ISSN: 0975-766X
CODEN: IJPTFI
Available Online through ***Research Article***
www.ijptonline.com

DERIVING SHORTEST ROUTE IN TOR NETWORK

Shantanu Dasgupta[1], Garima Rathore[2], P.M. Durai Raj Vincent[3]
[1,2] 2nd year MCA Student, SITE, VIT University.
[3] Associate Professor, SITE, VIT University.
Mail:shantanu.dasgupta2015@vit.ac.in

Received on 25-10-2016 *Accepted on 02-11-2016*

Abstract:

TOR stands for "The Onion Router". TOR is a prevalent privacy improving framework that is intended to shield the protection of Internet clients from traffic investigation attacks propelled by a non-worldwide adversary. Since TOR gives an anonymity benefit on top of TCP while keeping up generally low latency and high throughput, it is perfect for interactive apps, for example, web browsing, instant messaging and file sharing. Since its introductory development, analysts have broken down the system's execution and security properties. However, there has yet to be a research aimed at deriving the shortest route circuit in the TOR network. This paper contains the detailed specifications of the network path selection of the TOR network and how it can be shortened along with maintaining the anonymity of the connection to increase the speed of the communication.

Keywords: TOR, TCP, Path, Anonymity, Privacy, security, circuit, route.

Introduction

The Onion Router or TOR is a privacy enhancer network which is made for protecting the privacy of people on the internet from traffic investigation attacks propelled by a non-worldwide adversary. It is a free volunteered network service that enables anonymous communication. The Onion Router guides the traffic on the internet via free, global and volunteered networks formed by more than 7 thousand relays to hide the location and usage of an internet user, therefore, making it impossible to track the requests back to the users. Onion routing is actualized by encryption in the application layer of a correspondence protocol stack, settled like the layers of an onion. The Onion Router scrambles the information, including the goal IP address, different times and sends it through a virtual circuit involving progressive, haphazardly chose The TOR transfers. Every transfer decodes a layer of encryption to uncover just the following hand-off in the circuit to pass the rest of the encoded information onto it. The last transfer decodes the deepest layer of encryption and sends the first information to its goal without uncovering, or without knowing, the

source IP address. Since the steering of the correspondence is incompletely hidden at each bounce in the TOR circuit, this technique takes out any single time when the imparting associates can be resolved through system observation that depends onafter knowing its source and destination. Although the TOR network is a good option for the anonymous network but the main drawback of using TOR is that you get a very slow communication speed due to the heavy path relays of the network. Since there are more than seven thousand TOR relays, the circuit selection is very important. The current circuit selection of the TOR network consists of 3 routers. The first relay is the Entry Guard. Second is the middlerelay and third is the Exit node. Each TOR communication goes through these three relays to complete a request and response, therefore making the communication speed slower than normal networks. Our aim is to minimize the network speed by selecting the closest relays out of the seven approx. a thousand relays present in the TOR network. The TOR network is an overlay system; every onion router (OR) keeps running as a typical client level process with no extraordinary benefits. Every onion router keeps up a TLS association with each other onion switch. Every client runs nearby programming called an onion proxy (OP) to bring registries, build up circuits over the system, and handle associations from client applications. These onion intermediaries acknowledge TCP streams and multiplex them over the circuits. The onion switch on the opposite side of the circuit associates with the asked for goals and transfers information.

Every onion router keeps up a long term character key and a transient onion key. The unique key is utilized to sign TLS testaments, to sign the OR's switch descriptor (a rundown of its keys, address, data transfer capacity, exit strategy, et cetera), and (by index servers) to sign catalogs. The onion key is utilized to decode demands from clients to set up a circuit and arrange transient keys. The TLS convention additionally sets up a transient connection key when conveying between ORs. Fleeting keys are turned intermittently and freely, to restrain the effect of key exchange. Onion switches speak with each other, and with clients' OPs, by means of TLS associations with fleeting-keys. Utilizing TLS covers the information on the association with flawless forward mystery, and keeps an assailant from changing information on the wire or mimicking an OR.

Onion Routing initially assembled one circuit for every TCP stream. Since building a circuit can take a few tenths of a second (because of open key cryptography and system idleness), this outline forced high expenses on applications like web surfing that open numerous TCP streams.

