Peer-to-Peer Systems as Attack Platform for Distributed Denial-of-Service

Arno Wagner, Bernhard Plattner

Swiss Federal Institute of Technology Zurich, Computer Engineering and Networks Laboratory

{wagner,plattner}@tik.ee.ethz.ch

Corresponding Author: Arno Wagner, TIK, ETH Zurich, Gloriastr. 35, CH-8092 Zurich, Phone: +41 1 632 7004, Fax: +41 1 632 1035

Keywords: Distributed Denial of Service, Peer-to-Peer Systems

Abstract

Distributed Denial-of-Service attacks are an effective means to make a service unavailable, mask other attack activities and generally degrade or disrupt network functionality. The key characteristic is that analysis of and defence against this attack type is difficult because of the high number of attacking hosts and large amount of attack traffic that can be generated. The emerging Peer-to-Peer filesharing systems have characteristics that turn them into an attractive infrastructure that can be used as attack platform. Attackers that can compromise a P2P system can expect benefits such as a large number of participants, easy hiding of attack control traffic and good, global distribution of participating hosts. This gives attackers high flexibility and at the same time a smal risk of being identified. This paper explains these characteristics in detail and concludes that further research into this threat and into possible countermeasures is urgently needed.

1 Introduction

Denial-of-Service (DoS) is, as the name says, an attack type where a service is disrupted by the attack. A service can be a single server, e.g. a webserver, a service offered by a group of hosts or network connectivity itself. One of the easiest techniques to implement a DoS attack is to flood the target (be it network or host) with traffic so that non-attack traffic has only a small chance to get through. The disruption can be used for its own value in performing sabotage, but also as part of a more complex attack that depends on the unavailability of a service.

Distributed Denial-of-Service (DDoS) is an improved attack method where the attack comes from a large number of hosts, that are ideally distributed over many different places. The number of attacking hosts can reach into the millions, enough to seriously impact even large bandwidth Internet backbones.

While there are other kinds of (D)DoS attacks than the overload-type, it is a variant that is easy to design and implement. It needs only very little specific knowledge about the attacked target and the security of the target does not need to be compromised. Countermeasures to DDoS attacks are a current research topic and the problem is far from being solved.

The application of DDoS attacks to vandalism or terrorism is obvious: Decrease the public feeling of safety by making critical (or perceived to be critical) Internet infrastructure unavailable for some time. In addition DDoS attacks can be used to facilitate or mask other attacks, e.g.

by overloading intrusion detection systems or by generating a huge number of faked attacks so that it is difficult to find out which system was the real target.

Peer-to-Peer (P2P) systems, most notably in the form of P2P filesharing systems like Gnutella or Freenet are an emerging trend in Internet usage. These filesharing systems consist of an application, that is run by a participating user. A host participating in a P2P network is called a "node" of the P2P network or sometimes a "servent". Users can offer files to others and at the same time can search for files and download them.

We believe that P2P systems offer an attractive possibility to be used as attack platforms in DDoS attacks, as they have several characteristics desirable to an attacker. These are a large number of participants, many and difficult to predict connections from nodes to other nodes and difficult to predict traffic between nodes. As a result, compromise of P2P nodes is hard to detect from the outside and attack traffic and attack control traffic can be hidden in normal P2P traffic. While ordinary DDoS attacks are often scripted (i.e. pre-determined) in order to minimise attacker exposure, a compromised P2P system may offer enough security to an attacker that near-realtime attack control may be reasonably safe.

Furthermore there are reasons to believe that P2P systems in general may be easier to compromise than hosts, due to short development cycles, cross-platform implementations and very different level of competence of the designers and implementors. There are also some entities that want to shut down P2P networks, e.g. because of the sharing of copyrighted material. This further accelerates deployment of new designs and implementations that remove vulnerabilities to legal and technical attacks that aim to impair the functionality of a P2P system.

This work is part of a project that deals with detection and countermeasures for DDoS attacks as well as the possibilities for early warning systems that can detect attack preparation phases. The main focus of this paper is to explain an emerging threat from P2P systems.

