Private Browsing Forensic Analysis of Google Chrome™

by

Aung Thu Rha Hein (g5536871)

A proposal submitted in partial fulfilment
of the requirements of
Master of Science (Computer Science)

Faculty of ICT
Mahidol University

September 2015

Abstract

Usage of digital devices is increasing nowadays and these digital devices can be used to commit crimes in real and virtual world. Also, browser is one of the most common application people use not only to access information from the internet but also to get access to online services. There are so many forms a person can commit digital crimes though the browser such as in the cases of child pornography, phishing, blackmailing, hacking, electronic harassment and so on.

Apple inc. introduced a safer way to browse the internet without leaving traces to the host computer called *private browsing*. As for ordinary non-technical end-users & criminals, they can access to the internet with private browsing mode and commit digital crimes which can obscure their activities and browsing records. So, the question rises in digital forensic research community is that "*Is this browsing mode becoming anti-forensic technique?*". Researchers & DF practitioners have been studying and analyzing the possibility of recovering data from private browsing mode since 2010.

In this research proposal, it will address the issues of recovering data remnants from private browsing mode. it will be conducted experiment to study the possibility of recovery of data artifacts from Google Chrome's incognito mode and proposes a framework-based analysis approach to recover digital evidence from it. Google Chrome is the most advanced browser in terms of technology and also the one leading the browser market and overall usage. Moreover, it is necessary to carry out experiment with advanced digital forensic techniques and tools to discover behavior of private browsing mode. For these reasons, this study will be a good contribution to the Browser Forensic research community. As preliminary experiment, persistent storage experiment and volatile memory experiment to investigate the possibility of recovering data from Chrome's Incognito mode. From the analysis result from preliminary experiment, a model to recover and analyze private browsing artifacts will be proposed.

Table of Contents

List of Figures

List of Tables

1 Introduction

Web browser is the most common application which is used by end-users to get access to various applications and services. Digital artifacts left by browser contains important and valuable information such as user's internet activity and personal information. From point of view of forensic study, these information contain high forensic value and it is essential for forensic analysts to discover and acquire these information to turn into digital evidence. Since 2003, forensic researchers have been working on browser related forensic study for four major browsers on different perspectives.

At 2005, Apple inc. introduces a new browsing mode called 'private browsing' and it said that it can leverage user's privacy to another level by wiping out traces of user activity. However, it becomes a blockage for forensic analysts to recover data remnant from the private browsing sessions. E-criminals can take advantage of lacking of forensic tools and methodology, and commit crimes via accessing the internet with this browsing mode.

For these reasons, forensic researchers are studying behaviour and characteristics of private browsing mode to discover the possibility of recovering browsing records and user activity from private browsing mode for almost a decade. However, there are only few research works that analyze the browser behaviour and data structure systematically. Most of the research works ended with a conclusion of possibility of recovering digital remnant from private browsing session without systematic study of data structures and private browsing mode.

1.1 Motivation

Nowadays, private browsing mode is available in all major browsers and users are using it to hide their activities within browsing session. The criminals can also use this function simply to remove their activity tracks from the host system. Another scenario is that organization uses virtualization environment to isolate high priority and sensitive processes from the rest of the system environment (e.g., a browser solely run for banking and financial system.). For these virtualization systems,

recovering the process and activity within private browsing session can help a lot in post-mortem and live forensic analysis.

However, the previous research works on private browsing prove that it is not secure as it is supposed to be. There is a possible chance of recovering the private browsing activity which is proven by these research studies [1,7,9,10,11]. Among popular browsers, the market share of Google Chrome browser is increasing exponentially. From 2014 W3Counter July 2015 report [23], Chrome is leading the browser market with 46.2\% based on 34,339 tracked websites.

Therefore, analyzing the behaviour of Chrome Incognito mode and developing a methodology to discover browsing artifacts and analyzing these data artifacts is a great contribution to the digital forensic research community. Also, the persistence layer of popular browsers except Internet Explorer use *SQLite* database technology to store the browsing records and for this reason, it is possible to continue this research into further private browsing researches.

1.2 Problem Statement

The major issue in digital forensic research is that developing a common and standard methodology for digital artifacts' acquisition and analysis process. The same problem is encountering in browser forensic because it is immature research area and architectural variations of the web browsers. Generally, the digital forensic methodology can be categorized into two subcategories: traditional forensic methodology and non-traditional forensic methodology. The forensic analysis processed for traditional methods are more consistent since this area is more mature than non-traditional forensic approaches, especially in volatile memory forensic area. The forensic researchers are constructing the methodology without many references to each other's work because of their different methodologies and evaluation methods. More details will be pointed out in Related works, Chapter 3.

Even though the basic aim of private browsing mode is the same but the implementations vary across the browsers. This adds significant complexity to investigate the operations and protocols of private browsing mechanism. For browser forensic, it makes the newcomers encounter slight difficulty in

literature review because the process and the research works are different according to browser architecture. The browser companies are trying to fix security bugs and adding new features to stay ahead of competitors and it is likely to release updates quite often. It makes forensic tool builders and researchers to give up their works at some point for technical and economical difficulties. There are no formal documentation about behaviour and process of private browsing mode. It needs to study and analyze with experiment and result from analysis. From the theoretical study of this research, it will provide the current research status of private browsing mode of Google Chrome and also, it can help the newcomers of Browser Forensic research as a starting point.

Tools & method to extract digital artifacts from private browsing mode is still lack for Google Chrome's Incognito mode. Therefore, it is necessary to study and discover the possibility of recovering data artifacts from Google Chrome's Incognito mode.

1.3 Research Objective

The main objective of this research is to propose a method that can recover data remnant of private browsing records from Chrome's Incognito mode. To achieve this goal, the proposed method need to look not only at data storage but also physical memory of the disk image. There will be several open source forensic tools and frameworks applied through preliminary experiment to study Chrome's private browsing. While carrying out the main objective, it will be analyzed the possibility of recovering private browsing artifacts and files from disk image and physical memory. Moreover, data structure and data remnant of Google Chrome will be analyzed.

1.4 Research Method

The research methodology for this thesis will cover both theoretical and practical method. As a theoretical study, overview and status of digital forensic, browser forensic and memory forensic will be covered with existing methods, tools and problems. The literature review part will point out private browsing forensic research trend and existing findings and study.

The theoretical part of this research will carry out the main objective of this research, which is to analyze possibility of recovering private browsing session of Google Chrome from evaluation of frameworks, tools and method from theoretical study.

1.5 Outline

The structure of this proposal begins with background information about digital forensic and browser forensic. The content of the chapter 2 is to introduce digital forensic and its research trend and also a section about browser forensic briefly. Moreover, this chapter will also cover the background of traditional forensic process and non-traditional forensic methodology. A section of memory analysis will also included since it is a immature area of digital forensic area. Chapter 3 will discuss about related works and previous researches on browser forensic, especially focused on private browsing. In this chapter, it will also include comparison of methods and tools researchers used to achieve their objectives and also discuss the limitations and improvements that can be made on their researches. The list of prior findings and results will be listed at the end of this chapter. At chapter 4, the proposed model and experiment methods will be discussed. The result and process of preliminary experiment and its results will be discussed at chapter 5. Chapter 6 will describes the research plan and implementation outline to achieve the objectives of the research. The proposal will conclude with the summary points about contribution and future research works related with private browsing forensic study.

1.6 Summary

In this introductory chapter, it describes the motivation of this research and issues in digital forensic research which need to be addressed by the forensic researchers. The main objective of this research is to contribute the forensic analysis of Google Chrome browser on both persistent and volatile data. There are several research works done for private browsing study on persistent storage. However, this study will validate these known vulnerabilities of private browsing mode still resisting in Chrome version 34.0.1847.131 m. For volatile data forensic analysis, it still lacks to perform a systematic study with updated memory forensic frameworks and tools.

2 Background

This chapter will introduces about literature review and background information of the major aspects of this research projects which are Browser Forensic & Memory Forensic.The fundamental principles, process and methodology of Digital Forensic will also describes briefly. This chapter will also covers about the status of Digital Forensic with its existing open source and proprietary forensic tools. From this section, newcomers into Digital Forensic field will have a overview understanding about the status and nature of Digital Forensic. Browser forensic solely focus on trends and information about browser such as its architecture and forensic trends on it. The last session will discuss about memory forensic, its approach and status.

2.1 Digital Forensic

Digital forensic is a branch of forensic science and it have been applied in various areas as an extension to computer security. For e.g, network forensic post-mortem analysis can be performed after the attack to discover the attack pattern, the event to find out the methodology of the attack. The security system can be updated with the analysis result from forensic tools. Nowadays, people use electronic devices such as mobile, tablet, laptop daily basis and there is a high chance of involving one or more type of devices at criminal scenes. According to *Locard's exchange* principle, the perpetrator of a crime will bring something into the crime scene and leave with something from it such as DNA, murder weapons and electronic devices, which can turn into forensic evidence. The same concept applies for digital forensic. Traditional forensic analysis acquires, examine and extract evidence and events after powering down the system. Forensic analysts typically maintain and verify the integrity of the digital evidence, so it can be presented at the court.

Another well-known application of digital forensic is the process is called *e-discovery,* which both parties at the courtroom exchange electronic stored information (ESI act) . It is extracted and analyzed according to the forensic procedures and constructed as human readable format to assist at legal litigation process.

S. L. Garfinkel [13] states that the golden age of computer forensic is coming to an end. Between 1999 and 2007, the number of forensic research and contribution is increasing exponentially. However, it is coming to an end because the research area of digital forensic is encountering several issue. Unlike other research areas, the principle of 'software ahead of hardware' is reverse for digital forensic. According to hardware advancement, the complexity of implementing forensic software is becoming more sophisticated than ever. The following summary points out the issues and challenges encountering in digital forensic area after golden age:

1. New storage technology such as SSD, cloud storage and growing size of storage device makes the acquisition and analysis process takes more computational power and time.
2. Increasing number of file formats and operating system demands various acquisition tools and the software complexity to implement.
3. Forensic software is difficult to modify and release update patches since operating systems version and most commercial software are releasing updates frequently.
4. Lack of standardization makes the newcomers to this area difficult to continue others' works
5. Acquisition and data processing failed due to anti-forensic techniques and pervasive encryption

Table 1 describes the general characteristic comparison of digital forensic research. As S. Garfinkel[13] mentioned, the golden era of Digital Forensic is coming to an end and Table 1 reflects challenges and problems encountering at the new era. It can be seen that there are many research challenges and problems to solve for digital forensic area.

Era	**OS**	**File Format**	**Computing Architecture**	**Storage Architecture**	**Tools**
1997-2007	Windows Dominance	limited file formats	Stand-alone PC, Centralized Architecture	standard cable interfaces (SATA,PATA)	commercial tools have high capability
2007- recent	OS (Mobile, Windows)	Increasing file formats	Client/Server, Cloud computing	Flash, Cloud Storage	lack to modify and update

Table 1: Characteristic comparison of digital forensic

2.1.1 Digital Forensic Methodology

The basic traditional forensic process includes 4 distinct phases and the detailed steps of each phase will be varied upon specific methodology [14]. It also states that there are only 3 major forensic methodologies which are basic forensic methodology, European CTOSE methodology and Data Recovery UK (DRUK). In that research work, the author proposes the recommended method to standardize computer forensic methodology.

Figure 1 represents the basic forensic model [14] for all data sources. At first, the identification phase is to find out the information about the digital evidence and pre-planning for further process. It was followed by acquisition phase to copy digital evidence without any changes. The acquisition techniques will be differed according to the type of storage media and environment. The authentication step is to verify the integrity of the digital evidence before making analysis. It's a critical step because the inaccuracies of the obtained information can be rejected by the court. Making analysis of the digital evidence will be varied according to the evidence that analysts searching into. The final phase, presentation stage is to convert the analysis result into report or human readable format which can be presented at court.

Figure 1: Basic forensic process flow

At Digital Forensic Research workshop at 2011, the process of forensic analysis is discussed and agreed upon the attendances of the workshop who are digital forensic practitioners and researchers [12]. Figure 2 presents the major categories of the forensic process and for each category, there are forensic techniques and methods to achieve the objectives of the process.

Figure 2: Digital forensic investigation process(DFRW,2011)

2.1.2 Conventional digital forensic methodology & Non-traditional digital forensic approach

The digital forensic study can be branched into traditional forensic approaches and non-traditional forensic approaches. Traditional approaches tend to capture, study and analyze data from least volatile system data. Figure 3 describes the scale of volatility of digital evidence [24]. It shows that conventional forensic approach works on persistent storage, remote storage and archived data.

The general procedure of traditional forensic analysis methodology is power down the system, acquire the image with integrity, analyze the acquired digital image to turn into digital evidence. However, traditional forensic method encounters limitations because of technology advancements, e.g. increasing storage size of hard drives, memory resident software applications, and so on [15]. Therefore, forensic analysts are looking through most volatile data to analyze information which can not be obtained from conventional methods.