In TOR, every circuit can be shared by numerous TCP streams. To stay away from deferrals, clients develop circuits pre-emptively. To point of confinement linkability among their streams, clients' OPs construct another circuit

intermittently if the past ones have been utilized, and lapse old utilized circuits that no longer have any open streams. Operations consider pivoting to another circuit once per minute: along these lines even overwhelming clients invest unimportant energy building circuits, however, a predetermined number of solicitations can be connected to each other through a given leave hub. Likewise, on the grounds that circuits are implicit the foundation, OPs can recoup from fizzled circuit creation without hurting client encounter.

Literature Survey

Damon McCoy,et.al [1]:This paper gives a detailed introduction to the TOR network. It also focuses on some key points such as how TOR is used, how it is misused, who is using it.Also, they analyzed that the majority of the bandwidth and connections are taken by the web traffic and also they created a procedure to detect TOR exit router logging.

Michael Backes, et.al [2]: In this paper, they have presented a framework called MATOR which stands for Monitoring Anonymity of The Onion Router's Path Selection. The framework unequivocally addresses how client anonymity is affected by genuine qualities, for example, its path determination. MATOR involves light-weight continuous screens that process sender, beneficiary and relationship obscurity guarantees in view of the genuine TOR consensus information and the client requested ports.

Richard Clayton, et.al [3]: This paper shows the different notations that are used in the definition of MIX networks to illustrate the nested secured structures that are globally known as 'onions'. The disadvantages of these notations are illustrated and a new notation is described in this paper. The authors used many different ways trying to improve the onion notation. As a result, they came up with the star notation which they have described in this paper in detail.

Lesse Overlier, et.al[4]: This paper shows how usage of rapid and low-cost attacks disclose the position of a secret server using a single combative TOR node. The paper also focuses on improving the route selection of the TOR network to prevent attacks. The authors showed interest in changing the selection scheme and enactment for TOR. These changes do not need any increase in network bandwidth and are easy to implement and they accomplish the task of this paper.

Phillip Winter, et.al [7]: The main objective of this paper is to design and implement exitmap. It is a flexible and rapid exit relay scanner which is capable of finding many MITM attacks on the network. They look to uncover vindictive exit relays and record their actions. Their work makes it easy to watch the TOR exit relays continuously for malicious behavior such as theman in the middle attacks.

TOR Architecture

The TOR uses a technique called Onion Routing. It is a technique for anonymous and secured communication over the internet. In an Onion network, messages are epitomized in layers of encryption, similar to layers of an onion. The scrambled information is transmitted through a progression of system hubs called onion switches, each of which "peels" away from a solitary layer, revealing the information's next goal. At the point when the last layer is unscrambled, the message touches base at its goal. The sender stays mysterious in light of the fact that every middle router knows just the address of the promptly going before and taking after nod

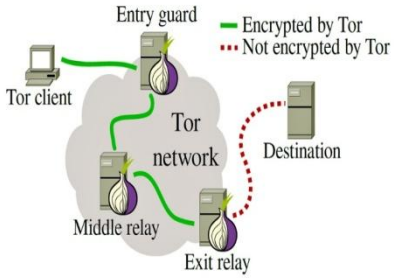


Fig. TOR Network Structure.

As the above figure shows, the structure of a TOR network is described as following:

- A TOR client sends a page request in the network. The TOR network encrypts the request information.
- The request is transferred to the Entry Guard as shown. The entry guard doesn't know from where the request was originated. It just knows the address of the next destination. After getting the page request, the Entry guard transfers the request to the next node.
- The next node in this process is the middle relay. The node scans for active exit nodes and picks a random node to perform the transfer and executes the transfer.
- The final node in the transfer is the exit node chosen by the middle relay. The exit node receives the encrypted information. Decrypts the information and then transfers it to the original destination. Thereby hiding the location of the original sender.

Proposed Methodology

In our attempt to minimize the time taken to complete a TOR communication while keeping it anonymous, we created a function that derives the shortest route circuit from the source IP to the destination. As we know, the TOR provides more than seven thousand relays for communicating, we used the IP addresses of those relays to dynamically create the closest possible circuit from source to destination.

Implementation: The above algorithm has been implemented by using PHP and MySQL languages. For testing the functionality of the code, a server is needed. Xampp Server is used to run the code. Xampp server contains modules such as apache to run PHP code and MySQL to run MySQL queries for the database needs.