The paper structure is as follows: In section two we give a brief overview of DDoS attacks and methods used to implement them. Section three discusses typical characteristics of P2P filesharing systems, section four discusses the advantages of a P2P system compromise as basis of a DDoS attack. Section five briefly discusses an example scenario. The paper ends with some brief thoughts on possible countermeasures in section six and a conclusion in section seven.

2 Distributed Denial of Service

While defence against DoS attacks can often be done by filtering the traffic from the few attacking hosts, defence against DDoS attacks is far more difficult and in fact a mostly open research question at the moment.

It is unclear at this time what type of countermeasures can help against massive DDoS attacks with thousands or millions of attacking hosts. Recent events [3, 1, 2] have shown that attacks on this scale are feasible with very short set-up times. A good overview of attack types observed and expected to be implemented in the future can be found in [11].

2.1 Practical Problems for an Attack

In order for a DDoS attack to be successful, several goals have to be reached:

1. Acquisition of sufficient attack resources.
2. Generation of suitable attack traffic from the compromised hosts.
3. Targeting of the attack traffic at the intended victim.
4. Robustness of the attack-scheme to countermeasures.

We will explain the resource acquisition problem in the next subsection. Let us assume for now that it can be solved.

Generating suitable attack traffic can be problematic, if the attack traffic arises the suspicion of users or of a (at the moment mostly hypothetical) automated counter measure systems. An example could be that a host that mostly had incoming connections (e.g. a webserver) suddenly starts to initiate a lot of outgoing connections. While this is not a practical problem for DDoS implementors at the moment, it would be desirable for them to do attacks without significant changes in host behaviour. One additional factor is that a DDoS attack originating from a large number of hosts (millions) can work with very little attack traffic generated by each participating host.

Targeting a DDoS attack is a more difficult question. Of course fully automated, scripted attacks are possible, but they have several drawbacks. For one, an attacker can only specify limited attack patterns. Another problem is that re-targeting is not possible. Re-targeting might be needed for multiple reasons. The most obvious are errors in the original attack plan and countermeasure evasion. But there is also a possibility that the attacker needs more flexibility because the target is not known at the time the attack is prepared or because the attacker wants to be able to stop an attack. Especially extortion schemes need finer control than a "Fire-and-Forget" weapon offers.

While flexibility in targeting a DDoS attack is desirable to an attacker it also has its drawbacks. One of the most important ones is that attack control traffic might be used to track and identify the attacker. The past has shown that at least some types of vandals and terrorists take great stock in getting away unrecognised after the crime. Furthermore if an attacker is identified fast, this can cut an ongoing attack short, making it an overall failure. While not really secure for an attacker, fully scripted attacks increase attacker security by limiting the window of exposure, typically to a time before the attack starts.

In addition control interfaces can be used to stop an attack. However, widely known cryptographic techniques (see e.g. [16]) may be used to make this hard.

Robustness against countermeasures is a grey area at this time. It is not really known what kind of countermeasures could be deployed in future networks, and how effective they can be. It seems reasonable so assume that attack traffic that does not differ much from ordinary traffic a host generates, improves robustness. Near-realtime control over the the attacking hosts should also improve robustness as it gives the possibility to evade and re-target.

2.2 Typical Attack Platforms

There are two main possibilities to obtain attack resources. One is to take over a host completely, usually using an exploit in a service the host offers or a weakness in an application used on that host. Typically a worm or virus is used to effect such a compromise, but there are also hierarchical schemes.

A general drawback of host compromise is that intimate knowledge of the operating system of the host is needed. As a result these attacks are usually targeted at hosts that run a specific operating system.

The second possibility is to take over just an application running on that host, preferably one that runs all the time, such as a webserver or a P2P application.

Application takeover has the advantage that the possibilities for detection are smaller, since a less drastic change is made in the system. Furthermore we believe that it could have a psychological advantage: People that have to fear a full host compromise might act far more alarmed than if just a specific application was compromised, while basically most data on the machine is still secure. As an effect "clean" application compromise, might be widely ignored, especially if everything, including the compromised application, continues to work reliably.