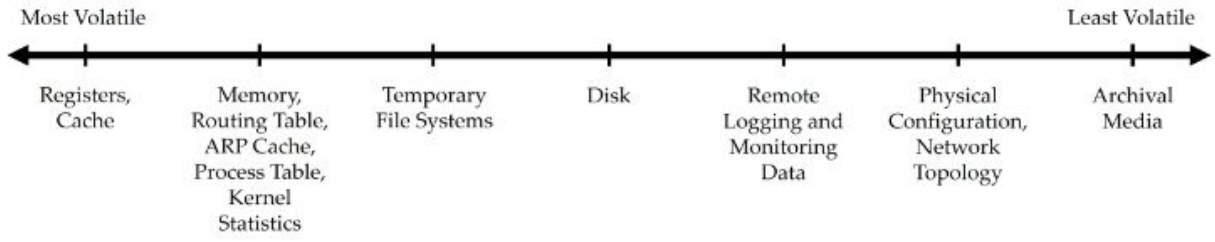


Figure 3: Order of volatility of digital evidence[24]

Starts from around 2005, several researches conducted in memory forensic field and there are many methods to acquire and analyze volatile memory of a system. DFRWS 2005 Forensics Challenge is focused on memory analysis and it was played a big role in the initiation of memory forensic research. At this time of writing, there are several open source memory forensic tools and frameworks to rely on for practical usage. S. Vömel and F. C. Freiling [15] reviews and compares memory acquisition and analysis techniques for the purpose of selecting adequate techniques for security practitioners and forensic researchers. Another point is that most of the research works and tools are focusing on Windows operating system family since Microsoft OS family is the most popularity and dominance

one among operating system market. Among the research works, most of them are conducted to retrieve operating system level information and few works have been worked on application level analysis [16]. According to the literature review, there are few works that experimented on analysis of volatile data on applications with valuable information such as email apps, messaging apps and video conferencing apps.

2.13 Digital Forensic software and tools

At the time of writing, there are several commercial and open source acquisition tools and also the frameworks and software suites for both persistent and volatile forensic analysis. However, most of the open source tools are stopped releasing updated versions to support latest Operating Systems because of the financial problem, rapid update releases of software platform and other technical feasibility issues. Table 2 presents list of popular proprietary software suites among forensic practitioners. AccessData and EnCase provide solutions for various security and forensic tasks start from disk imaging, disk analysis, memory analysis and so on.

Unlike open license software, these proprietary software bundles various tools to gain the most market shares. For open source tools, there is no software suite provide various solutions. On the other words, the user has to select suitable tools to achieve particular tasks. For e.g, FTKImager for image acquisition and SleuthKit for analysis and reporting. Another important point is that these tools are difficult for non-technical person to use. It requires specific training programs to get familiar with.
Compared with closed source software, there are also several open source software existed for traditional and non-traditional forensic analysis. Table 3 describes the status of these applications and frameworks, which mostly relies by forensics researchers and practitioners.

Tool	Platform	Usage	Development Status
AccessData	Windows	Multipurpose tool (Digital Forensic, Cyber Security, e-Discovery)	Active
EnCase	Windows	Multipurpose tool (e-Discovery, Data Security,	Active

		Digital Forensic)	
WindowsSCOPE	Windows	Memory Forensic (Acquisition and Analysis), Live Forensic, Computer Forensic	Active
Katana	Mac, iOS	Device Acquisition, Analysis, Mobile Forensic	Active

Table 2: Popular proprietary forensic frameworks and software

Tool	**Platform**	**Usage**	**Development Status**
SleuthKit & Autopsy	Windows/Linux	Analysis of file system from various digital image formats	Active
Digital Forensic Framework(DFF)	Windows/Linux	A platform to analyze forensic recovery on disk and volatile memory	Active
Volatility	Windows/Mac/Linux	Memory analysis framework with plug-ins	Active
Forensic Toolkit (FTK)	Windows	Court accepted Disk analysis tool and one of AccessData products	Active
SANS SIFT	Ubuntu(Linux)	A Linux distro that pre-installed open source forensic tools	Active
Kali Linux	Debian(Linux)	A bootable linux OS with security, penetration testing & forensic tool	Active

Table 3: Open-source forensic software

2.2 Browser Forensic

Browser forensic is one of the digital forensic branch focuses on discovering browsing records and user activity for forensic purpose. Nowadays, there are 5 major browsers which are Internet Explorer, Google Chrome, Apple Safari, Mozilla Firefox, and Opera. Each browser have different architectures and implementation with slight similar components and technology among some of them. For e.g., IE uses *ESE* as storage mechanism but Chrome, Safari, and Firefox use SQLite to store browsing information. Moreover, browsers tend to release new updates often to improve performance, to fix security bugs and to add new features. For this reason, the work of browser forensic researches are not

up to date at the time of publishing and also, changing the technology of one component to another makes previous research works unattributable.
In this section, the overview of the generic architecture of the browser, its data structure, private browsing mode, and existing data recovering methods for browsers will be explained.

2.2.1 Browser Architecture

As a generic architecture of browser, it consists of 7 components which are

1. ***User Interface***: it includes UI components of browser such as menu bar, buttons and so on
2. ***Browser engine***: it interacts and associates between the UI and the rendering engine.
3. R***endering Engine***: it is responsible for rendering web resources and displaying requested content. Mostly, requested web resources are HTML, CSS & client-side scripts such as Javascript and displays parsed contents on the browser's screen.
4. ***Networking***: it is a network layer to handle network calls such as HTTP requests, using different implementations depends on OS platforms.
5. ***JavaScript Interpreter***: Javascript Engine to parse and execute JavaScript code.
6. ***UI backend***: To present UI elements on User Interface, UI Backend engine creates basic web controls and widgets. This backend draws all UI elements, which are not OS platform Specific.
7. ***Data Storage***: This is a persistence layer. The browser may need to save all sorts of data locally, such as cookies, website caches and thumbnails . Browsers also support storage mechanisms such as localStorage, IndexedDB, WebSQL, and File System.

For forensic research on browsers, the researchers are typically focusing on the data persistence layer. The detailed information about data persistence layer is unpublished from the vendors due to security reasons and without research experiment and analysis, it is not feasible to find out the procedure of the data transactions and interactions between the browser and host machine. The analysis method to study browser architecture will be covered in Chapter 4. Among previous research works, H.Chivers[3] study the data structure of private browsing mode of Internet Explorer systematically and proposed a method to carve browsing records. Most of the existing researches didn't study thoroughly about the data structure of browser thoroughly. The browser forensic researchers mainly

focus on finding information through keyword searching through digital images and in volatile memory.

2.2.2 Private Browsing

Private browsing is introduced by Apple.Inc to provide an additional feature for Safari user to surf the internet without leaving traces and data artifacts on the system. At the time of writing, all 5 major browsers provide that functionality. There are only handful works that study the security of private browsing systematically [11]. The study of private browsing mode in 4 major browsers in 2010[1] shows that there are several vulnerabilities in private browsing modes. Another motivation to study private browsing mode is that the attackers and e-criminals can use this mode easily to hide their traces and activity. Since it's easy to use without advanced technical skills, there is a high possibility that common criminal minds will cloak their identity and get away with the crimes. It's becoming a controversial issue among forensic analysts that private browsing is becoming anti-forensic techniques or not.

From security threat model study[11], which is an improved version of previous work[1], the researchers mentioned that the attack models can be categorized into local and remote attackers. To sum up, the study shows that the vulnerable points of private browsing mode can be come from the local attacker who gains access to the system. The attacker pre-installs the third-party tools to the user's machine. The remote attack is the attack comes through internet traffic. The study mentions that the goal of the attacker is to reveal that the user is in private browsing mode or not.

For this research, it is focused on the private browsing mode of Google Chrome, namely as *Incognito* mode. According to design section of Chromium developer guide, Google Chrome uses *WebKit* as rendering engine until ver. 27 and it is switched to *blink*, that is a layout engine based on WebKit. For data persistence, Chrome uses SQLite ver.3 to store browsing history, passwords, and cookies. SQLite is a file based storage technology and it is efficient in terms of performance and data transactions. For security reasons, there is no formal developer guideline and documentation about *Incognito mode.* However, the Chromium project also provides the specification of Google safe browsing protocol

which is a part of private browsing mode. The following points describes basic characteristics of the protocol;

1. All URLs check against malware and phishing before loading.
2. URL is hashed and check synchronously against in-memory prefix list.
3. SafeBrowsingResourceHandler controls resource loading to prevent malware injection.
4. When a resource is suspected as unsafe, an interstitial page (SafeBrowsingBlockingPage) is interrupted.
5. URL checking for file downloads.
6. Hash checking for file downloads.
7. For phishing prevention, each web pages is checked against the safe browsing list from server.

Google[2] promises that its private browsing mode doesn't record history, leave cookie information and run parallelly with regular browsing mode with security sandboxing. It clearly mentions that Chrome can't control websites from recording information of the user with private browsing. In other word, there is a possibility of remote attacker can trace user's information. D. Ohana and N. Shashidhar [9] proves from their research result that the indicator of Chrome's Incognito mode can be discovered and also a possibility to recover browsing information from memory dump. The future work section mentions that memory analysis methods should be improved for this research and also said about lack of carving mechanisms and forensic tools for this matter.

2.3 Memory Forensic

Memory forensic is a non-traditional forensic method to discover volatile data. Technology advancement in network, computer systems, and mobile operating systems demand in need to look information that can not recover from persistence storage. The memory forensic research area is growing since 2005 and there are several acquisition tools and frameworks already been published. One bottleneck of memory forensic is that tools and techniques are worked with only specific version of operating systems and architecture. The security practitioners and forensic analysts have to evaluate memory analysis and acquisition techniques according to the environment and operating system.

In this section, It will describe the memory forensic sub-processes in order. The first stage is to acquire memory image and followed by the necessary analysis process by using tools, frameworks, and custom scripts.

2.3.1 Memory acquisition

According to [6], memory forensic is the process of capturing, dumping and sampling contents of volatile memory to nonvolatile storage. The acquisition is more dynamic in nature compared with traditional persistent storage acquisition. Preservation of non-volatile memory is the controversial issue because integrity and reliability of data can affect the forensic analysis process. Traditional acquisition process usually turns down the system and retrieve the digital image to reduce the distortion of data. Memory acquisition is not possible to perform the process with the same tradition since it can destroy the data of volatile storage (i.e., RAM, device memory). Therefore, the appropriate tools and technique need to select according to the needs of the case and environment.

A survey research [15] evaluates and compares the memory acquisition techniques and establish the model to choose a suitable technique. The memory techniques can be categorized into hardware based and software based techniques. Figure 4 describes the existing memory acquisition techniques. The hardware-based techniques access directly to the memory of the system to copy the memory image. There are two methods to save physical memory using hardware devices.The first method is saving memory content as a PCI device which copy physical memory through Direct Memory Access(DMA). Another method uses FireWire bus (IEEE I394 interface) to duplicate DMA content. This hardware based approach is more reliable and available. However, the hardware-based acquisition devices such as Goldfish, WindowsScope are quite expensive. The latter technique uses functionality provided by operating systems to dump physical memory. For software-based techniques, the answer of the question '*is the target system using virtualization or not*' plays a critical role. If the machine is in virtualization platform, a middleware software layer between hosted machine and virtual machine, called *Virtual Machine Monitor(VMM)* manages hardware resources and provides a monitoring interface of the installed virtual machines. Some virtualization technology provides a feature to dump the physical and virtual memory of virtual machines. For e.g., after sending suspended stage

VMware-based virtual machines, VMWare create memory (.mem) and snapshot (.vmss) files and also QEMU emulator provides a monitoring command dumping the current stage of memory and duplicate as a disk content.

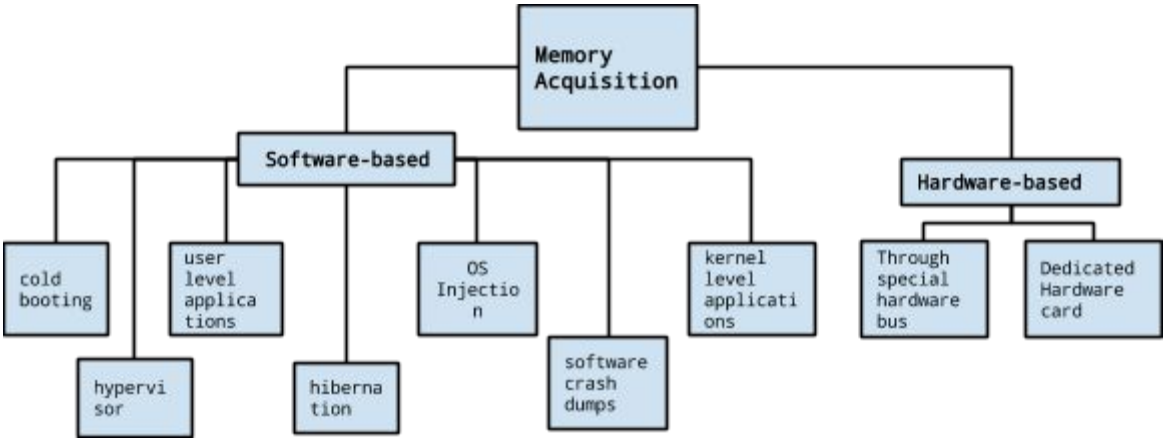


Figure 4: Categorization of memory acquisition techniques

Another consideration point in memory acquisition is that operating systems do not provide a native mechanism to acquire the physical memory [6]. The system might be left with unexpected stage after the process. For this reason, there are three main factors which need to consider when evaluating the memory forensic tools and the quality of acquired memory images, which are fidelity, reliability and availability. Firstly, the nature of RAM, atomic operation, is dynamic and the state is changing even the system is idle. Likewise, it can be also changing while the memory acquisition process is in progress. It can affect the *fidelity*, "the correctness of the memory content" and *reliability*, "the trustworthiness of the acquired memory images". The corrupted memory dump is not very useful because the operations are still in running stage and the analysis software can't extract information from it. The acquisition tool evaluation model [15] states that the ideal acquisition technique should have both high *atomicity* and *availability*. OS Injection, hibernation file, and virtualization based techniques have high atomicity with slight variations in availability. Secondly, availability in memory acquisition refers to the usability of tools in various platforms and operating systems. Vömel & Freiling [15] conducted a classification of existing acquisition techniques by these two criterias and table 4 describes the summary of their finding.

