Mitigation Techniques Against Deanonymization and Denial-of-Service Attacks In The Tor Network

Indrajeet Roy
Department of Electrical and Computer Engineering, Iowa State University
iaroy@iastate.edu

Abstract

The increasing prevalence of extensive internet surveillance has heightened the importance of secure and anonymous internet use. In response, innovative networking solutions like The Onion Router (Tor) have been developed. Tor is a low latency network that allows for anonymous and secure communication of Transmission Control Protocol (TCP) traffic. It operates on the principles of Onion Routing and decentralization. The Tor protocol uses end-to-end encryption to protect TCP streams as they pass through public networks. This encryption involves wrapping the traffic in encrypted units called Tor cells. These cells are encrypted multiple times at different points in the network, ensuring security at each transit stage.However, the decentralized nature of Tor, while vital to its function, also introduces potential risks. Its design allows any internet user to join the network, which could include malicious actors. These individuals could exploit the network by either compromising legitimate nodes or establishing harmful ones. While decentralization is key to Tor's operation, it also opens up possibilities for security vulnerabilities.

Introduction

As large-scale internet surveillance intensifies, the demand for secure and anonymous online communication has surged. This concern has spurred the development of advanced networking technologies, most notably The Onion Router (Tor). Tor is a low-latency network specifically designed to anonymize TCP-traffic communication, building on the principles of Onion Routing and decentralization.

The Tor network facilitates the secure and anonymous transmission of TCP traffic using a protocol that leverages Onion Routing. This method provides end-to-end encryption, effectively anonymizing and securing TCP streams across public networks. It achieves this by encapsulating TCP traffic within Tor cells, which are sequentially encrypted at multiple layers as they traverse a decentralized network of routers.

However, the decentralized architecture that stands as one of Tor's major strengths also introduces substantial challenges. By design, this decentralization permits any internet user to participate in the network, thereby potentially exposing it to security risks. Malicious actors can exploit this openness to compromise the integrity of the network, either by hijacking legitimate nodes or by setting up malicious nodes themselves.

Tor Network Architecture and Operation

1 Tor Network Components

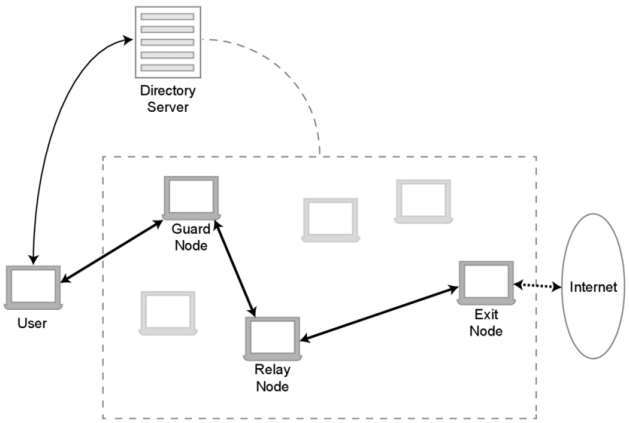

Figure 1: Components of the Tor network

1.1 Tor Cell

Tor cells are fundamental 512-byte transmission structures within the Tor network, comprising a payload and a header. Tor utilizes two primary types of cells: control cells and relay cells.

Control Cell: Control cells facilitate the construction of a transmission circuit across the network. Key commands associated with control cells include:

- CREATE: Establishes a new circuit.
- CREATED: Confirms that the circuit has been successfully established.
- DESTROY: Terminates an existing circuit.

Relay Cell: Relay cells are essential for the end-to-end transmission of TCP traffic through the established transmission circuit. Commands for relay cells include:

- BEGIN: Initiates a TCP connection.
- DATA: Transmits data through the circuit.
- END: Closes a TCP connection.
- SENDME: Requests more data.
- EXTEND: Extends the circuit to include another relay.
- DROP: Drops data from the buffer without transmitting it.

1.2 Tor Circuit

A multi-hop path composed of several Tor nodes, designed to facilitate anonymous, low-latency TCP stream communication from a Tor client to the client's destination within the Tor network. This circuit effectively anonymizes user data by routing it through various nodes, each adding a layer of encryption, thereby ensuring that each node along the path knows only its immediate predecessor and successor.