Typical Compromise Methods

A classical scheme to compromise hosts is to utilise insecure email clients. On some widely used systems these applications will often execute parts of emails and at the same time will run with system privileges. However this method is slow, as it typically requires an user to read email for the compromise to happen. There are similar schemes that use web-browsers. These vulnerabilities can be removed by better application and operating system design.

The second classical possibility for host compromise is a successful attack on an insecure service running on a host. For host compromise this requires that the attacked application has or can get system privileges. These attacks are often implemented by means of worms and can be quite fast, see e.g. [18].

3 Peer to Peer Filesharing Systems

Peer-to-Peer (P2P) filesharing systems have become widely popular today. When we talk about P2P systems in this paper, we mean the filesharing system type.

There are indications that P2P traffic is the largest single fraction of Internet traffic today, even peaking WWW traffic. Almost all of this filesharing activity is done by users, using a special application to turn a host into a node of the P2P system. Nodes connect directly to other nodes, without the need for a central server.

Operating a node typically allows the sharing (offer) of local files, as well as searching for and downloading of files stored at other nodes. With the increased availability of permanent Internet connectivity for home users, many of the hosts running such nodes are permanently active. As a result there is a trend for nodes to be optimised for long-term operation where the user just specifies what files to find, and everything else is done by the node automatically.

Early systems (e.g. Napster [12, 13]) rely on central, well known servers. More advanced designs either have no higher infrastructure at all, i.e. all there is are the nodes themselves (e.g. Gnutella [15, 10], Freenet [5]) or use servers that have small network needs, do not store any content and can change locations all the time (e.g. E-Donkey [4]). In some of the systems using a server infrastructure, there are efforts to do entirely without servers in the future.

We will not address questions dealing with the actual content being swapped. P2P filesharing is widely popular and we believe it will be one of the dominant uses of the Internet in the foreseeable future.

Typical P2P-Node Behaviour

A node has two phases of operation, initialisation and normal operation. During initialisation the node tries to discover other hosts that are nodes for the same P2P system or for those systems using servers, it tries to find a server. This can be done in several ways, but a typical solution is the use of a list of contacts that where made in the past. Even with very primitive systems like Gnutella this works surprisingly well.

After some connection to the rest of the P2P network has been established, a node keeps contact with several other nodes or servers. Connections are monitored for responsiveness. If a connection drops below a specific quality level it is usually terminated and replaced by a better one. These connections are used for host discovery, searching and administrative messages. Only very few systems (e.g. Freenet [6]) use them for the actual file transfer.

The usual method for file transfer is a direct host-to-host connection from the node where a file resides to the one that requested the file. Some systems use multi-download features where different parts of a file are delivered from different nodes concurrently. A typical node establishes many node-to-node connections with different lifetimes.

Future Development

We expect that future P2P filesharing systems will mostly use server-free architectures, where the functionality of all the nodes is the same.

It is likely that traffic will be encrypted and we expect that there will be attempts to hide the source and destination of transferred files. Furthermore we expect that nodes will typically keep some connections with other nodes that will have a longer lifespan. In addition there might be short-term connections.

4 P2P as Attack Platform

Most attention into DDoS attacks seems to be directed towards classical host compromise. There has been some research into attacks against P2P systems or within P2P systems, but we are not aware of research into attacks originating from P2P systems.

We believe that P2P systems ultimately are a far more attractive attack infrastructure than conventional host compromise. There are numerous reasons why a successful compromise of a popular P2P system might both be easier and at the same time offer better attack possibilities and less risk to an attacker mounting a DDoS attack. We will now discuss these reasons in some detail. We will give an example scenario in the next section.

4.1 Acquisition of Attack Resources

There are several aspects in P2P systems that can make a P2P system compromise easier than a conventional application compromise.

The first reason is that it is easy to discover other nodes of a specific P2P system. After all, the P2P nodes usually keep track of other nodes in course of their normal operations. While conventional attacks have to resort to methods like random scanning of hosts, P2P systems offer this functionality for free and within the system. Random scanning and related techniques are highly visible and can draw attention to the attacker and the fact that a resource acquisition phase is in progress. Use of the facilities of a P2P system on the other hand looks like part of the ordinary operation of the P2P system. In addition, should IPv6 [7] catch on, there is a possibility that scanning techniques will become far less efficient and even more suspicious, because of the vastly increased address space of IPv6 with its 128 bit address space, compared to the 32 bit used by the currently dominant IPv4.

