szenario mit geöffnetem Browser. Unter den gefundenen Artefakten befindet sich mit zwölf Vorkommen am häufigsten die Absenderadresse "computerforensikvl@gmail.com". Dieses Artefakt wurde als einziges Mail-Artefakt in anderen Prozessen außer Firefox gefunden.

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

Zu diesen Artefakten wurde gemäß Methodik in Kapitel X der String Kontext – die Zeichen vor und nach dem gefundenen Artefakt im Speicherbereich – ermittelt. Dazu wurde gemäß Methodik in Kapitel X mithilfe des Volatility memmap-Plugins die Abbildung der virtuellen Speicheradressen auf den Byte-Offset in der extrahierten Speicherseite des Prozesses ermittelt. Wie in Abbildung X gezeigt, sind in der Speicherseite des Prozesses mit PID 7420 konnte vor und nach dem gefundenen Passwort

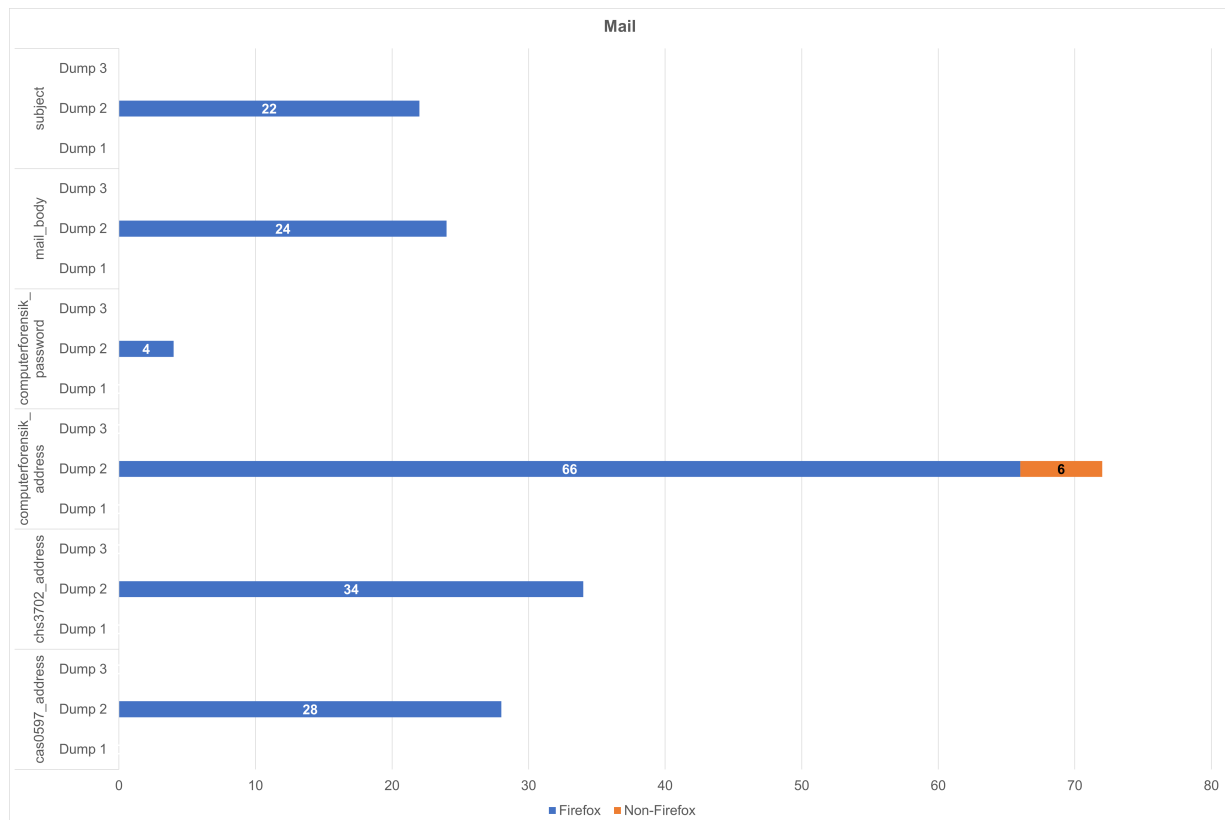


Abbildung 5.4: Mail

```

11DD4040 58 02 00 00 08 00 00 00 67 6D 70 41 64 64 6F 6E X.....gmpAddon
11DD4050 4B 4B 4B 4B 4B 4B 4B 4B DC F9 0E 50 4B 4B 4B 4B KKKKKKKKÜ.PKKKK
11DD4060 58 02 00 00 0D 00 00 00 67 6D 70 44 6F 77 6E 6C X.....gmpDownl
11DD4070 6F 61 64 65 72 4B 4B 4B 50 C3 FB EA 4B 4B 4B 4B oaderKKKPÄüKKKK
11DD4080 58 02 00 00 0D 00 00 00 47 4D 50 44 6F 77 6E 6C X.....GMPDownl
11DD4090 6F 61 64 65 72 4B 4B 4B D0 6F AE 8A 4B 4B 4B 4B oaderKKKDo@SKKKK
11DD40A0 58 02 00 00 0D 00 00 00 5F 69 73 45 4D 45 45 6E X....._isEMEEEn
11DD40B0 61 62 6C 65 64 4B 4B 4B 8F 4F 0E 73 4B 4B 4B 4B abledKKK.O.sKKKK
11DD40C0 58 02 00 00 0C 00 00 00 56 6F 72 6C 65 73 75 6E X.....Vorlesun
11DD40D0 67 32 33 21 4B 4B 4B 4B F8 35 7D 48 4B 4B 4B 4B g23!KKKK5}HKKKK
11DD40E0 58 02 00 00 0F 00 00 00 5F 69 73 41 64 64 6F 6E X....._isAddon
11DD40F0 45 6E 61 62 6C 65 64 4B 42 1D 99 C2 4B 4B 4B 4B EnabledKB.ÄÄKKKK
11DD4100 58 02 00 00 0F 00 00 00 6D 61 69 6C 2E 67 6F 6F X.....mail.goo
11DD4110 67 6C 65 2E 63 6F 6D 4B 44 47 D9 2D 4B 4B 4B 4B gle.comKDGÜ-KKKK
11DD4120 58 02 00 00 10 00 00 00 5F 75 70 64 61 74 65 4C X....._updateL
11DD4130 61 73 74 43 68 65 63 6B 43 1F 7D 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

```

Abbildung 5.5: Password in memory page of PID 7420 at Byte-Offset 0x11dd40c8

am Byte-Offset 0xb9ce29180c8 neben der Gmail-Url "mail.google.com" Code-Fragmente der "Gecko-Engine" 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. In der gleichen Datei konnte nach dem gefundenen Passwort am Byte-Offset 0x24083b41858 die Strings "Passwd" sowie "Sessionrestore" (siehe Common Location SSessionstore in Kapitel X) identifiziert werden. Wie in den Abbildungen X und Y (TODO!) gezeigt, können in den Byte-Offsets der gefundenen Passwörter in der Speicherseite der PID 8424 konnten kein Kontext ermittelt werden. Im Gegensatz


```

12E23470 00 00 00 00 00 00 00 00 10 02 00 00 34 00 00 00 .....4...
12E23480 00 C8 CC E5 29 02 00 00 00 00 00 00 00 00 00 00 .Eiä) .....
12E23490 10 02 00 00 27 00 00 00 40 D3 CC E5 29 02 00 00 ....'...@Öiä)...
12E234A0 00 00 00 00 00 00 00 00 10 02 00 00 2A 00 00 00 .....*...
12E234B0 70 D3 CC E5 29 02 00 00 00 00 00 00 00 00 00 00 pÖiä) .....
12E234C0 00 02 00 00 45 00 00 00 A0 47 7C 1A 55 23 00 00 ....E... G|.U#..
12E234D0 F8 DE FF F4 59 28 00 00 50 02 00 00 0C 00 00 00 øËÿÖY(.P.....
12E234E0 56 6F 72 6C 65 73 75 6E 67 32 33 21 A2 1D FB FF Vorlesung23!c.üÿ
12E234F0 50 02 00 00 0A 00 00 00 69 64 65 6E 74 69 66 69 P.....identifi
12E23500 65 72 F0 B8 FA 7F 00 00 50 02 00 00 06 00 00 00 erö,ü...P.....
12E23510 50 61 73 73 77 64 F9 FF 18 96 73 E5 29 02 00 00 Passwdüÿ.-sä)...
12E23520 50 02 00 00 0E 00 00 00 73 65 73 73 69 6F 6E 72 P.....sessionr
12E23530 65 73 74 6F 72 65 00 00 10 02 00 00 2E 00 00 00 estore.....
12E23540 80 D2 CC E5 29 02 00 00 00 00 00 00 00 00 00 00 eÖiä) .....
12E23550 10 02 00 00 1F 00 00 00 00 B8 CC E5 29 02 00 00 .....iä)...

```

Abbildung 5.6: Password in memory page of PID 7420 at Byte-Offset 0x12e234e0

zur Speicherseite der PID 7420 wird das Passwort dort mit 2 Bytes pro Zeichen enkodiert. Das eine Unicode-Zeichenenkodierung vermuten.

```

005837F0 02 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 .....
00583800 C0 9E EA FF 40 02 00 00 0E 00 00 00 00 00 00 00 Äzëÿ@.....
00583810 02 00 00 00 00 00 00 00 00 E5 E5 E5 E5 E5 E5 .....ääääääää
00583820 02 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 .....
00583830 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00583840 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
00583850 02 00 00 00 1A 00 00 00 56 00 6F 00 72 00 6C 00 .....V.o.r.l.
00583860 65 00 73 00 75 00 6E 00 67 00 32 00 33 00 21 00 e.s.u.n.g.2.3.!.
00583870 00 00 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ..ääääääääääääää
00583880 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
00583890 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
005838A0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
005838B0 08 00 00 00 D7 16 71 67 01 00 00 00 00 00 00 00 ....*.qg.....
005838C0 6F 00 6E 00 44 00 51 00 30 00 4B 00 55 00 62 00 o.n.D.Q.O.K.U.b.