1.3 Tor Relay/Node

A Tor relay, also known as a node, is a volunteer-operated and publicly listed server that plays a crucial role in forwarding traffic across the Tor network. The Tor network utilizes three types of relays:

Entry/Guard Node: The initial access point for a Tor client into the network. Guard Nodes are unique in that they are the only nodes within the Tor Network where the original IP address of the client can be observed through legitimate, non-malicious network operations.

Middle Node: Serves as the primary traffic forwarding relay within the Tor Network. The majority of the network consists of these Middle Nodes, which encrypt and forward traffic. Middle Nodes only have access to the IP addresses of the directly preceding and succeeding nodes in the relay chain.

Exit Node: Acts as the primary and final exit point of a Tor circuit. Exit Nodes are significant because they are the only nodes where the IP address of the client's destination can be observed. While the final transmission of the payload to the destination IP address occurs in plain text, it can be encrypted using TLS/SSL. Since this final payload transmission appears in plain text, exit nodes are often mistaken as the origin IP address of the traffic, effectively masking the IP address of the Tor client. **Directory Authority Node**: These nodes maintain the overall consensus of the Tor network by regularly updating the list of active nodes. Both Tor clients and other nodes receive continual updates about the state of the network and the Tor Consensus from Directory Authority Nodes, which track the network state by periodically receiving operational updates from all network nodes.

Leveraging the principles of Onion Routing, each node in a Tor circuit can only decrypt enough data to know where to send the data next, without being able to view the entire path or the payload's contents. This selective decryption is facilitated by a shared symmetric key, negotiated between the Tor node and the client during the TLS session that establishes the path or circuit. This design ensures that no single node has access to both the origin and destination of the data, significantly enhancing user privacy and security within the network.

1.4 Tor consensus

The Tor network employs a consensus protocol to maintain the network's stability and security. This consensus is crucial as it represents a regularly updated list of Tor nodes, which includes additions and removals based on current network conditions and node reliability. The consensus is achieved through a cooperative process involving all nine directory authority nodes. These nodes exchange, negotiate, and compile information about node statuses and network conditions from each directory authority. Once they reach agreement, the directory authorities sign off on this compiled data, forming what is known as the Tor consensus. This process ensures that only trustworthy nodes are active at any given time, enhancing the overall security and reliability of the network.

1.5 Tor Consensus weight

Tor Consensus Weight: This value plays a critical role in determining the likelihood that a particular Tor Node will be selected by the Tor Node Selection Algorithm during the construction of a transmission path through the network. The consensus weight of a node is primarily determined by its bandwidth capacity. Nodes with higher bandwidth capacities are assigned larger consensus weights by the Directory Authority Node. This mechanism ensures that nodes with greater capability to handle traffic efficiently are more frequently involved in the network routing, thereby optimizing the speed and reliability of the Tor network.

2 Tor Network Operation

2.1 Circuit Creation

The initiation of data transmission through the Tor network begins with the establishment of a secure circuit, spearheaded by the Tor client. This intricate process involves key components such as the Directory Authority nodes and the Tor Node Selection Algorithm.

Tor Node Selection Algorithm

This algorithm predominantly selects nodes based on their bandwidth capacity, as nodes with higher bandwidth are more likely to handle data efficiently and are thus preferred for inclusion in the circuit. However, the selection is not based solely on bandwidth. Other critical factors include:

- **Diversity in Network**: To mitigate risks such as surveillance and network bottlenecks, the algorithm ensures a diverse selection of nodes across different network families and geographic locations.
- **Entry Guards**: The Tor client designates a select few stable and trusted nodes as entry guards. These

guards are used consistently across multiple circuits to reduce the risk of interaction with compromised or malicious entry nodes.

- **Exit Policies**: The choice of exit nodes is influenced by their respective policies on traffic types. The algorithm selects nodes whose policies align with the user's specific traffic requirements to ensure the successful exit of data from the network.

Consensus Verification
Before finalizing the selection, the Tor client consults the latest Tor consensus. This vital document, updated and distributed by Directory Authority nodes, contains comprehensive information about each node's current status (active or down), designated flags (such as Guard, Exit, Stable), and performance metrics. By referencing the Tor consensus, the client ensures that all selected nodes are not only operational but also trustworthy at the time of circuit creation.

This methodical approach to circuit creation is foundational to maintaining the anonymity, security, and efficiency of the Tor network, safeguarding user data as it navigates through the complex web of nodes.

