Deanonymization and Denial-Of-Service Attacks Against the Tor Network

1. INTRODUCTION

The growing presence of large-scale internet surveillance has brought into question the concept of secure and anonymous use of the internet, and subsequently lead to networking innovations such as The Onion Router (Tor). Tor is a low latency TCP-traffic anonymizing communication network based on the principles of Onion Routing and decentralization.

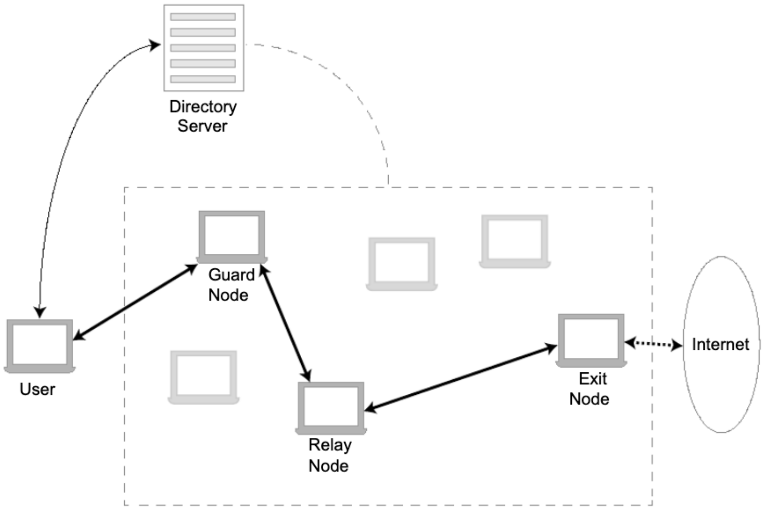
Tor achieves secure and anonymous TCP traffic transmission via the Tor protocol. The Tor protocol based on the principle of Onion Routing is essentially an end-to-end encryption protocol which facilitates anonymous and secure TCP stream over a public network by encapsulating TCP traffic in Tor cells, which are then encrypted with several layers of encryption at each hop through a decentralized network of routers.[1][9]

Tor's decentralized architecture is one of the networks primary positive, but it is also a primary drawback, as a decentralized network by design allows that any individual on the internet can become part of the network. In terms of security, one of the primary drawbacks of Tor is that the networks decentralized architecture may allow for malicious actors to compromise the network via hijacking legitimate nodes or maintaining malicious nodes. [8]

2. FUNCTIONING OF TOR

2.1 TOR COMPONENTS

In this section, the components of Tor architecture will be discussed.



2.1.1 Tor Cell

Cells are a 512-byte transmission structure used in Tor, which consist of a payload and a header. Tor consists of two cell types, being control and relay.[1][9]

A control cell is used to construct a transmission circuit through the Tor network. Control cell command are CREATE (Establish a circuit), CREATED and DESTROY (Delete a circuit).

Relay cells are used in in end-to-end transmission of TCP traffic via the transmission circuit. RELAY cell commands are BEGIN, DATA, END, SENDME, EXTEND, DROP, and RESOLVE. [1][9]

2.1.2 Tor Circuit

A multi-hop path consisting of multiple Tor nodes which facilitates anonymous low latency TCP stream form the Tor client to client destination through the Tor network.[1][9]

2.1.3 Tor Relay

A volunteer operated publicly listed server, which facilitates the forwarding of traffic across the Tor network. Additionally, Tor Relays are also known as Nodes. [1][9]

Tor consists of three relay types, being Entry/Guard Node, Middle Node and Exit Node. Each Node has varying privileged access to crucial identifying information points such as IP Address of the Tor client, client destination, and other Nodes.[1][9]

On the basis of the principle of Onion Routing, a Node cannot view the instructions or transmitted payload of another Node. The transmitted data and instructions of a specific Tor Node can only be viewed via decryption with a shared symmetric key negotiated between the Tor Node and the client during the TLS session during path/circuit establishment.[1][5][9]

