International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.803

Anatomizing Relay Disintegration and Hop Operations of Tor

Ujas Dhami[1*], Nisarg Shah[2]

[1, 2]Department of Computer Engineering, Silver Oak University, Ahmedabad - 380061, India
[*]Corresponding Author E-mail: ujasdhami[at]gmail.com

Abstract: ***The Tor Project is a prominent asset to privacy and security, metastasizing among commercials, and political leaders. The project has a designated browser, to route source requests to a generous consigned number of proxies, before reaching the header destination. Investigation: This paper is an in-depth observation made by setting up a Node Switch into the Tor Network, using a denominated Virtual Private Server (VPS), to capture, analyze, and compare Relay traffic to the Entry Node traffic to determine the entropy, encryption topology, and heuristics of the network. Method: This examination deduces the security, seclusion, and pliability of the network concerning the anonymity of browsing by using the methods of Static and Dynamic analysis. Principle Result: The Entry Node transmitted unencrypted domain names with IP addresses over the network, with the relay, encrypting the transmission suite for further propagation using TLSv1.2 encryption keys with a substantial amount of exchanges.***

Keywords: Onion Network, Tor Browser, Tor Nodes, Online Privacy and Security, Anonymous Surfing

1. Introduction

Tor is the web browser made for providing anonymity over the Internet. Internet is the connection of multiple computers connected, which interchange data with one another. The Tor Project decided to utilize this methodology, to create the Tor Browser, for protecting the privacy and integrity of the internet.

1.1 Purpose

The Internet is divided into many parts, some of which include the Surface web, the Dark web, and the Deep web. Surface web entails what a person searches over the Internet, that is, all the sites ending with .com, .in, .org, etcetera. It includes all the things you can access through web browsers, like Chrome, Mozilla, Fox, and Opera-mini. Most internet users access private databases such as email accounts and credit card accounts daily. These pages are not indexed over search engines, whereas in the case of the surface web, it has been indexed by the search engines, so we can easily access it.

The dark web consists of 90% of the content spread over the Internet and is used by Corporations, NGOs, and Government Agencies[1]. The deep web consists of 5% on the entire Internet. The deep web is the subset of the dark web. The deep and dark webs can be used for legal and illegal purposes, and both can be accessed through a special browser known as Tor Browser. In general, most users prevailing over the Internet would never need to access content on the dark web, hence, it is safe for private organizations to undergo covert communications by the use of Tor [1]. The Tor browser is made for anonymous browsing and to overcome the problem of eavesdropping.

Initially, Tor was built by the US Navy to protect the sensitive and confidential data of the US Government. Still, however, Tor continues its use done by the government, and slowly and gradually, it became the Open Source Web Browser which supports multiple platforms available to the public.

1.2 Modus Operandi

Tor Browser works on the principle of onion routing, which is a peer-to-peer network topology connection[2]. It enables anonymity of the Internet over the World Wide Web. It uses multiple layers of security and encryption, which encrypts its headers, i.e., its source and destination IP address, whereas its data, which is being sent over the network. It has been built so that no one can monitor users' activity online. Tor browser has three types of nodes through which the data is transmitted and received. The nodes are the Entry node, Middle node/Relay, and the Exit node.

Once a user installs Tor, the browser uses Tor servers to send information to the associated Exit node, pinpointed from which the data leaves the network. Once this information has been shipped, it's encrypted multiple times before being sent to the successive node. Continuation of this method makes it troublesome to trace the data back to the first supply [2]. Additionally, the Tor browser doesn't track browsing history or store cookies.

1.3 Node Details

This node established on our VPS was to act with hybridization, as an Entry node or a Relay, depending on network convenience[3]. The node was established solely to eavesdrop packets and inspect them under both, a static analysis, and dynamic or real-time analysis.

