Gnutella: Integrating Performance And Security In Fully Decentralized P2P Models

Rossana Motta
Florida State University,
Computer Science
1004 Academic Way
Tallahassee, FL, 32306 USA
motta@cs.fsu.edu

Wickus Nienaber
Florida State University,
Computer Science
1004 Academic Way
Tallahassee, FL, 32306 USA
nienaber@cs.fsu.edu

Jon Jenkins
Florida State University,
Computer Science
1004 Academic Way
Tallahassee, FL, 32306 USA
jenkins@cs.fsu.edu

ABSTRACT

Peer-To-Peer (P2P) systems have made an enormous impact on the Internet, directly affecting its performance and security. The litigation against P2P file sharing has led some designers to opt for purely decentralized P2P models. The latter have quickly become attractive to Internet users, who often consider pure P2P as more "secure" than hybrid systems (i.e. with some central entity).

In this paper, we concentrate on some relevant security threats and performance inefficiencies in the Gnutella P2P network, which is worldwide the most popular fully decentralized system. We present the results we obtain from the analysis of spurious content circulating in the network. We observe a significant propagation of unwanted and unrelated query replies, systematically taking place. This leads to the transfer of junk or unsafe files, potentially resulting in hosts' security violations and Denial of Service attacks. The analysis of IP addresses shows that peers responsible for spreading these files are recurrent over time and over specific network segments. They also share a specific pattern of common features, clearly suggesting the use of modified versions of Gnutella applications. Typically these peers run as supernodes (ultrapeers), which represent the highest level of control of the Gnutella system.

In spite of many different solutions proposed in the past to integrate security mechanisms into Gnutella, none of them have been adopted in practice. We discuss the necessary trade-offs of these proposed solutions and we also analyze the (unofficial) hypothesis that some entities, having commercial convenience in polluting the Gnutella network, may be involved. We propose solutions that help mitigating some of the problems, while still preserving the basic structure of the Gnutella protocol.

Categories and Subject Descriptors

D.4 [**Software**]: Miscellaneous; H.4 [**Information Systems Applications**]: Miscellaneous; J.8 [**Computer Applications**]: Miscellaneous

ACM-SE '08, March 28–29, 2008, Auburn, AL, USA.

General Terms

Experimentation, Security

Keywords

Gnutella, Security, P2P

1. INTRODUCTION

Peer-To-Peer (P2P) traffic represents the single largest–and still increasing– traffic type by volume on residential ISP networks. It is estimated that on consumer broadband networks P2P counts for 50-65% of the downstream and 75-90% of the upstream traffic [18]. The litigation against illegal P2P file sharing has led designers to develop pure P2P systems, which use a fully decentralized model, such as Freenet and Gnutella. Freenet has a proprietary protocol for peer communication, differentiating responsibilities, and circumvents lawsuits. It is essentially anti-copyright, anti-censorship and pro-anonymity. Gnutella is similar in some respects but is non-proprietary and uses HTTP for peers' intercommunication. A significant percentage of Internet users believe that pure P2P systems are more secure and less prone to users' traceability than hybrid ones and thus choose to use the former.

In this paper, we analyze major security threats and performance issues that systematically arise in completely decentralized architectures. The ubiquity of P2P easily extends P2P-related problems beyond a specific Peer-Network, impacting the security and performance of the whole Internet. We show that a fully decentralize architecture does not necessarily guarantee less traceability or better privacy. In our study, we examine the Gnutella network, which is the third most popular P2P system worldwide [18]. It is open source, fully decentralized and currently supported by over 30 different servent[1] applications. As of March 2006, Gnutella had, on average, over 4 million connected hosts [1].

Gnutella system has significant security threats, mostly deriving from the lack of any controlled mechanism of authentication and trust. Because of the query-flood mechanism, used for inter-node communication (e.g., for file discovery), high traffic volumes are generated by the propagation of fake query-replies, even when no file transfer occurs. This represents a significant performance flaw.

Our main contribution in this paper is the analysis of the malicious content circulating inside the Gnutella network

[1]Servent is a term applied to P2P applications that can act both as client and server.

and the traceback to a wide number of the peers holding it. While it has generally been assumed that such spurious content is random junk being shared by ordinary peers, our analysis shows specific, recurring patterns in the content type itself and recurrent anomalies in the peers sharing it. Such anomalies suggest the constant presence of some peers running modified versions of Gnutella servents and faking query responses in order to propagate large quantities of malware. We modified some modules of Limewire, a popular Gnutella servent. Proper modifications allow us to reproduce exactly the anomalous behaviors observed in malicious peers. Additionally, we consider an hypothesis, informally proposed in [12]. To the best of our knowledge, this has never been systematically studied. According to it, some commercial entities are responsible for at least part of the polluted content in Gnutella. Our results show the presence of malicious peers residing in corporate subnetworks that might indeed have economic incentives to pollute P2P systems as a means to combat illegal file sharing.