2.1.4 Entry/Guard Relay

The entry point of a Tor client to the network. Guard Nodes are privileged nodes as they are the only point in the Tor Network where the original IP Address of the client can be observed via legitimate non-malicious processes and operation of the network.[1][9]

2.1.5 Middle Relay

The primary traffic forwarding relay in the Tor Network. Majority of the Tor network consists of multiple Middle Nodes' encrypting and forwarding traffic. Middle Node's only have access to the IP addresses of the immediate predecessor relay and successor relay in the network.[1][9]

2.1.6 Exit Relay

The primary and final exit point in a Tor circuit. Exit Nodes are privileged nodes as they are the only point in the network where the IP address of the client destination can be observed. The final transmission of the payload is in plaintext but can potentially be encrypted if communication with destination IP Address is using TLS/SSL. As the final payload transmission is in plaintext, exit nodes are often considered the origin IP address of the traffic, consequently hiding the IP Address of the Tor client.[1][4][9]

2.1.7 Directory Authority

Directory authorities are nodes that maintain the consensus of the Tor network by regularly updating the list of active nodes. Tor clients and Nodes are continually updated with the state of the network and the Tor Consensus by directory authority nodes. Directory Authority Node keep track of the Tor network state by periodically receiving state and operating updates from all network Nodes.[1][9]

2.1.8 Tor consensus

Tor uses consensus to ensure the stability and security of the network. Tor consensus is a regularly updated list of Tor nodes that are added and removed from the network. When all 9 directory authority nodes exchange, negotiate, compile and sign node information and network statuses from all the Tor directory authority nodes, a Tor consensus is formed.[1][9]

2.1.9 Tor Consensus weight

A value which determines the probability of a Tor Node being chosen by the Tor Node Selection Algorithm during the process of constructing a transmission path through the Tor network. The Tor Consensus weight of a Node is determined by the Nodes bandwidth. A large bandwidth would result in the Node being assigned a large consensus weight by the Directory Authority Node.[1][9]

2.2 TOR CIRCUIT

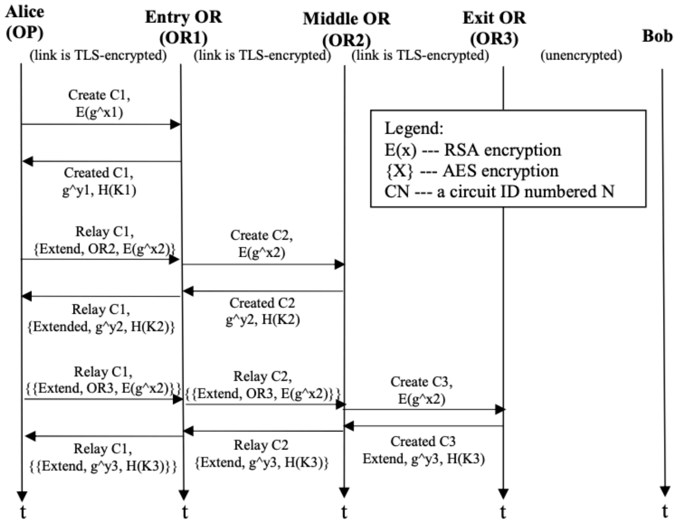
2.2.1 CIRCUIT NODE SELECTION

The first step of the data transmission process is to establish a circuit through the TOR network via the Tor client. The main components of the circuit establishment are the Tor directory nodes and the Tor Node Selection Algorithm.

