

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	5
2.1. Private Browsing	5
2.2. Angreifermodell	6
2.3. Browser Forensics und Artefakte	6
3. Ziel der Arbeit	8
4. Methodik	11
4.1. Preparation Stage	12
4.1.1. Konfiguration der Versuchsumgebung	12
4.1.2. Browserauswahl	12
4.1.3. Browsing Szenario	12
4.2. Acquisition Stage	13
4.3. Analysis Stage	13
4.3.1. Common Locations	13
4.3.2. Uncommon Locations	15
5. Ergebnisse	19
5.1. Firefox	19
5.2. Tor	30
5.3. Chrome	38
5.4. Brave	46
6. Vergleich der Browser	54
7. Diskussion	55
8. Fazit	57
Anhänge	58
A. Yara-Regeln	43
B. Ausführliche Analyse: Firefox	44
B.1. Common Locations	44
B.2. Uncommon Locations	49

C.	Ausführliche Analyse: Tor	53
C.1.	Common Locations	53
C.2.	Uncommon Locations	57
C.3.	Registry	58
D.	Ausführliche Analyse: Chrome	59
D.1.	Common Locations	59
E.	Ausführliche Analyse: Brave	65
E.1.	Common Locations	65
Literaturverzeichnis		66
Literatur		66

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 [13]: > Das Internet und seine Nutzer wachsen ständig, aber auch die Anzahl organisierter Verbrechen und illegale Aktivitäten nehmen zu.

“Webbrowser immer beliebter bla bla ...“ [12] > Webbrowser sind heutzutage ein wichtiges Werkzeug für Online-Aktivitäten wie Online-Banking, Online-Shopping und soziale Netzwerke.

Immer mehr Internet-Nutzer: [12] ■ Im Jahr 2019 gab es laut [13] fast 4,5 Milliarden Internetnutzer.

Zunehmende Bestrebungen nach Privatheit erschwert forensische Ermittlungen [16] > Zunehmende Verwendung von verschlüsselten Daten in der Dateispeicherung und Netzwerkcommunication 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 [9] ■ 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 Speicheraufzeichnungstools durchzuführen, um die Menge und Qualität der aus dem Speicherausgang 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: [15] ■ 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: [17] ■ 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 [22] ■ 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 [21] ■ 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

Zunächst werden die Begriffe *Privat Browsing*, das *Angreifermodell* sowie die *Artefakte des private Browsers* erläutert.

2.1. Private Browsing

Um den Begriff des private Browsing zu definieren, ist es zunächst wichtig, einen Browser und den „normalen“ Modus einzuführen.

Ein Web Browser, kurz *Browser*, ist eine Softwareanwendung zum Abrufen und Durchsuchen von Informationsquellen im Internet oder World Wide Web (WWW) [22]. Izuati und Ab Rahman [12] bezeichnen ihn als eine Software, die es Benutzern ermöglicht, das Internet über den von ihrem Dienstanbieter bereitgestellten Zugang zu nutzen. Sie werden für alltägliche Aktivitäten wie das Anschauen von Videos, das Durchsuchen von Websites, das Posten von Bildern oder Videos in sozialen Medien und das Herunterladen von Dateien genutzt. Die bekanntesten Webbrowser sind dabei Google Chrome, Mozilla Firefox, Microsoft Edge und Brave [12].

Beim „normalen“ Browsen speichert der Browser dabei alle Browsing-Aktivitäten wie Caches, Cookies, Suchbegriffe und URL-Verlauf auf dem Computer [12]. Um das zu verhindern wurde eine neue Funktion namens „Private Browsing“ in die Webbrowser mitaufgenommen, welche den Internetnutzern eine größere Kontrolle über ihre Privatsphäre und das Surfen ohne Rückstände von Datenspuren auf dem Computer ermöglicht [23]. Dabei unterscheidet man zwei wesentliche Ziele des privaten Browsers. Erstens sollen besuchte Webseiten keine Spuren auf dem lokalen Computer des Benutzers hinterlassen bzw. diese nach der Browsing-Session zu löschen, wie beispielsweise den Browsing-Verlauf. Zweitens soll die Anonymität des Benutzers vor einer Website gewährleistet werden, indem verhindert wird, dass Aktivitäten von Benutzern im privaten und im öffentlichen Modus verknüpft werden [1, 15]. Das Private-Browsing ist somit abzugrenzen von Anwendungen wie Tor, welche die Verfolgung und Überwachung aus der Ferne verhindern [10].

Der „Privater Browsing-Modus“ wurde erstmals 2005 mit Apple Safari 2.0 eingeführt [23]. Drei Jahre später folgte in Google Chrome der „Incognito-Modus“ und der „InPrivate Browsing Modus“ in Internet Explorer. Im Jahr 2009 führte Mozilla Firefox 3.5 seine Version des privaten Browsing-Modus ein [15].

Die wichtigsten Stakeholder, also Benutzer bzw. Interessensgruppen, des privaten Modus sind forensische Ermittler, Benutzer, Browser-Entwickler und Kriminelle. Forensische Ermittler versuchen dabei, Browsing Artefakte, also Rückstände durchgeführter Browsing-Sessions,

mit forensischen Tools und Techniken wiederherzustellen, um damit Kriminelle überführen zu können [15]. Kriminelle versuchen dabei gezielt, ihre Spuren von illegalen Aktivitäten mittels des privaten Modus zu verbergen [13]. Aus Nutzerperspektive ist es die weltweit beliebteste Form der Online-Datenschutzmaßnahme und geht direkt einher mit dem Löschen des Verlaufes [10]. Die Entwickler der Browser haben diesen Modus aufgrund des mangelnden Benutzerdatenschutzes eingeführt und wollen diesen damit garantieren [13].

2.2. Angreifermodell

Nachdem Private-Browsing jetzt eingeführt wurde, muss nun betrachtet werden, welche Arten es grundsätzlich gibt, um das Browsing-Szenario aufzuzeichnen bzw. zu rekonstruieren, falls die Durchführung bereits in der Vergangenheit liegt.

Eine Art des Angriffes ist der sogenannte „Local Attacker“. Dieser kann ein forensischer Prüfer, ein Familienmitglied oder Freund sein, welcher physischen Zugriff auf den Computer des Benutzer besitzt und dort beispielsweise versucht, auf den Browserverlauf zuzugreifen. Dies geschieht jedoch explizit, nachdem der Benutzer den privaten Modus verlässt. Erst danach erhält der Angreifer die vollständige Kontrolle über den Computer. Somit ist ein Zugriff auf die Maschine des Benutzers vor dem privaten Surfen ausgeschlossen, was die Sicherheit gegen einen lokalen Angreifer unmöglich macht, beispielsweise durch eine vorherige Installation eines Keyloggers [1].

Eine weitere Art des Angriffes auf das durchgeführte Browsing-Szenario ist der sogenannte „Web Attacker“. Dieser versucht Onlineaktivitäten des Benutzers im privaten Modus zu verfolgen und zu identifizieren. Dabei kann mittels Tracking-Tools oder durch das Sammeln von Informationen über die IP-Adresse des Benutzers versucht werden, diesen zu identifizieren. Dies kann zum Beispiel der ISP sein, welcher den Datenverkehr der Kunden verfolgt, um die Daten für Marketingzwecke zu monetarisieren, sofern sie anonymisiert und mit Zustimmung der Kunden erfolgt [1]. Ein Web-Attacker hat jedoch im Gegensatz zum lokalen Angreifer keinen tatsächlichen physischen Zugriff auf den Computer, von dem das Browsing durchgeführt wurde.

Nachdem bereits des öfteren über ein Browser-Szenario oder über Browsing-Aktivitäten gesprochen wurde, gilt es nun noch zu definieren, welche Überreste, auch Artefakte genannt, bei einem solchen entstehen können.

2.3. Browser Forensics und Artefakte

Bevor der Begriff der Browser-Forensik eingeführt wird, ist es zunächst wichtig, die digitale Forensik einzuführen.

Die digitale Forensik konzentriert sich auf die Wiederherstellung von Speichermedien, um digitale Beweise für Cybercrime-Untersuchungen zu sammeln. Diese müssen jedoch in ihrem

Originalzustand erhalten bleiben, um beispielsweise vor Gericht zulässig zu sein. Außerdem muss der Prozess der Erwerbung, Untersuchung, Analyse und Berichterstattung von digitalen Beweisen muss forensisch einwandfrei durchgeführt werden. Zusammenfassend hat die digitale Forensik also das Ziel, verwendbare Beweise für Computerkriminalität zu sammeln [12].

Die Browser-Forensik sammelt und identifiziert dabei Beweise und Informationen im Zusammenhang mit einem Verbrechen aus wiederhergestellten Browser-Sitzungen. Diese Art der Forensik wird für Ermittler immer wichtiger, da der Suchverlauf, die Download-Aktivität und Seitenaufrufe das Verständnis für das kriminelle Motiv verbessern können. Dabei werden diese Informationen bzw. Aktivitäten, welche auch als Browser-Artefakte bezeichnet werden, versucht zu finden oder wiederherzustellen, um jemanden als Verdächtigen aufgrund seiner Online-Aktivitäten überführen zu können [13].

Artefakte können neben dem angesprochenen Suchverlauf, der Download-Aktivität und Seitenaufrufe auch noch Cookies, Caches und anderen sensiblen Daten und temporäre Dateien darstellen [12].

Unterschieden kann bei der forensischen Untersuchung dann zwischen der Live und Dead Forensik. Bei der Live Forensik wird im Gegensatz zur traditionellen (dead) Forensik versucht, flüchtige Daten aufzubewahren (RAM, Caches) und Gegenmaßnahmen für verschlüsselte Dateien auf einem Live-System zu ergreifen. [7]. Eine große Herausforderung dabei ist das Kontaminieren von Beweismitteln, was während des Datenerfassungsprozesses geschieht [7], wie das Ausführen der Software zum Speichern des Arbeitsspeichers, welche selbst im RAM ausgeführt wird. Bei der Dead Forensik hingegen wird der Computer oder das Gerät, das untersucht werden soll, zuerst heruntergefahren, bevor das Speicherabbild erstellt wird bzw. die Datenextraktion beginnt [12].

Da nun die wichtigsten Begriffe erklärt wurden, folgt anschließend das Ziel dieser Arbeit.

3. Ziel der Arbeit

Wichtig: White-Box Ansatz gemäß local Attacker in [1] - Das Ziel des Angreifers besteht darin, für eine bestimmte Menge von HTTP-Anfragen, die er wählt, festzustellen, ob der Browser eine dieser Anfragen im privaten Browsing-Modus ausgeführt hat oder nicht. Wenn der lokale Angreifer dieses Ziel nicht erreichen kann, gilt die Implementierung des privaten Browsers als sicher. - Local Attacker weiß, wonach er sucht!

Forensiker müssen Funktionsweise von Private Browsing kennen [10] ■ Die Kenntnis der Erfolgsrate der PB-Technologie unterstützt die Strafverfolgungsbehörden bei digitalen Untersuchungen von Internetinhalten ■ Internetbeweise sind oft entscheidend für Untersuchungen ■ Bestimmung des Umfangs und des Erfolgs von PB-Technologie unterstützt die Strafverfolgungsbehörden bei digitalen Untersuchungen von Internetinhalten ■ Durch die Bestimmung des Umfangs und des Erfolgs von PB-Technologie können sie unnötige Datenverarbeitung und Zeitverschwendung vermeiden, die Untersuchungseffizienz verbessern und sicherstellen, dass keine wichtigen Inhalte übersehen werden. Daher können diese Punkte dazu beitragen, die Effektivität und Effizienz von Untersuchungen zu verbessern, insbesondere in Fällen, in denen Vor-Ort-Triage stattfindet oder in denen eine SHPO angeordnet wurde. Drei Punkte wichtig: ■ Wenn der Verdacht besteht, dass PB stattgefunden hat, hilft es zu wissen, wie erfolgreich die PB-Funktion eines bestimmten Browsers ist, um unnötige Datenverarbeitung (und Zeitverschwendung) zu vermeiden, wenn tatsächlich keine Browserdaten auf einem Gerät vorhanden sind. ■ Die Kenntnis darüber, wo PB möglicherweise Informationen zu Browsing-Sitzungen preisgibt, verbessert die Effizienz von Untersuchungen und verhindert, dass wichtige Inhalte übersehen werden. Dies ist besonders wichtig bei Vor-Ort-Triage, wie sie in einigen Fällen mit einer SHPO angeordnet wird.

Ziel der Arbeit: ===== - Welche Browsing Artefakte werden beim private Browsing auf einem Rechner hinterlassen, welche zeigen, dass eine Browsing Aktion vom Browser durchgeführt wurde? - Das heißt: o Es wird nach Browsing Artefakten gesucht, welche die Zuordnung „Durchgeführte Browsing Aktion“ <-> Browser ermöglichen o Vor, während und nach private Browsing Session nach Browsing Artefakten suchen, welche dem Browser zugeordnet werden können - Negativbeispiel: Suche in Hexdump nach im Browser gesuchtem String nicht als Beweis ausreichend, dass private Browsing Artefakte gefunden wurde. - Kategorisierung nach [17]: > Browsing History > Usernames/Email Accounts > Images

=> Thematisiert in [17]: o It appeared that the overall best way to recover residual data was to obtain the evidence from RAM or working memory, o Kritik: Oft nur String Match

in RAM-Hex als Nachweis für PB genannt -> ausreichend? (Evtl. Gegenexperiment mit Editor)

Warum muss String-Artefakt Browser zugeordnet werden können? [12] ■ Die Artefakte, die von den Browsing-Aktivitäten eines Kriminellen zurückgelassen wurden, können mit forensischen Tools extrahiert werden, um die Untersuchung des Ermittlers zu unterstützen. ■ Die erlangten Beweise müssen vor Gericht zugelassen werden, insbesondere digitale Beweise, da sie ohne ordnungsgemäße Verfahren leicht manipuliert werden können. ■ Es gibt bestimmte Merkmale von digitalen Beweisen, die Gerichte nach folgenden Kriterien akzeptieren: 1. Durchsuchungsbefehle - Beweise, die ohne Genehmigung erlangt wurden, können vor Gericht nicht anerkannt werden. 2. Berichte - Alle Prozesse, Werkzeuge, Methoden, Techniken, spezifischen Zeit- und Datumsangaben sowie die Beweiskette müssen formell dokumentiert werden, um die Authentizität der digitalen Beweise vor Gericht zu demonstrieren und zu unterstützen. 3. Beweisauthentifizierung - Der ursprünglich erhaltene Beweis sollte durch Vergleich der Hash-Werte mit dem Kopiebeweis übereinstimmen. Der erworbene Beweis muss unverändert bleiben, um die Gerichte mit genauen Informationen zu überzeugen. Gerichte akzeptieren Kopien von Beweisen, wenn der ursprüngliche Beweis verloren gegangen oder zerstört wurde, die Kopie jedoch noch intakt ist.

Ziele anderer Arbeiten: ===== > [12] - Die Art der extrahierbaren Daten zu untersuchen - den Unterschied zwischen privatem und normalem Surfen zu vergleichen - zu analysieren, welcher Browser die vollständigeren residualen Daten liefert. > [15] ■ ob bestimmte Arten von Browser-Daten gefunden werden konnten (Webseiten, Verlauf, Download-Verlauf, besuchte URLs und Suchbegriffe) > [22] ■ Das Ziel dieser Studie ist es, eine Rahmenbedingung für die Analysephasen des Webbrowsers im privaten Modus und Anti-Forensik vorzuschlagen, um eine effektive und effiziente forensische Untersuchung zu ermöglichen. ■ Die Studie nutzt Live-Forensik, um detailliertere Informationen über den Computer zu erhalten, während er noch in Betrieb ist, und eignet sich daher besser für die schnelle Datenerfassung in Echtzeit. > [24] ■ umfassende Analyse der privaten Browsing-Funktion in den vier beliebtesten Webbrowsers (IE, Firefox, Chrome und Safari) vorgestellt. > [12] - digitalen Forensikern helfen, Artefakte von Geräten zu verfolgen, die Live-Memory-Erfassung verwenden > [16] - Methodik entwerfen, um folgende Fragen zu beantworten: 1. Kann Tor den Benutzer schützen, indem es Beweise für dessen Nutzung aus dem RAM löscht, wenn die Browsing-Sitzung geschlossen wird? 2. Kann die Tor-Nutzung zu vier Schlüsselmomenten erkannt werden: während das Browser-Fenster geöffnet ist, nach Schließen des Browser-Fensters, nach dem Löschen des Installationsverzeichnisses/ zugehöriger Dateien und nach dem Ausloggen des Benutzers? 3. Können Dateien aus dem Browsing-Protokoll in der Live-Forensik mit Tor 7.5.2 wiederhergestellt werden, der zum Zeitpunkt der Schreibens aktuellsten Version? - Die Experimente wurden im mobilen Modus mit Tor wiederholt, d.h. von einem USB-Stick ausgeführt. (!!!) zu bestätigen, dass die Existenz und Nutzung des Tor-Browsers in Windows 10 nachweisbar ist. (!!!) nachweisen, dass Artefakte des Tor-Browsing-Protokolls auf dem Zielcomputer wiederhergestellt werden können. > [12] > In dieser Studie werden die residualen Daten zwischen Google Chrome und Mozilla Firefox Webbrowsers im normalen und privaten Browsermodus mithilfe eines forensischen Tools

analysiert und verglichen. > [15] ■ In dem Projekt wurden die Effektivität der "privaten" Modus von vier weit verbreiteten Webbrowsern analysiert. > [24] ■ Ziel: Bewertung der Sicherheit des privaten Surfens in den Browsern Chrome, Safari, Firefox und IE ■ Die Autoren haben eine umfassende forensische Analyse durchgeführt, die sowohl Live-Memory-Analyse als auch Post-Mortem-Analyse umfasste. > [15] ■ Vier getestet: Firefox, IE, Safari und Chrome

Keine Ziele der Arbeit: ===== - Private Browsing Indicators: Entering/Leaving Private Browsing [17] - Zeigen, dass ein Browser gestartet/geschlossen wurde - Zeigen, dass ein Browser im privaten Modus gestartet wurde - Zeigen, wann ein Browser gestartet/geschlossen wurde - Browser-Erweiterungen: [24] > Browser-Erweiterungen und ihre Auswirkungen auf das private Surfen wurden in einer Studie von Aggarwal et al. Im Jahr 2010 untersucht. -> Siehe Punkt „Add-Ons als Leck“ > Die Chrome-Erweiterung „Incognito Inspector“ kann im privaten Modus genutzt werden, um detaillierte Informationen über die Nutzeraktivitäten zu sammeln und in Echtzeit an einen Remote-Server zu senden. > Firefox-Erweiterungen sind standardmäßig im privaten Modus aktiviert und können genutzt werden, um Nutzeraktivitäten aufzuzeichnen. > Internet Explorer-Erweiterungen sind in der Regel deaktiviert und erfordern die manuelle Aktivierung im privaten Modus. Die von den Autoren entwickelte Erweiterung funktionierte jedoch nicht, da sie aufgrund eingeschränkter Privilegien nicht auf die BHO-Klasse zugreifen konnte - [1] > Unterschiedliche Handhabung durch Browser: Gefährliche Leckage für private Browsing Artefakte > Entwickler von Add-Ons haben möglicherweise den privaten Browsing-Modus bei der Entwicklung ihrer Software nicht berücksichtigt, und ihr Quellcode wird nicht derselben rigorosen Überprüfung unterzogen wie die Browser selbst. > Gegenmaßnahme: [1] ■ Es wurde eine Firefox-Erweiterung namens ExtensionBlocker entwickelt, um unsichere Erweiterungen im privaten Modus zu blockieren

4. Methodik

> Validation Stage (= Kapitel „Vergleich der Browser“)

Warum Methodik? > [1] Aufgrund der Komplexität moderner Browser ist eine systematische Methode erforderlich, um zu testen, ob der private Browsing-Modus ausreichend gegen die Bedrohungsmodelle aus Abschnitt 2 verteidigt. > [12] ■ Die Verfahren für die digitale Forensik für Browser-Forensik müssen angemessen befolgt werden, um dem Ermittler bei der Durchführung der Untersuchung zu helfen. Die Verfahren unterscheiden sich je nachdem, wie die Untersuchung durchgeführt werden soll. > [10] ■ Das Fehlen von Klarheit hat einen signifikanten Einfluss auf forensische Untersuchungen von Strafverfolgungsbehörden und deren Ansätze ■ Eine Kette von Beweisführung muss dokumentiert werden, um die Integrität und Zuverlässigkeit der Daten sicherzustellen. ■ Ein formaler forensischer Bericht wird dann vor Gericht präsentiert.

Bekanntes Computer Forensik Vorgehensmodell: [28]: Generic Model Computer Forensics Investigations (GCFIM) -> Daran orientieren sich alle in der Literatur

Phasen nach [15] ■ Die forensische Analyse erfolgt in zwei Phasen. 1. Zunächst wird die Analyse an sowohl "üblichen als auch ungewöhnlichen Speicherorten auf der Festplatte durchgeführt. 2. In der zweiten Phase wird der physische Arbeitsspeicher (RAM) untersucht.

Phasen nach [12]: ■ Es gibt verschiedene Modelle für digitale Forensik, die sich in ihren Phasen unterscheiden können. ■ Fünf Phasen sind besonders wichtig: Identifikation und Sammlung, Bewahrung, Erwerb, Analyse und Prüfung sowie Dokumentation. ■ In der Identifikations- und Sammelphase werden alle potenziellen Beweismittel identifiziert, gekennzeichnet und gesammelt, um sie in der nächsten Phase zu verwenden. ■ Beweismittel können z.B. Log-Dateien, temporäre Dateien, Netzwerkverbindungen, Browserverlauf und Cache sein. > Phasen: Preparation Phase o Versuchsplanung + Konfiguration der HW/SW + Durchführen des Experiments Acquisition Stage o Abbildung von der Festplatte (Static Forensics) und des RAMs (Live Forensics) Analysephase o Bilder der Speicherabbilder mit einem forensischen Tool untersuchen Validierungsphase o gefundenen Artefakte verglichen und dokumentiert

4.1. Preparation Stage

4.1.1. Konfiguration der Versuchsumgebung

VM Konfiguration

RAM: - Kaum Angaben in der Literatur: > [22]: 2 GB > [17]: 4 GB - Hier: 6 GB -> Ausblick: Kritik an Literatur, dass RAM-Größe kaum thematisiert wird, obwohl sie Auswirkungen auf Ergebnisse hat -> Siehe Kapitel X (TODO!)

Netzwerkeinstellungen: - TODO

Windows 10 Installation: - TODO

Tools auf VM: - Process Monitor - Regshot

Konfiguration des Analyse-Rechners

Volatility: Plugins-Liste: (Ähnlich zu [Hariharan] und [5]) - TODO

Autopsy: Evtl. hier Sleuthkit vs Autopsy thematisieren - TODO

Sonstige Tools: WinHex SQLite Viewer etc. - TODO

-> Evtl. am Schluss Tabelle mit allen Tool, Versionen und Plug-Ins

4.1.2. Browserauswahl

> Browserstudie [12] - Die Herstellerangaben unterschiedlicher Browser bzgl. Privatheit untersucht - Firefox 58.02: No Browsing History stored, No Cookies stored, No login Info stored, Tracking Protection Enabled: Disconnect, Download Files not Hidden - Chrome 63.0.3239: No Browsing History stored, No Cookies stored, No login Info stored, Tracking Protection Enabled: No, Download Files not Hidden

> design aim of Tor: [16] - preventing from writing to disk (Perry et al., 2018) - enabling secure deletion of the browser (Sandvik, 2013) (hier nicht relevant)

4.1.3. Browsing Szenario

- Wichtig für White-Box-Ansatz: Browsing Szenario ist bekannt - URL X ... (TODO!)

4.2. Acquisition Stage

> Browsing Szenario durchführen > Zeitpunkte von -> Orientieren an: [16] - RAM-Dumps - VM-Snapshots (nur letzter Snapshot ist Post-Mortem Forensik) - Process Monitor Logfiles - Registry Snapshots

- Warum Process Monitor während Browsing? o Während Browsing Szenario Filechanges untersuchen: DaemonFS set to monitor all activity within local hard drive[17]

- Registry: [22] ■ Das Windows-Registrierungsverzeichnis enthält viele Informationen zur Nutzung des Computers, Benutzerkonfigurationen, Anwendungen und Hardwaregeräte ■ Informationen im Registrierungsverzeichnis werden nach Ausführungsreihenfolge, Suchschlüsselwörtern, zuletzt aufgerufenen Ordnern, Anwendungsprotokollen und anderen Kategorien sortiert.

4.3. Analysis Stage

> Analysis Stage (= Kapitel „Results“) - Analyse der akquirierten Artefakte der vorherigen Phase: VM-Snapshots, RAM-Dumps, Process Monitor Logfiles und Registry Snapshots mit ggf. zusätzlichen Tools - Oberster Leitsatz dabei: gefundenes Artefakt muss eindeutig Browser zugeordnet werden können: Deshalb einfache Stringsuche in RAM mit WinHex ungenügend -> Hier evtl. negatives Beispiel zu Stringsuche einflechten

4.3.1. Common Locations

Whitebox-Analyse: (gezieltes Suchen nach Dateien) [2] Definition: "White-Box"Computer Forensik bezieht sich auf eine forensische Untersuchungsmethode, bei der der forensische Analyst über umfassende Kenntnisse und Zugriff auf das untersuchte System verfügt. Im Kontext der Computerforensik bezieht sich "White-Box"darauf, dass der Analyst über volle Transparenz und Zugriff auf alle Informationen, Ressourcen und Artefakte des Systems verfügt.

Die "White-Box"Forensik kann verschiedene Techniken und Tools umfassen (z.B. Process Monitor, Regshot, Registry Explorer, Dekomprimierungstools), um Daten wiederherzustellen, gelöschte Informationen wiederherzustellen, Metadaten zu analysieren, Netzwerkaktivitäten zu überwachen und weitere forensische Analysen durchzuführen. Der Fokus liegt darauf, das System vollständig zu verstehen und alle relevanten Beweise zu sammeln.

Hier: In Orten gesucht, die 1. Process Monitor ermittelt hat und 2. in der Literatur vorgeschlagen wurden.