Acquisition Technique	Atomicity	Availability
Dedicated hardware device	High	Low
Special hardware bus	Medium	Medium
Virtualisation	High	Medium
Software crash dumps	High	Low
User level applications	Low	Medium
Kernel level applications	Medium	High
Operating system injection	High	Medium
Cold booting	High	High
Hibernation file	High	High

Table 4: Acquisition Criteria and Acquisition technique evaluation

As it mentioned before, hypervisor provides a way to dump physical and virtual memory. QEMU is a KVM hypervisor based virtualization emulator and from QEMU console monitor, it is possible to send commands to VM such as suspend, stop and pause the state of machine. Moreover, it is also capable of taking a snapshot of physical and virtual memory by addressing the starting and ending memory offsets in a byte. The basic usage can be seen at figure 5. The limitation of this method is that the forensic analyst have to get access to the physical machine local or remotely. To detect virtual machine control structure (VMCS) of single or nested VMs within a hosted machine, a volatility plugin called *Actaeon*[18] is implemented to achieve the goal. However, the memory dumps can not be distinguished by virtual machines.

```
$ pmemsave <start offset> <end offset> <output name>
$ memsave <start offset> <end offset> <output name>
```

Fig 5: Basic usage of QEMU-Monitor to acquire memory dumps

2.3.2 Memory forensic Analysis

After the memory image is preserved, the analysis process will be performed to turn it into digital evidence. Investigating memory artifacts can be classified into several categories according to the type of artifacts looking from memory images such as process analysis, file analysis, network analysis, malware

& rootkit analysis and application analysis. The basic analysis of the content in physical memory can be performed with old school analysis methods such as using strings.exe tool or running grep searches to obtain keywords and readable texts [19]. The keywords can be password strings, email-address, and other valuable information. The forensic analysts focus more on finding the "known" expected information from the memory image. The output of this method can be false positives and can not obtain information such as a region of the keywords in memory image and related processes using the keywords. In other words, this method is short in terms of looking related and overall context of the extracted piece of information. At 2009, S. M. Hejazi, C. Talhi, M. Debbabi [22] proposes a new method to look for sensitive information from memory content. The conventional strings searching method lacks in finding hidden sensitive information because the forensic analysts looking for specific data. The proposed method, 'fingerprints' search method looking through application related constants from memory content. However, it still needs to review application source code and for proprietary software, this method is not quite useful.

After DFRWS 2005 Memory Challenge, the concept of memory analysis move from looking keyword strings from memory to analyze processes in memory and identify the structure of the process. The purpose is to analyze the nature of data in memory content and to identify the offsets of the data types. By locating process and other valuable objects from memory, the forensic analysts can focus on the essential processes and its data to search for the evidence. With Volatility framework, the process of analysing running process from memory dump is a straightforward task to do.

At this time of writing, there are two major methodologies in identifying running processes from memory images. The first methodology is the result of DFRWS 2005 Memory Challenge. One of the winners of the challenge, Chris Betz, proposed a tool called *memparser* to enumerate process lists and extract information from each process[21]. Generally, this methodology locates and enumerates the process list by fixed offsets and values. The starting location is global kernel variable. Figure 6 describes high-level structure of a process. The main object is called *_EPROCESS*, which is an Windows object that handle processes. Each process have threads to execute code, handle table to manage kernel resources, loaded shared libraries and modules and VADs tree to manage memory region used by each

processes. The _EPROCESS structure contains many valuable entries to analyze running and terminated process. One of the members is called *ActiveProcessLinks ,* that is a double linked list that points to previous and next _EPROCESS blocks. By enumerating these pointers from _EPROCESS block, it is possible to find list of process. Another important kernel structure is *PsActiveProcessHead,* that points to the starting point of the process. Figure 7 presents the association of *ActiveProcessLinks* and *PsActiveProcessHead.* Memparser, KnTTools are based on this method to find a list of process from Windows system. One of the drawbacks of this method is that the Windows Object can be varied upon Windows version.

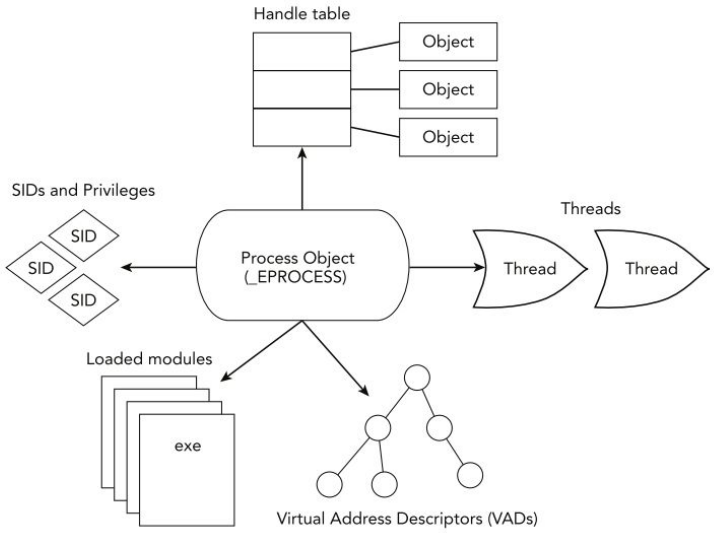


Figure 6: High-level diagram of basic process structure[6]

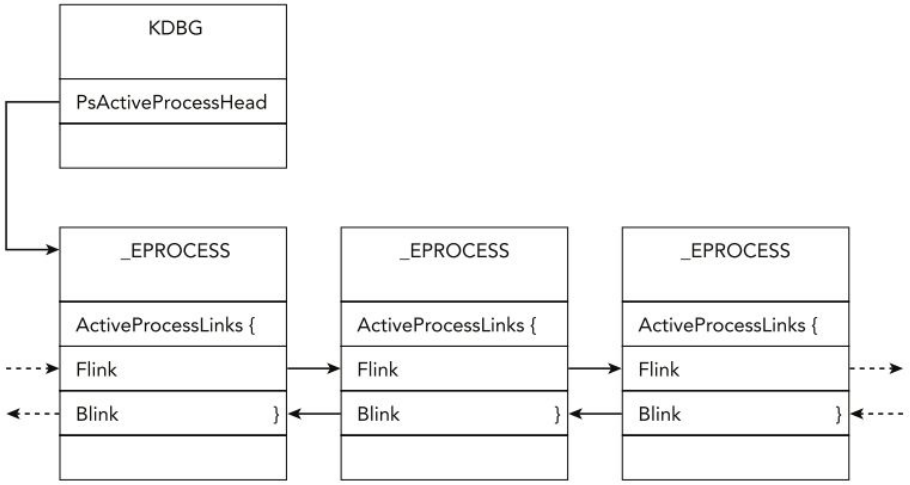


Figure 7: Processes are linked by *ActiveProcessLinks* pointed by *PsActiveProcessHead*[6]

A. Schuster[21] implements a different approach to parse the processes from memory. His method is by enumerating _EPROCESS structures from memory image by scanning with signature-based approach. The researcher identifies rules or fingerprints to enumerate a list of process from memory content physically. For hidden and exited processes, there are several techniques such as pool tag scanning, big pool scanning for memory allocations with larger than a page size, which is 4096 bytes.

Volatile data can also be in persistent storage because modern operating systems write system operations' data such as hibernation, paging and so on. Forensic researchers have to consider these alternate volatile sources because it can be the only source in some circumstance such as the target machine is power off. Acquiring these files can be done with conventional acquisition process by locating the files within the bit-by-bit copied disk image. Hibernation file can be analyzed with memory analysis frameworks. For page files, a tool called page_brute[6] can split the file into page-sized chunks and analyze with Yara rules. Page files are not likely to have consecutive pages and for this reason,it is unlikely to find out data size greater than 4kb. Yara is a pattern matching tool aimed for malware researchers, but it can be used in any case to search textual and binary pattern. Crash dump file analysis can be achieved by using Windows Debugger tool or manual analysis [15]. This technique can leave the system unstable.
Table 5 is the sources to investigate windows related volatile information on disk image:

Volatile source on disk	Possible file name	Analysis possibility
Hibernation file	hiberfil.sys	Yes
Page files	pagefile.sys	Yes (<4kb)
Crash dump files	memory.dmp	Yes

Table 5: Volatile memory on disk

2.4 Summary

To sum up, this chapter covers the background of digital forensic research area including research challenges and methodology comparison. Moreover, a section of browser forensic and private

browsing overview is introduced since it's the specific area of this research. It was followed by a section of memory forensic; acquisition techniques, memory analysis techniques and information about existing memory forensic frameworks and software.

---

3. Related Works

From the literature review, there are 9 major research works on browsing record recovery and forensic study on the browser. The first research work is published at 2003 and the latest research work is published at 2014. Among these research works, 6 of the works [1, 3, 7, 9, 10, 11] fall into *private browsing forensic* category. The rest 3 research works[4, 5, 8] is on data recover and analysis of *normal* (public) browsing mode. Only 3 of the studies [7, 9, 10] considers physical and virtual memory as data sources.

Before analyzing the methods and comparing the proposed model of the researches, this is the summary conclusion of the previous research works:

1. There is no standard framework in the research works. From the review of these works, there are no standard test data and evaluation model to compare the results. For a new contributor, it is difficult to make a judgement to improve the existing methods and models.
2. Architecture variations of browsers make a barrier for researchers to study the security architecture and database structure of the browser. The Newcastle university researchers [11] takes note that except the studies of [1, 10, 20], the rest of the study did not make the systematic study of browser architecture and its data structure. The researcher[11] is left to mention one more work [3] which studies Internet Explorer browser architecture and also implement a file carving tool for data remnants of private browsing records.
3. Some of the research works end up by making conclusion of possibility of extracting information from digital image, memory dump or, both without much focus on analysing the database structure on disk and the mapped file structure on memory [9, 10].

3.1 Methodology Comparison

In this section, it will be discussed the proposed method of existing methods by category. The related works can be distinguished according to the input data sources; persistent storage or volatile storage or fusion of both data sources. Like it mentioned before, [1,3,11] focuses mainly on the conventional forensic analysis model. [7, 9, 10] also considers physical memory as another input source. [4, 5, 8] are not research works related with private browsing research, however it is still essential to study the method and contributions of it. This section will discuss each research methodology by category.

The first research contribution of forensic analysis of private browsing is published at 2011[1]. This research contributes 4 objectives related with private browsing in 4 major browsers; the security threat model, experiment to survey the usage of private browsing mode, automated technique to discover private browsing implementation and propose the usage of extensions on private browsing. Their security threat model is enhanced with some considerations by researchers from Newcastle University[11]. The study approach is focused on security point of view rather than forensic study. The main objective is to measure out the security level and privacy level of private browsing. One major contribution of this research is that it can point out privacy violations and data accessibility between private and normal browsing mode. Howard Chivers[3] studies the *Inprivate browsing* of Internet Explorer 10 and implemented a model to carve the Inprivate browsing records using a tool from his previous research called *ESECarve*. This is one of the systematic contributions on private browsing research because it analyzes the data structure of Internet Explorer and discover identification to distinguish private browsing records with normal browsing records. However, The methodology can not be extended or applied on other browsers because of its significant architecture. Unlike Firefox, Chrome, and Safari, Internet Explorer uses Extensible Storage Engine (ESE) as its storage mechanism. The overview of the methodology can be viewed at figure 8. The most recent research publication [11] contributes by discovering new attacks that can violate privacy of private browsing. Moreover, it makes validation against previous known vulnerabilities to find out the browser vendors updated with security patches or not. The details of the result will be covered in next section.

At 2010, three researchers from CMU[7] published forensic analysis approach using memory analysis on private browsing mode. In this research, the researchers focus on detection of private browsing session on memory image and also experiment to see the security level. The raw memory takes during private browsing session alive and take another one after the browser session expires. After that, the process is reconstructed by using Memory Parser. This is the only research work that study on recovering volatile data from memory dump systematically. In addition to memory dump, the researchers from Zayed University[9] proposed a methodology which also analyze digital image. However, the methodology is more like *'searching keywords'* from memory image and disk image rather than the proper analysis of the process and data structure. Acquisition of physical memory is not a trivial process. The quality of memory dump depends on the technique and the acquired time. There is no clear information about the acquisition technique selection process and also no information about when the acquisition process performed. The researchers from Sam Houston University experimented on portable and private browser to validate the security level [10]. The methodology includes 2 experiments; preliminary experiment to monitor file integrity changes during the browsing sessions, the main experiment to recover data artifacts. The method of main experiment is similar to previous research[9]. There is no information provided about analysis process except it mentioned of the tools included in the process; FTK Toolkit and Nirsoft tools.

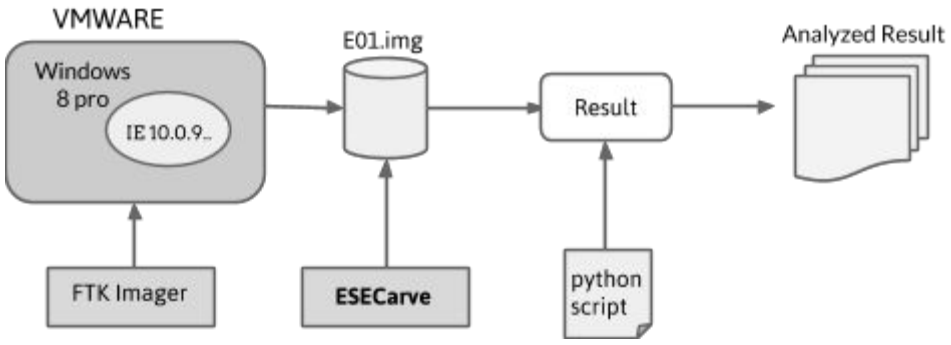


Figure 8: Process diagram of carving private browsing records from IE 10[3]

3.2 Prior findings and results

The following listing, figure 9, is findings and conclusions from previous works. Figure 10 is the list of possible attacks to browser [11]. It reflects the overall finding of the private browsing vulnerabilities. The list item with asterisk describes the new findings from their work [11].