The second reason is that there is a tendency to do cross-platform implementation of P2P applications, using technologies like Java, Python, Perl and other interpreted languages that can run on different operating systems and hardware platforms. There are several risks inherent to this approach: The first is monoculture with P2P systems that have only a single software implementation. If there is a security problem in this one implementation it affects all running instances. Another one is that once the application is compromised, platform independent attack code can be run in its context, removing the need for multi-platform attack code. Approaches like Java sandboxing do not help much, as application compromise and network access is enough to use the host in a DDoS attack. A further risk is that technologies like Java, Python or Perl support mobile code. In a bad P2P design, mobile code might be part of the design (in the spirit of "if it is there, make use of it") and it might not even be necessary to get full application compromise anymore. Getting the P2P network to execute and distribute some piece of mobile code might be enough.

The third reason is that at the moment P2P systems seem to have a short development cycle. Designers of P2P systems want their systems out there and establish a larger user base. In some cases this is motivated by commercial reasons, such as selling advertisement rights (E-Donkey), in other cases it is because entities that are not fond of filesharing start to attack the P2P system and a new, more attack resistant design is made. Of course people also like to play around and improve existing P2P systems. The problem is that there are various levels of competence in the P2P implementors community. While a badly working system will not attract many users and will therefore be unsuitable as DDoS attack platform, even a well working implementation might be done by people that do not really understand secure implementation techniques.

4.2 Attack Traffic

The biggest problem with attack traffic is that it tends to draw attention. If a user's machine suddenly starts to use a lot of network bandwidth, that may rise suspicion with the user and with system administrators. However P2P filesharing applications are supposed to generate a lot of traffic with a lot of different communication partners. If enough P2P nodes can be compromised, the attack traffic needed from each is small. This is an ideal solution to hide the attack traffic in the conventional P2P traffic. We believe this can be done in such a way that the overall traffic characteristics do not change significantly and furthermore the P2P application itself continues to work properly, avoiding user suspicion entirely.

There might be some limits on what kind of attack traffic can be generated, as it should look like P2P traffic, but if there are enough nodes at an attacker's disposal, unsophisticated network flooding attacks that just overload the target network or host can be enough to do the job.

4.3 Attack Control

Attack control is a double edged sword: On the one side it lets the attacker control what is going on and might be needed in some types of attack schemes (like extortion schemes). On the other hand control channels might be used to trace the attacker. The probability that such control channels are noticed gets higher when they use specific resources like a specific port on compromised machines or other channels like IRC [8] servers, as seen e.g. in the kwbot worm [9] for the KaZaA filesharing system.

P2P systems offer a simple and elegant solution for this. As nodes are communicating with each other all the time, it is possible to hide attack control traffic in the conventional P2P traffic. Using cryptography this can be done in a way that is non-obvious if only the network traffic can be observed. The example in the next section briefly describes such a scheme.

To generate attack control traffic, the attacker can simply run one or more nodes for the P2P network and use them to insert the attack control traffic into the network. To further increase the security level of the attacker, techniques from web-anonymizers can be used (e.g. [14]).

4.4 Attack Robustness

Most of the robustness in the attack preparation phase lies in not getting noticed. On the application side this entails that the characteristics of the attacked service do not change significantly. As P2P node behaviour is highly unpredictable due to the nature of these systems, short interruptions in the service are unlikely to be considerd suspicious. Even a crash and need for restart might be tolerated, as implementation quality is often not very high. On the network side of things, a little extra traffic is unlikely to be noticed or if noticed unlikely to be taken seriously, as many P2P systems are badly documented.

