A Forensic Audit of the Tor Browser Bundle

Matt Muir, Petra Leimich and William J Buchanan

School of Computing, Edinburgh Napier University

Abstract

The increasing use of encrypted data within file storage and in network communications leaves investigators with many challenges. One of the most challenging is the Tor protocol, as its main focus is to protect the privacy of the user, in both its local footprint within a host and over a network connection. The Tor browser, though, can leave behind digital artefacts which can be used by an investigator. This paper outlines an experimental methodology and provides results for evidence trails which can be used within real-life investigations.

Keywords: Digital forensics, Tor, Windows 10

1. Introduction

Use of the Tor Browser and network became mainstream in 2013, after the arrest and conviction of Ross Ulbricht - the alleged mastermind behind darknet marketplace Silk Road (Dewey, 2013). Prior to Ulbricht's arrest, a former American National Security Agency (NSA) contractor, Edward Snowden, had leaked classified files to The Guardian newspaper concerning the extent of mass surveillance by global intelligence services (Greenwald et al., 2013). This information, later known as the Snowden revelations, ignited a global debate surrounding the lack of privacy offered by digital communication.

Motivated by fear of privacy-intrusion, members of the public and cyber criminals alike increasingly turned to the Tor Browser, tempted by its anonymising features. A new influx of users and increasing media attention catalysed academic research into the effectiveness of Tor and its ability to protect user privacy. The majority of this research concentrated on improvements to the performance and scalability of the network, as well as the investigation of remote attacks intended to unmask users (Nordberg, 2015). However, the question is whether anonymity is maintained where an adversary is local, rather than remote?

This paper assesses the effectiveness of the Tor Browser in protecting the user against such an adversary by conducting a forensic analysis of the software and its interaction with the host operating system (OS). Our contribution is threefold:

- We show that artefacts proving installation/use of the browser are generated in memory and on disk in the form of default bookmarks. These artefacts are attributable to a particular user, uniquely identify the Tor Browser, and persist through uninstallation and logout.

- Furthermore, user activity within Tor is written to the Windows Registry as a consequence of recent updates to Windows 10. This allows a forensic adversary to determine the titles of pages visited using the browser.

- From the results of our forensic audit, we devise a forensic methodology.

Our approach simulates typical web browsing activity with Tor. Using virtualisation and a pre-determined browsing protocol allows artefact recovery with static and live forensic techniques, such as process monitoring, keyword searching and file carving, with the aid of Autopsy and the Volatility Framework.

Static analysis reveals significant leakage of user activity in the snapshots of machines used to perform the testing. This includes HTTP header information, web page titles and an instance of a URL. Further, live analysis identified traces of Tor processes even after the user had closed and uninstalled the browser and logged out. The absolute path to the browser executable was seen in RAM on several occasions, including the username of the user running the browser and the device from which it was run.

As a result, Tor is easily identified, cannot be securely deleted and activity from within a browsing session is determinable, suggesting that defence against a local adversary has not been fully addressed by the developers. This finding significantly expands on previous research, where artefacts pertaining to use of the browser, but not browsing history, were found.

2. Background

The novel approach to anonymous networking which Tor is (in)famous for has attracted interest from the cybersecurity community, largely resulting in research concerning deanonymisation of users from a network perspective. However, it is the intention of the Tor Project to protect the user from both network and local adversaries. This is achieved through the implementation of features and design choices intended to obfuscate network activity and employ anti-forensics techniques to prevent browsing session data from being written to disk (Perry et al., 2018).

2.1. Tor Overview

Tor, an acronym for *The Onion Router*, refers to both a browser and a networking protocol developed by The Tor Project. The purpose of the Tor network is to mask the original source of Internet traffic by redirecting data from source to destination through a series of randomly chosen, volunteer-run servers (Torproject.org, 2018). Unlike regular web browsing, this obfuscates the user's IP address, ensuring that their identity is not disclosed to the websites they visit. By redirecting the traffic through encrypted channels, data is also protected from passive traffic analysis.

Use of the network alone does not guarantee anonymity, making the Tor Browser Bundle (TBB) a necessary component in the privacy-oriented architecture (Torproject.org, 2018). The TBB is an extended support release (ESR) of Mozilla's Firefox browser, bundled with mandatory add-ons which protect user anonymity. The browser's design philosophy states that the TBB should protect the user from known web attacks designed to reveal a user's identity and also minimise the amount of browsing data written to disk (Perry et al., 2018). To enable this, plugins such as NoScript, which protects the user from malicious code on websites, are used by default, as is Firefox's Private Browsing mode. Forcing Private Browsing mode means that web history is not saved to disk by the browser itself, which should make it difficult for a forensic adversary to determine websites visited (Mozilla.org, 2019).

2.2. Motivation

Users of the Tor Browser and network rely on the perceived anonymity offered by the Tor Project. This research is motivated by the idea that little importance has been placed on protecting the user's anonymity from a forensic perspective, both by the Tor Project and by academic research in this area. If user anonymity is easily compromised by a forensic adversary, the browser could be considered a weak link in Tor's privacy-focused architecture. Whilst it is not currently illegal to use Tor in many states, shifts in attitudes towards the browser could take place as a reaction to political events, or as a wider drive for censorship. Moreover, users of Tor who see the browser as a tool for the greater good, such as journalists and political dissidents, may be easily identified if the browser leaves obvious traces of its use on computing systems. This could lead to persecution and further surveillance, regardless of legality. Understandably, the same features which make Tor attractive to the aforementioned users, also make it attractive to those who wish to cause harm. Therefore, a greater understanding of the forensic implications of using Tor is required by law enforcement investigating a range of digital crimes. It is this complex societal impact which has inspired the subject of this paper.

2.3. Traditional Browser Forensics

Traditional web browsers record a significant amount of data from a browsing session - particularly in the default browsing mode. Web browser forensics involves the recovery of browsing artefacts which reveal information regarding a suspect's online activity. Browser forensics has become increasingly important for investigators as search history, download activity and page views can aid understanding of the criminal motive.

Like other popular browsers, Firefox, the TBB's underlying browser engine, stores history, download and cookie information in SQLite databases, generally in clear text. This allows forensic investigators to easily retrieve browsing information (Roussas, 2009; Shepherd, 2018; Noorulla, 2014). Acknowledging this lack of user privacy, browser developers introduced a private browsing mode, intended to limit or, ideally, eliminate browsing artefacts being written to disk. In a comprehensive study, Montasari and Peltola (2015) found that the extent to which this aim has been achieved varies widely between browsers, with Firefox considered the second-most secure of the four mainstream browsers tested.

Using a methodology which incorporated virtualised Windows platforms and the use of Autopsy to analyse snapshots, Montasari and Peltola (2015) attempted to recover cached web pages, web browsing history, download history, visited URLs and search terms from the target computer after performing a pre-defined browsing protocol using each browser's version of private browsing. A variety of typical activities, involving different types of web content, were covered:

1. Searching for and viewing a YouTube video.
2. Using Google to find a web page and then performing another search on the page.
3. Downloading a Facebook profile picture.
4. Searching and viewing an Amazon item.
5. Searching for and viewing a PDF document using Google.

The analysis was split into two stages; examining firstly common and uncommon locations on a hard drive and secondly the contents of the computer's volatile memory (RAM). Of the four browsers tested, Firefox was deemed relatively secure, although traces of search terms issued during the browsing protocol were discovered in pagefile.sys and the profile picture downloaded from Facebook was also discovered using file carving.

These results confirm earlier work which found artefacts including search terms and URLs from private browsing sessions in pagefile.sys (David and Hargreaves, 2012; Findlay and Leimich, 2014) and in RAM (Aggarwal et al., 2010; Findlay and Leimich, 2014), and contradict Mozilla's claims that Firefox leaves no traces after termination of a private browsing session (Mozilla.org, 2019).

2.4. Overview of Tor Forensics Research

Any weaknesses in Mozilla's implementation of private browsing suggest that the TBB could be susceptible to similar exploitation unless the developers have managed to mitigate these vulnerabilities. Acknowledging the need for a forensic audit, Sandvik (2013) of the Tor Project conducted a cross-platform forensic analysis of the TBB on OSX, Linux and Windows. The TBB ”aims to ensure that the user is able to completely and safely remove the bundle without leaving other traces on her computer” (Sandvik, 2013). However, the analysis results showed that the default behaviour of the operating system hosting the TBB could compromise user anonymity through its generation of artefacts related to features such as prefetching, paging and index searching, negating the anti-forensics measures implemented by the developers.

Darcie et al. (2014) found that analysis of the Windows Registry could determine the presence of the TBB after uninstallation, while Epifani et al. (2015) discovered during a real-life case study, where a rogue employee's laptop was seized, that Tor use on the target computer was proven by artefacts residing in Tor's install directory as well as the behaviour of Windows prefetching. Hence the Tor Project either does not or cannot prevent a trail of evidence being created by the host OS.