1. There is implementation weakness in private browsing mode which leads to security vulnerabilities [1].
2. There is a possibility to carve volatile data from the acquisition of physical memory during and after the browsing session in a certain circumstance [7,10].
3. The chance of getting digital artifacts from private browsing session from volatile data is higher than from persistent storage [7,9,10].
4. From Internet Explorer's Inprivate browsing, it is possible to carve data from persistent storage [3].
5. Even though there is no data update on data artifacts on persistent storage, timestamps of the file changes in browsers except Firefox and IE.[11]
6. Google Chrome browser is the hardest browser to look for traces of digital artifacts [10].

Figure 9: Findings and Results from previous research works

	Firefox	Chrome	IE	Safari	Information leakage
Domain name system	√	√	√	√	Browsing history
Memory inspection	√	√	√	√	Browsing history, passwords, cookies
File timestamp	–	√	–	√	When private mode was last used
Index.dat*	N/A	N/A	√	N/A	When private mode was last used
SQLite database crash*	√	√	N/A	√	Minor to serious depending on browsers
SQLite added bookmark*	√	√	N/A	√	Minor to serious depending on browsers
Extension*	√	√	–	√	Browsing history
Cross-mode Interference*	N/A	√	N/A	N/A	User activities in private mode
Hyperlink attack	√	√	√	√	If the user is in private mode
Timing attack*	√	√	–	√	If the user is in private mode

Figure 10: List of attack types and its applicability to browsers [11]

3.3 Contribution

From these research works, it can be concluded that there is no complete systematic study on volatile memory analysis of private browsing except [7]. However, the experiment setup doesn't consider 64-bits memory architecture. Also, Windows XP support from Microsoft is ended at April 2014, which means it's becoming obsolete. The methodology can be improved with advanced forensic frameworks, software and acquisition methods, which can be evaluated to extract more forensic relevant data.

4.Proposed Work

From related works and theoretical study, it can be seen the possibility of recovering browsing records, history and user activities from private browsing mode. However, implementation variations of browsers make difficulty in implementing a tool for all application. Except the work of Howards Chivers[3], there is no practical tool implementation works from existing researches. It is necessary to perform a study with advanced tools and frameworks to validates types of data and chance of recovering private browsing records. From this proposed work, it will be a contribution for browser forensic research and it can continue to other browsers with similar methodology to study and analyze private browsing artifacts.

The chapter consists of five sections. The first section will discuss about research process and the plan to carry out the objectives. The second section is about input data and expected deliverables from the analysis of acquired images. In the third section, it will describes about the proposed method and the required steps to perform to achieve the objectives. It was followed by evaluation of the proposed method and limitation of the proposed method.

4.1 Research Process

As it mentions in introductory chapter, the main objective of this work is to propose a method to carve data artifacts from Incognito mode. While carrying out the main objective, theoretical study of browser forensic, memory forensic and browser architecture is carried out. Analyzing possibility of recovering data from Chrome's private browsing mode and a recovering method will be proposed from literature review, preliminary experiment and analysis of the preliminary result. As a preliminary experiment, persistent storage & volatile experiment is conducted in terms to analyze digital artifacts to identify existence of browsing data. The details of the experiment setup and the results will be

discussed at further sections. From the acquired images. a series of forensic analysis will implement to turn into digital evidence.

4.2 Input data and Expected deliverables

As input data for the experiment and proposed method, a list of websites with user activities and evidence keywords. Websites with different types of media (video, audio, images, social) are selected and browse within one incognito session for 5 minutes per websites. Acquisition of the disk image makes at three stages; while the browser session is alive, after the browser session expires and after the machine restarts. For volatile data experiment, acquisition of physical memory occurs every time a new tab is opened. The expected deliverables are the private browsing URLS and also the memory offset of the respective physical memory dump for further analysis. Recorded evidence keywords will be looked through the memory offsets and also uses a file carving method to extract audio, video,html and images from the memory dumps.

Figure 11 is a theoretical framework of the proposed methodology. After the digitals image is acquired from the device, proposed volatility framework-based analysis method will look for the existence of Google Chrome Incognito mode and extract process IDs, process images from it. Signature scanning is used to extract useful information such as URLs, email address, password & browsing keywords with corresponding PIDs and offsets. From Google's open source project called Chromium, it is possible to extract useful data structures such as variables, class names as signatures for Pattern/Signature scanning. The advantage of separating the acquisition process and analysis process is that memory dump from various acquisition methods can be applied if there is an existence of running process on it. File carving of memory mapped files will be executed by scanning physical memory dumps through lookups of file header and footer information. From the proposed method, it is expected to see both known and unknown data artifacts and also carved file by media types.

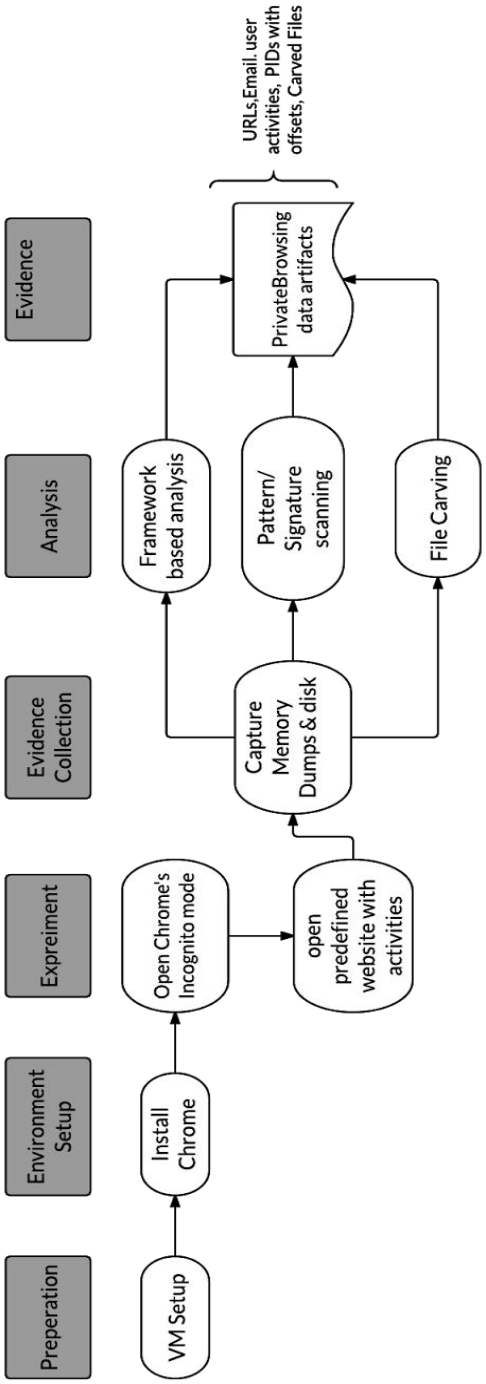


Figure 11: Theoretical framework of the proposed methodology

4.3 Proposed Method & Process

In this section, it will discuss about the environment and experiment setup of the research project and experiment, the 2 main process of the proposed method which are data acquisition and data artifact analysis and followed by evaluation of the proposed tool and finally, the limitation of the proposed method and tool.

4.3.1 Setup

Through out the research, Intel Core i7-3610QM @2.30 MHz, 4GB RAM Asus laptop with Debian Linux system is used and QEMU KVM virtual machine emulator is used and Windows 8 Service Pack 0 x64 bit architecture is installed as virtual machine. Details of Virtual environment can be see at table 6.

Properties	**Value**
Host Operating System	Linux Debian OS
VM Operating System	Windows 8 SP0
Chrome Version	34.0.1847.131 m
CPU Allocation	8 logical CPUs
RAM Share	1 GB
Allocated Image size	12 GB (fixed allocation)
Hypervisor	KVM
Emulator	QEMU

Table 6: Virtualization Setup

While evaluating the virtualization platform, QEMU is the suitable technology for memory forensic and live forensic experiment as:

1. It allows the manager to export the whole memory without interrupting virtual machines and provide an interface to analyze physical memory region,
2. It supports emulation of various operating system platforms and system architectures,

3. The experiment can perform not only as paravirtualization mode but also as full hardware virtualization which can simulate physical hardware environment.

Chrome version 34.0.1847.131 m is used during the preliminary experiment and for evaluation of the proposed tool, the latest Chrome version will be used to carry out the experiment. *Autopsy* forensic browser is used to analyze data artifacts from permanent storage. It's an open source software which is used in various forensic research and available for many Operating Systems and architectures. Some useful features of Autopsy forensic browser are:

1. It keeps activity log of a forensic analyst.
2. MD5 and SHA1 Hashing for disk image integrity
3. Case management
4. Keyword searching
5. Timeline analysis.

As for memory analysis, there are 3 major frameworks to consider, that are Volatility, Rekall, and FATKit. Among these 3 frameworks, Volatility provides features and advantages as:

1. It supports various memory dump formats such as EWF, Windows crash dump file, hiberfil.sys, .vmss, qemu memory dump and so on.
2. It supports memory analysis of various operating systems and architectures including Android mobile Operating System.
3. Forensic researchers and practitioners can implement third-party plugins with modules provided by Volatility.
4. Volatility is de facto standard memory forensic framework in terms of its modular architecture, ranges of Operating System supports and update releases.

Table 7 describes detail list of open source tools and software used in preliminary experiment and proposed method.

Tool	**Description**
FTK Imager	Open Source image acquisition tool
QEMU	Generic VM emulator based on KVM hypervisor
ProcessMonitor	Open Source File Integrity Monitoring applications

HexDump (hd)	CLI utility tool to view binary files
Autopsy Forensic browser	Open source GUI Forensic Image browser based on *Sleuthkit*
Volatility Framework	Open Source volatile memory analysis framework with availability to implement third- party plug-ins
Scalpel	File carving tool for memory mapped files based on file headers and footer information
Yara	Open source pattern matching tool for textual and binary data

Table 7: List of open source tools used in research

4.3.2 Data Acquisition

For Disk image acquisition, it is a straightforward process which bit-by-bit copy of the entire disk partitions and *FTK Imager* is used and it can convert disk images into forensic file format such as E01.dd and so on. From evaluation of acquisition method[15], dumping memory image directly from *QEMU* is selected for the proposed method and experiment. From the literature review, virtualization acquisition method has high atomicity and average availability of data. Unlike disk image acquisition, physical memory image acquisition is not a trivial task and it cannot be redone at real case scenario. Even though cool booting and hibernation file methods seems to give high availability compared with virtualization method, the system has to shut down improperly and it is not suitable for live forensic analysis and there are chances of losing data from running processes. According to Microsoft documentation of hibernation file, contents in hiberfil.sys is compressed and analysis of the contents is hard but possible. Volatility memory framework can handle Windows hibernation file format and crash dump file. These acquisition methods can be applied for some scenarios; for e.g, the case of forensic analysts can leverage themselves to windows administrator privilege and turn on hibernation feature. Due to the limitation of resources. hardware based acquisition methods are not able to exercise for this project. The selected acquisition method allows live acquisition without required to "pull-the-plug" the system and take memory snapshots from virtual machine though monitoring console. The commands are described at Figure 5 of literature review chapter.

4.3.3 Analysis

After getting disk and memory image, analysis of digital artifacts starts. For disk image, validation of the disk image can be done easily with cryptographic checksum after the disk image acquired. The MD5 & SHA1 hashing is used to check the integrity of the disk image throughout the analysis process. For physical memory images, there are no proven validation methods for acquired physical image. There are several conditions that need to check to make sure the success of acquisition, which are:

1. The size of the memory image, which should be the same as physical RAM of the acquired system.
2. first 256 MB of image filled with zeros,
3. Image with expected size but first 256 MB of image repeated.
4. page 0 is missing from the image.

Before working on further analysis process, these conditions are validated.

