Online Anonymity: Forensic Analysis of the Tor Browser Bundle

Winkler Darcie, BS[1]; Robert J. Boggs, Cpl.[2]; John Sammons MS[3]; Terry Fenger, PhD[1]

[1]Department of Forensic Science, Marshall University
[2]West Virginia State Police Digital Forensics Unit
[3]Department of Integrated Science and Technology, Marshall University

Abstract

The Onion Router (Tor) is a network of encrypted onion routers that helps to increase the level of anonymity experienced by its users. The security and privacy provided by the Tor Browser was originally intended to protect the communication of the government, journalists, and non-governmental organizations. Additionally, however, it is a facilitator for individuals participating in illicit activities. The National Security Agency (NSA) was able to crack through the anonymity provided by Tor using network forensics to take down Silk Road, a black market version of Amazon. It is hoped that beneficial information for dead-box and network forensics will become evident by capturing packets while the Tor Browser is navigating to .onion and .com websites, dumping the random-access memory (RAM), and comparing versions of the registry from various points of the installation process. Virtual machines (VM) were constructed to test four possible scenarios: a machine running on Internet Explorer alone, a machine with Tor downloaded but no active use recorded, a machine with Tor downloaded where it was used to navigate both the Internet and Darknet, and finally a machine where Tor had been installed, used, and then uninstalled. Furthermore, an additional VM was created solely for the purpose of tracking registry changes through the course of installing and uninstalling the Tor Browser Bundle. If this methodology is executed and the data collected is analyzed using pre-existing forensic techniques, then relevant evidence of the browsing history and usage of Tor will be acquired. Any evidence recovered from this forensic analysis and artifact recovery will be of significant importance to digital forensic investigators.

Each of the three methods employed reaped beneficial artifacts pertaining to the usage of Tor. The RAM dump provided several different file types of carved data that linked back to the browsing protocol used. The Registry files indicated that Tor was downloaded and executed on the machine and RegShot, open-source software that allows for the comparison of the state of the Registry at different moments in time, provided the types of changes made to the registry during installation and uninstallation of Tor. The packet capture indicated Tor usage in that the traffic stream and the Protocol Hierarchy Statistics were inconsistent with the appearance of normal traffic. A significant amount of evidence proved that Tor does not provide complete anonymity. The RAM dump and packet capture would appear to be more applicable in a network forensics environment using a remote process to monitor a suspect's activity. The techniques used to analyze the Registry, however, may still be beneficial in a digital lab that is restricted to dead-box forensics. To continue this area of research, it would be interesting to see a patch incorporated into WireShark® called Tor Dissector to see if it would adequately decrypt the traffic from using the Tor Browser. Additionally, the browsing protocol used in this research did not include the active downloading or saving of any files or images by the user, which may show up in a RAM dump.

Introduction

Technology tends to be a necessity in the lives of many people. From your professional to personal life, it is so readily available that it encroaches on all aspects of our lives. With greater advancements in the technology that we rely so heavily upon, it is obvious that we would also need equivalently sophisticated safety and security measures. Travelling to businesses, stores, and other destinations may soon become unnecessary due to the convenience and false sense of security provided by online transactions. Safeguards, such as Tor, not only protect those using online commerce for legitimate purposes but also those taking advantage of them to participate in illicit activities. This requires forensic investigation so that these crimes can be brought to justice.

While browsing the Internet, onion routers (OR) are capable of providing a degree of anonymity, which is defined as "the state of not being identifiable within the set of subjects" (8). The primary goal of various forms of digital communication, such as onion routers, is to hinder third parties from performing traffic analysis (3). Traffic analysis is "the examination of network traffic flows to establish identities of parties involved" (7). The OR structure is composed of encryption on top of encryption, similar to the layers of an onion, that are wrapped around the payload, or message, as shown in Figure 1 (8). Onion routers are distributed throughout the Internet and are tasked with encryption of the connections and acting as proxies (3). These proxies then assemble a virtual circuit route from sender to receiver based on OR response time and bandwidth (7). This process is carried out randomly such that any sort of "sequence prediction" would be implausible (3). Before the message can be sent, the client system must perform a Diffie-Hellman (DH) key exchange with each node in the circuit that will enable decryption of the header information to know which OR is next (7). Subsequently, each hop that the payload makes, "the onion shrinks as [an encryption] layer is peeled off," meaning that by the end of the circuit, the packet will arrive at its destination in plain text, as shown in Figure 2 (8,3,7). A fundamental part of the anonymity provided by onion routing is that when an OR receives a message, it knows only the immediate predecessor and successor (8). Therefore, no single node knows both the original sender and the final recipient.

Following the success of onion routers, the current technology evolved into what we know today as Tor, which stands for The Onion Router. Tor is a “circuit-based low-latency anonymous communication network, supporting [Transmission Control Protocol (TCP)] applications over the internet” (4). The Tor Browser Bundle (TBB) uses the aforementioned onion routing in order to prohibit third parties from viewing or participating in a specific interaction over the Internet (6). Tor was developed in 2004 to address the limitations of onion routing (8). It was “originally designed, implemented, and deployed as a third-generation onion routing project of the U.S. Naval Research Laboratory” (10). The establishment of Tor was primarily intended for government communication protection; however, it expanded to include the military, journalists, non-governmental organizations, and activists (10). There is even a branch of the U.S. Navy that uses the anonymity provided by Tor to expedite open source intelligence gathering (10). Additionally, Tor can potentially thrive in countries where the government attempts to restrict the access of its citizens because it is capable of bypassing such controls (7). Some of the main reasons behind the popularity of Tor originate in the freedom of speech and the traffic encryption that facilitates anonymity (2). Also, Tor is attractive to users because it provides the hidden services, which allow users to provide services via web sites without disclosing their physical location (10). Hidden services “makes use of rendezvous points, which serve as pre-arranged meeting points for service providers to meet up with service users” (7).

Tor enables users to access the Darknet, which contains close to six hundred terabytes (TB) of data and provides access to nearly five hundred times the amount of content than the Internet. The Darknet is inaccessible with your typical browser, such as Internet Explorer, which only allows access to a much smaller portion of the web (2). Furthermore, the Darknet is “a great benefit for cybercriminals and unethical users" (2). One of the more commonly known websites within the Darknet is Silk Road, which is essentially Amazon, except it offers products such as drugs, weapons, and hit-men for hire. Three of the main reasons that criminals use Tor are to facilitate and hide illicit activities such as "hacking, file exchange, and terrorism operation management" (3). The lack of monitoring capabilities due to the usage of Tor makes forensic identification of drug dealers, hired crimes, and peer-to-peer file sharing of child pornography nearly impossible (2).

In order to comprehensively describe how Tor software provides anonymity for its users, it is important to understand the basic components of the interactions taking place. The client, or sender, runs software that is referred to as the onion proxy (OP), which will anonymize the Tor traffic. The server, or recipient, runs web services, which are essentially TCP applications. Lastly, onion routers (OR) relay data from sender to receiver and contain transport layer security (TLS) connections that are used to develop the link encryption between two ORs (4). The data that is being sent is packed into cells, each of which contain a total of 512 bytes of information and is illustrated in Figure 3. All cells contain a header that comprises of 3 unencrypted bytes, leaving 509 bytes for the encrypted payload (4). The relay cells are those used to carry TCP stream data from sender to receiver. Ling and associates defined each of the components of a Tor relay cell after initially stating that there are many types of relay commands, including begin, end, and extend (4). The command field "Recognized" is used to "identify whether the cell is correctly recognized by the client or exit router." Multiple streams can be combined into a single circuit thus demanding the need for the command field "Stream_id," which is used to "identify the specific stream for the corresponding applications at the client or exit router. The command field "Integrity" verifies the integrity of the payload.

Lastly, the command field “Length” indicates the size of the actual data packed into the cell, not including any padding that may be present (4).

Before anonymous communication can commence, the client requires a list of the available Tor nodes, illustrated by Figure 4 (10). Once these are acquired, the OP will randomly select a default quantity of three ORs with which to create a communication circuit (4). Tor uses a random path, illustrated in Figure 5, instead of a direct path to thwart third parties from anticipating the nodes in the circuit (10). After the nodes have been chosen, the client will initiate the procedure of incrementally creating the circuit. Ling et al. also detailed the process of obtaining the encryption keys before anonymized communication can begin (4). As mentioned before with onion routing, the OP will negotiate encryption keys for each of the nodes in the circuit by using the Diffie-Hellman (DH) handshake protocol. With each handshake, a “forward symmetric key kf_1 and a backward symmetric key kb_1” are generated, thus establishing a one-hop circuit (4). These encryption keys would be acquired in the handshake with the first OR. In order to get the same keys for the middle OR, the OP must send an extend relay command to OR1, specifying the address of the next OR in the circuit. Once this is established, OR1 unencrypts the cell with kf_1 and negotiates the encryption keys for OR2 using the DH handshake protocol. OR1 then sends the kf_2 and kb_2 back to the OP. Likewise, OP will then send an extend relay command to OR2 accompanying the address for OR3. Finally, the OP will receive the keys for OR3, consequently creating an anonymous circuit. The only exception here is the connection from OR3 to the recipient is not link encrypted and therefore not protected by TLS within Tor (4).

Therefore, to transmit data from the client to the server, the client’s application, such as the Tor Browser, must first contact the OP, which will be told the destination IP address and port

(4). According to Ling and associates, after the key negotiations take place, the payload will be encrypted as "$\{\{\{Begin < IP, Port >\}_{kf3}\}_{kf2}\}_{kf1}$, where the subscript refers to the key used for encryption of one onion skin" (4). By decrypting each encrypted cell, the overall relay cell becomes thinner, similar to peeling away the layers of an onion as mentioned before with onion routing. When OR3 decrypts the last layer of encryption, it reveals the request to open a TCP stream to a port at the destination IP, which belongs to the server, or recipient (4). To reiterate, one of the main reasons that Tor can provide anonymity is that each node along the way knows only the relay that gave it data and the one that is next in line, thus preventing any one node from knowing the entire circuit (10). Additionally, Tor software only uses a particular circuit for connections that happen within a small time frame of each other. The entire DH handshake protocol will be enforced again to develop a whole new circuit after a given amount of time as seen in Figure 6 (10).

The use of the TBB to navigate the Internet and Darknet clearly requires far more technical knowledge than using a typical browser such as Internet Explorer. Consequently, the routing of traffic through several nodes results in enormous delay (5). Liška and associates found that the factor of round-trip time can increase to anywhere between 2 times and more than 100 times when using Tor, thus concluding that Tor is significantly slower than normal Internet browsers (5). Norcie and colleagues agree, "Tor [is traditionally slower] than a typical Internet connection" (6). This degree of latency could cause a decline in the overall use of Tor, which will undoubtedly harm the quality of anonymity experienced by its users. That is to say, the diverse population of people who use Tor is part of what allows it to provide anonymity because it essentially hides each user among all the others on the network (10). The anonymity experienced by Tor users is a function of 1/n, where n is the number of users (8). Any system like Tor where the degree of anonymity experienced by its users is measured by 1/n, an increase in numbers of users will increase the anonymity of the system (6). The impact of usability on anonymity is one of many weaknesses existing within Tor. Another is the selection of the entry and exit node in the communication circuit because if a third party is able to watch the edges of the circuit, they might be able to “confirm who is communicating due to the low-latency of traffic flowing over the circuit” (8).

The vulnerabilities present in the TBB encouraged a plethora of research projects to attempt penetration and identification of the communicating parties, thus destroying the anonymity of the software. One of the more basic attacks against the anonymity provided by the TBB is traffic analysis. Briefly mentioned before, traffic analysis allows a third-party to eavesdrop on Internet traffic and use statistical techniques to monitor patterns in communication (10). Essentially, the third-party is able to acquire information pertaining to the sender and receiver by watching the data flowing through a particular network, matching the amount of data, and connecting ports that are opening and closing at relatively the same time (8). As discussed previously, Tor encrypts the payload of the data transaction, but leaves the header visible in order for the ORs to know where to send it next. Unfortunately for Tor’s users, the header can provide metadata such as destination, size, and timing that can help a third-party performing traffic analysis to piece together the communication circuit (10). There are, however, several features of Tor that assist in thwarting the threat that traffic analysis presents. Traffic normalization uses a combination of splitting and padding the payload to ensure that all packets are of constant size (7). In addition, the messages can undergo sequence alteration, which prevents basic input to output matching (7).