2.2 Circuit Encryption

The encryption process within a Tor circuit employs an Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) key exchange, which occurs incrementally across the transmission circuit. As the data traverses each node within the circuit, a TLS session is negotiated between the Tor client and the node, facilitating the establishment of a shared secret key. This method ensures that each segment of the path between nodes is secured independently, reinforcing the overall integrity and confidentiality of the data transmission. This layered encryption strategy is fundamental in preserving user anonymity and protecting the data from potential interceptors at any point within the network.

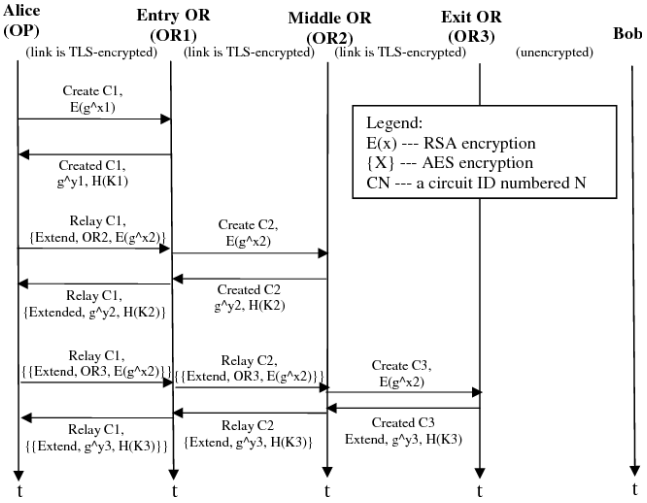

Figure 2: Tor network Circuit encryption

The encryption process within the Tor network begins with the initial key negotiation between the Tor client (referred to as Alice) and the Entry Node (OR1). Alice initiates the process by sending a **CREATE cell (C1)** to **OR1**, which contains the payload $\mathbf{E}(\mathbf{g^{x1}})$. In this expression, $\mathbf{g^{x1}}$ represents the first component of a Diffie-Hellman (DH) handshake, and **E**() denotes RSA encryption using the Entry Node's public onion key. Upon receipt, **OR1** decrypts the RSA encryption with its private key and responds with a **CREATED cell (C1)** that includes the second component of the DH handshake $\mathbf{g^{y1}}$ and a hash of the final key **H(K1)**. This exchange establishes a shared key between Alice and **OR1**, enabling secure data encryption and decryption during transmission.The subsequent key negotiation occurs between Alice and a Middle Node (OR2).

Alice sends a **RELAY EXTEND (C1)** cell to **OR1**, which includes the address of **OR2** and the payload $\mathbf{E}(\mathbf{g^{x2}})$. Here, $\mathbf{g^{x2}}$ is the initial part of another DH handshake, and **E**() signifies RSA encryption with **OR2**'s public onion key. **OR1** extracts the information, forwards it to **OR2** in a **CREATE cell (C2)**, and **OR2** responds with a **CREATED cell (C2)** that includes $\mathbf{g^{y2}}$ and a hash of the final key **H(K2)** after decrypting the RSA encryption with its private key. Once **OR1** receives **OR2**'s **CREATED cell**, it relays this information back to Alice in a **RELAY_EXTENDED cell (C1)**. This final step completes the establishment of a shared key between Alice and **OR2**, which is then used for subsequent data encryption and decryption during their communication.

2.3 Circuit Transmission

The data transmission process within a Tor circuit employs Relay DATA cells. These cells are used by the client to encapsulate the TCP data intended for transmission. Each Relay DATA cell is securely encrypted with a shared symmetric key that has been previously negotiated between the client and each Node in the circuit. As the data traverses the circuit, each Node along the path decrypts the Relay DATA cell using its respective shared key. This decryption process is essential for the node to read and then re-encrypt the data for the next hop in the circuit. This method ensures that the data remains secure from end to end, even as it passes through multiple nodes, each of which only has access to enough information to decrypt and forward the data without knowing its origin or final destination.