Robustness while an attack is in progress is mostly a question of attacking hosts not being noticed on their side of the network. An attacked host or network will always be able to identify some superset of the attacking hosts, and be it only the set of all hosts that send data or all hosts in a specific part of the network. If attacking hosts do not arise suspicion with their ISP, defence against the attack becomes a matter of the administrators of the attacked system trying to identify and notify the ISPs of the attacking hosts. Taking over a P2P system gives a high number of attacking hosts that reside in many different places. It is likely to be very hard to shut them all down in any reasonable amount of time. Furthermore cooperation of the ISPs and owners of the attacking hosts is uncertain. After all there is no problem on their side! In addition source spoofing techniques can make identification of the attacking hosts very time consuming.

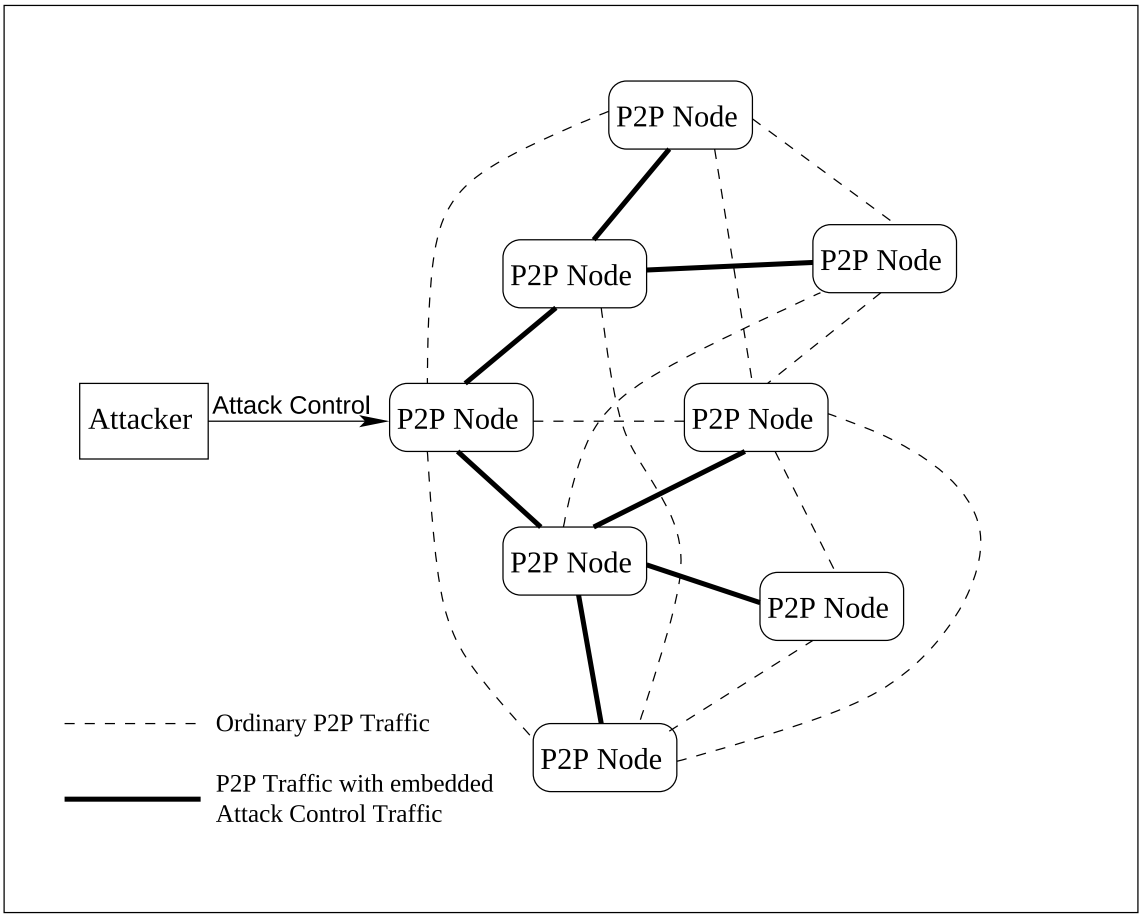

Figure 1: Possible Attack Control Scenario

5 Example scenario

We now discuss a possible attack scenario with near-realtime attack control. Figure 1 sketches such a scenario. To the right there is a fraction of a P2P network consisting of some P2P nodes. These have bidirectional connections to each other. (Other scenarios are possible, this scenario here is of the Gnutella type.) To the left is an attacker that has successfully compromised the P2P network. The leftmost node is under direct control of the attacker and allows the insertion of attack control traffic into the P2P network, giving the attacker the means to initiate, target and stop attacks from the P2P system directed against some target in the network. (Attack traffic is not included in the figure). Attack control messages are then distributed by a tree-like substructure (thick lines) of the node-to-node connections that form a part of the of the P2P network. Attack control traffic ideally forms only a small, hard to recognise part of the overall P2P traffic so that the thicker, attack control traffic carrying connections are indistinguishable from the dashed ordinary P2P connections for an external observer.

6 What Can be Done?

Fighting DDoS is hard. With a large number of evenly distributed attacking hosts it becomes even harder and is a mostly unsolved research question.

One approach could be to monitor P2P traffic in order to have a current list of most nodes in a specific P2P system. When an attack is identified as originating from this network, it could be attempted to globally shut down this specific P2P network for a time. Of course this approach would need global infrastructure that does not exist. Feasibility of such monitoring and shutdown mechanisms are pure speculation at this time.