```

Abbildung 5.7: Password in memory page of PID 8424 at Byte-Offset 0x583858

```

0096BA80 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BA90 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BAA0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BAB0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BAC0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BAD0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BAE0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BAF0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BB00 01 00 00 00 38 00 00 00 56 00 6F 00 72 00 6C 00 ....8...V.o.r.l.
0096BB10 65 00 73 00 75 00 6E 00 67 00 32 00 33 00 21 00 e.s.u.n.g.2.3.!.
0096BB20 00 00 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ..ääääääääääääää
0096BB30 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BB40 00 C8 93 B8 FA 7F 00 00 28 C8 93 B8 FA 7F 00 00 .E",ü... (E",ü...
0096BB50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0096BB60 04 00 00 00 00 00 00 00 80 9B 47 88 40 02 00 00 .....€>G^@...
0096BB70 2C 41 7B B8 FA 7F 00 00 00 00 00 00 00 00 00 00 ,A{,ü.....
0096BB80 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BB90 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää
0096BBA0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääääää

```

Abbildung 5.8: Password in memory page of PID 8424 at Byte-Offset 0x96bb08

Yararule Image Das im Browsing Szenario geöffnete Donaukurier Logo wurde ausschließlich im zweiten RAM Dump in drei mal in Firefox Prozessen gefunden.

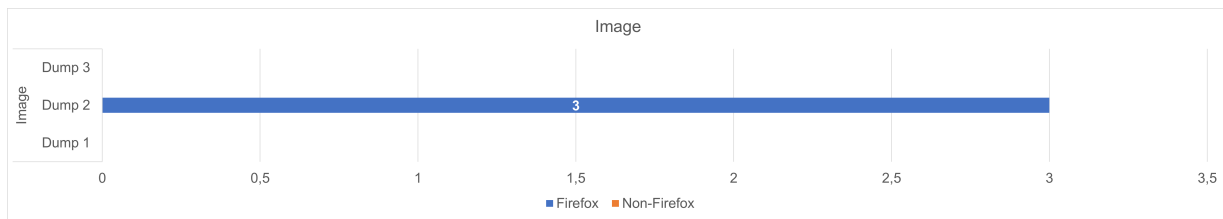


Abbildung 5.9: Image

Wie in Abbildung X zusammenfassend gezeigt wurden vor dem Browsing Szenario, keine private Browsing Artefakte im ersten RAM Dump gefunden. Nach dem Browsing Szenario mit geöffnetem Browser konnten die meisten Artefakte identifiziert werden. Dabei wurden am häufigsten URL Artefakte in Firefox Prozessen gefunden. Zudem konnte hier das E-Mail Passwort im Klartext lokalisiert werden. Nach Schließen des Browsers konnten im dritten Snapshots URLs im DNSCache Windows Service gefunden werden. Nach leeren des Caches und Beenden des DNSCache Services konnten keine Artefakte gefunden werden.

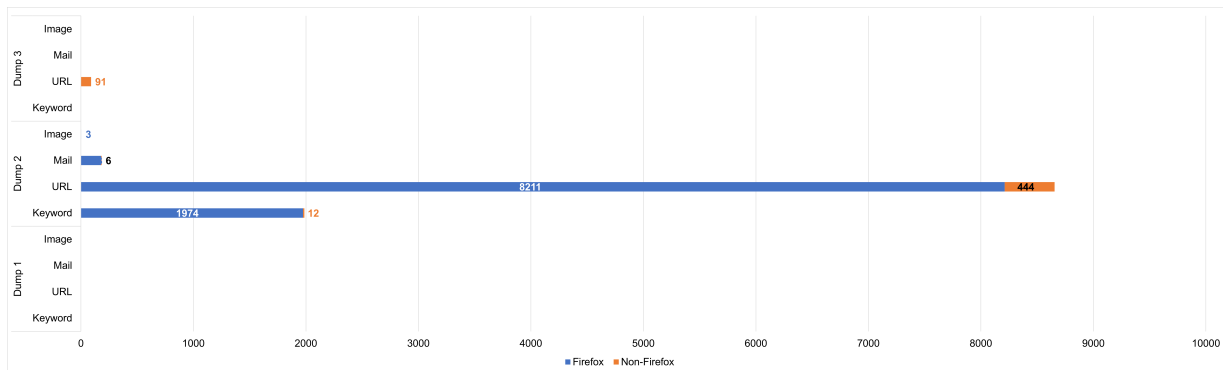


Abbildung 5.10: Summary

Registry

Die Analyse der Registry zählt gemäß Methodik in Kapitel X sowohl zu den Common als auch Uncommon Locations. Weder in den Process Monitor SSetValueOperations 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 X beschrieben.

*** TODO: Zusammenfassung Firefox ***

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

Als Erstes werden die Common Locations analysiert, um potenzielle Hinweise auf Internetaktivitäten des Browsing Szenarios zu finden. Bei der Untersuchung der gängigen Speicherorte wurde gemäß der im Kapitel X (TODO!) beschriebenen Methodik zwischen Schreibvorgängen in den Protokolldateien 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 "WriteFileOperations sowie SQLite-Datenbanken ist im Anhang X 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

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

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

Ebenso wurden in den von automatisch kategorisierten Dateien kein PB Artefakte gefunden. Im Anhang X ist eine detaillierte Analyse der kategorisierten Dateien beschrieben.

Analyse mit Volatility

Nachfolgend werden die Ergebnisse der Analyse des RAMs mithilfe Volatility beschrieben.

Yararule HTML Wie bei Firefox in Kapitel X (TODO!) konnten keine HTML Artefakte im RAM gefunden werden. Deshalb wird diese Kategorie nicht aufgeführt.

Yararule Keyword Wie in Abbildung X gezeigt, wurden ausschließlich während des Browsing-Szenarios (RAM-Dump 2) und nach Erstellen einer "Neuen Identität"(RAM Dump 3-1) Keyword Artefakte gefunden. Nachdem eine "Neue Identität" erstellt wurde, reduzierten sich die gefundenen Artefakte deutlich. Die Keyword-Artefakte wurden hauptsächlich in Firefox Prozessen gefunden. Kein Artefakt war im Tor.exe Prozess zu finden. Mit 4833 Artefakten wurden am häufigsten "pfaffenhofen" nach dem Browsing Szenario im zweiten RAM-Dump gefunden. Nach dem Schließen des Tor-Browsers wurde keine Keyword Artefakte mehr im RAM identifiziert.

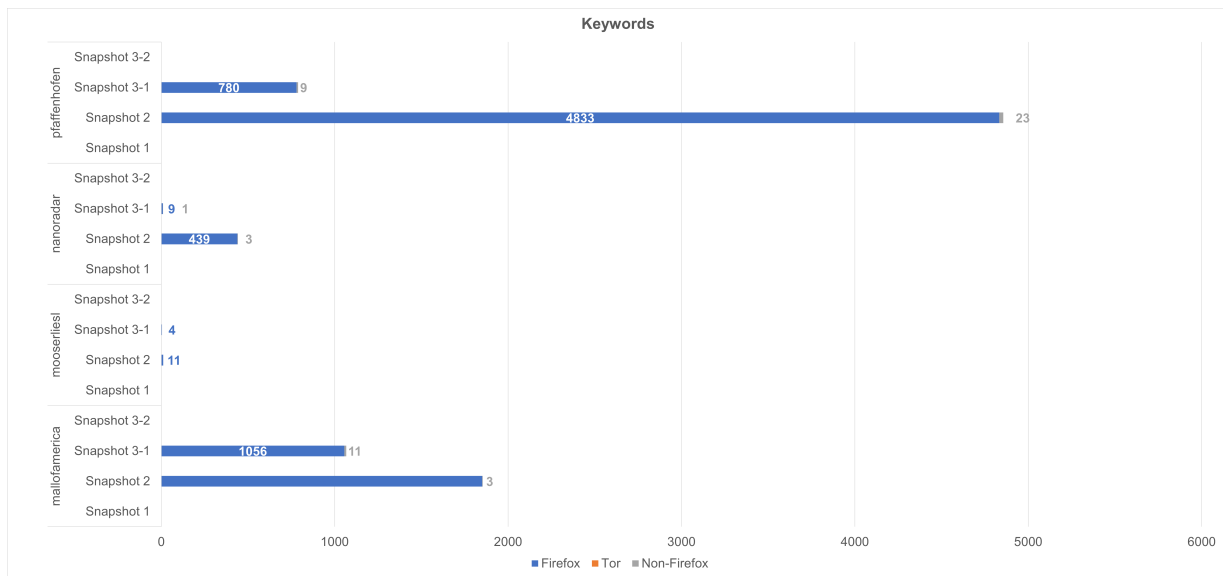


Abbildung 5.11: Keywords

Yararule URL Wie bei der "Keyword" Yararule wurden ausschließlich während des Browsing-Szenarios (RAM-Dump 2) und nach Zuweisung einer neuen Identität (RAM-Dump 3-1) URL Artefakte gefunden. Ebenso wurden im RAM Dump 3-1 bei deutlich weniger Artefakte URLs als in RAM Dump 2 gefunden. Für diese Yara-Regel wurden nach Firefox-Prozessen hauptsächlich Artefakte in Tor-Prozessen gefunden. Am wenigsten Artefakte waren in anderen Prozessen zu finden. Auffällig ist, dass die URL "mallofamerica.com" 26.505 mal in RAM-Dump 2 gefunden wurde. Im Gegensatz dazu wurde "mooserliesl.de" nur 508 mal im zweiten RAM Dump gefunden. Nach Schließen des Tor-Browsers wurden keine URL Artefakte mehr im RAM gefunden.

