Vulnerabilities of P2P Systems and a Critical Look at their Solutions

Marling Engle & Javed I. Khan
{mengle|javed}@cs.kent.edu
Networking and Media Communications Research Laboratories
Computer Science Dept., Kent State University
233 MSB, Kent, OH 44242

ABSTRACT

Peer-to-peer systems have emerged from a drive to realize a computing architecture which cannot be taken down by attacking any single point. Scale and massively distributed nature of its architecture are its characteristics defense. Interestingly, these two features also seem to have introduced new set of menacing vulnerabilities. The vulnerabilities become complex due to architectural goals such as load distribution, search facilitation, and easy of reconfigurability. A P2P network must be expanded to include nodes in a potentially unknown environment (such as the Internet). These untrusted nodes may be faulty, malicious, and act together to commit as much damage to the P2P network as possible. In this survey, we discuss some of the vulnerabilities of these P2P systems, and take a critical look at some of their solutions to better understand these new threats.

I. INTRODUCTION

Peer-to-peer (herein known as P2P) networks have gained immense popularity in the recent years. They have shown strength in providing many services, such as sharing files without the need for central servers, streaming multimedia with distributed load balancing, and distributed backup systems. These networks are able to provide these services because they are scalable, and resilient to node failure. For this reason, we have seen tremendous growth in all types of P2P systems. In order to continue growing, P2P networks must be robust, and fault tolerant. In a large and open domain, such as the internet, it is almost a certainty that malicious nodes will be joining the network. This fact means that the responsibility of handling attacks has now been placed on those who design and implement these networks.

Interestingly, P2P systems emerged from a motivation to realize a computing architecture which cannot be taken down by attacking any single point. This was the motivation behind the decentralized design of Napster and the transition from Napster to later architectures after its indexing system was attacked. Two characteristics strategies have been adopted to achieve this apparent invulnerability- massive scale and ultimate distribution of its all functions and services. Millions of distributed users spread all over the Internet keep the system ever running. There is minimum centralized component in its architecture which if taken down can injure it critically. However, are P2P systems really invulnerable? Interestingly, these two strategies seem to have created a new set of menacing vulnerabilities. A P2P system running on the Internet also faces some of the threats any network application will face. In this paper we present a survey of these new P2P vulnerabilities which are very specific to P2P systems. For brevity, we mostly discuss only the new problems such as insertion which are specific to P2P architecture. We also include few older problems which took a new dimension on P2P systems.

This paper uses a generic model when discussing P2P networks. This model is used as a tool of abstraction. The vulnerabilities discussed here do not apply to some singular network, but to all networks of this type. This model consists of a few basic components:

1. An ID space consisting of b bits (example: 128bit unsigned integers in a Pastry[2] network)

2. An ID mapping system (which defines the node space of the network, example: Chord[1] is a 1 dimensional, circular

3. A routing system, which uses a key to forward a message to a destination set. An established set of network maintenance rules, for updating the network upon node arrival or departure.

Clearly this is a simple system model, and as such, it can be applied fairly easily to all existing P2P networks.

The attacks discussed in this paper are classified into three categories: Low Level, Mid-Level, and P2P Layer (High Level). These levels correspond to what is actually being exploited in the attack. The levels (or layers) themselves, are similar to any network model (such as the TCP or OSI layer model, Figure 1). Lower layers get closer and closer to physical electronic signals, where higher levels operate with a more abstract property. For example: TCP is built on top of IP, so TCP is at a higher level than IP is. Similarly, P2P networks are (usually) application level networks. It means they are built on top of other layers which provide some form of communication between nodes at a network layer. For the

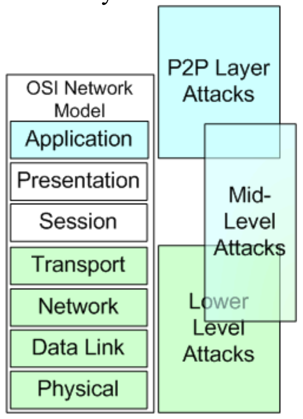

Figure 1: *Layer classification system*

purpose of this paper, no specific layer model is needed, as we are only interested in differentiating between the application level network (P2P) and the communication network (TCP, IP).

