Digital Investigation 11 (2014) 20–29

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

Private browsing: A window of forensic opportunity

Howard Chivers

Department of Computer Science, The University of York, Heslington, York YO10 5GH, UK

ARTICLE INFO

Article history:
Received 25 October 2013
Accepted 22 November 2013

Keywords:
Digital forensics
Internet Explorer
Microsoft windows
Database
Carving

ABSTRACT

The release of Internet Explorer 10 marks a significant change in how browsing artifacts are stored in the Windows file system, moving away from well-understood Index.dat files to use a high performance database, the Extensible Storage Engine. Researchers have suggested that despite this change there remain forensic opportunities to recover InPrivate browsing records from the new browser. The prospect of recovering such evidence, together with its potential forensic significance, prompts questions including where and when such evidence can be recovered, and if it is possible to prove that a recovered artefact originated from InPrivate browsing. This paper reports the results of experiments which answer these questions, and also provides some explanation of the increasingly complex data structures used to record Internet activity from both the desktop and Windows 8 Applications. We conclude that there is a time window between the private browsing session and the next use of the browser in which browsing records may be carved from database log files, after which it is necessary to carve from other areas of disk. It proved possible to recover a substantial record of a user's InPrivate browsing, and to reliably associate such records with InPrivate browsing.

1. Introduction

The release of Internet Explorer 10[1] marked a significant change in how Internet history and cache data are stored within the file system; the binary historical formats which have been widely documented in the forensic community (e.g. (Jones, 2003)) were replaced by a high performance database technology known as the Extensible Storage Engine (ESE). This database is used to support a range of other applications, including Windows Search, and was the subject of a previous paper in which we described the results of carving for deleted ESE database records from the Search Database (Chivers and Hargreaves, 2011). The carving tool is now known as *ESECarve*[2] and has subsequently been used to assist a number of real investigations.

InPrivate Browsing is an Internet Explorer mode which is launched by the user in a separate browsing window; the claim is that this mode *"prevents local storage on your computer"*(Microsoft, 2012). The prospect of evidential recovery from private browsing is of considerable forensic interest, and several researchers have reported using string searches to identify artifacts of interest; others have used ESECarve to survey residual browsing histories and suggest that such evidence is recoverable (Malmström and Teveldal, 2013).

The prospect of recovering evidence from InPrivate browsing prompts questions, including when such evidence can be recovered, the implications for seizure tactics, where the evidence can be found, and if it is possible to prove that a recovered artefact originated with InPrivate, as opposed to normal, browsing. This paper reports the results of experiments which answer these questions, and also provides some explanation of the increasingly complex data structures used to record Internet activity from the

E-mail addresses: hrchivers@iee.org, howard.chivers@york.ac.uk.

[1] We acknowledge Microsoft copyright in terms used in this paper to describe Microsoft products, including: Windows, Windows 8, Metro Application, Internet Explorer, Windows Search, InPrivate Browsing, Extensible Storage Engine (ESE), and esentutl.

[2] ESECarve is available from the author for forensic investigation, education and research.

1742-2876/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.diin.2013.11.002

desktop and by Windows 8 Metro Applications. Results of forensic interest include:

- InPrivate browsing artifacts can be positively identified using the *Type* field in cache content records.
- Pull-the-plug seizure may allow the recovery of InPrivate browsing records from the database file (*WebCacheV01.dat*); however, it may also result in a database that cannot be recovered for use with application interface-based tools because log files have not been completely written to disk.
- The window of opportunity for the recovery of InPrivate artifacts from database log files extends to the next time the browser is opened for use. During this window substantial recovery is possible, afterwards these data are securely deleted.
- Browsing evidence may also be recovered from areas of disk apart from normal database files and logs; this may persist for some time.
- The table structure within the database includes separate records for applications, allowing some fine grain distinctions to be made about the use of the computer.

