12-4-2013

Forensic Memory Dump Analysis And Recovery Of The Artefacts Of Using Tor Bundle Browser – The Need

Divya Dayalamurthy

Edith Cowan University

DOI: 10.4225/75/57b3c7f3fb86e

11th Australian Digital Forensics Conference. Held on the 2nd-4th December, 2013 at Edith Cowan University, Perth, Western Australia

FORENSIC MEMORY DUMP ANALYSIS AND RECOVERY OF THE ARTEFACTS OF USING TOR BUNDLE BROWSER – THE NEED

Divya Dayalamurthy
School of Computer and Security Science
Edith Cowan University, Perth, Australia
ddayalam@our.ecu.edu.au

Abstract

The Onion Routing (TOR) project is a network of virtual tunnels that facilitates secure, private communications on the internet. A recent article published in "The Registry" claims that TOR bundle browser usage has increased in recent years; statistics show that in January 2012, there were approximately 950,000 users globally and now in August 2013 that figure is estimated to have reached 1,200,000 users. The report also illustrates that The United states of America and the United Kingdom are major contributors towards the massive increase in TOR usage. Similarly, other countries like India and Brazil have increased usage to 32,000 and 85,000 respectively. This research paper will be an introduction and identifies the need for research in this area, and provides a literature review on existing research. The objective of this paper is to discuss the existing methodologies for analysing forensic artefacts from RAM from the use of the TOR browser bundle and to propose a synthesized forensic analysis framework that can be used for analysing TOR artefacts.

Keywords

Darknet, TOR, TAILS, Cybercrime

INTRODUCTION

Cyberwarzone reports claim that darknet contains index of more than 70,000 websites hosted illegally which are not visible to normal internet user. Some of the anonymous browsers are TOR bundle browser, Relakks, Waste Again, and freenet that can be used for establishing connection to darknet. As installation and usage of the anonymous browsers is very simple and user friendly, there is aamount of users accessing darknets for unethical or illegal activities(Fachkha et al., 2012).

One of the main reasons for this spike is because they provide freedom of speech to the users and also encrypts the traffic, thus hiding the identity of the users to some extent. At the same time, this has become a great benefit for cybercriminals and unethical users for accessing all the available illegal services or hosted hidden links. Therefore this easy access to darknet has hiked crime rates involving illegal activities such as online drug dealing, child pornography, hidden wiki hosting devices, crime-as-a-service(Sharwood, 2013).

According to the article by Cyberwarzone, reports claim that darknet contains nearly 600TB of data and 500 times larger than normal internet visibility(CWZ, 2013). As the majority of the content available are illegal online stores, assassins, drug dealers, stolen goods sale, and medium for terrorist communications; this makes darknet content more dangerous to surf.

In this current scenario, when a user uses TOR bundle browser for accessing illegal information, the lackof monitoring control makes it difficult to forensically detect and identify the traffic. The significance of the study is required to develop a forensic sound methodology to detect TOR bundle browser usage from memory dump of the windows machine. This enables analysis and recovery of the artefacts using TOR bundle browser for organization or even jurisdiction purpose (Murdoch & Danezis, 2005). However, recent arrest of The Silk Road founder has given additional hope of identifying the users accessing darknet for illegal purposes.

Many authors have perused studies and research on darknet and different ways for establishing connection through many anonymous browsers. This includes detection of TOR traffic on network infrastructure, detecting TOR traffic on a windows operating system, and general memory dump analysis. But there were no proper research being aimed to cover memory dump forensic specifically for analysing artefacts of TOR bundle browser usage. Thus, there is a huge gap in this area without being explored – and this research will focus to address this gap and aims to propose a methodology which synthesizes memory dump forensic analysis and detection of TOR bundle browser artefacts.

TOR BUNDLE BROWSER – THE WORKING

TOR uses a methodology called onion routing consisting of TOR relays, volunteers TOR servers, exit nodes. When a user installs the TOR bundle browser and runs it; it automatically establishes connection and is ready for the user. Figure 1, illustrates the TOR bundle browser consisting of three components for sending encrypted traffic, namely an entry node, a middleman, and an exit node. When a user initiates a connection it first reaches the entry node and is then sent through multiple middle-man nodes which volunteers to this traffic and goes to the exist node and then to the destination server. TOR uses public key for encrypting the transmitting messages for multiple layers until the exit nodes. However, the traffic from the exit node to the destination server is unencrypted(Biryukov, Pustogarov, & Weinmann, 2013).