The goal of this paper is to survey the vulnerabilities that are found in current P2P networks, and look at some possible solutions to them. To accomplish this goal, this paper consists of attack/solution pairs which start at the lower (network) level and finally end with the P2P specific attacks.

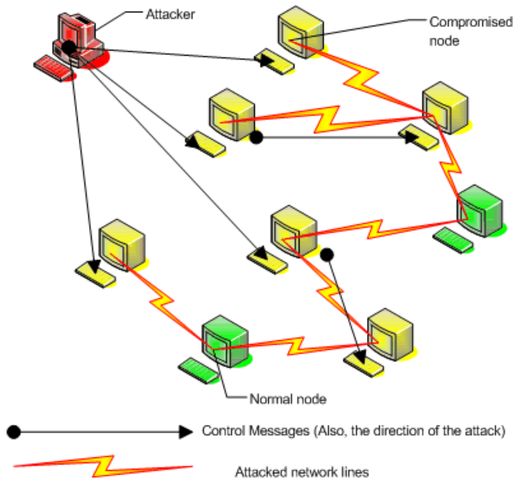

Figure 2: *A DDoS network which is successfully attacking all P2P communication channels*

II. Lower Level Attack

A. Denial of Service Attack

A Denial of Service (or DoS), as the name implies, is an attack which causes a service to stop functioning. There are infinite forms of DoS, but when it comes to P2P networks, the

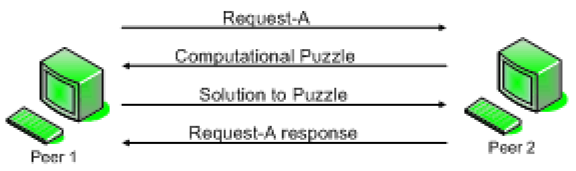

Figure 3: *Pricing is a method to limit the speed of requests. In this example Peer-1 is sending a Request-A to Peer-2. Without pricing the request would simply get a response of "Request-A Response" immediately.*

most common attack is a simple flood. This attack floods the network with invalid packets, therein preventing valid queries or messages from being delivered. Effectively this stops all communication along affected routes.

DDoS In such a flooding attack, a single host only has so much bandwidth to contribute. This is where Distributed Denial of Service (DDoS) comes in. A DDoS is classified by having many nodes participate in the attack. In this type of an attack, the attacker is hidden behind an extra layer of indirection, making it very difficult to find the original source (Figure 2).

The DoS (and DDoS) attack becomes more likely when a node is involved in a large P2P network. To be in the network, the node must be placed in some sort of reachable network zone (usually this involves being placed outside of corporate firewalls, or specifically allowing the P2P traffic through). This puts the node at a higher level of risk, simply because of the required reachability for accessing the P2P network.

If your paper is intended for a *conference,* please contact your conference editor concerning acceptable word processor formats for your particular conference.

B. DDoS Solution

The first problem with defending against DoS attacks is detecting them. The signs of a DoS (or even a DDoS) are very similar to the signs of high network utilization. Another key factor, is that DDoS is very difficult to block because of the large number of nodes that can be involved. This fact is amplified when the attacking nodes use legitimate nodes to bounce their attack (where the attack seems to be coming from legitimate nodes). These two facts make it basically impossible to block all DoS attacks.

That being said, there is a widely used technique to make DoS impractical, or at the very least slow it down tremendously. This method is known as 'pricing' (Figure 3). Pricing is used to limit the speed at which nodes make requests in the network (of any kind). When the attacker wants to request something of some node, the node responds with some sort of computationally intensive puzzle (example: What can you add to the string 'adabsdh1' in order to make the first X bits of it's SHA-1 hash all zero?). Then, the attacker (any node making the request) must solve this puzzle and provide a valid response before the request is even recognized.