Unlike acquisition process, analysis process can be repeatable and use various methods to analyze digital artifacts from it. For disk image analysis, *Autopsy* forensic viewer is used to view data artifacts, search by keywords and also can extract data and files from it. Even though the chance of finding private browsing data artifacts from persistence storage is low, it is necessary to perform an analysis of data evidence to validate the existing findings. The analysis process steps with persistent storage are

1. Identifying location of data artifacts of Chrome.
2. Examining data artifacts of Chrome on location identified from step 1.
3. Extracting Chrome SQLite files of chrome such as *Histroy.Sqlite,Cookie.Sqlite* and investigate the contents using SQLite browser and hexadecimal viewer.
4. Study data structures of Google Chrome SQLite databases & other important files.
5. Keyword searches of URLs, user input keywords to identify the Chrome data artifacts.
6. Investigate pagefile.sys to analyze browsing artifacts on virtual memory

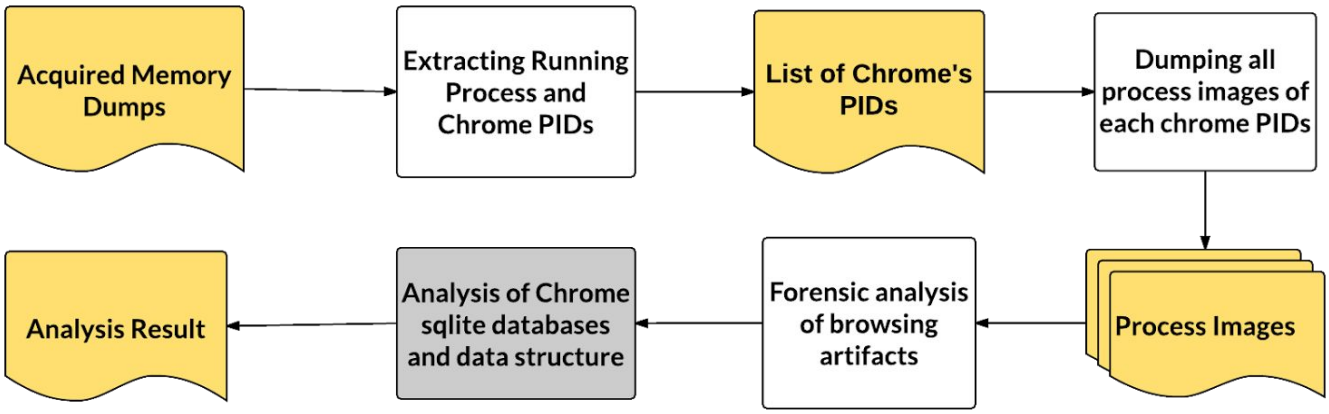
The process of the analysis of application-level data, "next generation analysis" of physical memory is focused on application address space. For application-level analysis, the main objective is to find location of data structure of desired application and it requires knowledge about data structures of the application. For open source applications, it can be achieved by code reviewing of the application and for closed source application, this methodology can not be applied. For Google Chrome, Google offers open source version of Chrome browser called Chromium and it have most of the same features as Google Chrome with slightly differences such as automatic update patch installation, default media codec installation and default flash player. The analysis process contains 4 major process and figure 12 describes the overview of analysis process of these 4 main process and detail analysis steps performed with input and output data.

The following listing is the possible abstract steps to investigate application from physical memory images:

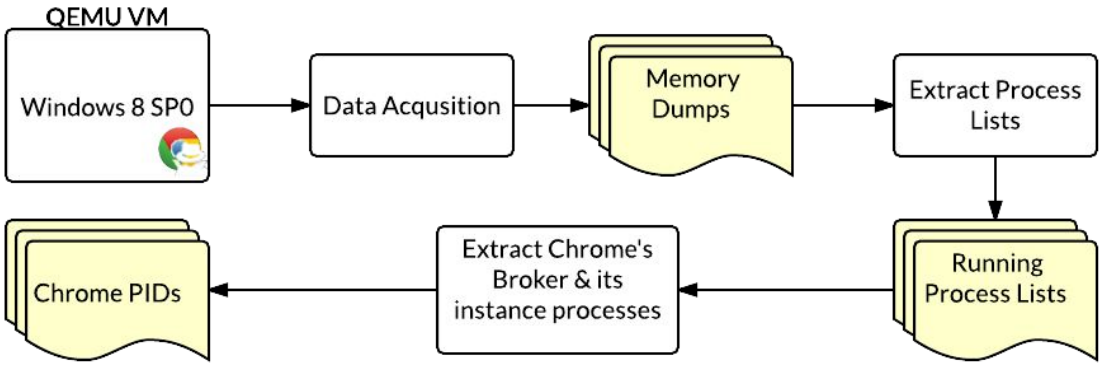
1. Examine the acquired memory image to verify that volatility detects memory image correctly
2. From the list of running processes from memory images, get Process ID (PID) of Chrome.exe, which is expected to see several child processes from a single parent chrome.exe process.
3. Dump all possible memory pages from the process images by Chrome PIDs and association tables between virtual and physical memory offsets.
4. From saved keywords, these strings are scan through the physical memory image to verify existence of expected data artifacts
5. Find traces of Chrome's Incognito mode from physical memory by
    a. Signature based scanning of evidence keywords, URLs & SQLite structure from memory content. Unlike data on persistent storage, mapped files and data on memory are fragmented because of memory swapping. For this reason, analysing Windows page file can get information that are not at memory. Signature-based scanning method is the most appropriate method to handle highly fragmented data on memory images.
    b. Reviewing source code of Chromium, especially at the class names through the execution path of Private browsing mode. The purpose is to identify data structures

from memory image which can be verified that the data extracted are not false positives and we can create useful signatures to scan the memory content. From this step, it can be proved that the schema used in in-memory SQLite databases of Chrome is identical as the databases used in normal browsing mode.

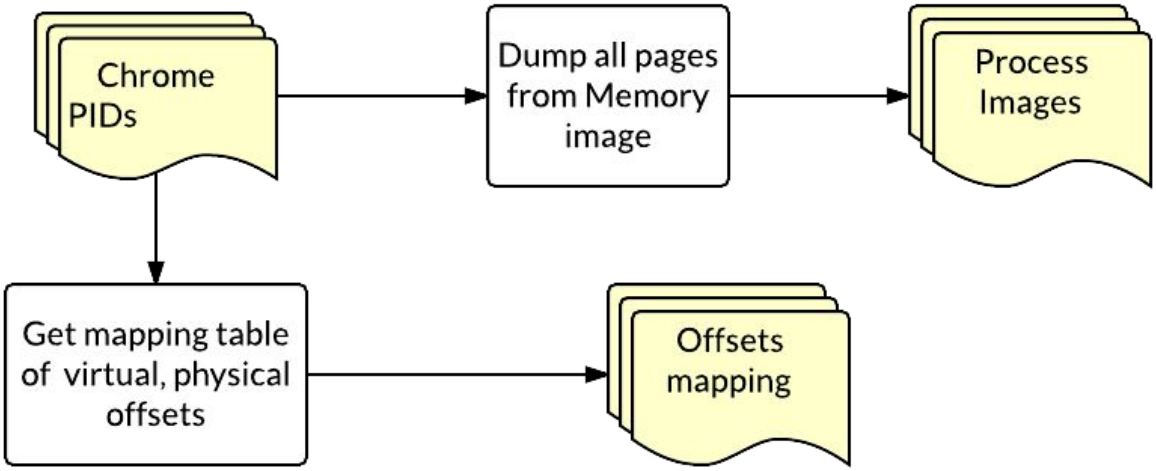
6. Use *Scalpel* file carver with header and footer configuration of various file formats and analyze recovered mapped files in memory.



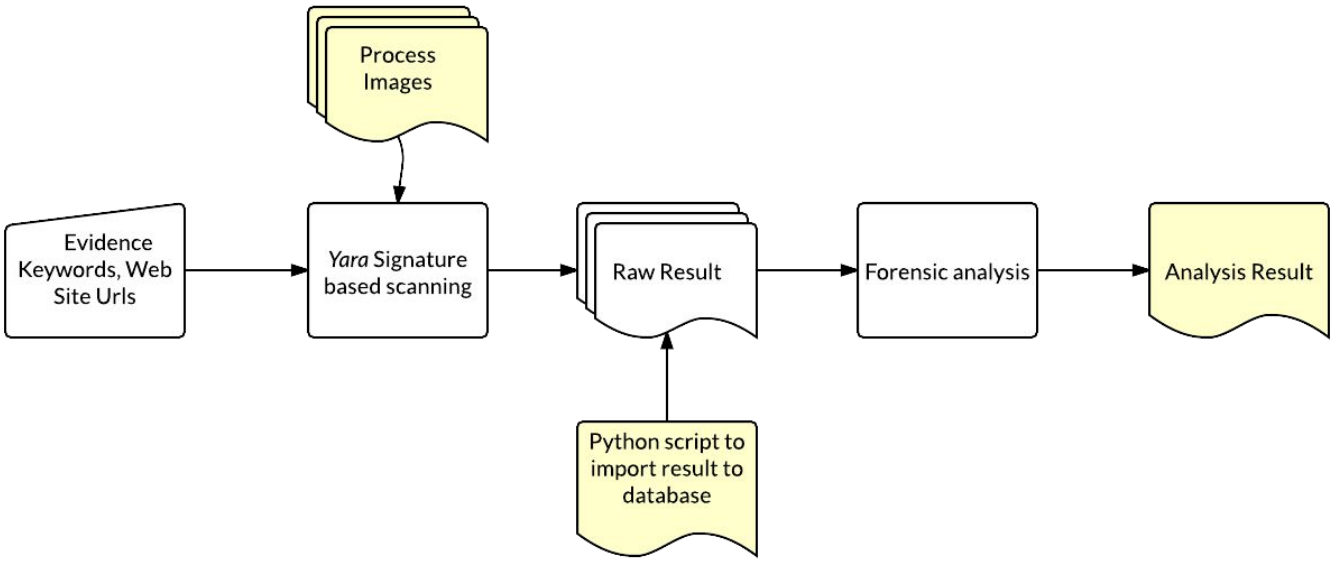
1. **Extracting Running Process and Chrome PIDs**



2. **Dumping all process images of each chrome PIDs**



3. Forensic analysis of browsing artifacts



4.Analysis of Chrome sqlite databases and data structure

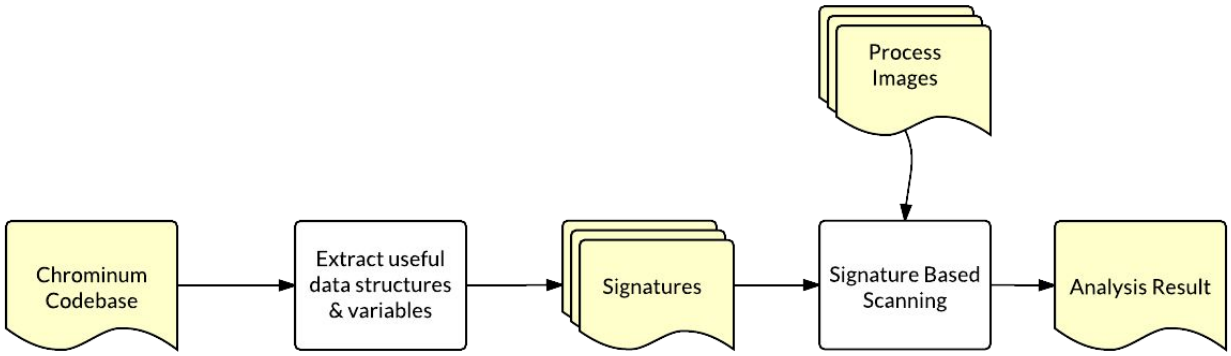


Figure 12: Analysis process series of physical memory images to extract Google Chrome's Incognito data

4.4 Evaluation

Even though there are public forensic corpora, there is no test data and disk images for application specific data. For this reason and also the variations of research approach and environment setup, it is not possible to evaluate the findings and result with other research works.

The proposed method will be evaluated with the possibility of recovering data from browsing activities, evidence keywords and number of carved files through three data sources,which are persistent storage, default storage location of Google Chrome and physical memory. The proportion of finding traces of URLs, keywords with predefined activities will be presented and also success of file recovery will be measured with types of file found by media type. All memory dumps and disk images will be validated with cryptographic checksum to verify the integrity of the acquired disk images.

4.5 Limitation

The listing is the limitation of the proposed method:

1. From the preliminary experiment, it can be seen that the possibility of recovering data after the private browsing session expires is low.
2. Memory images from the other software based and hardware-based acquisition methods are not tested.
3. The experiment are carried out with Windows Operating system and it is lack to implement similar experiment for Linux & Mac Operating Systems.

4. The methodology doesn't analyze traces from web traffic.

---

5. Preliminary Work

The experiment is to investigate traces of private browsing artifacts in volatile memory and to analyze digital remnants left in permanent storage. For this experiment, several open source tools are used to gather the information. From this experiment, it is able to validate the possibility of recovering data evidence from Google Chrome's Incognito mode.

5.1 Experiment Setup

In this section, it will describe the experiment setup and tools used in the experiment For the experiment, Windows 8 SPo is used and installed at QEMU virtualization environment. Google Chrome is installed to the clean system and initiate the experiment afterwards.

For the persistent storage part of experiment, *ProcessMonitor* is used to monitor file integrity while the browser installed to the system and to detect file activity while the experiment is going on. *FTKImager* is used to acquire the image before and after the browser session expires and uses *autopsy* as a forensic browser to analyze the image manually.Images are acquired with *Eo1* format. Five websites with different type of media (video, audio, images, social and e-commerce) is selected to browse and the activity and keyword inputs are noted to evaluate the analysis result. Table 8 describes the detail records of activity and keyword used throughout the experiment.

No.	Websites	Type	Activities	Evidence Keywords
1	youtube.com	Video streaming	● Search 'arctic monkeys' ● Search 'sports animal' ● Search 'bugatti veyron' ● *play some results*	● arctic monkeys ● sports animal ● bugatti veyron
2	9gag.com	Image repository	● search 'celebrity' ● search 'weird' ● browse trending pics	● celebrity ● weird

3	soundcloud.com	Audio streaming	● sign into my account ● play some tracks on stream ● Search 'bleachers' and play a track	● account name ● with Facebook ● bleachers
4	twitter.com	Social tweets stream	● login and browse streamline ● make a tweet	● account ● tweeted inside a nested machine
5	amazon.com	E-commerce	● search 'fire phone' ● search 'no destination' ● search 'Amazon book' ● click some results	● fire phone ● no destination ● Amazon book

Table 8: Lists of websites and keywords used in preliminary experiment

The following listing is the general procedure performed during the persistent storage experiment:

1. A private browsing window is open and each website is populated with a new tab.
2. For each website, approximately 5 minutes spend to browse, search and uses the functions of the website.
3. The disk image is acquired with *FTKImager* after the 5 browser tabs is opened. Another image is taken after the browsing session expires and reboot the machine and acquired the last one.
4. With *Autopsy* file browser, Data artifacts of Chrome browser are analyzed at default history file location and also perform forensic analysis of disk image.
    - History and cache SQLite databases are investigated to see whether it's populated with data or not.
    - Study Chrome data artifacts and its data structure on persistent storage.
    - Perform forensic analysis by strings (keywords) searching through allocated and unallocated storage space.
5. Test out existing tools (*ChromeAnalysis, Internet History Browser, ChromeForensic*) to see their capability to extract Chrome artifacts from disk image

As for volatile storage part of experiment, memory dumps are acquired using QEMU Monitor, which can dump physical and virtual memory. 5 memory dumps are captured after each website is browsed and additional one after the browser session expires. Analysis of the acquired memory image is performed using the modules and plugins from Volatility framework. Table 9 describes the list of plugins used in volatile memory experiment to investigate existence of browsing data and files mapped in memory.