Definition: Common Location (= i.d.R. Installationsverzeichnisse der Browser) = „Bekannte Speicherorte“, z.B. bei Firefox - TODO: Quelle > Welche Dateien in Common Locations mit

Process Monitor identifiziert > Wie haben sich Dateien verändert in verschiedenen Snapshots?
 > Was in Dateien gefunden? - Ziel: Befinden sich unter den Dateien, die ein Browser direkt auf die Festplatte schreibt private Browsing Artefakte? - Dateien sind Browserspezifisch, befinden sich in bekannten Pfaden. Beispiele: Datenbank-Dateien, Caches, temporäre Dateien.
 - String-Suche wäre nicht ausreichend, da Artefakte teilweise komprimiert (siehe .jsonlz4) -
 Beispiele: > Cache folder, Web history [15]

Schreiboperationen mit Process Monitor verfolgen

- Process Monitor: WriteFile Operationen von Browser - Vorgehen: (Siehe Aktivitätsdiagramm)
 o Basis = Process Monitor Logfile 1 und 2 o Processmonitor Filter: Browser-Prozess, Dateiope-
 rationen, nur WriteFile o Export als CSV o Datenaufbereitung in Excel o Irrelevante Spalten
 löschen: Time of Day (zeitl. Kontext nicht wichtig), Process Name (Da in Process Monitor
 bereits nach Namen gefiltert wurde -> Alle Prozesse haben gleichen Namen), Operation (Da
 in Process Monitor bereits nach Operation gefiltert wurde -> Alle Prozesse haben gleiche
 Operation „WriteFile“), Result, Detail o Gleiche Operationen (Duplikate) löschen o Neue
 Spalte mit Dateiendung -> Weiteres gruppieren, sortieren und analysieren ist browserspezi-
 fisch o Wenn Daten aufbereitet wurden: 1. Autopsy: Prüfen, ob Dateien noch in Snapshot
 Image vorhanden 2. Wenn ja, Dateien mit Autopsy extrahieren 3. Wenn nein, prüfen, ob
 Datei in RAM gecacht -> Hier beschreiben, wie mit Volatility filelist etc. Dateien aus RAM
 wiederhergestellt werden können 4. Prüfen ob Browsing Artefakte in Dateien enthalten sind:
 Stringsuche nach Aktionen des Browsing-Protokolls

TODO: Allgemein: Nur Dateien untersucht, die gemäß Methodik (Kapitel X) entweder im
 Snapshot vorhanden sind oder sich über Autopsy Carving PlugIn bzw. RAM wiederherstellen
 lassen. > Wenn Temp-Dateien nicht mehr vorhanden, wird die nicht-Temp Datei aufgeführt

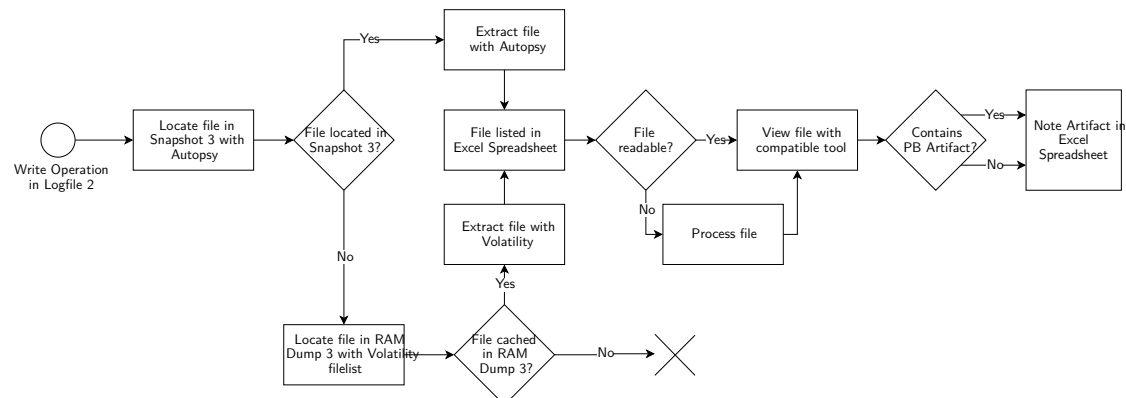


Abbildung 4.1.: TODO: Process Monitor Write Operation to Excel Spreadsheet

SQLite-Datenbanken

- Gesondert betrachtet: Zeitlicher Vergleich von SQLite Datenbanken > Begründung: In Literatur ermittelt, dass SQLite DB von zentraler Bedeutung bei Browser History -> Hier wird i.d.R. Suchverlauf gespeichert > Zählt zu den wichtigsten "Common Locations" > Vorgehen: Siehe Aktivitätsdiagramm TODO: WAL Checkpoint

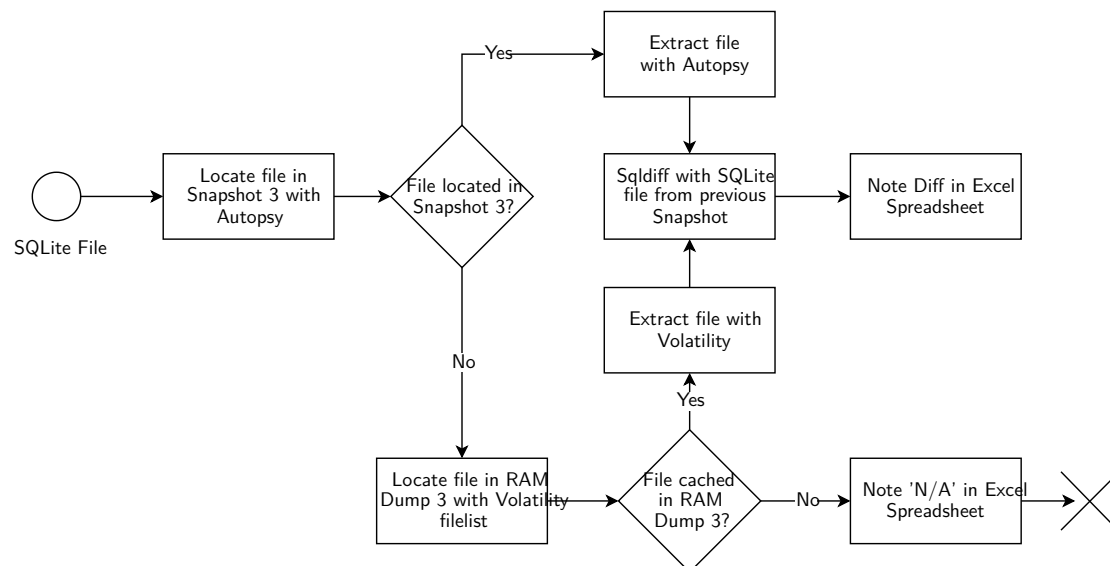


Abbildung 4.2.: TODO: Process Monitor Write Operation to Excel Spreadsheet

Registry

- Registry: > Process Monitor: SetValue Operationen von Browser -> Values der Keys untersucht (je nach Datentyp) -> Sonderfall: REG_BIN - Kategorien der Keys auflisten Diagramm: z.B. Kreisdiagramm mit Anteil der Kategorien an gesamten Schreiboperationen > Stringsuche in Registry Hives mit Registry Explorer (Siehe Liste) - Suchbegriffe auflisten
- Hives (Speicherorte) auflisten > Bshellactivities-ähnliche Keys untersucht - Arbeit von Bshellactivities-ähnliche Keys erklären

4.3.2. Uncommon Locations

Blackbox-Analyse: [2] (Stringsuchen im gesamten Image mithilfe von Tool) Definition: Auch "triage-style keyword search"[10] genannt, = Durchsuchung des Beweismaterials ohne Vorwissen über Browserverhalten (d.h. welche Dateien geschrieben wurden) sowie ohne Vorverarbeitung der Dateien (z.B. Entpacken von Dateien). Stattdessen: Untersuchen der Images nur

mittels vordefinierter Funktionen von Forensik-Tools "Triage", da dies schnelles erstes Mittel von Forensikern, um nach Acquisition Phase Ergebnisse zu erhalten

Hier entscheidend "Uncommon Locations": = „Unbekannte Speicherorte“, nur durch tiefgehende forensische Analyse entdeckt

- TODO: Quelle o Registry o Pagefile.sys o Unallocated Disk Space -> Suche nach „obfs4“ deckt Bridging IP-Adressen auf o Windows-Prefetching o Timestamps o \$MFT o \$Unalloc o \$LogFile o Favicons o etilqs o Manifest.json o slack space

- Beispiele in der Literatur: > "\$MFT", "\$LogFile", "Favicons", "etilqs", "Manifest.json", "pagefile.sys.", "unallocated space" and "slack space" [15]

- Ziel: Untypische Orte, wo private Browsing Artefakte gefunden werden können. Im Unterschied zu Common Locations: Weitergreifendes Konzept, umfasst Dateien, die nicht von Browsern in bekannten Browser-Ordnern gespeichert werden, sondern auch Speicherorte wie RAM, Registry oder Caches des Rechners, wie - In Literatur ermittelt: für private Browsing drei uncommon Locations relevant: o Stichwortsuchen in kompletten Speicherabbildern: Festplatte (Common Location Browser-Pfade ausgenommen) + RAM -> Wichtig: String-Treffer muss Browser zugeordnet werden können -> Negativbeispiele: o [22]: in WinHex: URLs, Passwörter gefunden -> Wie wird URL Browser zugeordnet? Reicht gefundener String in RAM-Hex als Beweis aus? o [14] WinHex: email account can be retrieved, retrieves all URL histories including the directories visited by a user o [15] Firefox: URLs und Keywords als Strings in WinHex gesucht und gefunden o [15] Chrome: URLs und Keywords als Strings in WinHex gesucht und gefunden

o In Literatur oft verwendet: Stichwortsuchen: > Autopsy Keyword-Suche außerhalb der Common Locations, in allen Partitionen ■ Definition der gesuchten Strings ■ Weiterführend: In Literatur nichts über verwendete Plugins gefunden. Hier: o Automatische Kategorisierung von Dateien o Timeliner-Plugin (Wenn verwendbar?) > RAM: Yarascan Treffer -> String Kontext ■ Definierte Yarasrules - TODO! ■ HTML-Fragmente: [23] We were also able to find blocks of HTML code that constructs Web sites we visited. ■ Image Carving: > Carved from Memdump [17] > Bildsuche mit: Griffey's DI Analyze Pro with LACE plug-in [10]

- Windows: Prozess-Struktur im RAM: (-> TODO: Wo gefunden?) The EPROCESS data structure contains information about process instances, such as image name and ProcessID, the resources allocated in terms of memory allocations (how much and where), types (private, mapped, shareable, etc.), memory protections (combinations of read, write, execute, and reserved), modules loaded, and pointers to ETHREADs and the process environment block.

Both EPROCESS and ETHREAD are considered opaque objects by Microsoft [28], inhibiting analysis; fortunately, third-party work has been done to understand these structures [29], [30]. Microsoft does provide symbol files¹, which help communicate the layout of data structures [31]. Indeed, Volatility uses these symbols for its own processing.

Included in EPROCESS, the ETHREAD object is an opaque structure which contains useful information about the stack. We calculated the size of a stack from the difference between its limit and base, both of which are attached to the ETHREAD.

Another member of the EPROCESS structure, the VAD tree, maps out the virtually allocated memory for a process [32]. VAD nodes refer to loaded modules (in the allocations in which they were referenced) and also have unique permission flags per node.

The PEB (process environment block) contains data about the number of heaps, which modules have been loaded into memory, and the command-line string that invoked the process [33]. The module list may not match the VAD tree's list exactly, the difference of these two sets indicating images of interest

In Literatur der Web Browser Forensik vorwiegend verwendet: - Autopsy Stichwort Suche nach PB Artefakten + Indizieren der Dateien durch Autopsy-PlugIns - Stichwortsuche in RAM mit Volatility Yarascan PlugIn. Vertiefende Untersuchung für jeden Yara-Rule-Treffer *** Hier werden Artefakte gefunden *** -> Flussdiagramm

Analyse mit Autopsy

Bei White-Box Analyse: Autopsy nur zur Dateiextraktion genutzt, hier: als konkretes forensisches Werkzeug Wichtig dabei: - Stichwortsuche -> Screenshot von Suchfunktion -> Suchbegriffe auflisten - Nutzen der Plug-Ins - evtl. "pagefile.sysProblematik ansprechen

Analyse mit Volatility

Bei White-Box Analyse: RAM nur zur Dateiextraktion genutzt, hier: als konkretes forensisches Werkzeug

Wichtig: Auf Ziel der Arbeit verweisen: gefundenes PB Artefakt muss zwingend Browser zugeordnet werden können -> d.h. gefundener String des Browsing Protocols in Hex des RAM-Dumps reicht nicht als Beweis für gefundenes PB Artefakt aus. Stattdessen: gefundenes PB Artefakt im RAM muss Browser zugeordnet werden können -> Passendes Werkzeug = Volatility PlugIn "Yarascan" TODO: Definition Yarascan -> TODO: Auflistung der Yara-Rules -> Vorgehen: Siehe Baumdiagramm

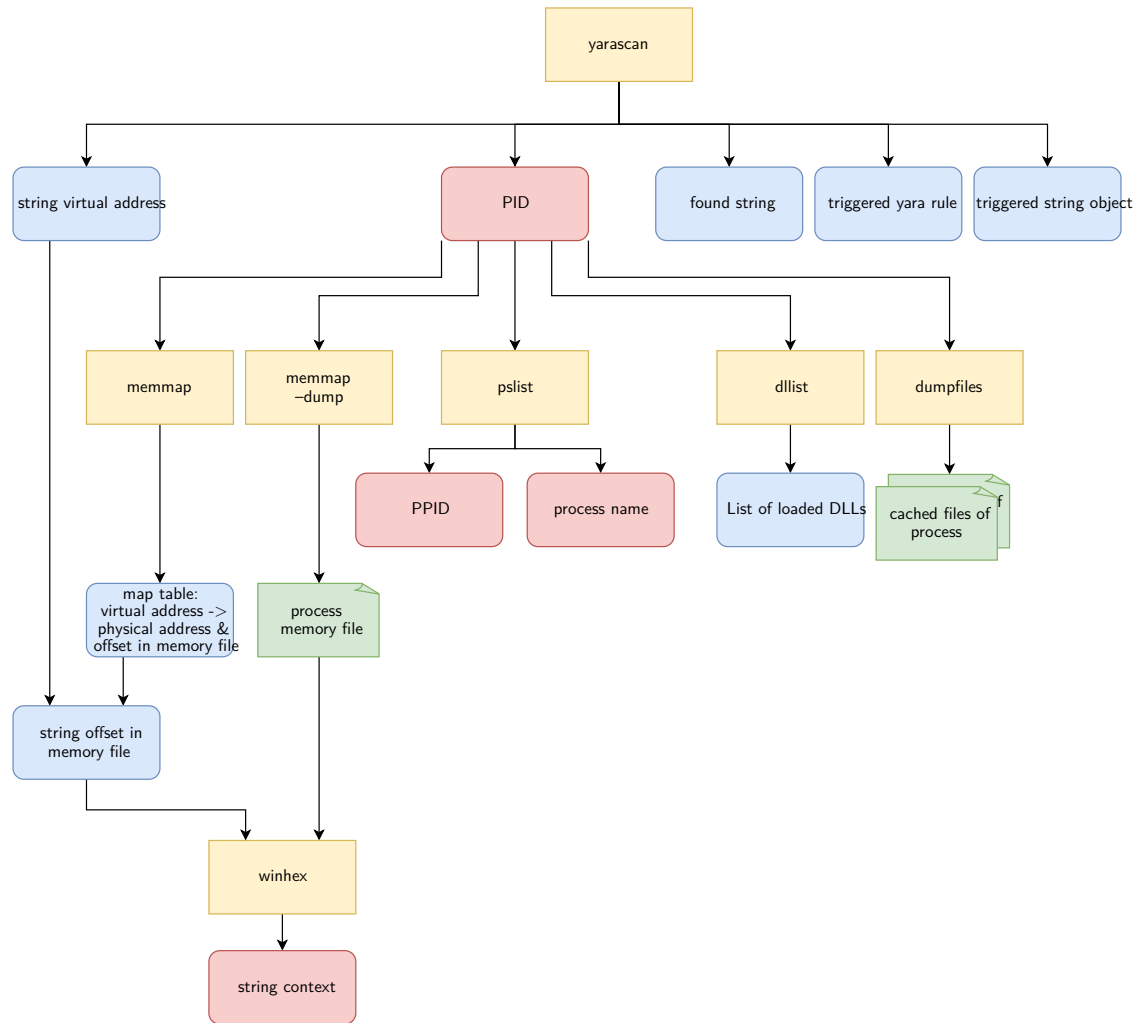


Abbildung 4.3.: TODO: Process Monitor Write Operation to Excel Spreadsheet

5. Ergebnisse

= LÄNGSTES/AUSFÜHRLICHSTES KAPITEL!!!

Für jedes Unterkapitel gilt: > Erst allgemeines Vorgehen/Methodik definieren > Danach spezifisch für jeden Browser: Unterschied zwischen Snapshot-Zeitpunkten, insb. zwischen Live- und Dead-Forensik

5.1. Firefox

White-Box Analyse/Common Locations

Schreiboperationen mit Process Monitor verfolgen:

Im Anhang: Tabelle mit allen geschriebenen Dateien (markiert, wenn nicht mehr wiederherstellbar + markiert, wenn Datei "verändert"(siehe oben: temp, WAL))

Aux-Dateien, welche nicht mehr vorhanden waren, aber dafür "richtige"Dateien: - - - - -

Ergebnis: Tabelle mit wiederherstellbaren Dateien: Logfile 1 vs. Logfile 2 + Tool mit dem Datei untersucht wurde - Dateien, die in beiden Logfiles nicht wiederherstellbar

		Logfile 1	Logfile 2
Cache	cache2\entries\037778A55E1B7E9BED3390289866D09402D6C913	Keine PB Artefakte	Keine Schreiboperation
	cache2\entries\1223A0378B8971FA4CD25EA1731C80B2B1676B42	Keine PB Artefakte	Keine Schreiboperation
	cache2\entries\250EE2BC03AFF526F1A1C3DB212A79DE3E860D5E	Keine PB Artefakte	Keine Schreiboperation
	jumpListCache\ZKJGVJPzPe7w4wOKwEY0Jw==.ico	Keine PB Artefakte	Keine Schreiboperation
	cache2\index.log	Keine Schreiboperation	Keine PB Artefakte
	cache2\index	Keine Schreiboperation	Keine PB Artefakte
Datareporting	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
	*datareporting\archived\2023-05\1683405837939.abf8b065-41a4-4e94-a044-1cead61e396a.main.jsonlz4	Keine Schreiboperation	Keine PB Artefakte
SQLite	storage\permanent\chrome\idb\3870112724rsegmnolttet-es.sqlite	Keine PB Artefakte	Keine Schreiboperation
	storage\permanent\chrome\idb\1657114595AmcateivrtiSty.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	*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 5.1.: Tabelle mit wiederherstellbaren Dateien: Logfile 1 vs. Logfile 2

Allgemein: Artefakte in zwei "Common Pfaden (Local) - (Roaming)

Kategorien der Logs: - Cache: > > > Zweck: "Firefox verwendet den Cache, um Webseiten und Ressourcen wie Bilder, Stylesheets, Skripte und andere Dateien temporär auf dem lokalen Computer zu speichern. Dadurch können wiederholte Anfragen an den Server vermieden und die Ladezeiten verringert werden, da der Browser die Inhalte aus dem Cache abrufen kann, anstatt sie erneut herunterzuladen. Die tatsächlichen Inhalte dieser Datei sind binär und können je nach Art der Ressource variieren, beispielsweise HTML, Bild- oder Audiodateien. Analyse:

- Tool: MozillaCacheView - TODO: Screenshot > Zweck: "Die Indexdatei im Cache dient als Datenbank, die Informationen über die gespeicherten Dateien enthält. Sie ermöglicht dem Firefox-Browser, schnell auf die zwischengespeicherten Ressourcen zuzugreifen und diese effizient zu verwalten. Analyse: - Tool: siehe Github und HxD > - Tool: Windows Foto App - Enthält kleines "mIcon

- datareporting: Allgemein: "Dateien im Ordner /datareporting/glean/db sind Teil des Glean-Systems, das von Mozilla (dem Entwickler von Firefox) für die Sammlung von Telemetriedaten verwendet wird. Telemetrie-Daten sind anonyme Informationen über die Nutzung des Browsers, die zur Verbesserung der Software und zur Behebung von Problemen verwendet werden können. > Zweck: "Die "data.safe.bin" Datei enthält verschlüsselte Telemetrie-Daten, um ihre Integrität und Sicherheit zu gewährleisten. Analyse: - Tool: HxD - keine PB Artefakte > Zweck: "Diese Dateien speichern Informationen über das Firefox-Profil, das von Glean verwendet wird, um Telemetriedaten zu sammeln. Analyse: - Mit firefox proprietärem jsonlz4 Algorithmus verschlüsselt - können mit speziellem Tool "dejsonlz4" dekomprimiert werden (Quelle Github) - Dateien enthalten Systeminformationen im Json-Format (Screenshot?)

- Sessionstore-Backup: > Zweck: "Die Datei "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. Analyse: - jsonlz4 Datei in sessionstore-backup lassen sich mit Online-Tool parsen (<https://www.jeffersonscher.com/ffu/scrounger.html>) - Ergebnisse: Tab 1: Willkommen bei Firefox [6.5.2023, 22:25:06, about:welcome; Tab 2: Firefox Datenschutzhinweis — Mozilla [6.5.2023, 22:24:59], - Sind Seiten, die sich automatisch geöffnet haben, nachdem Firefox zum ersten Mal geöffnet wurde - keine PB Artefakte > Zweck: "Die Datei sessionstore.jsonlz4 speichert den aktuellen Zustand der Firefox-Sitzung. Diese Datei wird während der Browsersitzung regelmäßig aktualisiert, um sicherzustellen, dass Änderungen im Zustand der Sitzung erfasst werden. Analyse: - Lässt sich nicht mit Online-Tool aus Logfile 1 parsen - Stattdessen: dejsonlz4, danach Notepad++ mit JSON Plugin - kaum Einträge zu Sitzung, hauptsächlich CSS Daten zu Fenstergröße- und position, insb. keine PB Artefakte - Interessant: image-Eintrag als base64 entdeckt, in PNG umgewandelt (<https://base64.guru/converter/decode/image>), mit Windows Foto-App: "mIcon (Mozilla-Logo)

- Sonstige Dateien: > Zweck: "Die Datei "prefs-1.js" enthält benutzerspezifische Einstellungen und Konfigurationen für den Firefox-Browser. Sie speichert die Präferenzen des Benutzers in Form von JavaScript-Objekten.

In dieser Datei werden verschiedene Arten von Einstellungen gespeichert, darunter:

Allgemeine Einstellungen: Dies umfasst Optionen wie die Standardsuchmaschine, die Startseite, den Zoomfaktor, die Spracheinstellungen und andere globale Einstellungen, die das Verhalten des Browsers beeinflussen.

Datenschutzeinstellungen: Hier werden Präferenzen bezüglich Cookies, Verlauf, Passwortverwaltung, Standortfreigabe und Tracking-Schutz gespeichert. Diese Einstellungen kontrollieren, wie der Browser mit persönlichen Daten und der Privatsphäre umgeht.

Add-On-Einstellungen: Wenn der Benutzer Erweiterungen oder Add-Ons installiert hat, können in dieser Datei die spezifischen Einstellungen und Konfigurationen für jedes Add-On gespeichert werden."

> Zweck: "Die Datei "xulstore.json" speichert benutzerspezifische Anpassungen und Konfigurationen für den Firefox-Browser. Analyse: - Weder JSON-Dateien (Notepad++) noch .tmp Dateien (HxD) enthalten PB Artefakte

- SQLite: (TODO: Abgleich mit Diffs-Exeltabelle, ob wirklich nur in places.sqlite geschrieben wurde) Dateien haben Sonderstellung: - Diese DBs dienen zur Verwaltung und Speicherung sämtlicher Browser Artefakte, insb. der Browser Historie - Aus diesem Grund: Dateien intensiver betrachtet Siehe Kapitel Methodik: > Entwicklung von Dateiinhalten in allen Snapshots (1, 2, 3 und 4) betrachtet > Für jeden Snapshot: - SQLite-Datei extrahiert und mit SQLite-Datei aus vorherigem Snapshot verglichen - Untersuchung der SQLite-Dateien mit SQLite-Viewer (GUI-Tools) - Wenn zu SQLite-Datei WAL-Datei existiert: mit sqlite3 Kommandozeilentool PRAGMA wal_checkpoint durchgeführt, danach neue SQLite-Datei mit ursprünglicher SQLite-Datei verglichen > Dabei gibt es drei Zustände: - leere Datei - neuer (nicht-leerer) Inhalt - gleichbleibender Inhalt Mit Process Monitor Logfiles festgestellt, dass in folgende SQLite-DBs geschrieben: - places.sqlite "Diese Datenbank enthält Informationen über die Lesezeichen, den Verlauf und die Tags im Firefox-Browser. Sie speichert die URLs der besuchten Websites, die Zeitstempel der Besuche, die Titel der Seiten und andere relevante Daten. cookies.sqlite Speichert Webseiten-Cookies In dieser Datenbank werden die Cookies gespeichert, die von Websites im Firefox-Browser verwendet werden. Cookies sind kleine Textdateien, die von Websites auf dem Computer des Benutzers abgelegt werden und verschiedene Informationen speichern können, z. B. Anmeldeinformationen, Sitzungsdaten oder Präferenzen. storage.sqlite Speicher für Webseiten "Diese Datenbank wird von Firefox verwendet, um verschiedene Arten von Webdaten zu speichern, wie z. B. die IndexedDB-Datenbanken von Websites, Offline-Cache-Daten, Webseiten-Skriptdaten und andere lokale Speicherinformationen. favicons.sqlite Speichert Icons für Lesezeichen "Diese Datenbank speichert die Favicons, also die kleinen Symbole, die in der Adressleiste und bei den Lesezeichen angezeigt werden, um Websites visuell zu identifizieren. Sie enthält die gespeicherten Favicons für die besuchten Websites. webappsstore.sqlite Speicher für Webseiten "Diese Datenbank speichert Informationen über installierte Webanwendungen im Firefox-Browser. Sie enthält Daten wie Berechtigungen, Einstellungen und andere spezifische Informationen für Webanwendungen. formhistory.sqlite In dieser Datenbank werden Informationen aus Webformularen gespeichert, die der Benutzer in Firefox ausfüllt. Sie enthält die eingegebenen Daten wie Name, E-Mail-Adresse, Adresse und andere Formulardaten, um das automatische Ausfüllen

von Formularen zu ermöglichen. 1657114595AmcateirvtiSty.sqlite Activity Stream for Firefox is a collection of all the things you do in the browser that you care about displayed in a rich and meaningful way 3870112724rsegmnoittet-es.sqlite "Remote Settings in Firefox sind eine Funktion, die es ermöglicht, Browser-Einstellungen zentral zu verwalten und an die Benutzer zu übertragen, ohne ein vollständiges Browser-Update durchführen zu müssen. SSQlite DB ist Datenspeicher dazu

Ergebnisse: > Nach Browser-Installation noch keine SQLite-Datei angelegt (Snapshot 1) >

	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, Zeilen, 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	Leer initialisiert (Nur Spaltennamen)	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, 1 Zeile: "origin: chrome"	no diff	Einträge (Binardaten) eingefügt, keine PB Artefakte (HxD)	no diff	no diff	no diff
	3870112724rsegmnoittet-es.sqlite	N/A	Initialisiert, 1 Zeile: "origin: chrome"	no diff	no diff	no diff	no diff	no diff
			Leer					
			Unverändert (nicht-leer)					
			Neuer (nicht-leerer) Inhalt					