The Algorithm is described as following:

Step 1:

- Define the Function Route with two arguments SRC_IP & LIST array.
- If SRC_IP is present in the list array then return it and end the function.
- Otherwise, proceed by creating a list with four variables and assigning the values as each part of the SRC_IP.
- Create three arrays namely Triples, Doubles, Singles.

Step 2:

- Start a foreach loop foreach LIST as IP.
- Create another list with four variables and assign the values as each part of the exit nodes IPs.
- Check if the first three parts of the SRC_IP and exit nodes IP are equal.
    - If equal then assign the IP to TRIPLES array
- Otherwise, check if the first two parts are equal.
    - If equal then assign the IP to DOUBLES array
- Otherwise, check if the first part is equal.
    - If equal then assign the IP to SINGLES array.
- End foreach loop.

Step 3:

- Check if the Count of TRIPLES array is more than 0.
    - If true then assign TRIPLES array to LIST array.
- Otherwise, check if Count of DOUBLES array is more than 0
    - If true then assign DOUBLES array to LIST array.
- Otherwise, check if Count of SINGLES array is more than 0
    - If true then assign SINGLES array to LIST array.

Step 4:

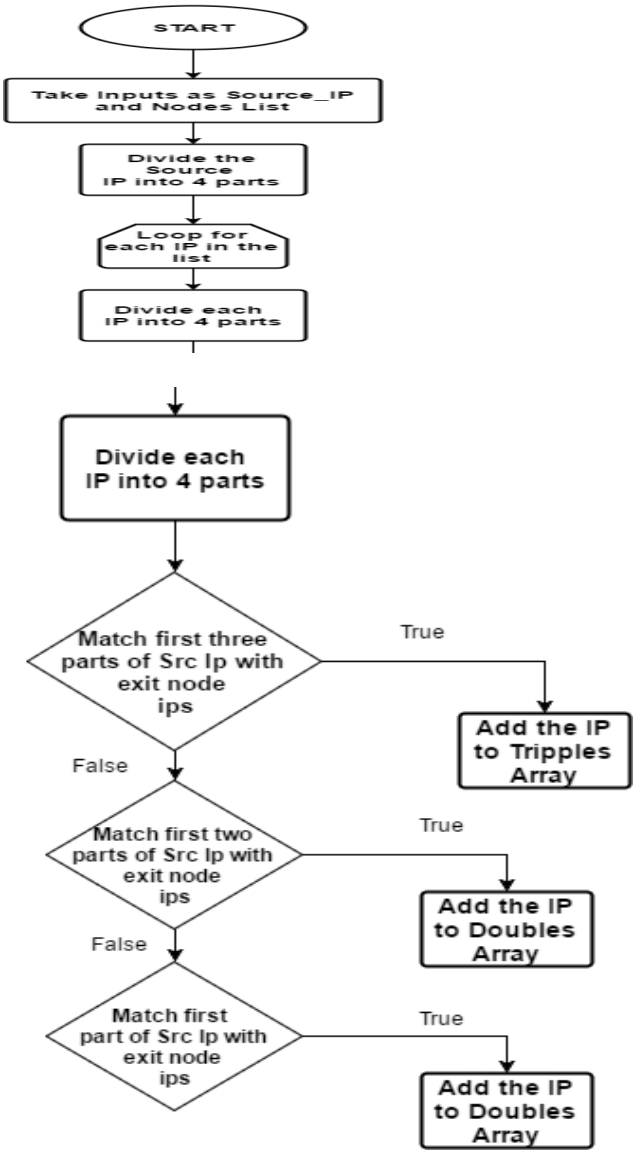
- Assign Maximum Integer Value to MIN variable.

- Assign false to RTN variable.
- Assign Long version of SRC_IP to L1 variable

Step 5:

- Start a foreach loop foreach LIST as IP.
- Assign Long version of IP to L2.
- Assign the difference between L1 and L2 to D.
- Check if MIN is greater than D
    - If True then assign IP to RTN and D to MIN.
- End foreach loop.
- End Function.