Volatility Plugins	**Description**
pslist	A plugin to enumerate list of running process from memory image by looking up linked list of processes (Logical Process Enumeration)
psscan	A plugin that uses another method to enumerate running process by scanning _EPROCESS Structure (Physical Process Enumeration)
procdump	A plugin to dump process executable by PID or physical offset
memdump	A plugin that can extract all possible pages of a process to a file
memmap	A plugin which get association of memory offsets between virtual and physical memory and associated offsets of memdump output
vadscan	A plugin that presents details of VAD allocations of a process.
vaddump	A plugin to dump VAD allocations of a file
Yarascan	Yara pattern matching plugin for Volatility framework to identify existence of patterns of keywords in textual or binary formats.

Table 9: Lists of Volatility Plugins used at preliminary experiment

The following listing is the procedure of the experiment:

1. A private browsing window is open and each website is populated with a new tab.
2. Physical memory is acquired after a site is loaded and spend around 5 minutes browsing the website with defined activities and keywords.
3. Examine the acquired memory image to verify that volatility detects memory image correctly and also to verify the fidelity and availability of the acquired physical memory images.

4. From each acquired physical memory dump, get list of running process(PID) of chrome browser.
5. Dump running Chrome Process Images by PIDs and memory offsets.
6. Apply *Yara* pattern searching tool to scan existence of data evidence in process images from the experiment's keywords and browsing URLs.
7. Implement python scripts to import the result from step 5 into the database for further analysis.
8. Count for occurrence of URLs, keywords and other meaningful information such as email address, username and passwords and so on.
9. Use *Scalpel* file carver with header and footer configuration of various file formats and analyze recovered mapped files in memory.
10. Test out with existing tools such as *Chromehistory* volatility plugin to extract SQLite artifacts.

The summary of the analysis result is that private browsing records and user activity are not traceable at permanent storage. According to the 2014 research study about security vulnerability of private browsing[11], in-memory SQLite is used to store data of chrome Incognito mode and no browsing data is written to SQLite databases the disk storage. From Chromium code review of sqlite database connection, the code comment and in-memory sqlite function also proves that Google Chrome uses in-memory SQLite for Incognito mode. Step by step experiment processes and its detail analysis result will be discussed in next section. From volatile data analysis, the experiment result shows that there are traces of browsing records and activities left in physical memory during and after the browsing session expires.

5.2 Experiment Result

This section will describe the findings and result from the preliminary experiment. The first section describes the analysis result from permanent storage part of experiment, followed by the result of volatile memory part of the experiment.

5.2.1 Permanent storage Experiment Result

From the log file of *ProcessMonitor,* which monitor changes of files to the disk storage, the default location of Google Chrome data is at

```
C:\Users\%username%\AppData\Local\Google\Chrome\User Data\
```

Google Chrome stores user data and creates temp files at that location as default. Figure 13 is the log file output of *ProcessMonitor* and it points out the location from file operations during Chrome installation.

78876	6:50:03.0024538 AM	chrome.exe	2132	CreateFile	C:\Windows\SysWOW64\winmm.dll
78877	6:50:03.0414284 AM	chrome.exe	532	QueryOpen	C:\Users\testpc\AppData\Local\Google\Chrome\User Data
78878	6:50:03.0415399 AM	chrome.exe	532	CreateFile	C:\Users\testpc\AppData\Local\Google\Chrome\User Data
78879	6:50:03.0416072 AM	chrome.exe	532	QueryBasicInformationFile	C:\Users\testpc\AppData\Local\Google\Chrome\User Data
78880	6:50:03.0416413 AM	chrome.exe	532	CloseFile	C:\Users\testpc\AppData\Local\Google\Chrome\User Data
78881	6:50:03.0416811 AM	chrome.exe	532	IRP_MJ_CLOSE	C:\Users\testpc\AppData\Local\Google\Chrome\User Data
78882	6:50:03.0418092 AM	chrome.exe	532	CreateFile	C:\Users\testpc\AppData\Local\Google\Chrome\User Data\C905.tmp
78883	6:50:03.0835112 AM	chrome.exe	3816	ReadFile	C:\Windows\SysWOW64\wininet.dll
78884	6:50:03.0840925 AM	chrome.exe	2548	ReadFile	C:\Users\testpc\AppData\Local\Google\Chrome\Application\34.0.1847.131\chrome_child.dll
78885	6:50:03.0852114 AM	chrome.exe	2912	ReadFile	C:\Users\testpc\AppData\Local\Google\Chrome\Application\34.0.1847.131\chrome_child.dll
78886	6:50:03.0855540 AM	chrome.exe	532	QueryNameInformationFile	C:\Users\testpc\AppData\Local\Temp\scoped_dir_532_9258\CRX_INSTALL_locales\cs\messages.json
78887	6:50:03.0856044 AM	chrome.exe	532	QueryNameInformationFile	C:\Users\testpc\AppData\Local\Temp\scoped_dir_532_9258\CRX_INSTALL_locales\cs\messages.json
78888	6:50:03.0860673 AM	chrome.exe	2300	QueryInformationVolume	C:\Users\testpc\AppData\Local\Temp\scoped_dir_532_9258\CRX_INSTALL_locales\cs\messages.json
78889	6:50:03.0861056 AM	chrome.exe	532	CloseFile	C:\Users\testpc\AppData\Local\Google\Chrome\User Data\C905.tmp
78890	6:50:03.0861250 AM	chrome.exe	2300	QueryAllInformationFile	C:\Users\testpc\AppData\Local\Temp\scoped_dir_532_9258\CRX_INSTALL_locales\cs\messages.json
78891	6:50:03.0861809 AM	chrome.exe	2300	ReadFile	C:\Users\testpc\AppData\Local\Temp\scoped_dir_532_9258\gmail.crx
78892	6:50:03.0862405 AM	chrome.exe	2300	WriteFile	C:\Users\testpc\AppData\Local\Temp\scoped_dir_532_9258\CRX_INSTALL_locales\cs\messages.json
78893	6:50:03.0863358 AM	chrome.exe	2300	CloseFile	C:\Users\testpc\AppData\Local\Temp\scoped_dir_532_9258\CRX_INSTALL_locales\cs\messages.json

Figure 13: ProcessMonitor log file of Chrome installation Process

Most of the temporary files are deleted and no data related with the private browsing session are written to these chrome SQLite databases and some important files from table 10. These are the databases Chrome use to store data such as visited sites, cookie location, download history. However, the traces of browsing records can be seen at *pagefile.sys* , which means data are not written to the disk storage but only in virtual memory. While Google Chrome is in private browsing mode, it initiates in-memory SQLite database with ":memory" and deleted after the browser session closes. Figure 14 describes the result of code review from Chromium Open source project, the comment and SQLite3 connection method proves the usage of in-memory SQLite database for private browsing mode.

Files	Location	SQLite Tables/ File format	Description
History	~Chrome/User Data/Default/	downloads, downloads_url_chains, keyword_search_terms , meta,	A SQLite database which keeps information about visited URLS and downloads history

		segment_usage, segments, urls,visit_source, visits, views	
Cookie	~Chrome/User Data/Default/ Application Cache/databases	cookies,meta	A SQLite database that keeps cookies information
Archived History	~Chrome/User Data/Default/	keyword_seach_terms, meta, urls, visited_source, visits	Google Chrome keeps history for only 3 months and archive them at this SQLite table.
Index	~Chrome/User Data/Default/ Application Cache/Databases	caches, deletedresponseids, entries, groups, meta, namespaces, onlinewhitelists, views	A SQLite database where websites keeps information and URLs of caches and thumbnails
cache/	~Chrome/User Data/Default/	index: hash table of stored URLs f_xxxxx: small file objects data_x: large file objects	A folder that keep disk caches which Chrome can fetch later for quick access
Current Session Current Tabs Last Session Last Tabs	~Chrome/User Data/Default/	SNSS files in binary format and this file format is still in need of study	These files store information about tabs and session.
Bookmarks Preferences	~Chrome/User Data/Default/ Bookmarks	JSON format	These JSON files store information about bookmarks & user preferences

Table 10: Lists of Google Chrome's SQLite Databases and other important files

```
** ^If the filename is ":memory:", then a private, temporary in-memory database
** is created for the connection.  ^This in-memory database will vanish when
** the database connection is closed.  Future versions of SQLite might
** make use of additional special filenames that begin with the ":" character.
** It is recommended that when a database filename actually does begin with
** a ":" character you should prefix the filename with a pathname such as
** "./" to avoid ambiguity.
**
** ^If the filename is an empty string, then a private, temporary
** on-disk database will be created.  ^This private database will be
** automatically deleted as soon as the database connection is closed.
```

```
bool Connection::OpenInMemory() {
  in_memory_ = true;
  return OpenInternal(":memory:", NO_RETRY);
}

bool Connection::OpenTemporary() {
  return OpenInternal("", NO_RETRY);
}
```

Figure 14: Usage of In-memory SQLite database from chromium codebase(/usr/include/SQLite3.h)

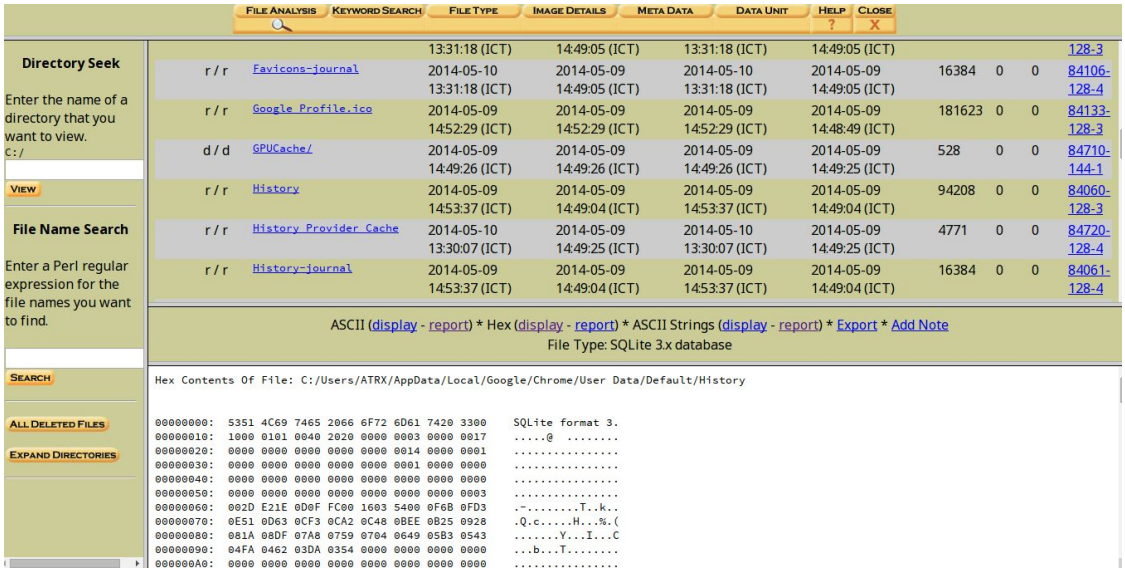


Figure 15: Autopsy forensic browser with Chrome Default Data Location

To see the capability of existing chrome browsing forensic tools, *ChromeAnalysis, Internet History Browser* & *ChromeForensic* are used to measure whether these tools can extract private browsing information. Generally, it looks for various Chrome SQLite files of Google Chrome and provides functions to analyze and export the result. Since there is no data artifacts of private browsing records are not stored at these SQLite databases, it is not expected to retrieve data with these tools. To sum up

the result of evaluation of existing tools, it shows that existing tools are not capable of retrieving private browsing artifacts from persistence storage.

5.2.2 Volatile Data Experiment Result

Since Chrome's Incognito mode uses SQLite in-memory storage, it is expected to discover private browsing data at physical memory. The analysis of acquired memory images is based on Volatility framework, its plugins, modules and also custom python scripts to analyze the raw result of signature scanning. As a summary of the experiment, the result of volatile memory experiment proves that existence of useful browsing information and can carve partial and full media files from memory mapped files. In this section, it will describe the result of each step of the forensic analysis process of physical memory dump files.

A total of 5 memory images is acquired after browsing a defined set of website URLs with activities at Chrome's Incognito mode. Additional one more image is acquired after the browser session is expired. Before analyzing the captured memory images, these images are validated with common characteristics of failed and incomplete memory images such as expected size, missing pages and repeated content throughout memory images, which is discussed at section 4.3.3 are checked. These conditions are validated for all acquired image by using hexdump utility tool.