Abbildung 5.2.: Comparison of found PB artifacts between RAM Dumps

Während Browsing Szenario alle DBs Initialisiert, außer "webappsstore.sqlite"(Snapshot 2) - Dabei wurden in places.sqlite die Seiten geschrieben, die sich automatisch nach Browserstart im public Modus geöffnet haben (Datenschutzhinweise zu Firefox) - Restliche Dateien ohne Inhalt, nur Spaltennamen - Nach WAL Checkpoints bleiben Dateien unverändert > Nach Schließen des Browsers (Snapshot 3) - in places.sqlite: Indizes bei eingetragenen Seiten aktualisiert - 1657114595AmcateirvtiSty.sqlite erhielt BLOB Eintrag, in HxD keine Muster erkennbar - webappsstore.sqlite: leer initialisiert, nur Spaltennamen - restliche Dateien unverändert - nach WAL Checkpoints bleiben Dateien unverändert > Nach herunterfahren der VM (Snapshot 4) - Alle Dateien unverändert, auch nach WAL Checkpoint

- Zusammenfassung: in keiner Datei PB Artefakte

Quantitativ: (Diagramme) > Balkendiagramm: Für jede Logfilekategorie: Anzahl Schreiboperationen Logfile 1 vs Logfile 2

Registry

> Process Monitor: SetValue Operationen von Browser TODO: Logfile 1 vs 2? Kategorien Registry Keys: 1) PreXULSkeletonUISettings: > Prefix: Absoluter Installationspfad von Firefox > Skeleton UI Einstellungen von Firefox Definition: > Der "PreXULSkeletonUISettings"Registry Key enthielt Einstellungen für die Benutzeroberfläche (UI) des Firefox-Browsers, insbesondere für das sogenannte SSkeleton UI". Das Skeleton UI ist eine vereinfachte Benutzeroberfläche, die während des Ladens des Browsers angezeigt wird, bevor die vollständige Benutzeroberfläche geladen ist. Es besteht aus grundlegenden Steuerelementen und Elementen, die dem Benutzer die Interaktion ermöglichen, während der Rest der Benutzeroberfläche noch geladen wird. > Der "PreXULSkeletonUISettingsSchlüssel enthielt Konfigurationsoptionen wie Farben, Positionen und andere Einstellungen für das Skeleton UI. Durch das Bearbeiten dieses

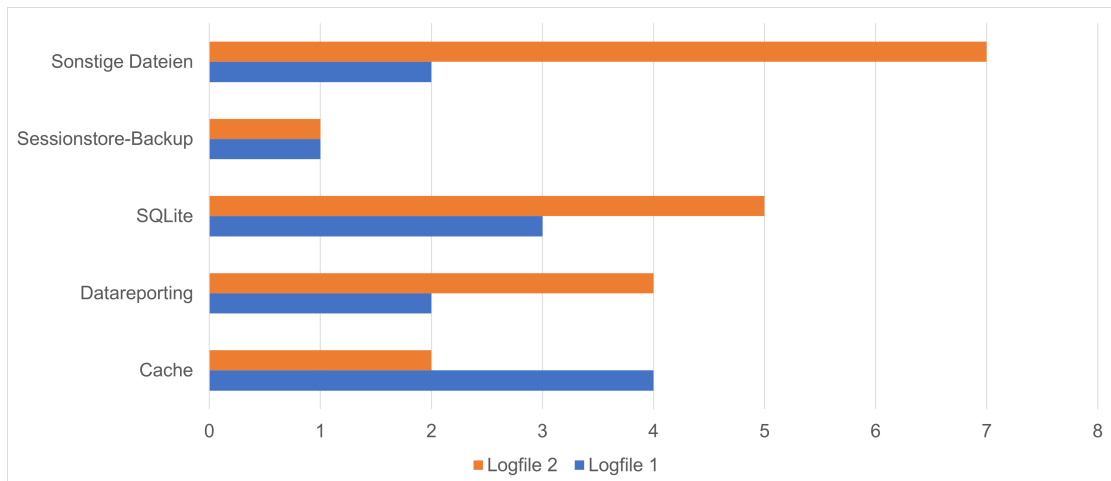


Abbildung 5.3.: Comparison of found PB artifacts between RAM Dumps

Schlüssels konnten Benutzer die Darstellung des Skeleton UI anpassen. Es ist jedoch wichtig zu beachten, dass das Ändern der Registrierungseinträge ein fortgeschrittenes Verfahren ist und Fehler zu Problemen mit dem Browser führen kann.

> Struktur der Keys: > Unterschiedliche UI Einstellungen - - - - - > keine PB Artefakte unter UI Einstellungen 2) Business Activity Monitoring > Quelle: > BAM is a mostly undocumented feature that controls the programs executed in the background. DAM is a feature for devices supporting the "Connected Standby" mode (i.e. when a device is turned on, but its display will be turned off). As a result, the BAM registry keys will contain data on any devices, while DAM registry keys will only contain data on mobile devices. > The BAM registry key contains multiple subkeys under bam

State

UserSettings, with one subkey per user, identified with the user SID. While the key is in the SYSTEM registry hive, program executions can thus still be tied to a specific user using this SID. > Each user-specific key contains a list of executed programs, with their full path and timestamp of last execution. > If a file is deleted, the eventual associated entry in the BAM is deleted as well after the system reboot. Additionally, BAM entries older than 7 days are deleted upon system boot. The BAM thus provides limited information on historic execution of programs > No entries are created in the BAM keys for executables on removable media and/or on network shares. > Key:

Quantitativ: (Diagramme) - Stacked Balkendiagramm jeweils für Logfile 1 und Logfile2: Anteil Kategorie 1 bzw. 2 an allen Registry-Schreiboperationen

> Stringsuche in Registry Hives mit Registry Explorer (Siehe Liste) In allen Hives kein Treffer für alle Suchbegriffe

Literatur: > angeblich in Shellactivities Ergebnisse. -> Nicht mehr vorhanden in aktueller Version (Verweis auf E-Mail)

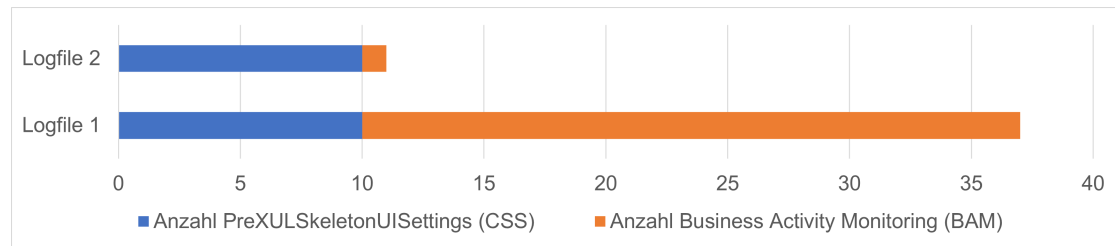


Abbildung 5.4.: Comparison of found PB artifacts between RAM Dumps

Black-Box Analyse/Uncommon Locations

Analyse mit Autopsy

Bei White-Box Analyse/Common Locations: Autopsy nur zur Dateieextraktion genutzt, hier: als konkretes forensisches Werkzeug

Stichwortsuche: - In allen Snapshots keine Treffer (auch innerhalb \$Carved) - TODO: Pagefile gefunden?

Von Autopsy automatisch indizierte Dateien: In allen Fällen: keine Dateien gelöscht, nur über Zeitraum der Snapshots neue dazugekommen - Web Bookmarks: Snapshot 1: > Bing.url

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
places.sqlite		6		https://support.mozilla.org/products/firefox	Hilfe erhalten	2023-05-06 22:25:00 MESZ	Firefox Analyzer	mozilla.org	CFV_Firefox_Klon_Snapshot_3.img
places.sqlite		6		https://support.mozilla.org/kb/customize-firefox-controls-b...	Firefox anpassen	2023-05-06 22:25:00 MESZ	Firefox Analyzer	mozilla.org	CFV_Firefox_Klon_Snapshot_3.img
places.sqlite		6		https://www.mozilla.org/contribute/	Mitmachen	2023-05-06 22:25:00 MESZ	Firefox Analyzer	mozilla.org	CFV_Firefox_Klon_Snapshot_3.img
places.sqlite		6		https://www.mozilla.org/about/	Über uns	2023-05-06 22:25:00 MESZ	Firefox Analyzer	mozilla.org	CFV_Firefox_Klon_Snapshot_3.img
places.sqlite		6		https://www.mozilla.org/firefox/?utm_medium=firefox-des...	Erste Schritte	2023-05-06 22:25:01 MESZ	Firefox Analyzer	mozilla.org	CFV_Firefox_Klon_Snapshot_3.img

Abbildung 5.5.: Autopsy Web Bookmarks

(Unter C:/User/Forensik/Favorites/Links) enthält Bing Startseite Snapshot 2: > 5 Einträge in places.sqlite: (Firefox Standardseiten -> Deckt sich mit Beobachtungen aus Process Monitor Analyse, siehe Kapitel X) Snapshot 3: > unverändert zu 2 Snapshot 4: > unverändert zu 3 - Web Cookies: Snapshot 1: > 10 Einträge in WebCacheV01.dat (= DB des Internet

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	MUID8	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=28&GUID=B2C50AD6CB984234A9FE140B81DCB91D8dm...	Microsoft Edge Analyzer	bing.com	CFV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat		15		bing.com	2023-05-06 19:50:17 MESZ	SRCHUSR	DOB=20230506	Microsoft Edge Analyzer	bing.com	CFV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat		15		bing.com	2023-05-06 19:50:20 MESZ	SRCHPGUSR	SRCHLANG=de&LUT=16834026192238JPMH=des204058J...	Microsoft Edge Analyzer	bing.com	CFV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat		15		bing.com	2023-05-06 19:50:19 MESZ	CortanaAppUID	C164AA3A4D47E127DDC66AD915CFD04C	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=A385B679A14D59B0AA027635FFFFFFF	Microsoft Edge Analyzer	bing.com	CFV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat		15		live.com	2023-05-06 19:50:30 MESZ	MUID	118A534093A9626528C5404997A96688	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-MSAUTHP		Microsoft Edge Analyzer	live.com	CFV_Firefox_Klon_Snapshot_3.img

Abbildung 5.6.: Autopsy Web Cookies

Explorers zum speichern von Browserdaten): Cookies für bing.com und live.com (= outlook)

Snapshot 2: > unverändert zu 1 Snapshot 3: > unverändert zu 2 Snapshot 4: > unverändert zu 3 - Web History: Snapshot 1: > 2 Einträge in WebCacheV01.dat: - 2x live.com (=

Source Name	S	C	O	URL	Date Accessed	Referer URL	Title	Program Name	Domain	Data Source	Username
places.sqlite			6	https://www.mozilla.org/de/privacy/firefox/	2023-05-08 22:35:00 MESZ	https://www.mozilla.org/privacy/firefox/	Firefox Datenschutzseite — Mozilla	Firefox Analyzer	mozilla.org	CPV_Firefox_Klon_Snapshot_3.img	
WebCacheV01.dat			15	https://login.live.com/oauth20_desktop.srf?lc=1031	2023-05-08 19:51:06 MESZ			Microsoft Edge Analyzer	live.com	CPV_Firefox_Klon_Snapshot_3.img	Forensik
WebCacheV01.dat			15	https://login.live.com/oauth20_authorize.srf?ident_1...	2023-05-08 19:51:08 MESZ			Microsoft Edge Analyzer	live.com	CPV_Firefox_Klon_Snapshot_3.img	Forensik
WebCacheV01.dat				file:///Z:/Logfile_1	2023-05-08 20:29:36 MESZ			Microsoft Edge Analyzer		CPV_Firefox_Klon_Snapshot_3.img	Forensik
WebCacheV01.dat				file:///Z:/Logfile_2	2023-05-08 20:44:19 MESZ			Microsoft Edge Analyzer		CPV_Firefox_Klon_Snapshot_3.img	Forensik

Abbildung 5.7.: Autopsy Web History

outlook) Snapshot 2: > 1 Eintrag in places.sqlite: -> Zurückzuführen auf Seite, die sich automatisch geöffnet hat, als Firefox gestartet (bevor privates Fenster geöffnet wurde) > 1 neuer Einträge in WebCacheV01.dat: - file:///Z:/Logfile_1 (= Process Monitor Logfile, die in shared-Folder geladen wurde) -> Erklärung? Snapshot 3: > 1 neuer Eintrag in WebCacheV01.dat: - file:///Z:/Logfile_2 (= Process Monitor Logfile, die in shared-Folder geladen wurde) -> Erklärung? Snapshot 4: > unverändert zu 3 - Web Categories: Snapshot

Source Name	AS	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 5.8.: Autopsy Web Categories

1: > 2x WebCacheV01.dat aufgelistet => Mit HxD untersucht, keine PB Artefakte Snapshot 2: > unverändert zu 2 Snapshot 3: > unverändert zu 3 Snapshot 4: > unverändert zu 4

Zusammenfassung: - keine PB Artefakte - Keine neuen Erkenntnisse vgl. mit intensiver Analyse mittels Process Monitor in Kapitel X - Eintrag von Datenschutzseite in places.sqlite wurde erkannt.

Analyse mit Volatility

Vorgehen: Siehe "Methodik" Kapitel - Ausgangslage: Volatility Yarascan Treffer - Für jeden Treffer: virtueller Offset des Strings, PID, getriggerte Yarasrule, getriggerte Yara Component z(= Variablenname des gesuchten Strings), gefundener String - Neue Spalte: "Prozessname" > zu jeder PID Prozessnamen - Ergebnisse Aufbereitet nach folgendem Schema: > Für jeden RAM Dump > Für jede Yarasrule > Für jede Component > Filter: Prozessname = Firefox -> Anzahl zählen > Filter: Prozessname = Alle Prozesse außer Firefox -> Anzahl zählen

HTML Artefakte wurden in keinem RAM Dump gefunden => Nicht aufgeführt

Yarasrule "Keyword": Analyse: > Ausschließlich in RAM Dump 2 Keyword Artefakte gefunden > Hauptsächlich in Firefox Prozess > Mit 1301 Artefakten, am häufigsten pfaenhofen vertreten. Vermutung: Evtl. weil Google Maps viele zusätzliche Artefakte lädt.

Yarasrule ÜRL": Analyse: > Wie bei anderen Kategorien: Die meisten Artefakte in RAM Dump 2, in Firefox Prozessen > mooserliesl tritt am wenigsten auf, donaukurier am meisten (vmtl. auf Öffnen von Bild zurückzuführen) > Hier bemerkenswert, dass in RAM Dump 3 Artefakte

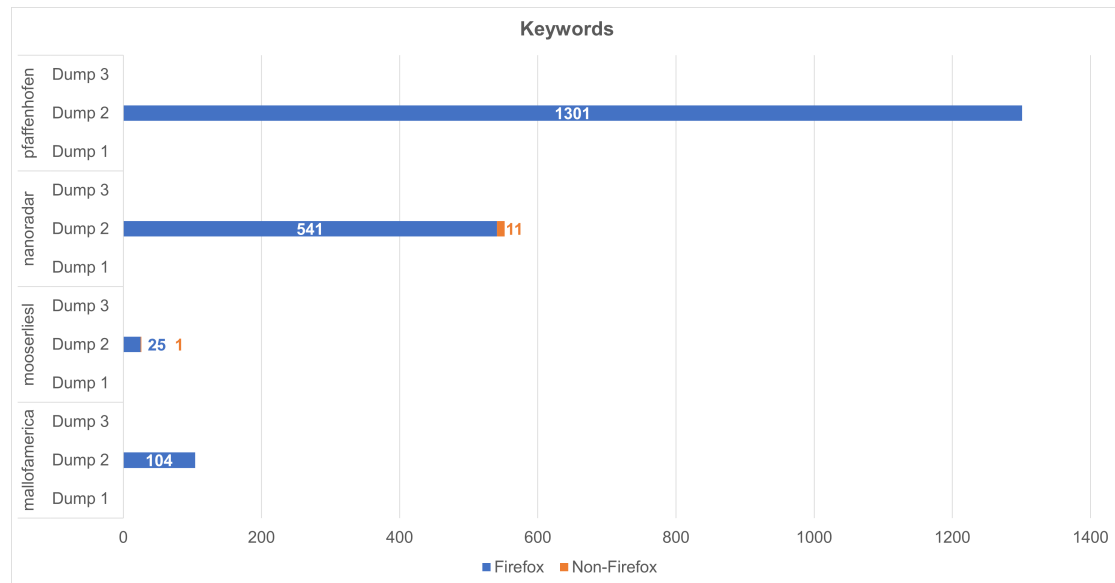


Abbildung 5.9.: Keywords

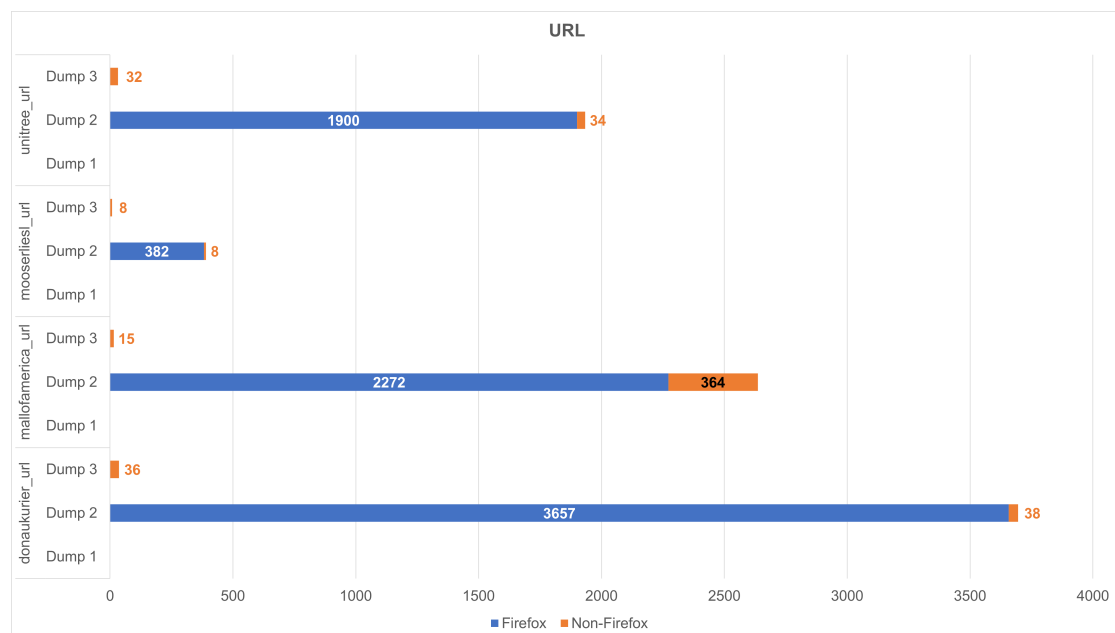


Abbildung 5.10.: URL

von allen vier URLs zu finden sind > Bei genauerer Analyse des Process Monitor Logfiles herausgefunden: Artefakte alle in svchost.exe Prozess gefunden > Deshalb RAM Dump erneut mit Volatility windows.svcscan Plugin untersucht: "The svcscan plugin allows the analyst to list out the services running. This plugin gives more detail to the running processes in

the event that the analyst requires additional details such as the display name, binary path, or service type.» Ausgabe aller im RAM gefundener Services > Problem: Volatility svcscan liefert keine PID zu laufenden Services > Deshalb: "White-Box-Analyse: Snapshot 3 erneut aufgetaut, danach mit Process Explorer PID X (TODO!) von SVChost Prozess gesucht, in dem PB Artefakte gefunden wurden Def. Process Explorer: "Prozess Explorer zeigt Ihnen Informationen darüber an, welche Handles und DLLs-Prozesse geöffnet oder geladen wurden. Process Explorer, from Sysinternals, is a process management program that allows you to see the running processes on your computer and a great deal of information about each process. One of the nice features of Process Explorer is that it also gives you the ability to see what services a particular SVCHOST.EXE process is controlling.» Ergebnis: DNSCache Service mit PID X (TODO!) = DNSCache Service TODO: Screenshot > Ausführung von ipconfig /displaydns liefert gecachte URLs TODO: Screenshot > Nach Löschen des DNSCaches mit ipconfig /flushdns + Schließen aller Process Monitor Instanzen + Beenden des DNSCaches Services + Erneuter RAM-Dump -> Keine PB Artefakte mehr gefunden! Yazarule "Mail": Analyse: > Alle Mail Artefakte gefunden > Ausschließlich in RAM Dump 2 Mail Artefakte

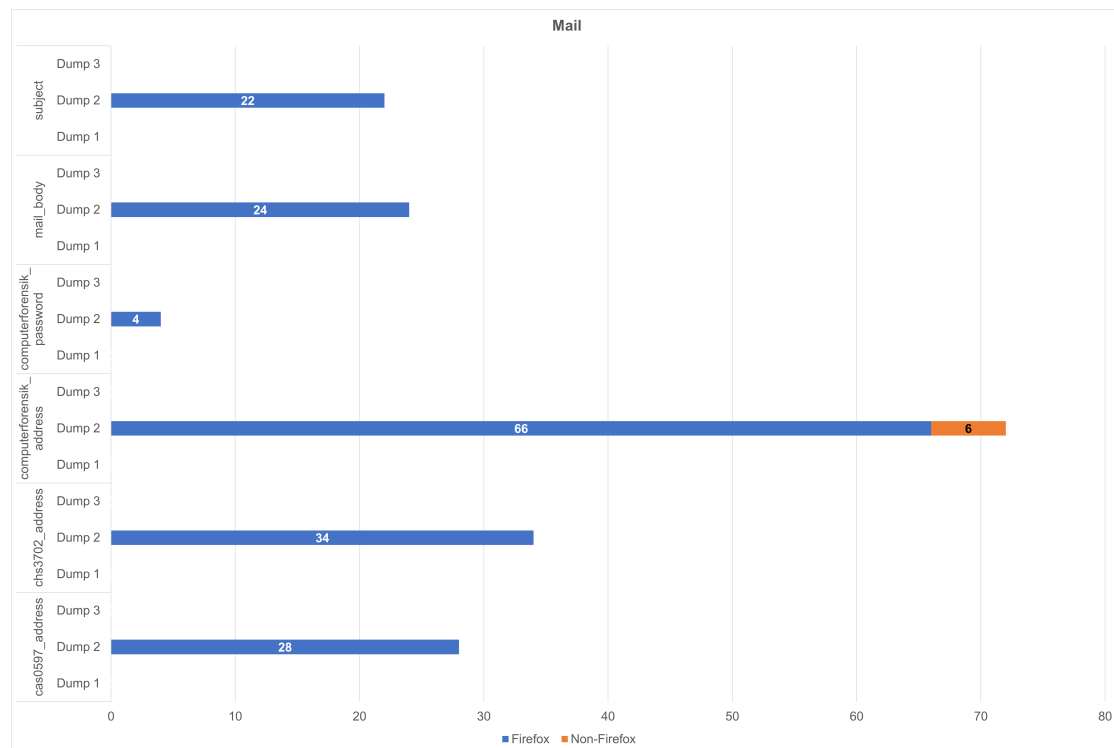


Abbildung 5.11.: Mail

gefunden > Am häufigsten Absenderadresse "computerforensikvl@gmail.com" gefunden, als einziges Artefakt auch in anderen Prozessen gefunden. > Bemerkenswert: Passwort wurde 4x als Klartext im RAM gefunden! String Kontext: Offsets: PIDs: 0xb9ce29180c8 7420 0x2859f4ffd4e0 7420 0x24083b41858 8424 0x240840e5b08 8424

Memmap: Pid 7420 virtual physical size offset in file 0xb9ce2918000 0xcb20a000 0x1000 0x11dd4000 -> 0xb9ce29180c8 = 0x11dd40c8 0x2859f4ffd000 0x96812000 0x1000 0x12e23000

```

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 oaderKKKpÄûèKKKK
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@ŠKKKK
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!KKKKøS)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.™AKKKK
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 4B astCheckC.)KKKKK
11DD4140 58 02 00 00 10 00 00 00 73 65 63 6F 6E 64 73 53 X.....secondsS

```

Abbildung 5.12.: Password in memory page of PID 7420 at offset 0xb9ce29180c8

Disabled -> 0x2859f4ffd4e0 = 0x12e234e0 Memmap: Pid 8424 virtual physical size offset in

```

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 øPyôY(..P.....
12E234E0 56 6F 72 6C 65 73 75 6E 67 32 33 21 A2 1D FB FF Vorlesung23!ø.ûý
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 Passwdüý.-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 €Öiä).....
12E23550 10 02 00 00 1F 00 00 00 0B CC E5 29 02 00 00 .....iä)...

```

Abbildung 5.13.: Password in memory page of PID 7420 at offset 0x2859f4ffd4e0

file 0x24083b41000 0xc1d52000 0x1000 0x583000 Disabled -> 0x24083b41858 = 0x583858 0x240840e5000 0x2d3fb000 0x1000 0x96b000 Disabled -> 0x240840e5b08 = 0x96bb08 > In

```

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 Äzëý@.....
00583810 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
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 E5 ÁÁÁÁÁÁÁÁÁÁÁÁÁÁ
00583850 02 00 00 00 1A 00 00 00 56 00 6F 00 72 00 6C 08 .....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 ..ÁÁÁÁÁÁÁÁÁÁÁÁ
00583880 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ÁÁÁÁÁÁÁÁÁÁÁÁÁÁ
00583890 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ÁÁÁÁÁÁÁÁÁÁÁÁÁÁ
005838A0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ÁÁÁÁÁÁÁÁÁÁÁÁÁÁ
005838B0 08 00 00 00 D7 16 71 67 01 00 00 00 00 00 00 00 ....*.qq.....
005838C0 6F 00 6E 00 44 00 51 00 30 00 4B 00 55 00 62 00 o.n.D.Q.0.K.U.b.