Vulnerabilities, Exploits and Mitigation Techniques

1 Sybill Attack

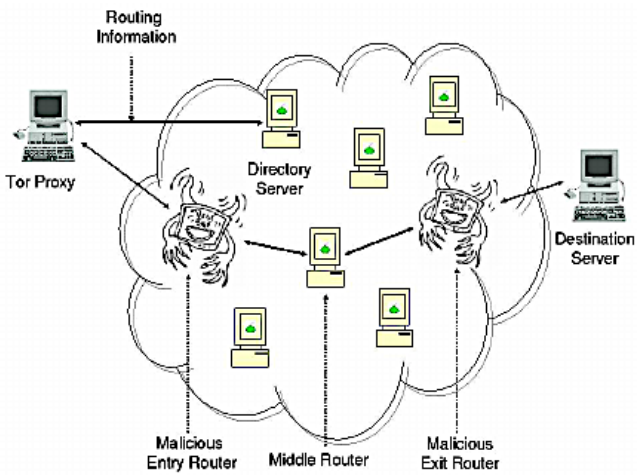

Figure 3: Sybil Network Attack

1.1 Description

A Sybil attack is a significant threat to decentralized networks, exploiting the fundamental principle that any individual can contribute nodes to the network. This type of attack occurs when an attacker gains a disproportionate influence within the network by creating a large number of pseudonymous entities—either by hijacking legitimate machines or deploying a multitude of bots or malicious nodes.

The attacker's goal is to manipulate network consensus, which is crucial for maintaining the integrity and functionality of operations within decentralized systems. By controlling a substantial portion of the network, the attacker can influence decisions that are supposed to be made collectively by the network, potentially leading to fraudulent transactions, double spending, or censorship.

Such attacks undermine the trust and security principles that decentralized networks are built upon, making it crucial to implement robust mechanisms to detect and mitigate the presence of Sybil nodes.

1.2 Vulnerability in Tor Node Selection Process

In the Tor network, the initial step for a Tor client to initiate encrypted TCP stream transmission is to construct a transmission circuit using the Tor Node Selection Algorithm. This algorithm is designed to optimize the circuit for efficient transmission by selecting nodes based on their bandwidth. The nodes are chosen with reference to the Tor consensus, which is periodically updated by the Tor Directory Authority Nodes.

The consensus weight of a Tor Node is directly correlated with the node's bandwidth. Nodes with higher bandwidth have a higher consensus weight, increasing their likelihood of being selected by the Tor Node Selection Algorithm. This results in nodes with higher consensus weights handling more TCP traffic across the network. While this mechanism enhances network efficiency, it also introduces a vulnerability: a node with artificially inflated bandwidth can gain disproportionate influence over network traffic. This can potentially expose the network to risks such as traffic analysis, increased latency due to overloading, and targeted attacks by malicious entities aiming to compromise the integrity and anonymity that Tor aims to provide.

1.3 Sybil Attack Vector

An attacker can manipulate the consensus weight of their nodes to control and observe more traffic across the Tor network. This manipulation can be achieved by setting up high bandwidth, high uptime Sybil nodes or by artificially increasing the bandwidth of a single node. Although a single Sybil Node with unusually high traffic and large bandwidth might alert the Tor Directory Authority Nodes, potentially leading to its removal from the Tor consensus, the attacker might also set up multiple Sybil Nodes that share a single IP address.

1.3.1 Attack Vectors by Network Point of Entry:

Entry/Guard Node

- **Man-In-The-Middle Attacks**: A malicious Sybil Node serving as an Entry/Guard Node could intercept or tamper with data. Since these nodes are the first point of entry into the network, they can manipulate or eavesdrop on the data right at the beginning of the Tor circuit.

Middle Node

- **Website Fingerprinting**: Website Fingerprinting allows the attacker to deduce the specific destination IP address from the source IP address, thus deanonymizing the client's activity. While Middle Nodes in a Tor circuit do not see the encrypted TCP stream of the entire Tor circuit, an attacker might exploit packet instructions and transmitted information via the node to correlate the origin and destination IP addresses.

- **Bridge Address Harvesting**: Bridge Address Harvesting targets the identification of Bridge nodes within the Tor network. Unlike publicly listed Tor Nodes, Bridge Nodes are private and not easily identifiable, providing an added level of anonymity especially when publicly listed nodes are offline or monitored. The attacker can use reverse referencing of the source IP address of observed TCP packets against publicly listed Tor Node IP addresses. If the incoming IP address does not match any listed Node addresses, the connection might be from a Bridge Node, thus identifying it.

Exit Node

- **Traffic Interception**: If a malicious node operates as an Exit Node, it could intercept the client's plaintext traffic after the final onion 'delayering' or decryption process with the client-shared negotiated key. This compromises both the integrity and authenticity of the transmitted payload, exposing the user to potential fraud or data theft.