The node was put under observation for around 14 days, with packet-capturing done every day, to determine the

[1]https://www.imf.org/external/pubs/ft/fandd/2019/09/the-truth-about-the-dark-web-kumar.htm

[2]In peer to peer architecture, every node is connected to other nodes directly. Every computer node is referred to as a peer.

[3]https://community.torproject.org/relay/setup/guard

change in entropy of the traffic. Observations were cataloged with each day.

2. Literature Review [3][4][5]

Title: Selfrando: Securing the Tor Browser against De-Anonymization Exploits

A De-Anonymization[4] mitigation is presented about the Selfrando - which is a load time randomization technique that proves to be efficient, enhanced, and practical, which defends the browser against malicious attacks, exploits, etcetera, with one example of the one which FBI reported to be used against the Tor. The above-stated solution which is mentioned in the research claims to improve security over standard address space layout randomization (ASLR)[5] techniques currently used by Firefox and other browsers. Tor is said to be the mirroring of the Firefox browser. Moreover, the organization collaborated with the Tor Project to ensure that Selfrando is fully compatible with Address Sanitizer (ASan), a compiler featured to detect memory corruption. ASan is used in a hardened version of the Tor Browser, made for test purposes. The Tor Project decided to include these solutions in the hardened releases of the Tor Browser, which are currently undergoing field testing.

Title: A Forensic Audit of the Tor Browser Bundle

The increasing use of encrypted knowledge among file storage and in network communications leaves investigators with several challenges[6]. One of the foremost difficulties is the Tor protocol, as its main focus is to guard the privacy of the user, in each of its native footprints within a host and over a network association. The Tor browser, though, will leave behind digital artifacts which might be employed by the Associated investigator.

This paper outlines investigators' experimental methodology and provides results for proof trails that might be used among real-life investigations. The restricted analysis within the space of Tor Forensics discovered throughout the Literature Review steered that a necessity for live forensics had become progressively vital. This was for the most part. Thanks to recent papers, that Tor failed to write browsing knowledge to disk (Warren, 2017). Before this, Darcie et al (2014) had with success managed to retrieve files residing in RAM from their browsing protocol, even once the browser was deleted. This coincided with the rhetorical methodology planned by Dayalamurthy (2013), which additionally placed stress on live forensics. a brand new methodology was planned, that favored a live rhetorical analysis followed by a static analysis of the virtualized check machine.

Title: Peeling the Onion's User Experience Layer: Examining Naturalistic Use of the Tor Browser

The strength of an anonymity system such as Tor depends on the number of indistinguishable users, called its anonymity set. In an effort to strengthen the network and expand the set of indistinguishable Tor users, the Tor Project provides the Tor Browser that makes users less distinguishable by countering some application-layer tracking techniques, such as cookies, user-agent strings, or browser fingerprinting mechanisms[7]. Since the extent of anonymity is dependent on the number of indistinguishable users, it is important to provide user-centered security[8] by paying attention to the User eXperience (UX) of the Tor Browser.

3. Methodology

We first installed the Tor bundle from the official repository of Kali Linux, a Debian Testing distribution. The Tor bundle gets auto-installed into the system, in the /opt/ folder.

3.1 Configuration

After unhashing the prerequisites of initialization for the Tor Hybrid node from the torrc file, we restarted the service using systemctl with root privileges. Some of the parameters required for setting up the Tor node are included below.

Nickname: Username
ContactInfo: email@firemail.cc
ORPort: 9001
ExitRelay: 0
SocksPort: 0

As Tor has 3 nodes in itself, entering '0' in front of the Exit node, among the Entry/Guard, Middle/Relay, and Exit nodes, will skip the Exit node to be set up beside the Guard node and Relay. The Tor browser routes its every request through a custom proxy [6] if set by the user, also known as the SOCKS5 internet proxy[9]. Hence, SOCKS5 was set to 0 for this analysis, because of the absence of the apparatus.