Yararule Mail Nach dem Browsing-Szenario, vor Zuweisung einer "Neuen Identität" (RAM-Dump 2) konnten alle Mail Artefakte gefunden werden. Wie in Abbildung X zu sehen ist, wurden die Artefakte ausschließlich in Firefox Prozess gefunden. Nur die Absenderadresse "computerforensikvl@gmail.com" wurde nach Erstellen der neuen Identität in RAM-Dump 3-1 gefunden. Die Absenderadresse ist ebenso das am häufigsten gefundene Mail Artefakt. Wie bei Firefox in Kapitel X (TODO!) wurde das Passwort als Klartext nach dem Browsing-Szenario im zweiten RAM-Dump gefunden. Das Passwort wurde in zwei Mal im Firefox Prozess mit der PID 708 gefunden. Tabelle X zeigt die virtuellen Speicheradressen der Artefakte aus der Yarascan Ausgabe sowie deren Abbildung auf die mittels "memmap" identifizierten

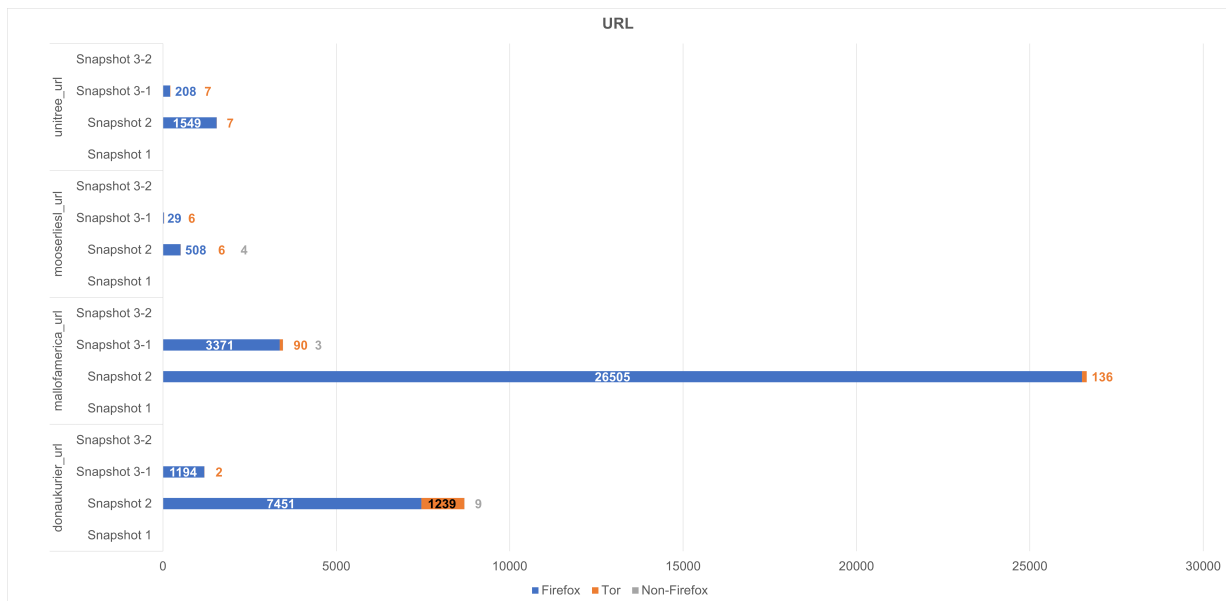


Abbildung 5.12: URL

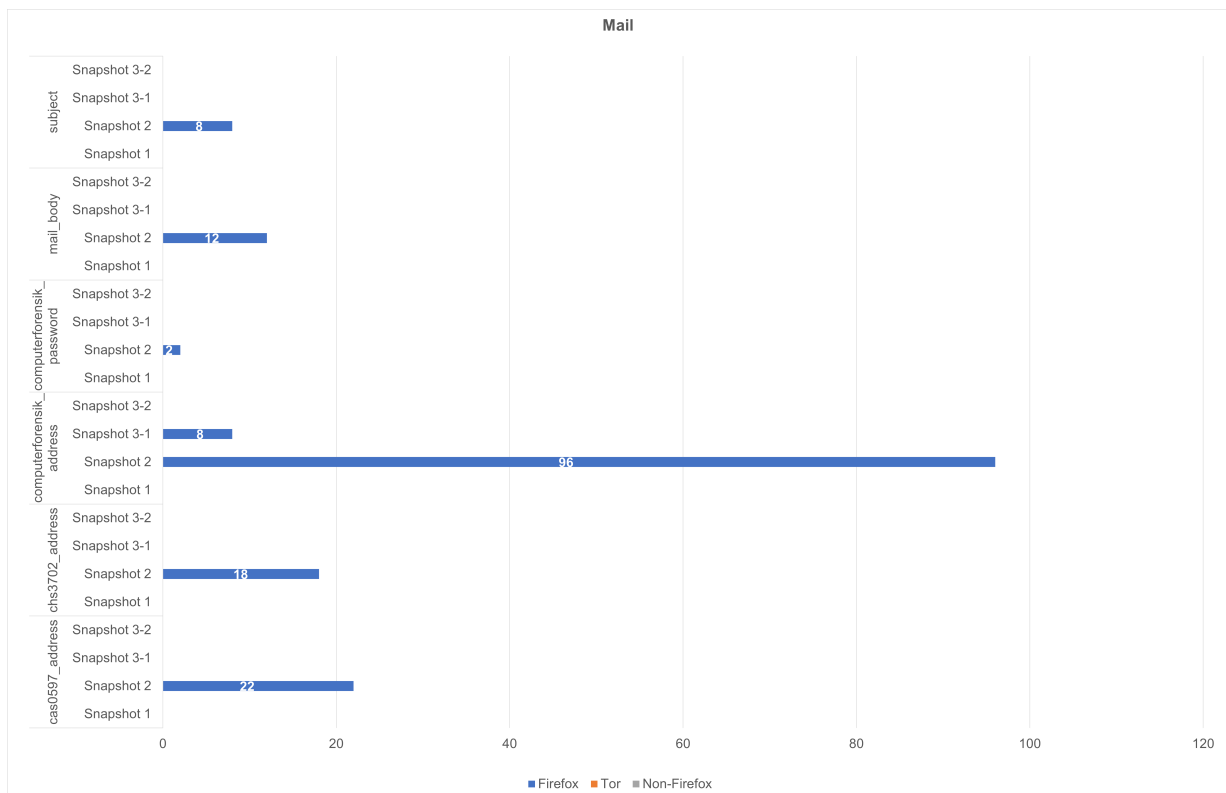


Abbildung 5.13: Mail

Byte-Offsets der extrahierten Speicherseiten. Bei Untersuchung des String-Kontexts wurden für das

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

Passwort am Byte-Offset 0xea0318 wie in Abbildung X gezeigt keine Auffällige Artefakte entdeckt. Wie in Abbildung X gezeigt, wurde im Bereich des gefundenen Passworts am Byte-Offset 0x10f7748

```

00EA02D0 00 00 00 00 E5 E5 E5 E5 01 00 E5 E5 E5 E5 E5 E5 ....ääää..ääää
00EA02E0 C0 94 63 77 FC 7F 00 00 E8 22 C2 E2 B1 02 00 00 Å"cwü...è"Ää±...
00EA02F0 E8 22 C2 E2 B1 02 00 00 01 E5 E5 E5 E5 E5 E5 E5 è"Ää±...ääää
00EA0300 00 69 49 EC B1 02 00 00 E5 E5 E5 E5 E5 E5 E5 E5 .iIi±...ääää
00EA0310 02 00 00 00 1A 00 00 00 56 00 6F 00 72 00 6C 00 .....V.o.r.l.
00EA0320 65 00 73 00 75 00 6E 00 67 00 32 00 33 00 21 00 e.s.u.n.g.2.3.!
00EA0330 00 00 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ..ääääääääää
00EA0340 00 00 00 00 40 D8 F6 FF 60 EA 00 00 F8 2B 09 00 ....@0öÿ'è...ø+..
00EA0350 00 00 00 00 40 D8 F6 FF 60 EA 00 00 F8 2B 09 00 ....@0öÿ'è...ø+..
00EA0360 01 00 00 C0 01 00 00 C0 50 3C 4D EC B1 02 00 00 ...Ä...ÄP<Mi±...