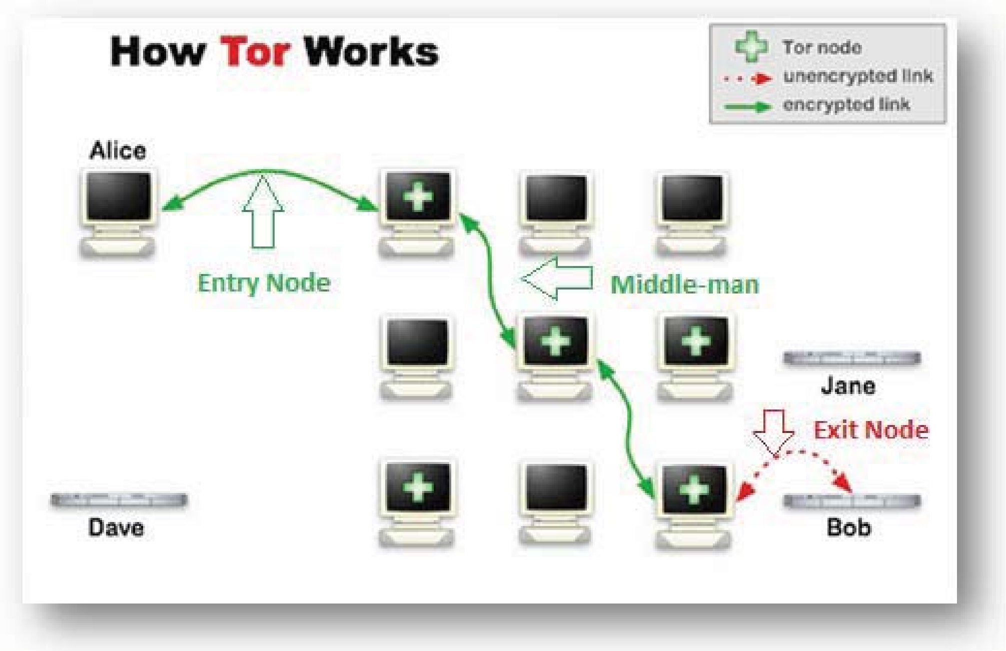


Figure 1: TOR bundle Browser Working

On analysing the TOR bundle browser operation and workings, it is evident that when a user establishes TCP connection it highly impossible to detect the traffic since it has multi-layer of encryption. Hence, detecting the TOR traffic has always been extremely important from a security perspective and forensic recovery perspective as well.

Published research available on TOR bundle browser memory dump forensic analysis is currently very limited. Existing research is available for:

- General memory dump forensic analysis and recovery.
- Monitoring or detection of TOR browser in a network.

There is a huge gap in this area – and this research aims to address this gap using the proposed forensic process.

EXISTING DETECTION METHODS AVAILABLE

Artefacts from Operating System

In reference to the research paper by Runa A.Sandvik (2013), the author has illustrated the forensic analysis and documented traceable evidence that TOR bundle browser can leave in a windows machine. The author has also analysed and recorded the directory path of TOR bundle browser artefacts acquired from windows OS. From the research paper, the path or directories with TOR artefacts identified were the following:

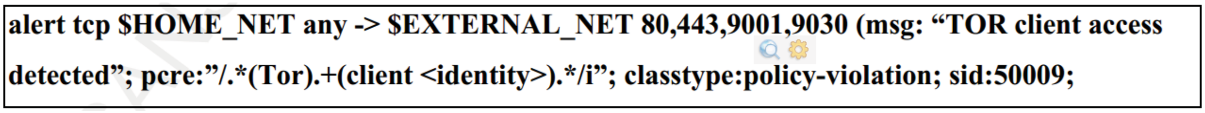
- Prefetch folder C:\Windows\Prefetch\.
- Thumbnail cache memory
- Windows Paging File
- Windows Registry {Sandvik, 2013 #49}.

A similar analysis was performed by Andrew Case on *"De-Anonymizing Live CDs through Physical Memory Analysis",*where the author has discussed different forensic techniques for recovery of using The Amnesic Incognito Live System (TAILS)(Case, n.d.). The TAILS is a LIVE operating system (Linux) bootable from DVD or any portable device. All internet connection is established and traffic is forced to pass through TOR network. In this paper, the author has covered small sections on initial memory dump analysis for forensically retrieving artefacts. The author illustrates that Python scripts can be used to analyse specific TOR data structures where information regarding the artefacts are stored. However, the author hasn't proved the script's practical workability for artefacts recovery {Case, n.d. #50}(Dodge, Mullins, Peterson, & Okolica, 2010)(Sutherland, Evans, Tryfonas, & Blyth, 2008).

Artefacts from Network Traffic

As the TOR bundle browser encrypts all possible traffic sent through TOR entry node, middleman, and exit nodes by adding high level multi-layer encryption each time it passes through a middle-man node. Detecting the TOR traffic and any kind of proxy usage is highly essential on network forensic perspective.(Fachkha et al., 2012)(Berthier & Cukier, 2008).