Authentication, trust and reputations models have been proposed but never integrated in official releases of the Gnutella protocol or any servents. We analyze the trade-offs that such integration requires and why the Gnutella Community is ready, or willing, to accept them.

Nonetheless, some security solutions are necessary in Gnutella, to isolate misbehaving peers and provide an acceptable level of trust in the system. We propose some mitigation strategies, that are easy to integrate and fully compatible with the Gnutella protocol. Based on the results of our analysis we claim that under the current pattern of exploitative behavior in Gnutella a distributed blacklist and modules to prevent (at least the involuntary) downloads or uploads of files with a clearly malicious pattern, would be effective, in several cases, isolating misbehaving peers and preventing them from joining the system in the first place.

2. PROBLEM DEFINITION

2.1 Overview of the Gnutella protocol

Since its beginning in the early 2000, Gnutella has been a pure P2P system. This design choice quickly led to its rise in popularity, especially as a consequence of the recurrent litigations against other P2P systems, facilitated by their semi-centralized structures. Millions of Internet users regularly run a Gnutella servent [1] and feel more confident using a pure P2P system rather than a hybrid one, where central entities may collect information about the peers.

The communication among Gnutella peers occurs through hop-messages (*query-flood*), each having a given Time-To-Live. The Peer-Network is organized in two levels, a *leaf* network and a *ultrapeer* (i.e., super-peer) overlay. Each ultrapeer controls a group of leaves and each leaf normally connects to a few ultrapeers. The latter hold information about leaves' location and their shared files, and mediate any search to and from leaves. In Gnutella, ultrapeers are the highest level of control and no other entity stands above them, that would ensure they do not misbehave. Gnutella servents have the code for both leaf and ultrapeer roles, and–most importantly–the code to choose between them. When a Gnutella client is started, it autonomously decides which role (leaf or ultrapeer) it should assume. Although this process is called "ultrapeer *election*", the single host determines, before going online, whether to be an ultrapeer or a leaf, depending on given criteria (e.g. available bandwidth, computational power, participation in the system, etc.).

2.2 Security/performance issues and related work

Gnutella protocol is relatively simple and does not embed any security features. As a consequence, the system is prone to a number of vulnerabilities. Some security threats are common to most P2P systems, for instance, IP address harvesting and privacy violation through traceability of peers [10, 16, 13]. Traditional ways to hide IP addresses, such as proxies or bouncers, are not viable for large scale massive file transfers. In a P2P network, any file transfer implies establishing a direct connection between nodes and consequently the possibility of tracing IP addresses, scanning hosts and knowing at least part of the shared files of other peers. Several papers have examined such threats, but none have proposed a practical solution[16, 10, 13]. However, such studies have shown that the widespread assumption that a fully decentralized system like Gnutella protects users' privacy, more than a hybrid, is wrong [13]. In fact centralized directories represent a *higher* risk than pure P2P only for the owners of the system, who can be easily traced by authorities and held responsible for the whole P2P system. Leaf users can be equally traced and sued for sharing illegal content in any type of P2P system, unless the single users adopts some preventive measures, e.g. blacklists [13].

Gnutella protocol is prone to some specific threats, as well as performance flaws. Several studies have shown the possibility of DoS and DDos attacks and the high bandwidth consumption, attributed to Gnutella query-flood mechanism [15, 8, 16, 10]. While it has been generally assumed that the risk of downloading malware in Gnutella is not higher than any other P2P network [11], the analysis of several other P2P systems shows that none appear to be similarly populated with noxious content. Our study shows that the probability of inadvertently downloading malware and junk files is greater in Gnutella, because large numbers of unrelated *query results* take place (Figure 1). The fake query results camouflage malicious content as a specific filename and/or attribute matching the query request. This induces the requesting peer to download the file. The incorrect results methodically routed back in response to each query request suggests a constant presence inside the system of malicious peers, who must have some incentives in cheating and spreading big quantities of unsolicited content.