As the TBB was found to be successful in avoiding browsing data being written to the hard disk (besides in low-RAM situations) (Darcie et al., 2014), a static forensic analysis would likely yield evidence of installation and use. This could be advantageous to an investigator wishing to prove the use of Tor over other browsers, however, lacking artefacts of browsing history, it does not offer a picture of a subject's activity within Tor. A requirement for live forensic analysis became evident.

Dayalamurthy (2013) evaluated the work of previous forensic research using RAM dumps and suggested that this live forensics approach would be feasible when analysing the use of the TBB and attempting to de-anonymise Tor users. Her research proposed a forensic methodology which includes the use of the Volatility framework for memory analysis, along with the recovery of web graphics and analysis of the Windows Registry. However, this methodology was never tested.

Darcie et al. (2014) discovered that by creating a RAM dump of a target system, indexing it and searching for the keyword *Tor*, numerous hits were returned in a computer used to download Tor, a computer where Tor was used and finally, a computer where Tor had been used and then uninstalled. This method of performing string searches on RAM captures echoes the methodology used by Montasari and Peltola (2015). Warren (2017) later expanded on this analysis of RAM by using the Volatility Framework, with the addition of community-developed plugins, to conduct an analysis of the RAM of a Windows 10 VMWare virtual machine. In doing so, evidence of Tor was found through the identification of Tor DLLs, environment variables, SIDs and command line usage.

3. Methodology

3.1. Design considerations

Throughout this project, forensic analysis is intended to have two clear aims:

1. Expanding on previous research to confirm that the existence and use of the Tor Browser is still determinable in Windows 10, and
2. Proving that artefacts from the Tor browsing protocol can be recovered from the target computer.

Live analysis of an application during runtime is particularly beneficial in order to understand how the host operating system and application interact. As the design efforts of the Tor project have focused on preventing writes to disk (Perry et al., 2018), a live analysis could potentially yield more information pertaining to the browsing session. Using an earlier version of Tor (3.6.1), Darcie et al. (2014) found evidence of web browsing in the form of JPEG and HTML files in live forensics but dead-box (static) forensics was unsuccessful. In a previous live forensics analysis of the Firefox browser, artefacts from a private browsing session were recovered from memory whilst the browsing session was open, and - to a lesser extent after the browsing session had closed (Findlay and Leimich, 2014). This showed that Firefox was able to terminate running processes, effectively flushing memory of artefacts of the browsing protocol when the user closed the Private Browsing window. However, whether or not this is also true for the TBB has not been established; this will be taken into account in our methodology.

To build upon previous research, our approach has been designed to answer the following questions:

- Does Tor manage to protect the user by flushing evidence of its use from RAM when the browsing session is closed?
- Can Tor use be detected at four key moments; whilst the browser window is open, after closing the browsing window, after deleting the installation directory/associated files, and after the user has logged out?
- Are files from the browsing protocol still recoverable in live forensics using Tor 7.5.2, the version current at the time of writing?

Answers to the above should protect the user's anonymity from an adversary who manages to seize a personal computer during or shortly after Tor use, as well as after the seizure of a shared computer. Timing is a key parameter when designing a live forensics methodology. Figure 1 gives an overview of the experimental methodology used in this project, clearly showing the timing of RAM captures for live forensics and virtual machine snapshots for static forensics. The experiments were repeated using Tor in mobile mode, i.e. run from a thumb drive, echoing Findlay and Leimich (2014) who found that mobile browsing reduced the artefacts left behind on the host, and using private browsing mode in native Firefox as a control.

Inspired by Montasari and Peltola (2015), our test protocol mimics a user's browsing session with a variety of actions (i.e. searching, interacting with web forms, downloading, streaming media):

1. Type theguardian.co.uk into the address bar and view the page.
2. Go to https://support.mozilla.org and enter profile folder into the search box (submission to a web form).
3. Go to images.google.com, search for moog mother 32 and download the first image.
4. Go to https://soundcloud.com/monolake/liveego1999 and stream the first three minutes of the embedded audio.
5. At ebay.co.uk, search for nike air force 1 and view the first item.

The URLs and search terms were selected to include unusual strings to facilitate later keyword searches without false positives.

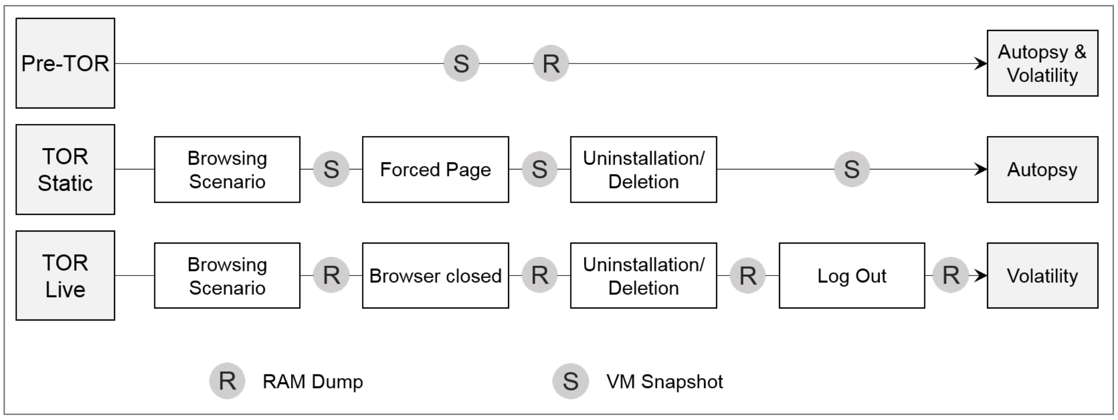


Figure 1: Overview of the Experimental Methodology

3.2. RAM Capture and Analysis Tools

VirtualBox supports the exporting of the contents of RAM, provided the virtual machine is started from the command line with debugging mode enabled. The .pgmtophysfile filename command can then be issued at any point to dump the contents of RAM. The resulting ELF 64 file can then be analysed by the Volatility Framework (http://www.volatilityfoundation.org).

Volatility was chosen due to its versatility as a command line tool, the wide range of plugins available for it and its effective use in previous Tor forensics research (Warren, 2017). With its vast array of plugins, Volatility is highly effective in analysing the state of memory at different points in time, building a comprehensive picture of the use of physical memory by an application. To address the aims stated in Section 3.1, the following Volatility plugins were chosen (Volatility Foundation, 2017):

- cmdline: displays command line arguments for currently running processes.
- dlllist: displays loaded DLLs for a given process.
- dumpfiles: extracts memory-mapped and cached files.
- envars: displays the environment variables used by processes.
- pslist: prints all running processes by following the EPROCESS kernel structure.
- pstree: prints the running processes as a tree, identifies parent and child processes.
- shellbags: displays shellbag information taken from appropriate registry keys.
- timeliner: creates a timeline from various artefacts in memory.

A common practice in live forensics is to observe running processes through the use of tools such as procmon (Darcie et al., 2014). Processes are assigned unique IDs (PIDs) in Windows and can be used to confirm that an application was running during the time of the analysis. Once the PID of a process is discovered, the process can then be searched in the output of Volatility's plugins, allowing an investigator to map its behaviour at certain points in time. Live analysis will take place first so that knowledge of the results which identify Tor use (for example, the names of DLLs) can then be used in a static analysis. Since it is considered likely that individually dumped RAM captures will expose contents of the browsing session (i.e. through caching), this was deemed an important inclusion to this project, particularly as Warren (2017) was unable to do so due to time constraints. Volatility's dumpfiles plugin will be used to achieve this and the resulting files analysed manually. This approach, inspired by Dayalamurthy (2013), should show changes to the Windows Registry, paths to the Tor install directory and contents of the browsing protocol.

4. Results

4.1. Live Forensics

RAM was captured at four key moments in the experiment:

1. After the browsing scenario with the browser window still open.
2. After closing the browser window.
3. After dragging the installation directory to the Recycle Bin and emptying (simulating an uninstall).
4. After the user logs out.

This should allow conclusions to be drawn which answer the questions asked in Section 3.1. As we found the artefacts left behind to decrease after each of the four stages, we present the results from the final stage.