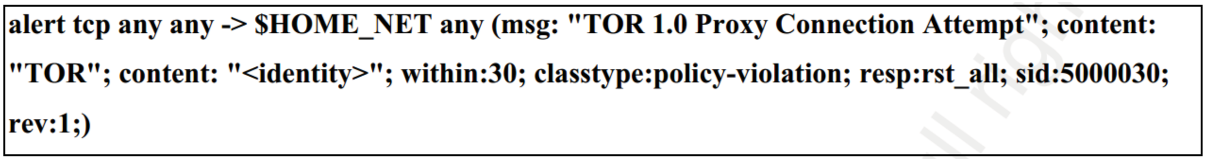
Author John Brozycki, in the research paper *"Detecting and Preventing Anonymous Proxy usage"* has penned down the techniques that could be used for detecting any anonymous proxies by using SNORT rules. For this purpose, the author used Vidalia package – a TOR package for establishing TOR connections. The author claims that using the below SNORT rule as given in figure 2, can detect TOR bundle browser traffic {Brozycki, 2008 #36}(Mizoguchi, Fukushima, Kasahara, Hori, & Sakurai, 2010).



```
alert tcp $HOME_NET any -> $EXTERNAL_NET 80,443,9001,9030 (msg: "TOR client access
detected"; pcre:"/.*(Tor).+(client <identity>).*/i"; classtype:policy-violation; sid:50009;
```

Figure 2: SNORT rule developed by John Brozycki

A similar method was also supported by David's SNORT rule (figure 3), founder of Seclits blog. The author stated that it could also detect the TOR browser usage. However, there were not any proven results documented in the paper.



```
alert tcp any any -> $HOME_NET any (msg: "TOR 1.0 Proxy Connection Attempt"; content:
"TOR"; content: "<identity>"; within:30; classtype:policy-violation; resp:rst_all; sid:5000030;
rev:1;)
```

Figure 3: SNORT rule developed by David

{Brozycki, 2008 #36}

Furthermore, Sambuddho et al(Sambuddho Chakravarty) has performed research on detecting TOR traffic using Decoys. For this research, the author used decoy traffic (bait) for detecting the proxy connections including TOR bundle browser traffic. In this paper, the authors have created many bait servers and made the servers available in the TOR network. This implementation has been performed by using IMAP and SMTP servers and entice credentials have been created so that cybercriminals can be attracted to access their services in their hosted servers. Figure 4, illustrates the "Rogue exit node" which acts as a bait for TOR users, all the traffic passing through rogue exit node is recorded. This setup was kept online or active for duration of ten months and different activities trying to compromise the servers using HTTP session hijacking attacks were monitored and recorded.

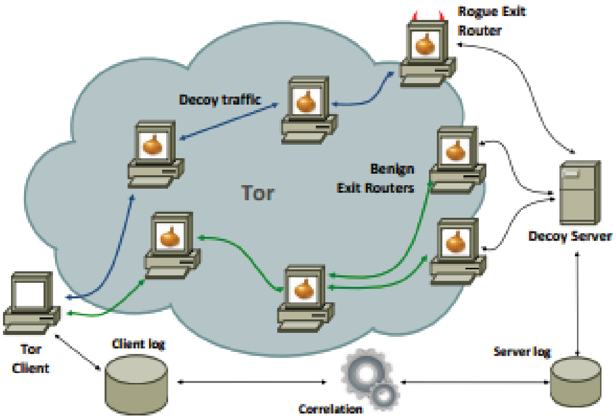


Figure 4: Decoy traffic for detecting the proxy connections(Sambuddho Chakravarty, 2011)

The author states that this research paper will be beneficial to identify and analyse the behaviour and statistic of activities on the rogue exit node. As stated earlier, the traffic from and to exit nodes are the only unencrypted traffic by the TOR network. However, when passing through the rogue exit node; this setup can collect statistics and particulars of the hosts connecting through that compromised exit node and it cannot trace other middleman nodes (previous hops) involved in the TOR network routing. This method reveals the IP address of the hosts entering the exit node but not the actual IP address of the machine or even any of the middleman nodes involved in the TOR network (Xuefeng, Yong, & XiaMu, 2008).

Malicious browser plugins

In August 2013, it was confirmed that attackers had exploited the vulnerability in TOR bundle browser Firefox plugin which can disclose the source IP address of the TOR users. This malicious code was injected from a darknet host called "Freedom Hosting", the users who visited this hidden service website were compromised as it exploits the memory-management vulnerability in Firefox browser (Goodin, 2013). This malicious JavaScript would make Firefox send a unique identifier to a public server by which the source IP address can be traced back. I addition to this, a reverse engineering security specialist claimed that this malicious code reveals some of the IP address in Reston, Virgina(POULSEN, 2013).

However, if the TOR user does not visit that compromised hosted website then, the chance of Firefox plugin being exploited is comparatively low to detect the TOR artefacts. On the other hand, there is high possibility of retrieving significant forensic evidence of TOR bundle browser artefacts from a windows machine using the "Memory Dump". The next section willaimn to provide literature review on memory dump forensic analysis(Brozycki, 2008).

