582615: Overlay and peer-to-peer Networks, Autumn 2011

Exercise 2
Example solutions

Questions

1. a) Explain onion routing. (3 points)

Onion routing is a method for anonymizing network traffic with the help of overlay technologies. In onion routing, data packets are recursively encrypted by the sender and source-routed to the final destination through intermediate nodes called onion routers. Each onion router along the path removes a layer of encryption from the packet as if peeling an onion, also revealing the next-hop destination address of the packet, and forwards the packet to the next router along the path. After passing the last onion router along the chosen path, all the onion-related encryption has been removed, and the final destination only sees normal TCP/IP traffic.

Onion routing can be implemented using either public key cryptography or symmetric key cryptography. In the former case, the originator of a packet chooses a path and recursively encrypts the packet with the public keys of the onion routers along the chosen path. In the latter case the sender chooses a circuit along which the packets will be transmitted, and performs a symmetric key-exchange with the onion routers along the circuit before encrypting and sending the packets.

b) Explain how Tor implements onion routing. (3 points)

Tor implements onion routing for TCP streams. A client, who wishes to use the Tor overlay, contacts an onion proxy using SOCKS protocol. The proxy constructs a circuit on the overlay and requests one of the onion routers on the circuit to create a normal TCP connection to the actual TCP server. Data are then forwarded between the client and the server through the proxy and the routers on the circuit. The details of onion routing are hidden from the client by the proxy.

Each onion router of the Tor overlay network maintains a TLS connection to every other onion router. Circuits are multiplexed over these connections. TLS encryption is also used by onion proxies to connect to the overlay. Data are forwarded on the connections in packets of constant size that are called “cells”. Onion proxies obtain the list of onion routers from a directory server. Directory servers are well-known onion routers that track the state of the overlay. They collect status announcements from other onion routers and generate a combined list of available routers and their properties.

Tor circuits are constructed incrementally. In order to create a circuit, an onion proxy connects the first onion router on the path that is has chosen by sending a “create” message. Then the proxy and the router negotiate a symmetric key for the hop by using Diffie-Hellman key exchange. After this the proxy adds N more hops to the circuit. The proxy achieves this by iteratively sending a “relay extend” message to the router that is currently the last one in the circuit. The router reacts to this by sending a “create” message to the router specified in the parameters of the “relay extend” message that it received from the proxy.

When a client requests its onion proxy to set up a TCP connection to a TCP server, the proxy sends a “relay begin” message through the circuit to the onion router on the circuit that the proxy has chosen to be the exit node for this connection. This opens a stream through the circuit between the proxy and the exit node. The exit node connects this stream using a normal TCP connection to the TCP server that the client wanted to connect to. The stream is later used for forwarding the actual TCP data. Multiple TCP streams can be multiplexed over one single circuit in order to save the effort of creating new circuits .

2. a) What are the design goals of Freenet 0.7.5? (3 points)
Among the design goals of Freenet 0.7.5 are the following.

1. It should be difficult to determine whether a given node is part of the Freenet network.
2. It should be difficult to remove data.
3. It should be difficult to determine which nodes store which data.
4. The system should be able to cope well with daily churn and even malicious attacks.
5. The privacy of both publishers and readers has to be protected from outsiders and from other participants.

b) Explain how Freenet 0.7.5 works to reach these goals? (3 points)

1. Freenet 0.7.5 introduces a Darknet mode where each node connects to trusted nodes only. The trust relationship has to be established manually. The resulting network's topology is more or less static. The actual IP address of a node is known to its neighbours alone. Consequently, it is difficult for an outsider, or even non-neighbour nodes, to determine whether a Freenet node is operated at a given IP address.

2. and 3. Data are encrypted so them cannot be recognized easily without the decryption keys. Not even a node operator knows for certain which data are hosted at the node. In addition, data are cached aggressively at multiple locations. Although the key for a piece of data determines locally a preferred identifier of a node where to store the data, the actual storage location is not strictly bounded. Besides, the node identifiers are random and hence they are decoupled from the actual IP addresses.

4. Aggressive caching attempts to keep data available even when some of the nodes are down. The search for a piece of data uses backtracking. When a dead-end is reached, the search attempts another route. In the Darknet mode, the IP address of a node is known only to its neighbours. As a result, it is hard to launch a directed attack against a particular node. The requirement of trust-relationships between nodes makes it hard for an attacker to infiltrate the network through multiple identities.

5. The origin of data insertions and retrievals become concealed once the messages are forwarded through multiple hops. Any reply messages are forwarded by using the reverse path. The originating node is not contacted directly. In addition, publishers are protected by the fact that the network stores the data. Data caching helps to protect readers since it limits the span of requests of popular items. However, the neighbours of a node are able to see the messages that the node sends and receives.