

Designing a secure file sharing system

by: Stelios Tigkas, 07/07/2005

<http://www.securitydocs.com/library/3443>

Peer to peer systems have gained tremendous popularity over the last few years, partly due to the unimaginable success of the Napster [1] file sharing system. This phenomenon initiated a new era of computing, which included the development and deployment of many similarly designed systems, targeting different types of usage.

Despite the fact that people have associated the term 'p2p' with file sharing of music files, there is much more to p2p. A large number of legitimate applications are feasible, including publish-subscribe middleware, distributed storage and data backup services. As a proof of 'legitimate' concept, this article attempts to provide the infrastructure for a secure, semi-centralised file-sharing system. Some centralisation, in one form or another, is necessary to achieve security. In our model, we will assign critical tasks to trusted components.

System goals

The system proposed here, in a high-level approach, could be useful in a distributed environment for the purpose of file/documentation exchange. Suppose that a large number of programmers are working on a project at several places around the world. The project in question has several confidential aspects to it, and it is desirable that the programmers can exchange information & portions of code on the project according to a 'privilege' system. Let us further assume that the participants are classified into three categories: A (Unclassified), B (Classified) & C (Secret). C Class participants should be able to retrieve content from all three types of Users (A, B & C); B Class should be able to retrieve A & B type content, and unclassified users should only be able to access unclassified information. The system proposed will attempt to address this need by means of 'visibility' of the network. Diagrammatically, the visibility of other nodes should be as follows:

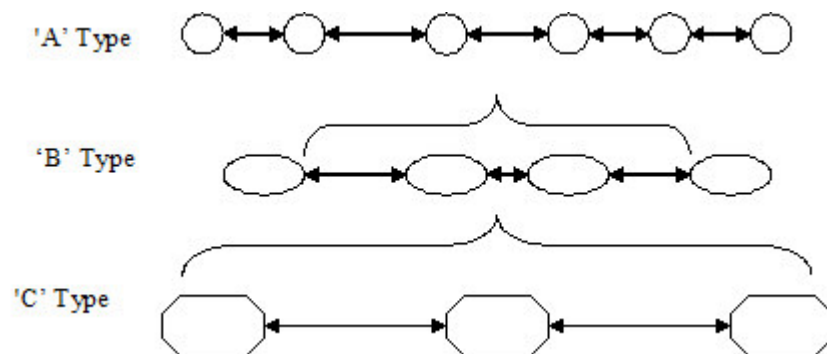


Figure 1: This is a simplified version of the participants' visibility of the network. Powerful C nodes should see and be able to retrieve all content. B nodes can only see their peers, and 'A' nodes. 'A' nodes can only see their peers.

Joining the network

In order to achieve the previously mentioned goals, a Certification authority (CA) has to undertake some critical tasks in our system. Each node wishing to join the network first contacts the CA to acquire a certificate. Suppose that a 'C' node wishes to join. Prior to granting the certificate, the CA performs all the necessary checks to define the new node's Classification (A, B or C) before proceeding. Upon completion, a certificate is granted to the new node, including a NodeID, possibly computed as the XOR of the node's IP address and his/her Classification. The certificate also includes the IP address, Classification, Node's Public key, and possibly a Time stamp, all digitally signed by the CA. The IP address of a special, trusted bootstrap node is then passed on to the newcomer. The bootstrap node does not participate in the file swapping service. It only serves as a repository for IP addresses and Classifications for existing nodes. The new node is now ready to join the network, by routing a JOIN request to the bootstrap node. For our system to work, certificate checking is performed prior to the new node

being accepted. Once the bootstrap node checks the certificate and ensures that the newcomer is classified as a 'C' node, the CA's signature is verified, and it sends the new node's details to all the C nodes it is aware of. All the C nodes update their routing tables to include the newcomer, and the new node is now an active participant of the network. All that the new node needs to do is copy some initial routing information about all types of nodes and key association tables from the bootstrap node.

A similar process is followed for 'B' and 'A' type nodes. When a 'B' type node is introduced in the network, the bootstrap node has to notify both C and B nodes to update their routing tables, but should carefully provide 'B' and 'A' type routing information to the newcomer. Lastly, when an 'A' type node joins, the bootstrap node performs the necessary checks and notifies all existing participants, serving 'A' nodes' routing information to the newly arrived peer. If this scheme works as planned, it results in the desirable 'selective' visibility of the network, according to the peer's privilege level.