```

Abbildung 5.14: Password in memory page of PID 708 at Byte-Offset 0xea0318

der String "CSP_ignoringSrcForStrictDynamic", dessen Bedeutung nicht bestimmt werden konnte. Weiterhin wurde die Zeichenkette 'invalidation/lcs/client' in der Nähe des Passworts gefunden. Dieser String wird in einem Firefox Bug-Ticket verwendet, welches vor 2017 geschlossen wurde. Der Bug betraf ein Speicher-Leck.

```

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 E5 Dynamic.ääää
010F76F0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääää
010F7700 A0 07 18 77 FC 7F 00 00 E8 07 18 77 FC 7F 00 00 ..wü...è...wü...
010F7710 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
010F7720 04 00 00 00 00 00 00 00 00 4D CF E4 B1 02 00 00 .....Mĩä±...
010F7730 98 2C E9 76 FC 7F 00 00 00 00 00 00 00 00 00 00 ",évü.....
010F7740 01 00 00 00 38 00 00 00 56 00 6F 00 72 00 6C 00 ....8...V.o.r.l.
010F7750 65 00 73 00 75 00 6E 00 67 00 32 00 33 00 21 00 e.s.u.n.g.2.3.!
010F7760 00 00 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ..ääääääääää
010F7770 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 äääääääääääää
010F7780 02 00 00 00 03 00 00 00 C8 06 0F 77 FC 7F 00 00 .....È...wü...
010F7790 60 A3 07 EB B1 02 00 00 50 7B 64 77 FC 7F 00 00 `£.è±...P(dwü...
010F77A0 10 2C D8 E2 B1 02 00 00 10 E7 67 77 FC 7F 00 00 ..øä±...çgwü...
010F77B0 A0 9F 0B E4 B1 02 00 00 E5 E5 E5 E5 E5 E5 E5 E5 Ÿ.ä±...ääää
010F77C0 01 00 00 00 38 00 00 00 2F 00 69 00 6E 00 76 00 ....8.../i.n.v.
010F77D0 61 00 6C 00 69 00 64 00 61 00 74 00 69 00 6F 00 a.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./l.c.s./c.l.
010F77F0 69 00 65 00 6E 00 74 00 00 00 E5 E5 E5 E5 E5 E5 i.e.n.t...ääää

```

Abbildung 5.15: Password in memory page of PID 708 at Byte-Offset 0x10f7748

Yararule Image Wie in Abbildung X dargestellt, wurde der Hexadezimal-Wert des Donaukurier-Logos ein einziges Mal nach dem Browsing Szenario, vor Erstellen der "Neuen Identität" im zweiten RAM-Dump in einem Firefox Prozess gefunden.

Wie in Abbildung X zusammenfassend gezeigt ausschließlich nach dem Browsing-Szenario vor (RAM-Dump 2) und nach (RAM-Dump 3-1) Zuweisung einer "Neuen Identität" im Browsing Artefakte im Arbeitsspeicher gefunden. Nach dem Browsing Szenario mit geöffnetem Browser konnten die meisten Artefakte identifiziert werden. Dabei wurden am häufigsten URL Artefakte in Firefox Prozessen

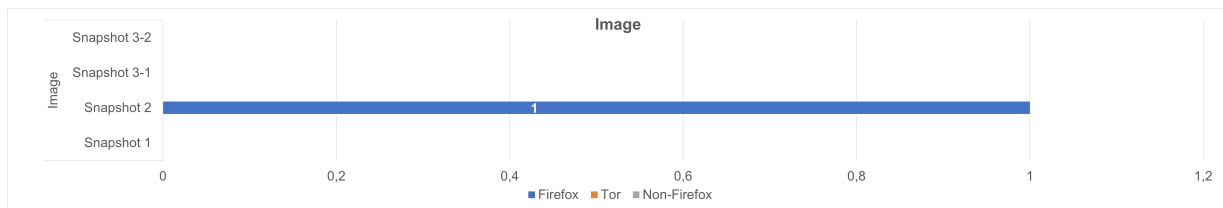


Abbildung 5.16: Image

gefunden. Zudem konnte hier das E-Mail Passwort im Klartext lokalisiert werden. Nach Schließen des Browsers konnten im dritten Snapshots URLs im DNSCache Windows Service gefunden werden. Nach leeren des Caches und Beenden des DNSCache Services konnten wurden keine Artefakte gefunden. Die Erstellung einer "Neuen Identität" reduzierte dabei deutlich die gefundenen Artefakte im RAM. In keinem RAM-Speicherabbild konnten HTML-Fragmente der im Browsing-Szenario besuchten Seiten identifiziert werden. Weiterhin wurde zwei Mal das Passwort des Google-Accounts im Klartext gefunden.

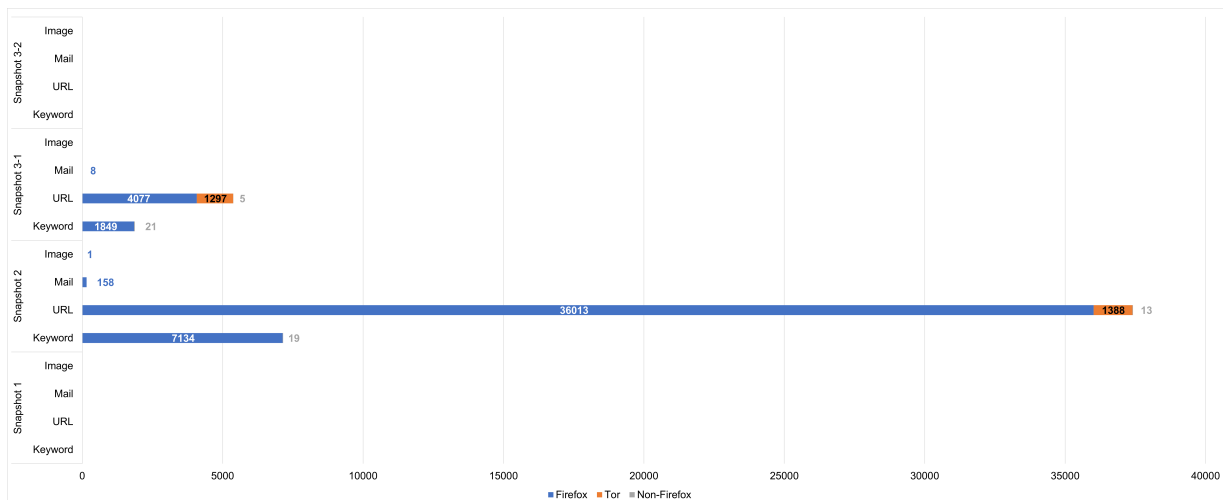


Abbildung 5.17: Summary

Registry

Wie in der Methodik in Kapitel X beschrieben, teilt sich die Analyse der Registry sowohl Common als auch Uncommon Locations. Weder in den Process Monitor SSetValueOperations noch über die Stringsuche in den System- und User-Hives konnten PB Artefakte gefunden werden. Eine detaillierte Analyse der Registry ist im Anhang X beschrieben.

5.3 Chrome

Uncommon Locations

o Autopsy Keyword-Suche: > Chrome and Edge produced five artefacts as reported by both tools. (FTK, Autopsy) [5] -> Artefakte werden nicht genannt! > only two temporary files (Figure 7) were recovered with Minitool Power Data Recovery but it was a dead end; Location: appdata/.../Chrome/.../Preferences/RF1533fa.TMP [4] > pagefile.sys file showed no traces at all [22]

5.4 Brave

6 Vergleich der Browser

- Zusammenfassung: Vergleich Tor v. Firefox und Brave v. Chrome

Firefox vs Tor: > Gestacktes Balkendiagramm zu veränderten SQLite DBs => Erst bei Vergleich mit Tor!

SSessionstore-Backup"fehlt in Tor

TODO: Kreisdiagramme/Balkendiagramme mit Gesamtzahl an (Non-)Firefox Yarascan-Treffer erst im Vergleich mit Tor - Firefox v. Chrome (SStandardbrowser") - Tor v. Brave (SSichere Browser") - Zum Schluss: Eine große Tabelle"mit den wichtigsten Kategorien?

7 Diskussion

> Artefakte im DNS Cache: [24] ■ DNS-Caching ist eine Bedrohung für private Browsing ■ Diese Schwachstelle entsteht, weil das Betriebssystem DNS-Anfragen des Browsers im Cache speichert, unabhängig davon, ob der Browser im privaten Modus ist oder nicht ■ Mehrere Jahre nach der Meldung dieser Schwachstelle besteht sie immer noch in allen Browsern fort ■ Es wurden einige Erweiterungen von Drittanbietern entwickelt, um dieses Problem zu beheben, aber keine davon wurde von den Browserherstellern übernommen.

> Viele RAM-Artefakte - Firefox [15] ■ Darcie et al. (2014) fanden Beweise für das Web-Browsing in Form von JPEG- und HTML-Dateien in Live-Forensik, aber eine statische Forensik war erfolglos. ■ Eine vorherige Live-Forensik-Analyse des Firefox-Browsers zeigte, dass Artefakte aus einer privaten Browsing-Sitzung aus dem Speicher wiederhergestellt werden konnten. (Findlay and Leimich, 2014).

> IE hinterlässt viele Spuren im Gegensatz zu Ergebnissen: [13] o hidden folders are usually stored at C:/Users/User/AppData o evidence searches are conducted extensively in the C: partition o bookmarks remain and can be viewed o downloads remain in the downloads folder until the user manually deletes them o CacheView trace entire URL and browsing histories including the temporary files CacheView enables to find the image's URL and from specific website

> Urteil über die Privatheit von Tor nach [15] The design aim of preventing Tor from writing to disk (Perry et al., 2018) is not achieved in this version. ■ Configuration files, downloaded files, and browserrelated data are recoverable from the file system. ■ Significant data-leakage from the browsing session occurred: HTTP header information, titles of web pages and an instance of a URL were found in registry files, system files, and unallocated space. ■ The data-leakage contained the German word for 'search' in reference to a Google search. This hints at the locale of the Tor server used to exit the network (exit relay). The Tor Project's design aim of enabling secure deletion of the browser (Sandvik, 2013) is not achieved in this version. ■ References to: the installation directory, Firefox SQLite files, bridging IPs/ports, default bookmarks, Tor-related DLLs and Tor product information were all recovered after the browser was deleted. ■ In a scenario where the operating system paged memory, an instance

Weiterführende Arbeiten: > Cross-mode interference [9]: o the Chrome://memory page displays all the opened tabs in the browser regardless if they are in the usual or private mode -> Nicht mehr aktuell -> Stattdessen: Chrome Task-manager (Ctrl + Esc), Funktioniert auch bei Firefox > Unser Scope: Process Monitor nach Prozessnamen gefiltert - Weiterführend: Nach Pathnamen filtern: "Common Locations"

> Für wen wird Browser entwickelt > Warum und für wen wird Private Browsing analysiert? > Ist das Auffinden privater Browsing Artefakte Schuld von Browser Entwicklern? (Oder Schuld des Betriebssystems, wie in (TODO!) erwähnt)

> bei Process Monitor nur nach Browser-Prozessen gefiltert

> Warum in Literatur nur Windows untersucht?