After uninstalling the TBB and logging out, the Tor-related processes had been ended, but outputs from parsing the volatile memory with cmdline, pslist and dlllist plugins still showed the firefox.exe process with PID 4384; psscan and timeliner additionally showed timestamps (Figure 2a,b), while envars and shellbags produced nothing notable. While these findings could have been caused by a vanilla installation of Firefox, the output of the pstree plugin (Figure 2c) includes the absolute path to the Tor install directory, tying the process to the Tor browser. Via the PID and PPID, the start and end times from psscan and timeliner can then also be attributed to Tor.

```
Offset (V)           Name           PID  PPID Thds Hnds Sess Wow64 Start                            Exit
------------------   -----------    ---  ---- ---- ---- ---- ----  -------------------------------  -------------------
0xffff838058b13640   svchost.exe    2476 584  16   0    1    0     2018-03-04 15:51:00 UTC+0000
0xffff8380587de640   SearchIndexer. 2952 584  13   0    0    0     2018-03-04 15:51:01 UTC+0080
0xffff838059528080   firefox. exe   4384 948  0 ----    1    1     2018-03-04 15:51:21 UTC+0000     2018-03-04 15:58:15 UTC+0000
```

(a) psscan

```
2018-03-04 15:49:53 UTC+0000| [PROCESS] | MemCompression | PID: 2112/PPID: 4/POffset: 0x31d08040
2018-03-04 15:49:50 UTC+0000| [PROCESS] | winlogon.exe   | PID: 564/PPID:  464/POffset: 0x18711080 End: 2018-03-04 15:59:40 UTC+0000
2018-03-04 15:51:21 UTC+0000| [PROCESS] | firefox.exe    | PID: 4384/PPID: 948/POffset: 0x12e53080 End: 2018-03-04 15:58:15 UTC+0000
1970-01-01 00:00:00 UTC+0000| [THREAD]  | System         | PID: 4/TID: 164
1970-01-01 00:00:00 UTC+0000| [THREAD]  | System         | PID: 4/TID: 168
1970-01-01 00:00:00 UTC+0000| [THREAD]  | svchost.exe    | PID: 712/TID: 4332
1970-01-01 00:00:00 UTC+0000| [THREAD]  | System         | PID: 4/TID: 4696
```

(b) timeliner

```
Name                                      Pid  PPid Thds Hnds Time
----------------------------------------- ---- ---- ---- ---- ----
....0xffff838057943640: fontdrvhost.ex     4892 5020 7    0    2018-03-04 15:59:40 UTC+0000
        audit: \Device\HarddiskVolume2\Windows\System32\fontdrvhost.exe
        cmd: "fontdrvhost.exe"
        path: C:\Windows\system32\fontdrvhost.exe
....0xffff8380579bb640: dwm.exe            5024 5020 15   0    2018-03-04 15:59:40 UTC+0000
        audit: \Device\HarddiskVolume2\Windows\System321dwm.exe
        cmd: "dwm.exe"
        path: C:Windows\system32\dwm.exe
...0xffff838059967080:csrss.exe            4772 4640 14   0    2018-03-04 15:59:40 UTC+0000
        audit: Device\HarddiskVolume2\Windows\System32\csrss.exe
        cmd : %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection-1024, 20480, 768 Windows-on
        path: C:Windows\system32\csrss.exe
. 0xffff838059932040:MemCompression        2112 4    10   0    2018-03-04 15:49:53 UTC+0000
        audit: MemCompression
0xffff838059528080: firefox.exe            4384 948  0         2018-03-04 15:51:21 UTC+0000
        audit: \Device\HarddiskVolume2\Users 40187070 Desktop Tor Browser\Browser\firefox.exe
```

(c) pstree

Figure 2: Selected Volatility Plugin Outputs - After logout

```
sseldorf June 5 199 by monolake - Tor Browser
SC:\Users\40187070\Desktop\Tor Browser\Browser\firefox.exe
firefox.exe
(dMonolake Live at Ego D
sseldorf June 5 1999 by monolake | Free Listening on SoundCloud - Tor Browser
SC:\Users\40187070\Desktop\Tor Browser\Browser\firefox.exe
firefox.exe
```

Figure 3: Results of *monolake* string search, reflecting the title of the webpage (<title>Monolake Live at Ego Düsseldorf June 5 1999 by monolake |Free Listening on SoundCloud</title>) suffixed with - *Tor Browser*

```
(+moog mother 32 Google-Suche Tor Browser
9C:\Users\40187070\Desktop\Tor Browser\Browser\firefox.exe
firefox.exe
(Mother-32 Moog Music Inc - Tor Browser
9C:\Users\40187070\Desktop\Tor Browser\Browser\firefox.exe
firefox.exe
Tor Browser
```

Figure 4: String search for *moog mother 32* in NTUSER.DAT (extract)

4.2. Static Forensics

Snapshots of the VM were taken after three key stages in the experiment to enable a static forensic analysis to assess Tor's ability to prevent writing data to disk. Again, we focus on the final snapshot, taken after the uninstallation/deletion of Tor (Figure 1).

4.2.1. Keyword Searches

The first step in the analysis was to carry out keyword searches for *monolake*, *soundcloud*, *the guardian* and *moog mother 32* to determine if and where these strings could be found in the forensic image. All search strings were found to appear in the same three files, with several occurrences in each file: NTUSER.DAT, ntuser.dat.LOG1 (a log of changes to NTUSER.DAT) and MEMORY.DMP. The only exception was *moog mother 32*, which did not appear in ntuser.dat.LOG1 but instead appeared in unallocated space, suggesting that the log may have been flushed. Closer inspection of the contents of the three files revealed that all had exactly the same content, suggesting that the content of NTUSER.DAT was being logged and then replicated in MEMORY.DMP. Surprisingly, the file contents included the entire page title of each website visited suffixed with - *Tor Browser* as well as the absolute path to the Tor install directory, clearly showing the username (40187070) as well as referencing firefox.exe within the Tor Browser directory. Figure 3 shows the *monolake* results as an example.

The keyword matches for the string moog mother 32 reflect its use in a Google search (Figure 4). Interestingly, the page title contained the German word for search, Suche, suggesting that the Tor exit node was located in a German-speaking country. The Downloads folder and explorer.exe also appear in this context, reflecting the saving of the downloaded image. As with the other keyword searches, the absolute path to the Tor directory is shown, with the Firefox executable clearly identified.

```
ebay.co.uk/itm/Nike-Air-Force-1-x-Clot-silk-Nikelab-UK-11-Us-12/1325349394302hash-itemledbb2af26:g:AAO
privateBrowsingId=1&firstPartyDomain=ebay.co.uk
EtwR
 W9r
```

(a) Ebay URL and Private Browsing evidence in .dll and .edb Files

```
Accept-Encoding
gzip, deflate, br
Referer
https://www.ebay.co.uk/itm/Nike-Air-Force-1-x-Clot-silk-Nielab-UK-11-US-12/1325349394302hash=itemledbb2a
f26:g:~kwAAOSwljJalrwS
Connection
keep-alive
Upgrade-Insecure-Requests
```

(b) Ebay URL and HTTP Header Information in .reg Files

```
Pgs*
&RJGV
\Device\HarddiskVolume2\Users\40187070\Desktop\Tor Browser\Browser\TorBrowser\Data\Browser\profile
.default\cookies.sqlite-journal
P1~*
\Device\HarddiskVolume2\Users\40187070\Desktop\Tor Browser\Browser\TorBrowser\Data\Browser\profile
.default\cookies.sqlite-journal
```

(c) References to Tor Install Directory (extract)

```
pref("extensions.torlauncher.default_bridge.obfs4.20", "obfs4 38.229.1.76:80
C8CBDB2464FC9804A69531437BCF2BE31FDD2EE4
cert=Hmyfd2ev46gGY7NoVxA9ngrPF2sCZtsskRTsoWXbxNkmeVnGFPWmrTtILRyqCTjHR+s9dg iat-mode=0");
/**/pref/**/"extensions.torlauncher.default_bridge.obfs4.21"/**/, /**/"obfs4 38.229.33.83:80
0BAC39417268B9B9F514ED7F63FA6FBAIA788955
cert=VwEFpk9F/UN9JED7XpG1XOjm/O8ZCXK80oPecgWnNDZDv5pdkhqlOpbAH0wNqOT6H6BmRQ iat-mode=0");
pref("extensions.torlauncher.default_bridge.obfs4.22", "obfs4 [2001:470:b381:bfff:216:3eff:fe23:d6c3J:443
CDF2E852BF539B62BD10E27E9115A31734E378C2
cert=qUVQ0srL1JI/vO6V6m/24anYXiJD3QP2HgsUKQtQ7GRqqUvs7P+tG42RtAqdhLOALP7DJQ iat-mode=0");
pref("extensions.torlauncher.default_bridge.obfs4.23", "obfs4 37.216.240.24:40035
88CD3ED45A35271963EF82E511Ce827A24730913
cert=eGXYfW0DcgqIdPJ+rRupg4GGvVGfh25FWaIXZkit2060Sngsp7GAIiGIXOJJROMxEqFKJg iat-mode=0");
pref("extensions.torlauncher.default_bridge.obfs4.24", "obfs4 37.216.245.14:38224
```

(d) Obfs4 Bridging IP Addresses (extract)

Figure 5: Artefacts in Unallocated Space - After Delete

4.2.2. *Unallocated Space*

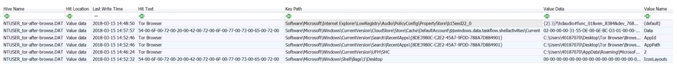
Several artefacts were recovered from unallocated space using Autopsy's carver module. When searching for the string 'clot' from the browsing protocol, six .dll, .edb and .reg files were discovered in unallocated space. Closer examination of these files showed considerable browsing data leakage: all contained the Ebay URL visited; the .dll and .edb files also referenced Private Browsing while the .reg file included remnants of HTTP header information (Figure 5a-b). Further searching of unallocated space uncovered references to the Tor installation directory and the obfs4 bridging IP addresses (Figure 5c, d). The browsing data found in NTUSER.DAT was also replicated in unallocated space. This may have occurred as a result of the machine crashing, however further analysis would need to be conducted to prove this.