```

Abbildung 5.14.: Password in memory page of PID 8424 at offset 0x24083b41858

PID 8424: 2 Bytes pro Character, bspw. Unicode

Yararule Ĭmage": Analyse: > Hex-Wert von Donaukurier Bild wurde im 2. RAM Dump in 3

[illegible]

Abbildung 5.15.: Password in memory page of PID 8424 at offset 0x240840e5b08

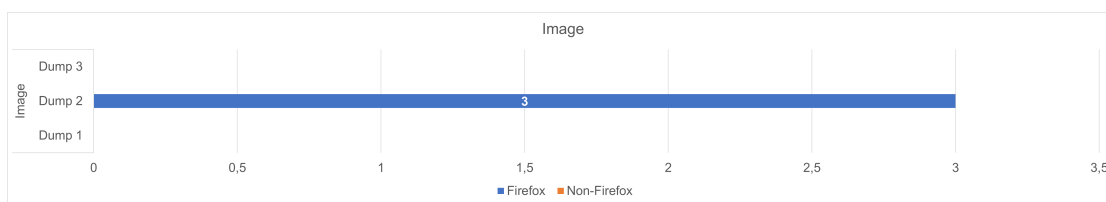


Abbildung 5.16.: Image

Firefox Prozessen gefunden

Zusammenfassung = Stacked Bar Chart:

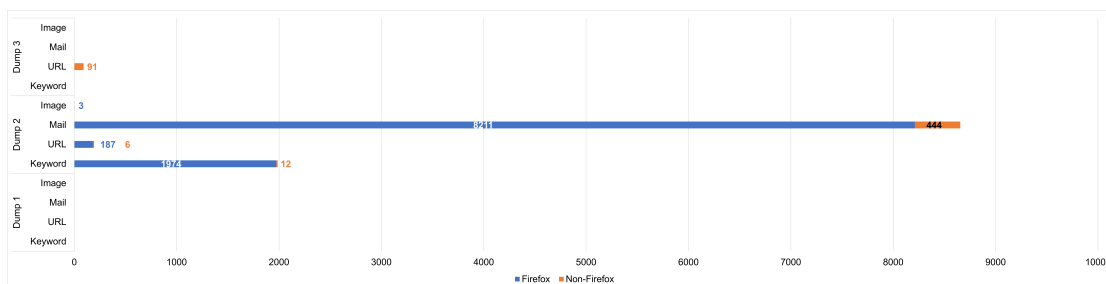


Abbildung 5.17.: Summary

TODO: Kreisdiagramme/Balkendiagramme mit Gesamtzahl an (Non-)Firefox Yarascan-Treffer erst im Vergleich mit Tor

5.2. Tor

White-Box Analyse/Common Locations

Schreiboperationen mit Process Monitor verfolgen:

Im Anhang: Tabelle mit allen geschriebenen Dateien (markiert, wenn nicht mehr wiederherstellbar + markiert, wenn Datei "verändert"(siehe oben: temp, WAL))

Aux-Dateien, welche nicht mehr vorhanden waren, aber dafür "richtige"Dateien:

Ergebnis: Tabelle mit wiederherstellbaren Dateien: Logfile 1 vs. Logfile 2 + Tool mit dem Datei untersucht wurde - Dateien, die in beiden Logfiles nicht wiederherstellbar

Kategorie	Datei	Logfile 1	Logfile 2	Logfile 3
Cache	C:\Cache\profile.default\startupCache\startupCache.8.little	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\startupCache\startupCache.8.little	Keine PB Artefakte	Keine Schreiboperationen	Keine PB Artefakte
Datareporting	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
SQLite	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
Sonstige Dateien	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
	C:\Cache\profile.default\data\reporting\state.json	Keine PB Artefakte	Keine Schreiboperationen	Keine Schreiboperationen
Caches				
Profile Default				

Abbildung 5.18.: Tabelle mit wiederherstellbaren Dateien: Logfile 1 vs. Logfile 2

Allgemein: Tor hat nur einen "Common Pfad - Dateien tauchen in zwei unterschiedlichen Ordnern auf: - -

- Alle Schreiboperationen von Prozess "firefox.exe"durchgeführt, nicht "tor.exe"

=> Keine der Dateien enthält PB Artefakte, trotzdem nachfolgende genauere Betrachtung der wichtigsten Dateien im Zusammenhang des Tor Browsers

Kategorien der Logs: - Cache: > Zweck: "Die Datei BstartupCache.8.little ist eine interne Datei, die von Firefox und dem Tor Browser erstellt wird, um den Startvorgang des Browsers zu beschleunigen. Sie enthält im Wesentlichen eine Zwischenspeicherung von Daten, die beim Starten des Browsers benötigt werden.

Diese Datei enthält Informationen über bereits geladene Browser-Komponenten wie JavaScript-Code, CSS-Dateien, Bilder und andere Ressourcen. Indem der Browser diese Informationen zwischenspeichert, kann er sie beim erneuten Starten des Browsers wiederverwenden, anstatt

sie erneut herunterladen und verarbeiten zu müssen. Dadurch wird die Startzeit des Browsers verkürzt und die allgemeine Leistung verbessert. Analyse: - Tool: HxD - kein PB Artefakte

- datareporting: > Zweck: "Die Datei %s\m Ordner /datareporting enthält Informationen über den Zustand und die Konfiguration des Firefox- oder Tor Browsers. Diese Datei kann Daten über die Verwendung des Browsers, wie z.B. installierte Add-Ons, zuletzt besuchte Websites, Browser-Einstellungen und andere Informationen enthalten. Sie wird verwendet, um dem Browser bei Bedarf den Zustand und die Einstellungen wiederherzustellen.Ähnliche Analyse: - Tool Notepad++ mit JSON Plugin - keine PB Artefakte
- Sonstige Dateien: > Zweck: enthält onion URLs, HTTP Alternative Services is a mechanism that allows servers to tell clients that the service they are accessing is available at another network location or over another protocol. This mapping can be stored in a file in the profile folder. This allows websites that do not support HTTPS to communicate in a secure way via port 443 (Opportunistic Encryption).» Zweck: Ist "NoScriptExtension. Wenn in Firefox geöffnet, kann installiert werden-> TODO: Screenshot, wenn in Firefox per "drag-and-drop"gezogen > Zweck: Enthält SecureDrop Adressen: z.B. sueddeutsche.securedrop.tor.onion (z.B. > Zweck: The file containing the updated security data > Enthält früher private Browsing Artefakte (<https://gitlab.torproject.org/tpo/applications/tor-browser/-/issues/18589>), jetzt aber keine private Browsing Artefakte => Keine der Dateien enthält PB Artefakte
- SQLite: Aus Process Monitor Logfiles erkennbar: Tor verwaltet und beschreibt die exakt gleichen SQLite Datenbanken wie Firefox.

Hier ebenfalls gesondert betrachtet: Fokus auf die Entwicklung von Dateinhalt in allen Snapshots (1, 2, 3-1, 3 und 4) betrachtet

Ergebnisse: > Nach Browser-Installation noch keine SQLite-Datei angelegt (Snapshot 1) >

	Snapshot 1:		Snapshot 2: After Browning Scenario, Brower open		(For only) Snapshot 3-5: After Identity reset		Snapshot 3: After Browning Scenario, Brower closed		Snapshot 4: VM Shutdown	
	File	Browser installation	Vor WAL		Nach WAL		Vor		Nach WAL	
			Index	Diff	Index	Diff	Index	Diff	Index	Diff
SQUIDS	phases.dat	N/A	Initialisiert, Zufall, Online URLs für "The Tor Browser" und "Tor Browser Manual" und "Squid-Side-Store" (http://www.phases.net/squid/squid-side-store/index.aspx?5)	no diff	Index bei vorhandener Seite aktualisiert	no diff	Index bei vorhandener Seite aktualisiert	no diff	no diff	no diff
	cookies.dat	N/A	Leer initialisiert (Nur Squid-Name)	leer	leer	leer	leer	leer	leer	leer
	storage.dat	N/A	Leer initialisiert (Nur Squid-Name)	leer	leer	leer	leer	leer	leer	leer
	favicons.dat	N/A	Initialisiert, Konfiguriert mit Firefox, aber Konfiguration, online neue URLs für "Tor Sidestore", wie "The Tor Blog" oder "Tor Browser Manual" und "Squid-Side-Store" (http://www.phases.net/squid/squid-side-store/index.aspx?5)	no diff	no diff	no diff	in allen drei Fällen Index aktualisiert	no diff	no diff	no diff
	webappstore.dat	N/A	Leer initialisiert (Nur Squid-Name)	leer	leer	leer	leer	leer	leer	leer
	formhistory.dat	N/A	Leer initialisiert (Nur Squid-Name)	leer	leer	leer	leer	leer	leer	leer
	8657145356secretary.dat	N/A	Leer initialisiert (Nur Squid-Name)	leer	leer	leer	leer	leer	leer	leer
	3570102286mcginnist-at.dat	N/A	Initialisiert, Zufall, online neue URLs	no diff	no diff	no diff	gleich aktualisiert	no diff	no diff	no diff
		Leer	Leer							
		User-Agent	User-Agent							

Abbildung 5.19.: Comparison of found PB artifacts between RAM Dumps

Während Browsing Szenario alle DBs Initialisiert, außer "webappsstore.sqlite"(Snapshot 2) - Dabei wurden in places.sqlite automatisch .onion URLs geschrieben, die zu Tor Standardseiten führen, wie "The Tor Blogöder "Tor Browser Manual"bzw. die Tor Spenden-Seite, obwohl keine dieser Seiten aufgerufen wurde TODO: Screenshot von URLs? - in Favicons.sqlite wurden die exakt gleichen Einträge geschrieben, mit dem Präfix "Fake-favicon-uri". Ein tatsächliches Icon wurde nicht in die DB geschrieben - remote settings Datenbank enthielt den gleichen Eintrag wie es bereits bei Firefox der Fall war. Keine PB Artefakte - Restliche Dateien ohne Inhalt, nur Spaltennamen - Nach WAL Checkpoints bleiben Dateien unverändert

> Nach Zurücksetzen der Browser-Identität (Snapshot 3-1) - in places.sqlite: Indizes bei eingetragenen Seiten aktualisiert - restliche Dateien unverändert > Nach Schließen des Browsers (Snapshot 3) - in places.sqlite sowie favicons.sqlite: Indizes bei eingetragenen Seiten aktualisiert - restliche Dateien unverändert - nach WAL Checkpoints bleiben Dateien unverändert > Nach herunterfahren der VM (Snapshot 4) - Alle Dateien unverändert, auch nach WAL Checkpoint

- Zusammenfassung: in keiner Datei PB Artefakte

Quantitativ: (Diagramme) > Balkendiagramm: Für jede Logfilekategorie: Anzahl Schreiboperationen Logfile 1 vs Logfile 2

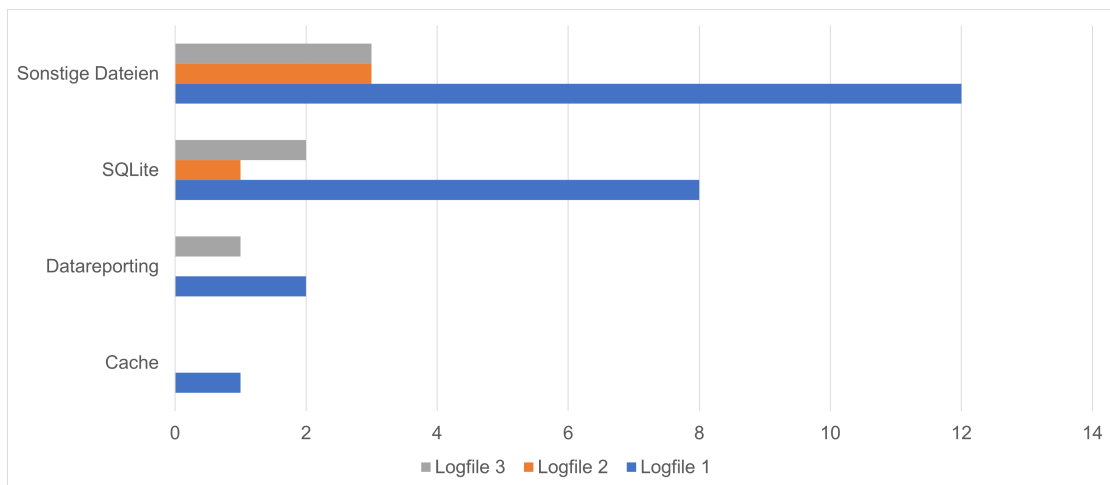


Abbildung 5.20.: Comparison of found PB artifacts between RAM Dumps

Registry

> Process Monitor: SetValue Operationen von Browser Kategorien Registry Keys: Analog zu Firefox 1) PreXULSkeletonUISettings: > Prefix: Absoluter Installationspfad von Firefox > Skeleton UI Einstellungen von Firefox Definition: > Der "PreXULSkeletonUISettings"Registry Key enthielt Einstellungen für die Benutzeroberfläche (UI) des Firefox-Browsers, insbesondere für das sogenannte SSkeleton UI". Das Skeleton UI ist eine vereinfachte Benutzeroberfläche, die während des Ladens des Browsers angezeigt wird, bevor die vollständige Benutzeroberfläche geladen ist. Es besteht aus grundlegenden Steuerelementen und Elementen, die dem Benutzer die Interaktion ermöglichen, während der Rest der Benutzeroberfläche noch geladen wird. > Der "PreXULSkeletonUISettingsSchlüssel enthielt Konfigurationsoptionen wie Farben, Positionen und andere Einstellungen für das Skeleton UI. Durch das Bearbeiten dieses Schlüssels konnten Benutzer die Darstellung des Skeleton UI anpassen. Es ist jedoch wichtig zu beachten, dass das Ändern der Registrierungseinträge ein fortgeschrittenes Verfahren ist und Fehler zu Problemen mit dem Browser führen kann.

> Struktur der Keys: > Unterschiedliche UI Einstellungen - - - - - > keine PB Artefakte unter UI Einstellungen 2) Business Activity Monitoring > Quelle: > BAM is a mostly undocumented feature that controls the programs executed in the background. DAM is a feature for devices supporting the "Connected Standby" mode (i.e when a device is turned on, but its display will be turned off). As a result, the BAM registry keys will contain data on any devices, while DAM registry keys will only contain data on mobile devices. > The BAM registry key contains multiple subkeys under bam

State

UserSettings, with one subkey per user, identified with the user SID. While the key is in the SYSTEM registry hive, program executions can thus still be tied to a specific user using this SID. > Each user-specific key contains a list of executed programs, with their full path and timestamp of last execution. > If a file is deleted, the eventual associated entry in the BAM is deleted as well after the system reboot. Additionally, BAM entries older than 7 days are deleted upon system boot. The BAM thus provides limited information on historic execution of programs > No entries are created in the BAM keys for executables on removable media and/or on network shares. > Key:

Quantitativ: (Diagramme) - Stacked Balkendiagramm jeweils für Logfile 1 und Logfile2: Anteil Kategorie 1 bzw.2 an allen Registry-Schreiboperationen

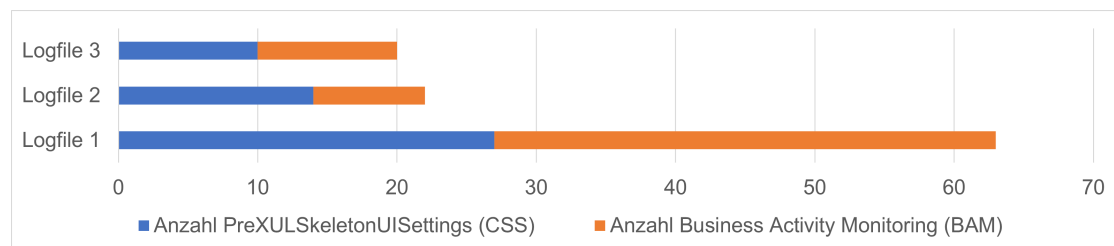


Abbildung 5.21.: Comparison of found PB artifacts between RAM Dumps

> Stringsuche in Registry Hives mit Registry Explorer (Siehe Liste) In allen Hives kein Treffer für alle Suchbegriffe

Literatur: > Wie bei Firefox: Shellactivities Key existiert nicht mehr -> Nicht mehr vorhanden in aktueller Version (Verweis auf E-Mail)

Black-Box Analyse/Uncommon Locations

Analyse mit Autopsy

Bei White-Box Analyse/Common Locations: Autopsy nur zur Dateixtraktion genutzt, hier: als konkretes forensisches Werkzeug

Stichwortsuche: - In allen Snapshots keine Treffer (auch innerhalb \$Carved) - TODO: Pagefile gefunden?

Von Autopsy automatisch indexierte Dateien: In allen Fällen: keine Dateien gelöscht, nur über Zeitraum der Snapshots neue dazugekommen - Web Bookmarks: Snapshot 1: > Bing.url

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 5.22.: Autopsy Web Bookmarks

(Unter C:/User/Forensik/Favorites/Links) enthält Bing Startseite Snapshot 2: > unverändert zu 1 Snapshot 3-1: > unverändert zu 2 Snapshot 3-2: > unverändert zu 3-1 Snapshot 4: > unverändert zu 3-2 - Web Cookies: Snapshot 1: > 9 Einträge in WebCacheV01.dat (= DB










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=D6530A977A1F5DAD20B78D8CFFFEFFFF	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			15	login.live.com	2023-05-07 12:26:40 MESZ	__Host-MSAATHP		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=28GUID=62F5FD78E9D94446BAFDF9DEC8188103&dm...	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			15	bing.com	2023-05-07 12:25:44 MESZ	SRCHUSR	DOB=20230507	Microsoft Edge Analyzer	bing.com
 WebCacheV01.dat			15	bing.com	2023-05-07 12:25:44 MESZ	SRCHHPGUSR	SRCHLANG=de	Microsoft Edge Analyzer	bing.com

Abbildung 5.23.: Autopsy Web Cookies

des Internet Explorers zum speichern von Browserdaten): Cookies für bing.com und live.com (= outlook) Snapshot 2: > unverändert zu 1 Snapshot 3-1: > unverändert zu 2 Snapshot 3-2: > unverändert zu 3-1 Snapshot 4: > unverändert zu 3-2 - Web History: Snapshot 1: >

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?ic=1031	2023-05-07 12:26:40 MESZ	Microsoft Edge Analyzer	live.com	Forensik

Abbildung 5.24.: Autopsy Web History

2 Einträge in WebCacheV01.dat: - 2x live.com (= outlook) Snapshot 2: > 1 neuer Einträge in WebCacheV01.dat: - file:///Z:/Logfile_1 (= Process Monitor Logfile, die in shared-Folder geladen wurde) -> Erklärung? Snapshot 3-1: > 1 neuer Eintrag in WebCacheV01.dat: - file:///Z:/Logfile_2-1 (= Process Monitor Logfile, die in shared-Folder geladen wurde) -> Erklärung? Snapshot 3-2: > 1 neuer Eintrag in WebCacheV01.dat: - file:///Z:/Logfile_2-2 (= Process Monitor Logfile, die in shared-Folder geladen wurde) -> Erklärung? Snapshot 4: > unverändert zu 3-2 - Web Categories: Snapshot 1: > 2x WebCacheV01.dat aufgelistet => Mit HxD untersucht, keine PB Artefakte Snapshot 2: > unverändert zu 2 Snapshot 3-1: > unverändert zu 3 Snapshot 3-2: > unverändert zu 3-1 Snapshot 4: > unverändert zu 3-2

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 5.25.: Autopsy Web Categories

Zusammenfassung: - keine PB Artefakte - Keine neuen Erkenntnisse vgl. mit intensiver Analyse mittels Process Monitor in Kapitel X - .onion URL Einträge in places.sql nicht erkannt

Analyse mit Volatility

Vorgehen: Siehe "Methodik" Kapitel - Ausgangslage: Volatility Yarascan Treffer - Für jeden Treffer: virtueller Offset des Strings, PID, getriggerte Yararule, getriggerte Yara Component z(= Variablenname des gesuchten Strings), gefundener String - Neue Spalte: "Prozessname" zu jeder PID Prozessnamen - Ergebnisse Aufbereitet nach folgendem Schema: > Für jeden RAM Dump > Für jede Yararule > Für jede Component > Filter: Prozessname = Firefox -> Anzahl zählen > Filter: Prozessname = Alle Prozesse außer Firefox -> Anzahl zählen

Wie bei Firefox: HTML Artefakte wurden in keinem RAM Dump gefunden => Nicht aufgeführt

Yararule "Keyword": Analyse: > Ausschließlich in RAM Dump 2 und RAM Dump 3-1 Keyword

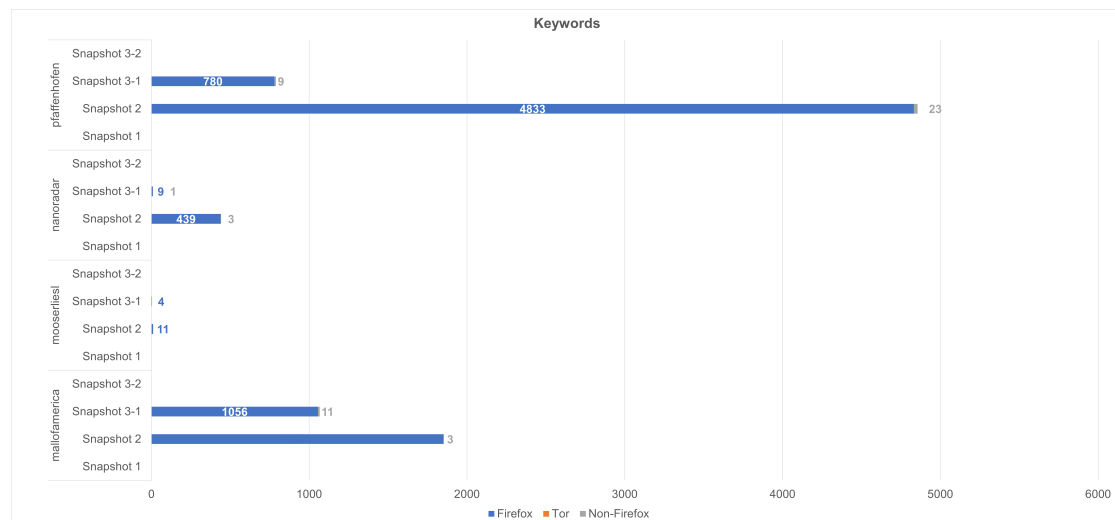


Abbildung 5.26.: Keywords

Artefakte gefunden > In RAM Dump 3-1 bei jedem Keyword deutlich weniger Artefakte als in RAM Dump 2 => Identitäts-Reset reduziert Keyword Artefakte deutlich > Hauptsächlich in Firefox Prozess, kein Artefakt in Tor.exe Prozess > Mit 4833 Artefakten in RAM Dump 2

am häufigsten "pfaffenhofen" vertreten. Vermutung: Evtl. weil Google Maps viele zusätzliche Artefakte lädt. > Nach Schließen von Tor Browser: keine Keyword Artefakte mehr in RAM

Yararule ÜRL": Analyse: > Wie bei Yararule "Keyword": Ausschließlich in RAM Dump 2

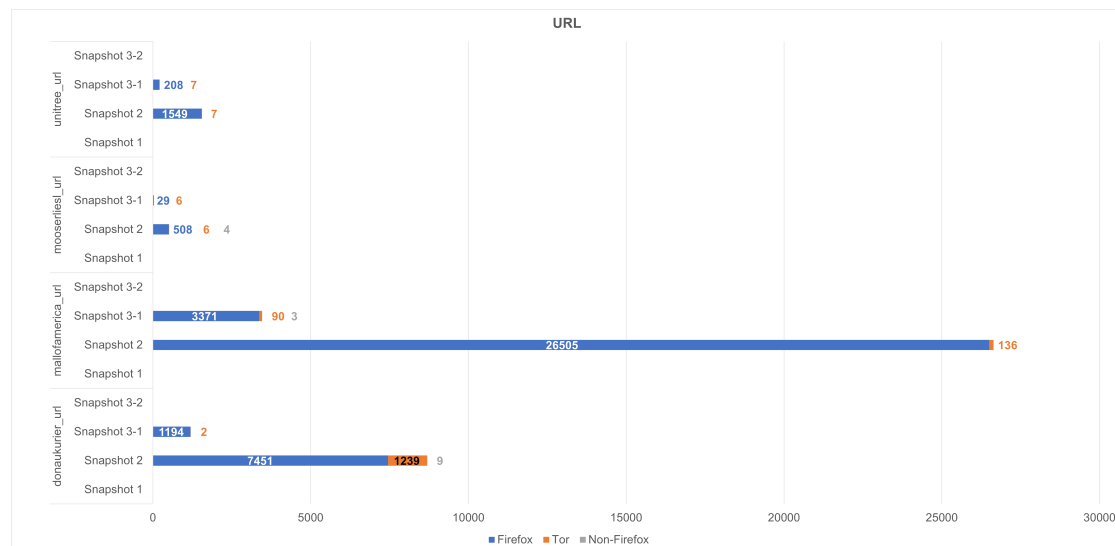


Abbildung 5.27.: URL

und RAM Dump 3-1 Keyword Artefakte gefunden > In RAM Dump 3-1 bei jedem Keyword deutlich weniger Artefakte als in RAM Dump 2 => Identitäts-Reset reduziert URL Artefakte deutlich > Hauptsächlich in Firefox Prozess, danach am häufigsten Tor.exe Prozess und am wenigsten Artefakte in anderen Prozessen > Bemerkenswert: "mallofamerica.com" ist mit 26.505 mal in RAM Dump 2 am häufigsten als Artefakt gefunden worden. Vergleich: "mooserlies.de" wurde nur 508 mal in RAM Dump 2 gefunden > Nach Schließen von Tor Browser: keine URL Artefakte mehr in RAM

> TODO: DNSCache?

