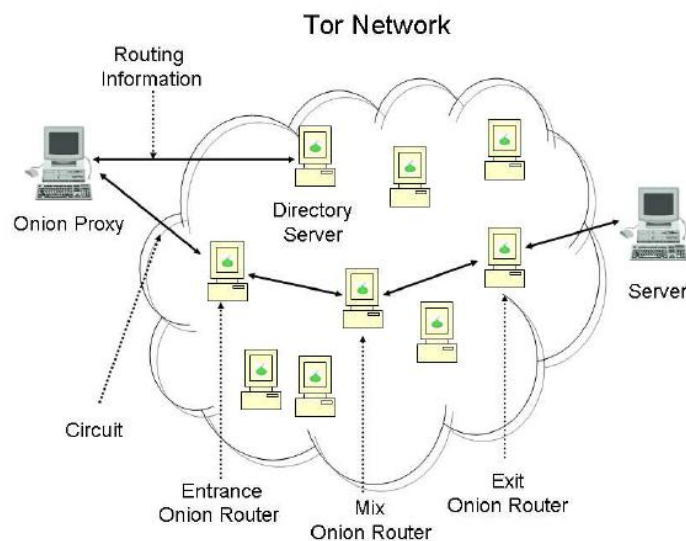


Low-Resource Routing Attacks Against Anonymous Systems

HY558- Roman Čížek

This technical report deals with attack to mix-networks by using low resource nodes. Authors using like an example attack against Tor network.

Tor network



Main goal of Tor project is protect the privacy of TCP connections. Tor aims to provide end-user anonymity with constraints such as low-latency, deployability, usability, flexibility and simple design. Tor can anonymize TCP streams, providing a high-throughput and low-latency mix network.

Fundamental concepts:

Onion router (OR) server component, responsible for forwarding traffic through the mix network. *Onion proxy* (OP) client part of the network, injects user's traffic into the network. A *circuit* is a path of three onion routers (by default). First router is called *entrance router*, second *mix router* and the last is *exit router*. Router information is distributed by set of well-known and trusted *directory servers*. Unit of transmission is called *cell*, which is a fixed-size 512 byte packet.

The Tor also uses Transport-Layer Security (TLS) and other standard cryptographic methods to prevent an eavesdropper from linking incoming cells to outgoing cells. Originator of the circuit encrypts the cell using a layered encrypting scheme. Each hop of the circuit removes a

layer until the cell reaches the exit node. Only final node can see fully decrypted message. This process is called onion routing.

Tor selection algorithm has two steps. The first step is select the entrance router and second is select subsequent routers in the circuit. Entrance selection is automatic selection of set (default 3) routers which are marked by trusted directory server as “fast” and “stable”. Fast router means that reports bandwidth above the median of all bandwidth advertisements. The same is for stable but it advertises an uptime instead of bandwidth. The client will choose new set of entry routers only when one is unreachable.

Non-entrance router selection is intended to optimize onion router selection for bandwidth and uptime, while not always choosing the very best nodes every time. It ensures that all nodes in the system are used to some extent, but nodes with more bandwidth are used most often.

Tor has a set of TCP ports that are designed as “long-lived”. If they are chosen than the path is optimized for stability. There is also algorithm for optimize path for bandwidth. The most significant feature of this algorithm is that the more bandwidth a particular router advertise, the greater probability that the router is chosen.

Tor’s threat model – attacker can control or monitor a subset of network. Attacker can also inject, delay, alter or drop the traffic along some of the links. There is theoretical analysis from Tor’s designers for threat for this network. This analysis is based on a combinatorial model that assumes nodes are chosen at random from a uniform distribution. It is possible to correlate the traffic flow with only entrance and exit nodes.

Compromising Anonymity:

In their basic attack, they have access to a non-global set of malicious nodes and they used fast and stable nodes, as characterized by high bandwidths and high uptimes. After that they remove these restrictions by using low-resource nodes, but they report incorrect to the directory servers their uptime and bandwidth. Trusted servers didn’t verify these values, so this false advertises will remain undetected.

Phase one: setting up

To mount their attack, they have to control a subset of nodes in the pool of active routers. An adversary can put new nodes into the Tor network or compromise existing nodes.