3.2 Attestation Delay

After successful configuration, the Tor Network initialized the pre-verification of the node and started verifying the node by sending junk, unstable packets to check for latency. The Tor node setup took around 7 to 14 days to verify the node. After verifying the node, it started sending a high amount of traffic, limited to the bandwidth set by

[4]De-anonymization is a technique used in data mining that attempts to re-identify encrypted or obscured information.
[5]Address Space Layout Randomization (ASLR) is a technique used to mitigate Buffer Overflows
[6]https://www.contralegem.ch/2019-2-l-encryption-the-challenges-for-criminal-investigation-authorities-and-the-role-of-human-rights-guarantees/#top

[7]These techniques track the browser, IP address, and some hardware information of the user's system, including MAC addresses
[8]https://people.cs.vt.edu/~kafura/cs6204/Readings/Usability/UserCenteredSecurity.pdf
[9]SOCKS5 is a general-purpose proxy that sits at layer 5 of the OSI model and uses the tunneling method. Tor has a provision for it.

us, which was 10 Megabytes per second, along with the overall permissible internet usage, which was set to 900 Gigabytes. Once the bandwidth/speed of the traffic was set to acquire, the Tor network will only terminate the connection, if the usage goes beyond that limit. Initially, a less amount of usage was deduced, when the node was undergoing verification.

3.3 Packet Capture

We had used two tools, namely Nyx and Tshark. Nyx is used for monitoring the traffic over the network, whereas Tshark is used for eavesdropping, packet capture, and dynamic analysis. We had used a high-bandwidth SSD VPS[10] for smooth functioning of the node over the Tor network. We had made an automated bash script for packet monitoring and packet capturing activities. The script not only monitored and captured the packets, but also packed them into a file, and uploaded it over an anonymous P2P server, through which we could download it and analyze the same using more sophisticated tools, like Wireshark.

The script was installed on the VPS itself, and whenever we required its assistance in packet capturing, we would trigger the same, and it would capture the first 5,000 packets, and pack them into a text file.

```
sudo tshark -w /cap/nlog.pcap -c 5000 tcp && zip -r
capture.zip cap
```

3.4 Migration

The script would compress the file into a zip, and upload it to our other self-hosted server, which was our analysis host, through an API by using the cURL module[11].

```
curl -F "file=@capture.zip"
https://api.server.io:4444/upload
```

The host had a portal to upload files, and permitted anonymous file upload. After the upload was complete, we decompressed the file and imported it in Wireshark.

The Wireshark modules helped sort the packets according to their response time, and sequence number. We most commonly captured the TCP stream, to cover up all the packets transmitted from the VPS onto the Tor network.

3.5 Mainstream Transmission Automation

After the node had been set up, to avoid repetitive access, we designed another script to run as a daemon process, periodically capturing a specific amount of traffic, compressing it, and uploading it to our server. Periodic monitoring was although maintained to keep a track of the number of packets that were flowing through the system, by using Nyx[12], which is specifically made for visualizing data flow over port 9001, distributed by the Tor Project.

After performing a basic analysis by Wireshark, the packets were then imported to Savvius Omnipeek for detailed analysis. Observations were made from packets transmitted pre-attestation and post-attestation, and were compared for entropies.

All differences were meticulously cataloged.

4. Outcomes and Analysis

Meanwhile the node being verified by the Tor network, we started collecting packets. These packets were then migrated to Wireshark for analysis.

4.1 Heretofore Verification

Before the node was entirely set up, Tor directed a bunch of unbalanced packets, with transmission errors. These packets, being prerogative, were transmitted back to the source by the VPS with suitably processed packets, by the packet parser with improper fragmentation.