Tor: Unsere Mission: Menschenrechte und Freiheiten durch die Entwicklung und Verbreitung von Open Source Anonymitäts- und Privatsphäre-Technologien zu fördern, ihre ungehinderte Verfügbarkeit zu unterstützen und ihr Verständnis in Wissenschaft und der Allgemeinheit zu vergrößern."

8 Fazit

Einleitend werden Struktur, Motivation und die abgeleiteten Forschungsfragen diskutiert.

Appendices

Firefox Common Locations

Process Monitor WriteFile Operations

Gemäß Versuchsdurchführung in Abbildung X (TODO!) wurden für Firefox mit dem Process Monitor Tool zwei Logfiles erstellt. Diese Dateien enthalten alle aufgezeichneten Prozessaktivitäten während und nach dem Browsing Szenario. Zunächst werden die beiden Logfiles gemäß Methodik in Kapitel X (TODO!) in Excel aufbereitet. Im Anhang X (TODO!) ist dazu eine Tabelle mit allen in den gefilterten Logfiles identifizierten Dateien aufgeführt. Dabei wurde für jede Datei vermerkt ob und wie sie wiederherstellbar war, mit welchem Tool die Datei analysiert wurde und ob PB Artefakte enthalten sind.

Abbildung X (TODO!) zeigt diese Tabelle in reduzierter Darstellung. Dazu wurden ausschließlich wiederherstellbare Dateien aufgeführt. Die Dateien wurden in die fünf Kategorien "Cache", "datareporting", "Sessionstore-Backup" und "Sonstige" Dateien eingeordnet. Für jede Datei wurde vermerkt, ob in der entsprechenden Logfile PB Artefakte geschrieben wurden. Dies trifft für keine der identifizierten Dateien zu.

		Logfile 1	Logfile 2
Cache	cache2\entries\037778A55E187E9BED3390289866D09402D6C913	Keine PB Artefakte	Keine Schreiboperation
	cache2\entries\1223A0378B8971FA4CD25EA1731C80B2B1676B42	Keine PB Artefakte	Keine Schreiboperation
	cache2\entries\250EE2BC03AFF526F1A1C3DB212A79DE3EB60D5E	Keine PB Artefakte	Keine Schreiboperation
	jumpListCache\ZKJGVJPzPe7w4wOKwEY0Jw==.ico	Keine PB Artefakte	Keine Schreiboperation
	cache2\index.log	Keine Schreiboperation	Keine PB Artefakte
Datareporting	cache2\index	Keine Schreiboperation	Keine PB Artefakte
	datareporting\glean\events\pageload	Keine PB Artefakte	Keine Schreiboperation
	datareporting\glean\db\data.safe	Keine PB Artefakte	Keine PB Artefakte
	*datareporting\archived\2023-05\1683405837882.9102466b-e465-4ecb-810f-74ae90c64c63.new-profile.jsonlz4	Keine Schreiboperation	Keine PB Artefakte
	*datareporting\archived\2023-05\1683405837905.86f4c992-6329-415b-8c29-911a2d4b7f9d.event.jsonlz4	Keine Schreiboperation	Keine PB Artefakte
SQLite	*datareporting\archived\2023-05\1683405837939.abf8b065-41a4-4e94-a044-1cead61e396a.main.jsonlz4	Keine Schreiboperation	Keine PB Artefakte
	storage\permanent\chrome\idb\3870112724rsegmnoittet-es.sqlite	Keine PB Artefakte	Keine Schreiboperation
	storage\permanent\chrome\idb\1657114595AmcateirvtiSty.sqlite	Keine PB Artefakte	Keine PB Artefakte
	places.sqlite	Keine PB Artefakte	Keine PB Artefakte
	cookies.sqlite	Keine PB Artefakte	Keine Schreiboperation
	formhistory.sqlite	Keine PB Artefakte	Keine Schreiboperation
	webappsstore.sqlite	Keine Schreiboperation	Keine PB Artefakte
	favicons.sqlite	Keine PB Artefakte	Keine Schreiboperation
	storage.sqlite	Keine PB Artefakte	Keine Schreiboperation
	sessionstore-backups\recovery.jsonlz4	Keine PB Artefakte	Keine PB Artefakte
Sonstige	prefs-1.js	Keine PB Artefakte	Keine PB Artefakte
Dateien	xulstore.json	Keine PB Artefakte	Keine PB Artefakte

Abbildung .1: Tabelle mit wiederherstellbaren Dateien: Logfile 1 vs. Logfile 2

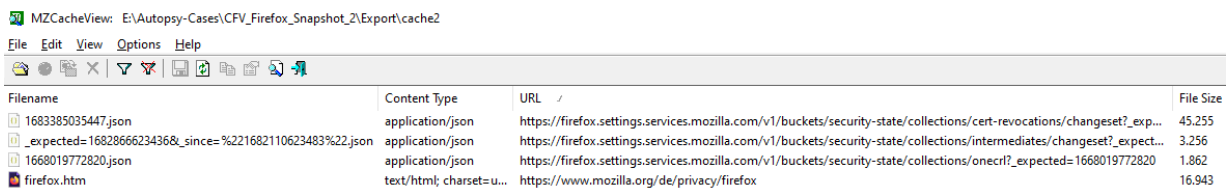
Bei detaillierter Untersuchung der Dateien, können zwei Pfade identifiziert werden, in die Firefox während des Versuchs Dateien schreibt. Nur die Dateien in der Cache Kategorie sind im Local Pfad gespeichert.

Local C:\Users\

Roaming C:\Users\

In Tabelle X (TODO!) sind die Dateien je nach Speicherort "Local" (Hellblau) oder "Roaming" (Dunkelblau) entsprechend eingefärbt.

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 Dateien im Format `\cache2\entries\<ID>` werden dem Cache zugeordnet und im Local Pfad gespeichert. Wie in Kapitel X beschrieben, können diese Dateien mit dem Tool MZCacheView eingelesen werden. Wie in Abbildung X gezeigt, konnten im Cache-Ordner im zweiten Snapshot drei JSON Dateien identifiziert werden. Dabei handelt es sich um Zertifikatsdateien, die von der Öne 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. Weiterhin befindet sich im Cache das HTML-Dokument der Firefox Datenschutzseite, welche sich beim ersten Start des Browsers automatisch öffnete. Weitere Cache Dateien konnten in keinem Snapshot gefunden werden. Die Indexdatei `\cache2\index` dient als Datenbank im Cache.



MZCacheView: E:\Autopsy-Cases\CFV_Firefox_Snapshot_2\Export\cache2

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/onecrf?_expected=1668019772820	1,862
firefox.htm	text/html; charset=u...	https://www.mozilla.org/de/privacy/firefox	16,943

Abbildung .2: Tabelle mit wiederherstellbaren Dateien: Logfile 1 vs. Logfile 2

Sie ermöglicht dem Firefox-Browser, schnell auf die zwischengespeicherten Ressourcen zuzugreifen und diese effizient zu verwalten. Sowohl mit HxD als auch dem Tool FirefoxCache2 konnten keine PB Artefakte identifiziert werden. *** TODO: Erst in Logfile 2 geschrieben ***

Schließlich enthält die Datei `\jumpListCache\ZKJGVJPzPe7w4w0KwEY0jw==.ico` ein 64x64 Pixel großes Mozilla Logo. Dieses Logo ist keinem Schritt aus dem Browsing Szenario zuzuordnen

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. Die Datei `data.safe.bin` enthält verschlüsselte und anonyme Informationen über die Nutzung des Browsers. In HxD konnten keine PB Artefakte gefunden werden *** TODO: Vergleich Logfile 1 vs 2 ***

Dateien im Foremat `\datareporting\glean\db\<Profilname>.new-profile.jsonlz4` speichern Informationen über das Firefox-Profil, das von Glean verwendet wird. Wie in Kapitel X beschrieben, lassen sich Dateien, im proprietären `jsonlz4`-Format mit dem Tool `dejsonlz4` dekomprimieren. Die entstandene JSON Datei wird mit dem Notepad++ JSON Plugin untersucht. Dabei konnten keine PB Artefakte gefunden werden. *** TODO: Nur in Logfile 2 geschrieben ***

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 ein Online-Tool zur Analyse von *Sessionstore-Backup* Dateien. In der Sitzungswiederherstellung konnten wie in Abbildung X gezeigt lediglich die automatisch geöffnete Seite über Firefox Datenschutzhinweise identifiziert werden. *** TODO: Logfile 1 vs Logfile 2 ***

Closed Window 1

Tab 1

Firefox Datenschutzhinweis — Mozilla [5/6/2023, 10:24:59 PM]

<https://www.mozilla.org/de/privacy/firefox/>

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

Abbildung .3: Tabelle mit wiederherstellbaren Dateien: Logfile 1 vs. Logfile 2

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

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 mit HxD keine PB Artefakte gefunden werden. *** TODO: Vergleich Logfile 2 ***

Schließlich speichert die Datei xulstore.json benutzerspezifische Anpassungen und Konfigurationen für den Firefox-Browser. In der Datei konnten mit Notepad++ keine PB Artefakte gefunden werden. *** TODO: Vergleich Logfile 2 ***

SQLite Datenbanken

Wie in Kapitel X (Methodik, TODO!) erwähnt, werden SQLite Datenbanken als Datenstrukturen für Nutzerdaten genauer untersucht. Mithilfe der Process Monitor Logfiles wurden die in Tabelle X dargestellten SQLite-Datenbanken für Firefox identifiziert:

Jede dieser Datenbanken wurde in allen vier Snapshots miteinander verglichen. Die Dateiextraktion und Dateianalyse erfolgte analog zur Methodik in Kapitel X (TODO!). Die Ergebnisse wurden in Tabelle X (TODO!) dargestellt.