4.3. *Analysis of NTUSER.DAT*

As static analysis revealed an abundance of artefacts in the user profile's NTUSER.DAT registry hive, but Autopsy's parsing of registry is limited to string searches, NTUSER.DAT hives from several VM snapshots were extracted for further analysis.

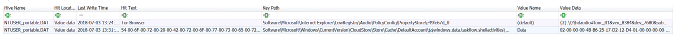
Firstly, a string search for 'Tor Browser' was carried out on the NTUSER.DAT file from the Tor After Browse snapshot with the free forensic tool Registry Explorer. Figure 6a shows the results of this search. Of particular interest is the Key Path column which shows the paths of the six different Registry keys where the string was found.

The browsing data strings previously seen in Autopsy occur in HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\CloudStore\Store\Cache\DefaultAccount\$$windows.data.taskflow.shellactivities\Current (the *shellactivities key* for short), making it the most interesting. The value, of REG_BINARY type, contains page titles from the pages visited during the browsing protocol in Unicode



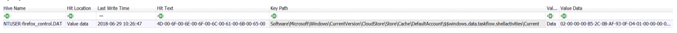
Hive Name	Hit Location	Last Write Time	Hit Text	Key Path	Value Data	Value Name
NTUSER_tor-after-browse.DAT	Value data	2018-03-15 14:48:50	Tor Browser	Software\Microsoft\Internet Explorer\LowRegistry\Audio\PolicyConfig\PropertyStore\b15eed22_0	{2}.\\?\hdaudio#func_01&ven_8384&dev_768...	(default)
NTUSER_tor-after-browse.DAT	Value data	2018-03-15 14:57:57	54-00-6F-00-72-00-20-00-42-00-72-00-6F-00-77-00-73-00-65-00-72-00	Software\Microsoft\Windows\CurrentVersion\CloudStore\Store\Cache\DefaultAccount\$$windows.data.taskflow.shellactivities\Current	02-00-00-00-31-55-DE-00-6E-BC-D3-01-00-00-...	Data
NTUSER_tor-after-browse.DAT	Value data	2018-03-15 14:52:46	Tor Browser	Software\Microsoft\Windows\CurrentVersion\Search\RecentApps\{8DE3980C-C2E2-45A7-9FDD-788A7D884901}	C:\Users\40187070\Desktop\Tor Browser\Brows...	AppId
NTUSER_tor-after-browse.DAT	Value data	2018-03-15 14:52:46	Tor Browser	Software\Microsoft\Windows\CurrentVersion\Search\RecentApps\{8DE3980C-C2E2-45A7-9FDD-788A7D884901}	C:\Users\40187070\Desktop\Tor Browser\Brows...	AppPath
NTUSER_tor-after-browse.DAT	Value data	2018-03-15 14:46:24	Tor Browser	Software\Microsoft\Windows\CurrentVersion\UFH\SHC	C:\Users\40187070\AppData\Roaming\Microsof...	2
NTUSER_tor-after-browse.DAT	Value data	2018-03-15 14:52:32	54-00-6F-00-72-00-20-00-42-00-72-00-6F-00-77-00-73-00-65-00-72-00	Software\Microsoft\Windows\Shell\Bags\1\Desktop	00-00-00-00-00-00-00-00-00-00-00-00-00-00-0...	IconLayouts

(a) Tor, Standard Mode



Hive Name	Hit Locat...	Last Write Time	Hit Text	Key Path	Value Name	Value Data
NTUSER_portable.DAT	Value data	2018-07-03 13:24:...	Tor Browser	Software\Microsoft\Internet Explorer\LowRegistry\Audio\PolicyConfig\PropertyStore\e49fe67d_0	(default)	{2}.\\?\hdaudio#func_01&ven_8384&dev_7680&sub...
NTUSER_portable.DAT	Value data	2018-07-03 13:31:...	54-00-6F-00-72-00-20-00-42-00-72-00-6F-00-77-00-73-00-65-00-72...	Software\Microsoft\Windows\CurrentVersion\CloudStore\Store\Cache\DefaultAccount\$$windows.data.taskflow.shellactivities\...	Data	02-00-00-00-4B-B6-25-17-D2-12-D4-01-00-00-00-00-...

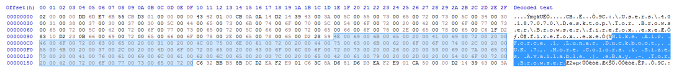
(b) TOR, Portable Mode



Hive Name	Hit Location	Last Write Time	Hit Text	Key Path	Val...	Value Data
NTUSER-firefox_control.DAT	Value data	2018-06-29 10:26:47	4D-00-6F-00-6E-00-6F-00-6C-00-61-00-6B-00-65-00	Software\Microsoft\Windows\CurrentVersion\CloudStore\Store\Cache\DefaultAccount\$$windows.data.taskflow.shellactivities\Current	Data	02-00-00-00-B5-2C-0B-AF-93-0F-D4-01-00-00-00-0...

(c) Firefox, Private Mode

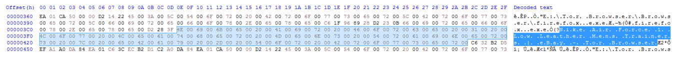
Figure 6: NTUSER.DAT Keys containing Search Strings