Similar to traffic analysis, correlation-like attacks allow a third party to reveal which users visit which destination if they control the entry and exit ORs in the circuit (11). The third-party is able to control these routers because Tor does not exploit the use of identity checking mechanisms for their ORs (11). The solution to this vulnerability is to deploy a trust-based OR system, which was proven effective at preventing correlation-like attacks due to its capability to verify the identities of ORs, thus excluding corrupt routers with a high degree of certainty (11). However, it opened the door for another type of attack: inference attacks (11). Inference attackers have knowledge from theoretical deduction of trust relationships between users and ORs, which provides a high probability of guessing the circuit if they can observe the network (11). Luckily, by keeping in mind assumptions of the capabilities of the attackers, Zhou and associates were able to develop a trust degree based algorithm that was determined to be a key feature against inference attacks (11).

Furthermore, protocol-level attacks bring to light a severe deficiency in the anonymity of the TBB. For the encryption and decryption of packets at each OR, Tor uses the Counter Mode of Advanced Encryption Standard (AES-CTR). When an attacker controls multiple onion routers, they need only manipulate one cell via modifying, duplicating, inserting, or deleting cells of the TCP stream to confirm a communication relationship (4). The manipulated cell will interfere with the AES-CTR in a way that will give rise to cell recognition errors, which are unique to protocol-level attacks, thus allowing validation of those participating in the correspondence (4). These attacks are highly effective as they demand control of only one cell and do not rely on traffic timing to be successful (4). Protocol-level attacks significantly degrade the quality of anonymity provided by Tor and, unfortunately for its users, remain incapable of being defended (4).

Most of the aforementioned methods of attack describe ways in which to exploit the vulnerabilities present within the onion routing process. Another way to circumvent the anonymity provided by Tor is by attacking the Tor browser instead of Tor network. Due to the criminal activities enabled by online anonymity and access to Silk Road, Tor is a high priority of the NSA (9). The first step in catching these criminals is identifying them as Tor users. The NSA maintains partnerships with US Telecom firms that help them to monitor large sections of the Internet. “Fingerprints” are created that “detect http requests from the Tor network to particular servers” (9). These are then loaded into XKeyscore, which is an analysis tool that enables visualization of mostly all internet activity of the target (9). Once an individual is identified, secret servers are used to redirect the user to another secret server, FoxAcid, which will infect the user’s computer with a FoxAcid tag that seemingly appears harmless to the user (9). Additional attacks are performed after the initial infection to ensure long-term eavesdropping (9). This type of attack gives the NSA full control over the user’s computer, “including access to files, all keystrokes and all online activity” (1). EgotisticalGiraffe, the technique used to attack Tor users through software vulnerabilities, involves exploiting the TBB through its version of the Firefox web browser (1). The vulnerabilities within Firefox that may be exploited in an attack may be associated with Flash or Javascript (9).

Despite statements that the Tor Browser Bundle provides impermeable anonymity, there are several types of attacks that suggest otherwise. From a forensic perspective, information stored in the random-access memory and in the registry will provide substantial evidence of Tor use and browsing history. Dayalamurthy suggested that “retrieving the memory dump from the suspect machine and analyzing it forensically could provide more forensic evidence of [TBB] usage” (2). Therefore, research ensued to forensically analyze a Random-Access Memory

(RAM) dump in addition to analyzing the registry and capturing the packets from Internet and Darknet browsing in order to prove that relevant evidence can be obtained. To test this theory, several virtual machines were created with identical parameters in order to monitor these key aspects in hopes of discovering evidence of use or installation of the Tor Browser Bundle. The results of this study will hopefully be of significant importance to the forensic science community in that it will provide information for digital analysts in the event that they come across a suspect allegedly participating in illicit activities using the TBB.

Materials and Methods

An NCS Gemini (CK3-A368) desktop computer running a 32-bit version of the Windows 7 operating system with 8 gigabytes (GB) of RAM was used as the host machine for several virtual machines. VMware® Workstation version 10.0.2 build 1744117 was installed onto the host machine before creating four primary VMs. The parameters of these VMs are detailed in Table 1 and the descriptions of their functions with respect to this project are shown in Table 2. A fifth VM was generated in the event that one of the VMs crashed or data was irretrievably lost. Windows Pre-Tor was constructed by following the prompts within the VMware® Workstation, which resulted in a 64-bit version of the Windows 7 operating system. Once it was powered on, Windows was activated and necessary security updates were installed. The VM was then connected to Marshall University's virtual private network (VPN) so that Symantec™ Endpoint Protection version 11.0.5002.333 could be installed. Additionally, Adobe® Reader® XI version 11.0.07 was installed on Windows Pre-Tor. Once this VM underwent Windows updates and installation of the necessary software, it was cloned four times. When cloning an already created virtual machine, VMware® Workstation gives the option of either a Full Clone or a Linked Clone. In this case, all four subsequent virtual machines were created as Full Clones of

Windows Pre-Tor, meaning that they were complete copies of the original VM in its current state but did not require access to the original VM to function.

Following the preparation of the VMs, several analysis programs were downloaded before data collection could begin. First, Tor Browser Bundle version 3.6.1 was downloaded and installed on all of the essential VMs except for Windows Pre-Tor. Next, WireShark® version 1.10.7 was installed on the host machine to enable the capturing of packets within the virtual environments. Finally, Forensic ToolKit® (FTK) Imager Lite version 3.1.1 by AccessData® was installed and saved on a WD 1311B 320GB external hard drive (HD). Once this software was acquired, the next step was to devise a browsing protocol, simulating a typical user interacting with the browser, using both the Internet and the Darknet, which can be seen in detail in Figures 7 and 8, respectively.

After these preliminary steps were accomplished, data collection from Windows Pre-Tor was able to commence. To start, Windows Pre-Tor was powered on while WireShark® was opened on the host machine and directed to capture packets from VMware® Network Adapter VMnet8. Internet Explorer (IE) version 11.0.9600.17107 was used to execute the Internet browsing protocol while WireShark® captured the packets that were being sent and received. After the browsing was completed, WireShark® was commanded to cease the packet capture and IE was kept open on the last visited page. Then, FTK® Imager Lite was opened within the VM and was used to seize volatile information, which is known as a RAM dump. During the RAM capture process, FTK Imager Lite also collected the pagefile. This created an AD1 file, which is proprietary file format for AccessData. Those two files were then added as custom content sources to generate a custom content image to later be analyzed in Forensic ToolKit®. Subsequently, the image of that VM was uploaded into FTK® Imager Lite and the following registry files were exported: NTUSER.DAT, SAM, SECURITY, SYSTEM, and SOFTWARE, thus concluding data acquisition for Windows Pre-Tor.

In Windows Tor Download, neither Internet nor Darknet browsing protocol were carried out, therefore not requiring packet capture. However, FTK® Imager Lite was still used to capture the memory and export the aforementioned registry files in order to determine if any traces of downloading the TBB were being stored in either of those locations.

Similarly to the process carried out with Windows Pre-Tor, Windows Tor Active was powered on in its virtual environment simultaneously with WireShark® on the host machine. Likewise, WireShark® was directed to capture packets from VMware® Network Adapter VMnet8. However, instead of using IE, TBB was used for both Internet and Darknet browsing while the packet capture was proceeding. As expected, browsing within the Tor Browser can show latency. Accordingly, the fifth .onion website visited from the Darknet browsing protocol failed to load. The browser was then directed to the sixth and final .onion website and did not attempt the fifth one again. Before closing TBB after the browsing protocol had concluded, FTK® Imager Lite was again used to capture the memory and extract the existing registry files from the image of Windows Tor Active.

Lastly, Windows Post-Tor underwent a similar procedure including using WireShark® to capture the packets, using TBB to perform both the Internet and Darknet browsing protocols, and FTK® Imager Lite to gather the RAM dump and the necessary registry files. This concluded the initial round of data collection. However, upon further consideration, it was decided to create another VM, which would be used to solely track any registry changes that occurred during the installation and uninstallation of the TBB.

The parameters and description pertaining to Windows Registry can also be seen in Tables 1 and 2, respectively. There were two different programs that were used to capture the registry at different stages of TBB installation. The first was Process Monitor version 3.1, which gathered the changes in the registry files in real time as the TBB was installed and uninstalled. The second was Regshot version 1.9.0.0 that took snapshots of the registry at three different key points: before TBB was installed, after TBB was installed, and after TBB was uninstalled.

A Dell Optiplex 960 desktop computer running a 64-bit version of the Windows 7 operating system with 8GB of RAM was used to analyze the data collected throughout the course of this research. First, Forensic Toolkit® version 5.4.0.37 was used to data carve the AD1 files created using FTK® Imager Lite for each of the four original virtual machines. Then Registry Viewer® version 1.7.4.2 by AccessData® was used to navigate through the registry files collected for each of the VMs. WireShark® version 1.10.8 was used to interpret the packets captured from Windows Pre-Tor and Windows Tor Active to differentiate between those from Internet Explorer and Tor. Subsequently, NetworkMiner version 1.5 was used to parse the packets captured from WireShark® into a more user-friendly format. Analysis of the fifth VM, Windows Registry, was performed with Process Monitor version 3.1. This program allowed the visualization of the changes in the registry during installation and uninstallation of the TBB. Lastly, RegShot version 1.9.0.0 was used to compare two sets of snapshots of the registry. One comparison took place between the registries before and after the TBB was installed and the second between the registries after the TBB was installed and after it was subsequently uninstalled.

Results

Subsequent to analyzing the carved RAM dump data in FTK® from each of the virtual machines, it was evident that the use of the TBB did leave behind some traces of its existence. As a control, Table 3 shows the evidence of carrying out the Internet browsing protocol, seen in Figure 7, with Internet Explorer in Windows Pre-Tor. There were several indications, including multiple file types, each presenting some form of proof of the websites visited. Despite not performing any online browsing in Windows Tor Download, there was still a possibility of finding evidence of the presence of Tor. By performing an index search of "Tor" in FTK®, multiple hits were found within the memory dump and pagefiles that can be seen in Tables 4 and 5, respectively. Not all results are displayed, but several of the hits with more evidentiary value were selected. Contrary to the belief of anonymity while using Tor, Table 6 shows several files obtained during analysis of the RAM dump from Windows Tor Active that indicate the use of the TBB. Again, an index search of "Tor" was performed and quite a few hits resulted, however, not all are displayed. Tables 7 and 8 each show a couple of the more pertinent results from the memory dump and pagefiles, respectively. The same data carve analysis was performed on the Windows Post-Tor virtual machine and, unfortunately, reaped fewer results, as can be seen in Table 9. Similarly, an index search of "Tor" was performed and only a few results from the memory dump and pagefiles can be seen in Tables 10 and 11, respectively.

Monitoring and analyzing the registry files throughout the course of this research with Registry Viewer®, RegShot, and Process Monitor provided a few pieces of evidence that indicated the presence of Tor. The information provided by Registry Viewer® for Windows Pre-Tor uncovered more relevant artifacts than the other VMs and can be seen in Table 12. Those same file paths were attempted in each of the subsequent VMs, with no successful information being discovered. There were, however, additional pieces of evidence found. Table 13 shows evidence that Tor was installed and saved to the desktop in Windows Tor Download. It was expected that the registry information found in Windows Tor Active and Windows Post-Tor would differ, however, Table 14 proves otherwise. The same information was found in both registries.

The types and quantity of changes made when Tor was installed and then uninstalled as captured and presented by RegShot are exemplified in Table 15. It was evident that there were more changes made to the registry when Tor was installed than when it was uninstalled. Some of the changes made during the installation were undone when Tor was uninstalled, however, some of them remained. These alterations to the registry can be seen in Tables 16 and 17.

The Process Monitor tracked changes to the registry in real time during the installation of the TBB. Table 18 shows several of the Tor related registry changes that occurred during the installation process. There were many system and software changes under the process name of "Start Tor Browser." The registry keys of these alterations culminated in language configuration, control panel, and desktop settings. There were also many process names of "firefox.exe" that included system and software changes as well. Some of these local machine registry keys included code identifiers and session managers, to name a couple. Additionally, there were process names of "tor.exe" that also contained system and software changes. Most notable of these registry keys ended with Microsoft Strong Cryptographic Provider and FipsAlgorithmPolicy that may be changes that contribute to the encryption within the onion routing. Process Monitor was not able to pick up any information pertaining to Tor after it was uninstalled.