Furthermore, D-SACK[13] sequences were noticed to be transmitted in significance to the VPS. The Selective Right Edge (SRE) contained data that was avoided to be retransmitted, and Selective Left Edge (SLE) had no data but was relative. Since the ACK was a duplicate, TCP skipped the acknowledgment, but these types of packets were transmitted in the majority of the exchanges. A snippet of the transmission containing a SACK sequence is presented below.

```
Kind: SACK (5)
Length: 10
left edge = 1 (relative)
right edge = 1051 (relative)
[TCP SACK Count: 1]
[D-SACK Left Edge = 1 (relative)]
[D-SACK Right Edge = 1051 (relative)]
D-SACK Sequence
[Expert Info (Warning/Sequence):
D-SACK Sequence]
```

The SACK sequence is counted as a warning in the Wireshark dump and is highlighted black. The exchanges over port 9001 pre-verification are significantly less, and periodic. The packets with these headers are grouped as sequences and are given the duplicate ACK flags. The snippet also presents the number of the frame which was found containing the flag, which is categorized as a note. Wireshark saves these notes inside the IP Header for the analysis, and if the data is transferred through a digital certificate, like TLSv1.2, it binds the certificate-related information inside the Transport Layer Security stack [7].

[10]VPS (Virtual Private Server) is a hosting service that uses virtualization technology to provide you with dedicated (private) resources on a server.
[11]cURL is a tool for transferring data requests to and from a server using PycURL. This tool is used for testing REST APIs, downloading files, etc.
[12]https://blog.torproject.org/meet-nyx-command-line-tor-relay-monitor/
[13]http://www.icsi.berkeley.edu/ftp/global/pub/techreports/2002/tr-02-006.pdf

The snippet of the Frame is shown below, with the duplicate ACK, treated as a note.

[D-SACK Sequence]
[Severity level: Warning]
[Group: Sequence]
[SEQ/ACK analysis]
[TCP Analysis Flags]
[This is a TCP duplicate ack]
[Duplicate ACK #: 1]
[Duplicate to the ACK in frame: 33]
[Expert Info (Note/Sequence):
Duplicate ACK (#1)]
[Duplicate ACK (#1)]
[Severity level: Note]
[Group: Sequence]

Moreover, timestamps were detected which represented the time of packets transmitted for that TCP stream, which was observed to be significantly low for the SACK sequence.

[Timestamps]
[Time since first frame in this TCP stream: 0.802280353 seconds]
[Time since previous frame in this TCP stream: -0.000011318 seconds]

Alongside having a substantial speed of transmission, the packets also had a comparatively low window size. The checksum of these packets was unverified, and could not represent itself as a derivative to an integral source transmitting the packet. The block supporting this analysis was stored for post-verification transmission.

Window: 32
[Calculated window size: 32]
[Window size scaling factor: -1 (unknown)]
Checksum: 0xc6c2 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0

The source IP address was captured, which was later analyzed and deduced to be an IP address connecting to a Parked Domain. It had no headers or certificates depicting to be a VPN, or a Proxy. However, some transmissions were carried out by domains over TLSv1.2, and the data was found to be encrypted. Otherwise, the domain names were visible in Wireshark.

To check for the prolongation of the TLS connection which was established by the domain and sources from the node, the TLS Stack performed checks by sending ACK packets to the source, known as the Handshake protocol [8], periodically, to ensure connection longevity. When the connection was established without a security certificate, the request packed up all the header and content details unencrypted, as shown below.

doelinkswr.fe: Type A, class IN, addr xxx.xxx.xx.xxx
Time to Live: 58
Protocol: TCP (6)
Header Checksum: 0x8e25 [validation disabled]
[Header checksum status: Unverified]
Source Address: xxx.xxx.xx.xxx
Destination Address: xx.xx.xxx.xx
Transmission Control Protocol, Src Port: 34642, Dst Port: 9001, Seq: 537, Ack: 1051, Len: 0
Source Port: 34642
Destination Port: 9001
[Stream index: 1]

Whereas, if the transmission was carried out by a TLS connection, all of the headers and the destination address were encrypted, like the following snippet.