Yararule "Mail": Analyse: > Alle Mail Artefakte gefunden > Artefakte ausschließlich in Firefox Prozess gefunden > Artefakte fast ausschließlich in RAM Dump 2 Mail gefunden > Nur die Absenderadresse "computerforensikvl@gmail.com" wurde nach Identitäts-Reset in RAM Dump 3-1 gefunden > Absenderadresse ist häufigstes Mail Artefakt > Bemerkenswert: Passwort wurde 2x als Klartext im RAM gefunden! String Kontext: Offsets: PIDs: 0xb9ce29180c8 7420 0x2859f4ffd4e0 7420 0x24083b41858 8424 0x240840e5b08 8424

Yararule "Image": Analyse: > Hex-Wert von Donaukurier Bild wurde ein einziges mal im 2. RAM Dump in einem Firefox Prozess gefunden

Zusammenfassung = Stacked Bar Chart: - PB Artefakte ausschließlich in RAM Dump 2 und 3-1 gefunden - Nach Identitäts-Reset deutlich weniger Artefakte vorhanden - Am meisten

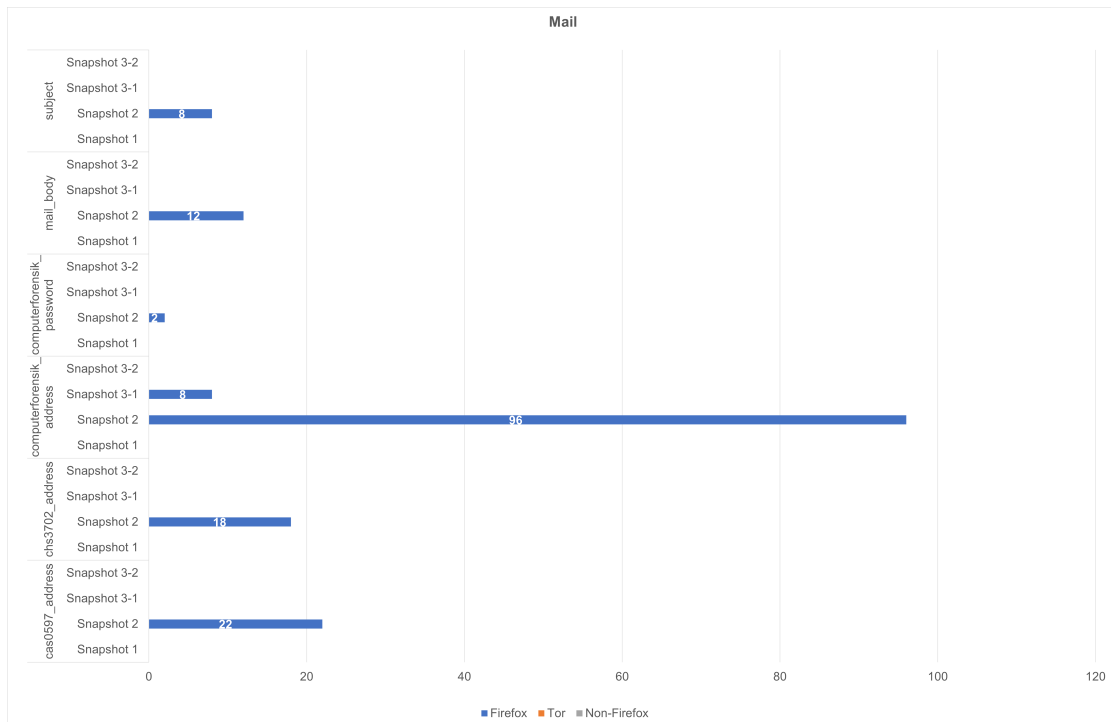


Abbildung 5.28.: Mail

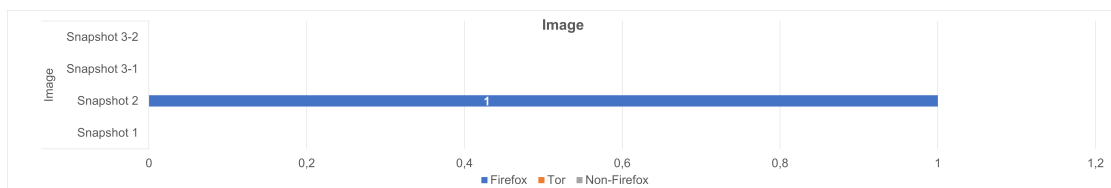


Abbildung 5.29.: Image

URL-Artefakte gefunden, wobei mallofamerica.com dominant - HTML Artefakte wurden in keinem RAM Dump gefunden

TODO: Kreisdiagramme/Balkendiagramme mit Gesamtzahl an (Non-)Firefox Yarascan-Treffer erst im Vergleich mit Tor

Uncommon Locations

Literatur:

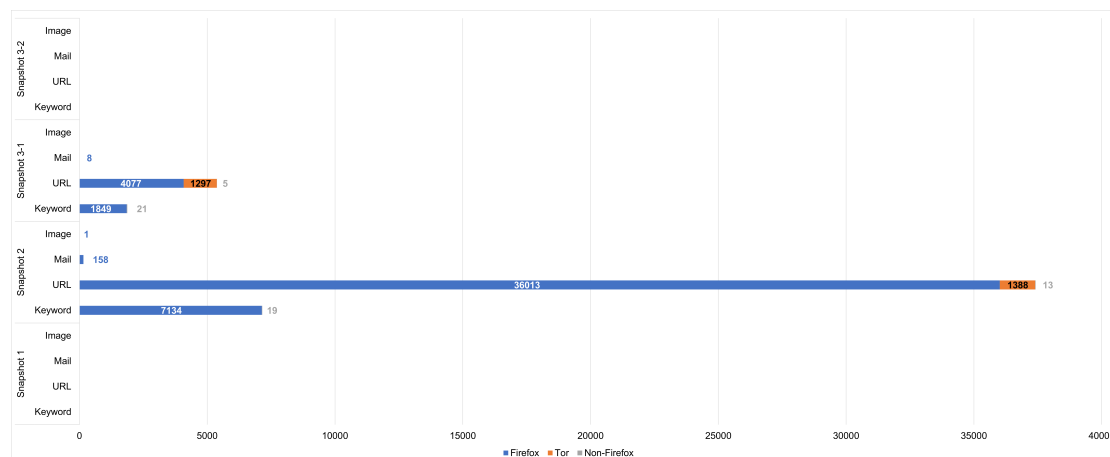


Abbildung 5.30.: Summary

5.3. Chrome

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

Common Locations

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

Eine weitergehende Analyse der Dateien sowie den Datenbänken ist im [link Anhang] detailliert aufgeführt.

Uncommon Locations

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

Analyse mit Autopsy

Auch in Autopsy konnten keine Artefakte der durchgeführten Browsing-Session gefunden werden, weder durch die Substringsuche, noch durch die Analyse der von Autopsy kategorisierten Dateien. Eine weitergehende Analyse einiger kategorisierter Dateien ist in [link Anhang] zu sehen.

Analyse mit Volatility

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

Yara-Regel „HTML“ Im Gegensatz zu den Ergebnissen der Analyse der beiden vorherigen Browser, konnte ein HTML-Fragment bei Chrome im Arbeitsspeicher gefunden werden. Wie in Tabelle 5.1 dargestellt, wird der String „>Themen:“ im zweiten RAM-Dump in einem Chrome-Prozess gefunden.

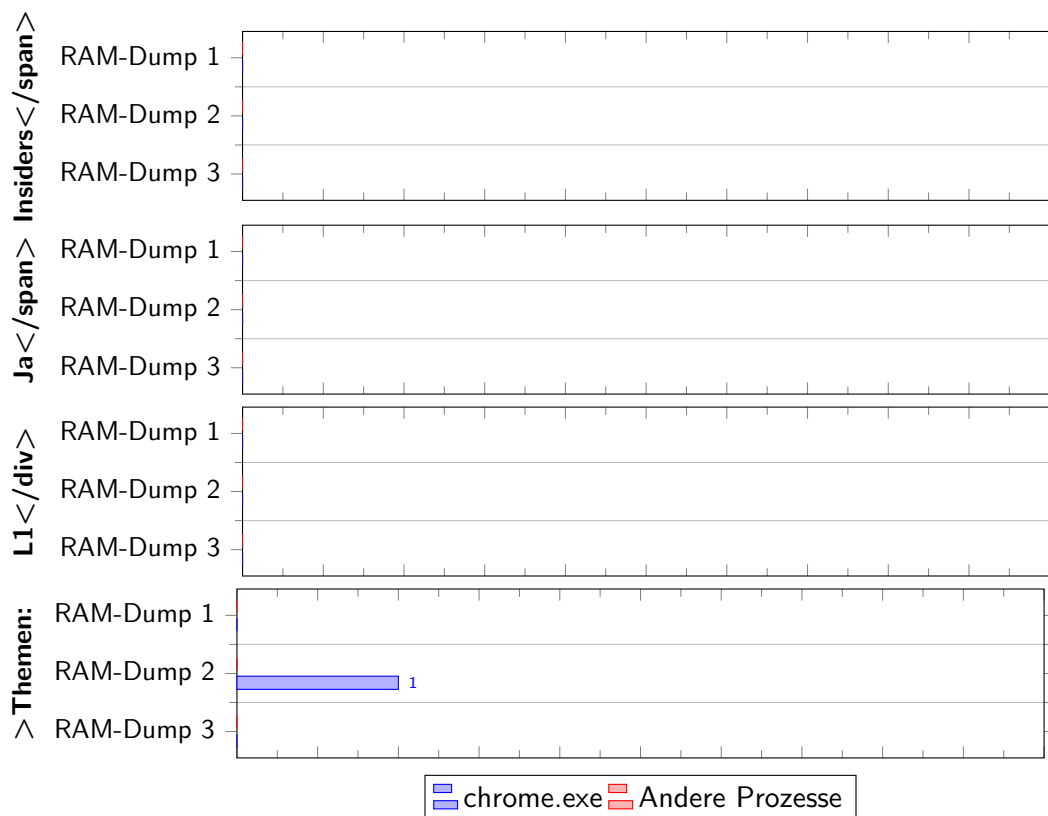


Tabelle 5.1.: Anzahl gefundener HTML-Fragmente im Chrome RAM

Das Volatility-Plugin lieferte hierbei neben dem Treffer noch die PID (7012 in diesem Fall) des Prozesses, in welchem der String gefunden wurde, sowie die virtuelle Adresse, an welchem

das Fragment gefunden wurde (0xa6c018187e1). Dies ist auch in [link Graphik] links zu sehen. Mit dieser Information wurde dann zunächst der Befehl `python3 vol.py -f LOCATION-DUMP windows.memmap -pid 7012` ausgeführt, um das Memory-Mapping des Prozesses zu erhalten, also die Zuordnung der virtuellen zu den physischen Adressen des Prozesses, sowie deren Offset im Memory-Dump-File. Nachfolgend ist ein Ausschnitt der Ausgabe des Befehls zu sehen:

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

In der Farbe rot ist hervorgehoben, dass die virtuelle Adresse 0xa6c01818000 auf die physische Adresse 0xd2dfb000 gemapped ist und der Offset im memory-file 0x105e000 entspricht. Mithilfe dieser Adressen konnte dann die tatsächliche Position des String im memory-file errechnet werden. Diese betrug somit 0x105e7e1. Anschließend wurde der gesamte Prozess mithilfe von Volatility gedumped und dieses memory-file wurde anschließend in den Hex-Editor HxD geladen. Dort wurde dann genau an dem errechneten Offset nach dem String gesucht und konnte genau dort auch gefunden werden, was auch in Abbildung 5.31 zu sehen ist.

```

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Dekodierter Text
0105E6B0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0105E6C0 20 20 20 20 20 20 20 20 20 20 20 20 3C 64 69 76 <div
0105E6D0 20 63 6C 61 73 73 3D 22 6C 61 6E 64 69 6E 67 2D class="landing-
0105E6E0 70 61 67 65 5F 5F 62 6C 6F 63 6B 22 20 64 61 74 page_block" dat
0105E6F0 61 2D 65 7A 2D 62 6C 6F 63 6B 2D 69 64 3D 22 33 a-ez-block-id="3
0105E700 33 32 34 34 34 36 22 3E 0A 20 20 20 20 20 20 20 324446">.
0105E710 20 20 20 20 20 20 20 20 20 20 20 20 20 20 3C 64 69 <di
0105E720 76 20 63 6C 61 73 73 3D 22 63 6F 6E 74 61 69 6E v class="contain
0105E730 65 72 20 62 67 2D 67 72 65 79 22 20 69 64 3D 22 er bg-grey" id="
0105E740 74 68 65 6D 65 6E 2D 74 6F 70 22 3E 3C 73 65 63 themen-top"><sec
0105E750 74 69 6F 6E 20 63 6C 61 73 73 3D 22 74 68 65 6D tion class="them
0105E760 65 6E 20 70 61 64 64 69 6E 67 2D 74 6F 70 2D 31 en padding-top-l
0105E770 35 20 70 61 64 64 69 6E 67 2D 62 6F 74 74 6F 6D 5 padding-bottom
0105E780 2D 31 35 20 73 63 72 6F 6C 6C 62 61 72 2D 68 22 -15 scrollbar-h"
0105E790 3E 3C 64 69 76 20 63 6C 61 73 73 3D 22 64 2D 66 ><div class="d-f
0105E7A0 6C 65 78 20 61 6C 69 67 6E 2D 69 74 65 6D 73 2D lex align-items-
0105E7B0 63 65 6E 74 65 72 22 3E 3C 64 69 76 20 63 6C 61 center"><div cla
0105E7C0 73 73 3D 22 6D 61 72 67 69 6E 2D 72 69 67 68 74 ss="margin-right
0105E7D0 2D 31 35 20 66 6F 6E 74 2D 73 69 7A 65 2D 31 36 -15 font-size-l6
0105E7E0 22 3E 54 68 65 6D 65 6E 3A 0A 09 09 09 09 3C 2F ">Themen:.....</
0105E7F0 64 69 76 3E 3C 64 69 76 3E 3C 75 6C 20 63 6C 61 div><div><ul cla
0105E800 73 73 3D 22 6C 69 73 74 2D 69 6E 6C 69 6E 65 20 ss="list-inline
0105E810 6D 62 2D 30 22 3E 3C 6C 69 20 63 6C 61 73 73 3D mb-0"><li class=
0105E820 22 6C 69 73 74 2D 69 6E 6C 69 6E 65 2D 69 74 65 "list-inline-ite
0105E830 6D 22 3E 3C 61 20 68 72 65 66 3D 22 68 74 74 70 m"><a href="http
0105E840 73 3A 2F 2F 65 70 61 70 65 72 2E 64 6F 6E 61 75 s://epaper.donau
0105E850 6B 75 72 69 65 72 2E 64 65 22 20 74 61 72 67 65 kurier.de" targe
0105E860 74 3D 22 5F 62 6C 61 6E 6B 22 20 63 6C 61 73 73 t="blank" class
0105E870 3D 22 62 74 6E 20 62 74 6E 2D 74 61 67 20 72 6F ="btn btn-tag ro
0105E880 75 6E 64 65 64 2D 31 35 22 3E 0A 09 09 09 09 unded-15">.....

```

Abbildung 5.31.: Ausschnitt aus dem Memory-Dump-File an der Stelle des gefundenen Strings

Daraufhin wurde weitergehend versucht zu analysieren, ob nur ein Teil der Website im RAM verfügbar ist, oder ob das die komplette Donaukurier-Website als HTML Artefakt im RAM vorhanden ist. Dafür wurde die Internetseite donaukurier.de nochmals aufgerufen,

heruntergeladen und der Beginn sowie das Ende der HTML-Datei analysiert. Nachfolgend sind die ersten sowie letzten Wörter der HTML-Datei zu sehen:

Beginn und Ende der Donaukurier-Website

```
<!DOCTYPE html>
<html lang="de">
<head>

<!-- ... -->

</script>
</body>
</html>
```

Davon ausgehend wurde in HxD zuerst nach dem String „<!DOCTYPE html>“ und anschließend nach „</html>“ gesucht. Beide Strings wurden mehrfach in dem Memory-Dump gefunden. Da jedoch keine weiteren Website-typischen Strings mehr durch Yarascan identifiziert werden konnten, wurden die übrigen String-Treffer nicht mehr weiter analysiert.

Da bei der Suche des Strings mehr als einen Treffer gab, wurde genau der String ausgewählt, welcher einen niedrigeren Offset als 0x105e7e1 aufweist. In diesem Fall war das der Treffer bei dem Offset von 105201b, was auch in Abbildung 5.32 zu sehen ist (markierter Teil). Dabei ist deutlich zu erkennen, dass die Website hier beginnt, da alles Nachfolgende mit dem Beginn der heruntergeladenen Website übereinstimmt und darüber nichts im RAM gespeichert wurde.

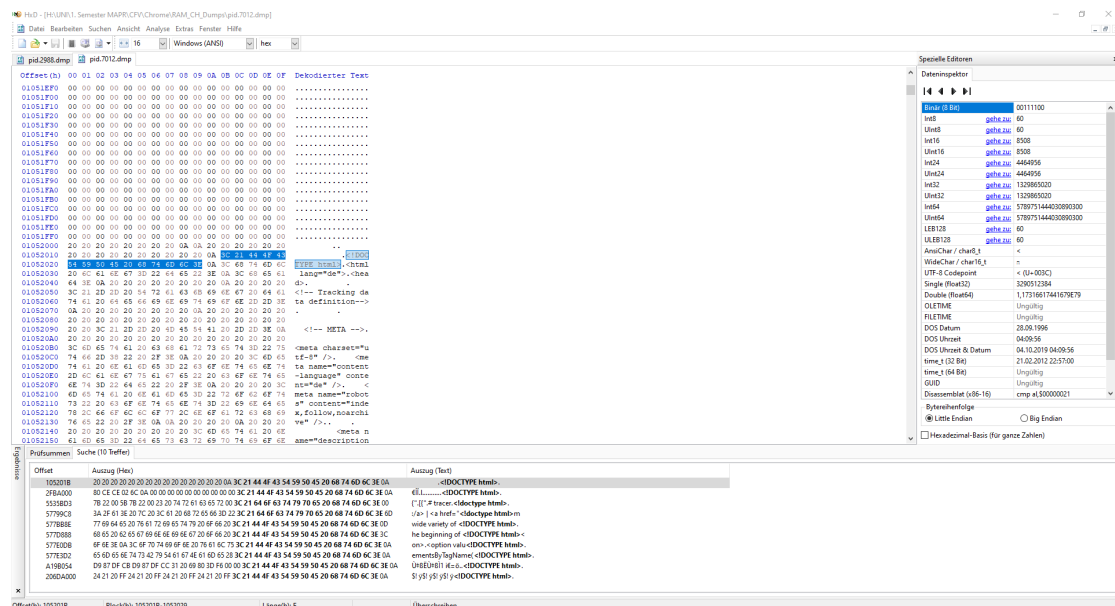


Abbildung 5.32.: Beginn der Donaukurier-Website

Gleiches wurde mit dem Ende der HTML-Datei durchgeführt. Abbildung 5.33 zeigt einen

Screenshot mit dem relevanten String-Treffer, welcher das Ende der Website donaukurier.de darstellt.

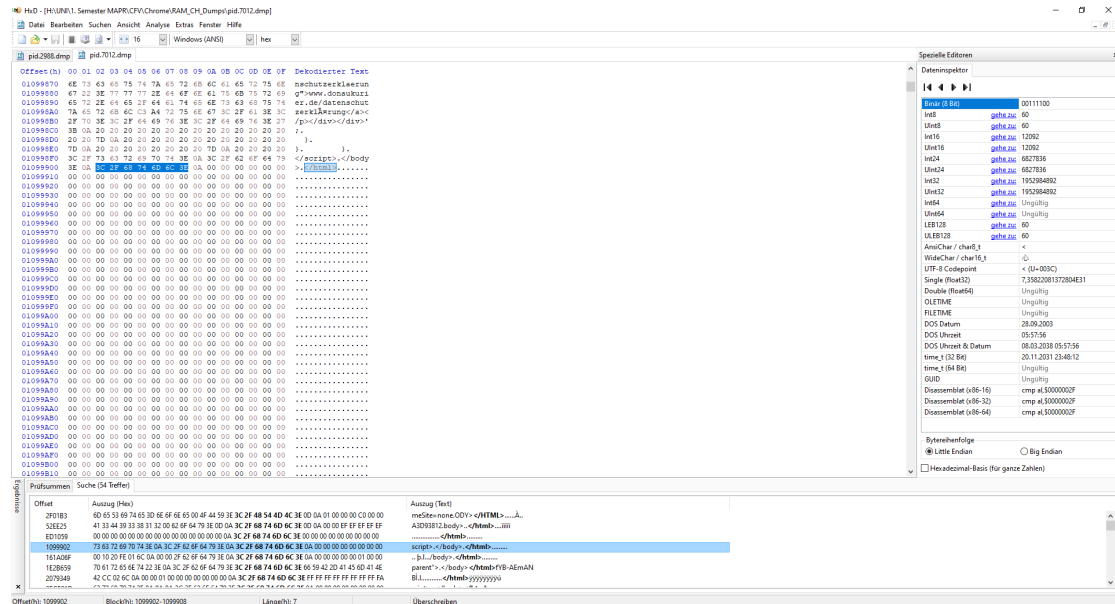


Abbildung 5.33.: Ende der Donaukurier-Website

Anschließend wurde alles, was sich zwischen dem gefundenem Anfang und Ende befindet, als Text kopiert und in eine html-Datei gespeichert. Ein detaillierter Vergleich dieses extrahierten Artefakts mit der heruntergeladenen Datei lieferte das Ergebnis, dass die komplette Website rekonstruiert werden kann, die Websites sich nur in den neuesten Artikeln und Überschriften unterschieden. Übriges war zu 100% identisch.

Zusammenfassend konnte somit die Website „donaukurier.de“ im zweiten Memory-Dump komplett aus dem RAM extrahiert und rekonstruiert werden.

Yara-Regel „Suchbegriffe“ Wie in Tabelle 5.2 dargestellt, wurden die Suchbegriffe „pfaffenhofen“, „nanoradar“, „mooserliesl“ und „mallofamerica“ ausschließlich bei dem noch geöffnetem Chrome-Browser nach dem Browsing-Szenario (zweiter RAM Dump) gefunden. Es wurden dabei alle Suchbegriffe bis auf einen einzigen in Speicherbereichen des Chrome-Browsers gefunden. Am häufigsten konnte der Suchbegriff „pfaffenhofen“ mit 1922 Treffern identifiziert werden. Am seltensten war das der Fall bei dem Begriff „mooserliesl“ mit nur 281 Suchtreffern.

Der eine Treffer bei der Suchphrase „mallofamerica“ war im Speicher eines Prozesses mit dem Namen *MemCompression*, was anhand der PID 1828 festgestellt wurde. Dieser Prozess ist dafür zuständig, dass Windows einen Teil des Arbeitsspeichers komprimiert speichert, was zwar zusätzliche CPU-Ressourcen beansprucht, dafür jedoch deutlich schneller ist, als den Hauptspeicher in das pagefile auszulagern [11]. Daher kann es sein, dass ein Teil des RAMs,

in welchem der Suchbegriff vorhanden war, komprimiert wurde und folglich dieser Prozess diesen Begriff beinhaltete.

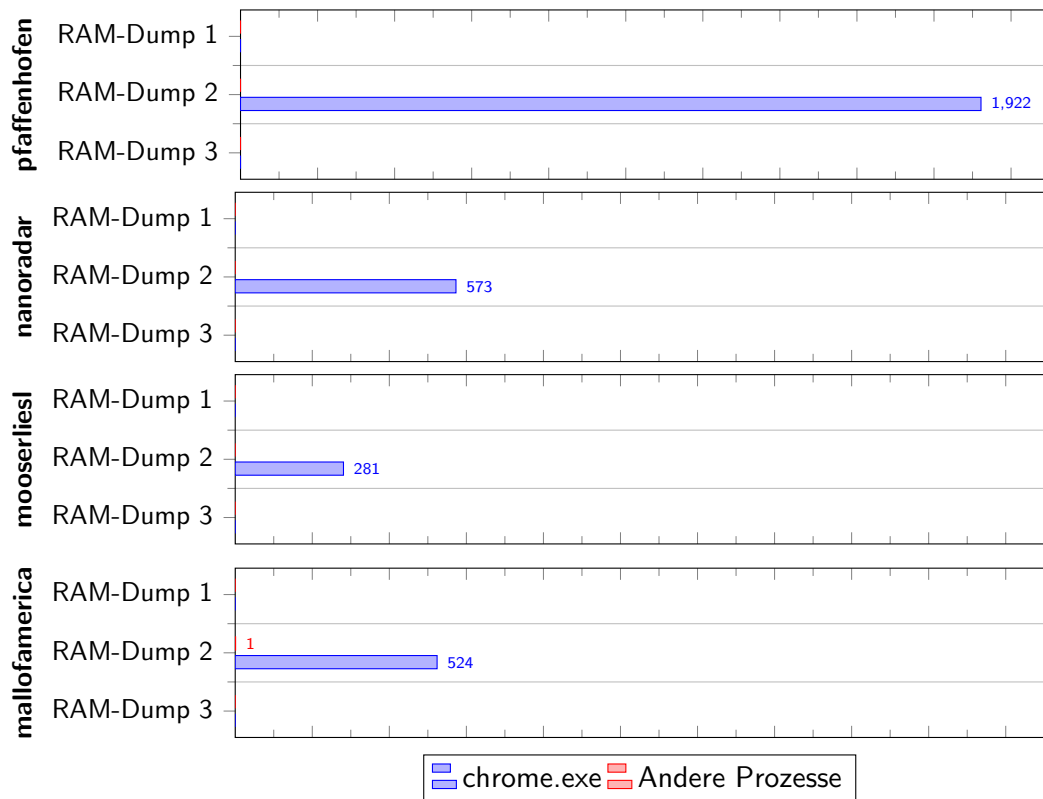


Tabelle 5.2.: Anzahl gefundener Suchbegriffe im Chrome RAM

Yara-Regel „URLs“ Tabelle 5.3 zeigt, dass in den RAM-Dumps alle Suchbegriffe identifiziert werden konnten. Im ersten Speicherabbild wurden dabei keine URLs gefunden, im zweiten am meisten und im dritten, also nach Beenden des Browsers, konnten auch noch 5 URLs identifiziert werden. Dabei waren es bei der URL „donaukurier.com“ am meisten Suchtreffer mit insgesamt 8157 Treffern, 10 davon waren nicht im Speicher von Chrome-Prozessen zu finden. Zwei davon waren im Prozess *MemCompression*, die restlichen acht wurden im Prozess mit der PID 3760 identifiziert, was in diesem Fall der *sihost.exe* war. Dieses Programm entspricht dem *Shell Infrastructure Host*, welcher die Grafikbenutzeroberflächen erstellt und verwaltet, wie beispielsweise Desktop-Hintergründe, Popup-Benachrichtigungen und Taskleisten [26].