WireShark® captured the packets during browsing with both Internet Explorer and Tor. The typed URLs found within the packet capture from Internet Explorer are shown in Table 19. The packets captured from browsing within Tor did not generate any URLs from neither the Internet nor the Darknet. There exists a notable distinction between the Protocol Hierarchy Statistics from Windows Pre-Tor and Windows Tor Active and can be observed in Figures 9 and 10, respectively. Some of the obvious differences include the presence of line-based text data and JPEG file interchange format in Pre-Tor packets and the absence of these in Tor Active packets. There also tends to be a lot more data under HTTP in Pre-Tor than in Tor Active. Generally speaking, the traffic from Tor does not resemble typical network traffic. Furthermore, there are quite a few more IP addresses and destinations within the traffic from Internet Explorer than in Tor. By analyzing the different IP addresses and how frequently each one of them is used, it is assumed that the IP address of the entry node within the Tor circuit can be found. For example, the IP address of 217.114.213.19 was used in 26.17% of the Tor traffic and was found to be from a server in Germany. Additionally, the overall appearance of the packet streams between Internet Explorer and Tor are extremely different and can be seen in Figures 11 and 12, respectively. Lastly, NetworkMiner organized the packets imported from WireShark® into a more user-friendly format and the differences between the packet data are exemplified in Table 20.

Discussion and Conclusions

Given the results of the research, it would appear that Tor is not as anonymous as it advertises. However, most of the evidence acquired through the course of this research would otherwise be unattainable in dead-box forensic investigations. For example, the RAM dumps were performed in a live atmosphere, meaning the web browser was still open immediately after carrying out the specified browsing protocols. The computer would have to be in the custody of law enforcement before it was shut down for a RAM dump to generate any useful data, which is not always possible. In any case, Windows Tor Active mostly showed the end of the browsing protocol, which may have been due to the virtual machines only having 2GB of RAM meaning some of the earlier parts of browsing may have been overwritten. The packet capture is an area of research that is a wealth of information within network forensics. This type of evidence collection may also be applicable if the analysts were tasked with remotely hacking suspect computers similar to the work done by the National Security Agency. Furthermore, the analyst would have to be able to recognize the appearance of normal traffic within the Protocol Hierarchy Statistics and packet stream to know whether Tor was being used. In addition, the identification of the entry node using the IP addresses found in the Tor packet stream may be able to further help identify the communication circuit since the exit node is typically unencrypted. Lastly, the most common artifacts found in dead-box forensics is the information pertaining to changes in the registry. RegShot indicated that the uninstallation of Tor was not complete. The text file generated by this program provides paths to each of the altered files and folders, which could then be followed through Registry Viewer® in an attempt to find the specific value that was modified.

Overall, these methods have limited uses in dead-box forensics but much more so in network forensics. The initial hypothesis of being able to acquire evidence of illicit activities from the RAM dump, registry files, and packet capture was proven. However, the goal of providing dead-box forensic investigators with an effective method to gather potentially relevant evidence of Tor usage was not as successful. The techniques involving packet capture and RAM dumping require access to the information on the device either during or immediately after the crime was committed. Nonetheless, the methods used to analyze the Registry are applicable to dead-box forensics but only to determine the presence, not the browsing activity, of Tor. The file modifications gathered from RegShot may be of use to forensic investigators as they can follow those specified paths to see if those values were changed, thus indicating Tor usage.

Additional questions were raised as a result of this research. A patch for WireShark® called Tor Dissector will, theoretically, decrypt the Tor traffic. It would be interesting to see the resulting traffic if this patch could be applied to the previously obtained packet streams. Additionally, websites in the browsing protocol throughout this research were only viewed, meaning no files or images were actively downloaded and saved to the computer by the user. A future direction of this work may be to enhance the browsing protocol into a more interactive manner. Actively saving files from the Internet and Darknet may be attainable through similar procedures. Naturally, the digital forensic community will remain persistent in their quest to refine an applicable technique that will adequately gather potentially relevant evidence from a hard drive subsequent to collection. However, without having the abundant resources at the disposal of the NSA, digital analysts will be hard pressed to find a reliable method of breaking through the anonymity provided by the Tor Browser Bundle.

References

(1) Ball J, Schneier B, Greenwald G. NSA and GCHQ target Tor network that protects anonymity of web users, 2013; http://www.theguardian.com/world/2013/oct/04/nsa-gchq-attack-tor-network-encryption (accessed June 5, 2014).

(2) Dayalamurthy D. Forensic Memory Dump Analysis and Recovery of the Artefacts of Using Tor Bundle Browser - The Need. Proceedings of the 11th Australian Digital Forensics Conference 2013 December 2-4;Edith Cowan University, Perth, Western Australia.

(3) Forte D. Advances in Onion Routing: Description and backtracing/investigation problems. Digit Invest 2006;3(2):85-8.

(4) Ling Z, Luo J, Yu W, Fu X, Jia W, Zhao W. Protocol-level attacks against Tor. Comput Netw 2013;57(4):869-86.

(5) Liška T, Sochor T, Sochorová H. Comparison between normal and Tor-Anonymized Web Client Traffic. Procedia Comput Sci 2011;3:888-92.

(6) Norcie G, Blythe J, Caine K, Camp L.J. Why Johnny Can't Blow the Whistle: Identify and Reducing Usability Issues in Anonymity Systems. Proceedings of the NDSS Workshop on Usability Security 2014 February 23;San Diego, CA, USA.

(7) Owen M. Fun with onion routing. Netw Secur 2007;(4):8-12.

(8) Ren J, Wu J. Survey on anonymous communications in computer networks. Comput Commun 2010;33(4):420-31.

(9) Schneier B. Attacking Tor: how the NSA targets users' online anonymity. 2013; http://www.theguardian.com/world/2013/oct/04/tor-attacks-nsa-users-online-anonymity (accessed June 5, 2014).

(10) Tor. Project: Overview. https://www.torproject.org/about/overview.html.en (accessed June 21, 2014).

(11) Zhou P, Luo X, Chang R.K.C. Interference attacks against trust-based onion routing: Trust degree to the rescue. Comput & Secur 2013;39(B):431-46.

Acknowledgements

The author thanks Dr. Terry Fenger, Dr. Pamela Staton, Joshua Brunty, and Ian Levstein from Marshall University Forensic Science Center for their assistance throughout the course of this research. Additionally, the author thanks Corporal Robert Boggs from the West Virginia State Police Digital Forensics Unit as well as John Sammons from the Department of Integrated Science and Technology at Marshall University for their instruction and expertise. This project was supported by Award No. 2010-IJ-CX-K025 awarded by the National Institute of Justice, Office of Justice Programs, U.S. Department of Justice. The opinions, findings, and conclusions or recommendations expressed in this publication/program/exhibition are those of the author(s) and do not necessarily reflect the views of the Department of Justice.

Appendix

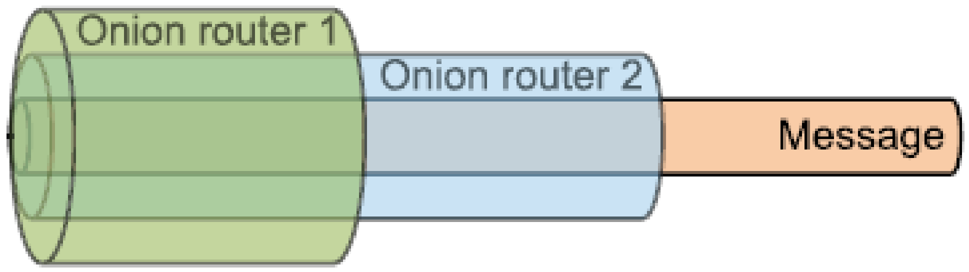


Figure 1. Layers of encrypted onion routers around the payload (8)

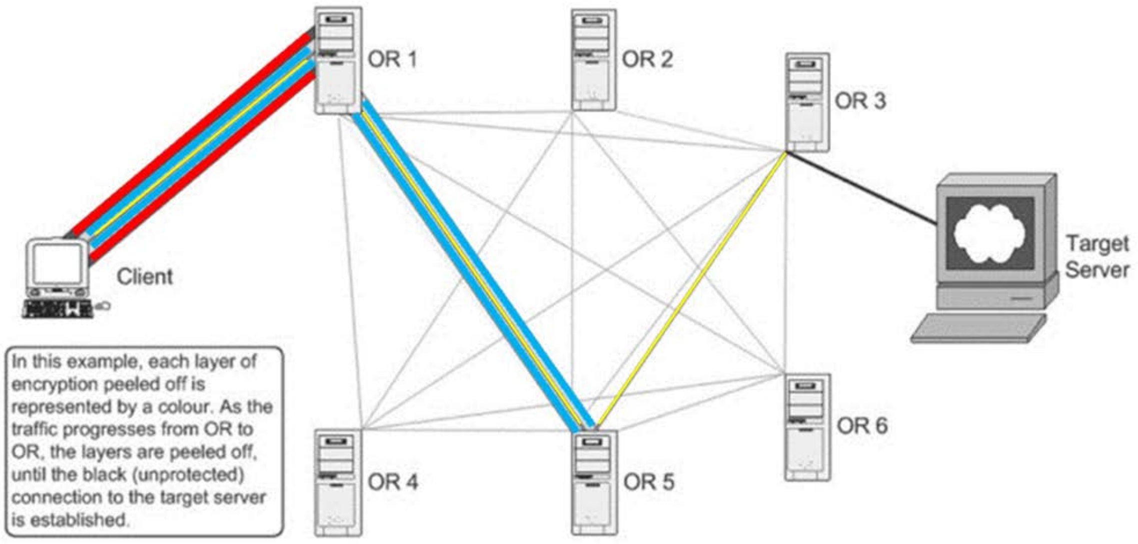

OR 1
OR 2
OR 3
Target Server
Client
OR 6
OR 4
OR 5
In this example, each layer of encryption peeled off is represented by a colour. As the traffic progresses from OR to OR, the layers are peeled off, until the black (unprotected) connection to the target server is established.

Figure 2. Peeling of the layers of encryption as the payload arrives at the target server (7)

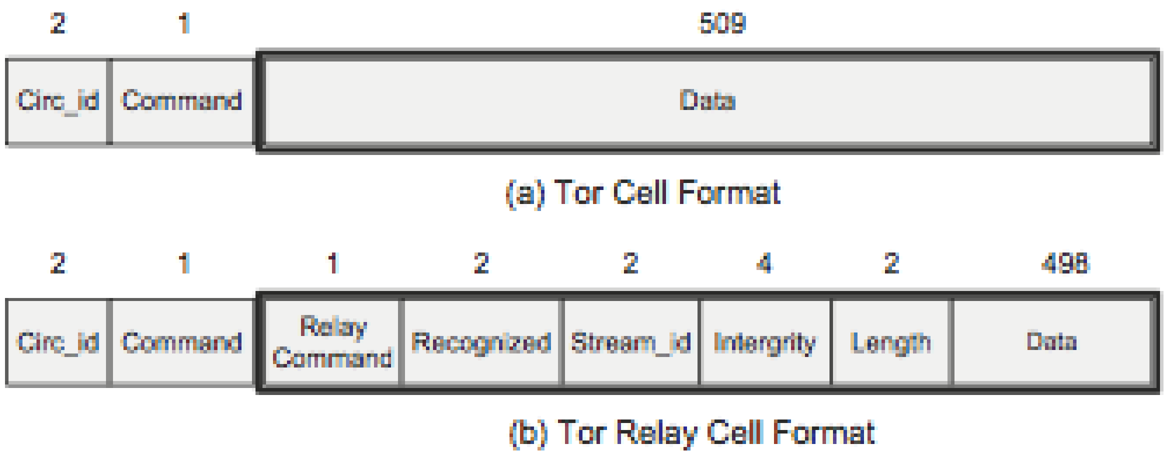


Figure 3. Tor Cell and Tor Relay Cell Format; Numbers above fields indicate amount of bytes (4)