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F  Decoded text

00000000  02 00 00 00 DD 6D E7 6B 55 CB D3 01 00 00 00 00 43 42 01 00 CB 0A 0A 14 D2 14 39 43 00 3A 00 5C 00 55 00 73 00 65 00 72 00 73 00 5C 00 34 00 30  ....ÝmçkUËÓ.....CB..Ë...Ò.9C.:.\.U.s.e.r.s.\.4.0
00000030  00 31 00 38 00 37 00 30 00 37 00 30 00 5C 00 44 00 65 00 73 00 6B 00 74 00 6F 00 70 00 5C 00 54 00 6F 00 72 00 20 00 42 00 72 00 6F 00 77 00 73  .1.8.7.0.7.0.\.D.e.s.k.t.o.p.\.T.o.r. .B.r.o.w.s
00000060  00 65 00 72 00 5C 00 42 00 72 00 6F 00 77 00 73 00 65 00 72 00 5C 00 66 00 69 00 72 00 65 00 66 00 6F 00 78 00 2E 00 65 00 78 00 65 00 C6 1F D2  .e.r.\.B.r.o.w.s.e.r.\.f.i.r.e.f.o.x...e.x.e.Æ.Ò
00000090  83 10 D2 23 0B 66 00 69 00 72 00 65 00 66 00 6F 00 78 00 2E 00 65 00 78 00 65 00 D2 28 59 4E 00 69 00 6B 00 65 00 20 00 41 00 69 00 72 00 20 00  ƒ.Ò#.f.i.r.e.f.o.x...e.x.e.Ò(YN.i.k.e. .A.i.r. .
000000C0  46 00 6F 00 72 00 63 00 65 00 20 00 31 00 20 00 4C 00 75 00 6E 00 61 00 72 00 20 00 44 00 75 00 63 00 6B 00 62 00 6F 00 6F 00 74 00 2C 00 20 00  F.o.r.c.e. .1. .L.u.n.a.r. .D.u.c.k.b.o.o.t.,. .
000000F0  55 00 4B 00 20 00 37 00 2C 00 20 00 4D 00 6F 00 72 00 65 00 20 00 43 00 6F 00 6C 00 6F 00 72 00 73 00 20 00 26 00 20 00 53 00 69 00 7A 00 65 00  U.K. .7.,. .M.o.r.e. .C.o.l.o.r.s. .&. .S.i.z.e.
00000120  73 00 20 00 41 00 76 00 61 00 69 00 6C 00 61 00 62 00 6C 00 65 00 20 00 7C 00 20 00 65 00 42 00 61 00 79 00 20 00 2D 00 20 00 54 00 6F 00 72 00  s. .A.v.a.i.l.a.b.l.e. .|. .e.B.a.y. .-. .T.o.r.
00000150  20 00 42 00 72 00 6F 00 77 00 73 00 65 00 72 00 C6 32 BB B5 8B DC D2 EA F2 E9 01 C6 3C 8A D6 81 D6 D3 EA F2 E9 01 CA 50 00 00 D2 14 39 43 00 3A   .B.r.o.w.s.e.r.Æ2»µ‹ÜÒêòé.Æ<ŠÖ.ÖÓêòé.ÊP..Ò.9C.:
```

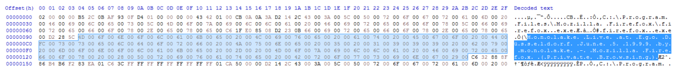
(a) TOR, Standard Mode



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F  Decoded text

00000360  EA 01 CA 50 00 00 D2 14 22 45 00 3A 00 5C 00 54 00 6F 00 72 00 20 00 42 00 72 00 6F 00 77 00 73 00 65 00 72 00 5C 00 42 00 72 00 6F 00 77 00 73  ê.ÊP..Ò."E.:.\.T.o.r. .B.r.o.w.s.e.r.\.B.r.o.w.s
00000390  00 65 00 72 00 5C 00 66 00 69 00 72 00 65 00 66 00 6F 00 78 00 2E 00 65 00 78 00 65 00 C6 1F 96 89 28 D2 23 0B 66 00 69 00 72 00 65 00 66 00 6F  .e.r.\.f.i.r.e.f.o.x...e.x.e.Æ.–‰(Ò#.f.i.r.e.f.o
000003C0  00 78 00 2E 00 65 00 78 00 65 00 D2 28 3F 4E 00 69 00 6B 00 65 00 20 00 41 00 69 00 72 00 20 00 46 00 6F 00 72 00 63 00 65 00 20 00 31 00 20 00  .x...e.x.e.Ò(?N.i.k.e. .A.i.r. .F.o.r.c.e. .1. .
000003F0  4C 00 6F 00 77 00 20 00 4C 00 65 00 61 00 74 00 68 00 65 00 72 00 20 00 4D 00 65 00 6E 00 73 00 20 00 54 00 72 00 61 00 69 00 6E 00 65 00 72 00  L.o.w. .L.e.a.t.h.e.r. .M.e.n.s. .T.r.a.i.n.e.r.
00000420  73 00 20 00 7C 00 20 00 65 00 42 00 61 00 79 00 20 00 2D 00 20 00 54 00 6F 00 72 00 20 00 42 00 72 00 6F 00 77 00 73 00 65 00 72 00 C6 32 B2 D5  s. .|. .e.B.a.y. .-. .T.o.r. .B.r.o.w.s.e.r.Æ2²Õ
00000450  EF A1 A0 DA 84 EA 01 C6 3C EC B2 D1 C2 A0 DA 84 EA 01 CA 50 00 00 D2 14 22 45 00 3A 00 5C 00 54 00 6F 00 72 00 20 00 42 00 72 00 6F 00 77 00 73  ï¡ Ú„ê.Æ<ì²ÑÂ Ú„ê.ÊP..Ò."E.:.\.T.o.r. .B.r.o.w.s
```

(b) TOR, Portable Mode



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F  Decoded text

00000000  02 00 00 00 B5 2C 0B AF 93 0F D4 01 00 00 00 00 43 42 01 00 CB 0A 0A 3A D2 14 2C 43 00 3A 00 5C 00 50 00 72 00 6F 00 67 00 72 00 61 00 6D 00 20  ....µ,.¯“.Ô.....CB..Ë..:Ò.,C.:.\.P.r.o.g.r.a.m.
00000030  00 46 00 69 00 6C 00 65 00 73 00 5C 00 4D 00 6F 00 7A 00 69 00 6C 00 6C 00 61 00 20 00 46 00 69 00 72 00 65 00 66 00 6F 00 78 00 5C 00 66 00 69  .F.i.l.e.s.\.M.o.z.i.l.l.a. .F.i.r.e.f.o.x.\.f.i
00000060  00 72 00 65 00 66 00 6F 00 78 00 2E 00 65 00 78 00 65 00 C6 1F E0 85 08 D2 23 0B 66 00 69 00 72 00 65 00 66 00 6F 00 78 00 2E 00 65 00 78 00 65  .r.e.f.o.x...e.x.e.Æ.à….Ò#.f.i.r.e.f.o.x...e.x.e
00000090  00 D2 28 5C 4D 00 6F 00 6E 00 6F 00 6C 00 61 00 6B 00 65 00 20 00 4C 00 69 00 76 00 65 00 20 00 61 00 74 00 20 00 45 00 67 00 6F 00 20 00 44 00  .Ò(\M.o.n.o.l.a.k.e. .L.i.v.e. .a.t. .E.g.o. .D.
000000C0  FC 00 73 00 73 00 65 00 6C 00 64 00 6F 00 72 00 66 00 20 00 4A 00 75 00 6E 00 65 00 20 00 35 00 20 00 31 00 39 00 39 00 39 00 20 00 62 00 79 00  ü.s.s.e.l.d.o.r.f. .J.u.n.e. .5. .1.9.9.9. .b.y.
000000F0  20 00 6D 00 6F 00 6E 00 6F 00 6C 00 61 00 6B 00 65 00 20 00 2D 00 20 00 4D 00 6F 00 7A 00 69 00 6C 00 6C 00 61 00 20 00 46 00 69 00 72 00 65 00   .m.o.n.o.l.a.k.e. .-. .M.o.z.i.l.l.a. .F.i.r.e.
00000120  66 00 6F 00 78 00 20 00 28 00 50 00 72 00 69 00 76 00 61 00 74 00 65 00 20 00 42 00 72 00 6F 00 77 00 73 00 69 00 6E 00 67 00 29 00 C6 32 88 8F  f.o.x. .(.P.r.i.v.a.t.e. .B.r.o.w.s.i.n.g.).Æ2ˆ.
00000150  86 B4 B6 F2 83 EA 01 C6 3C FF FF FF FF FF FF FF FF FF 01 CA 50 00 00 D2 14 2C 43 00 3A 00 5C 00 50 00 72 00 6F 00 67 00 72 00 61 00 6D 00 20 00  †´¶òƒê.Æ<ÿÿÿÿÿÿÿÿÿ.ÊP..Ò.,C.:.\.P.r.o.g.r.a.m. .
```

(c) Firefox, Private Browsing

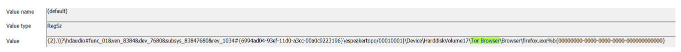
Figure 7: Extracts of the shellactivities registry key showing titles of browsing tabs

as well as the absolute path to the application generating the browsing tabs, C:\Users\40187070\Desktop\Tor Browser\Browser\firefox.exe (Figure 7a).

To test whether writing of data to the Registry still occurs when the browser is run from an external device (in portable mode), the same analysis was performed on the NTUSER.DAT file extracted from a Tor Portable snapshot. While the string 'Tor Browser' was found in only two keys (Figure 6b), the shellactivities key again contained significant evidence from the browsing protocol including page titles and the absolute path to the Firefox executable within Tor's installation directory (Figure 7b). The latter, E:\Tor Browser\Browser\firefox.exe, reflects the use of an external device, more details about which can be found by correlating the drive letter with registry keys such as USBSTOR.

In addition to the shellactivities key, the key HKCU\Software\Microsoft\InternetExplorer\LowRegistry\Audio\PolicyConfig\PropertyStore\<value> contained the 'Tor Browser' string in both the snapshot of Tor being run locally and in portable mode. Closer examination of the value data showed that it contains part of the path to the Firefox executable within the TOR browser installation directory (Figure 8). This key is likely used in the handling of audio by the host computer and the data would have been written to it when audio was streamed from the Internet as part of the browsing protocol. While the shellactivities key is a more comprehensive record of user activity, this audio key could corroborate the evidence, providing the user has used it to stream audio.

As a control, the same experiment was performed using the 'vanilla' Firefox browser in private browsing mode on a fresh VM, with a snapshot taken after executing the pre-determined browsing activities. As this control could not contain any references to the Tor browser, 'monolake' was used as the search