Yara-Regel „E-Mail“ Wie in Tabelle 5.4 gezeigt, wurden sowohl nach dem Browsing Szenario bei geöffnetem Browser, als auch nach Schließen desselben E-Mail Artefakte gefunden. Darunter war die E-Mail-Adresse mit 172 Treffern, 8 davon außerhalb von Chrome-Prozessen,

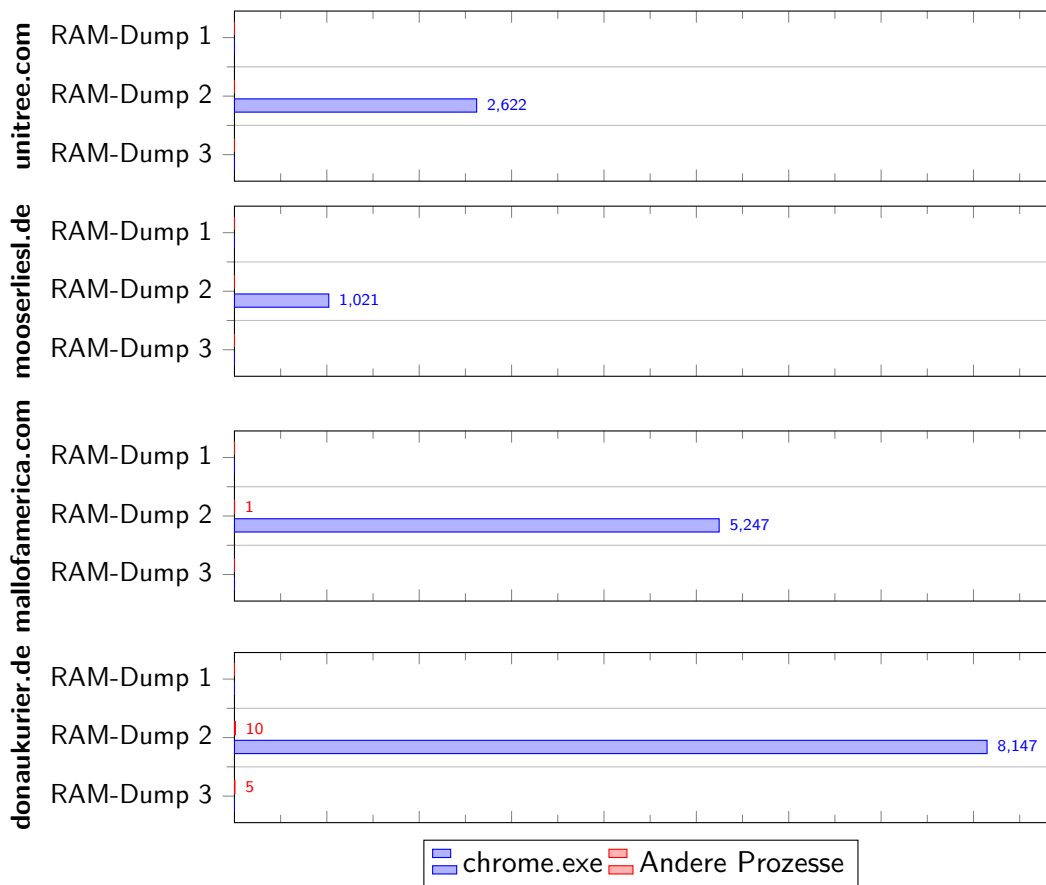


Tabelle 5.3.: Anzahl gefundener URLs im Chrome RAM

das häufigste Suchresultat. unter den acht externen Prozessen waren der *Desktop Window Manager*, welcher für visuelle Effekte wie Desktop-Animationen und halbtransparente Fenster verantwortlich ist [18], *explorer.exe*, *sihost.exe* und weiteren Windows-Prozessen. Da dies aber meist Prozesse waren, welche grafische Aktionen durchführten, wurden diese nicht weiter analysiert.

Die beiden Studenten-Mailadressen, an welche die Mail versendet wurden, waren mit 121 bzw. 97 Suchtreffern in Chrome-Prozessen vertreten. Der Mailtext war mit 136 Artefakten mehr als doppelt so oft im RAM zu finden als der Betrefftext. Im dritten RAM-Dump wurde dann noch die E-Mail-Adresse im Prozess *explorer.exe* gefunden.

Grundsätzlich wurden bei dieser Yara-Regel wenige Artefakte identifiziert im Vergleich zu den URLs beispielsweise.

Yara-Regel „DK-Logo“ Bei der letzten Yara-Regel zeigt Tabelle 5.5 die Ergebnisse, in welchem RAM-Dump und wie oft das Donaukurier Logo im RAM identifiziert werden konnte. Zu sehen ist, dass dies nur im zweiten Speicherabbild der Fall war und insgesamt 3 mal dort

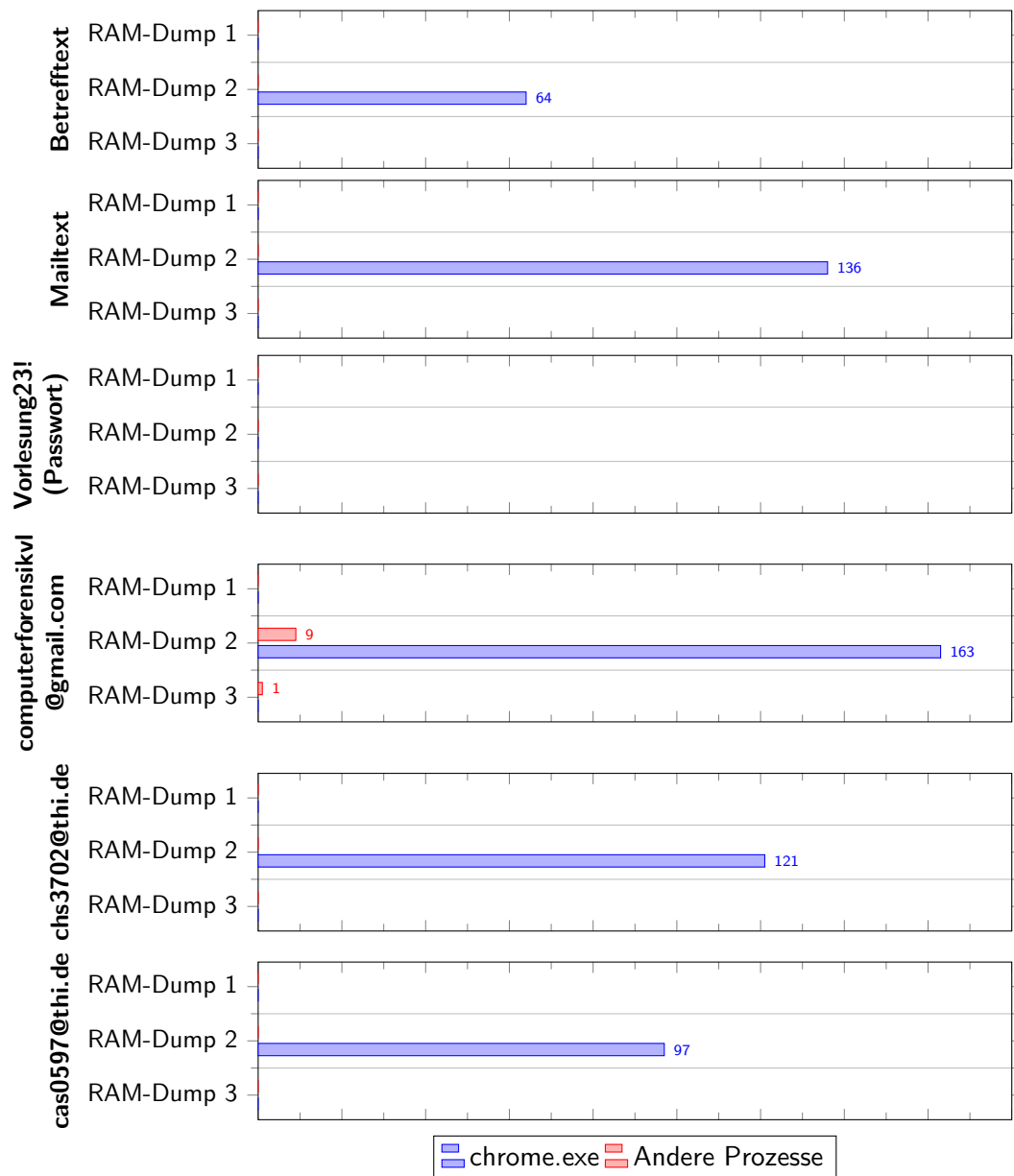


Tabelle 5.4.: Anzahl gefundener E-Mail Artefakte im Chrome RAM

in einem Chrome-Prozess zu finden war.

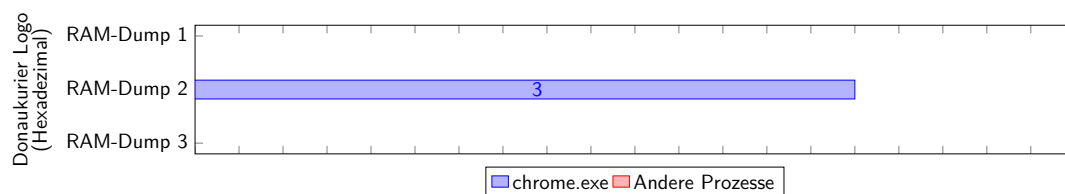


Tabelle 5.5.: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im Chrome RAM

Registry

Wie in [\[link Kapitel\]](#) bereits beschrieben, zählt die Analyse der Registry sowohl zu den Common als auch zu den Uncommon Locations. Es können weder in den „SetValue“-Operationen in den Process Monitor Logs noch durch die Analyse der System- und User-Hives durch den RegistryExplorer Artefakte gefunden werden. Eine weitergehende Analyse der Registry ist im [\[link Anhang\]](#) beschrieben.

5.4. Brave

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

Common Locations

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

Eine umfangreiche Auswertung der Daten sowie den Datenbanken befindet sich in [\[link Anhang\]](#).

Uncommon Locations

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

Analyse mit Autopsy

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

Zunächst wurde die Stringsuche gemäß [link Kapitel] eingesetzt, wobei keine Suchtreffer herauskamen.

Zusätzlich wurden automatisch kategorisierte Dateien [ergänzen] untersucht, wobei hier auch keine Artefakte zu finden waren. [link Anhang] geht auf diese Dateien ausführlicher ein.

Analyse mit Volatility

Für die Analyse des Arbeitsspeichers wurde das Forensik-Tool Volatility verwendet, womit eine Stringsuche mittels des Plugins Yarascan durchgeführt wird. Damit durchsucht man ein Arbeitsspeicherabbild nach gewissen Strings, welche zuvor in einer yara-rule festgelegt werden können. Die für diese Arbeit verwendete Datei ist in [link Anhang runter] aufgeführt.

Yara-Regel „HTML“ Beim Brave-Browser konnte ein HTML-Fragment wiederhergestellt werden, was in Tabelle 5.6 dargestellt ist. Es ist wieder der String „>Themen:“, welcher einmal im zweiten RAM-Dump vorliegt.

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

Anschließend an die Wiederherstellung der HTML-Datei wurden die beiden extrahierten Dateien nochmals gegeneinander verglichen. Abbildung 5.34 zeigt dies anhand der compare-Funktionalität von Visual Studio Code. An der rechten Seite in der Bildlaufleiste sind die Sektionen farbig dargestellt, welche Unterschiede aufweisen. Dabei wurde durch Analyse derer deutlich, dass der Aufbau und die Struktur beider extrahierter Webseiten übereinstimmt, es nur geringe Unterschiede bei einigen Namen der Artikel und Meldungen gibt. Auch sind die Dateien auch (nahezu) identisch groß (290kB).

Yara-Regel „Suchbegriffe“ Tabelle 5.7 zeigt, dass alle der vier Suchbegriffe jeweils gefunden wurden, jedoch nur im zweiten RAM-Dump, also nach der Durchführung des Browsing-Szenarios bei noch geöffnetem Browser. Der String „pfaffenhofen“ wurde dabei am häufigsten mit 1447 Treffern, „nanoradar“ am seltensten mit nur 51 Suchergebnissen gefunden. Dabei wurden auch alle Suchbegriffe rein in Brave-Prozessen ausfindig gemacht.

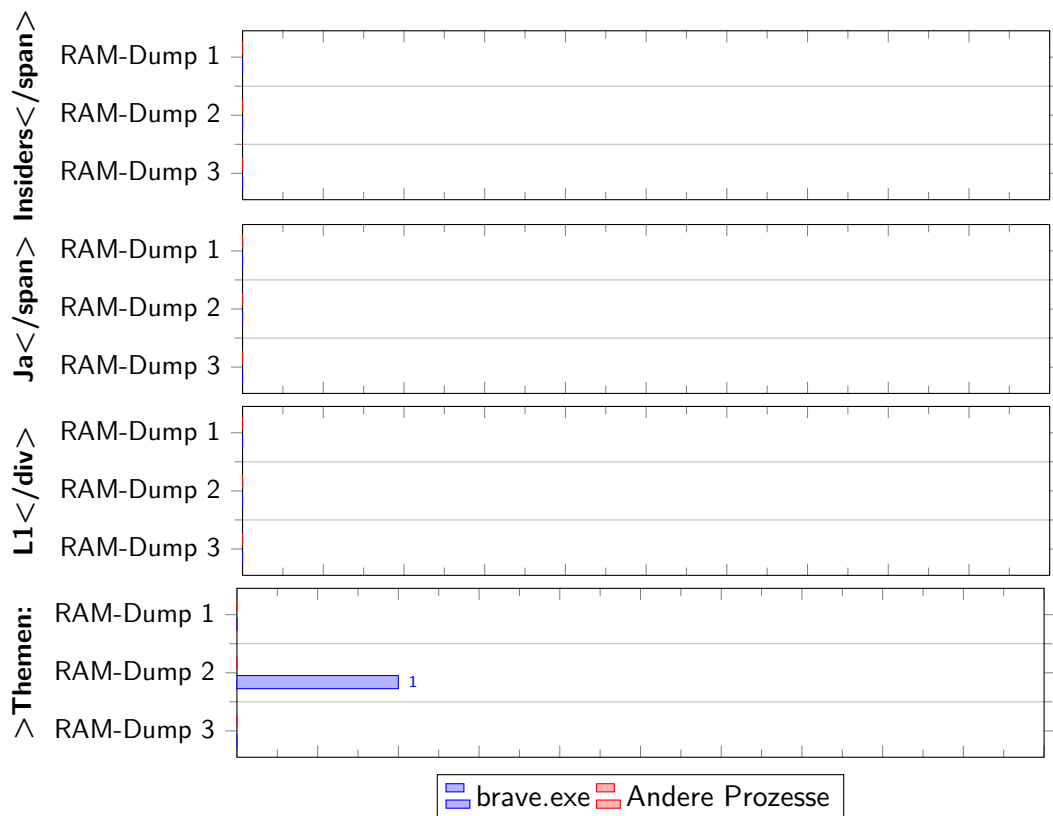


Tabelle 5.6.: Anzahl gefundener HTML-Fragmente im Brave RAM

Yara-Regel „URLs“ Auch bei der URL-Yara-Rule konnte wieder alle URLs im RAM nachgewiesen werden. Bei „donaukurier.de“ gab es sogar einen Treffer im dritten RAM-Dump, alle anderen waren nur im zweiten Speicherabbild auffindbar. Dabei waren es auch bei der Donaukurier-URL die meisten Treffer mit insgesamt 3514 Artefakten, wobei sechs davon in anderen Prozessen neben Brave gefunden wurden. Sehr präsent war hier der Prozess „NisSrv.exe“. Dieser ist ein Teil des Microsoft Defenders [20], was grundsätzlich merkwürdig erscheint, sich aber dadurch erklären lässt, dass dieser vielleicht im Hintergrund den Datenverkehr mitliest und entscheidet, ob gewisse Aktivitäten in Ordnung sind oder evtl. ein Virus oder anderweitige bösartige Software unsicheren Netzwerkverkehr tätigt.

Yara-Regel „E-Mail“ Wie in Tabelle 5.9 dargestellt, wurden im zweiten als auch im dritten Arbeitsspeicherabbild Artefakte gefunden. Davon waren die meisten bei der E-Mail-Adresse „computerforensikvl@gmail.com“ mit insgesamt 134 Suchtreffern vorhanden. Neuen dieser waren aus anderen Prozessen, die restlichen 125 waren direkt in einem Speicherbereich des Chrome-Browsers zu finden. Bei dieser wurde auch noch ein Artefakt im dritten RAM-Dump gefunden im Speicherbereich des dwm.exe Prozesses, welcher zuvor bereits angesprochen

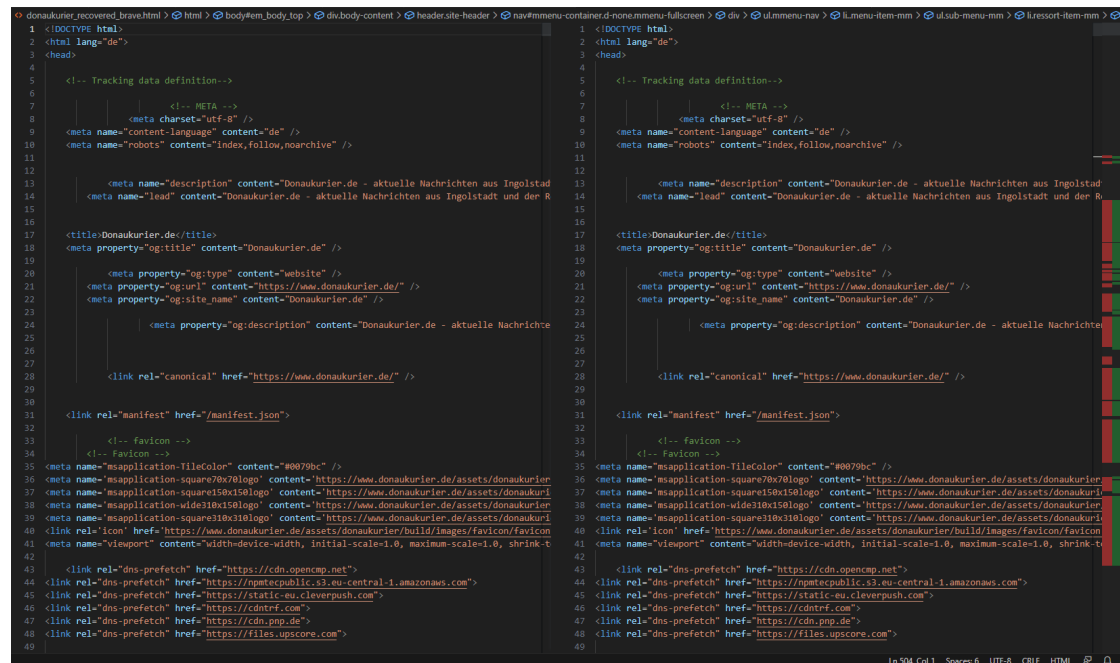


Abbildung 5.34.: Ausschnitt aus VS Code mit dem Vergleich der beiden Donaukurier Webseiten

wurde. Am wenigsten Suchtreffer hab es bei dem Betrefftext.

Yara-Regel „DK-Logo“ Auch bei Brave wurde wieder in den Arbeitsspeicherabbildern nach den Bytes des Donaukurier-Logos gesucht. Hier kam es zu drei Treffern im zweiten RAM-Dump, was in der Tabelle 5.10 zu erkennen ist.

Registry

Bei der Registry konnten weder in den Process Monitor Logs noch durch die Analyse der verschiedenen Hives Artefakte gefunden werden. Weitergehende Informationen zu den Schreiboperationen und der Stringsuche in den verschiedenen Hives ist in [link Anhang] zu finden.

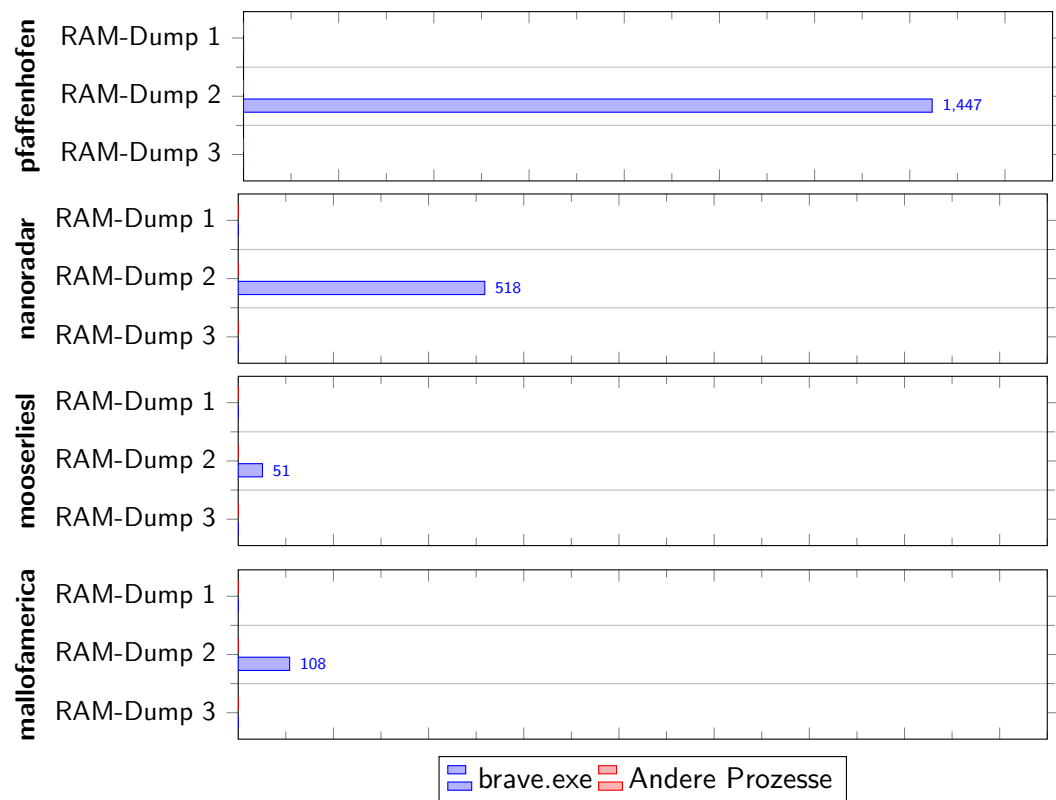


Tabelle 5.7.: Anzahl gefundener Suchbegriffe im Brave RAM

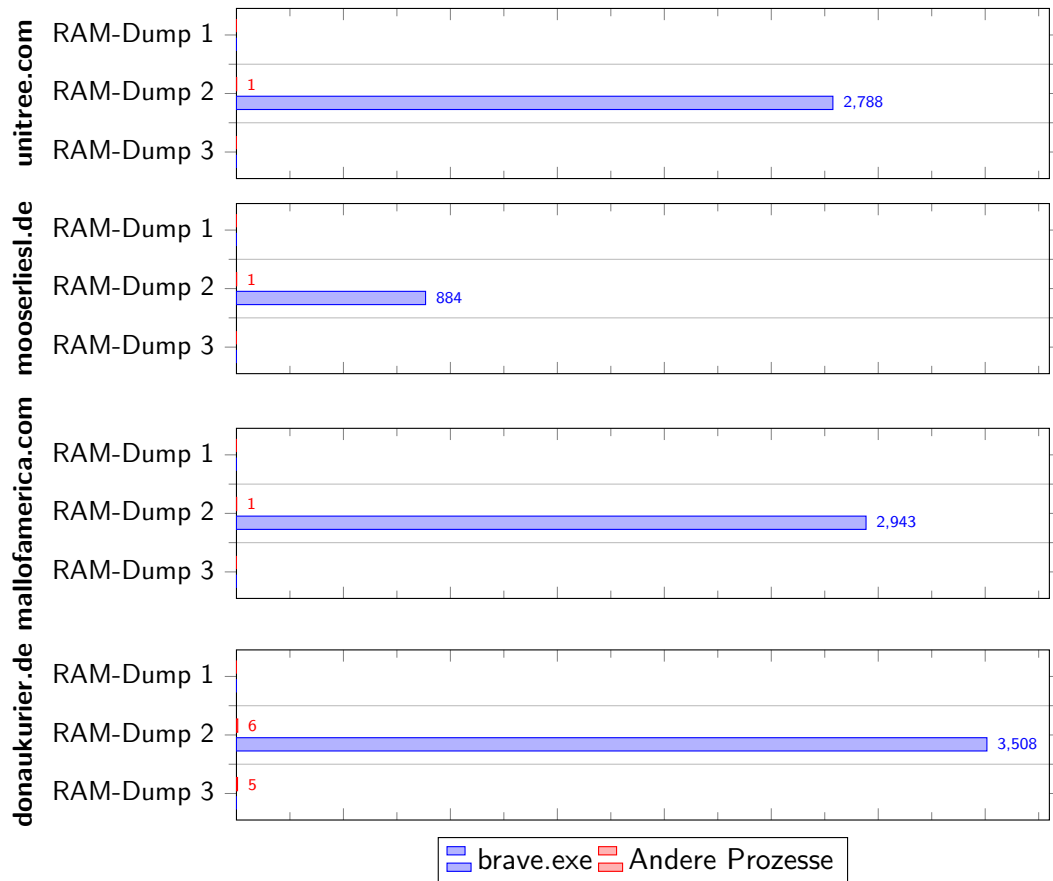


Tabelle 5.8.: Anzahl gefundener URLs im Brave RAM

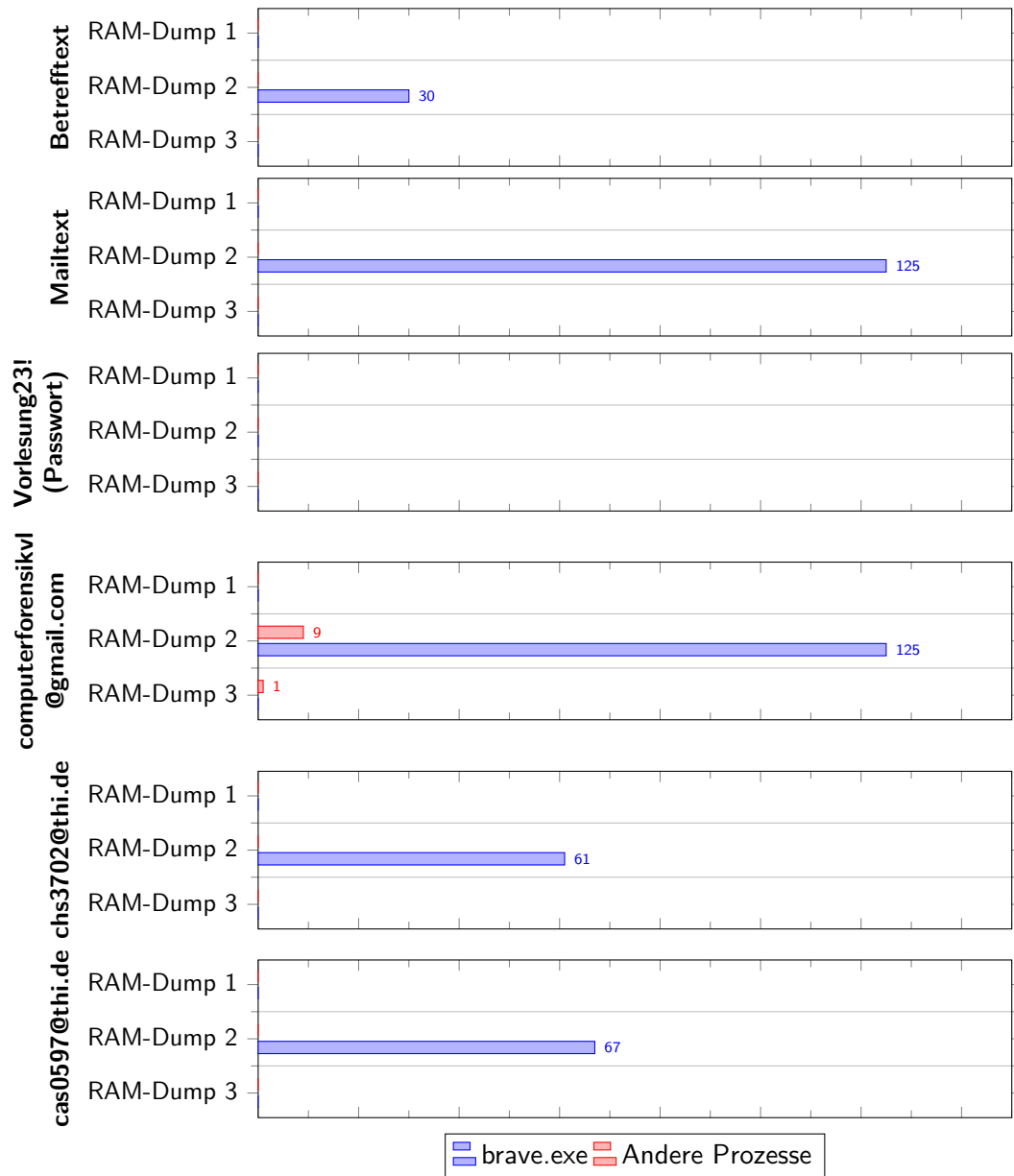


Tabelle 5.9.: Anzahl gefundener E-Mail Artefakte im Brave RAM

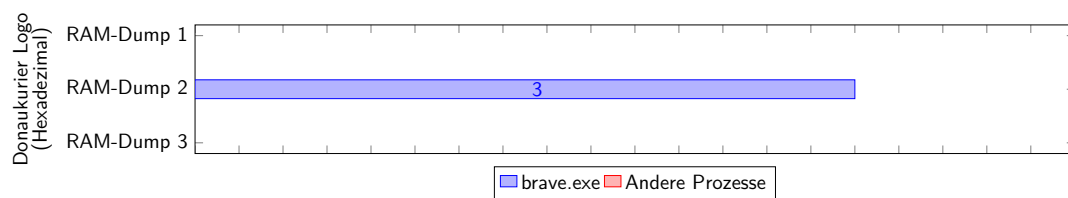


Tabelle 5.10.: Anzahl gefundener Hexadezimalwerte des Donaukurier-Logos im Brave RAM

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!

