Securing Peer-to-Peer Networks

Christopher Piraino
piraino.chris@gmail.com
Ming Chow

Abstract

By their nature, Peer-to-Peer applications are significantly harder to secure than their counterparts, such as the client/server model. Since every node within a P2P network acts as both a client and server, having malicious nodes within the network presents far greater a risk than a malicious client does. Likewise, the distributed nature of P2P networks allows malicious nodes much easier--and harder to detect--entry into the network. P2P networks do offer a number of significant advantages over the usual central authority paradigm, notably greater anonymity, greater scalability, and greater resilience. These qualities are of interest to many companies and developers, and a wide array of research has been conducted on how to most effectively secure P2P applications and networks. This paper will be an overview of both P2P networks and the various security flaws and defenses, with the ultimate goal being a summary of the relative security of P2P networks and protocols, and their usefulness in today's security-conscious world.

1.0 - Introduction

The essence of a P2P network is the fact that nodes within the network act as both consumers and suppliers of resources within the network, creating a decentralized and distributed network topology that has significant advantages over the normal client-server network model. First popularized through the music-sharing application Napster, P2P software has always been of interest to many people, especially those in the grey area of legality.

The P2P network is inherently scalable, since each new node that joins the network also adds its own resources to the network, while in the client-server model each new client adds additional load onto the server without increasing the amount of resources available. This also allows the P2P network to have greater resilience than the client-server network. If the server in the client-server model goes down for whatever reason, the whole network goes down. However, in a P2P network, if a node within the network goes down, the remaining nodes are able to compensate for its loss. A P2P network does not have any central authorizing agent, decreasing knowledge of a node's existence to only a small subset of the network that the node is connected to. This property is of particular importance to privacy-minded applications, such as criminal botnets, as the compromising of one node within the network will not result in the entire network being compromised as well. The P2P architecture does lead to some serious security vulnerabilities however, such as routing and storage attacks, and a fundamental issue with P2P networks is the issue of whether to trust other nodes in the network.

2.0 - Importance to the Community

Popularized by file-sharing networks, P2P networks are mostly seen as a tool for the criminal- or privacy-minded individual. According to a study in 2009 done by Ipoque, a leading European Internet inspection management company, P2P protocols account for a majority of the web traffic throughout the world, with BitTorrent being the protocol most used [1]. These file-sharing applications are not the only place where P2P networks are useful, however. For privacy-minded individuals, projects like the FreeNet are an answer, providing anonymous communication between peers, all through the use of free software. The fact that P2P networks increase in resource size the more nodes connected has also drawn the attention of industry. Spotify uses a hybrid P2P model to serve content to its extensive network of users, and P2P protocols are also starting to be used in distributed computing, where the concept of independent nodes within a network works extremely well. On another front, knowledge of P2P vulnerabilities will enable authorities to infiltrate P2P botnets, notoriously one of the most difficult botnet topologies to take down. The future holds a great deal of promise for P2P applications, but in order to get there a number of serious security vulnerabilities within P2P networks must be addressed.

3.0 - What Are Peer-To-Peer Networks?

The security issues that P2P networks face result from some of the fundamental goals and assumptions used in the design and implementation of the protocols, and thus it is necessary to delve deep into the technical details underlying these protocols. In general, there are two different types of P2P networks, structured and unstructured. The difference between the two is obvious, the unstructured network adds peers randomly to the network while the structured network adds peers to specific places according to the protocol used. For this paper, the focus will be on structured networks, as those networks provide the best performance and search capabilities. These structured networks are built upon a concept called a Distributed Hash Table, DHT, which is exactly what it sounds like, a hash table distributed throughout the network. A DHT has two main components, the distance function used to compute the distance between two keys/nodes and the network's topology which determine how nodes are connected to each other [2]. The way these two features are implemented lead directly to the P2P network's main strengths: scalability, fault-tolerance, and decentralization [2].