	File	Snapshot 1: Browser Installation	Snapshot 2: After Browsing Scenario, Browser open		Snapshot 3: After Browsing Scenario, Browser closed		Snapshot 3: Browser closed	
			Vor WAL	Nach WAL	Vor WAL	Nach WAL	Vor WAL	Nach WAL
SQLite	places.sqlite	N/A	Initialisiert, Zellen: Einträge für autom. geöffnete Seiten	no diff	Indizes bei vorhandenen Seiten aktualisiert	no diff	no diff	no diff
	cookies.sqlite	N/A	Leer initialisiert (Nur Spaltennamen)	leer	leer	leer	leer	leer
	storage.sqlite	N/A	Leer initialisiert (Nur Spaltennamen)	leer	leer	leer	leer	leer
	favicons.sqlite	N/A	Leer initialisiert (Nur Spaltennamen)	leer	leer	leer	leer	leer
	webappsstore.sqlite	N/A	N/A	N/A	Leer initialisiert (Nur Spaltennamen)	leer	leer	leer
	formhistory.sqlite	N/A	Leer initialisiert (Nur Spaltennamen)	leer	leer	leer	leer	leer
	1657114595AmcateirvtiSty.sqlite	N/A	Initialisiert, 12 Zelle: "origin: chrome"	no diff	Einträge (Binärdaten) eingefügt, keine PB Artefakte (HxD)	no diff	no diff	no diff
	3870112724rsegmnottet-es.sqlite	N/A	Initialisiert, 12 Zelle: "origin: chrome"	no diff	no diff	no diff	no diff	no diff
			Leer					
			Unverändert (nicht-leer)					
			Neuer (nicht-leerer) Inhalt					

Abbildung .4: Comparison of found PB artifacts between RAM Dumps

Nach Browser-Installation (Snapshot 1) existierte noch keine der SQLite-Dateien.

Nach dem Browsing Szenario (Snapshot 2) wurde festgestellt, dass alle SQLite-Datenbanken initialisiert wurden, außer webappsstore.sqlite. Dabei wurden in places.sqlite die automatisch im normalen

Modus geöffnete Datenschutzhinweise Seite eingetragen. In restlichen Datenbanken wurden leer initialisiert, nur die Spaltennamen wurden eingetragen. Der Inhalt aller erstellten Datenbanken blieb nach Durchführung von PRAGMA WAL Checkpoints unverändert.

Nach Schließen des Browsers (Snapshot 3) wurden in `places.sqlite` die Indizes bei 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.

Nach herunterfahren der VM (Snapshot 4) gab es keine Änderungen in den SQLite Datenbanken, auch nach Durchführung der PRAGMA WAL Checkpoints.

Somit wurden in den SQLite Datenbanken von Firefox keine zurückverfolgbaren PB Artefakte im privaten Modus hinterlassen.

Mithilfe des Process Monitors wurde festgestellt, dass sowohl während des Browsing Szenarios (Logfile 1) als auch danach (Logfile 2) Inhalte in Dateien geschrieben wurden. Wie zusammenfassend in Abbildung X (TODO!) dargestellt, wurde mit Ausnahme der Datareporting Dateien gab es in Logfile 1 stets mehr oder genauso viele Schreiboperationen wie in Logfile 2. Keine Schreiboperation hinterließ jedoch Private Browsing Artefakte.

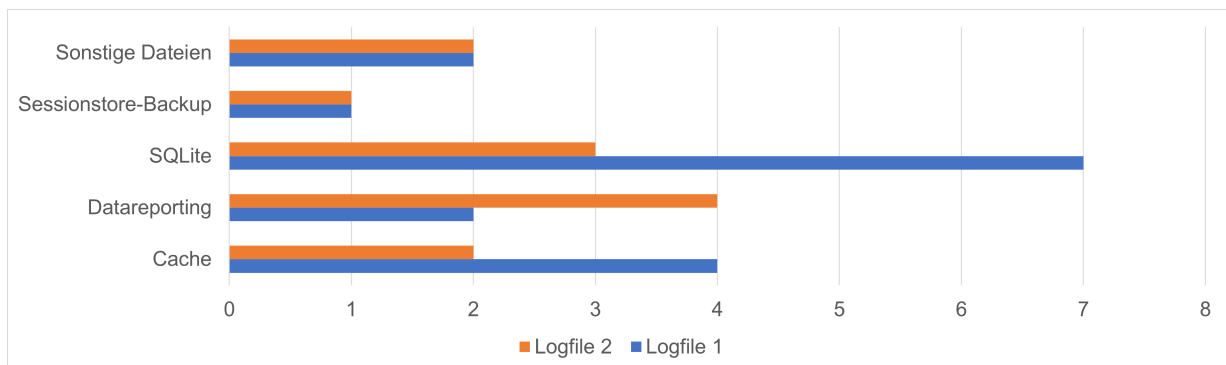


Abbildung .5: Comparison of found PB artifacts between RAM Dumps

Firefox Uncommon Locations

Analyse mit Autopsy

Gemäß Methodik in Kapitel X wurden die Dateien der Kategorien "Web Bookmarks", "Web Cookies", "Web History" sowie "Web Categories" analysiert. Beim Vergleich der Festplattenabbilder wurde festgestellt, dass ein Snapshot stets die kategorisierten Dateien des vorherigen Snapshots enthielt. Es sind innerhalb einer Kategorie nur neue Dateien dazugekommen. Somit enthält Snapshot 4 in jeder Kategorie alle Dateien der vorherigen Snapshots.

*** TODO: Edge vorinstalliert erwähnen ***

Web Bookmarks Bereits vor Durchführung des Browsing Szenarios enthielt Firefox im ersten Snapshot die 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	Data Source
 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	CFV_Firefox_Klon_Snapshot_3.img

Abbildung .6: Autopsy Web Bookmarks

Web Cookies Auch diese Kategorie enthält bereits vor Beginn des Browsing Szenarios zehn 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. Die Datei enthält. 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	CFV_Firefox_Klon_Snapshot_3.img
 WebCacheV01.dat			15	www.bing.com	2023-05-06 19:51:24 MESZ	MUIDB	31708C5FC3CF47068AFADC1CB47D0111	Microsoft Edge Analyzer	bing.com	CFV_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	CFV_Firefox_Klon_Snapshot_3.img
 WebCacheV01.dat			15	bing.com	2023-05-06 19:50:17 MESZ	SRCHUID	V=28GUID=B2C50ADB8CB984234A9FE14DB81DCB91D8dm...	Microsoft Edge Analyzer	bing.com	CFV_Firefox_Klon_Snapshot_3.img
 WebCacheV01.dat			15	bing.com	2023-05-06 19:50:17 MESZ	SRCHJSR	DOB=20230506	Microsoft Edge Analyzer	bing.com	CFV_Firefox_Klon_Snapshot_3.img
 WebCacheV01.dat			15	bing.com	2023-05-06 19:50:20 MESZ	SRCH#PGUSR	SRCHLANG=de&LUT=16834026192238JPMH=des20405&L...	Microsoft Edge Analyzer	bing.com	CFV_Firefox_Klon_Snapshot_3.img
 WebCacheV01.dat			15	bing.com	2023-05-06 19:50:19 MESZ	CortanaAppUID	C164AA3A4D47E127DC66AD915CFD04C	Microsoft Edge Analyzer	bing.com	CFV_Firefox_Klon_Snapshot_3.img
 WebCacheV01.dat			15	bing.com	2023-05-06 19:55:22 MESZ	ANON	A=A3B5B679A14D59B0AA027635FFFFFFF	Microsoft Edge Analyzer	bing.com	CFV_Firefox_Klon_Snapshot_3.img
 WebCacheV01.dat			15	live.com	2023-05-06 19:50:30 MESZ	MUID	118A534093A9626528C5404997A966B8	Microsoft Edge Analyzer	live.com	CFV_Firefox_Klon_Snapshot_3.img
 WebCacheV01.dat			15	login.live.com	2023-05-06 19:51:06 MESZ	__Host-MSAAUTHP		Microsoft Edge Analyzer	live.com	CFV_Firefox_Klon_Snapshot_3.img

Abbildung .7: Autopsy Web Cookies

Web History Diese Kategorie listet alle Dateien mit gespeichertem Suchverlauf auf. Vor Beginn des Browsing Szenarios (Snapshot 1) enthält die Kategorie ebenfalls 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 Kapitel X. 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 X (TODO!) über den gemeinsamen VM-Ordner zum Analyse-Rechner transportiert wurde. Ergänzt wird das in Snapshot 3 durch den Eintrag file:///Z:/Logfile_2, dem zweiten Process Monitor Logfile. In Snapshot 4 werden in dieser Kategorie keine neuen Dateien erfasst.