1.4 Sybil Attack Mitigation Strategies

While the decentralized architecture of the Tor network inherently allows for any individual to contribute nodes, this flexibility also introduces vulnerabilities such as Sybil attacks. Mitigating these attacks does not entail altering the decentralized nature of the network but rather improving the Directory Authority Nodes mechanisms for node addition and selection.

1.4.1 Directory Authority Nodes Oversight

Active Monitoring: Tor Directory Authority Nodes continuously monitor the network, decisively adding or removing nodes from the Tor consensus based on their observed behavior and trustworthiness.

IP Address Restrictions: Recent versions of Tor have implemented stricter IP address restrictions to counteract Sybil Nodes. These measures include allowing only up to three Tor Nodes per single public IP Address, which helps prevent a single actor from controlling too many nodes from the same IP and thereby gaining disproportionate influence in the network.

2 Cellflood Attack

2.1 Description

A Cell Flood attack represents a specific type of Denial-of-Service (DoS) attack targeting the Tor network. This attack disrupts the normal operation by preventing legitimate Tor users from constructing transmission circuits, essential for data transmission through the network.

During a Cell Flood attack, a malicious client bombards Tor Nodes with computationally intensive circuit creation requests (Control CREATE cell). This flood of requests strains the processing capabilities of the nodes, leading to a degradation in their performance. Consequently, the affected Tor Nodes begin rejecting and dropping all circuit creation commands, effectively denying service to legitimate Tor clients.

As the overwhelmed nodes become incapacitated, they are likely to be removed from the Tor consensus by the Tor Directory Authority Nodes. This removal reduces the number of active nodes within the network, which can further destabilize the network's functionality, potentially diminishing user numbers and compromising both the stability and anonymity of the network.

2.2 Vulnerability in Tor Node Encryption

The security of a Tor Node circuit relies on the use of public and private keys for encryption and decryption processes, respectively. Security research indicates a significant disparity in computational effort between these two processes. Specifically, the decryption process using private keys is markedly more computationally intensive—it takes approximately four times longer to execute compared to encryption with public keys. This discrepancy not only introduces a potential bottleneck but also presents a vulnerability, as it can significantly slow down the network under heavy load conditions, making the system more susceptible to timing attacks and potentially impacting overall network performance and security.

2.3 Cellflood Attack Vector

An attacker can exploit a vulnerability in the Tor network's route establishment process by initiating multiple circuit construction requests at a rate exceeding the processing capabilities of the Tor Nodes. Specifically, the attacker sends Control CREATE cell requests more rapidly than the nodes can handle. This overwhelming flow of requests, being computationally intensive, strains the node's resources, leading to a significant reduction in available bandwidth and processing power. Consequently, the overloaded nodes respond by issuing Control DESTROY cells to the offending client to halt the circuit construction process. This action effectively disrupts the client's ability to successfully establish a transmission circuit within the Tor network, resulting in a denial-of-service scenario.

2.4 Cellflood Attack Mitigation Strategies

CellFlood attacks aim to incapacitate Tor Nodes by overwhelming them with excessive CREATE cell requests, leading to automatic deletion of cells and eventual rejection of all incoming requests. By requiring a cryptographic solution before processing, this strategy significantly raises the resource cost for the attacker.

2.4.1 Cryptographic Puzzles Against CellFlood Attacks

A viable countermeasure against CellFlood attacks on the Tor network involves the implementation of computational cryptographic puzzles, similar to mechanisms seen in blockchain technologies like Ethereum's proof-of-stake. In this approach, a Tor client must solve a cryptographic puzzle before its CREATE cell request is processed by the node. This requirement is designed to validate the legitimacy and intent of the client, preventing automated flood attacks.The introduction of a cryptographic puzzle places a computational burden on the client, necessitating significant computational resources to solve the puzzle before a node will execute deletion commands. This additional step mitigates the attack by:

- Demanding computational work from the attacker, thereby slowing down the rate of request submissions.
- Allowing the node to manage incoming CREATE cell requests effectively without the need to reject or delete them prematurely.

This method not only protects individual nodes from being taken offline by distributed denial-of-service attacks but also preserves the overall integrity and functionality of the Tor network by maintaining node operability and response capacity.

References

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