3.1 - Kademlia

Kademlia is a DHT implementation created by Petar Maymounkov and David Mazières in 2002. Kademlia has been chosen as the DHT implementation to look at for this paper because of a number of properties, the most important of which is the fact that Kademlia is one of the more secure DHT implementations because of certain design decisions that will become apparent soon. Kademlia uses as its distance metric the XOR function because it upholds three important properties: the distance between a node and itself is zero, the distance from A to B is the same as from B to A, and the distance from A to B is less than or equal to the distance from A to C plus the distance from C to B (the triangle inequality) [3]. In addition, the XOR metric allows for significant increases in the speed of calculation of the distance between nodes, which is carried out exceedingly often throughout the protocol.

For its network topology, Kademlia employs a strategy designed to ensure that each node is able to contact every other node in the network. This is achieved by conceptually dividing up the nodes into a binary tree, featured in the picture below [3]:

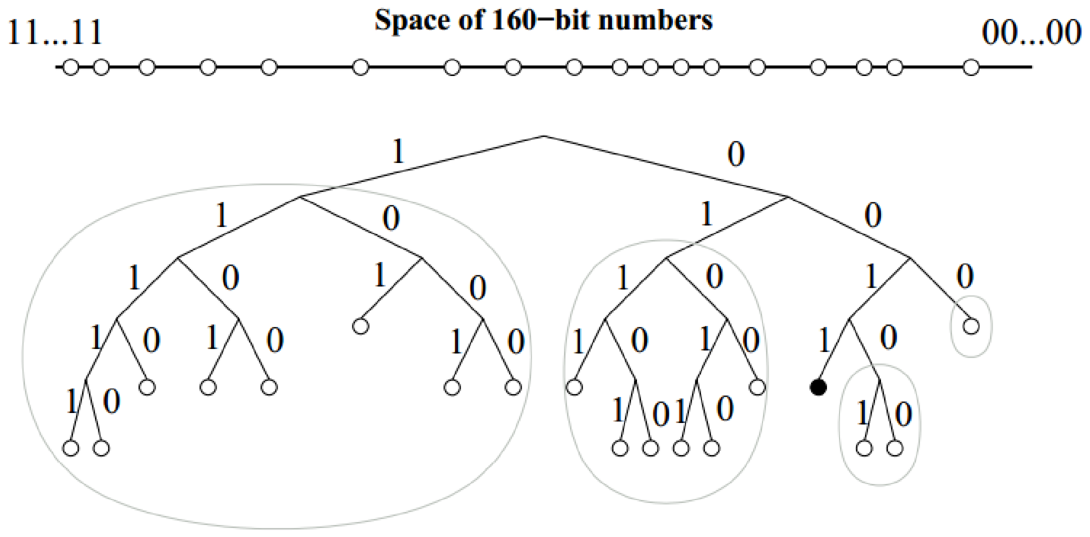

Fig. 1: Kademlia binary tree. The black dot shows the location of node $0011\cdots$ **in the tree. Gray ovals show subtrees in which node** $0011\cdots$ **must have a contact.**

Starting at the root of the tree and with a given node's binary ID, the given node must have a *k*-bucket for each divergent subtree in order to be able to contact any node within that subtree. *k* is a protocol-defined number, usually around 20, which specifies how many nodes the bucket can hold. In the example above, with the node ID of 0011, each grey oval represents a divergent subtree from node, starting at the most significant bit. The Kademlia protocol also offers some performance optimization by making sure to keep the nodes that have been alive the longest within its *k*-buckets, since it has been shown that the longer a node has been alive the more likely it is to stay alive [3].

The basis of Kademlia's search and storage operations is its node lookup algorithm, in which the node tries to find the *k* closest nodes to a given ID, and is implemented as follows. The lookup starts by the node picking *a* nodes, with *a* being a system-wide concurrency parameter, and sending a FIND_NODE RPC (Remote Procedure Call) to them, with each node returning a list of *k* nodes closer to the ID than they are. The lookup initiator then resends the FIND_NODE RPC to the *a* nodes from among the *k* nodes returned. The lookup stops once the initiator has queried the *k* closest nodes to the given ID parameter and received responses that did not result in a closer match [3]. Storage and value lookup are both implemented using this algorithm, a <key, value> pair is stored at the *k* closest nodes to its key and, when looking for a key in the network, a node will return the value instead of the *k* closest nodes if it contains the specified key.