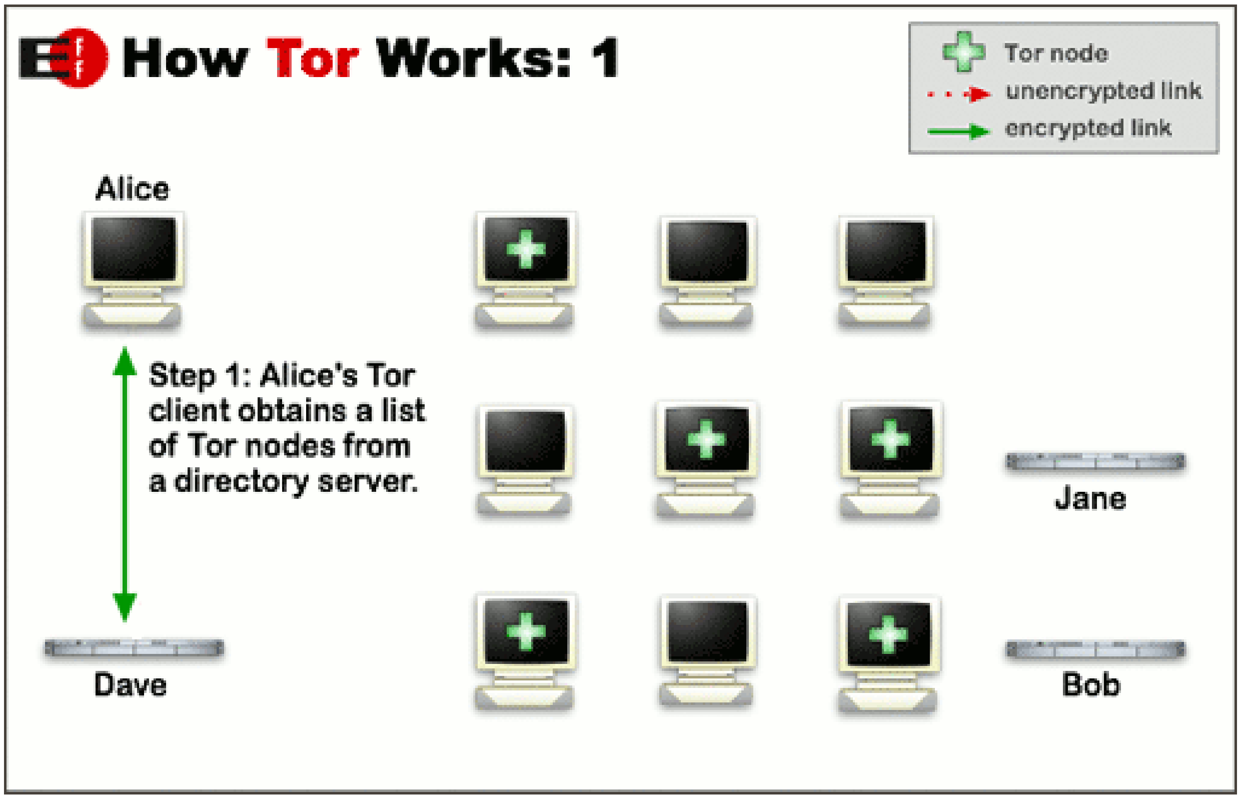

How Tor Works: 1
Tor node
unencrypted link
encrypted link
Alice
Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.
Jane
Dave
Bob

Figure 4. Client obtaining a list of Tor nodes (10)

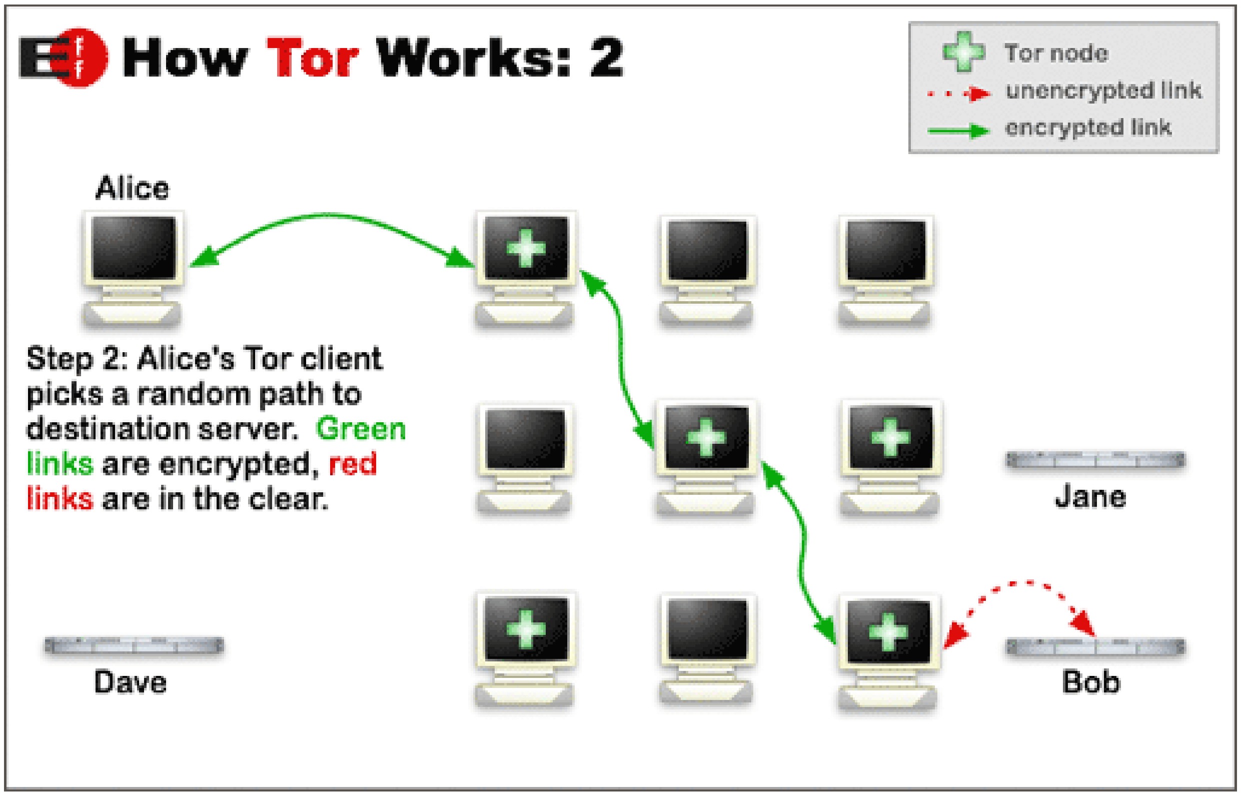

How Tor Works: 2
Tor node
unencrypted link
encrypted link
Alice
Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.
Jane
Dave
Bob

Figure 5. Random encrypted path through the Tor nodes (10)

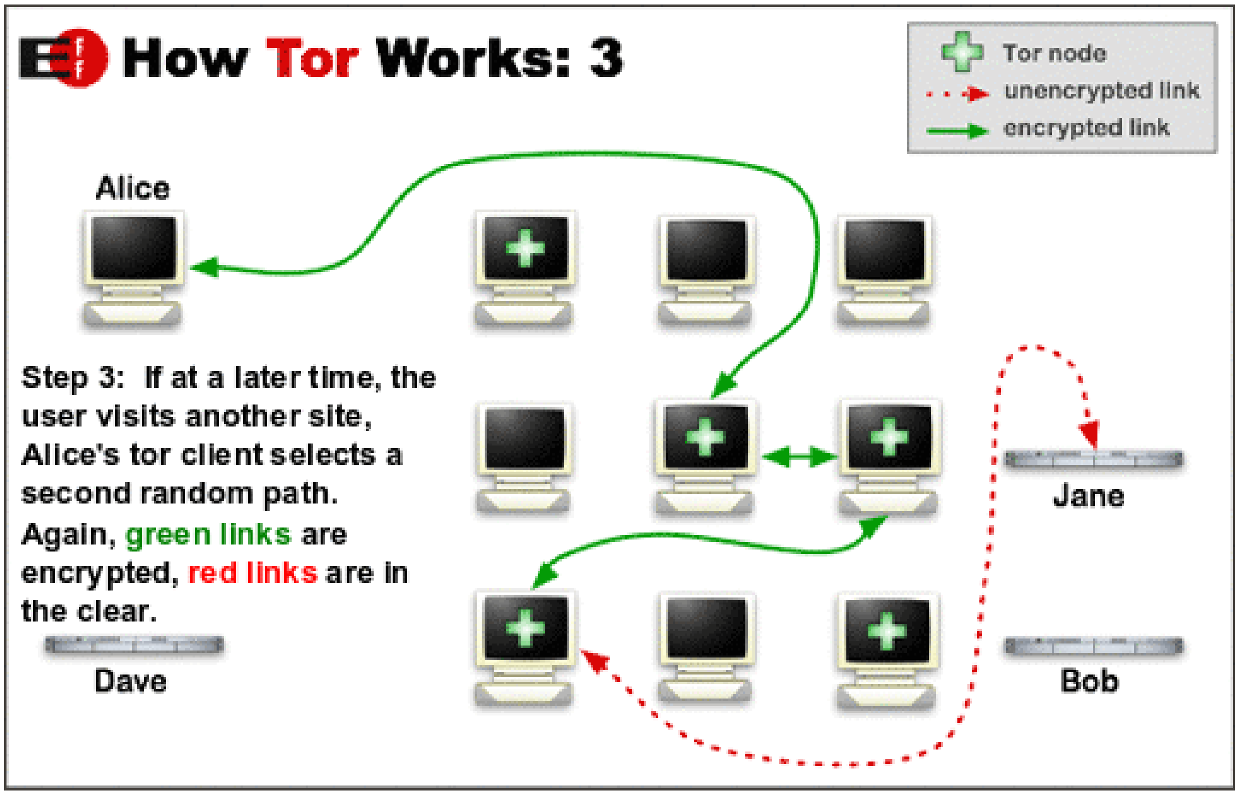
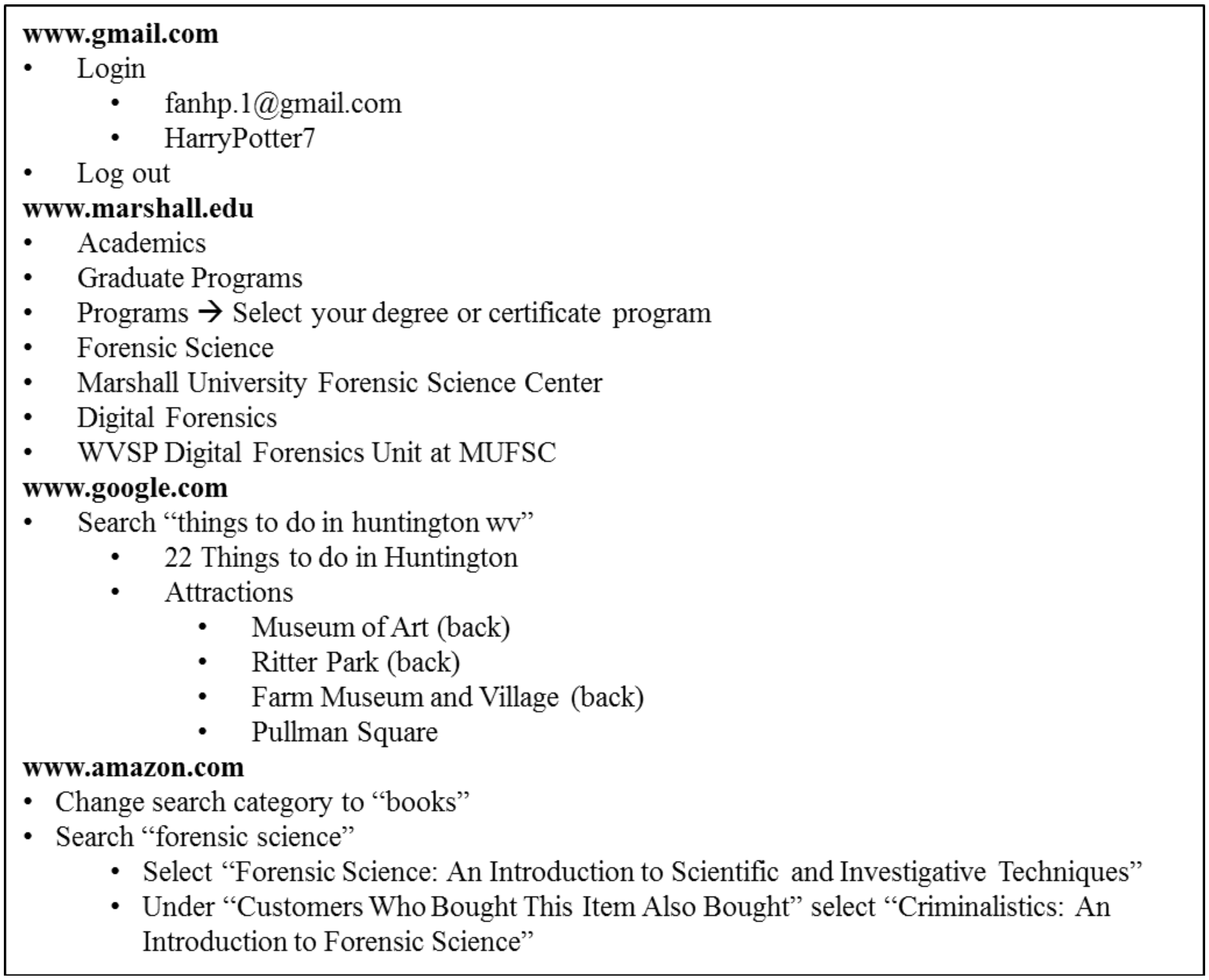