d83d12c8000a79f0aa8587ac4197b271ee9beca594e445296
038b95ce7c2f50aeaf16cad3e1e96697599ba0843d1c0cb04
1bb6828e1863140d6a31dc857b0674ea9c565e96bfdc6c35
662db853aa4fc59ecf346793af1274a4f0a8d6695ae533257
345f696207d53bab4595fa13861bc51fa8116c23594e3bd4b
9d2be7b537483c03260a844c029e58633dc7e645172f2fea1
a20944580c977c626bd03607e3d193a7cce46929ec4c6af15
6cc3e3a5e29ad89e7d6ee16e4d54e90cbc514254847359b0
e68f5695c0de3179e5495a348db6106fd18139e48cdd6645
bbee7bb7e8815c58a4dcb07090f77486b6dcf2788b0ae4c8d
c8a7f7fc6dd285cb869f65b69e0352b4b587539c7faf05959f
ba99ec0207311050f6102898250390fb5b624c901c1487eb
760d9685766022c59195daeba60b6a0acc187106828d9785
1ebac001e5b77349b0358d7822b6a6f0648eaa48811a20fe7
a9469d874dde675a0345da966c350aca0e5923e1ba806dc4
183aea44c75ef90eed2cc46142d16f2ab14fcdb6ff92b4a0cf
63362fe70facff8bb900b7b681c159d0e7bd1202cd79c4957
1d61f391e73934349b41a3bbd0552a14f5a236f3b3e17847
d5596aaf1dacb28430996a649241ab214e248bdcce1b3a7a2
d170a0168f1e1c7b9c6f04a1d6970e6d21d7a40cb1589be65
ceb869863694c513a2159fb629c97489c405e86ac8eadbd3
87c3d18d02fbcf9dc9f01bb22ec1482eef3ce5a

All of the data transmitted over the ens3 interface[14], depicting the use of VPS services.

4.2 Post-Verification Analysis

After the verification was concluded, the node started receiving humongous amounts of traffic, within its usage limits. In the early stages of the ongoing verification period, Nyx captured minor traffic within a given period of time. From the analysis, Tshark captured around 20 packets getting transmitted over port 9001 in 3 seconds. After the verification was complete, port 9001 was flooded with exchanges, resulting in the capture of over 7,600 packets in 3 seconds, with a given 10 Mb/s of permissible speed. Moreover, additional two fields were observed in the normal exchanges, which were now transmitting TCP Payloads over the network.

TCP payload (536 bytes)
Retransmitted TCP segment data (536 bytes)

The rest of the packet content remained the same, except for these two fields.

[14]The ens3 interface is used by guests on a system to communicate to the host. In this case, it's a guest on a VPS

4.3 Juxtaposition of Outcomes

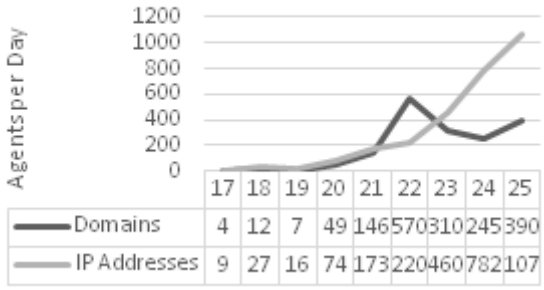
All the observations were then cataloged in tabular form, with an aim to compare the node traffic before and after the verification. The observations are listed in Table 1.

Table 1: Comparison between the Hybridized Node Pre and Post Verification

Parameters	Prior Node Attestation	After Node Attestation
Transferal Speed (Avg.)	20 packets/s.	7,650 packets/s.
Connection Security (Avg.)	Loose. Less to no Encryption. TLS enforced scantily.	Strict. TLSv1.2. RSA-512 exchange keys.
Exchanges	High speed. Smaller TCP window size.	Very high speed. Larger TCP window size.
Node Broadcast	To Attestation Agents.	To the Tor Network.
System Memory Usage	Low.	Low.