A number of papers in the literature have tried to address Gnutella security issues, through building reputation systems [17, 21], integrating market-based models of misaligned incentives, [9, 20] or departing from a purely decentralized architecture [19]. Nonetheless, as of the latest Gnutella protocol version, no countermeasures have been adopted. The only security tools currently available in some servents are (1) a manual block-host feature, which blacklists specific IP addresses; and (2) manual content-filtering, which prevents query results with specific keywords from being displayed. Both tools are clearly interim, not robust solutions, as they are manual and single-host based. They are ineffective and do not offer any real protection. Additionally, even if an IP address is blacklisted by a node, the query result coming from that host address is still routed, since the other nodes have no knowledge of which IP addresses or keywords the requester wishes to block. Thus the waste of resources, by

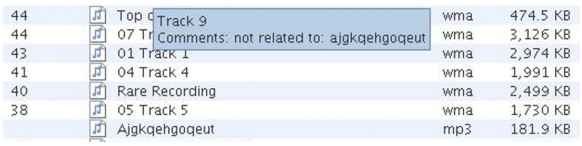

Figure 1: Junk results returned by a query with a nonsense string as a keyword (*ajgkqhgoqeut*).

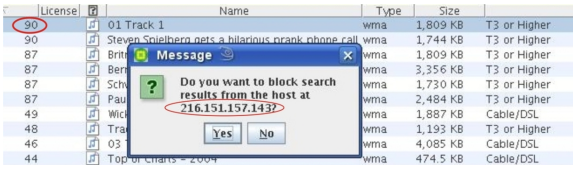

Figure 2: Although 90 peers are listed in the search results of these spurious files, only 1 of them actually appears to be sharing the file.

traffic generated by malicious query replies, remains. Content filtering is equally ineffective, as it is based on filenames in query results, which are dynamically modified and camouflaged by malicious peers. This effectively by-passes keyword filtering.

3. EXPERIMENT AND RESULTS

One of the goals of this study is to gather information about peers returning fake query replies and spreading malicious content in the Gnutella network. We exploite the *Block Host* feature in Limewire to retrieve the IP addresses for hosts that return a number of spurious files to random searches. We regularly repeated the experiments over a period of 7 months (April 2007 through October 2007). We discovered a small set of recurrent hosts (we retrieved a few hundred), which appear to be sharing these files, even if the number of peers reported as sharing the files is much higher (Figure 2). Specifically Table 1 displays some of the most recurrent addresses. The peers sharing spurious content show some common, anomalous features, for example they constantly appear to have excellent network connectivity, the file displays as popular, (i.e. shared by many peers) even when the Block Hosts feature shows a much smaller number of IP addresses sharing that file (Figure 2). Additionally, those peers typically have disabled the *Browse Host* feature, which is not optional in any of the popularly available Gnutella-compliant programs (Figure 3).

We downloaded and investigated some junk files with various extensions. Among the .zip and .exe files, a Trojan downloader known as Win32.Agent.auv [2], is the most prevalent. Files with .mp3 extension are typically advertisements promoting online stores, while .wmv and .mpeg files are mostly licensed material attempting to start remote connections to paid services. None of the retrieved files are viruses with the intent to harm the system, but instead are advertising or spam attempts.

The retrieved IP addresses are from USA, Europe and Canada. They might be dynamic, however, we found recurrent subnetworks and in several cases specific single addresses recurring over time. Threads on Gnutella forums, posted some months ago, also report malicious content that repeatedly originated from the same address ranges [3]. Malicious addresses are often observed to cluster within the same network. We can infer that such spreading of malicious contentoften does not come from isolated hosts that occasionally–voluntarily or involuntarily–share spurious content, but is more likely to represent an ongoing, organized activity.

Most of the retrieved IP addresses belong to residential ISPs, surprisingly however, a few of them originated from corporate or Government networks, such as Apple or Merck and most unexpectedly from the U.S. Department of Defense. All of these hosts were not only sharing suspicious content, which might theoretically be downloaded and shared inadvertently, but also showed the before mentioned features not complying Gnutella specifications. In our study we were able to connect to these hosts and download files. This shows that the IP addresses are not spoofed, as then no connection would have been possible.

Nmap [4] scans on a sample of 18 IP addresses revealed that all of those hosts were routers. It is likely that we could only reach the entry point of the network, and not see the end hosts, which were screened. Scan results for IP addresses belonging to the same network, for instance 216.151.157.140 and 216.151.157.135, were different to the open ports. However, all of the hosts were running remote controlling services such as VNC (Virtual Network Computing, a remote control program which allows one to view and fully interact with a computer remotely), AFS (Andrew File System) or, in most cases, both of them. Several of these peers accepted a Gnutella connection on AFS ports (range 7000–7009). Most of the ports normally used in Botnets [5] were not open. The analysis of the open ports and running services suggests that these hosts are not likely to be compromised machines. Additionally, none of the IP addresses that we collected appeared to be an anonymous proxy.