How Tor Works: 3
Tor node
unencrypted link
encrypted link
Alice
Step 3: If at a later time, the user visits another site, Alice's tor client selects a second random path. Again, green links are encrypted, red links are in the clear.
Dave
Jane
Bob

Figure 6. Creation of a new anonymous circuit (10)

Table 1. VM Parameters	
Memory	2 GB
Processors	2
Hard Disk (SCSI)	60 GB
CD/DVD (SATA)	Auto Detect
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto Detect
Printer	Present
Display	Auto Detect

Table 2. VM Descriptions	
Windows Pre-Tor	Windows updates installed and connected to Marshall VPN; does NOT have Tor Browser Bundle installed; all browsing performed in Internet Explorer
Windows Tor Download	Windows updates installed and connected to Marshall VPN; Tor Browser Bundle is installed on this machine, however, will not be used to navigate the Internet or Darknet
Windows Tor Active	Windows updates installed and connected to Marshall VPN; Tor Browser Bundle is installed and will be used to navigate both the Internet and Darknet
Windows Post-Tor	Windows updates installed and connected to Marshall VPN; Tor Browser Bundle was installed initially and underwent the same Internet and Darknet browsing as the "Windows Tor Active" VM but was then uninstalled
Windows Registry	Windows updates installed and connected to Marshall VPN; used to monitor changes in the registry as Tor Browser Bundle was installed and then uninstalled



www.gmail.com

- Login
    - fanhp.1@gmail.com
    - HarryPotter7
- Log out

www.marshall.edu

- Academics
- Graduate Programs
- Programs → Select your degree or certificate program
- Forensic Science
- Marshall University Forensic Science Center
- Digital Forensics
- WVSP Digital Forensics Unit at MUFSC

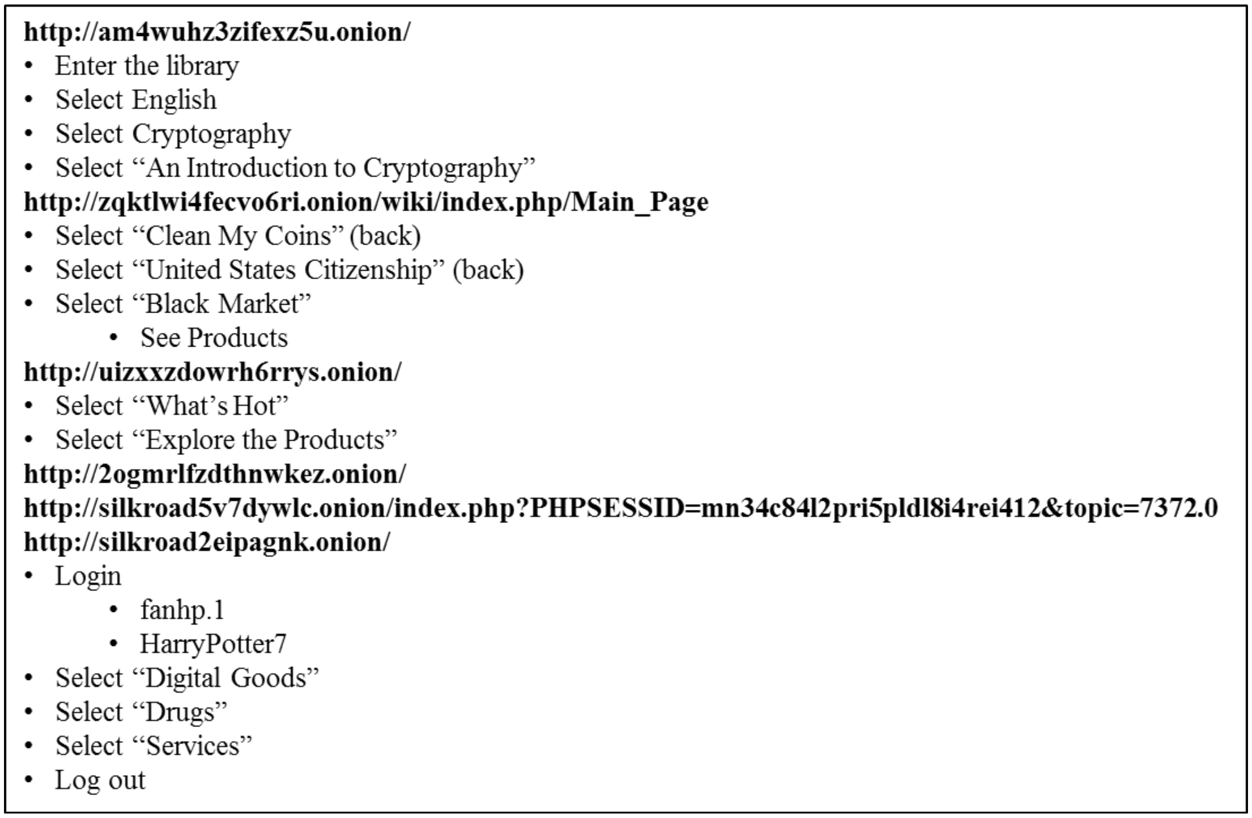
www.google.com

- Search “things to do in huntington wv”
    - 22 Things to do in Huntington
    - Attractions
        - Museum of Art (back)
        - Ritter Park (back)
        - Farm Museum and Village (back)
        - Pullman Square

www.amazon.com

- Change search category to “books”
- Search “forensic science”
    - Select “Forensic Science: An Introduction to Scientific and Investigative Techniques”
    - Under “Customers Who Bought This Item Also Bought” select “Criminalistics: An Introduction to Forensic Science”

Figure 7. Internet Browsing Protocol



http://am4wuhz3zifexz5u.onion/
- Enter the library
- Select English
- Select Cryptography
- Select "An Introduction to Cryptography"

http://zqktlwi4fecvo6ri.onion/wiki/index.php/Main_Page
- Select "Clean My Coins" (back)
- Select "United States Citizenship" (back)
- Select "Black Market"
    - See Products

http://uizxxzdowrh6rrys.onion/
- Select "What's Hot"
- Select "Explore the Products"

http://2ogmrlfzdthnwkez.onion/
http://silkroad5v7dywlc.onion/index.php?PHPSESSID=mn34c84l2pri5pldl8i4rei412&topic=7372.0
http://silkroad2eipagnk.onion/
- Login
    - fanhp.1
    - HarryPotter7
- Select "Digital Goods"
- Select "Drugs"
- Select "Services"
- Log out

Figure 8. Darknet Browsing Protocol

Table 3. Windows Pre-Tor RAM Dump Data

Carved File	File Type	Evidence
1742724031	html	Keywords content from Marshall University Forensic Science Center
1999010751	html	Keywords content from Marshall University Forensic Science Center
282857993	html	Customer reviews from Amazon
300174271	html	Digital Forensics Graduate Program Emphasis & Certificate
1048027504	jpeg	WVSP Digital Forensics Lab
1274565160	jpeg	WVSP Digital Forensics Lab
219619824	jpeg	WVSP Digital Forensics Lab
244188438	jpeg	WVSP Digital Forensics Lab
302308856	jpeg	WVSP Digital Forensics Lab
308277800	jpeg	WVSP Digital Forensics Lab
395759984	jpeg	WVSP Digital Forensics Lab
415955584	jpeg	Forensic Science Book from Amazon
424508032	jpeg	Forensic Science Book from Amazon
7488104	jpeg	Criminalistics Book from Amazon
156696576	ole	URLs for MUFSC and FS graduate program
272224400	ole	"things to do in huntington wv" Google search
360423424	ole	URLs for MUFSC and FS graduate program
416923696	ole	WVSP ICAC Task Force
874856448	ole	URLs for MUFSC and FS graduate program
285597936	png	"Free Two-Day Shipping for College Students" from Amazon
307990152	png	Google

Table 4. Windows Tor Download Index Search - Memory Dump Evidence (2102 hits)
DFU-RESEARCH\DESKTOP\TOR BROWSER\DATA\BROWSER\PROF
DFU-Research\Desktop\Tor Browser\Tor\PluggableTran
\Desktop\Tor Browser\Tor\PluggableTransports\w9xpo
\Desktop\Tor Browser\Tor\PluggableTransports\flash
DFU-Research\Desktop\Tor Browser\Docs\Licenses\Tor

Table 5. Windows Tor Download Index Search - Pagefile Evidence (542 hits)
dfu-research\desktop\tor browser\browser \users\d
dfu-research\desktop\tor browser\browser\firefox.e
r browser.exe start tor browser.exe tor browser
DFU-Research\Desktop\Tor Browser\Start Tor Browser
@torproject.org.xpi tor-launcher@torproject.org.x

Table 6. Windows Tor Active RAM Dump Data

Carved File	File Type	Evidence
30843	html	Tor Browser Bundle for Windows Download
34320024	html	Index of/Library/English/Cryptography/
39555	html	Tor homepage
113135960	jpeg	WVSP Digital Forensics Lab
114992848	jpeg	WVSP Digital Forensics Lab
116777840	jpeg	WVSP Digital Forensics Lab
138543864	jpeg	Criminalistics Book from Amazon
1998049408	jpeg	Apple iPad from .onion site
2019227440	jpeg	YouTube from Silk Road
2024888964	jpeg	Instagram from Silk Road
2055586912	jpeg	Criminalistics Book from Amazon
2140398250	jpeg	Drugs from Silk Road
23107458	jpeg	Drugs from Silk Road
539082736	jpeg	Tor Onion image
8924056	jpeg	Drugs from Silk Road
2043916784	png	Apple iPhone from .onion site
2061738472	png	Apple iPad from .onion site

Table 7. Windows Tor Active Index Search - Memory Dump Evidence (18753 hits)
dfu-research\desktop\tor browser\browser\firefox.e
dfu-research\desktop\tor browser\tor\tor.exe

Table 8. Windows Tor Active Index Search - Pagefile Evidence (1010 hits)
torToSecurityDescrip tor CreateFileMapping
torToSecurityDescrip tor defAttr get_Sources Me

Table 9. Windows Post-Tor RAM Dump Data		
Carved File	**File Type**	**Evidence**
587228031	html	Tor homepage
510492752	jpeg	Tor Onion image
587254076	jpeg	Tor Orbot for Android Devices
587262788	jpeg	Tor Tails image
587214612	png	Tor Download image
134657420	lnk	Shortcut File: C:\Users\DFU-Research\Desktop\Tor Browser\Browser\firefox.exe
245874724	lnk	Shortcut File: C:\Users\DFU-Research\Desktop\Tor Browser\Browser\firefox.exe

Table 10. Windows Post-Tor Index Search - Memory Dump Evidence (4657 hits)
\DESKTOP\TOR BROWSER\TOR\PLUGGABLETRANSPORTS\FLASH
DFU-RESEARCH\DESKTOP\TOR BROWSER\DATA\BROWSER\PROF
DFU-RESEARCH\DESKTOP\TOR BROWSER\DATA\TOR \WINDOW
DFU-RESEARCH\DESKTOP\TOR BROWSER\DATA\TOR\CACHED-M

Table 11. Windows Post-Tor Index Search - Pagefile Evidence (1085 hits)
torToSecurityDescrip tor CreateFileMapping MapVi
torToSecurityDescrip tor defAttr get_Sources Me
securityDescr tor="binary base64:AQAAgBQAAA

Table 12. Windows Pre-Tor Registry	
SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths	Executable file for Internet Explorer
SOFTWARE\Wow6432Node\Microsoft\Internet Explorer	Installed applications
SOFTWARE\Clients\StartMenuInternet	Installed web browsers
NTUSER.DAT\Software\Microsoft\Internet Explorer\TypedURLs	Typed URLs within Internet Explorer

Table 13. Windows Tor Download Registry	
NTUSER.DAT\Software\Microsoft\Windows\Shell\Bags\1\Desktop	Tor Browser

Table 14. Windows Tor Active and Post-Tor Registries	
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count	C:\Users\DFU-Research\Desktop\Tor Browser\Start Tor Browser.exe
NTUSER.DAT\Software\Microsoft\Windows\Shell\Bags\1\Desktop	Tor Browser

Table 15. RegShot – Changes in Registries with the Installation/Uninstallation of Tor