The remainder of this paper is organized as follows: Section 2 briefly describes the Extensible Storage Engine and Data Storage in HTTP/HTML, both of which are needed to understand the descriptions of database behaviour and browser artifacts that follow; this is followed by a review of publications related to private browsing. Section 3 describes how the experiments used to determine browser behaviour were conducted. The next sections present detailed results; Section 4 describes the files that support Internet Explorer and how the database tables are structured, Section 5 describes the conclusions of experiments to determine if InPrivate browsing records can be recovered. Findings are further discussed at Section 6 and the paper is concluded in Section 7. Appendixes describe the restoration of a database to a clean state, and record carving using *ESECarve*.

1.1. Terminology

This paper uses the term ‘record’ to mean a single database record or row. Browsing records include a URL with an associated date and time. They document a single Internet action; examples include a cached response to an HTTP request, a download, a history record of a visit to a domain, or the storage of a cookie. The term ‘browsing record’ here should not be taken as an implication that it originated from human action.

2. Background

2.1. Extensible Storage Engine (ESE)

The Extensible Storage Engine is documented on-line by Microsoft (Microsoft, 2013), and details of its internal structure have been published by Joachim Metz (Metz, 2010). A previous paper (Chivers and Hargreaves, 2011) provides an overview of the database and the reliability of records recovered by carving. This section briefly describes transaction processing, as a background to why database records are often found in log files or in cached memory such as the *pagefile*.

The Extensible Storage Engine is designed to process high transaction volumes and be recoverable from failures, such as a system crash while data are being written to disk. A typical transaction sequence is shown in Fig. 1, with the file names currently used by Internet Explorer 10.

An incoming transaction is first held in a memory log cache (1), then any necessary database pages are brought into memory (2) and the transaction applied (3); as soon as possible the updated database record is written to the log file (4). Eventually the database file is updated with the page which contains the new transaction. A database whose file has not been fully updated is known as *dirty*. On a normal shutdown the log cache is flushed to disk, whereas the database file (*WebCacheV01.dat*) may not necessarily be updated and may be left in a dirty state.

If the database is dirty it must first be *recovered* (in ESE terminology), before it can be accessed using the database application interface. This process recovers the database to a consistent state by replaying log transactions from a known checkpoint. The checkpoint is stored in a *V01.chk* file and the logs are recorded in files numbered in a hexadecimal sequence (e.g.*V010009.log, V010000A.log*) together with the current working log (*V01.log*). When the current working log is full it is renamed to the next name in the hexadecimal sequence, and a new V01.log file is created. Logs that are no longer needed are deleted.

Both database and log records use the same record format, so records from either can be recovered by carving.

It is evident from this process that database records are found in memory and perhaps therefore in the pagefile, in log files, and in the database file. The action of allocating and freeing files for logs may also leave records in unallocated or slack space in the file system. Because this is a high

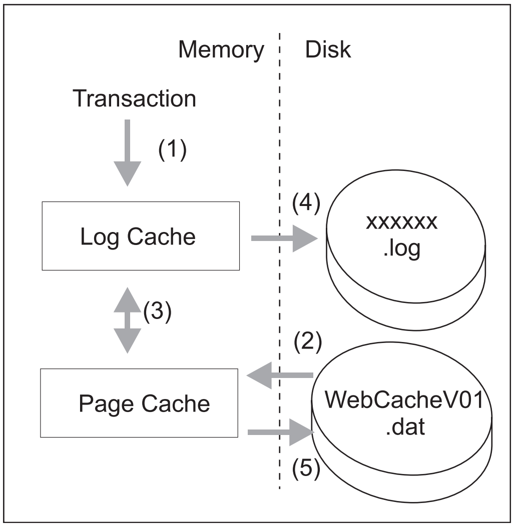

Fig. 1. *The Propagation of Transaction Data into Disk Files.* Transactions are cached in memory and written quickly to log files; the database file is subsequently updated from memory or recovered from the logs.