The Tor Node Selection Algorithm primarily selects nodes based on the bandwidth. Nodes that have more bandwidth, have a higher probability to be chosen in the circuit creation. The tor client also checks the Tor consensus, in order to verify the validity, status and flags of the selected Node to be used in the circuit. [3][5][9]

2.2.2 CIRCUIT ENCRYPTION

The encryption process of the circuit is essentially ECDHE key exchange incrementally across the transmission circuit, negotiating TLS sessions with each Node, resulting in a shared secret key. [5][6][9]



The first key negotiation of the encryption process is between the Tor client and Entry Node. The client (Alice) initially sends a CREATE cell (C1) to Entry Node (OR1), with the payload E(g^x1). g^x1 is the first half of a DH handshake and E () is RSA encryption with the OR1's public onion key. In response, OR1 sends a CREATED cell (C1) to the client with the second half of the DH handshake (g^y1) along with a hash of the final key (H(K1)), after decrypting the RSA encryption with the private key. The client and OR1 have created a shared key which can be used by OR1 to decrypt and encrypt data during transmission. [2][9]

The second key negotiation of the encryption process is between the Tor client and a Middle Node. The client initially sends a RELAY EXTEND (C1) cell to the Entry Node (OR1) containing the address of the successor Middle Node (OR2), and the payload E(g^x2). g^x2 is the first half of a DH handshake and E () is RSA encryption with the OR2's public onion key. Upon receiving the RELAY EXTEND cell (C1), the Entry Node (OR1) extracts the Middle Node's (OR2) public key and forwards the public key to the Middle Node (OR2) in a CREATE cell (C2). In response, OR2 sends a CREATED cell (C1) to OR1 with the second half of the DH handshake (g^y2) along with a hash of the final key (H (K2)), after decrypting the RSA encryption with the private key. Upon Receiving OR2's CREATED cell, OR1 passes the cell to the client in a RELAY_EXTENDED cell (C1). The client and OR2 have successfully created a shared key which can be used by OR2 to decrypt and encrypt data during transmission. [2][9]

2.2.3 CIRCUIT TRANSMISSION

The data transmission process via a Tor circuit is done via RELAY DATA tor cells. The client loads the TCP data into the RELAY DATA payload and the RELAY DATA cell is encrypted with the shared symmetric key for each Node in the circuit. At each hop in the circuit, the Node decrypts the cell encryption with the shared key negotiated between the client and the Node.

3. VULNERABILITIES, EXPLOITS AND MITIGATION

3.1 SYBIL ATTACK

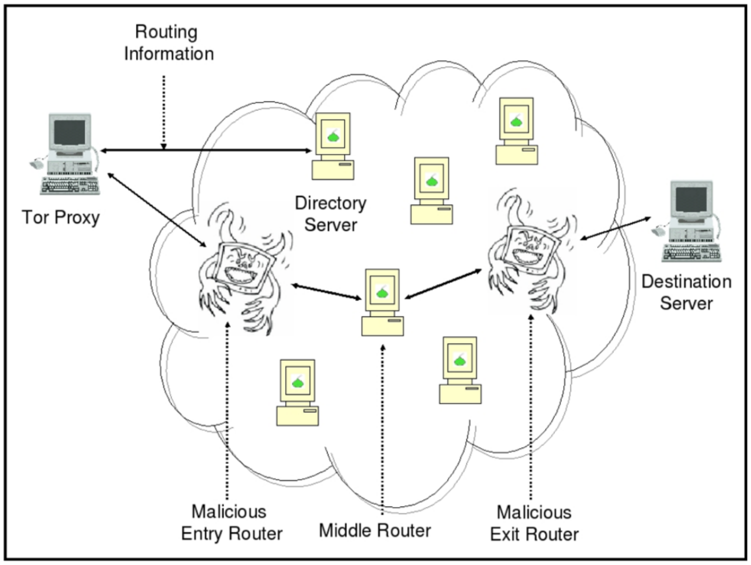
A Sybil attack is essentially a consensus-based attack on a decentralized network, wherein an attacker is attempting to shift the network consensus in their favor by obtaining disproportionately large control in a network via the hijacking of legitimate machines or the deployment of bots or malicious machines. The attack essentially exploits the one of the primary bases of decentralization, being that any individual can become a component of the network.[8][9]