MEMORY DUMP ANALYSIS

Why memory dump analysis

Memory dump analysis has always been a critical and interesting area for forensic investigations. The information stored in the memory of the computer has significant importance. For example, when a cybercriminal uses bootable LIVE CD or USB such as TAILS with a windows or Linux operating system, then no significant information is stored in the physical host computer. This is because the machine boots from the CD or portable USB with self-contained hard drive and even if the physical computer is captured and forensically analysed not much evidence can be retrieved.

Consider, a suspect uses TAILS for connecting to some illegal darknet website. In this scenario, all the internet connections established from that machine are forced to go through the TOR network with multi-level encryption and thus the identity of the user is hidden to some extent – leaving behind no potential evidence. Thus, retrieving the memory dump from the suspect machine and analysing it forensically could provide more forensic evidence of TOR bundle browser usage (Aljaedi, Lindskog, Zavarsky, Ruhl, & Almari, 2011).

What information are stored in memory dump?

The glimpse of the information stored in the memory dump are given below:

- All the details about the image including date, time, and CPU usage are recorded.
- Processes, process ID – all running process in the operating system.
- Network connections – what network connections were available at the time of memory dump captured.
- DLLs, memory maps, objects, encryption keys.
- Programs, hidden programs, rootkits, promiscuous codes.
- Registry information of the operating system.
- API functions, system call tables.
- Graphic contents.

As the memory dump contains significant information, analysing it forensically will help to detect, recover and analyse artefacts of TOR bundle browser usage.

TOOLS AND TECHNIQUES AVAILABLE FOR RETRIEVING DIFFERENT TYPE OF EVIDENCE FROM MEMORYDUMP

Extraction of graphic content

In reference to the research paper by Stephan et al (2011), the authors illustrated the process (figure 5) for extracting the graphic content information from the memory dump of windows based machine(Kiltz, Hoppe, & Dittmann, 2009). In addition, the authors also developed a forensic model (figure 6) used for this research. The extraction process (figure 5) used by Stephan et al (2011) involves strategic preparation, operational preparation, data gathering, data investigation, data analysis, and final documentation with evidence presentation .

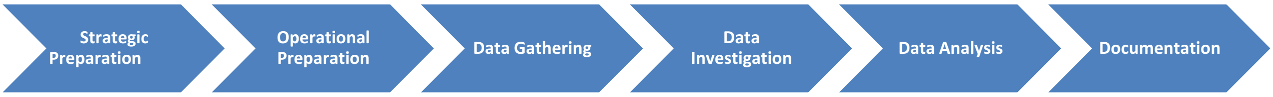


Figure 5: Extraction Process of graphic content from memory dump

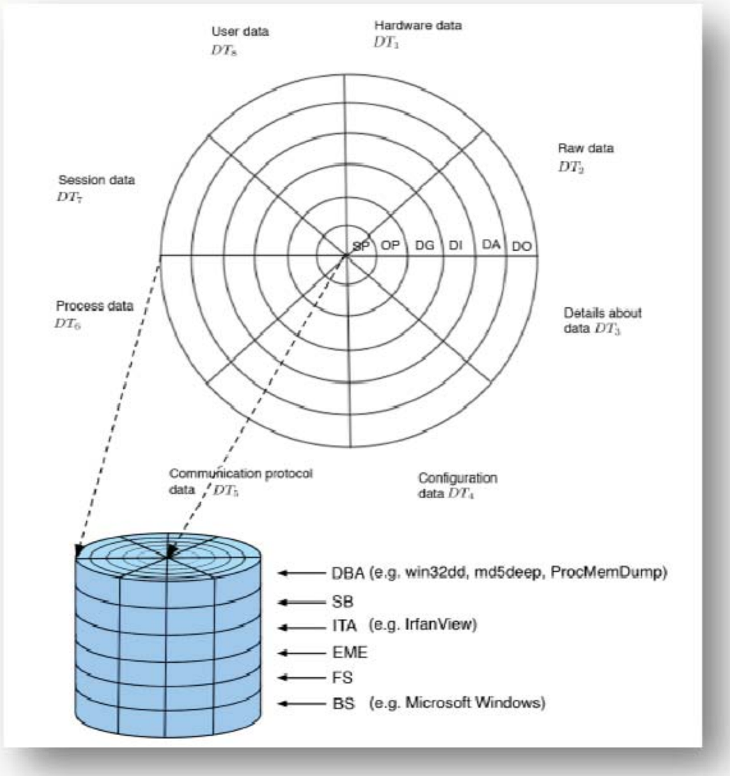

User data
DT_8
Hardware data
DT_1
Raw data
DT_2
Session data
DT_7
SP
OP
DG
DI
DA
DO
Process data
DT_6
Details about data DT_3
Communication protocol data DT_5
Configuration data DT_4
DBA (e.g. win32dd, md5deep, ProcMemDump)
SB
ITA (e.g. IrfanView)
EME
FS
BS (e.g. Microsoft Windows)

Figure 6: Stephan et al (2011) proposed Forensic model

Figure 6, illustrates the proposed forensic model with different data type for extracting graphic content. According to the author, forensically relevant data types are hardware data, raw data, details about data, configuration data, communication protocol data, process data, session data, and user data. In addition, the other tools used to retrieve the graphic content from the memory dump were Irfan View and volatility framework(Kiltz et al., 2009) (McRee).