C. *Man in the MiddleAttack*

A Man in the Middle (MitM) attack is when an attacker places himself between two other nodes in the network, where all communication between the two nodes passes through the attacker (Figure 4). Such an attack can remain undetected, as long as the attacker remains passive. This allows the attacker to listen to all communications between the two nodes for as long as desired. After gathering sufficient information (if needed), an attacker can choose to become more active. In this case, the attacker can modify messages as he forwards them, but he can also insert fake messages to either node from the other. Through this mechanism, the attacker can assume the identity of either node (or both). Also, because the attacker can influence the perspective that either node has of the network, he can fabricate a new (false) identity and simulate messages from it (and receive messages sent to it).

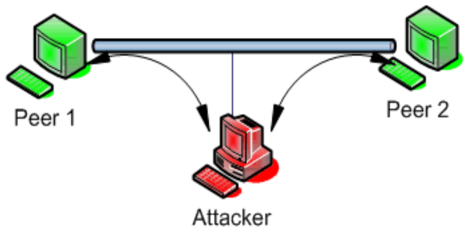

Figure 4: *Here an attacker has placed himself between Peer-1 and Peer-2. This enables the attacker to execute all classes of MitM attack*

This particular attack can be executed at the network layer of communication. In which case the attacker can see all lower level communications between the two nodes, and because this layer is below the P2P layer, the attacker has no problem faking any kind of P2P message he desires.

As in the DoS attack, a MitM attack becomes even more likely in a P2P network (very much so, in fact). To attack the network level, with MitM, the attacker must find a way to place himself in the route between two nodes (which may be a complex task). However, to place himself between two nodes at the P2P level, this is usually no problem at all. All P2P systems which have no control over node placement in the ID space (most current networks: pastry, chord, etc), are *extremely* vulnerable to this level of attack. An attacker can place himself anywhere in the network he wishes. This allows for a very deterministic and targeted attack (such as preventing a specific node, from reaching a specific other node).

D. *Man in the Middle Solution*

The main method of defending against MitM attacks is to make them as worthless as possible. Without some sort of important node (like a Certificate Authority (CA), an indexing server, or a super node (which controls a cluster)) a MitM attack boils down to compromising the integrity of two nodes in the network. Such an attack is not a threat to the entire network as a whole, so long as the fault tolerance of the network can handle the loss of the three nodes involved (2 peers and 1 attacker).

Many networks do have some form of super node or certificate authority, as a tool of preventing other forms of attack. Even for those networks which lack these central points, MitM attack could be done on a larger scale (Eclipse, as discussed in the P2P attack section). This is enough to warrant some form of protection from this attack.

The most widely accepted prevention of information tampering is the usage of digital signatures. These signatures are based on public key cryptography, and allow the integrity of a document/request/message to be verified. Such technology has been used in e-mail for many years, to provide authentication of a message. This same method can be used to detect if a message has been modified in a P2P network. We simply attach a digital signature to the end of our message, and the receiving node (or any node in between) will be able to verify that the message is unaltered and did indeed come from the true source.

After preventing modification of messages, and the inserting of fake messages, we now need to prevent the MitM attacker from being able to read the messages. Again the solution comes from public key cryptography. Along with attaching the signature, the sending node can also encrypt the message with the destination nodes public key: thereby making it improbable for any node other than the destination to read the true contents of the message.

III. Mid-Level Attacks

A. *Worms*

A worm is a "self-replicating computer program, similar to a computer virus. A virus attaches itself to, and becomes part of, another executable program; however, a worm is self-contained and does not need to be part of another program to propagate itself." [11]

A worm produces very significant threats to P2P networks. I classify the worm as a Mid-Level attack, because most worms will propagate at the lower level network, and will spread through vulnerabilities that are not generated by the P2P network itself.

Although the vulnerability is not created by the network itself, the network definitely amplifies the threat. The biggest reason, is that many P2P networks will be running the same software. This means that when a vulnerability in that software (such as a buffer overflow), all of the nodes in the