To get list of running process and PID of chrome, 2 plugins from Volatility plugins are used. Chrome uses sandbox mode to restrict resource privileges of the process to prevent from malware and exploits. According to Chrome sandboxing documentation,Google Chrome use sandboxing security model and each tab runs as separate process. It is expected to see 2types of Chrome process level, which are broker process and target process. Broker process controls and creates targeted sandbox process. Therefore, for each memory image, it is expected to see the number of Chrome process as

```
Number of open tabs + 1 Chrome broker process
```

There are several methods to get list of running process and a Volatility plugin called psxplugin can list the result of running process of 7 different methods (pslist, psscan, thrdproc, pspcid, csrss, session

deskthrd). For logical and physical enumeration of process list from images , pslist and psscan plugin can be used. As it mentions in literature review, there is a chance that physical enumeration of process list can extract not only running process but also terminated process. Figure 16 is the example output of process list from 9gag.com memory dump. With this three plugins, a list of chrome PID are extracted from 5 memory dumps and it can be seen at Table 11. Parent Broker PID of Chrome is 3060 and becomes PPID of all spawning tabs. From memory dump after browser session expires, it can be seen that broker chrome Process,3060 is still running as background process but its child process are terminated.

```
aung@aung-Inspiron-3847:~/Icognito Project$ vol.py -f Incognito-Project-Disk-Images/9gag.dum
Volatility Foundation Volatility Framework 2.4
Offset(P)          Name                PID   PPID PDB                Time created
------------------ ---------------- ------ ------ ------------------ ------------------------
0x000000003cc66940 wmpnetwk.exe       2832    484 0x00000000151f1000 2014-11-11 21:02:11 UTC
0x000000003cc79940 chrome.exe         2700   3060 0x0000000383cc000 2014-11-11 21:15:26 UTC
0x000000003cd77080 dllhost.exe        2336    564 0x0000000033977000 2014-11-11 21:00:31 UTC
0x000000003ced0940 taskhostex.exe     2120    484 0x0000000025da9000 2014-11-11 21:00:15 UTC
0x000000003ced2940 explorer.exe       2340   2160 0x0000000029b09000 2014-11-11 21:00:17 UTC
0x000000003ced6940 taskhost.exe       2164    484 0x0000000008543000 2014-11-11 21:02:59 UTC
0x000000003ceed940 chrome.exe         2652   3060 0x000000000b5e3000 2014-11-11 21:14:53 UTC
0x000000003cfd3600 SearchIndexer.     2816    484 0x0000000313ce000 2014-11-11 21:00:22 UTC
0x000000003d0bb940 svchost.exe        1580    484 0x000000001c719000 2014-11-11 21:00:09 UTC
```

Figure 16: List of Running Processes from psscan plugins

Physical Offset	Virtual Offset	Name	PID	PPID	Threads	Handles	Session	Start Time
0x000000003d7fb940	0xfffffa80027fb940	Chrome.exe **(Broker)**	3060	1908	35	1	1	*2014-11-11 21:10:35 UTC+0000*
0x000000003e5fc940	*0xfffffa80027fb940*	*chrome.exe (**Youtube**)*	600	3060	12	0	1	*2014-11-11 21:10:35 UTC+0000*
0x000000003ceed940	*0xfffffa8002eed940*	Chrome.exe **(9gag)**	*2652*	3060	8	0	1	*2014-11-11 21:14:53 UTC+0000*
0x000000003cc79940	*0xfffffa8003079940*	*chrome.exe (**9gag**)*	*2700*	3060	17	0	1	*014-11-11 21:15:26 UTC+0000*
0x000000003de71080	*0xfffffa8001e71080*	*Chrome.exe (**soundcloud**)*	*2876*	3060	11	0	1	*2014-11-11 21:17:21 UTC+0000*

0x00000 0003d338 940	*0xfffffa8002b38 940*	*Chrome.exe (**soundclou d**)*	*1176*	3060	7	0	1	*1 2014-11-11 21:18:44 UTC+0000*
0x00000 0003d53f 800	*0xfffffa800293f 800*	*chrome.exe (**twitter**)*	*2304*	3060	8	0	1	*2014-11-11 21:20:20 UTC+000*
0x00000 0003cd21 840	0xfffffa8003121 840	*chrome.exe **(amazon)***	*1620*	3060	8	0	1	*2014-11-11 21:22:48 UTC+0000*

Table 11. List of extracted Chrome process from memory images

After the memory offset of each Chrome processes are identified, it is possible to extract all possible pages of process memory and also mapping table of memory content among virtual, physical and offsets of the dumpfile. One point to consider at this point is that process memory dump will be fragmented because of windows paging and it is expected to encounter fragmented information from physical memory. Instead of traditional string matching, YARA is integrated with volatility to perform effective pattern matching for textual and binary patterns. A series of yara rules are generated to scan through each memory dumps to verify existence of data artifacts. The output of yara provides information about process owner of the data artifacts, memory offsets of the artifacts and ASCII strings of the artifacts. Python script is implemented to categorize data and import it into database for further analysis.

No.	**Rule**	**Description**
1	/(www|http).+\.(com|net|org)/ wide ascii	search for URL address in both ASCII character order and also for strings encoded with two bytes per character
2	/\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b/ wide ascii	search for patterns of email address in both ASCII character order and also for strings encoded with two bytes per character
3	/([0-9]{1,3}\.){3}[0-9]{1,3}\x00/ wide ascii	search for patterns of null terminated IP address both ASCII character order and also for strings encoded with two bytes per character
4	"password" nocase	Search for occurrence of password string with no specific case
5	"arctic monkeys" nocase	search for occurrence of keywords of youtube.com activities

	"sports animal" nocase "bugatti veyron" nocase	in no specific case
6	"celebrity" nocase "weird" nocase	search for occurrence of keywords of 9gag.com activities in no specific case
7	"aung" nocase "@.aol.com" nocase "bleachers" nocase	search for occurrence of keywords of soundcloud.com activities including username in no specific case
8	"atrhein" nocase "@.gmail.com" nocase "tweeted inside a nested machine" nocase	search for occurrence of keywords of twitter.com activities including username in no specific case
9	"fire phone" nocase "no destination" nocase "Amazon book" nocase	search for occurrence of keywords of amazon.com activitiesin no specific case

Table 12. List of Yara rules generated to scan browsing artifacts

The result shows that keywords,website urls, email address, login username and password can be found for all websites from each respective process images. In the case of Soundcloud and Twitter, it is able to found user information such as username, password and email address. Moreover, user activities such as searching, status updating (tweeting) can be found at these process images. Figure 17 shows raw output of Yara, which points out the username and password used to log in to the service.

However, Interested information found in process image can't provide associated memory region of mapped files and data resource. Looking up virtual address descriptor (VAD) tree can provide information of associated objects and other useful information about these interested memory regions. Table 13 describes the occurrence of browsing artifacts from acquired memory. VAD is a kernel data structure that can presents starting and ending memory offsets of memory mapped files in a particular process. Table 13 describes the occurrence of browsing artifacts from acquired memory.

Websites	**PID**	**Evidence Keywords**	**Occurrences of Data artifacts**
youtube.com	600	● youtube.com ● arctic monkeys ● sports animal ● bugatti veyron	1977 2 9 4
9gag.com	*2652* *2700*	● 9gag.com ● celebrity ● weird	73 13 20

soundcloud.com	*2876* *1176*	● soundcloud.com ● account name (*Aung Thu Rha Hein*) with Facebook Authentication ● bleachers	5523 20 204
twitter.com	*2304*	● twitter.com ● account(*Atrhein*) ● password ● a tweet "tweeted inside a nested machine"	586 928 2 5
amazon.com	*1620*	● amazon.com ● fire phone ● no destination ● Amazon book	6157 81 16 21

Table 13. Browsing URLs and Traces of Keywords found on Memory Dumps

Figure 17. Extracted URLs, keywords, twitter username,password & tweets from physical memory dump

To see that whether browsing data from expired Incognito session are still in background process, signature-scanning analysis is used to scan keywords and URLs of the physical memory dump after session expires. From Chrome broker process, it still shows keywords from previous sessions and also URLs of auto imported bookmarks from previous default browser, which is Internet Explorer. Twitter username is still showing in background Chrome process but login password is no longer appears in physical memory. The process resident pages of browsing session is likely to persist for a particular time limit. that depends on factors such as system activity, data size, memory type and operating system. It can be said that the experiment system environment is not running lots of third-party process and

chrome process still persists application data even though after the processes terminates.

As an another stage of analysis, Scalpel file carver is used to carve memory mapped files for the memory dump. Basically, Scalpel file carver looks through physical memory by providing header and footer information of files and data structure. For this experiment, a set of file format for each media types are carved. Table 14 is the Scalpel configuration list of file types with its headers and footer information. These information are collected from online file signature databases and also manual analysis of file formats. Scalpel file carving algorithm will look for file header and carves a file after it sees a matched footer information or the file size limit is exceeded. Scalpel configuration provides common header and footer of common file types and this information is verified by looking at hexdump file viewer of sample files. A limitation of scalpel algorithm is that for file types with no defined footer such as bmp and mov, it can give lots of false positives result. REVERSE keyword scan backward from file size limit of the footer to the header which can carve files with multiple occurrences of footer.

File Format (Case Sensitivity)	**Size (byte)**	**Header**	**Footer**
gif(y)	5000000	\x47\x49\x46\x38\x37\x61 \x47\x49\x46\x38\x39\x61	\x00\x3b
jpg(y)	200000000	\xff\xd8\xff\xe0\x00\x10	\xff\xd9
png(y)	20000000	\x50\x4e\x47?	\xff\xfc\xfd\xfe
bmp(y)	100000	BM??\x00\x00\x00	-
mov(y)	10000000	????mdat ????widev ????skip ????free ????idsc ????pckg	-
mpg(y)	50000000	\x00\x00\x01\xba \x00\x00\x01\xb3	\x00\x00\x01\xb9 \x00\x00\x01\xb7
htm(n)	50000	<html	</html>
pdf(y)	5000000	%PDF	%EOF\x0d REVERSE

			%EOF\x0a REVERSE
wav(y)	200000	RIFF????WAVE	-
ra(y)	1000000	\x2e\x72\x61\xfd .RMF	-
mp3(y)	8000000	\xFF\xFB??\x44\x00\x00 \x57\x41\x56\45 \xFF\xFB\xD0\ \x49\x44\x33\ \x4C\x41\x4D\x45\	\x00\x00\xFF\ \xD1\x35\x51\xCC\

Table 14. File header & footer information for carving from physical memory

Table 15 is the summary result of data types found in physical memory dumps. As for audio files, truncated files can be carved but it is not playable with media player. Images files from website streams and thumbnails can be carved and high proportion of images are complete with less proportion of partial images. As for video media, it is not expected to find from browsed web site except youtube.com. However, youtube streaming protocol is *"Dynamic Adaptive Over HTTP"* and it is infeasible to extract MPEG-Dash fragments from memory with header/footer scanning algorithm. Figure 18 is the screenshot of Scalpel file carver output of youtube.com jpg thumbnail files.

Number of memory mapped files are significantly lower after browser session is expired, which means that possibility of carving memory mapped files is low after the processes terminate. The output of Scalpel includes an audit file and folder with subdirectories of carved files by data type.

Data Type	**Findings**
Audio	● truncated audios files can be carved. ● single mp3 file from soundcloud.com is carved but it's not playable.
Video	● many false positives video files are carved.
Images	● **Image**s are found partial and complete images from websites (jpgs, pngs and gifs) ● All kind of images such as profile picture, images used in website, bitmap files, thumbnail files and images from search results can be extracted.
HTML	● carved **HTML** pages are fragmented and can be found in all physical memory. ● (Need to analyze these files content…)

Table 15. Summary of memory mapped files by data type

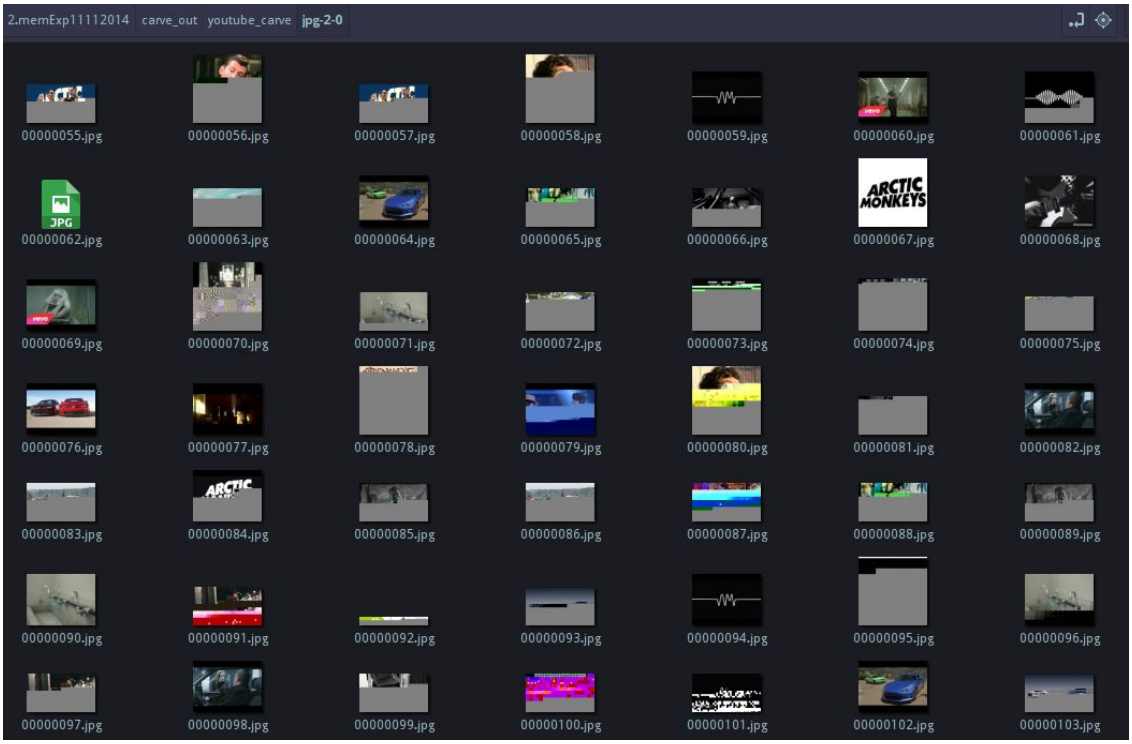


Figure 18: Scalpel output of youtube.com jpg thumbnail files

Browsing websites	**Functions of website**	**Audio**	**Video**	**Images**	**HTML**
youtube.com	Video Streaming	-	-	y	y
9gag.com	Pictures and GIF	-	-	y	y
soundcloud.com	Audio streaming	y(1 truncated mp3)	-	y	y
twitter.com	Social Interaction(Texts,Images)	-	-	y	y
amazon.com	E-Commerce Site (Texts,Images)	-	-	y	y
After Browser session expires	-	-	-	y (images from amazon.com)	y

Table 16. Matrix of memory mapped files via spawned websites

For existing tool evaluation to see its capability of discovering Incognito browsing artifacts,a third-party Volatility based plugin called *Chromehistory*[25] is executed to see its capability to reconstruct sqlite structures from Google Chrome's Incognito mode. The algorithm of the plugin is that it scan the defined signatures of Chrome History sqlite database's tables and reconstruct the hits. The output shows that this approach can not extract Incognito browsing URLs. It can only extracts URLs from normal browsing mode and URLs saved from previous default browser while installing Google Chrome.