These three concepts form the basis of the Kademlia implementation, and allow one to evaluate the DHT implementation from a security perspective.

4.0 - Peer-To-Peer Attacks

Now that the foundation of the P2P network has been explained, one can use this knowledge base to look at and analyze the different attacks that can be performed on a P2P network. This paper will ignore any attack based on outside applications or outside vulnerabilities, it will instead focus on the attacks allowed by the nature of the DHT implementation within the P2P network. Any attack on a P2P network has one of two ideas in mind; either mess with the network's routing by connecting nodes to the wrong nodes within the network, or send corrupted data back on a value lookup. To accomplish this, an attacker will typically use two different attacks, called the Sybil attack and the Eclipse attack.

4.1 - The Sybil Attack

P2P protocols typically rely on the fact that the network is made up of numerous remote entities that are independent of each other in order to maintain data correctness and security [4]. However, since the protocols are unable to determine the actual physical location of each entity, a malicious attacker can exploit the network by running a number of nodes in the network that are all colluding with each other. This use of multiple remote entities that are all controlled by the same actual entity is called a Sybil attack. The purpose of this attack is to gain a disproportionate amount of control over the P2P network by overriding the built-in redundancy of the network [4]. While by itself the Sybil attack does not do anything malicious, it can be used as an attack vector for both storage and routing attacks.

4.2 - The Eclipse Attack

An Eclipse attack is similar to that of a Sybil attack, except that the purpose of an Eclipse attack is to fill the routing table of a node with a majority amount of malicious nodes. When successful, the attacked node is said to have been “eclipsed” [5]. By holding a majority of a certain node’s contacts, the attack is able to effectively isolate that node from the network and send it whatever information that they want, as well as use that node to relay incorrect information to other honest nodes. The usual attack vector of an Eclipse attack is to send incorrect routing updates to a node, continuously filling a nodes routing table with more and more malicious nodes. An Eclipse attack is successfully defended against if the protocol can ensure that no more than some small fraction of a node’s routing contacts are malicious [5]. Like the Sybil attack, the Eclipse attack does not do anything malicious in and of itself, but can be an effective attack vector for storage and routing attacks.

5.0 - Hardening Peer-to-Peer Protocols

Given these attack vectors, the time has come to consider how secure P2P networks, and the Kademlia DHT implementation, are, and whether it is possible to harden them even more. The first point to realize is that, according to Douceur in his 2002 paper, it is impossible to be completely secure against a Sybil attack in the absence of a central authorization entity and assuming real-world constraints such as computing power and other resources [4]. As one of the goals of a P2P network is to do away with central authority, this paper will ignore the idea of having a central node authorization process. This changes the problem space entirely, since instead of trying to ensure that no malicious nodes can enter the network, one must assume that there is some fraction of malicious nodes within the network and design the P2P protocol around that idea.

5.1 - Social Network-based Security

A number of different solutions to the Sybil attack have been proposed, none of which without their disadvantages. One of the most effective, and practical, defenses is based on a social network approach to identifying malicious nodes. Proposed by Yu et al. [6], their protocol, called SybilGuard, is centered around the insight that, while Sybil nodes can form as many social connections among other Sybil nodes, they are limited in their connection to the network of honest nodes to so called “attack edges”. If the attacker generates too many Sybil nodes, the social graph will start to have a bipartite look to it, with honest nodes on one side and Sybil nodes on the other. To guard against Sybil nodes, SybilGuard computes the route of a node’s edges. When a node wants to verify a suspicious node *S*, SybilGuard will take the intersection of all the verifying node’s routes with the routes of *S*, and will accept node *S* only if more than half of its routes intersect [5]. This verification works because of the relatively small number of attack edges, which means that an honest node’s routes will likely contain a large majority of honest nodes.

A major drawback of the SybilGuard approach is that it requires each node to have a public/private key pair in order to establish trust-relationships between nodes, which must be distributed offline for complete security. This makes sense in situations where a type of social network already exists, such as instant messaging, but complicates P2P networks with no implicit idea of trust relationships [5].