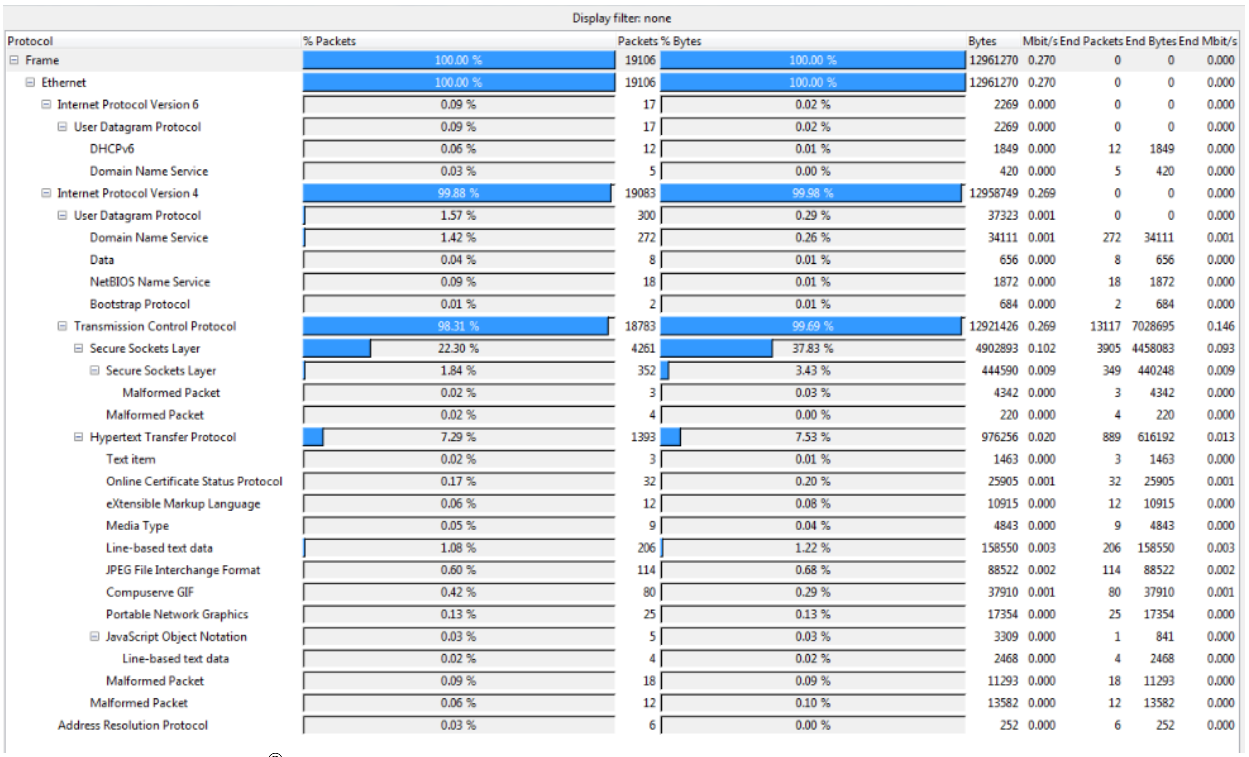
Action	Installation of Tor	Uninstallation of Tor
Keys Deleted	97	--
Keys Added	57	9
Values Deleted	173	--
Values Added	495	13
Values Modified	219	7
Files Added	566	--
Files Deleted	149	278
Files [Attributes?] Modified	57	10
Folders Added	153	--
Folders Deleted	3	74
Total Changes	1969	391

Table 16. RegShot Comparison of Pre-Tor to Tor Download (Files/Folders Added)
C:\Users\DFU-Research\AppData\Local\Microsoft\Windows\Temporary Internet Files\Low\Content.IE5\FQZE7OQ6\icon-TorBrowser[1].jpg
C:\Users\DFU-Research\AppData\Local\Microsoft\Windows\Temporary Internet Files\Low\Content.IE5\CXM5RLJI\icon-TorStatus[1].jpg
C:\Users\DFU-Research\AppData\Local\Microsoft\Windows\Temporary Internet Files\Low\Content.IE5\O2MO2E81\tor-logo[1].jpg
C:\Users\DFU-Research\Desktop\Tor Browser\Browser\firefox.exe
C:\Users\DFU-Research\Desktop\Tor Browser\Docs\Licenses\Tor-Launcher.txt
C:\Users\DFU-Research\Desktop\Tor Browser\Start Tor Browser.exe
C:\Users\DFU-Research\Desktop\Tor Broswer\Data\Browser\Caches

Table 17. RegShot Comparison of Tor Download to Post-Tor (Files/Folders Deleted)
C:\Users\DFU-Research\Desktop\Tor Browser
C:\Users\DFU-Research\Desktop\Tor Browser\Data\Browser\Caches
C:\Users\DFU-Research\Desktop\Tor Browser\Browser\firefox.exe
C:\Users\DFU-Research\Desktop\Tor Browser\Docs\Licenses\Tor-Launcher.txt

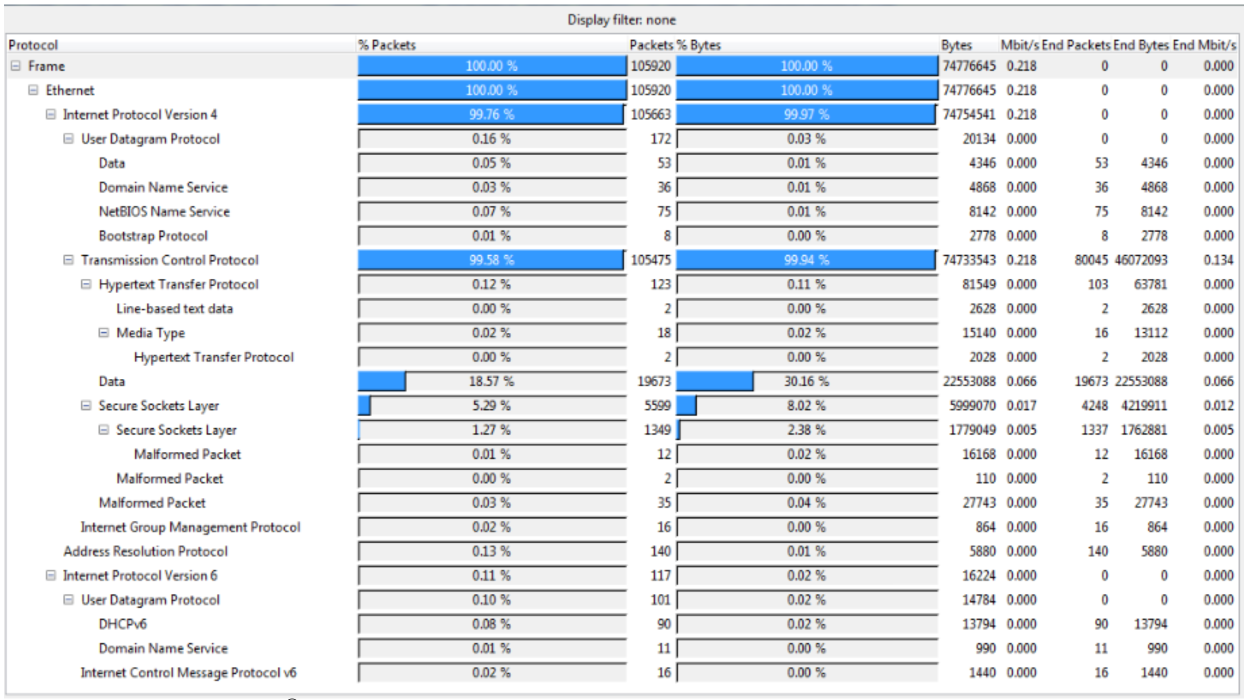
Table 18. Process Monitoring of the Registry During Tor Installation
HKLM\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Custom\Start Tor Browser.exe
HKLM\Software\Microsoft\windows NT\CurrentVersion\AppCompatFlags\Layers\C:\Users\DFU-Research\Desktop\TorBrowser\Start Tor Browser.exe
HKLM\Software\Microsoft\Windows\CurrentVersion\App Paths\Start Tor Browser.exe
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Start Tor Browser.exe
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Custom\Start Tor Browser.exe
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Compatibility32\Start Tor Browser
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers\C:\Users\DFU-Research\Desktop\Tor Browser\Browser\firefox.exe
HKLM\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Custom\firefox.exe
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\App Paths\tor.exe
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\tor.exe
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\tor.exe
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers\C:\Users\DFU-Research\Desktop\TorBrowser\Tor\tor.exe
HKLM\Software\Wow6432Node\Mozilla\Firefox\TaskBarIDs
HKLM\Software\Mozilla\Firefox\TaskBarIDs

Table 19. WireShark® Packet URLs for Windows Pre-Tor
mail.google.com
www.marshall.edu
www.google.com
www.tripadvisor.com
www.amazon.com


Display filter: none

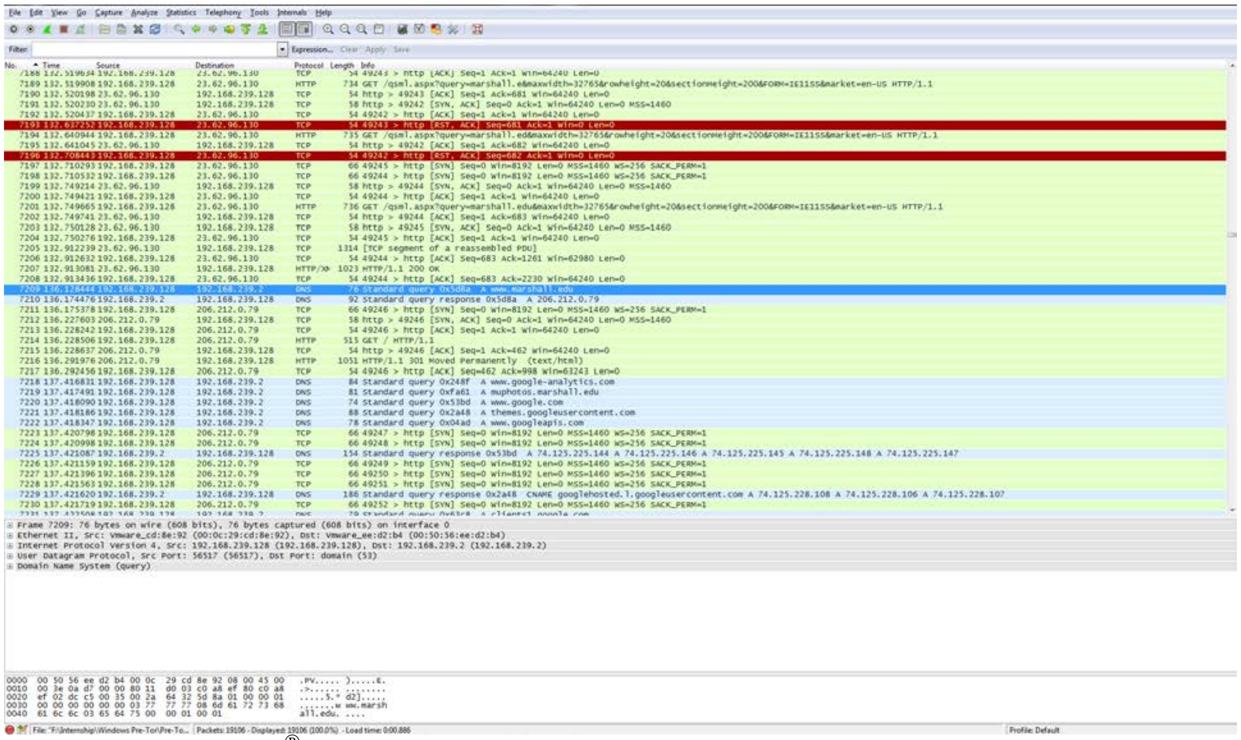
Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
⊟ Frame	100.00 %	19106	100.00 %	12961270	0.270	0	0	0.000
⊟ Ethernet	100.00 %	19106	100.00 %	12961270	0.270	0	0	0.000
⊟ Internet Protocol Version 6	0.09 %	17	0.02 %	2269	0.000	0	0	0.000
⊟ User Datagram Protocol	0.09 %	17	0.02 %	2269	0.000	0	0	0.000
DHCPv6	0.06 %	12	0.01 %	1849	0.000	12	1849	0.000
Domain Name Service	0.03 %	5	0.00 %	420	0.000	5	420	0.000
⊟ Internet Protocol Version 4	99.88 %	19083	99.98 %	12958749	0.269	0	0	0.000
⊟ User Datagram Protocol	1.57 %	300	0.29 %	37323	0.001	0	0	0.000
Domain Name Service	1.42 %	272	0.26 %	34111	0.001	272	34111	0.001
Data	0.04 %	8	0.01 %	656	0.000	8	656	0.000
NetBIOS Name Service	0.09 %	18	0.01 %	1872	0.000	18	1872	0.000
Bootstrap Protocol	0.01 %	2	0.01 %	684	0.000	2	684	0.000
⊟ Transmission Control Protocol	98.31 %	18783	99.69 %	12921426	0.269	13117	7028695	0.146
⊟ Secure Sockets Layer	22.30 %	4261	37.83 %	4902893	0.102	3905	4458083	0.093
⊟ Secure Sockets Layer	1.84 %	352	3.43 %	444590	0.009	349	440248	0.009
Malformed Packet	0.02 %	3	0.03 %	4342	0.000	3	4342	0.000
Malformed Packet	0.02 %	4	0.00 %	220	0.000	4	220	0.000
⊟ Hypertext Transfer Protocol	7.29 %	1393	7.53 %	976256	0.020	889	616192	0.013
Text item	0.02 %	3	0.01 %	1463	0.000	3	1463	0.000
Online Certificate Status Protocol	0.17 %	32	0.20 %	25905	0.001	32	25905	0.001
eXtensible Markup Language	0.06 %	12	0.08 %	10915	0.000	12	10915	0.000
Media Type	0.05 %	9	0.04 %	4843	0.000	9	4843	0.000
Line-based text data	1.08 %	206	1.22 %	158550	0.003	206	158550	0.003
JPEG File Interchange Format	0.60 %	114	0.68 %	88522	0.002	114	88522	0.002
Compuserve GIF	0.42 %	80	0.29 %	37910	0.001	80	37910	0.001
Portable Network Graphics	0.13 %	25	0.13 %	17354	0.000	25	17354	0.000
⊟ JavaScript Object Notation	0.03 %	5	0.03 %	3309	0.000	1	841	0.000
Line-based text data	0.02 %	4	0.02 %	2468	0.000	4	2468	0.000
Malformed Packet	0.09 %	18	0.09 %	11293	0.000	18	11293	0.000
Malformed Packet	0.06 %	12	0.10 %	13582	0.000	12	13582	0.000
Address Resolution Protocol	0.03 %	6	0.00 %	252	0.000	6	252	0.000