Data about nodes connected at any time to the Gnutella network are visible through the Gnutella crawler [22, 14]. By collecting real time lists, we could discover that all of the malicious peers that we analyzed were also acting as ultrapeers. The latter control and route the traffic from and to other ultrapeers connected to other leaves and especially mediate the query results. Clearly, no security or trust can be guaranteed inside the Gnutella peer-network as long as no mechanism that enforces security checks on ultrapeers exists. Nothing at present prevents ultrapeers from hijacking or faking query results and this represents a major vulnerability of the Gnutella system.

In our experimental study, we modified an official version of Limewire (release 4.13.1 of April 2007). By modifying the code that determines, when still offline, if a peer will connect as a leaf or as an ultrapeer, we successfully auto-elected ourselves as an ultrapeer when we did not have the prerequisites for the election. Other source code modifications allow us to fake query responses to every received query, camouflaging arbitrary files on the basis of the parameters received as a query request. The modification of less than 12 modules of Limewire is enough to cause the observed malicious behavior in a Gnutella servent.

4. DISCUSSION

4.1 The source of the vulnerabilities

Since Gnutella only relies on the servents' source code for ultrapeer election, it is quite easy to exploit code modifi-

Table 1: Details about some of the most recurrent malicious peers. Wildcards (*) mean that several different addresses belonging to the same network were found at the same time.

IP Addresses	Country	ISP
67.*.*.*, 68.*.*.*, 76.120.*.*	USA	Comcast
83.216.148.*	UK	Parbin Limited
208.122.*.*	NY, USA	Voxel Dot Net, Inc
216.151.157.*	CA, USA	ETHR.NET LLC
17.186.156.131	CA, USA	Apple Computer, Inc
26.229.201.64	USA	Dept of Defense Network Information Center
82.44.*.*	UK	Broadband Audit
54.188.11.87	NJ, USA	Merck And Co. Inc
64.210.144.*	Romania	Geekhosters
64.72.*.*	NY, USA	Alpha Red Inc
67.55.65.*	NY USA	Webair Internet Development Inc
70.86.48.180	TX, USA	Theplanet.com Internet Services Inc

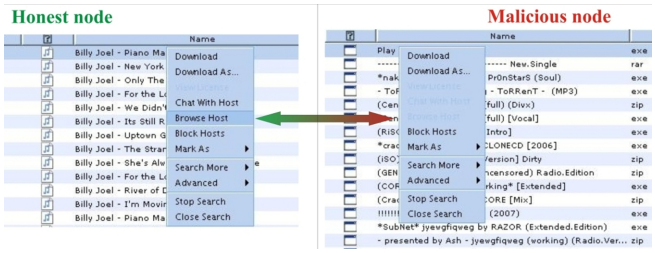

Figure 3: The Browse Host functionality is not supposed to be optional; however, whenever the content is clearly spurious, typically the hosts sharing those files cannot be browsed.

cations to force a ultrapeer connection, even without the prerequisites for this election. Similarly, hard-coded modules can cause ultrapeers to propagate fake query responses, instantly responding to any query request and camouflaging filenames, attributes, number of peers sharing the file and so on.

Ultimately, there is no way to guarantee that peers are running legitimate servents. Trying to authenticate the modules that a peer is running would be circumvented by a malicious node through re-engineering the authentication module of the legitimate source–an effort facilitated by the availability of source code–to reproduce legitimate authentication responses.

Theoretically, it would be possible to implement en-route filters on the query replies propagated by the ultrapeers, consisting in practise of selective gates on each servents. Under the assumption that most of the peers are not malicious, this would allow for shielding against bad query results, being routed back through the ultrapeers overlay network. However, there are two major problems with this approach, highlighted by Limewire developers, with whom we discuss this solution [6]. First of all, not all query replies travel through hops: the latest version of Gnutella protocol reduces flooding queries results by using OOB (Out-Of-Band) communication, that is the replier sends a UDP response directly to the requester. Additionally, even when query responses are flooded through the network, according to the protocol they are in the form of a container of data, i.e. a blackbox. There is no explicit separation of fields, such as filename, size, which would allow the program to efficiently parse and evaluate these responses. Thus, implementing a parser for the huge number of replies passing through an ultrapeer would represent an unacceptably expensive CPU burden. Ultrapeers are not real servers, instead typically they are normal Internet users that run P2P systems in the background on their personal computers. Few users would accept to be severely impacted in the normal usage of their machine, because of computational demands imposed by the P2P client acting as ultrapeer.