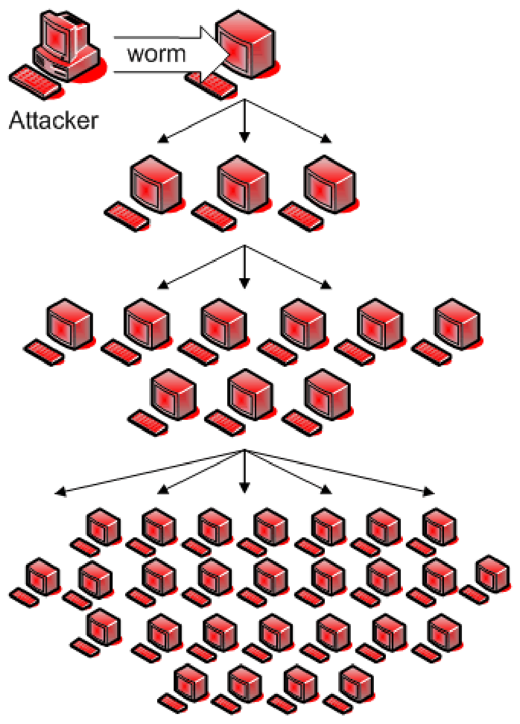

Figure 5: *Once the attacker inserts the worm, it spreads through the neighbor set (collected from the P2P network). Each step of spreading the worm doesn't involve the slow task of scanning randomly, the worm simply grows in the network exponentially (by the average degree of nodes).*

network are also vulnerable. So where a normal worm must scan (usually randomly) the internet looking for vulnerable hosts, a P2P worm need only look at the P2P routing tables and infect the hosts neighbor set. For this reason, the worm will spread exponentially (by the average degree of nodes) through the network. When compared with normal network worms, a P2P worm infects all nodes almost instantaneously (Figure 5).

Other than the fast spreading style of P2P worms, there are other factors which make it a large threat as well. P2P networks usually have a large transfer capacity (especially those that were intended to transfer files), so the worm can be a large piece of complex software (when compared to other worms, some of which must fit in a single TCP/IP packet) which is capable of much more complicated tasks/attacks. Also, since the computers in a P2P network on the internet are usually personal computers, they can be used to collect all sorts of information: credit cards, account passwords, etc. This makes the P2P network a large target, on top of its already vulnerable nature.

Finally, the worm can use the network itself as a tool. P2P networks on the internet are usually large, meaning that when compromised, the entire network can be used as a tool to execute other attacks on other hosts or networks. For example, a P2P worm could be spread to collect credit cards and passwords, and then while this information is being collected, the network could be used as a DDoS tool to flood Microsoft.com and take it offline, costing millions for Microsoft, and potentially more millions to those involved (spread out over all the P2P nodes whose information has been stolen).

B. *Worm Solutions*

The main idea to defend against such worms, is to keep the application itself secure. Without this common vulnerability the worm could not spread as effectively throughout the network. One suggestion that was given was to write P2P clients in strongly typed languages, which could avoid many security flaws (like buffer overflows).

To decrease the efficiency of the worm, we can avoid the hybrid networks (which contain 'super nodes'). These super nodes provide major increases in the rate at which a worm will spread (because of their high degree of connectivity).

Another possibility for reducing the danger of worms, is to use a hardened operating system. OpenBSD (>=3.8), for example, uses pseudo-random memory addresses when allocating memory. This, again, makes it more difficult to execute many attacks successfully.

The most practical defense to worms on P2P networks (that are implemented on the Internet) is to use the open nature of the network itself. That is, to develop open standards. Freely releasing the protocol and even code to implement network clients encourages developers to make their own client for that specific P2P network. These new clients will diversify the network, so not everyone will be vulnerable to the same exact flaw found in one client.

IV. P2P Layer Attacks

A. *Rational Attack*

For P2P networks to be effective, nodes participating in the network must cooperate (in general). However, when human nature is allowed to intervene, this does not always happen in a fair and efficient manner. In these cases cooperation is not enforced. The assumption is made that most nodes will exhibit rational behavior. That is, they seek to minimize their own resource sharing, while maximizing their resource consumption. There are many reasons behind this behavior, including:

1. Save upload bandwidth which is heavily regulated by most internet service providers.

2. Legal issues - Sharing copyrighted material which may results in legal action being taken against the owner of the node. In most networks it is easy to track the nodes which

are sharing the content.