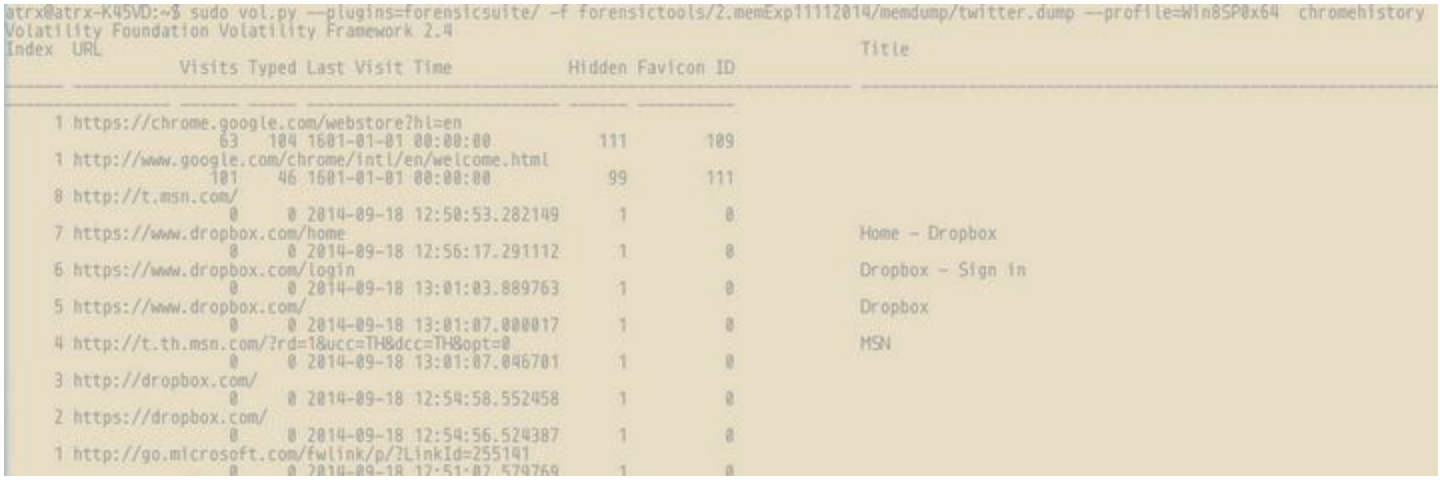


Figure 19: Output of Chromehistrory Volatility plugin

5.3 Summary

To conclude the summary result and findings from the preliminary experiment, Google Chrome browser don't leave browsing artifacts at its data location and SQLite databases. However, traces of browsing data can be seen at pagefile.sys, which is virtual memory paging file of Windows operating system and also at physical memory. Carving memory mapped files from memory dump shows that images and HTML text can be retrieved. Audio & Video files are truncated and full with false positives. Table 17 is the summary matrix of the findings from preliminary experiment.

Analysis of browser history,cache in default location	**Forensic Analysis of acquired disk image**	**Forensic Analysis of Physical Memory**
- No browsing records are	- Pagefile.sys shows traces of	- URLs, Browsing & images can be

appended to Chrome SQLite databases. - From code review of Chromium codebase, it shows that Chrome uses in-memory sqlite database for Incognito browsing mode.	Incognito browsing artifacts.	extracted while the browser session alives. - After the browser session expires, the Chrome background process shows URLs of expired session. - Number of carved files reduced dramatically after browser session expires with slight number of images and text from last opened website.

Table 17. Summary of result of preliminary experiment

In this chapter, it describes about the process of practical study of Google Chrome's private browsing mode with the preliminary experiment and analyze to validate that browsing artifacts from Incognito mode can be discovered and also memory mapped files.

5.4 Cryptographic checksum of acquired digital images

This section provides information about cryptographic checksum of disk image and physical memory for preliminary experiment. The first 3 digital images is from permanent storage experiment and the rest 6 physical memory dump is from volatile data experiment.

#	Digital image	Description	MD5	SHA1
1	browser_sessionAlive.E01	disk image acquired while private browsing session is alive	84a23a21dc9fd733c72e7d14655102d1	5c9fbb425154b49ab8f371495ffe6510f125fda0
2	browser_sessionExpires.E01	disk image acquired after private browsing session expires	4c02c0dddd598b21a903070625e445cb	dd96dc3dc03ff699915ada351ff08c1c7dbb2248
3	System_reboot.E01	disk image acquired after system reboots	7b675c747f5e13aa7233aecc8b59c151	631113ca9cd2a98a77313ba68d0484d0d327f972
4	9gag.dump	memory dump after 9gag.com website is loaded	a02e91d54c6aac5309ad1faee4ea7c88	45648403b689313fc68431ffdd75f405f7985335
5	youtube.dump	memory dump after youtube.com website is loaded	1fbe2ed09ae557538a2059b3d810353b	358fe980052c8ced84c7dcf1befe062b70271b0b

6	soundcloud.dump	memory dump after soundcloud.com website is loaded	82cd0efc137e80be15c952db40ab560f	31a11777f997e2e8fa36e1031a6931f75c8c11f3
7	twitter.dump	memory dump after twitter.com website is loaded	03d85fb95d08265f8967ab97e92da061	4b128d79adce09b0649da73a137ac19e892372e1
8	amazon.dump	memory dump after amazon.com website is loaded	eab8dc96f7e81692dda6cc13e10e027c	2d2c041b8f77a2e47b0d8febb3cb331fc391a1d0
9	after_browsersessionExpires.dump	memory dump after the private browsing session expires	f1d0f8717adac770626273b4b6973b65	1b196043f6d30e3f4a158afb52b126522b8ba052

Table 18. Cryptographic checksums of acquired digital images

6.Research Plan

In this section, it will describes the research plan with timeline and objectives to carry out analysis of Chrome Incognito browsing artifacts from persistent storage and physical memory. The following listing is the analysis tasks to continue to extract and study Incognito browsing artifacts from Google Chrome.

1. **Page file Analysis**
   From Persistent storage experiment, traces of Incognito browsing artifacts can be seen at pagefile.sys, which is the swap file of Windows operating system and swap space analysis can give information which can not find on disk. In the case of the power-off suspect system, swap file is the only source to analyze memory contents.
2. **Code Review & Signature Analysis of Private Browsing Data Structure**
   From Chromium open source project, it is possible to review the codebase of Chrome browser. From reverse engineering of Chrome's Incognito browsing mode, it is possible to analyze important data structures and scan through physical memory with these discovered signatures to extract traces and valuable information about Incognito mode.
3. **Evaluation of Experiment**
   The evaluation of experiment will be executed with proportion of data recovery by comparing actual data loaded to each website at the time of acquisition with data recovery from physical memory dump.
4. **Document Write-Up**
   The manuscript will be updated with result from the analysis and evaluation.

The following gantt chart describes the schedule of the research plan.

#	Task Name	Start	Finish	Duration
1	Virtual Memory (Pagefile.sys) Analysis	9/1/2015	9/14/2015	2w
2	Code Review & Signature Analysis of Private Browsing Data Structure	9/14/2015	9/25/2015	2w
3	Evaluation of experiment	9/28/2015	10/9/2015	2w
4	Documentation Write-up	10/12/2015	10/23/2015	2w

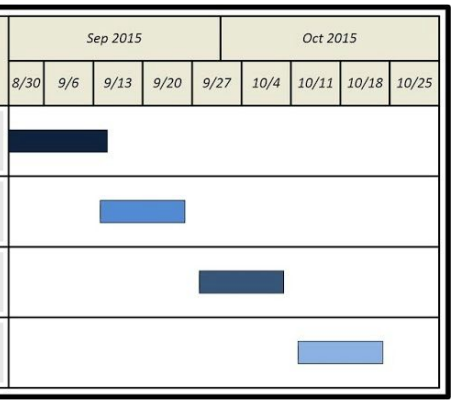


Figure 20: Research plan

7.References

[1] G. Aggarwal and E. Bursztein, "An Analysis of Private Browsing Modes in Modern Browsers.," USENIX Secur. ..., pp. 1–8, 2010.

[2] https://support.google.com/chrome/answer/95464?hl=en

[3] H. Chivers, "Private browsing: A window of forensic opportunity," [1] H. Chivers, "Private Brows. A Wind. forensic Oppor. Digit. Investig., 2013.Digital Investig., 2013.

[4] M. T. Pereira, "Forensic analysis of the Firefox 3 Internet history and recovery of deleted SQLite records," Digit. Investig., vol. 5, no. 3–4, pp. 93–103, Mar. 2009.

[5] K. Jones and Keith J. Jones, "Forensic Analysis of Internet Explorer Activity Files," Forensic Anal. Microsoft Wind. ..., 2003.

[6] The Art of Memory Forensics. .

[7] A. Mahendrakar, J. Irving, S. Patel, Aditya Mahendrakar, and James Irving, "Forensic Analysis of Private Browsing Mode in Popular Browsers," mocktest.net, 2010.

[8] J. Oh, S. Lee, and S. Lee, "Advanced evidence collection and analysis of web browser activity," Digit. Investig., vol. 8, pp. S62–S70, Aug. 2011.

[9] D. Ohana and N. Shashidhar, "Do private and portable web browsers leave incriminating evidence?: a forensic analysis of residual artifacts from private and portable web browsing sessions," EURASIP J. Inf. Secur., pp. 135–142, May 2013.

[10] H. Said and N. Al Mutawa, "Forensic analysis of private browsing artifacts," ... Technol. (IIT), 2011 ..., pp. 197–202, Apr. 2011.

[11] K. Satvat, M. Forshaw, F. Hao, and E. Toreini, "On the privacy of private browsing – A forensic approach," J. Inf. Secur. Appl., Apr. 2014.

[12] M. Pollitt, "An ad hoc review of digital forensic models," Syst. Approaches to Digit. Forensic ..., pp. 43–54, 2007.

[13] S. L. Garfinkel, "Digital forensics research: The next 10 years," Digit. Investig., vol. 7, Supplem, pp. S64–S73, Aug. 2010.

[14] K. Sansurooah, "Taxonomy of computer forensics methodologies and procedures for digital evidence seizure.," Aust. Digit. Forensics Conf., Apr. 2006.

[15] S. Vömel and F. C. Freiling, "A survey of main memory acquisition and analysis techniques for the windows operating system," Digit. Investig., vol. 8, no. 1, pp. 3–22, Jul. 2011.

[16] M. Simon and J. Slay, "Enhancement of Forensic Computing Investigations through Memory Forensic Techniques," in 2009 International Conference on Availability, Reliability and Security, 2009, pp. 995–1000.

[17] "A Study of Application Level Information From The Volatile Memory of Windows Computer Systems Funminiyi Akanfe Olajide September 2011," no. September, pp. 3–206, 2011.

[18] M. Graziano, A. Lanzi, and D. Balzarotti, "Hypervisor Memory Forensics."

[19] L. Yin, "Research on Windows Physical Memory Forensic Analysis," 2012 Fourth Int. Symp. Inf. Sci. Eng., vol. 5, pp. 493–496, Dec. 2012.

[20] B. S. Lerner, L. Elberty, N. Poole, and S. Krishnamurthi, "Verifying web browser extensions' compliance with private-browsing mode," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2013, vol. 8134 LNCS, pp. 57–74.

[21] A. Schuster, "Searching for processes and threads in Microsoft Windows memory dumps," Digit. Investig., vol. 3, pp. 10–16, Sep. 2006.

[22] S. M. Hejazi, C. Talhi, and M. Debbabi, "Extraction of forensically sensitive information from windows physical memory," Digit. Investig., vol. 6, pp. S121–S131, Sep. 2009.

[23] http://www.w3counter.com/globalstats.php?year=2015&month=7

[24] Brezinski, D., & Killalea, T. (2002). Guidelines for Evidence Collection and Archiving. Retrieved

[25] https://github.com/superponible/volatility-plugins