Volatility memory analysis

In line with this analysis, the volatility storage medium plays a vital part in this section. This is also termed as temporary memory in some area of research which stores information about running process, process ID, DLLs information and other forensic evidences. According to the paper *"Techniques and Tools for Recovering and Analysing Data from Volatile Memory"(Amari*) published by SANS Institute the author claims possible ways to acquire the memory as listed below:

Hardware-based acquisition:

This process requires the computer to be suspended and DMA (direct memory access) is used to obtain the copy of the memory dump.

Software based acquisition:

This method is easier and more user friendly for acquiring the volatile memory using software such as memorydump or dd. This technique is widely used in the forensic analysis area. In addition, the author has explained how the volatile memory works in windows and Linux operating systems. He claims that the memory-map files can be recovered by viewing the root of the VAD tree which stores specific information about the processes, events, application errors, logs. The other significant area in windows based system for memory analysis is "control area" which could help in retrieving the information regarding the file names and data stored. Similarly, in Linux system the data structure is termed as inode. Furthermore, the paper also documents the available memory forensic tools that can be used for volatility memory analysis. (Amari)(Mrdovic, Huseinovic, & Zajko, 2009)

A similar memory dump analysis using volatility framework was performed by Chad Tilbury and cheat sheet was developed named ***"Memory Forensics Cheat sheet v1.1"***. In this paper, the author documented the different commands that can be used to trace different types of artefacts using volatility framework as shown in table 1(Tilbury, n.d.)

Volatility Framework on memory dump analysis	
Registry Analysis Artefacts	Hivelist - # vol.py hivelist Hivedump - # vol.py hivedump –o 0xe1a14b60 Printkey - # vol.py printkey –K Userassist - # vol.py userassist Hashdump - # vol.py hashdump –y 0x8781c008 –s 0x87f6b9c8
Rootkit Artefacts	Psxview - # vol.py psxview Driverscan - # vol.py driverscan Apihooks - # vol.py apihooks Ssdt - # vol.py ssdt | egrep –v Driverirp - # vol.py driverirp –r tcpip Idt - # vol.py idt
Network Artefacts	Connections - # vol.py connections Connscan - # vol.py connscan Sockets - # vol.py sockets Sockscan - # vol.py sockscan Netscan - # vol.py netscan
Rogue Process Artefacts	Pslist - # vol.py pslist Psscan - # vol.py psscan Pstree - # vol.py pstree
Promiscuous Mode Artefacts	Malfind - # vol.py malfind --dump-dir ./output_dir Ldrmodules - # vol.py ldrmodules –p 868 -v
Promiscuous Process and Drivers	Dlldump - # vol.py dlldump --dump-dir ./output –r metsrv Moddump - # vol.py moddump --dump-dir ./output –r gaopdx Procmemdump - # vol.py procmemdump --dump-dir ./out –p 868 Memdump - # vol.py memdump –dump-dir ./output –p 868

Table 1: Memory Forensics Cheat sheet v1.1by Chad Tilbury

Recovering Windows registry information from memory dump

Windows registry is significant area for recovering and analysing potential evidence about each event on a windows machine. Shuhui Zhang et al (2011) have proposed a method for recovering windows registry information from the memory dump. Figure 7 is the proposed flowchart by Shuhui Zhang et al (2011) for extracting the HIVE files from the memory dump. Since the Hive files contain all the necessary information including metadata, handles, objects, keys, data structures and file maps which are potentially significant for forensic memory analysis.

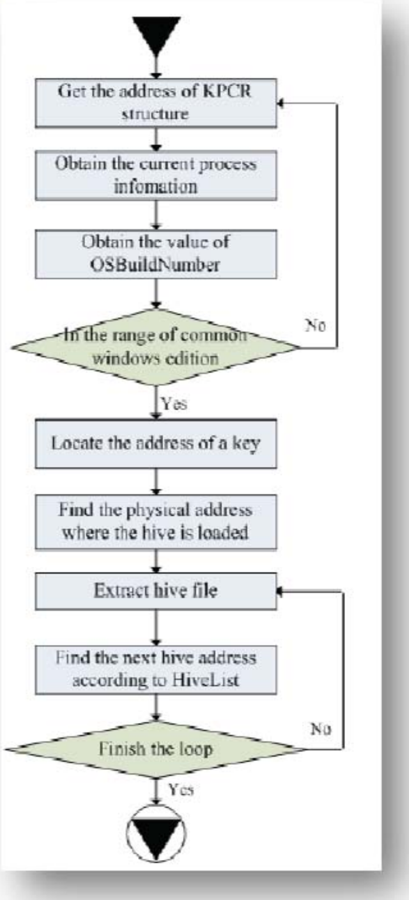

Get the address of KPCR structure
Obtain the current process infomation
Obtain the value of OSBuildNumber
In the range of common windows edition
No
Yes
Locate the address of a key
Find the physical address where the hive is loaded
Extract hive file
Find the next hive address according to HiveList
Finish the loop
No
Yes