3.1.1 VULNERABILITY

In order to initiate the encrypted TCP stream transmission, the Tor client initially constructs a transmission circuit through the Tor network via with the Tor Node Selection Algorithm which essentially optimizes the circuit for efficient transmission by choosing high bandwidth Nodes. The Tor client chooses the Nodes by referencing the Tor consensus which is periodically updated by Tor Directory Authority Nodes.[2][8][9]

The consensus weight of a Tor Node is essentially a direct correlation to the bandwidth of the Node. A Tor Node with a high bandwidth will have a high consensus weight. A high consensus weight will increase the probability that the Node will be chosen by Tor Node Selection Algorithm, consequently allowing the Tor Node to observe more TCP traffic across the network.[8][9]

3.1.2 EXPLOIT



An attacker can potentially increase the consensus weight of his nodes and successfully control and observe more traffic across the tor network. The attacker can set up a network of high-bandwidth, high-uptime Sybil nodes or increase the bandwidth of a single Node. A single Sybil

Node with a usually high amount of traffic and a large bandwidth may alert the Tor Directory Authority Nodes and allow the Tor Directory Authority Nodes to more easily remove the malicious Node from the Tor consensus. Additionally, the attacker can setup multiple Sybil Nodes that share a single IP address.[8][9]

A malicious Sybil Node being chosen as a Middle Node in a Tor circuit may allow the attacker to execute deanonymization attacks such Website fingerprinting and Bridge Address Harvesting.

A website fingerprinting attack essentially enables the attacker to trace back a specific destination IP address to the source IP address, consequently de-anonymizing the client's activity. Even though the Middle Nodes in a Tor circuit cannot see the encrypted TCP stream of the entire Tor circuit, an attacker can potentially make use of packet instructions and information transmitted via the Node to form a correlation between the origin IP Address and destination IP Address.

A Bridge address harvesting attack essentially enables the attacker to identify Bridge nodes in the Tor network. Bridge Nodes are private Tor Nodes, and are unlike other Tor Nodes, as they are not publicly listed and cannot be easily identified. Bridge Nodes essentially provide an added level of anonymity, in cases wherein publicly listed Tor Nodes are offline or actively monitored. The attacker can reverse reference the source IP address of observed TCP packets with the publicly listed Tor Node IP addresses. If the incoming IP address does not match any Node addresses, incoming connection may be from a Bridge Node, thus identifying a Bridge Node.[1][9]

A malicious Sybil Node being chosen as an Entry/Guard Node or Exit Node in a Tor circuit may allow the attacker to execute Man-In-The-Middle attacks such as traffic interception and tampering.

In a traffic interception and tampering, if the Exit Node is a malicious Sybil Nodes, after the final onion 'delayering' or decryption process with the client shared negotiated key, the attacker can potentially intercept the client's plaintext traffic during plaintext transmission with the destination, thus comprising both integrity and authenticity of the transmitted payload.[8]

3.1.3 MITIGATION

The decentralized architecture of Tor cannot be changed in order to mitigate Sybil attacks.

In order to mitigate the Sybil attacks, the primary objective would be to limit the addition of malicious Sybil nodes to the Tor Network.

Tor Directory Authority Node's monitor Nodes, and decidedly add or remove the Nodes from the Tor consensus. The latest versions of Tor have implemented countermeasures against Sybil Nodes by enforcing IP Address restrictions allowing a single public IP Address to host only 3 Tor Nodes.

Another Sybil Attack mitigation technique is a change to the Tor Node Selection Algorithm, which is one of the primary bases for the Sybil Attacks success, as Sybil Nodes usually advertise high bandwidths in order to be selected in Tor circuit construction by the Tor Node Selection Algorithm.