Following is the Flowchart of the algorithm stated above



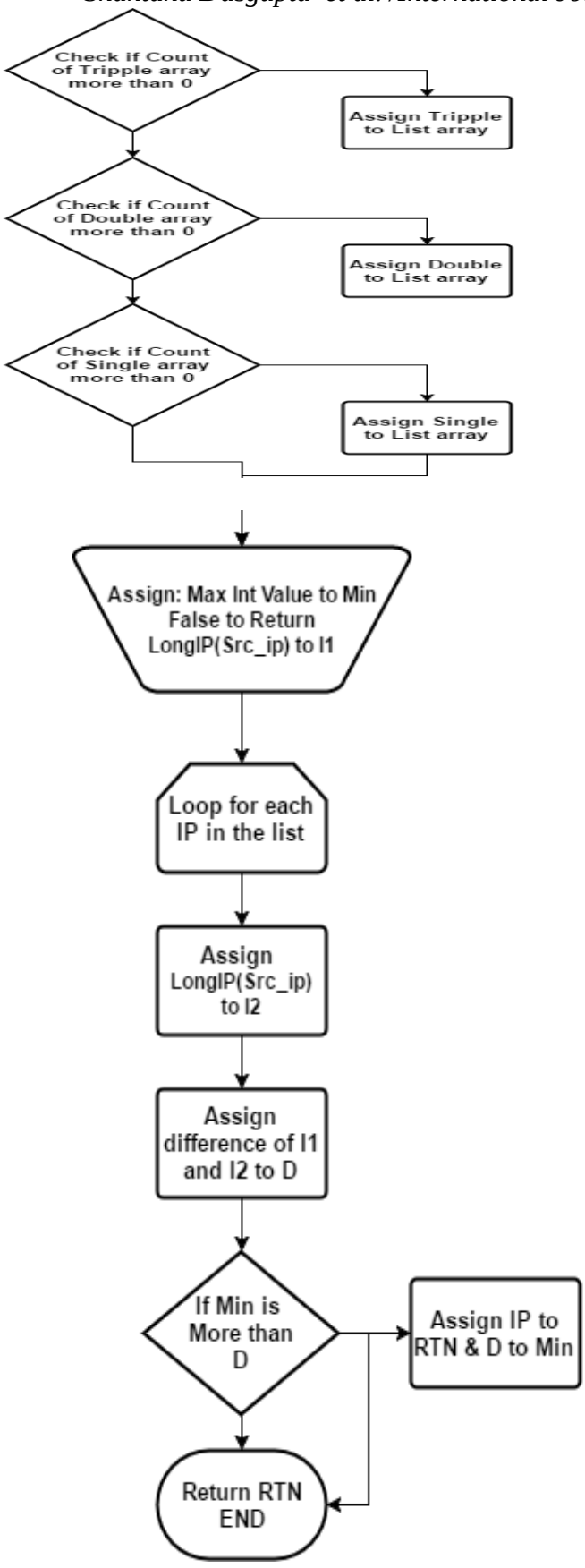


Fig. - Flowchart of Algorithm.

Results

The algorithm that is implemented finds the closest IP address to the given source IP address. Initially, it checks the first three blocks and checks if they are equal or not. If they are equal then assigns the IP to the triples array. The said

process is followed for first two and the first part of the IP address. After numerous comparisons and conversions, the closest IP address to the source IP that is available in the exit nodes list is returned. This process is repeated three times with each time providing the resultant IP as the source IP for the next iteration and hence generating a routed circuit. We provided the Source IP address as 27.251.197.194 which belongs to Vellore, Tamil Nadu, India. The Entry Guard node that was derived is 27.50.94.251 which belongs to Tuggerah, New South Wales, Australia. The Middle-Relay came as 27.32.220.198 which belongs to Williamstown, Victoria, Australia. The final Exit node emerged as 27.0.235.57 which belongs to New Delhi, National Capital Territory of Delhi, India. And then connected to the destination. The screenshot of the Exit nodes and the Route generation is provided below:

S.No.	IP Address	S.No.	IP Address	S.No.	IP Adc
1	100.11.109.99	2	100.12.241.131	3	100.36
5	101.161.175.212	6	101.164.68.36	7	101.16
9	101.98.11.146	10	101.98.200.66	11	101.98
13	103.10.199.100	14	103.14.68.50	15	103.19
17	103.233.25.57	18	103.234.36.144	19	103.23
21	103.25.58.34	22	103.252.221.229	23	103.3.
25	103.37.128.253	26	103.41.177.49	27	103.41
29	103.44.149.45	30	103.8.79.229	31	103.9.
33	104.128.226.73	34	104.130.169.121	35	104.13
37	104.131.108.203	38	104.131.108.7	39	104.13
41	104.131.123.16	42	104.131.128.247	43	104.13
45	104.131.137.159	46	104.131.148.86	47	104.13
49	104.131.19.119	50	104.131.204.147	51	104.13
53	104.131.245.55	54	104.131.28.54	55	104.13