Figure 7: Proposed Methodology for recovering Windows Registry Artefacts

(Shuhui, Lianhai, & Lei, 2011)

A similar paper was written by Farmer et al (n.d.) on forensic analysis of windows registry. However, in this research paper, the authors have illustrated forensic analysis by directly viewing the windows registry and not analysing from the memory dump. The author have analysed and documented the possible evidences that could be recovered from windows registry such as:

- Registry Hive Locations
- MRU Lists (Most Recently Used List)
- Wireless Networks details
- Network details
- LAN computer connected through the machine
- Portable devices connected
- Artefacts of IE
- Windows passwords
- IM chat details

Literature Review Matrix

	Memory Dump Analysis	**TOR Browser Detection**
Windows | Windows Registry OS Analysis	✓	✓
Volatile Memory Data Analysis | Volatility Framework	✓	✗
Network Detection Analysis	✓	✓
Other Operating system forensic analysis	✓	✗

Table 2: Literature Review Matrix

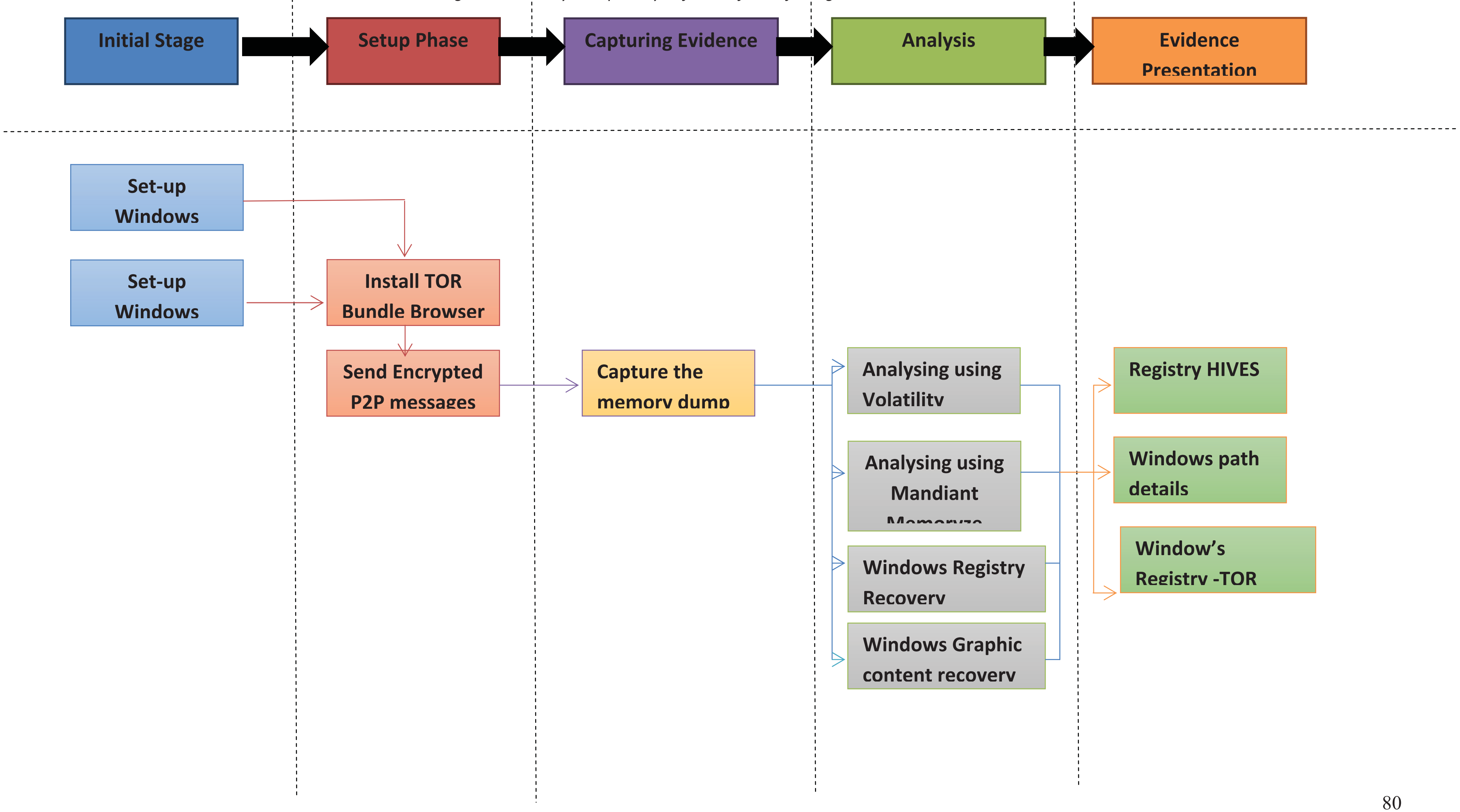
✓ - Research area covered | ✗ - Research area with scope or yet to explore!