When we have these nodes in Tor network and new client joins and initiates a flow the corresponding onion proxy attempts to optimize its path by choosing fast and stable proxy servers, because one of Tor’s goals is to provide a low-latency service. By deploying more nodes into the network the adversary can increase the probability that its nodes are chosen as both entry and exit nodes for new client’s circuit. Compromising the entrance and exit

position of a path is necessary condition in order for the second phase of their attack to successfully correlate traffic.

Phase two: Linking paths

They show that if the full path has been populated with malicious nodes, it is trivial to compromise the anonymity of the path. In many cases we have only entrance and exit nodes so they developed technique that allows this path to be compromised with a high-probability.

Each malicious router logs these information for each cell received: (1) location on the circuit's path, (2) local timestamp, (3) previous circuit ID, (4) previous IP address, (5) previous connection's port, (6) next hop's IP address, (7) next hop's port, (8) next hop's circuit ID. With this information, an attacker can associate the sender with the receiver. In order to execute this algorithm, the malicious nodes must be coordinated. The simplest approach is to use a centralized authority to which all nodes report their logs. The centralized authority can then execute the circuit-linking algorithm in real-time.

Tor's circuit building algorithm sends a deterministic number of packets in an easily recognizable pattern. A proxy creates a new circuit through Tor as follows: First, the proxy issues a circuit building request to its chosen entrance router and the entrance router sends an acknowledgment. Next, the proxy sends another circuit building request to the entrance router to extend the circuit through a chosen mix router. The mix router acknowledges the new circuit by sending an acknowledgment back to the client via the entrance node. Finally, the proxy sends a circuit request to the exit node, which is forwarded through the entrance and mix routers to the chosen exit server. Once the exit router's acknowledgment has been received through the mix and entrance nodes, the circuit has been successfully built.

Their circuit building algorithm works as follows: The entrance node verifies that the circuit request is originated from an onion proxy, not a router. This is easily determined since there will be no routing advertisements for this node at the trusted directory servers. Next, the algorithm ensures that steps 1, 2 and 3 occurs in increasing chronological order. Also it is necessary to verify that the next hop for an entrance node is the same as the previous hop of the exit node. Finally, in step 3, it is verified that the cell headed towards the exit node from the entrance node is received before the reply from the exit node. If every step in the algorithm is satisfied, then the circuit has been compromised.

Authors did this attack for the Tor network in laboratory. They created two isolated Tor networks on PlanetLab, consisting 40 and 60 nodes, each running one onion router per node. Each experimental deployment has precisely three directory servers. They also ensure that their network is as realistic as possible. They used 2 and 4 malicious nodes for 40 nodes network and 3 and 6 malicious nodes for 60 nodes network. Authors start network without malicious nodes and generates traffic for two hours to get the network to stable state, after

that time they add malicious nodes and generates traffic another two hours to get the results.

In the case (6 malicious nodes and 60 nodes) the results were that adversary could compromise over 46% of the paths in the network. This is in stark contrast to the 70% of paths predicted by the previous analytical model.

They also consider another further attacks and improvements like:

- Compromising existing clients - using DoS attack to compel nodes to rebuilt path and have probability that he choose our nodes.
- Selective path disruption – block paths where we have only one malicious node to compel rebuilt path.
- Displacing honest entry nodes - by flooding network with a lot of malicious nodes.
- Compromising only the entrance node.

They also propose defense against these attacks. Naïve solutions:

- Verify uptime – uptime could be tested by periodical sending messages
- Centralized bandwidth verification – directory servers will monitor bandwidth of nodes
- Distributed bandwidth verification – nodes could proactive monitor each other

Their defense solution:

Local observation based reputation system: They propose reputation system similar like in peer-to-peer networks. They are using reputation vector and to this vector is increased or decreased by value depends on positive or negative feedback. And after some time we can recognize malicious nodes because they will have small reputation vector because they couldn't provide their advertised bandwidth.

Distributed reputation system: there is some improvement, because local reputation system takes a long time to converge to the set of malicious nodes. So in this case each node computes his reputation vector and provides it to the centralized directory server. But they are using some pseudonym for this vector to prevent compromises anonymity.

Preventing Sybil attacks by allowing only one router from one IP address and use some another metrics instead of median.

And the conclusion is that it is very effective attack instead of analytical expectations. The main problem is that the directory servers didn't verify advertised bandwidth and uptime. Authors purpose some another attacks and some possibilities how to protect against these attacks.