3.2 CELLFLOOD ATTACK

A Cell Flood attack is essentially a Denial-Of-Service attack on the Tor network, as it essentially prevents legitimate Tor users to transmit data through the Tor Network, as the user's Tor clients will not be able to initially construct a transmission circuit through the network.[2][6]

A malicious client essentially floods Tor Nodes with computationally intensive execution circuit creation requests (CREATE), which consequently reduces the Tor Nodes processing power, and finally results in the Tor Nodes rejecting and dropping all circuit creation commands. As the Tor Nodes are rejecting all circuit creation commands from Tor Clients, Tor clients are essentially denied service. Additionally, the Tor network may become incapacitated, as the overloaded Tor Nodes, will be removed from the Tor consensus by the Tor Directory Authority Nodes, thus reducing the number of active online Nodes in Network. A reduction in the number of Nodes, could potentially lead to the reduction in users, which may compromise the stability and anonymity of the network.[2][6]

3.2.1 VULNERABILITY

As discussed in section 2.2.2, the encryption of a Tor Node in a circuit involves the Nodes public and private key. According to security researchers, the decryption process with private keys (CREATED) is a more computationally expensive task, as it takes 4 times longer to execute in comparison to encryption with public keys (CREATE).[2][6]

3.2.2 EXPLOIT

An attacker can potentially initiate multiple route establishment requests by sending the Tor Nodes CREATE cells at a rate larger than the Tor Nodes processing rate. As the rate of receiving CREATE cells is larger than the Tor Nodes processing rate, the computationally intensive task will lower the bandwidth of the Node and the Node process will eventually drop CREATE cells by relaying DESTROY cells to the client, thus resulting in the client not being able to successfully construct a Tor transmission circuit.[2][6]

3.2.3 MITIGATION

A potential countermeasure to CellFlood attacks can be computational cryptographic puzzles. Similar to the Ethereum blockchains proof-of-stake implementation, a Tor client will have to provide a provide computational proof to the Node in order for the Node to execute commands.

An CellFlood attack primary objective is to essentially take the Tor Node offline using overflow attacks which will cause the Tor Node to automatically delete cells and eventually reject all incoming CREATE requests. [2][6]

The implementation of the cryptographic puzzle essentially allows a single Tor Node to individually mitigate a CellFlood attack against itself. The puzzle countermeasure de-automates the cell deletion and demands that the malicious client will have to solve a computationally intensive cryptographic puzzle in order for the Tor Node to initiate cell deletion. Consequently, the malicious client will have to use more computational power on the puzzle solving process, which will add a computational constraint to the CREATE cell deployment process, disabling or slowing down the attacking malicious client. As the CREATE request overflow has slowed, the Tor Node can process each CREATE request without having to delete or reject the requests.[2][6]

4. Works Cited

[1] "About – Tor Metrics." *Metrics.Torproject.org*, metrics.torproject.org/glossary.html.

[2] Barbera, Marco, et al. *CellFlood: Attacking Tor Onion Routers on the Cheap*.

[3] Bauer, Kevin, et al. *Low-Resource Routing Attacks Against Tor*.

[4] Diego, San. *USENIX Association Proceedings of the 13th USENIX Security Symposium*. 2004.

[5] "New Tor Denial of Service Attacks and Defenses | Tor Blog." *Blog.Torproject.org*, blog.torproject.org/new-tor-denial-service-attacks-and-defenses.

[6] "'One Cell Is Enough to Break Tor's Anonymity' | Tor Blog." *Blog.Torproject.org*, blog.torproject.org/one-cell-enough-break-tors-anonymity.

[7] Pries, Ryan, et al. *A New Replay Attack Against Anonymous Communication Networks*.

[8] Winter, Philipp, et al. *Identifying and Characterizing Sybils in the Tor Network Identifying and Characterizing Sybils in the Tor Network*. 2016.

[9] "Tor: The Second-Generation Onion Router." *Svn-Archive.Torproject.org*, svn-archive.torproject.org/svn/projects/design-paper/tor-design.html.