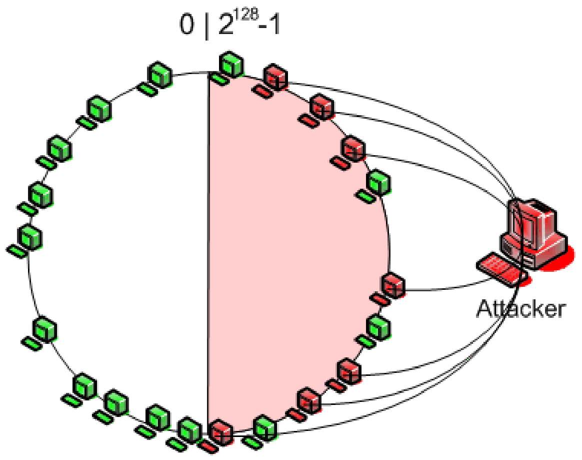

Figure 6: *This is an example of a Sybil attack against a Pastry network (Pastry was chosen because it is easiest to depict a network segment in the 1 dimensional ID space). Here the attacker has gained control of half of the network, by placing many fake nodes within the upper half of the ID space.*

3. Principle alone - When left to their own choices, some people will not cooperate based solely on the desire to not help the network, no matter how minimal the cost to them may be.

There are two basic classes of this attack:

1. *Content Restriction* - Users are not sharing content on the network.

2. *Resource Restriction* - Users do not contribute their resources to the network.

B. *Rational Attack Solutions*

Many networks experience this problem, but few of them attempt to solve it. Napster, one of the first P2P networks, tried to solve the content restriction and resource restriction by giving people a "title" for the level at which they shared (making it an issue of fame among other users). Samsara [10] (a P2P backup system) ensures that a node may only use as much space on another node as it is giving up to the network (again, solving resource restriction).

Of all the networks, Bit Torrent seems to do the best job of defending against rational attacks. Bit Torrent is uninterested in the number of files users may share, or any of the content itself, so the only problem it needs to address, is that of resource restriction. To do this, it implements a system for bartering for chunks of data. The more a node shares with others, the more it will get back. So, the more a node is willing to upload to others, the faster download it gets.

C. *Sybil Attack*

A Sybil attack is when a single malicious entity represents

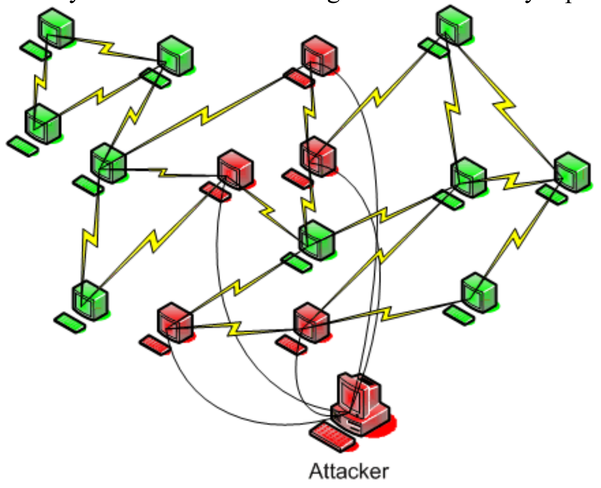

Figure 7: *Here an eclipse attack has partitioned the network into two separate spaces. The attacker now controls each side's view of the other, and all messages flowing between them*

a (usually large) number of users on the P2P network, in order to gain control of a segment of the network (Figure 6). This attack is executed by the attacker joining as many different nodes in the network near the same portion of the ID space. The network becomes more vulnerable to this attack of the attacker can manually influence where in the ID space the new nodes are placed. In this case, the attacker can use a minimal number of nodes and inflict a large amount of damage to the network. Once the attacker has enough nodes in that segment (as compared to the number of legitimate nodes) the attacker can control all messages that pass through the segment. This attack is also a gateway attack, meaning it can be used to execute large scale attacks of other types (such as Eclipse, discussed in the next section).

D. *Sybil Attack Solutions*

The main problem found while trying to defend against a Sybil attack, is the open nature of the P2P network. Without a central trusted authority it is impossible to entirely stop a Sybil attack [5]. The best any open, and decentralized P2P network can do, is to make it impractical.