We believe that more research in fighting general DDoS attacks as well as research into specific countermeasures to threats from P2P networks is urgently needed. Some promising ideas in a more general setting can be found in [17]. In addition, the authors of [17] propose the creation of a central or distributed body in analogy to the CDC (Centre for Disease Control) that monitors threats and infections and provides fast analysis and countermeasure services for new attacks.

We believe that without a reasonably well functioning infrastructure of people and technical systems targeted at analysis of and defence against DDoS and other types of attacks, the Internet will loose much of its present value in the future.

7 Conclusion

DDoS is a serious threat to Internet service availability and public perception of the Internet as a reliable infrastructure. We have shown that P2P systems are an infrastructure that is both vulnerable to compromise and at the same time has qualities that are highly desirable to attackers. We argued that compromising a P2P system in order to mount DDoS attacks could be a more attractive possibility than other attack solutions and one that it is harder to defend against. Our conclusion is that more research into threats from P2P systems and more generally into defences against DDoS attacks is urgently needed.

References

[1] Brown, J., and Moore, D. The spread of the code-red worm (crv2). http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml%, 2001.

[2] CAIDA. Caida analysis of code-red. http://www.caida.org/analysis/security/code-red/. visited May, 2002.

[3] Danyliw, R., and Householder, A. Cert advisory ca-2001-19 "code red" worm exploiting buffer overflow in iis indexing service dll. http://www.cert.org/advisories/CA-2001-19.html, 2001.

[4] Edonkey. http://www.edonkey2000.com.

[5] Freenet. http://freenetproject.org.

[6] http://freenetproject.org.

[7] http://www.ipv6.org.

[8] http://www.irchelp.org.

[9] http://securityresponse.symantec.com/avcenter/venc/data/w32.kwbot.worm.ht%ml.

[10] The limewire gnutella client. www.limewire.com.

[11] Mirkovic, J., Martin, J., and Reiher, P. A taxinomy of ddos attacks and ddos defense mechanisms. http://www.lasr.cs.ucla.edu/ddos/ucla_tech_report_020018.pdf, 2002.

[12] Napster. http://www.napster.com.

[13] Opennap. http://opennap.sourceforge.net/.

[14] Rennhard, M., Rafaeli, S., Mathy, L., Plattner, B., and Hutchison, D. Analysis of an Anonymity Network for Web Browsing. In *Proceedings of the IEEE 11th Intl. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2002)* (Pittsburgh, USA, June 10–12 2002).

[15] Ripeanu, M. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of the First International Conference on Peer-to-Peer Computing (P2P01)* (2001), IEEE.

[16] Schneier, B. *Applied Cryptography*, 2nd ed. John Wiley and Sons, New York, 1996.

[17] Staniford, S., Paxson, V., and Weaver, N. How to 0wn the internet in your spare time. In *Proc. USENIX Security Symposium* (2002).

[18] Weaver, N. C. http://www.cs.berkeley.edu/~nweaver/warhol.html, 2001.