Figure 9. WireShark® Protocol Hierarchy Statistics from Windows Pre-Tor


Display filter: none

Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
⊟ Frame	100.00 %	105920	100.00 %	74776645	0.218	0	0	0.000
⊟ Ethernet	100.00 %	105920	100.00 %	74776645	0.218	0	0	0.000
⊟ Internet Protocol Version 4	99.76 %	105663	99.97 %	74754541	0.218	0	0	0.000
⊟ User Datagram Protocol	0.16 %	172	0.03 %	20134	0.000	0	0	0.000
Data	0.05 %	53	0.01 %	4346	0.000	53	4346	0.000
Domain Name Service	0.03 %	36	0.01 %	4868	0.000	36	4868	0.000
NetBIOS Name Service	0.07 %	75	0.01 %	8142	0.000	75	8142	0.000
Bootstrap Protocol	0.01 %	8	0.00 %	2778	0.000	8	2778	0.000
⊟ Transmission Control Protocol	99.58 %	105475	99.94 %	74733543	0.218	80045	46072093	0.134
⊟ Hypertext Transfer Protocol	0.12 %	123	0.11 %	81549	0.000	103	63781	0.000
Line-based text data	0.00 %	2	0.00 %	2628	0.000	2	2628	0.000
⊟ Media Type	0.02 %	18	0.02 %	15140	0.000	16	13112	0.000
Hypertext Transfer Protocol	0.00 %	2	0.00 %	2028	0.000	2	2028	0.000
Data	18.57 %	19673	30.16 %	22553088	0.066	19673	22553088	0.066
⊟ Secure Sockets Layer	5.29 %	5599	8.02 %	5999070	0.017	4248	4219911	0.012
⊟ Secure Sockets Layer	1.27 %	1349	2.38 %	1779049	0.005	1337	1762881	0.005
Malformed Packet	0.01 %	12	0.02 %	16168	0.000	12	16168	0.000
Malformed Packet	0.00 %	2	0.00 %	110	0.000	2	110	0.000
Malformed Packet	0.03 %	35	0.04 %	27743	0.000	35	27743	0.000
Internet Group Management Protocol	0.02 %	16	0.00 %	864	0.000	16	864	0.000
Address Resolution Protocol	0.13 %	140	0.01 %	5880	0.000	140	5880	0.000
⊟ Internet Protocol Version 6	0.11 %	117	0.02 %	16224	0.000	0	0	0.000
⊟ User Datagram Protocol	0.10 %	101	0.02 %	14784	0.000	0	0	0.000
DHCPv6	0.08 %	90	0.02 %	13794	0.000	90	13794	0.000
Domain Name Service	0.01 %	11	0.00 %	990	0.000	11	990	0.000
Internet Control Message Protocol v6	0.02 %	16	0.00 %	1440	0.000	16	1440	0.000

Figure 10. WireShark® Protocol Hierarchy Statistics from Windows Tor Active



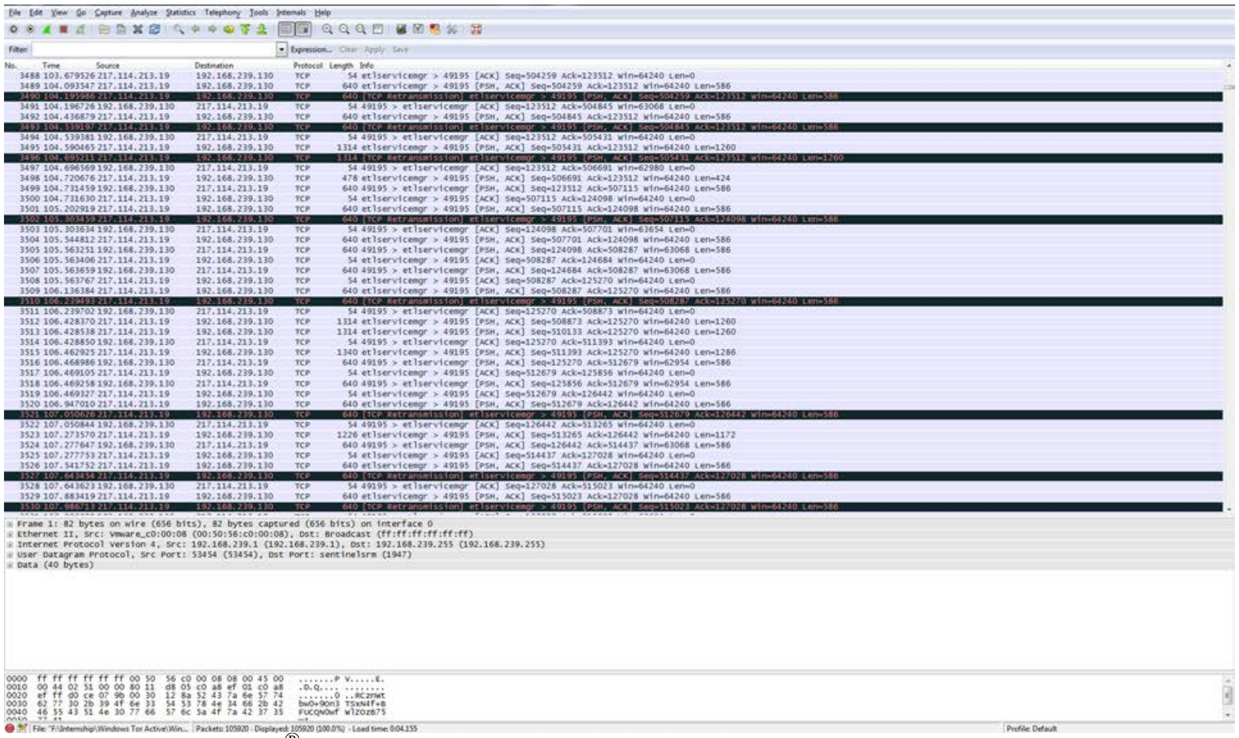
```
No.   Time       Source          Destination     Protocol Length Info
7188 132.519634 192.168.239.128 23.62.96.130    TCP      54 49243 > http [ACK] Seq=1 Ack=1 Win=64240 Len=0
7189 132.519908 192.168.239.128 23.62.96.130    HTTP    734 GET /qsml.aspx?query=marshall.e&maxwidth=32765&rowheight=20&sectionHeight=200&FORM=IE11SS&market=en-US HTTP/1.1
7190 132.520198 23.62.96.130    192.168.239.128 TCP      54 http > 49243 [ACK] Seq=1 Ack=681 Win=64240 Len=0
7191 132.520230 23.62.96.130    192.168.239.128 TCP      58 http > 49242 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
7192 132.520437 192.168.239.128 23.62.96.130    TCP      54 49242 > http [ACK] Seq=1 Ack=1 Win=64240 Len=0
7193 132.637252 192.168.239.128 23.62.96.130    TCP      54 49243 > http [RST, ACK] Seq=681 Ack=1 Win=0 Len=0
7194 132.640944 192.168.239.128 23.62.96.130    HTTP    735 GET /qsml.aspx?query=marshall.ed&maxwidth=32765&rowheight=20&sectionHeight=200&FORM=IE11SS&market=en-US HTTP/1.1
7195 132.641045 23.62.96.130    192.168.239.128 TCP      54 http > 49242 [ACK] Seq=1 Ack=682 Win=64240 Len=0
7196 132.708443 192.168.239.128 23.62.96.130    TCP      54 49242 > http [RST, ACK] Seq=682 Ack=1 Win=0 Len=0
7197 132.710293 192.168.239.128 23.62.96.130    TCP      66 49245 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7198 132.710532 192.168.239.128 23.62.96.130    TCP      66 49244 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7199 132.749214 23.62.96.130    192.168.239.128 TCP      58 http > 49244 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
7200 132.749421 192.168.239.128 23.62.96.130    TCP      54 49244 > http [ACK] Seq=1 Ack=1 Win=64240 Len=0
7201 132.749665 192.168.239.128 23.62.96.130    HTTP    736 GET /qsml.aspx?query=marshall.edu&maxwidth=32765&rowheight=20&sectionHeight=200&FORM=IE11SS&market=en-US HTTP/1.1
7202 132.749741 23.62.96.130    192.168.239.128 TCP      54 http > 49244 [ACK] Seq=1 Ack=683 Win=64240 Len=0
7203 132.750128 23.62.96.130    192.168.239.128 TCP      58 http > 49245 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
7204 132.750276 192.168.239.128 23.62.96.130    TCP      54 49245 > http [ACK] Seq=1 Ack=1 Win=64240 Len=0
7205 132.912239 23.62.96.130    192.168.239.128 TCP    1314 [TCP segment of a reassembled PDU]
7206 132.912632 192.168.239.128 23.62.96.130    TCP      54 49244 > http [ACK] Seq=683 Ack=1261 Win=62980 Len=0
7207 132.913081 23.62.96.130    192.168.239.128 HTTP/X 1023 HTTP/1.1 200 OK
7208 132.913436 192.168.239.128 23.62.96.130    TCP      54 49244 > http [ACK] Seq=683 Ack=2230 Win=64240 Len=0
7209 136.128444 192.168.239.128 192.168.239.2   DNS      76 Standard query 0x5d8a  A www.marshall.edu
7210 136.174476 192.168.239.2   192.168.239.128 DNS      92 Standard query response 0x5d8a  A 206.212.0.79
7211 136.175378 192.168.239.128 206.212.0.79    TCP      66 49246 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7212 136.227603 206.212.0.79    192.168.239.128 TCP      58 http > 49246 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
7213 136.228242 192.168.239.128 206.212.0.79    TCP      54 49246 > http [ACK] Seq=1 Ack=1 Win=64240 Len=0
7214 136.228506 192.168.239.128 206.212.0.79    HTTP    515 GET / HTTP/1.1
7215 136.228637 206.212.0.79    192.168.239.128 TCP      54 http > 49246 [ACK] Seq=1 Ack=462 Win=64240 Len=0
7216 136.291976 206.212.0.79    192.168.239.128 HTTP   1051 HTTP/1.1 301 Moved Permanently  (text/html)
7217 136.292456 192.168.239.128 206.212.0.79    TCP      54 49246 > http [ACK] Seq=462 Ack=998 Win=63243 Len=0
7218 137.416831 192.168.239.128 192.168.239.2   DNS      84 Standard query 0x248f  A www.google-analytics.com
7219 137.417491 192.168.239.128 192.168.239.2   DNS      81 Standard query 0xfa61  A muphotos.marshall.edu
7220 137.418090 192.168.239.128 192.168.239.2   DNS      74 Standard query 0x53bd  A www.google.com
7221 137.418186 192.168.239.128 192.168.239.2   DNS      88 Standard query 0x2a48  A themes.googleusercontent.com
7222 137.418347 192.168.239.128 192.168.239.2   DNS      78 Standard query 0x04ad  A www.googleapis.com
7223 137.420798 192.168.239.128 206.212.0.79    TCP      66 49247 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7224 137.420998 192.168.239.128 206.212.0.79    TCP      66 49248 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7225 137.421087 192.168.239.2   192.168.239.128 DNS     154 Standard query response 0x53bd  A 74.125.225.144 A 74.125.225.146 A 74.125.225.145 A 74.125.225.148 A 74.125.225.147
7226 137.421159 192.168.239.128 206.212.0.79    TCP      66 49249 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7227 137.421396 192.168.239.128 206.212.0.79    TCP      66 49250 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7228 137.421563 192.168.239.128 206.212.0.79    TCP      66 49251 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7229 137.421620 192.168.239.2   192.168.239.128 DNS     186 Standard query response 0x2a48  CNAME googlehosted.l.googleusercontent.com A 74.125.228.108 A 74.125.228.106 A 74.125.228.107
7230 137.421719 192.168.239.128 206.212.0.79    TCP      66 49252 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1

⊞ Frame 7209: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
⊞ Ethernet II, Src: Vmware_cd:8e:92 (00:0c:29:cd:8e:92), Dst: Vmware_ee:d2:b4 (00:50:56:ee:d2:b4)
⊞ Internet Protocol Version 4, Src: 192.168.239.128 (192.168.239.128), Dst: 192.168.239.2 (192.168.239.2)
⊞ User Datagram Protocol, Src Port: 56517 (56517), Dst Port: domain (53)
⊞ Domain Name System (query)

0000  00 50 56 ee d2 b4 00 0c  29 cd 8e 92 08 00 45 00   .PV..... ).....E.
0010  00 3e 0a d7 00 00 80 11  d0 03 c0 a8 ef 80 c0 a8   .>...... ........
0020  ef 02 dc c5 00 35 00 2a  64 32 5d 8a 01 00 00 01   .....5.* d2].....
0030  00 00 00 00 00 00 03 77  77 77 08 6d 61 72 73 68   .......w ww.marsh
0040  61 6c 6c 03 65 64 75 00  00 01 00 01              all.edu. ....

File: "F:\Internship\Windows Pre-Tor\Pre-To...  Packets: 19106 - Displayed: 19106 (100.0%) - Load time: 0:00.886   Profile: Default
```