To slow a Sybil attack, it is possible to use the same method that is used to slow a DoS attack: pricing. To join the network, a node must solve some sort of computational puzzle. Using this method, an attacker would have to continuously solve these puzzles in order to join more nodes to the network. If the puzzles are complex enough, it would take an attacker a great deal of time to place sufficient nodes in the network. If you also apply some sort of node ID expiration to the network, then this even more effectively limits the Sybil attack. As now the attacker only has a limited amount of time to generate enough nodes to execute the attack, before the first nodes that joined start to become invalid.

E. Eclipse Attack

The goal of an Eclipse attack is to separate the network into two or more partitions. When successful, all communication that passes between them must be forwarded by a malicious node (Figure 7). This attack basically is a large-scale MitM attack, but executed at the P2P network level. To execute the attack, the attacker places nodes on strategic routing paths that exist between the two partitions. After the network has been partitioned, the attacker can continue to large scale MitM attacks, such as: Faking messages from either side to the other, creating fake nodes on side B- as seen from side A. A successful eclipse attack, combined with creating fake nodes, could bring most networks entirely down (especially networks with relaxed rules for maintaining efficient routing tables). This is because the fake nodes could be populated in such a way, as to fill the routing tables of each node with invalid entries.

F. Eclipse Solutions

The key to preventing an Eclipse attack is the same as preventing a MitM attack. Digital signatures and public key cryptography will prevent fake messages, modification of messages, and passive reading of messages. However, because of the scale of an Eclipse attack, it still poses a threat to the entire network (where MitM did not). If messages are all dropped, then the entire network is split into two (or more) partitions. Given enough strategic locations, the attacker could partition the network into as many partitions as desired (thereby limiting the size of each network, and limiting the usage of the whole network).

As in the Sybil attack, it is important to prevent an attacker from choosing where new nodes are placed in the ID space. This will mean it takes a large number of nodes to probabilistically obtain enough control to partition the network. Thus, it is important to note, that with a large enough Sybil attack it is *always* possible to execute an Eclipse attack.

V. 5.0 Conclusion

P2P networks need to be robust against faults in the network and sudden node departure, as they are currently being designed, but they also need to be robust against security threats. If the network can prevent these attacks, then the network can allow any node on the Internet to join, and begin to fully realize the power of the P2P paradigm. There are a few basic problems in current P2P networks which must be addressed.

1. Prevent the node from choosing its node ID

2. Limit the rate at which nodes may join the network, and send requests (perhaps with pricing)

3. Use public key cryptography and digital signatures to eliminate message tampering, fake messages, and unauthorized reading.

4. Use and develop open standards, in order to diversify the software used in the network

If these four properties of the network are maintained, then all of the threats to the network discussed in this paper are effectively limited. These properties provide a great benefit for the overhead that they cost. They allow the network to remain in whatever structure it is in (pure P2P, hybrid networks, centralized servers) while adding a great deal of protection against security threats.

VI. References

[1] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications".
[2] Antony Rowstron, Peter Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems".
[3] Atul Singh, Miguel Castro, Peter Druschel, Antony Rowstron, "Defending against Eclipse attacks on overlay networks".
[4] Lidong Zhou, Lintao Zhang, Frank McSherry, Nicole Immorlica, Manuel Costa, Steve Chien, "A First Look at Peer-to-Peer Worms: Threats and Defenses".
[5] Hosam Rowaihy, William Enck, Patrick McDaniel, Thomas La Porta, "Limiting Sybil Attacks in Structured Peer-to-Peer Networks".
[6] Mudhakar Srivatsa, Ling Liu, "Vulnerabilities and Security Threats in Structured Peer-to-Peer Systems: A Quantitative Analysis".
[7] Baptiste Pretre, Roger Wattenhofer, "Attacks on Peer-to-Peer Networks".
[8] John R. Douceur, "The Sybil Attack".
[9] Emil Sit, Robert Morris, "Security Considerations for Peer-to-Peer Distributed Hash Tables".
[10] Landon P. Cox , Brian D. Noble, "Samsara: Honor Among Thieves in Peer-to-Peer Storage".
[11] http://www.wikipedia.com "Wikipedia".