```
Value name   (default)
Value type   RegSz
Value        {2}.\\?\hdaudio#func_01&ven_8384&dev_7680&subsys_83847680&rev_1034#{6994ad04-93ef-11d0-a3cc-00a0c9223196}\espeakertopo/00010001|\Device\HarddiskVolume17\Tor Browser\Browser\firefox.exe%b{00000000-0000-0000-0000-000000000000}
```

Figure 8: Tor Browser String in \Audio\PolicyConfig\PropertyStore - TOR Portable

string. The only match was in the shellactivities key (Figure 6c). Further inspection of this key in a hex editor revealed the same data leakage seen in the two Tor snapshots (Figure 7c).

4.4. Decoding the shellactivities key contents

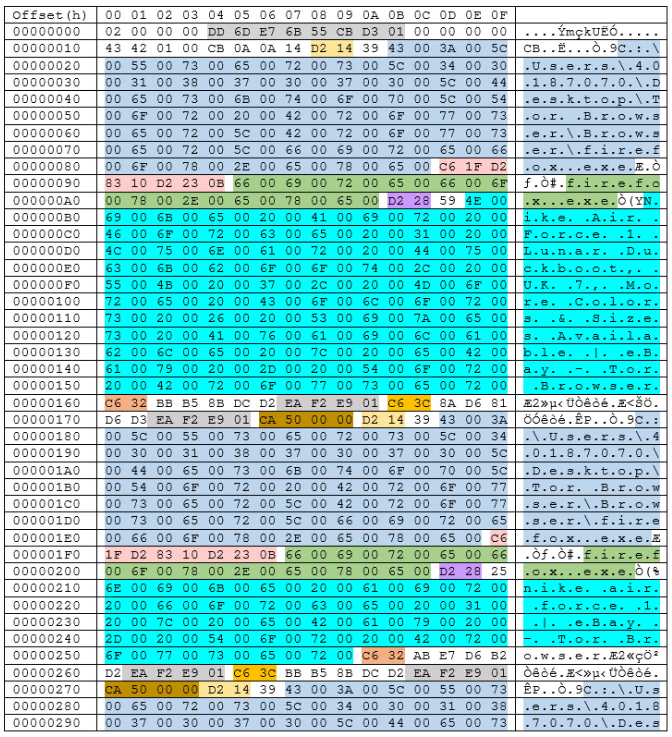
It is evident that, under Windows 10, browsing data from user sessions is written to non-volatile storage, specifically the shellactivities registry key. This data leakage occurs regardless whether Tor, built upon the Extended Support Release (ESR) of Firefox, is used or Firefox's default Private Browsing mode, and also when a portable browser is used. This consistency suggests that the observed data leakage is a result of how the Operating System handles applications rather than some anomaly in the operation of Tor or Firefox. As a result, the anonymity-preserving intentions of both the Tor Project and Mozilla's private browsing mode are negated. Conversely, the shellactivities registry key is likely to contain valuable forensic artefacts and thus warrants further investigation.

The parent of the shellactivities registry key, Cloudstore, does not exist in Windows 7, nor in the 1511 update of Windows 10. It first appears in the Windows 10 with the Anniversary Update, (1607 / 1603 in Win Education), but at that point it was not populated, having no subkeys or values. In Windows 10 machines running the Creator's Update 1703, the CloudStore key exists and has numerous subkeys that are populated with values, including the shellactivities key.

The data within the shellactivities is of REG_BINARY type, i.e. hexadecimal. It appears to have an internal structure, which warrants decoding fully as this would allow plugins for forensic tools such as Regripper, Encase etc to be created.

In Figure 9, we have colour-coded the start of the contents of this key after TOR was run in standard mode, uninstalled and the user logged out, in order to expose the internal structure.

- The key header starts with 0x02000000; this is followed by an eight-byte Windows Filetime Timestamp (highlighted grey, here representing 3 April 2018 14:09:47), then 4 bytes of zero and 8 bytes of values (here 0x43420100CB0A0A14).
- The first record entry starts at offset 0x18.
- Every record starts with a record header signature of 0xD214 (buff), followed by one byte (here consistently 0x39, but varies in other examples; this could be a type, length or offset indicator of some sort).
- This is followed by the absolute path to executable (blue-grey, variable length), then 8 bytes which are consistently 0xC61FD28310D2230B (pink), the name of the executable (green, variable length), 0xD228 (purple), 1 variable byte.
- After the above, Page title (turquoise) + 0xC632 (orange) + 5 bytes + 0xEAF2E901 + 0xC63C (yellow) + 5 bytes + 0xEAF2E901 + 0xCA500000 (dark gold).
- The underlined could be 1 byte + 8 byte timestamp of some sort - but see conversion attempts above.
- This structure looks to be consistent for the actual web pages viewed, but the final few records vary somewhat.
- If this is correct, the individual entries are delimited by 0xD21439 header and 0xCA500000 footer, and internal components separated by specific values as well, so size of entry would not need to be stored.
- This entry header and footer might then be searchable, e.g in registry entries, or possibly unallocated space if deleted? (but I don't know how registry deals with deletion - possibly internally).



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  02 00 00 00 DD 6D E7 6B 55 CB D3 01 00 00 00 00  ....ÝmçkUËÓ.....
00000010  43 42 01 00 CB 0A 0A 14 D2 14 39 43 00 3A 00 5C  CB..Ë...Ò.9C.:.\
00000020  00 55 00 73 00 65 00 72 00 73 00 5C 00 34 00 30  .U.s.e.r.s.\.4.0
00000030  00 31 00 38 00 37 00 30 00 37 00 30 00 5C 00 44  .1.8.7.0.7.0.\.D
00000040  00 65 00 73 00 6B 00 74 00 6F 00 70 00 5C 00 54  .e.s.k.t.o.p.\.T
00000050  00 6F 00 72 00 20 00 42 00 72 00 6F 00 77 00 73  .o.r. .B.r.o.w.s
00000060  00 65 00 72 00 5C 00 42 00 72 00 6F 00 77 00 73  .e.r.\.B.r.o.w.s
00000070  00 65 00 72 00 5C 00 66 00 69 00 72 00 65 00 66  .e.r.\.f.i.r.e.f
00000080  00 6F 00 78 00 2E 00 65 00 78 00 65 00 C6 1F D2  .o.x...e.x.e.Æ.Ò
00000090  83 10 D2 23 0B 66 00 69 00 72 00 65 00 66 00 6F  ƒ.Ò#.f.i.r.e.f.o
000000A0  00 78 00 2E 00 65 00 78 00 65 00 D2 28 59 4E 00  .x...e.x.e.Ò(YN.
000000B0  69 00 6B 00 65 00 20 00 41 00 69 00 72 00 20 00  i.k.e. .A.i.r. .
000000C0  46 00 6F 00 72 00 63 00 65 00 20 00 31 00 20 00  F.o.r.c.e. .1. .
000000D0  4C 00 75 00 6E 00 61 00 72 00 20 00 44 00 75 00  L.u.n.a.r. .D.u.
000000E0  63 00 6B 00 62 00 6F 00 6F 00 74 00 2C 00 20 00  c.k.b.o.o.t.,. .
000000F0  55 00 4B 00 20 00 37 00 2C 00 20 00 4D 00 6F 00  U.K. .7.,. .M.o.
00000100  72 00 65 00 20 00 43 00 6F 00 6C 00 6F 00 72 00  r.e. .C.o.l.o.r.
00000110  73 00 20 00 26 00 20 00 53 00 69 00 7A 00 65 00  s. .&. .S.i.z.e.
00000120  73 00 20 00 41 00 76 00 61 00 69 00 6C 00 61 00  s. .A.v.a.i.l.a.
00000130  62 00 6C 00 65 00 20 00 7C 00 20 00 65 00 42 00  b.l.e. .|. .e.B.
00000140  61 00 79 00 20 00 2D 00 20 00 54 00 6F 00 72 00  a.y. .-. .T.o.r.
00000150  20 00 42 00 72 00 6F 00 77 00 73 00 65 00 72 00   .B.r.o.w.s.e.r.
00000160  C6 32 BB B5 8B DC D2 EA F2 E9 01 C6 3C 8A D6 81  Æ2»µ‹ÜÒêòé.Æ<ŠÖ.
00000170  D6 D3 EA F2 E9 01 CA 50 00 00 D2 14 39 43 00 3A  ÖÓêòé.ÊP..Ò.9C.:
00000180  00 5C 00 55 00 73 00 65 00 72 00 73 00 5C 00 34  .\.U.s.e.r.s.\.4
00000190  00 30 00 31 00 38 00 37 00 30 00 37 00 30 00 5C  .0.1.8.7.0.7.0.\
000001A0  00 44 00 65 00 73 00 6B 00 74 00 6F 00 70 00 5C  .D.e.s.k.t.o.p.\
000001B0  00 54 00 6F 00 72 00 20 00 42 00 72 00 6F 00 77  .T.o.r. .B.r.o.w
000001C0  00 73 00 65 00 72 00 5C 00 42 00 72 00 6F 00 77  .s.e.r.\.B.r.o.w
000001D0  00 73 00 65 00 72 00 5C 00 66 00 69 00 72 00 65  .s.e.r.\.f.i.r.e
000001E0  00 66 00 6F 00 78 00 2E 00 65 00 78 00 65 00 C6  .f.o.x...e.x.e.Æ
000001F0  1F D2 83 10 D2 23 0B 66 00 69 00 72 00 65 00 66  .Òƒ.Ò#.f.i.r.e.f
00000200  00 6F 00 78 00 2E 00 65 00 78 00 65 00 D2 28 25  .o.x...e.x.e.Ò(%
00000210  6E 00 69 00 6B 00 65 00 20 00 61 00 69 00 72 00  n.i.k.e. .a.i.r.
00000220  20 00 66 00 6F 00 72 00 63 00 65 00 20 00 31 00   .f.o.r.c.e. .1.
00000230  20 00 7C 00 20 00 65 00 42 00 61 00 79 00 20 00   .|. .e.B.a.y. .
00000240  2D 00 20 00 54 00 6F 00 72 00 20 00 42 00 72 00  -. .T.o.r. .B.r.
00000250  6F 00 77 00 73 00 65 00 72 00 C6 32 AB E7 D6 B2  o.w.s.e.r.Æ2«çÖ²
00000260  D2 EA F2 E9 01 C6 3C BB B5 8B DC D2 EA F2 E9 01  Òêòé.Æ<»µ‹ÜÒêòé.
00000270  CA 50 00 00 D2 14 39 43 00 3A 00 5C 00 55 00 73  ÊP..Ò.9C.:.\.U.s
00000280  00 65 00 72 00 73 00 5C 00 34 00 30 00 31 00 38  .e.r.s.\.4.0.1.8
00000290  00 37 00 30 00 37 00 30 00 5C 00 44 00 65 00 73  .7.0.7.0.\.D.e.s
```