Figure 11. WireShark® packets for Windows Pre-Tor



```
No.   Time       Source          Destination     Protocol Length Info
3488 103.679526 217.114.213.19  192.168.239.130 TCP      54 etlservicemgr > 49195 [ACK] Seq=504259 Ack=123512 Win=64240 Len=0
3489 104.093547 217.114.213.19  192.168.239.130 TCP     640 etlservicemgr > 49195 [PSH, ACK] Seq=504259 Ack=123512 Win=64240 Len=586
3490 104.195986 217.114.213.19  192.168.239.130 TCP     640 [TCP Retransmission] etlservicemgr > 49195 [PSH, ACK] Seq=504259 Ack=123512 Win=64240 Len=586
3491 104.196726 192.168.239.130 217.114.213.19  TCP      54 49195 > etlservicemgr [ACK] Seq=123512 Ack=504845 Win=63068 Len=0
3492 104.436879 217.114.213.19  192.168.239.130 TCP     640 etlservicemgr > 49195 [PSH, ACK] Seq=504845 Ack=123512 Win=64240 Len=586
3493 104.539197 217.114.213.19  192.168.239.130 TCP     640 [TCP Retransmission] etlservicemgr > 49195 [PSH, ACK] Seq=504845 Ack=123512 Win=64240 Len=586
3494 104.539381 192.168.239.130 217.114.213.19  TCP      54 49195 > etlservicemgr [ACK] Seq=123512 Ack=505431 Win=64240 Len=0
3495 104.590465 217.114.213.19  192.168.239.130 TCP    1314 etlservicemgr > 49195 [PSH, ACK] Seq=505431 Ack=123512 Win=64240 Len=1260
3496 104.695211 217.114.213.19  192.168.239.130 TCP    1314 [TCP Retransmission] etlservicemgr > 49195 [PSH, ACK] Seq=505431 Ack=123512 Win=64240 Len=1260
3497 104.696569 192.168.239.130 217.114.213.19  TCP      54 49195 > etlservicemgr [ACK] Seq=123512 Ack=506691 Win=62980 Len=0
3498 104.720676 217.114.213.19  192.168.239.130 TCP     478 etlservicemgr > 49195 [PSH, ACK] Seq=506691 Ack=123512 Win=64240 Len=424
3499 104.731459 192.168.239.130 217.114.213.19  TCP     640 49195 > etlservicemgr [PSH, ACK] Seq=123512 Ack=507115 Win=64240 Len=586
3500 104.731630 217.114.213.19  192.168.239.130 TCP      54 etlservicemgr > 49195 [ACK] Seq=507115 Ack=124098 Win=64240 Len=0
3501 105.202919 217.114.213.19  192.168.239.130 TCP     640 etlservicemgr > 49195 [PSH, ACK] Seq=507115 Ack=124098 Win=64240 Len=586
3502 105.303459 217.114.213.19  192.168.239.130 TCP     640 [TCP Retransmission] etlservicemgr > 49195 [PSH, ACK] Seq=507115 Ack=124098 Win=64240 Len=586
3503 105.303634 192.168.239.130 217.114.213.19  TCP      54 49195 > etlservicemgr [ACK] Seq=124098 Ack=507701 Win=63654 Len=0
3504 105.544812 217.114.213.19  192.168.239.130 TCP     640 etlservicemgr > 49195 [PSH, ACK] Seq=507701 Ack=124098 Win=64240 Len=586
3505 105.563251 192.168.239.130 217.114.213.19  TCP     640 49195 > etlservicemgr [PSH, ACK] Seq=124098 Ack=508287 Win=63068 Len=586
3506 105.563406 217.114.213.19  192.168.239.130 TCP      54 etlservicemgr > 49195 [ACK] Seq=508287 Ack=124684 Win=64240 Len=0
3507 105.563659 192.168.239.130 217.114.213.19  TCP     640 49195 > etlservicemgr [PSH, ACK] Seq=124684 Ack=508287 Win=63068 Len=586
3508 105.563767 217.114.213.19  192.168.239.130 TCP      54 etlservicemgr > 49195 [ACK] Seq=508287 Ack=125270 Win=64240 Len=0
3509 106.136384 217.114.213.19  192.168.239.130 TCP     640 etlservicemgr > 49195 [PSH, ACK] Seq=508287 Ack=125270 Win=64240 Len=586
3510 106.239493 217.114.213.19  192.168.239.130 TCP     640 [TCP Retransmission] etlservicemgr > 49195 [PSH, ACK] Seq=508287 Ack=125270 Win=64240 Len=586
3511 106.239702 192.168.239.130 217.114.213.19  TCP      54 49195 > etlservicemgr [ACK] Seq=125270 Ack=508873 Win=64240 Len=0
3512 106.428370 217.114.213.19  192.168.239.130 TCP    1314 etlservicemgr > 49195 [PSH, ACK] Seq=508873 Ack=125270 Win=64240 Len=1260
3513 106.428538 217.114.213.19  192.168.239.130 TCP    1314 etlservicemgr > 49195 [PSH, ACK] Seq=510133 Ack=125270 Win=64240 Len=1260
3514 106.428850 192.168.239.130 217.114.213.19  TCP      54 49195 > etlservicemgr [ACK] Seq=125270 Ack=511393 Win=64240 Len=0
3515 106.462925 217.114.213.19  192.168.239.130 TCP    1340 etlservicemgr > 49195 [PSH, ACK] Seq=511393 Ack=125270 Win=64240 Len=1286
3516 106.468986 192.168.239.130 217.114.213.19  TCP     640 49195 > etlservicemgr [PSH, ACK] Seq=125270 Ack=512679 Win=62954 Len=586
3517 106.469105 217.114.213.19  192.168.239.130 TCP      54 etlservicemgr > 49195 [ACK] Seq=512679 Ack=125856 Win=64240 Len=0
3518 106.469258 192.168.239.130 217.114.213.19  TCP     640 49195 > etlservicemgr [PSH, ACK] Seq=125856 Ack=512679 Win=62954 Len=586
3519 106.469327 217.114.213.19  192.168.239.130 TCP      54 etlservicemgr > 49195 [ACK] Seq=512679 Ack=126442 Win=64240 Len=0
3520 106.947010 217.114.213.19  192.168.239.130 TCP     640 etlservicemgr > 49195 [PSH, ACK] Seq=512679 Ack=126442 Win=64240 Len=586
3521 107.050626 217.114.213.19  192.168.239.130 TCP     640 [TCP Retransmission] etlservicemgr > 49195 [PSH, ACK] Seq=512679 Ack=126442 Win=64240 Len=586
3522 107.050844 192.168.239.130 217.114.213.19  TCP      54 49195 > etlservicemgr [ACK] Seq=126442 Ack=513265 Win=64240 Len=0
3523 107.273570 217.114.213.19  192.168.239.130 TCP    1226 etlservicemgr > 49195 [PSH, ACK] Seq=513265 Ack=126442 Win=64240 Len=1172
3524 107.277647 192.168.239.130 217.114.213.19  TCP     640 49195 > etlservicemgr [PSH, ACK] Seq=126442 Ack=514437 Win=63068 Len=586
3525 107.277753 217.114.213.19  192.168.239.130 TCP      54 etlservicemgr > 49195 [ACK] Seq=514437 Ack=127028 Win=64240 Len=0
3526 107.541752 217.114.213.19  192.168.239.130 TCP     640 etlservicemgr > 49195 [PSH, ACK] Seq=514437 Ack=127028 Win=64240 Len=586
3527 107.643454 217.114.213.19  192.168.239.130 TCP     640 [TCP Retransmission] etlservicemgr > 49195 [PSH, ACK] Seq=514437 Ack=127028 Win=64240 Len=586
3528 107.643623 192.168.239.130 217.114.213.19  TCP      54 49195 > etlservicemgr [ACK] Seq=127028 Ack=515023 Win=64240 Len=0
3529 107.883419 217.114.213.19  192.168.239.130 TCP     640 etlservicemgr > 49195 [PSH, ACK] Seq=515023 Ack=127028 Win=64240 Len=586
3530 107.986713 217.114.213.19  192.168.239.130 TCP     640 [TCP Retransmission] etlservicemgr > 49195 [PSH, ACK] Seq=515023 Ack=127028 Win=64240 Len=586

⊞ Frame 1: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
⊞ Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol Version 4, Src: 192.168.239.1 (192.168.239.1), Dst: 192.168.239.255 (192.168.239.255)
⊞ User Datagram Protocol, Src Port: 53454 (53454), Dst Port: sentinelsrm (1947)
⊞ Data (40 bytes)

0000  ff ff ff ff ff ff 00 50  56 c0 00 08 08 00 45 00   .......P V.....E.
0010  00 44 02 51 00 00 80 11  d8 05 c0 a8 ef 01 c0 a8   .D.Q.... ........
0020  ef ff d0 ce 07 9b 00 30  12 8a 52 43 7a 6e 57 74   .......0 ..RCznWt
0030  62 77 30 2b 39 4f 6e 33  54 53 78 4e 34 66 2b 42   bw0+9On3 TSxN4f+B
0040  46 55 43 51 4e 30 77 66  57 6c 5a 4f 7a 42 37 35   FUCQN0wf WlZOzB75

File: "F:\Internship\Windows Tor Active\Win...  Packets: 105920 - Displayed: 105920 (100.0%) - Load time: 0:04.155   Profile: Default
```

Figure 12. WireShark® packets for Windows Tor Active

Table 20. NetworkMiner comparison between Pre-Tor and Tor Active

Category	Pre-Tor	Tor Active
Hosts	253	39
Frames	19xxx	10xxx
Files	722	60
Images	224	0
Messages	0	0
Credentials	112	0
Sessions	377	19
DNS	636	72
Parameters	9234	201
Keywords	0	0
Cleartext	0	0
Anomalies	0	0