- Firefox v. Chrome (SStandardbrowser") - Tor v. Brave (SSichere Browser") - Zum Schluss: Eine große Tabelle"mit den wichtigsten Kategorien?

7. Diskussion

Zusammenfassend kann die Aussage getroffen werden, dass alle der vier betrachteten Browser ein gutes Ergebnis lieferten, da keine Browsing-Artefakte im nichtvolatilen, also persistenten Speicher identifiziert werden konnten. Jene konnten nur im RAM gefunden werden und teilweise im DNS-Cache, wobei diese auch durch das Leeren des Caches erfolgreich beseitigt werden konnten. Artefakte im DNS-Cache sind dabei eine bekannte Problematik. DNS-Anfragen des Browsers werden dabei von Betriebssystem darin gespeichert, unabhängig davon, ob der Browser im privaten Modus ist oder nicht [24]. Trotz der Bekanntheit dieser Schwachstelle besteht dieser immer noch in allen Browsern fort. Es wurden gezielt dafür auch schon Erweiterungen von Drittanbietern entwickelt, um dieses Problem des Speicherns von DNS-Anfragen zu cachern, jedoch wurde keine davon von den Browserherstellern übernommen [24].

Unserer Meinung nach ist der beste Browser, um anonym und sicher zu surfen, der Tor-Browser. Wie bereits im vorherigen Kapitel gezeigt, hinterlässt er am wenigsten Artefakte im RAM und somit am wenigsten Artefakte überhaupt, wie bereits öfter angesprochen wurde. Zusätzlich zum Schutz vor einem local attacker bietet Tor den Vorteil, dass man dadurch auch noch vor web attackern geschützt ist, da der Datenverkehr über mehrere Router, den sogenannten Nodes, umleitet und somit die Herkunft der Anfrage verschleiert. Somit ist Tor als Gesamtpaket gesehen ein sicherer und guter Browser, um sicher und anonym im Internet zu browsen.

Trotz der gerade angesprochenen Vorteile muss man jedoch auch die Nachteile von Tor sehen. Erstens ist das Browsen an sich meistens langsamer, da der Datenverkehr eben über mehrere Knoten geleitet wird. Zusätzlich wird dabei der Internetverkehr auch noch verschlüsselt. Beides ist positiv für die Privatsphäre, schränkt aber die User Experience ein [cite webpage]. Zusätzlich dazu kann es dazu kommen, dass Websites den Datenverkehr von Tor-Benutzern als potenzielles Sicherheitsrisiko ansehen und somit zusätzliche Captchas und Sicherheitsmaßnahmen einfügen, was wieder die User Experience einschränkt. Dazu kommt auch noch, dass man von der Infrastruktur des Tor-Netzwerkes angewiesen ist und es zu Verbindungsproblemen führen kann, wenn das Netzwerk gestört ist oder bestimmte nodes nicht verfügbar sind.

Schließlich ist noch die Frage zu klären, für welche Personen diese Analyse grundsätzlich durchgeführt wird bzw. wer von solch einer forensischen Analyse einen Nutzen zieht. Grundsätzlich wird eine solche forensische Analyse durchgeführt, um Schwächen in Browsern und den privaten Modi aufzudecken. Von diesen Ergebnissen profitieren zunächst einmal die

Browser-Entwickler, welche auf Basis der Ergebnisse das Ziel haben (sollten), die aufgedeckten Schwächen zu beseitigen, um ein privateres und sichereres Browsen zu ermöglichen. Dieses private Browsen kann dann entweder genutzt werden, um von „normalen“ Benutzern für legale Aktivitäten verwendet zu werden, oder von Kriminellen für illegale Tätigkeiten. Für diese wäre es dann sinnvoll, wenn der Browser möglichst wenige Artefakte verursacht, um möglichst unerkannt diesen Tätigkeiten nachzugehen. Dies würde es jedoch Forensikern wieder schwieriger machen, jene Tätigkeiten aufzudecken und nachzuweisen. Auch profitieren von solchen Analysen die Forensiker, welche auf Basis der Ergebnisse gezielt nach Artefakten suchen können und es somit leichter gemacht wird, kriminelle Tätigkeiten aufzudecken. Wie daraus ersichtlich wird, ist es nicht einfach zu sagen, wem eine solche forensische Analyse zunächst mehr nutzt und ob es gut oder schlecht ist, eine solche zu betreiben, da man nicht vorhersagen kann, ob die Ergebnisse zuerst gutartig zum besseren Aufdecken von illegalen Aktivitäten oder bössartig zum besseren Verschleiern von diesen verwendet wird.

Da jetzt alle Ergebnisse ausführlich ausgeführt wurden und noch weitergehende Fragestellungen angesprochen wurden, ist es nun noch wichtig anzusprechen, was man anschließend an diese Arbeit noch analysieren könnte, um die Browser-Forensik im Allgemeinen noch genauer zu betrachten bzw. um noch einen Scope, welcher für diese Arbeit getroffen wurde, erweitert werden. Hier wurde beim Process Monitor nur nach Prozessnamen, also nach Browser gefiltert. Eine weitere Möglichkeit wäre es, nach Speicherorten zu filtern, wie beispielsweise den browser-typischen common locations, und dann zu analysieren, welche Prozesse eine Schreibaktivität darin durchführen. Auch könnten noch weitere Aktivitäten des Betriebssystems näher betrachtet werden wie der DNS-Cache, ob es noch weitere Orte gibt, an welchem Windows hier noch Dateien anlegt oder bearbeitet, welche nicht von Browser bearbeitet werden. Auch eine Analyse der Browser für Mac oder Linux wäre wichtige Aufgabe, da sich die gesamte Literatur nur hauptsächlich mit Windows als Betriebssystem beschäftigt. Zusätzlich könnte man mehr auf den web attacker eingehen, was in dieser Arbeit komplett ausgeschlossen wurde. Ebenfalls könnten man einen direkten Vergleich mit den Aktivitäten der public-Modi ziehen, was aus zeitlichen Gründen und wegen des beschränkten Umfangs hier nicht mehr durchführbar war und den Rahmen dieser Arbeit gesprengt hätte. Zuletzt wäre es auch noch möglich, andere Browser wie Edge oder Safari dieser forensischen Analyse zu unterziehen.

8. Fazit

Da in den vorangegangenen Kapiteln ausführlich Begriffe definiert, die Methodik erläutert, die Ergebnisse präsentiert sowie vergleichen und weitergehende Punkte aufgeführt wurden, folgt abschließend für diese Arbeit ein kurzes persönliches Fazit.

Zunächst war es sehr informativ, über den allgemeinen Prozess bei einer forensischen Untersuchung mehr zu erfahren. Darunter zählt eine klar definierte Methodik, das Wissen über die verwendeten Tools sowie Schlussendlich die Durchführung der Analyse. Dabei war es stets wichtig, alles stets wie vorgeschrieben durchzuführen, um keine Ergebnisse in irgendeiner Art zu verfälschen. Auch die Verwendung von virtuellen Maschinen sowie Möglichkeiten, Speicherabbilder zu erstellen, war eine neue Erkenntnis. Ebenso wurden erstmals verschiedene Forensik-Tools wie Autopsy und Volatility verwendet sowie Möglichkeiten innerhalb dieser Tools benutzt, um Speicherabbilder mit unterschiedlichen Verfahren zu analysieren. Vor Allem beim Einlesen der Snapshots in Autopsy ist deutlich geworden, welcher zeitliche Aufwand hinter solch einer Analyse steckt, da selbst bei unserem Beispiel dieser Prozess ca. drei Stunden andauerte, wenn man alle Plugins in Autopsy verwenden möchte. Außerdem konnte neues Wissen im Bezug auf die Funktionsweise von Browsern erlangt werden, hier speziell über das Abspeichern von Informationen in bestimmten Dateien in speziellen Dateipfaden und deren Bedeutungen. Auch verschiedene Analyse-Tools wie ein Hex-Editor oder Database-Viewer wurden dafür benötigt und es konnte ein tieferes Verständnis für verschiedene Datentypen sowie Kodierungen gewonnen werden.

Abschließend war es eine sehr spannende und herausfordernde Arbeit, bei der man sehr viel über Browser, deren Funktionsweise und über Funktionen und Prozesse in Windows gelernt hat.