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	CFV_Firefox_Klon_Snapshot_3.img	
 WebCacheV01.dat			15	https://login.live.com/oauth20_desktop.srf?lc=1031	2023-05-06 19:51:06 MESZ			Microsoft Edge Analyzer	live.com	CFV_Firefox_Klon_Snapshot_3.img	Forensik
 WebCacheV01.dat			15	https://login.live.com/oauth20_authorize.srf?client_...	2023-05-06 19:51:08 MESZ			Microsoft Edge Analyzer	live.com	CFV_Firefox_Klon_Snapshot_3.img	Forensik
 WebCacheV01.dat				file:///Z:/Logfile_1	2023-05-06 20:29:36 MESZ			Microsoft Edge Analyzer		CFV_Firefox_Klon_Snapshot_3.img	Forensik
 WebCacheV01.dat				file:///Z:/Logfile_2	2023-05-06 20:44:19 MESZ			Microsoft Edge Analyzer		CFV_Firefox_Klon_Snapshot_3.img	Forensik

Abbildung .8: Autopsy Web History

Web Categories Diese Kategorie klassifiziert im Speicherabbild gefundene Browsing Artefakte nach Inhalt. Vor Beginn des Browsing Szenarios (Snapshot 1) werden hier bereits zwei 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 in den Snapshots 2 bis 4.

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/vol3/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/vol3/Users/Forensik/AppData/Local/Microsoft/Windows/WebCache/WebCacheV01.dat

Abbildung .9: Autopsy Web Categories

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`. Die eingetragene Firefox-Standardseite deckt sich mit den Ergebnissen der Common Locations in Tabelle X. 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. In keiner der Kategorien konnten private Browsing Artefakte identifiziert werden.

Firefox 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 X ist der Anteil der Schreiboperationen je Kategorie für beide Logfiles gezeigt.

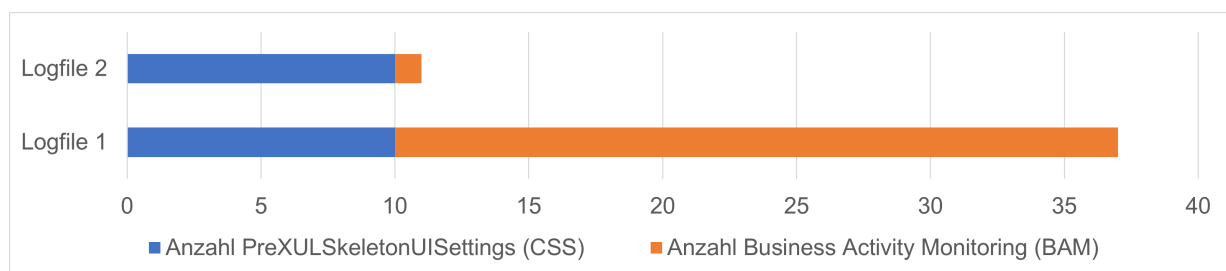


Abbildung .10: Comparison of found PB artifacts between RAM Dumps

PreXULSkeletonUISettings Der "PreXULSkeletonUISettings"Registry Key enthält Einstellungen für die Benutzeroberfläche (UI) des Firefox-Browsers, insbesondere für das sogenannte SSkeleton 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.

- 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\Harddisk\<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.

Stringsuche in Registry Hives

Gemäß Methodik in Kapitel X 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 X (TODO!) aus jedem Snapshot extrahiert und mithilfe des Registry Explorers nach PB Artefakten durchsucht. Dabei wurde in keinem Snapshot in keinem Hive ein PB Artefakt gefunden.

Tor Common Locations

Process Monitor WriteFile Operations

Bei der Versuchsdurchführung für den Tor-Browser gemäß Kapitel X wurden drei Process Monitor Logfiles erstellt. Diese Dateien enthalten alle aufgezeichneten Prozessaktivitäten während des Browsing Szenarios, nach dem Erzeugen einer "Neuen Identität" sowie nach Schließen des Browsers. Eine gemäß Methodik in Kapitel X verarbeitete Tabelle mit allen in den Logfiles identifizierten Dateien ist im Anhang X aufgeführt. Für jede Datei wurde vermerkt ob und wie sie wiederherstellbar war, mit welchem Tool die Datei analysiert wurde und ob PB Artefakte enthalten sind.

In Tabelle X (TODO!) sind die ausschließlich wiederherstellbaren Dateien aufgeführt. Die Dateien wurden in die vier Kategorien "Cache", "datareporting", und "Sonstige Dateien" eingeordnet. In keiner der identifizierten Dateien konnten PB Artefakte gefunden werden.

Kategorie	Datei	Logfile 1	Logfile 2	Logfile 3
Cache	Cache\profile.default\startupCache\startupCache.8.html	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
Datareporting	datareporting\glenn\dd\data.safe.bin	Keine PB Artefakte	Keine Schreiboperationen	Keine PB Artefakte
	datareporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
SQLite	storage\permanent\chrome\idb\3870112724\segment-es.sqlite	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	storage\permanent\chrome\idb\1657114535\Amc\cache\sty.sqlite	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	places.sqlite	Keine PB Artefakte	Keine PB Artefakte	Keine PB Artefakte
	cookies.sqlite	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	storage.sqlite	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	navicons.sqlite	Keine PB Artefakte	Keine Schreiboperationen	Keine PB Artefakte
	webappstore.sqlite	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	formhistory.sqlite	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	addonStartup.json.ls4	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	AlternateServices.txt	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
Sonstige Dateien	broadcast-listeners.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	extensions.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	extensions\staged\73a6fe31-595d-460b-a920-fcc0f8843232.xpi	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	tonion-alaises.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	prefs-1.jp	Keine PB Artefakte	Keine PB Artefakte	Keine PB Artefakte
	security_state\data.safe.bin	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	settings\data.safe.bin	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	SiteSecurityServiceState.txt	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	SiteSecurityServiceState-1.txt	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	profile.default\kustore.json	Keine PB Artefakte	Keine Schreiboperationen	Keine PB Artefakte
	profile.default\cert_override.txt	Keine Schreiboperationen	Keine PB Artefakte	Keine Schreiboperationen
	profile.default\enumerated_devices.txt	Keine Schreiboperationen	Keine PB Artefakte	Keine Schreiboperationen
	profile.default\sessionCheckpoints.json.tmp	Keine Schreiboperationen	Keine Schreiboperationen	Keine PB Artefakte
	storage\default\moz-extension+++3041a34e-316a-4fca-8ea0-53f966d7a11f\userContextId=42343672351\metadata-v2	Keine Schreiboperationen	Keine Schreiboperationen	Keine PB Artefakte
	Caches			
	Profile Default			

Abbildung .11: Tabelle mit wiederherstellbaren Dateien: Logfile 1 vs. Logfile 2

Ähnlich wie bei der Analyse der Schreiboperationen von Firefox in Kapitel X (TODO!), konnten für den Tor-Browser zwei Pfade identifiziert werden, wo sich Dateien befinden, in die geschrieben wurde.

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 Tabelle X (TODO!) 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.defaultPfad. 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. Bei Untersuchung mit HxD konnten keine PB Artefakte gefunden werden.

Datareporting Im "DatareportingOrdner wird vom Tor-Browser die Datei `\datareporting\data.safe.bin` geschrieben. Sie enthält verschlüsselte und anonyme "Glean"-Informationen über die Nutzung des Browsers. 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. 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 ist ein Mechanismus. Dieser 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.

Weiterhin wird während des Browsing Szenarios die Datei `\extensions\staged\73a6fe31-595d-460b-a920-fcc0f8` geschrieben. Dabei handelt es sich um das von Tor verwendete "NoScriptAddOn 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-Software-Tool, das von Journalisten und Nachrichtenorganisationen verwendet wird, um anonyme Informationen von Whistleblowern entgegenzunehmen. Es ermöglicht den sicheren Austausch von Informationen, ohne die Identität der Quelle preiszugeben. Whistleblower können über .onion-URLs auf die SecureDrop-Websites zugreifen und vertrauliche Dokumente oder Nachrichten sicher und anonym übermitteln.

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. 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 X (TODO!) verwaltet und beschreibt.

Wie bei der Analyse der SQLite-Datenbanken bei Firefox wird die Entwicklung von Dateiinhalt in allen fünf Festplatten-Images der Snapshots 1, 2, 3-1, 3-2 und 4 betrachtet. Die Ergebnisse sind in Tabelle X (TODO!) dargestellt. Nach Browser-Installation wurde noch keine SQLite-Datei angelegt (Snapshot

	File	Snapshot 1: Browser installation	Snapshot 2: After Browsing Scenario, Browser open		(For only) Snapshot 3-1: After Identity reset		Snapshot 3: After Browsing Scenario, Browser closed		Snapshot 4: VM Shutdown	
			For WAL	Back WAL	For (Diff)	Back WAL	For	Back WAL	For WAL	Back WAL
SQLite	places.sqlite	N/A	Initialisiert. Zeilen: Online URLs für Tor Standardseiten, wie "The Tor Blog" oder "Tor Browser Manual" und Spenden-Seite (http://revoipcd4b3i3daz3yqzgm3tq4f6b0b2d2dsgzazg35de-united.onion)	no diff	Indizes bei vorhandenen Seiten aktualisiert	no diff	Indizes bei vorhandenen Seiten aktualisiert	no diff	no diff	no diff
	cookies.sqlite	N/A	Leer initialisiert (Nur Spaltennamen)	leer	leer	leer	leer	leer	leer	leer
	storage.sqlite	N/A	Leer initialisiert (Nur Spaltennamen)	leer	leer	leer	leer	leer	leer	leer
	favicons.sqlite	N/A	Initialisiert. Einträge mit Präfix "Fake-favicon-uri" für Tor Standardseiten, wie "The Tor Blog" oder "Tor Browser Manual" und Spenden-Seite (http://revoipcd4b3i3daz3yqzgm3tq4f6b0b2d2dsgzazg35de-united.onion)	no diff	no diff	no diff	in allen drei Tabellen Indizes aktualisiert	no diff	no diff	no diff
	webappstor.sqlite	N/A	Leer initialisiert (Nur Spaltennamen)	leer	leer	leer	leer	leer	leer	leer
	formhistory.sqlite	N/A	Leer initialisiert (Nur Spaltennamen)	leer	leer	leer	leer	leer	leer	leer
	9557114595Amcsh9m18ty.sqlite	N/A	Leer initialisiert (Nur Spaltennamen)	leer	leer	leer	leer	leer	leer	leer
	3570112724rsgmeo0t0t0ez.sqlite	N/A	Initialisiert. 1 Zeile: "origin: chrome"	no diff	no diff	no diff	gleich bleibend	no diff	no diff	no diff
			Leer							
			Unverändert							

Abbildung .12: Comparison of found PB artifacts between RAM Dumps

1).