Table 2 shows summarization of the literature review on techniques available for retrieving TOR bundle browser artefacts. It is evident that many researchers have explored the techniques for recovering evidence of TOR traffic on networks and by setting up bait exit node (rogue exit node). Similarly, researches have carried on retrieving the artefacts of TOR usage from Windows and Linux operating systems. However, no research materials were found on recovering and analysing artefacts of TOR bundle browser from the memory dump of a windows machine. To move forward with this project, this lack of coverage, or gap will be addressed and a methodology will be proposed to analyse the TOR bundle browser artefacts from the memory dump of a windows machine.

THEORITICAL FRAMEWORK

Figure 8 illustrates the theoretical framework of how the research / experiment will be performed to analyse the memory dump to recover the artefacts of TOR bundle browser usage. In addition, the main agenda is to generate a synthesis methodology by combining TOR bundle browser detection and memory dump analysis.

Figure 8: Memory Dump Analysis for Artefacts of using TOR Bundle Browser



1. **Initial Phase**
   Two windows virtual machines will be configured.

2. **Setup Phase**
   The PC will be installed with latest version of TOR bundle browser for this project. Communications will be established via TOR network to the two machines and illegal hidden services will not be examined. For this reason, both the end points are set up by the author and only the TOR network is being used to route traffic.

3. **Capturing Evidence**
   After successfully sending encrypted messages from end point to other, each of the PC's memory will be captured and memory dump will be created using software "Dumpit" in a forensically sound manner. The created memory dump will be hashed to maintaining integrity and copies will be made before performing the analysis using forensic tools.

4. **Analysis**
   Analysis and examination will be performed using memory dump forensic tools such as the volatility framework and Mandiant Memoryze analyser for analysing TOR bundle browser artefacts. In addition, as stated by Stephan et al (2011) the windows graphic content evidence in reference to TOR traffic will be analysed. Furthermore, Shuhui Zhang et al (2011) methodology for recovering the windows registry information from the memory dump will be performed and log event running sheet as given in table 3 will be maintained capturing all the events.

Table 3: Sample Log Event Running Sheet

Date	Time	Comments	Action Taken By	Investigator 1

5. **Evidence Presentation**
   This section will ain to provide the artefacts and evidence from registry HIVES, windows directory, and windows registry details. The evidence and artefacts that were identified and recovered from the previous analysis step will be presented in forensic report manner. All the recovered evidence will be hashed and log of events will show the analysis flow followed by the author. Finally, the evidence recovered from volatility framework and Mandiant Memoryze will be documented (refer table 4).

Table 4: Sample Evidence Presentation sheet

Date		**Time**		**Filename**		**Investigator**	
Path Found							
Hash Value	**MD5**		**SHA1**				
Evidence File		**Evidence Comments**					