Key association & Locating content

A problem with this hierarchical p2p system could be the association of keys to nodes. An unclassified document, for instance, should only be mapped to an 'A' type node and a secret, 'C' type portion of code should only be mapped to a 'C' type node. To achieve that, our system needs some restrictions in its key association rules. If we choose to associate keys with NodeIDs according to numerical closeness, we need to ensure that key roots are chosen correctly. A simple way to achieve this is to use different Node Identifier spaces for the three different privilege levels supported. Each privilege level would be assigned a node Identifier from its own private pool, and the key Identifiers would be chosen from the same IDspace. By that means, our system ensures that keys are only associated with nodes of the same privilege level.

Pastry [2] is an efficient overlay, and its routing algorithm and general properties could be suitable for our purposes. To retrieve content, nodes should first check if they have ownership of the key being searched. If not, they would consult their routing table to track a node that has a greater numerical closeness with the key and forward the request appropriately. In our system, though, the basic routing algorithm has to perform a Certificate-status check prior to delivering information back to the requestor. If a cunning 'A' type peer requests information from a 'C' node, the latter should drop the request. This is to better safeguard the security of the system in cases that there has been information leakage about 'C' nodes' locations in the network.

Temporary downgrades

Highly privileged peers might occasionally need some basic 'A' type documents or code. A subtle way to achieve this goal without raising suspicions to unclassified peers is to provide a 'temporary downgrade' mechanism in the Nodes' certificates. This idea is inspired from the Bell La Padula [3] Access Control model. A 'C' node wishing to retrieve content rated Unclassified, should temporarily downgrade to 'A' level prior to performing its search. As a 'C' node, it already keeps all the necessary routing information on all types of nodes so tracking the desirable content will be simple. As an extra layer of security, in parallel with the Class downgrade mechanism, the certificate should be modified with respect to the nodeID. Because we mentioned that different classifications will choose their node IDs from different NodeID spaces, an unclassified node contacted from an allegedly A class peer with C class nodeID, would be suspicious. Applying temporary A (or B) type NodeIDs would solve this problem.

However, it is not practical to modify a certificate once the Certification Authority has signed it. Rather than contacting the CA each time a highly privileged node needs to access unclassified content, the CA could provide 'C' and 'B' class nodes with 'A' class certificates in the initial phase of joining the network.

Downloading content

Once content is located, documents need to be transferred to the requesting peer. For 'A' type nodes, encryption is not necessary, as their material is not of sensitive nature. For 'B' and 'C' type nodes, however, it is advisable to use encryption

when exchanging information over the network. AN SSL-like mechanism could be used to achieve this goal. In a Client – Server environment, a typical Secure Sockets Layer session includes several steps; initially, the client receives & validates the Server's public key, then session keys are created & encrypted under the Server's public key, and used to achieve the secure communication. This technique can be used in our p2p environment. The peer serving the content can be visualised as the Server, the requestor as the client. The Serving peer's public key is provably genuine, if signed by the CA.

Evaluation & Enhancements

Our system has two strong elements, which can also be regarded as its weaknesses, depending on the reader's perspective. These are the Certification Authority and the Bootstrap node. It cannot be stressed enough how crucial the functions of those two entities are for the security and overall performance of the system. Inadequate checking of node credentials could result in an unclassified node acquiring a 'C' certificate, with all the consequences that may have. The bootstrap node should also be very cautious when checking new nodes' classifications to include in the network, and when providing routing updates, to avoid information leakage. The bootstrap node is vulnerable to DoS attacks, since its IP address is not hidden. To ensure the system remains functional, additional bootstrap mirror nodes should be available, and used in cases that the main node is down. It is possible to detect and prevent DoS attacks underway in many cases, using techniques such as attack signatures, acceptable limits of bandwidth usage, or filtering of inbound traffic according to several rules. This is a broad topic, and for a more detailed coverage, the reader is referred to [4].

The model we have envisaged is far from perfect; however, we hope that it is a vivid example in the path of 'harnessing the power of disruptive technologies', and a sound proof that there is no such thing as bad technology, only misused technology.

References

1. Napster, www.Napster.com (2000)
2. Antony Rowstron & Peter Druschel: Pastry, Scalable Decentralised Object Location and routing for large-scale peer-to-peer systems, in proc. 18th IFIP/ACM Middleware, Heidelberg Germany - research.microsoft.com/~antr/PAST/pastry.pdf (Nov 2001)
3. Deiter Gollman – 'Computer Security' (Wiley) , pp 47-51 'The Bell La Padula model' (1999)
4. Ahsan Habib, Mohamed Hefeeda & Bharat Bargava: Detecting Service Violations & DoS attacks, in proc. 10th Annual Network and Distributed System Security Symposium , San Diego, CA - www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/12.pdf (Feb 2003)