Während des Browsing Szenarios wurden alle Datenbänkte 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 wie bei Firefox in Kapitel X 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) 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.

Im Balkendiagramm X (TODO!) 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. *** TODO: Was noch? ***

Tor Uncommon Locations

Analyse mit Autopsy

Wie bei Firefox in Kapitel X wurden keine der kategorisierten Dateien gelöscht. Somit befanden sich im letzten Festplatten-Image des Snapshots 4 alle kategorisierten Dateien der vorherigen Festplatten-Images

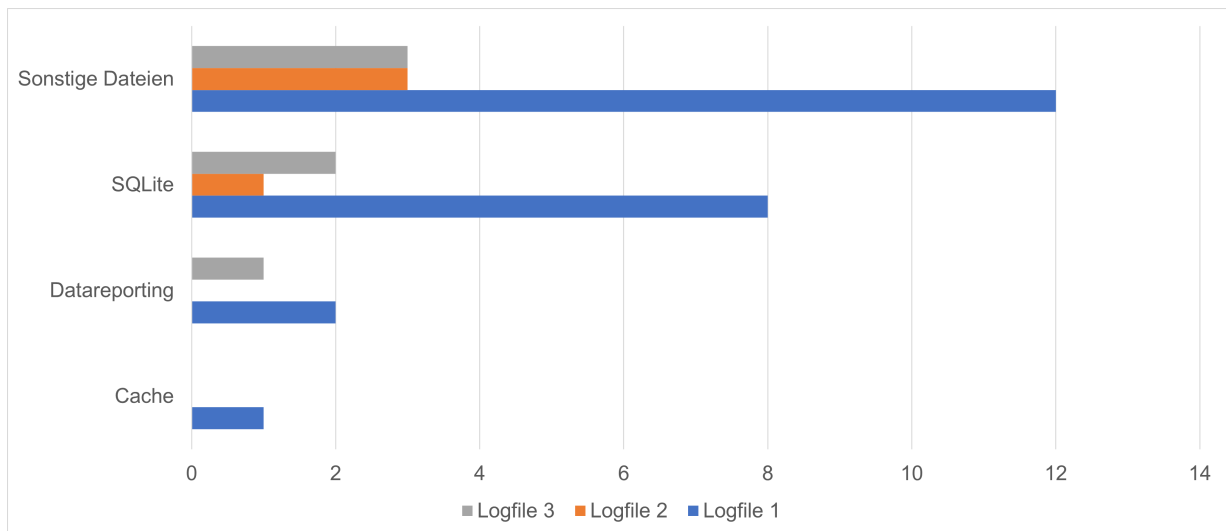


Abbildung .13: Comparison of found PB artifacts between RAM Dumps

Web Bookmarks Wie in Abbildung X (TODO!) 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 .14: Autopsy Web Bookmarks

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

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	V=28&GUID=62F5FD78E9D94446BAFDF9DEC81881038dm...	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 .15: Autopsy Web Cookies

Web History Zwei Einträge mit Browsingverläufen wurden im Festplatten-Image des ersten VM-Snapshots in der Datei WebCacheV01.dat geschrieben. Dabei handelt es sich zweimal um die Outlook-Startseite, obwohl diese nie bei der Versuchsdurchführung geöffnet wurde. Wie bei Firefox in Kapitel

X (TODO!) 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 .16: Autopsy Web History

Web Categories Autopsy klassifizierte im Festplatten-Image des ersten VM-Snapshots den Eintrag `bing.com` als `SSuchmaschine` und `live.com` als `"Web-Email"`. 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 .17: Autopsy Web Categories

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 registrierte nicht die in der `places.sqlite` Datenbank geschriebenen `.onion`-URLs der Tor-Standardseiten.

Tor Registry

Process Monitor SetValue Operations

Bei Betrachtung als Common Locations werden gemäß Methodik in Kapitel X alle `SSetValue`-Schreiboperationen in den Process Monitor Logfiles für die Prozesse `"tor.exe"` und `"firefox.exe"` untersucht.

Dabei wurden die gleichen beiden Registry Keys identifiziert, wie es bereits bei Untersuchung der Firefox Registry in Kapitel X (TODO!) der Fall war: `PreXULSkeletonUISettings` und `Business Activity Monitoring`. In keinem Registry-Key befinden sich PB Artefakte. Wie in Abbildung X dargestellt, wurden beide Registry Keys annähernd gleich oft geschrieben. Bei Vergleich der drei Process Monitor Logfiles 1, 2 und 3 nimmt Anzahl der Registry `SSetValue`-Operationen bei Logfile 2 und 3 kontinuierlich ab.

Stringsuche in Registry Hives

Bei Betrachtung der Registry als Uncommon Locations, wurden die in Tabelle X im Kapitel der Methodik X 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.

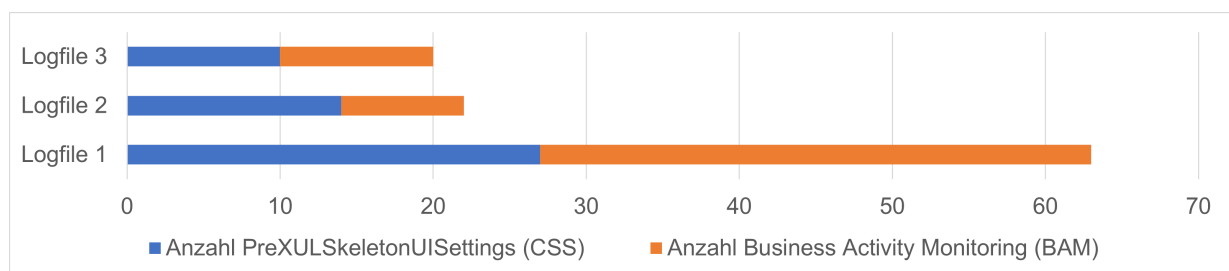


Abbildung .18: Comparison of found PB artifacts between RAM Dumps

Literatur

- [1] Gaurav Aggarwal u. a. "An Analysis of Private Browsing Modes in Modern Browsers." In: *USENIX security symposium*. 2010, S. 79–94.
- [2] 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.
- [3] Howard Chivers. "Private browsing: A window of forensic opportunity". In: *Digital Investigation* 11.1 (2014), S. 20–29.
- [4] Hasan Fayyad-Kazan u. a. "Forensic analysis of private browsing mechanisms: Tracing internet activities". In: (2021).
- [5] 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.
- [6] Ms Pooja Gupta. "Capturing Ephemeral Evidence Using Live Forensics". In: *IOSR J. Electron. Commun. Eng* (2013), S. 109–113.
- [7] Meenu Hariharan, Akash Thakar und Parvesh Sharma. "Forensic Analysis of Private Mode Browsing Artifacts in Portable Web Browsers Using Memory Forensics". In: *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)*. IEEE. 2022, S. 1–5.
- [8] Nihad A Hassan. *Digital forensics basics: A practical guide using Windows OS*. Apress, 2019.
- [9] Ashley Hedberg. *The privacy of private browsing*. Techn. Ber. Technical Report, Tufts University, MA, USA, 2013.
- [10] Graeme Horsman u. a. "A forensic examination of web browser privacy-modes". In: *Forensic Science International: Reports* 1 (2019), S. 100036.
- [11] 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.
- [12] 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.
- [13] 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.

-
- [14] 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.
- [15] Matt Muir, Petra Leimich und William J Buchanan. "A forensic audit of the tor browser bundle". In: *Digital Investigation* 29 (2019), S. 118–128.
- [16] 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.
- [17] Junghoon Oh, Seungbong Lee und Sangjin Lee. "Advanced evidence collection and analysis of web browser activity". In: *Digital investigation* 8 (2011), S62–S70.
- [18] 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.
- [19] Daniel Perdices u. a. "Web browsing privacy in the deep learning era: Beyond VPNs and encryption". In: *Computer Networks* 220 (2023), S. 109471.
- [20] Digvijaysinh Rathod. "Darknet forensics". In: *future* 11 (2017), S. 12.
- [21] 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.
- [22] Huwida Said u. a. "Forensic analysis of private browsing artifacts". In: *2011 International Conference on Innovations in Information Technology*. IEEE. 2011, S. 197–202.
- [23] Priya P Sajan u. a. "Tor Browser Forensics". In: *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12.11 (2021), S. 5599–5608.
- [24] 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.
- [25] 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.