Fig. Exit Nodes Screenshot

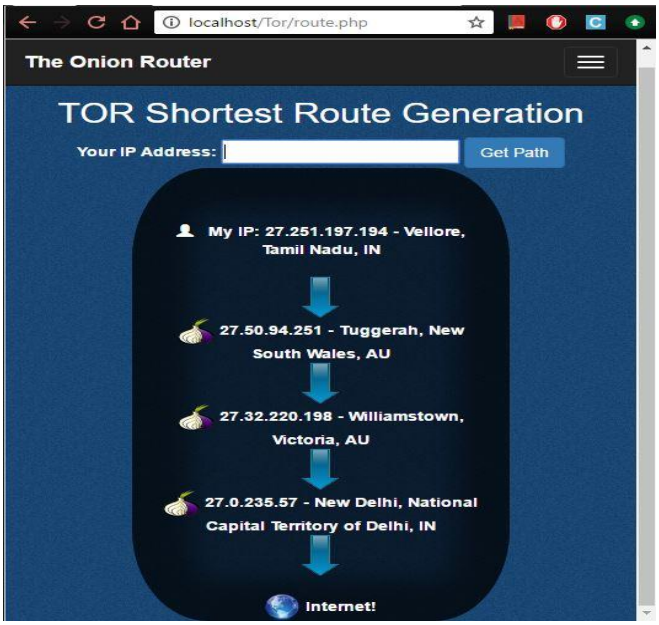


Fig. Route Circuit Generation

Conclusion & Future Work

The main aspect of the TOR network is its anonymity. Therefore we were limited to choose from only around 7000 network IP addresses to create the shortest route circuit for the TOR network. Although, we successfully generated the shortest possible route circuit according to the source IP address given to the program. Local testing of the function gave positive results but the code is yet to be tested in the TOR browser for testing of the anonymity and speed. For testing of anonymity and speed, we would require access to Web Server running Apache and MySQL, a TOR Browser and Updated TOR exit nodes list. For now, the purpose of the paper to create the shortest route in the TOR network is fulfilled.

References

1. McCoy, Damon, et al. "Shining light in dark places: Understanding the Tor network." International Symposium on Privacy Enhancing Technologies Symposium. Springer Berlin Heidelberg, 2008.
2. Backes, Michael, et al. "(Nothing else) MATor (s): Monitoring the Anonymity of Tor's Path Selection." Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014.
3. Clayton, Richard. "Improving onion notation." International Workshop on Privacy Enhancing Technologies. Springer Berlin Heidelberg, 2003.
4. Overlier, Lasse, and Paul Syverson. "Locating hidden servers." 2006 IEEE Symposium on Security and Privacy (S&P'06). IEEE, 2006.
5. Mayer, Jonathan R., and John C. Mitchell. "Third-party web tracking: Policy and technology." 2012 IEEE Symposium on Security and Privacy. IEEE, 2012.
6. Paul Syverson, David M. Goldschlag, Micheal G. Reed, Anonymous connections and Onion Routing "Security and Privacy", Proceedings 1997 IEEE Symposium on IEEE, 1997.
7. Reed, Michael G., Paul F. Syverson, and David M. Goldschlag. "Anonymous connections and onion routing." IEEE Journal on Selected areas in Communications 16.4 (1998): 482-494.
8. Goldschlag, David, Michael Reed, and Paul Syverson. "Onion routing." Communications of the ACM 42.2 (1999): 39-41.
9. Reed, Michael G., Paul F. Syverson, and David M. Goldschlag. "Proxies for anonymous routing." Computer Security Applications Conference, 1996., 12th Annual. IEEE, 1996.

10. Diaz, Claudia, et al. "Towards measuring anonymity." International Workshop on Privacy Enhancing Technologies. Springer Berlin Heidelberg, 2002.
11. Syverson, Paul. "Onion routing for resistance to traffic analysis." DARPA Information Survivability Conference and Exposition, 2003. Proceedings. Vol. 2. IEEE, 2003.
12. Vincent P.M.D.R, Sathiyamoorthy E, "A novel and efficient key sharing technique for web applications" in IEEE Fourth International Conference on Computing, Communications and Networking Technologies. 2013.