After the node was set up, the node was cataloged by the Tor network onto their official website[15], for people to get information about the same, along with the details of the node owner. After one to two days of verification, the node agents significantly increased to make use of the node for transmissions. The agent count experienced a drastic elevation, and the bandwidth reached the maximum, being 10 Megabytes per second, which did not go above 330 Kilobytes per second, when under verification. The agent elevation is depicted in the graph below, which was derived from the observations.

Graph 1. Node Agents (June 2021)



After the node reached its maximum data usage limit, it automatically got shut down and stopped broadcasting itself into the Tor network. Sooner, Tor acknowledged this change and mailed about the downtime of the node.

After three days of downtime, the node was removed from the list which was hosted on the Tor website. Port 9001 stopped receiving transmissions and the node got off the grid. Nyx displayed zero new transmissions, henceforth.

[15]https://metrics.torproject.org/rs.html#advanced

5. Conclusion

Relays in the Tor network have an extremely fast routing, which utilize all of the bandwidth they are given, when under configuration. The initial ratification might be a bit transparent for the Middle and the Guard nodes, but after verification, the content on the Middle Relay is encrypted entirely, with some unencrypted data transferred from the Guard node.

The speed of the network is significantly high, powered by individual sources and circuits. The packet size is less when under the examination period by the Tor network, containing checksums, probes, and TLS headers, pertaining to the connection using the handshake protocol, than the packet size after the verification, which then starts transmitting payloads through the TCP stream.

Irrespective of verification, the Tor network keeps pinging the node once two days, for the acknowledgment of the uptime or downtime of the relay, and updates it to its official website. Though, when sources are suspicious or malicious, Tor can enforce policy violations and can suspend the nodes.

The Entry/Guard node always displays the TCP stream headers in an unencrypted format, involving the IP address. If a TLS certificate is authorized at the source, the domain name is encrypted; otherwise, the domain name is also visible inside the header.

The more volunteers set up their nodes into the network, the faster the routing is.

References

[1] Haraty, Ramzi & Zantout, Bassam. (2014). The TOR data communication system: A survey. Communications and Networks, Journal of. 16. 415-420. 10.1109/JCN.2014.000071.
[2] F. Shirazi, M. Goehring and C. Diaz, "Tor Experimentation Tools," 2015 IEEE Security and Privacy Workshops, 2015, pp. 206-213, doi: 10.1109/SPW.2015.20.
[3] Mauro Conti, Stephen Crane, Tommaso Frassetto, Andrei Homescu, Georg Koppen, Per Larsen, Christopher Liebchen, Mike Perry, and Ahmad-Reza Sadeghi. (2016). Selfrando: Securing the Tor Browser against De-anonymization Exploits, Journal of. 4. 454–469. 10.1515/popets-2016-0050
[4] Matt Muir, Petra Leimich, William J. Buchanan, A Forensic Audit of the Tor Browser Bundle, Digital Investigation, Volume 29, 2019, Pages 118-128, ISSN 1742-2876, https://doi.org/10.1016/j.diin.2019.03.009.
[5] Kevin Gallagher, Sameer Patil, Brendan Dolan-Gavitt, Damon McCoy, and Nasir Memon. 2018. Peeling the Onion's User Experience Layer: Examining Naturalistic Use of the Tor Browser. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18). Association for Computing Machinery, New York, NY, USA, 1290–1305. DOI: https://doi.org/10.1145/3243734.3243803

[6] Saleh, Saad & Qadir, Junaid & Ilyas, Muhammad. (2018). Shedding Light on the Dark Corners of the Internet: A Survey of Tor Research. Journal of Network and Computer Applications. 114. 10.1016/j.jnca.2018.04.002.
[7] Satapathy, Ashutosh & Livingston, Jenila. (2016). A Comprehensive Survey on SSL/ TLS and their Vulnerabilities. International Journal of Computer Applications. 153. 31-38. 10.5120/ijca2016912063.
[8] E. Stark et al., "Does Certificate Transparency Break the Web? Measuring Adoption and Error Rate," 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 211-226, doi: 10.1109/SP.2019.00027.