Figure 9: Internal structure of the shellactivities Key contents

5. Findings

5.1. Proposed Forensic Methodology

The interactions between Tor and Windows described in Section 4 can be used to design a generalised forensic methodology for identifying the use of TOR and Web pages visited using TOR or Private Browsing.

- Where possible, take a RAM dump. Analyse with Volatility's psscan, pstree and timeliner plugins to (a) establish the use of TOR and (b) find the username. This will also reveal timestamps and can be carried out even after the user has uninstalled TOR and logged out. Where they exist, pagefile.sys or hiberfile.sys can be used instead of a RAM dump. The analysis of all three of these data sources from the same system could result in the recovery of different, but nonetheless relevant - and corroborative or complimentary - evidence.

- From non-volatile storage, extract the NTUSER.DAT registry hive of the previously identified user, or, failing this, of all users.

- Extract the contents of the shellactivities key. Search for Tor and/or firefox to find the titles of web pages visited. Similarly, a search for other browser executables will reveal visited pages even if private browsing was used.

- A keyword search for 'obfs4' in unallocated space can reveal bridging IP addresses that may have been used by TOR.

5.2. *Live Forensics*

The live forensics methodology in this project showed that by using the Volatility Framework and associated plugins, four processes which can be attributed to Tor are created at runtime. These processes are as follows:

- firefox.exe: two occurrences of this process were seen.

- tor.exe.

- obfs4proxy.exe: it is unlikely this would have appeared had the obfs4 proxy not been used.

Whilst the names of these processes are obvious, the output of several of the Volatility Framework plugins also revealed the absolute path to the installation directory where the browser executable resides - providing further indication of Tor use. In doing so, the disk from which the browser was run, the username of the user running the browser, and the name of the browser can be clearly seen. This means that by using the Volatility Framework on a RAM dump seized whilst the browsing session is still active, a forensic adversary can positively identify the use of Tor.

After termination of the session, The Tor Browser manages to end tor.exe, obfs4proxy.exe and one of the firefox.exe processes. This makes it significantly more difficult for a forensic adversary to confirm that Tor was running on the machine at the time of seizure and coincides with the behaviour of Firefox observed by Findlay and Leimich (2014). Despite this, one firefox.exe process persisted. Although the process can still be seen, the environment variables, command line arguments, and DLLs which were attributable to this and the other three processes could no longer be identified after session termination. This suggests that the persisting firefox.exe process could not be fully terminated by closing the browser window and now exists in a traceable but inactive state. Regardless, the output of Volatility's pstree plugin still managed to attribute it to Tor by identifying the absolute path to the install directory. Without this finding, it would be impossible to determine that this process had originated from Tor and not a regular installation of Firefox.

This behaviour continued even after the user logged-out, proving the pstree plugin to be invaluable when attempting to prove the use of Tor on a shared computer.

In summary, Tor use can be easily detected using live forensics, particularly when the browsing session is still active. Ensuring that the browsing session is closed after use helps to conceal the fact that Tor was used. However, an artefact (firefox.exe) remains detectable in RAM after closure, deletion, and log-out. It is likely that the traceable artefact is the result of an anomaly in Firefox's handling of running processes. This belief is strengthened by the fact that Tor manages to remove all evidence of the processes directly attributed to its browser, yet one Firefox process remains. Perhaps this abnormality was introduced in an update of Firefox's Extended Support Release, or it may even be an unforeseen result of the interaction between Tor's plugins and the underlying browser. Nonetheless, it shows that reliance on a third-party browser can introduce problems which undermine user anonymity.

To expand on this, an examination of the files dumped from the contents of RAM will need to be performed. This will determine whether evidence of the browsing protocol is also present in RAM, a significantly greater finding than the identification of running processes. Unfortunately, time-constraints made this impossible in this project, an evaluation of which follows in a later section.

5.3. *Static Forensics*

5.3.1. *Browsing Data Leakage*

Perhaps the most interesting and unexpected conclusion of this project is that Tor does, in fact, write browsing data to disk. Artefacts from the browsing protocol, including HTTP header information, titles of web pages, and an instance of a typed URL were found in several Registry files. This means that use of static forensics by forensic investigators is potentially more worthwhile than examining the contents of RAM, at least for this version of the browser. Epifani et al. (2015) and Darcie et al. (2014) both mention the Registry as an area where Tor evidence could be recovered. In the Registry, Darcie et al. (2014) identified evidence of Tor presence, but not of browsing data. However, their searches were limited in only including the keyword Tor. Similarly, Epifani et al. (2015) showed that Tor could be identified in the NTUSER.DAT Registry hive, within the User Assist Key. Yet, they determined that only the number and time of execution could be found. Therefore, the findings of this research which show that the majority of the browsing protocol can be recovered from NTUSER.DAT are particularly significant. This suggests that in the versions between 4.0.2 (Darcie et al., 2014) and 7.5.0, a design change occurred which has led to this data leakage.

The vast majority of the browsing protocol was found in the NTUSER.DAT Windows Registry file. This makes it possible for a forensic adversary to reconstruct the activities of the user in relation to the browser. It was unclear what caused this information to be written to that area of the Registry, however multiple.reg files which are used to add changes to the Registry were found in the snapshot taken after the browser had been deleted. These files contained an Ebay URL visited during the browsing protocol, making it likely that some aspect of Tor was writing the contents of memory to the Registry. This is backed-up by the discovery of the ntuser.dat.LOG1 file, this file logs changes to the NTUSER.DAT Registry file and contained the same browsing data-leakage.

5.3.2. Firefox Artefacts

The two SQLite databases used by Firefox to track cookies and history (cookies.sqlite & places.sqlite) were both recoverable from the file system after deletion. Cookies.sqlite contained nothing of relevance but its existence could indicate that Tor was installed on the computer - if it could be proven that it did not come from a regular Firefox installation. Places.sqlite contained no information about the browsing protocol, however, it did contain entries for the Tor Project's blog and a Learn About Tor page. This seems counter-productive when designing a browser which is designed to be anonymous. The developers of the Tor Project may see this as a trade-off between security and usability, yet the fact that it can still be recovered after deletion is further proof that the browser is incapable of being securely deleted. Since the use of Tor is not currently illegal in most countries, mere identification of the browser may not be considered significant. However, these artefacts may have forensic value when trying to prove that Tor was used over another browser or as part of a wider investigation.

6. Key Findings

This section summarizes some of the key findings.

6.1. Live Forensics

- Four processes can be attributed to Tor whilst the browser is open.
- Use of the bundled obfs4 proxy (used to bridge connections to the Tor network) can be detected.
- After the user terminates the browsing session, only one Firefox process remains.
- Using the Volatility Framework, the path to the browser and the run time can be identified. Even after the user has closed the browsing window, deleted the browser and logged-out.
- As a result of finding the absolute path to the browser, the username, location of the browser and use of Tor versus Firefox can be confirmed.

6.2. Static Forensics

The following are observations for static forensics:

- The design aim of preventing Tor from writing to disk (Perry et al., 2018) is not achieved in this version.
  - Configuration files, downloaded files, and browser-related data are recoverable from the file system.
  - Significant data-leakage from the browsing session occurred: HTTP header information, titles of web pages and an instance of a URL were found in registry files, system files, and unallocated space.
  - The data-leakage contained the German word for 'search' in reference to a Google search. This hints at the locale of the Tor server used to exit the network (exit relay).
- The Tor Project's design aim of enabling secure deletion of the browser (Sandvik, 2013) is not achieved in this version.
  - References to: the installation directory, Firefox SQLite files, bridging IPs/ports, default bookmarks, Tor-related DLLs and Tor product information were all recovered after the browser was deleted.
  - In a scenario where the operating system paged memory, an instance of a URL from the browsing protocol along with references to private browsing was found.
  - The file state used by Tor to track successful launches of the application was found after browser deletion. Prior to being deleted, this file contained the last run time, however, in the recovered file this timestamp was overwritten. This suggests that Tor manages to overwrite this data.