Research Framework

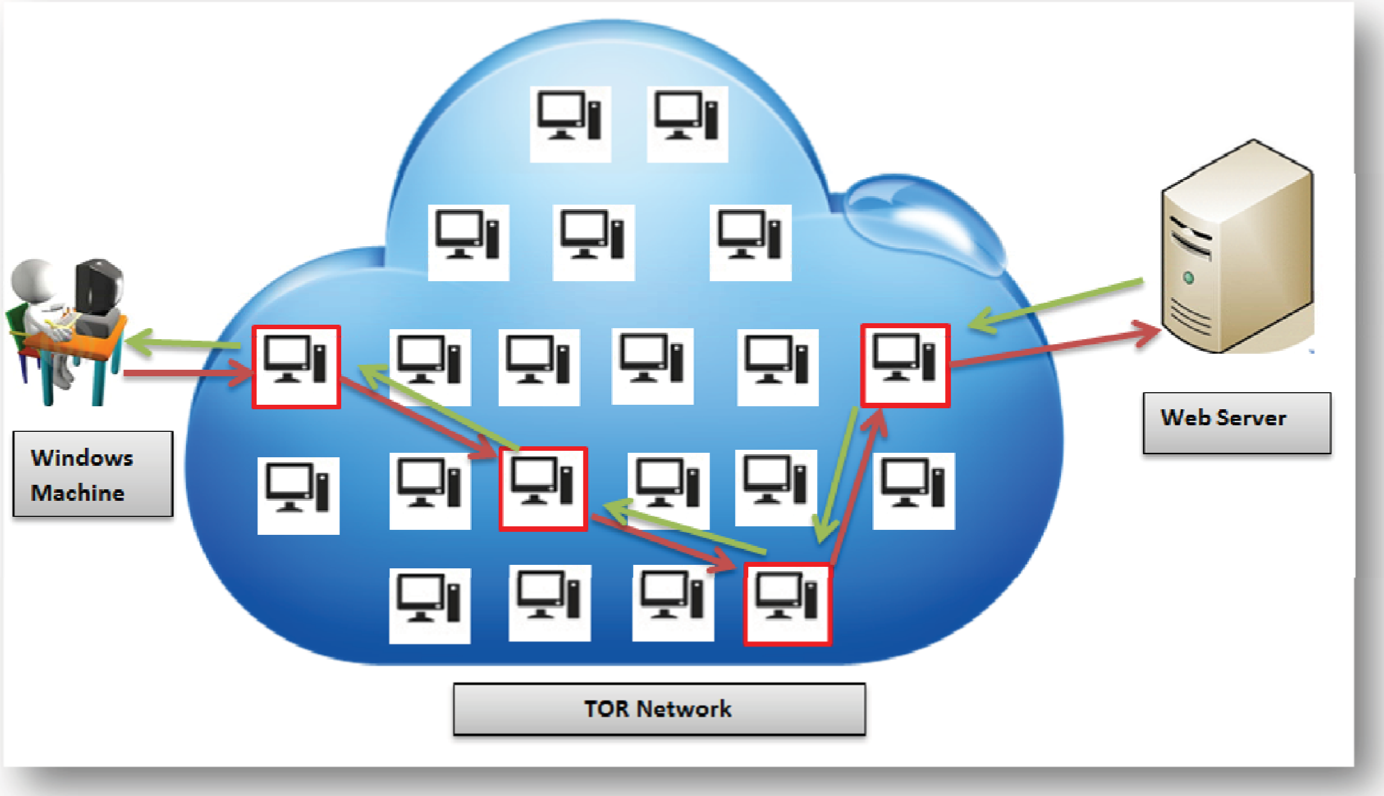


Figure 10: Proposed Research Environment for this Project

Figure 10 illustrates the proposed research environment setup for perusing this project. In this scenario, two end points are be arranged by the author, one of the end points is a machine with windows operating system and the second end point will be a windows web server. Both the windows machines will be installed with TOR bundle browser with appropriate plugins. Once the setup is completed, the windows machine will access files or communicate with the web server and route only TOR traffic. As depicted in the above figure 10, the communication between the windows machine and web server is carried by volunteered TOR nodes in the TOR network. These are the machine all over the world with TOR bundle browser installed. After accessing the files using TOR network, the memory dump of the windows machine will be captured in forensically sound manner using “dumpit” software. Finally, the memory dump will be analysed and artefacts of TOR bundle browser and any other suspicious proxy traffic will be recorded and presented in the evidence presentation section.

LIMITATIONS

A current limitation in TOR bundle browser detection research is that there a is lack of established research-driven methods available for retrieving and recovering TOR artefacts from memory dump. This area is so new, that there is no material or and have been no experiments conducted by industry and researchers on analysing and retrieving artefacts evidence from memory dump of TOR browser usage. A challenge for this researchis to address this limitation as this research will be first of its kind, and to establish methodologies for testing and research.

FUTURE WORK

Due to the limitations and time constraints the proposed methodology hasn't been implemented and tested. As a future work, the proposed methodology will be tested as illustrated in the forensic process and evidence will be presented in the next paper as a prototype model.

CONCLUSION

This main objective of this paper was to present the need of research in memory dump analysis on detecting TOR bundle browser. This would be first step to detect the usage of TOR from the physical machine. Once the proposed synthesis methodology is developed and tested; this can also be used by organisations for monitoring, analysing or providing evidence for using TOR bundle browser. This area needs more attention and more research will be performed once this methodology has been developed.

REFERENCES

Amari, Kristine. Techniques and Tools for Recovering and Analyzing

Data from Volatile Memory.

CWZ. (2013). Guide: How to find the darknet or deepweb.

Kiltz, S., Hoppe, T., & Dittmann, J. (2009, 5-7 July 2009). *A new forensic model and its application to the collection, extraction and long term storage of screen content off a memory dump.* Paper presented at the Digital Signal Processing, 2009 16th International Conference on.

Mrdovic, S., Huseinovic, A., & Zajko, E. (2009, 29-31 Oct. 2009). *Combining static and live digital forensic analysis in virtual environment.* Paper presented at the Information, Communication and Automation Technologies, 2009. ICAT 2009. XXII International Symposium on.

Sambuddho Chakravarty, Georgios Portokalidis, Michalis Polychronakis, and Angelos D. Keromytis. (2011). Detecting Traffc Snooping in Tor Using Decoys.

Sharwood, Simon. (2013). Silent Circle shutters email service.

Shuhui, Zhang, Lianhai, Wang, & Lei, Zhang. (2011, 11-13 March 2011). *Extracting windows registry information from physical memory.* Paper presented at the Computer Research and Development (ICCRD), 2011 3rd International Conference on.

Tilbury, Chad. (n.d.). Memory Forensics Cheat Sheet v1.1.