Anhänge

A. Yara-Regeln

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

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

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

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

rule html {
  strings:
```

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

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

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

B. Ausführliche Analyse: Firefox

B.1. Common Locations

Process Monitor WriteFile Operations

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

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

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

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

In Tabelle B.1 sind die Dateien je nach Speicherort *Local* (Hellblau) oder *Roaming* (Dunkelblau) entsprechend eingefärbt. Ausschließlich Dateien der Kategorie "Cache" sind im Local Pfad gespeichert.

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

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

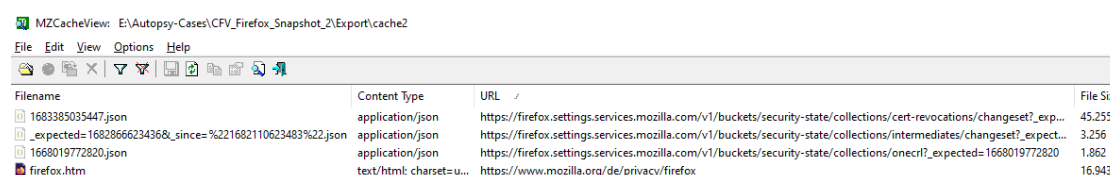
LOGFILE 1:

Kategorie	Dateiname	Dateistatus	Tool für Analyse	Enthaltene Artefakte
Cache	\cache2\entries\037778A55E1B7E9BED3390289866D09402D6C913	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
	\cache2\entries\1223A0378B8971FA4CD25EA1731C80B2B1676B42	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
	\cache2\entries\250EE2BC03AFF526F1A1C3DB212A79DE3EB60D5E	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
	\jumpListCache\ZKJGVJPzPe7w4w0KwEY0jw==.ico	Datei vorhanden	Windows Foto App	Keine PB Artefakte
	\cache2\entries\D16E4E5DFB15B4C8DE88842C05A47A07C611E01D	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
	\cache2\entries\2F040683A85A4372A73572713C6C52B510854566	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
Datareporting	\datareporting\glean\events\pageload	Datei vorhanden	HxD	Keine PB Artefakte
	\datareporting\glean\db\data.safe.tmp	Nicht-temp-Datei verwendet	HxD	Keine PB Artefakte
	\datareporting\glean\tmp\95ea3e10-e732-4642-8e92-515f4c4e090c	Datei nicht wiederherstellbar	N/A	N/A
	\datareporting\glean\tmp\16ab2ae2-c7b8-4390-a26e-7dcd95f5ff24	Datei nicht wiederherstellbar	N/A	N/A
Sessionstore	\sessionstore-backups\recovery.jsonlz4.tmp	Nicht-temp-Datei verwendet	dejsonlz4 + Notepad++	Keine PB Artefakte
Sonstige Dateien	\prefs-1.js	Datei vorhanden	HxD	Keine PB Artefakte
	\xulstore.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB Artefakte

LOGFILE 2:

Kategorie	Dateiname	Dateistatus	Tool für Analyse	Enthaltene Artefakte
Cache	\cache2\index.log	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
	\cache2\index	Datei vorhanden	MozillaCacheView	Keine PB Artefakte
Datareporting	\datareporting\glean\db\data.safe.tmp	Nicht-temp-Datei verwendet	HxD	Keine PB Artefakte
	\datareporting\archived\2023-05\1683405837882.9102466b-e465-4ecb-810f-74ae90c64c63.new-profile.jsonlz4.tmp	Nicht-temp-Datei verwendet	Session History Scrounger	Keine PB Artefakte
	\datareporting\archived\2023-05\1683405837905.86f4c992-6329-415b-8c29-911a2d4b7f9d.event.jsonlz4.tmp	Nicht-temp-Datei verwendet	Session History Scrounger	N/A
	\datareporting\archived\2023-05\1683405837939.abf8b065-41a4-4e94-a044-1cead61e396a.main.jsonlz4.tmp	Nicht-temp-Datei verwendet	Session History Scrounger	N/A
Sessionstore	\sessionstore.jsonlz4.tmp	Nicht-temp-Datei verwendet	dejsonlz4 + Notepad++	Keine PB Artefakte
Sonstige Dateien	\sessionCheckpoints.json.tmp	Nicht-temp-Datei verwendet	HxD	Keine PB Artefakte
	\prefs-1.js	Datei vorhanden	HxD	Keine PB Artefakte
	\xulstore.json.tmp	Nicht-temp-Datei verwendet	Notepad++	Keine PB Artefakte
	\saved-telemetry-pings\9102466b-e465-4ecb-810f-74ae90c64c63.tmp	Datei nicht wiederherstellbar	N/A	N/A
	\saved-telemetry-pings\86f4c992-6329-415b-8c29-911a2d4b7f9d.tmp	Datei nicht wiederherstellbar	N/A	N/A
	\saved-telemetry-pings\abf8b065-41a4-4e94-a044-1cead61e396a.tmp	Datei nicht wiederherstellbar	N/A	N/A
	\saved-telemetry-pings\35decee-d7c6-4820-a381-2dc89ff33c76.tmp	Datei nicht wiederherstellbar	N/A	N/A

Diese Dateien können mit dem Tool MZCacheView eingelesen werden. Wie in Abbildung B.1 gezeigt, konnten im Firefox Cache-Ordner des Festplatten-Images vom zweiten Snapshot drei JSON Dateien identifiziert werden. Dabei handelt es sich um Zertifikatsdateien, die von der *One Certificate Revocation List* stammen, ein Mechanismus von Firefox zur Überprüfung von Zertifikaten. In keinem der Zertifikate konnten mit HxD private Browsing Artefakte oder besuchte Seiten gefunden werden. [TechSupportGuy.05.06.2023] Weiterhin befindet sich im Cache das HTML-Dokument der Firefox Datenschutzseite, welche sich beim ersten Start des Browsers automatisch öffnete, siehe Kapitel ?? Weitere Cache Dateien konnten in keinem Festplatten-Image gefunden werden. Die Indexdatei \cache2\index dient als Datenbank



Filename	Content Type	URL	File Size
1683385035447.json	application/json	https://firefox.settings.services.mozilla.com/v1/buckets/security-state/collections/cert-revocations/changeset?_exp...	45.255
_expected=168286623436&_since=%221682110623483%22.json	application/json	https://firefox.settings.services.mozilla.com/v1/buckets/security-state/collections/intermediates/changeset?_expect...	3.256
1668019772820.json	application/json	https://firefox.settings.services.mozilla.com/v1/buckets/security-state/collections/onecrl?_expected=1668019772820	1.862
firefox.htm	text/html; charset=u...	https://www.mozilla.org/de/privacy/firefox	16.943

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

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

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

Datareporting Dateien im Ordner \datareporting\glean\db sind Teil des Glean-Systems, das für die Sammlung von Telemetriedaten und deren Übermittlung an Mozilla verwendet wird. [GitHub.05.06.2023] Die Datei data.safe.bin enthält verschlüsselte und anonyme Informationen über die Nutzung des Browsers. In HxD konnten keine PB Artefakte in den Dateien gefunden werden.

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

Sessionstore Die Datei \sessionstore-backups\recovery.jsonlz4 enthält eine Sicherungskopie der vorherigen Sitzung. Sie wird erstellt, wenn der Firefox-Browser nach einem Absturz oder einem unerwarteten Beenden neu gestartet wird. Jefferson Scher entwickelte das Online-Tool *Session History Scrounger for Firefox* zur Analyse dieser "Sessionstore-Backup"

Dateien. [JeffersonScher.29.11.2020] Wie Abbildung B.2 gezeigt, enthielt die Datei sowohl im Festplatten-Image 2 (Logfile 1) und 3 (Logfile 2) nur die automatisch geöffnete Seite der Firefox Datenschutzhinweise.

Closed Window 1



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

Sonstige Dateien In der Datei prefs-1.js werden benutzerspezifische Einstellungen und Konfigurationen für den Firefox-Browser gespeichert. Die Datei enthält Präferenzen des Benutzers in Form von JavaScript-Objekten. Es konnten in den Dateien beider Logfiles mit HxD keine PB Artefakte gefunden werden. Schließlich speichert die Datei xulstore.json benutzerspezifische Anpassungen und Konfigurationen des Firefox-Browsers. In der Datei konnten in den Festplatten-Images beider Logfiles mit Notepad++ keine PB Artefakte gefunden werden. [mozillazine.29.12.2022]

SQLite Datenbanken

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

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

Datenbank	Gespeicherte Daten [GitHub.08.04.2019]
places.sqlite	Informationen über Lesezeichen und Verlauf. Zu jeder besuchten Webseite: URL, Seitentitel, Zeitstempel des Besuchs etc.
cookies.sqlite	Von besuchten Webseiten verwendete Cookies.
storage.sqlite	Diverse Webdaten, z. B. Indexed-Datenbanken, Offline-Cache-Daten und andere lokale Speicherinformationen.
favicons.sqlite	Enthält Favicons (kleine Symbole in der Adressleiste) um besuchte Webseiten visuell zu identifizieren.
webappsstore.sqlite	Speichert Informationen über installierte Webanwendungen im Firefox-Browser, z.B. Berechtigungen und Einstellungen.
1657114595AmcateirvtiSty.sqlite	Datenspeicher für Activity Stream, eine personalisierte Übersicht über Browser-Aktivitäten beim Öffnen eines neuen Tabs.
3870112724rsegmnoittet-es.sqlite	Datenspeicher für Remote Settings, eine zentrale Verwaltung von benutzerspezifischen Browsereinstellungen.

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

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

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

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

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

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

Sowohl nachdem die VM heruntergefahren wurde, (Snapshot 4) als auch nach Durchführung der PRAGMA WAL Checkpoints, entstanden keine Änderungen in den SQLite Datenbanken. Somit wurden in den SQLite Datenbanken von Firefox keine zurückverfolgbaren PB Artefakte im privaten Modus hinterlassen.

Zusammenfassung Firefox Common Locations

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

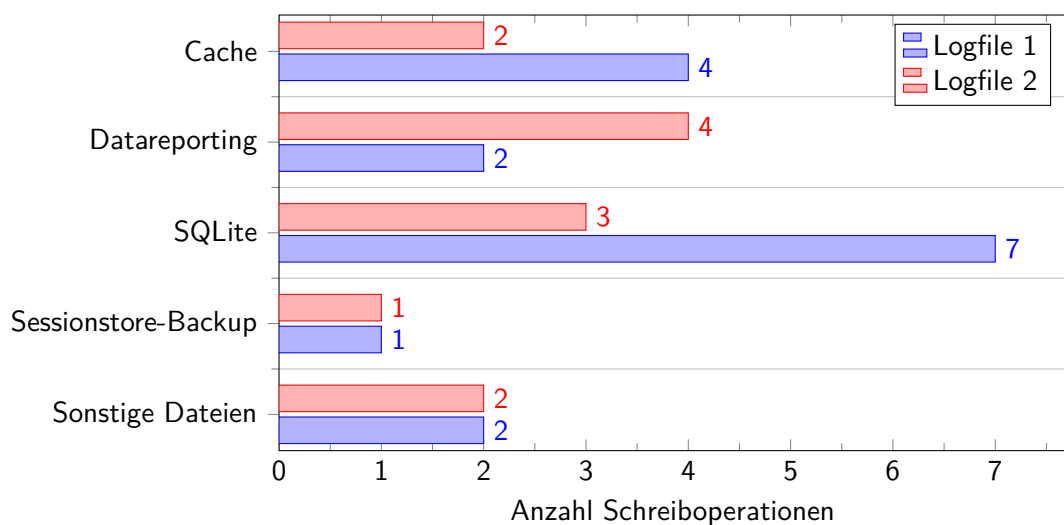


Abbildung B.3.: Firefox: Anzahl Schreiboperationen Logfile 1 vs Logfile 2, geordnet nach Kategorie

B.2. Uncommon Locations

Analyse mit Autopsy - Kategorisierte Dateien

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

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

Source Name	S	C	O	URL	Title	Date Created	Program Name	Domain	Data Source
Bing.url			19	http://go.microsoft.com/fwlink/?LinkId=255142	Bing.url	2023-04-25 16:09:28 MESZ	Internet Explorer Analyzer	microsoft.com	CFV_Firefox_Klon_Snapshot_3.img
places.sqlite			6	https://support.mozilla.org/products/firefox	Hilfe erhalten	2023-05-06 22:25:00 MESZ	Firefox Analyzer	mozilla.org	CFV_Firefox_Klon_Snapshot_3.img
places.sqlite			6	https://support.mozilla.org/kb/customize-firefox-controls-b...	Firefox anpassen	2023-05-06 22:25:00 MESZ	Firefox Analyzer	mozilla.org	CFV_Firefox_Klon_Snapshot_3.img
places.sqlite			6	https://www.mozilla.org/contribute/	Mitmachen	2023-05-06 22:25:00 MESZ	Firefox Analyzer	mozilla.org	CFV_Firefox_Klon_Snapshot_3.img
places.sqlite			6	https://www.mozilla.org/about/	Über uns	2023-05-06 22:25:00 MESZ	Firefox Analyzer	mozilla.org	CFV_Firefox_Klon_Snapshot_3.img
places.sqlite			6	https://www.mozilla.org/firefox/?utm_medium=firefox-des...	Erste Schritte	2023-05-06 22:25:01 MESZ	Firefox Analyzer	mozilla.org	CFV_Firefox_Klon_Snapshot_3.img

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

Web Cookies Die Kategorie “Web Cookies” enthält bereits vor Beginn des Browsing Szenarios zehn in Abbildung B.5 gezeigte Cookie-Einträge in der Datei WebCacheV01.dat. Dabei handelt es sich um eine Datenbank des Microsoft Edge Browsers zur Speicherung

von Nutzerdaten. Diese Datei verhält sich ähnlich wie die in diesem Versuch relevanten SQLite-Dateien. Bei den Einträgen handelt es sich um Cookies für Bing und die Outlook Webseite, obwohl diese Seiten nie in Microsoft Edge geöffnet wurden. In den Snapshots 2 – 4 kamen keine weiteren Einträge in dieser Kategorie hinzu.

Source Name	S	C	O	URL	Date Accessed	Name	Value	Program Name	Domain	Data Source
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:17 MESZ	SUID	M	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	www.bing.com	2023-05-06 19:51:24 MESZ	MUIDB	31708C5FC3CF47068AFADC1CB47D0111	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:17 MESZ	SRCHD	AF=NOFORM	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:17 MESZ	SRCHUID	V=28;GUID=62C50ADBCEB84234A9FE14D681DCB91D8dm...	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:17 MESZ	SRCHUSR	DOB=20230506	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:20 MESZ	SRCHHPGLSR	SRCHLANG=de&LUT=16834026192238JPMH=des204058J...	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:50:19 MESZ	CortanaAppUID	C164AA3A4D47E127DDC66AD915CFD04C	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	bing.com	2023-05-06 19:55:22 MESZ	ANON	A=A385B679A14D5980AA02763FFFFFFFFFF	Microsoft Edge Analyzer	bing.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	live.com	2023-05-06 19:50:30 MESZ	MUID	118A534093A962628C5404997A96688	Microsoft Edge Analyzer	live.com	CPV_Firefox_Klon_Snapshot_3.img
WebCacheV01.dat			15	login.live.com	2023-05-06 19:51:06 MESZ	Host-MSAAUTHP		Microsoft Edge Analyzer	live.com	CPV_Firefox_Klon_Snapshot_3.img

Abbildung B.5.: Firefox: Von Autopsy als “Web Cookies” kategorisierte Dateien

Web History Die Kategorie “Web History” listet alle Dateien mit gespeichertem Suchverlauf auf. Vor Beginn des Browsing Szenarios (Snapshot 1) enthält die Kategorie zwei Einträge zur Outlook Webseite in der Datei WebCacheV01.dat. Nach Durchführung des Browsing Szenarios (Snapshot 2) wurde ein Eintrag in der places.sqlite Datenbank hinzugefügt. Dabei handelt es sich um die automatisch im normalen Browsingmodus geöffnete Firefox-Standardseite über Datenschutzhinweise. Dies deckt sich mit den Beobachtungen der Common Locations in Anhang B.1. Darüber hinaus enthält dieser Snapshot für die Datei WebCacheV01.dat den Eintrag file:///Z:/Logfile_1. Dabei handelt es sich um das Process Monitor Logfile, das gemäß Methodik in Kapitel ?? über den gemeinsamen VM-Ordner zum Analyse-Rechner transportiert wurde. Ergänzt wird die Kategorie nach Schließen des Browsers (Snapshot 3) durch den Eintrag file:///Z:/Logfile_2, dem zweiten Process Monitor Logfile. Nach Herunterfahren der virtuellen Maschine (Snapshot 4) werden in dieser Kategorie keine neuen Dateien erfasst. Die kategorisierten Dateien sind in Abbildung B.6 dargestellt.

Source Name	S	C	O	URL	Date Accessed	Referrer URL	Title	Program Name	Domain	Data Source	Username
places.sqlite			6	https://www.mozilla.org/de/privacy/firefox/	2023-05-06 22:25:00 MESZ	https://www.mozilla.org/privacy/firefox/	Firefox Datenschutzhinweis — Mozilla	Firefox Analyzer	mozilla.org	CPV_Firefox_Klon_Snapshot_3.img	
WebCacheV01.dat			15	https://login.live.com/oauth20_desktop.srf?lc=1031	2023-05-06 19:51:06 MESZ			Microsoft Edge Analyzer	live.com	CPV_Firefox_Klon_Snapshot_3.img	Forensik
WebCacheV01.dat			15	https://login.live.com/oauth20_authors.srf?client_id=...	2023-05-06 19:51:08 MESZ			Microsoft Edge Analyzer	live.com	CPV_Firefox_Klon_Snapshot_3.img	Forensik
WebCacheV01.dat				file:///Z:/Logfile_1	2023-05-06 20:29:36 MESZ			Microsoft Edge Analyzer		CPV_Firefox_Klon_Snapshot_3.img	Forensik
WebCacheV01.dat				file:///Z:/Logfile_2	2023-05-06 20:44:19 MESZ			Microsoft Edge Analyzer		CPV_Firefox_Klon_Snapshot_3.img	Forensik

Abbildung B.6.: Firefox: Von Autopsy als “Web History” kategorisierte Dateien

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

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

Abbildung B.7.: Firefox: Von Autopsy als “Web Categories” kategorisierte Dateien

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

Registry

Process Monitor SetValue Operations

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

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

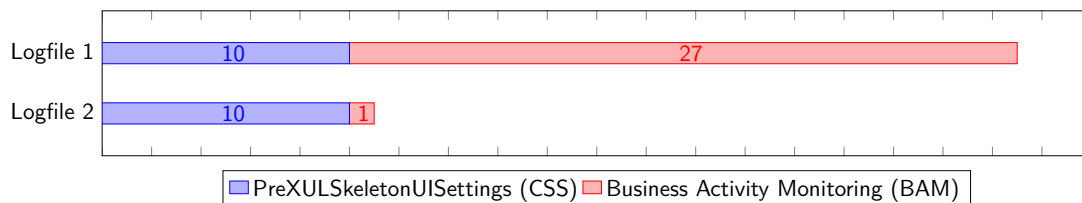


Abbildung B.8.: Firefox Registry “SetValue” Operationen in den Process Monitor Logfiles 1 und 2

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

|<Skeleton UI Setting>. Somit enthält der Key den absoluten Installationspfad von Firefox gefolgt von einer Skeleton UI Einstellung. Nachfolgend sind alle möglichen UI Einstellungen aufgelistet, gefolgt vom Datentyp des Keys. [Mills.2021]

- ScreenX (DWORD)
- ScreenY (DWORD)
- Width (DWORD)
- Height (DWORD)
- Maximized (DWORD)
- Flags (DWORD)
- CssToDevPixelScaling (REG_BINARY)
- UrlbarCSSSpan (REG_BINARY)
- SearchbarCSSSpan (REG_BINARY)
- SpringsCSSSpan (REG_BINARY)

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

Business Activity Monitoring *Business Activity Monitoring*, kurz *BAM* ist eine weitgehend undokumentierte Windows Funktion, die im Hintergrund ausgeführte Programme steuert. Der Registry Key hat das Format HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\<SID>\Device\HarddiskVolume2\<Absoluter Firefox Installationspfad>\firefox.exe und den Datentyp REG_BINARY. Jeder Schlüssel wird durch die Sicherheits-ID (SID) des Benutzers identifiziert. Ein BAM Registry Key schreibt für alle ausgeführten Programme – hier Firefox – den Zeitstempel der letzten Ausführung. PB Artefakte sind dabei nicht enthalten. [MandiOhlinger.05.06.2023, InfoSecNotes.05.06.2023]

Stringsuche in Registry Hives

Gemäß Methodik in Kapitel ?? 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 ?? aus jedem Festplatten-Image extrahiert und mithilfe des Registry Explorers nach PB Artefakten durchsucht. Dabei wurde in keinem Snapshot in keinem Hive ein PB Artefakt gefunden.

C. Ausführliche Analyse: Tor

C.1. Common Locations

Process Monitor WriteFile Operations

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

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

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

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

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

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

Cache Der Tor-Browser schreibt eine einzige Cache-Datei \Caches\profile.default\startupCache\startupCache.8.little im Caches-Pfad. Alle anderen geschriebenen Dateien befinden sich im Profile.default-Pfad. Die Datei "startupCache.8.little" ist eine interne Datei, welche erstellt wird, um den Startvorgang des Browsers zu beschleunigen. Sie enthält Informationen über bereits geladene Browser-Komponenten wie JavaScript-Code, CSS-Dateien, Bilder und andere Ressourcen. [MozillaWiki.05.06.2023] Bei Untersuchung mit HxD konnten keine PB Artefakte gefunden werden.

Tabelle C.1.: Tor alle "WriteFile"-Operationen der Logfiles 1, 2-1 und 2-2

LOGFILE 1:

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

LOGFILE 2-1

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

LOGFILE 2-2

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

Datareporting Im "Datareporting"-Ordner wird vom Tor-Browser die Datei `\datareporting\data.safe.bin` geschrieben. Sie enthält verschlüsselte und anonyme *Glean* Informationen über die Nutzung des Browsers. [GitHub.05.06.2023b] 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. [GitHub.08.04.2019] Eine Analyse mit Notepad++ und dem JSON-Plugin brachte keine PB-Artefakte.

Sonstige Dateien Die im ersten Logfile geschriebene Datei `AlternateServices.txt` enthält .onion-URLs des HTTP Alternative Services. Dieser Mechanismus ermöglicht es Servern, Clients mitzuteilen, dass der Dienst, auf den sie zugreifen, an einem anderen Netzwerkstandort oder über ein anderes Protokoll verfügbar ist. Die Datei speichert diese Zuordnung. [MozillaSupport.26.10.2020]

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

Die geschriebene Datei `onion-aliases.json` enthält *SecureDrop* Adressen, beispielsweise für die Süddeutsche Zeitung. SecureDrop ist ein Open-Source-Softwaretool, das von Journalisten und Nachrichtenorganisationen verwendet wird, um anonyme Informationen von Whistleblowern entgegenzunehmen. Es ermöglicht den sicheren Austausch von Informationen, ohne die Identität der Quelle preiszugeben. Whistleblower können über .onion-URLs auf die SecureDrop-Websites zugreifen und vertrauliche Dokumente oder Nachrichten sicher und anonym übermitteln. [SecureDrop.05.06.2023]

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. [Gitlab.05.06.2023] In der aktuellen Tor-Browser-Version konnten keine private Browsing Artefakte gefunden werden.

SQLite Datenbanken

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

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

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

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

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

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

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

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

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

Zusammenfassung Tor Common Locations

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

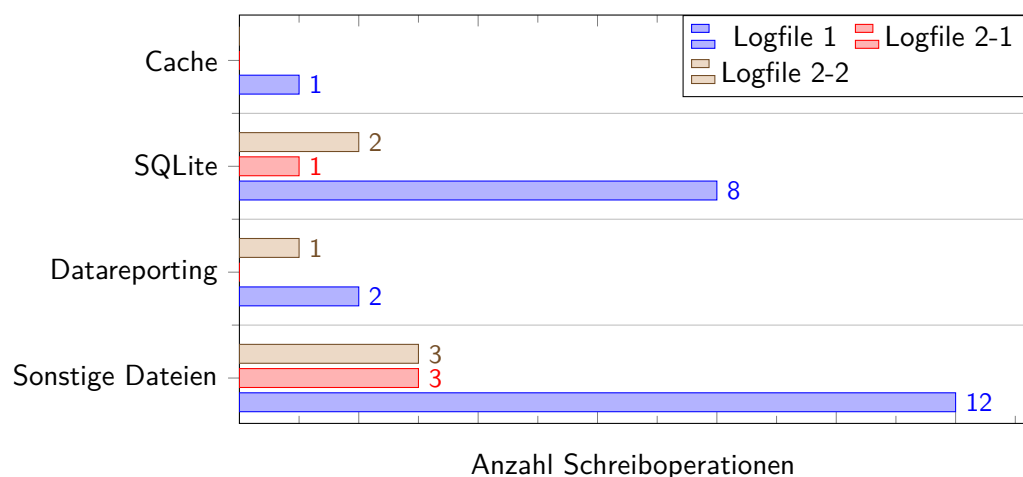


Abbildung C.1.: Tor: Anzahl Schreiboperationen Logfile 1 vs Logfile 2-1 vs Logfile 2-2, geordnet nach Kategorie

C.2. Uncommon Locations

Analyse mit Autopsy - Kategorisierte Dateien

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

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

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

Abbildung C.2.: Tor: Von Autopsy als "Web Bookmarks" kategorisierte Dateien

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

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

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

Abbildung C.3.: Tor: Von Autopsy als "Web Cookies" kategorisierte Dateien

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

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

Abbildung C.4.: Tor: Von Autopsy als "Web History" kategorisierte Dateien

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

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

Abbildung C.5.: Tor: Von Autopsy als "Web Categories" kategorisierte Dateien

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

C.3. Registry

Process Monitor SetValue Operations

Bei Betrachtung als Common Locations werden gemäß Methodik in Kapitel ?? alle "SetValue" Schreiboperationen in den Process Monitor Logfiles für die Prozesse "tor.exe" und "firefox.exe"

untersucht.

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

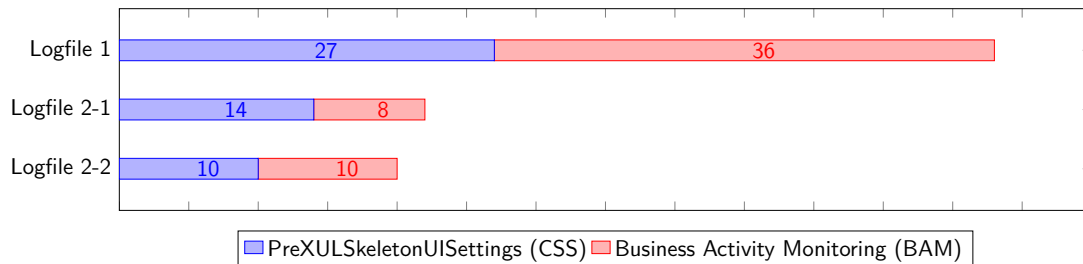


Abbildung C.6.: Tor Registry "SetValue" Operationen in den Process Monitor Logfiles 1, 2-1 und 2-2

Stringsuche in Registry Hives

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

D. Ausführliche Analyse: Chrome

D.1. Common Locations

Process Monitor WriteFile Operations

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

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

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

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

LOGFILE 2:					
Kategorie	Dateiname	Dateistatus	Verwendetes Tool zur Analyse	Enthaltene Artefakte	
<i>settings.dat</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Crashpad\settings.dat	Datei vorhanden	HxD	Keine PB Artefakte	
<i>Temp files (.tmp)</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\35debf2e-9a97-4829-b0d1-2c6efb7246bc.tmp	Datei nicht wiederherstellbar	N/A	N/A	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\4dce7d9d-2753-424d-ad13-eb84e1ea9326.tmp	Datei nicht wiederherstellbar	N/A	N/A	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\51ff0ac1-e188-4c8a-8b3d-891f326bb890.tmp	Datei nicht wiederherstellbar	N/A	N/A	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\52837c44-01bc-43d1-b859-0fe50c823372.tmp	Datei nicht wiederherstellbar	N/A	N/A	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Storage\ext\nmmhkkccagldldgiimedpiccmgmieda\def\Network\c280cbe5-825f-482f-8c5f-e4b0f0e8d560.tmp	Datei vorhanden	HxD	Keine PB Artefakte	
<i>SQLite -journal Dateien</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\History-journal	Datei leer (0 Bytes groß)	N/A	N/A	
<i>JSON-Datei</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Variations	Datei vorhanden	HxD	Keine PB Artefakte	
<i>000003.log Datei (LevelDB)</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Session Storage\000003.log	Datei vorhanden	HxD	Keine PB Artefakte	
<i>data_1 files</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\Default\Cache\Cache_Data\data_1	Datei vorhanden	HxD	Keine PB Artefakte	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\GrShaderCache\data_1	Datei vorhanden	HxD	Keine PB Artefakte	
	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\ShaderCache\data_1	Datei vorhanden	HxD	Keine PB Artefakte	
<i>shutdown_ms.txt</i>	C:\Users\Forensik\AppData\Local\Google\Chrome\User Data\chrome_shutdown_ms.txt	Datei vorhanden	HxD	Keine PB Artefakte	

Dabei wurde in Tabelle D.1 unterschieden in den **Local**-Browser Pfad (hellblau) und in andere Pfade (rot).

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

Alle *.tmp*-Dateien waren nicht wiederherstellbar, was eine Analyse unmöglich macht. Die Dateien mit dem Namen *data_1* waren jeweils vorhanden, enthielten jedoch keine PB Artefakte nach ausführlicher Analyse. Die *-journal*-Dateien Diese sind sogenannte „Rollback Journals“ und sind relevant für atomare Commit- und Rollback-Funktionen [25]. Da diese Dateien jedoch alle 0 Bytes groß waren, konnte hier kein Artefakt gefunden werden.

Auch wurden noch *000003.log*-Dateien geschrieben. Diese sind Teil eines LevelDB-Schlüsselwertespeichers [6, 8] . Mithilfe von HxD und einer Analyse mittels eines selbst erstellten Python-Scripts zur Ausgabe der Schlüssel-Werte-Paare derjenigen Ordner, welche die *.log* und weitere LevelDB spezifische Dateien enthalten, kann auch in diesen Ordner bzw. Dateien auch keine Artefakte gefunden werden.

Wie in Tabelle D.1 bereits dargestellt wurde, konnten auch keine temporären Icons wiederhergestellt werden. Diese waren weder in Autopsy im Snapshot oder im Arbeitsspeicher auffindbar.

Weiterhin wurden Dateien geschrieben, welche zum Microsoft Windows Wörterbuch (dictionary) zuzuordnen sind. Diese Dateien sorgen dafür, dass systemweit gewisse Wörter zu einer Sprache hinzugefügt werden oder Ausnahmen für bestimmte Begriffe gemacht werden [27]. Da alle drei Dateien hier den Inhalt „FF FE“ aufwiesen, konnte hier auch kein Artefakt gefunden werden.

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

Anschließend folgt die Analyse des zweiten Logfiles.

Die Datei *settings.dat* im Crashpad Ordner ist Teil der Crashpad library, welche den Maschinen- und Programmzustand im Falle eines Prozessabsturzes erfasst und einen Absturzbericht an einen Backend-Server übermittelt [4]. Die (binäre) Datei an sich beinhaltet dabei die Einstellungen für die Bibliothek [4]. Diese Datei beinhaltet jedoch keine Browsing-Artefakte.

Weiterhin wurden wieder temporäre Dateien geschrieben, welche bis auf eine Datei nicht rekonstruierbar sind. Diese Eine enthält jedoch keine Artefakte.

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

Auch wird wieder eine *000003.log*-Datei geschrieben, welche aber wieder keine Artefakte enthält.

Gleiches gilt für die geschriebenen *data_1*-Dateien.

Zuletzt wurde noch eine Textdatei namens *chrome_shotdown_ms.txt* geschrieben. Diese enthält die Zeit in Millisekunden, welche Chrome für das Schließen benötigt [3]. Dort fanden sich neben der Zeit in Millisekunden keine weiteren Artefakte.

Cache

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

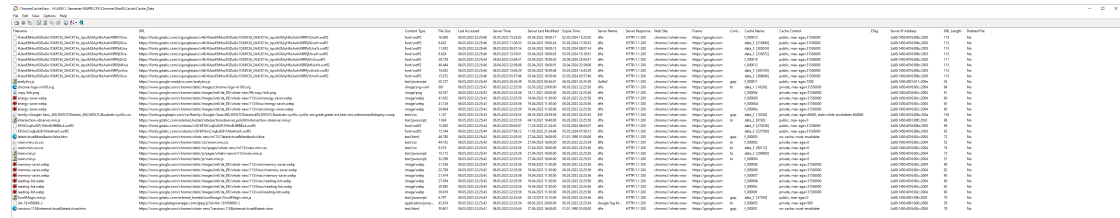


Abbildung D.1.: Chrome Cache Analyse mit ChromeCacheView

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

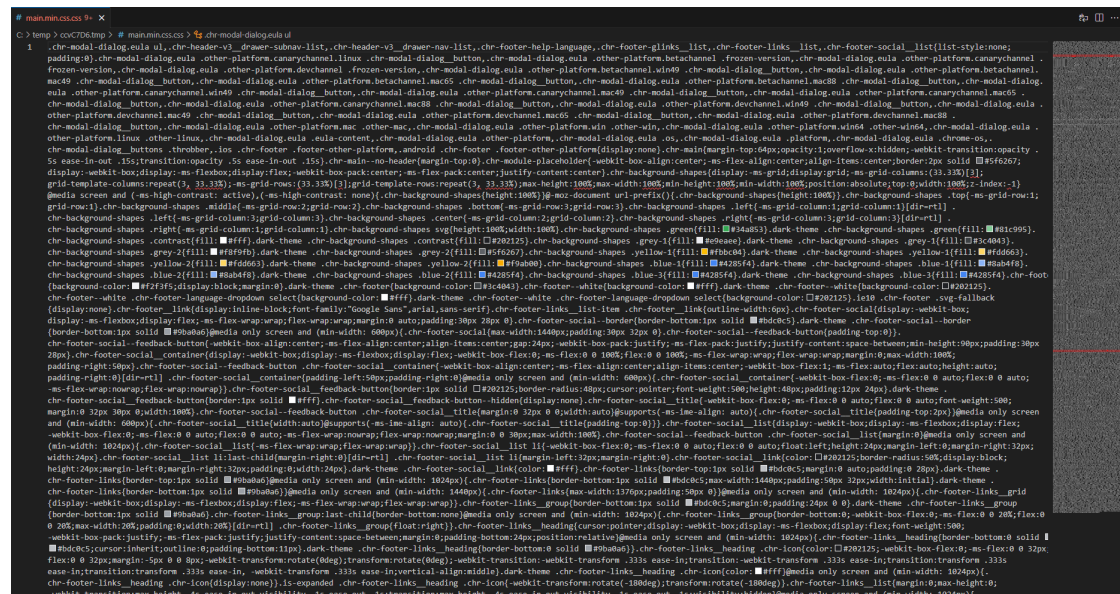


Abbildung D.2.: Analyse eines Chrome Cache-files

SQLite Datenbanken

Wie in [link] erwähnt, werden SQLite Datenbanken als Datenstrukturen für Nutzerdaten detailliert und getrennt von den Schreiboperationen der Process Logfiles untersucht. Bei Chrome gibt es im Vergleich zu Firefox und Tor den Unterschied, dass die Datenbanken nicht

direkt als .sqlite Dateien erkennbar sind, sondern als Dateien ohne bestimmte Endung im Browser-Verzeichnis vorliegen. Alle relevanten Datenbanken zeigt [\[link\]](#).

Auch hier wurde jede Datenbank aus den Festplatten-Speicherabbildern extrahiert (welche vorhanden waren) und diese anschließend im Hinblick auf deren Inhalte, Tabellen und Schreiboperationen verglichen und analysiert. [\[link\]](#) zeigt davon eine Zusammenfassung.

Erklärung hier...

Hier wurden zusätzlich die Tools „ChromeHistoryView“ und „ChromeCookiesView“ zur Analyse verwendet. Da diese jedoch nur die Datenbanken und darin enthaltenen Tabellen auslesen, konnte auch mithilfe dieser Programme keine weiteren Artefakte festgestellt werden.

Registry

Zusätzlich zu Schreiboperationen und Datenbanken wurden auch noch die Logfiles des Process Monitors nach *RegSetValue*-Operationen gefiltert. Die darin enthaltenen Schlüssel wurden dann analysiert. Dabei war ein Teil der Schlüssel direkt Strings, welche man somit direkt ablesen konnte. Alle anderen Werte wurden zusätzlich zu Zeichenketten umgewandelt, um sicher keine Artefakte zu übersehen. Hier konnte aber auch kein Hinweis auf das PB Szenario gefunden werden.

Zusammenfassend, wurden also in allen Common Locations keine PB Artefakte hinterlassen.

Uncommon Locations

Analyse mit Autopsy - Kategorisierte Dateien

Wie in [\[link hoch\]](#) bereits schon ausführlicher dargestellt, wurden auch für Chrome alle von Autopsy kategorisierten Dateien betrachtet. Auch diese Analyse brachte keine Artefakte hervor.

Stringsuche in Registry Hives

Zusätzlich zu den Analyse der kategorisierten Dateien wurden auch noch alle in [\[Tabelle link\]](#) aufgelisteten Registry Hives in den Registry Explorer eingelesen und dann sowohl nach URLs, Keywords, HTML-Fragmente und E-Mail Artefakten gesucht. Abbildung D.3 zeigt einen Screenshot davon. Im Suchfenster unten ist zu sehen, dass hier keine Strings gefunden wurden. Exemplarisch ist hier das Registry-Explorer-Projekt zum vierten Snapshot gezeigt. Auch die Hives aller anderen Speicherabbilder zeigten hier keine Treffer. Somit sind keine Artefakte in der Registry auffindbar.

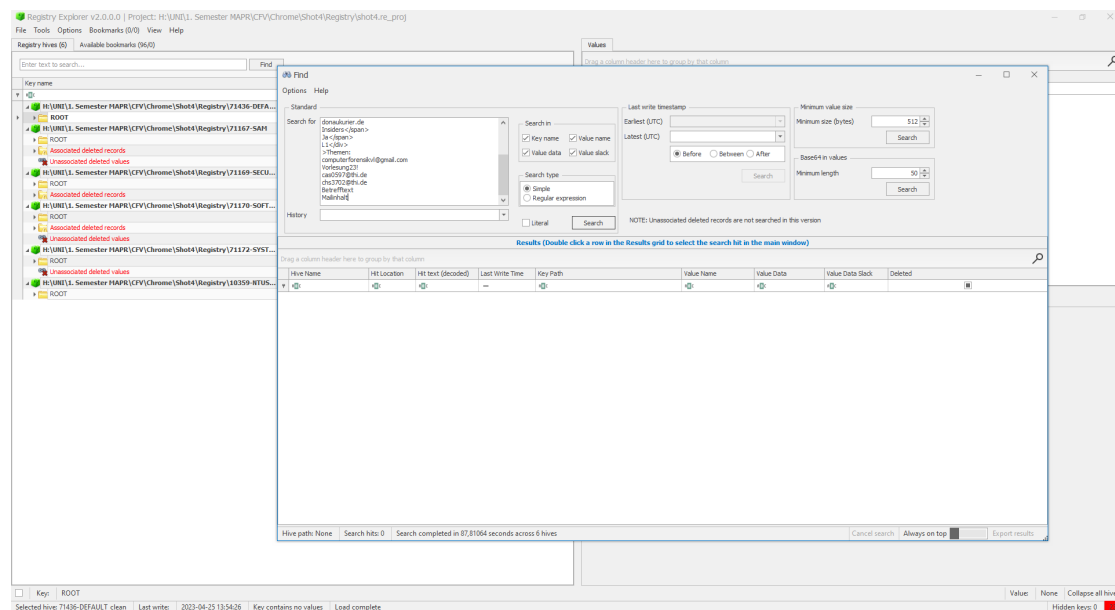


Abbildung D.3.: Stringsuche in den Hives mittels des Registry Explorers

E. Ausführliche Analyse: Brave

E.1. Common Locations

Process Monitor WriteFile Operations

Cache

SQLite Datenbanken

Registry

Uncommon Locations

Analyse mit Autopsy - Kategorisierte Dateien

Stringsuche in Registry Hives

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] Chromium. *Contents of /trunk/src/chrome/browser/browser_shutdown.cc*. Zuletzt zugegriffen am 28.05.2023 um 16:34 Uhr. 2009.
URL: https://src.chromium.org/viewvc/chrome/trunk/src/chrome/browser/browser_shutdown.cc?pathrev=26058.
- [4] Chromium. *Crashpad Overview Design*. Zuletzt zugegriffen am 28.05.2023 um 21:58 Uhr. URL: https://chromium.googlesource.com/crashpad/crashpad/+HEAD/doc/overview_design.md.
- [5] Divya Dayalamurthy. "Forensic memory dump analysis and recovery of the artefacts of using tor bundle browser—the need". In: (2013).
- [6] Google. *leveldb*. Zuletzt zugegriffen am 28.05.2023 um 22:20 Uhr. 2023.
URL: <https://github.com/google/leveldb>.
- [7] Ms Pooja Gupta. "Capturing Ephemeral Evidence Using Live Forensics". In: *IOSR J. Electron. Commun. Eng* (2013), S. 109–113.
- [8] *Hang on! That's not SQLite! Chrome, Electron and LevelDB*. Zuletzt zugegriffen am 28.05.2023 um 22:21 Uhr. Okt. 2020.
URL: <https://www.cclsolutionsgroup.com/post/hang-on-thats-not-sqlite-chrome-electron-and-leveldb>.
- [9] 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.
- [10] Graeme Horsman u. a. "A forensic examination of web browser privacy-modes". In: *Forensic Science International: Reports* 1 (2019), S. 100036.
- [11] Windows OS Hub. *Memory Compression Process: High Memory and CPU Usage in Windows 10 and 11*. Zuletzt zugegriffen am 08.06.2023 um 11:51 Uhr. Jan. 2022. URL: <https://woshub.com/memory-compression-process-high-usage-windows-10/>.

- [12] 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.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] Matt Muir, Petra Leimich und William J Buchanan. "A forensic audit of the tor browser bundle". In: *Digital Investigation* 29 (2019), S. 118–128.
- [17] 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.
- [18] PCTipps. *Dwm.exe: Was ist der Desktop Window Manager?*
<https://pctipps.de/dwm-exe/>. Zuletzt zugegriffen am 08.06.2023 um 12:29 Uhr.
- [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] Denis Pogonin und Igor Korkin. *Microsoft Defender Will Be Defended: MemoryRanger Prevents Blinding Windows AV*. 2022. DOI: <https://doi.org/10.48550/arXiv.2210.02821>.
arXiv: 2210.02821 [cs.CR].
- [21] Digvijaysinh Rathod. "Darknet forensics". In: *future* 11 (2017), S. 12.
- [22] 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.
- [23] Huwida Said u. a. "Forensic analysis of private browsing artifacts". In: *2011 International Conference on Innovations in Information Technology*. IEEE. 2011, S. 197–202.

-
- [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] SQLite. *Temporary Files Used By SQLite*.
Zuletzt zugegriffen am 28.05.2023 um 22:10 Uhr. Jan. 2022.
URL: <https://www.sqlite.org/tempfiles.html>.
- [26] Sylie. *Lösung: Shell Infrastructure Host hohe CPU-Auslastung [MiniTool]*.
<https://de.minitool.com/nachrichten/shell-infrastructure-host-hohe-cpu-auslastung.html>. Zuletzt zugegriffen am 08.06.2023 um 12:20 Uhr. Feb. 2023.
- [27] Sergey Tkachenko. *Add or Remove Words in Spell Checking Dictionary in Windows 10*.
Zuletzt zugegriffen am 28.05.2023 um 21:59 Uhr. Jan. 2018. URL:
<https://winaero.com/words-spell-checking-dictionary-windows-10/>.
- [28] 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.