6.3. Forensic Implications of Using Tor

Acknowledging the above results, a forensic investigator could prove:

- The use of Tor even after a user has attempted to delete the browser.
- The last execution time/date.
- Evidence of pages visited during a browsing session.
- The language of the country where the server used to exit the Tor network resides.
- That Tor was used over Firefox.

6.4. Limitations

Some limitations in the methodology of this paper can be observed after completion. One such limitation is that the networking aspect of Tor was not used to aid the forensic examination. For example, Warren (2017) uses Volatility's netscan plugin to observe open network sockets, determining the TCP ports used by Tor in live forensics. This could be used to further prove the use of Tor. Equally, time constraints in this project meant that the individual files dumped from memory could not be analysed. This is likely to have resulted in further evidence of the browsing protocol. Furthermore, to find the browsing data-leakage seen in this project, a knowledge of unique strings related to the browsing protocol was required. This would be unlikely in a real-life static analysis, although if it was known that data from Tor is leaked to areas of the Registry such as NTUSER.DAT, the same data-leakage could be found. Finally, the experiments performed suffered from a lack of control. With more time, the same browsing protocol and analyses could be performed on a system running Firefox. This would allow conclusions to be drawn about which artefacts were attributable to Tor and which were attributable to Firefox. From this, an assessment of whether Firefox's implementation of private browsing or Tor's privacy-enhancing mechanisms led to the data-leakage observed in the results could be made.

7. Conclusions

The limited research in the area of Tor Forensics discovered during the Literature Review suggested that a need for live forensics had become increasingly important. This was largely due to recent papers concluding that Tor did not write browsing data to disk (Warren, 2017). Prior to this, Darcie et al. (2014) had successfully managed to retrieve files residing in RAM from their browsing protocol, even after the browser was deleted. This coincided with the forensic methodology proposed by Dayalamurthy (2013), which also placed emphasis on live forensics. A new methodology was proposed, which favoured a live forensic analysis followed by a static analysis of the virtualised test machine. This resulted in a more thorough forensic process, allowing the maximum level of knowledge about the browser to be gained in the limited timeframe available.

Although previous researchers had successfully recovered artefacts of Tor-use, a comprehensive analysis of both volatile and non-volatile memory had yet to be completed with most of the existing research concentrating on a singular aspect of computer memory. Likewise, the idea of the Tor Browser being used in a shared-computing scenario had yet to be fully explored. Sandvik (2013) had contemplated this, yet it seemed inadequately addressed in her results. Therefore, it became obvious that the design of an experiment which incorporated the action of a user logging-out should be devised. It was determined that in such a scenario, a forensic adversary would likely conduct live forensics on the shared computer (Tsalis et al., 2017) and the purpose of this experiment would be to determine whether Tor could be identified running on the system and whether evidence from the browsing protocol could be extracted from live memory.

Due to the volatile nature of RAM, acquisition of live memory is rarely possible in the field. This is applicable even in shared-computing environments, as often the user can power cycle a shared computer without consequence. Considering that the intended audience for this project was both users of the browser and forensic investigators wishing to analyse it, the omission of a subsequent static analysis would constitute an incomplete methodology. This is especially true as a large number of Tor users will likely use their personal computer which could be subject to seizure by a forensic adversary. Therefore, the multi-faceted experimental design was required. This proved successful in the end as many unexpected results were born from the static analysis, an aspect which may have been omitted if too much reliance had been placed on the results of previous research.

The technique of indexing the hard drive and applying keyword searches based on known Tor artefacts and the browsing protocol was simple yet is something that the browser should protect against. In particular, Perry et al. (2018) state explicitly in their design philosophy that browsing data leakage should not happen. This indicates that the Tor Browser does not adequately protect the user from a forensic adversary.

References

Aggarwal, G., Bursztein, E., Jackson, C., Boneh, D., 2010. An analysis of private browsing modes in modern browsers, in: Proceedings of the 19th USENIX Security Symposium, USENIX Association. pp. 1–8. doi:10.1109/ISSA.2013.6641058.

Darcie, W., Boggs, R.J., Sammons, J., Fenger, T., 2014. Online Anonymity: Forensic Analysis of the Tor Browser Bundle. Technical Report. Marshall University. URL: http://www.marshall.edu/forensics/files/WinklerDarcie_ResearchPaper_8-6-141.pdf.

David, A., Hargreaves, C., 2012. An Analysis of Hotmail Artefacts in Firefox, in: 6th Cybercrime Forensics Education and Training Conference, Canterbury, UK. pp. 1–20. URL: http://dspace.lib.cranfield.ac.uk/handle/1826/8085.

Dayalamurthy, D., 2013. Forensic Memory Dump Analysis and Recovery of the Artefacts of Using Tor Bundle Browser: The Need, in: Australian Digital Forensics Conference, pp. 71–83. doi:10.4225/75/57b3c7f3fb86e.

Dewey, C., 2013. Everything We Know about Ross Ulbricht, the Outdoorsy Libertarian behind Silk Road. The Washington Post. URL: https://www.washingtonpost.com/news/the-switch/wp/2013/10/03/everything-we-know-about-ross-ulbricht-the-outdoorsy-libertarian-behind-silk-road.

Epifani, M., Scarito, M., Picasso, F., 2015. Tor Forensics on Windows OS, in: DFRWS EU, Dublin. URL: https://www.dfrws.org/sites/default/files/session-files/pres-tor_forensics_on_windows_os.pdf.

Findlay, C., Leimich, P., 2014. An Assessment of Data Leakage in Firefox under Different Conditions, in: 7th International Conference on Cybercrime Forensics Education & Training (CFET 2014), Canterbury, UK. URL: https://www.researchgate.net/publication/330925976_An_Assessment_of_Data_Leakage_in_Firefox_under_Different_Conditions.

Greenwald, G., MacAskill, E., Poitras, L., 2013. Edward Snowden: The Whistleblower behind the NSA Surveillance Revelations. The Guardian. URL: https://www.theguardian.com/world/2013/jun/09/edward-snowden-nsa-whistleblower-surveillance.

Montasari, R., Peltola, P., 2015. Computer Forensic Analysis of Private Browsing Modes, in: Global Security, Safety and Sustainability: Tomorrow's Challenges of Cyber Security. ICGS3 2015. Communications in Computer and Information Science, vol 534. Springer, pp. 96–109. doi:10.1007/978-3-319-23276-8_9.

Mozilla.org, 2019. Private Browsing: Use Firefox without Saving History. URL: https://support.mozilla.org/en-US/kb/private-browsing-use-firefox-without-history.

Noorulla, E.S., 2014. Web Browser Private Mode Forensics Analysis. Msc. Rochester Institute of Technology. URL: https://scholarworks.rit.edu/theses/8451/.

Nordberg, L., 2015. The State of Academic Research in Tor, IFIP Summer School, Karlstad, Sweden. URL: https://www.frisc.no/wp-content/uploads/2015/05/tor-finse15.pdf.

Perry, M., Clark, E., Murdoch, S., Koppen, G., 2018. Design and Implementation of the Tor Browser. Technical Report. The Tor Project. URL: https://www.torproject.org/projects/torbrowser/design/.

Roussas, G., 2009. Visualization of Client-Side Web Browsing and Email Activity. Msc. Naval Postgraduate School, Monterey, CA. URL: https://core.ac.uk/download/pdf/36698452.pdf.

Sandvik, R.A., 2013. Forensic Analysis of the Tor Browser Bundle on OS X, Linux, and Windows. Technical Report. The Tor Project. URL: https://research.torproject.org/techreports/tbb-forensic-analysis-2013-06-28.pdf.

Shepherd, E., 2018. The Places Database. Mozilla Developer Network Web Docs. URL: https://developer.mozilla.org/en-US/docs/Mozilla/Tech/Places/Database.

Torproject.org, 2018. Tor: Overview. URL: https://www.torproject.org/about/overview.html.en.

Tsalis, N., Mylonas, A., Nisioti, A., Gritzalis, D., Katos, V., 2017. Exploring the Protection of Private Browsing in Desktop Browsers. Computers & Security 67, 181–197. URL: https://www-sciencedirect-com.ezproxy.napier.ac.uk/science/article/pii/S0167404817300597, doi:10.1016/J.COSE.2017.03.006.

Volatility Foundation, 2017. Command Reference. URL: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference.

Warren, A., 2017. Tor Browser Artifacts in Windows 10. Technical Report. SANS. URL: https://www.sans.org/reading-room/whitepapers/forensics/tor-browser-artifacts-windows-10-37642.