5.2 - The Eclipse Defense

With SybilGuard as a non-perfect but adequate defense against Sybil attacks, the Eclipse attack is now considered. Again, an Eclipse attack is considered defended against if one can ensure that only a certain fraction of nodes in the routing tables are malicious nodes. This means that even if successfully defended, P2P protocols must also use redundant routing to fully ensure the receipt of correct RPC return values. This is one reason why Kademlia is considered more secure than other DHT implementations, since redundant routing is built into the protocol.

The simplest defense against an Eclipse attack is to tightly constrain the node identifiers that will be accepted by a node's routing table. This defense depends on the node identifiers being random and stable, eg. attackers are not allowed to chose their own identifiers. However, this approach has a significant performance drawback in that the P2P protocol can no longer optimize based on network proximity [5]. The solution to this problem, according to the research done, was to hold two sets of routing tables, an optimized table and a constrained table. Lookups would be performed using the optimized table until a routing failure happened, at which point the constrained table would be used. This approach depends on the precision of the routing failure test, and the need to ensure the optimized table is not overly compromised by malicious nodes [5]. A separate solution proposed is to limit the number of ingoing and outgoing connections that each node has. This has been shown to effectively eliminate the threat of an Eclipse attack, but only with small bounds on the number of connections, which significantly decreases the performance of the routing algorithm [5].

In general, the defenses to the Eclipse attack require a trade-off between security and performance. This means that, in real-world applications, careful consideration should be taken as to the extent to which security and performance are prioritized.

6.0 - Conclusion

Having covered the major vulnerabilities that underlie P2P networks and their DHT implementations, a qualification of P2P network's ability to adequately respond to today's security-minded world is necessary. When talking about security in P2P applications, it is important to be explicit in what is meant by security. By the very nature of a P2P network, it is impossible to guarantee that no malicious entities will be within the network unless outside verification is used. The hardening of P2P networks aims to ensure that, first, only a small fraction of malicious nodes will be allowed to enter the network of honest nodes, and second, that every node's routing tables contain a sufficient number of honest nodes so as to ensure proper routing. In the discussion above on the defenses to these attack vectors, a number of proposed solutions were discussed, none without their drawbacks and limitations. While the Sybil and Eclipse attacks are not directly malicious, they are the main attack vectors that an attack would use to perpetrate a storage or routing attack, and that is why the emphasis was placed on them. Beyond the proposed solutions, any security-minded P2P protocol must implement both redundant routing and data replication to avoid relying on any single node to find a piece of information.

For real world applications, the decision to use P2P protocols should be carefully considered, but should not be dismissed out of hand. P2P networks provide significant advantages over the usual client-server model, and should be taken advantage of where possible. In situations where security is the most important concern, P2P networks are definitely not the preferred solution, as an 100% guarantee of security cannot be made. However, in situations where the information passed along the network is not a major security risk, and where the use case involves scaling dramatically as a necessity for fault-tolerance, P2P networks should definitely seriously considered.

References

[1] Schulze, Hendrik and Mochalski, Klaus. Internet Study 2008/2009. Ipoque. http://www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2008-2009.pdf

[2] Ali Ghodsi. "Distributed k-ary System: Algorithms for Distributed Hash Tables." KTH-Royal Institute of Technology, 2006. http://www.sics.se/~ali/thesis/dks.pdf

[3] Maymounkov, Petar and Mazieres, David. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. New York University. http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

[4] Douceur, John R. (2002). "The Sybil Attack". *International workshop on Peer-To-Peer Systems*. http://www.few.vu.nl/~mconti/teaching/ATCNS2010/ATCS/Sybil/Sybil.pdf

[5] Urdaneta, Guido et al.. A Survey of DHT Security Techniques. ACM Computing surveys, 2009. http://www.globule.org/publi/SDST_acmcs2009.pdf

[6] Yu, Haifeng et al. SybilGuard: Defending Against Sybil Attacks via Social Networks. http://www.math.cmu.edu/~adf/research/SybilGuard.pdf