4.2 Identity of malicious Ultrapeers

The observed trend of recurring IP addresses found over time for malicious hosts suggests that such nodes are constantly allocating their resources to act as ultrapeers and thus they must have some convenience in this activity. Most of the retrieved IP addresses do not provide precise clues, since they belong to residential ISP networks. A few of them, however, belong to corporate networks and even Government, as in the case of the Department of Defense. The latter is well known to scan hosts on various P2P networks [7], however, in this case, they are also actively injecting junk content, propagating fake queries and running modified versions of Gnutella servents. While theoretically the nodes located in Company subnetworks could be compromised machines being used as relays, we believe that it is extremely unlikely that the traffic generated by such nodes would go unnoticed in a corporate network.

According to an unofficial hypothesis [16], the spurious content might be part of a deal between Limewire and RIAA, after the latter started litigations with Limewire. This, however, has never been confirmed–or denied–on official Gnutella-related sites. Like it happened with Kazaa, the pollution inside the network might be a deliberate plan to decrease the popularity of a system in which a great amount of copyright infringement continuously take place. In fact, a fully decentralized system is virtually "shutdown" only when all of the peers decide not to run it any more. For example, Apple Corporation, which appears among the collected addresses, may have an incentive in fighting P2P because it competes with the Company's products (for instance iTunes). The Department of Defense, as mentioned, is a well known enemy of P2P. In other cases, such as for the hosts belonging to the Meck Co. network, we could not find any reasonable ex-

planation, since there is no appearent reason the Company would be affected by P2P file sharing.

4.3 Who wants a Solution, and who doesn't

It is admittedly unusual that one of the largest worldwide P2P systems does not support any form of security, authentication or trust. Gnutella has gone through 7 years of development, new protocol versions, tens of different servent applications. As mentioned in 2.2, a number of solutions have been proposed to integrate some level of security even not changing the protocol radically. One hypothesis for the reason why none of them has ever been integrated, even for an alpha-testing, might be some form of convenience of specific entities in not changing the actual in-security situation, for example for spreading advertisements or, as suggested in [12], not making the network too competitive with other systems that deliver the same material, but in a legal and–most importantly–paid manner. Such hypothesis is admittedly hard to prove, since no specific information will most likely be released by the involved commercial entities.

Many complains have been sent from users in Gnutella-related official sites. The responses, however, have been typically elusive and ineffective. It is often suggested to manually add IP addresses of suspicious hosts in the servents' filterlist or to pay attention to what is being downloaded. These solutions clearly do not address the essence of the existing security and performance problems. A system where the entities holding the greatest control (i.e. the ultrapeers) can be malicious is intrinsically unsafe, even more considering its huge extension and thus the prevalence, on the Internet, of these threats.

It is understandable that some of the most radical solutions have never been successful in Gnutella, for instance mutating the system to hybrid P2P, as proposed for instance in [19]. First of all this would require the protocol to be re-written basically from scratch and, especially, it would imply a very high risk of legal prosecution for whoever would agree to play the role of central entities, because of the great amount of copyrighted content illegally shared inside the Gnutella system.

On the other hand, it is not so trivial to explain why minor countermeasures to improve the security and trust of the system have never been taken into account. For instance, significant benefits could derive from distributed reputation or incentives models [17, 21, 9, 20], which on the other hand would not imply drastic changes in Gnutella specifications or noticeable over-load for the system.

5. PROPOSED SECURITY IMPROVEMENTS

If a security solution is really sought by developers in Gnutella Community, the most convenient and easily acceptable approach would probably reflect the fully decentralized nature of the system and for instance rely on distributed contributions, motivated by the fact that most of the peers are not malicious.

Ideally, any node that has exhibited malicious behavior should be rejected from the system and unable to join in first place. Since all the nodes bootstrap from ultrapeers, the latter, if honest, can be exploited as gates, to prevent bad peers from joining the system. As mentioned, it is not feasible to verify the authenticity of the application that nodes are running, so that we can only rely on some pre-knowledge about the users themselves. Nicknames and identifiers have been proven to be easily spoofed [9], however we can exploit the fact that the IP addresses of the malicious ultrapeers appear to be recurrent, and dynamically build and update distributed filterlists.

All the most popular Gnutella-compliant programs already provide manual block host and keywords-filtering facilities, so that peers can manually prevent results from suspicious hosts from being displayed. Suspicion can derive from human inspection and in practice it is very hard to efficiently automate this process, without paying the price of a large number of false positives. To our best knowledge, indeed, no automatic "intelligent" filters have ever been developed in Gnutella systems. In fact, in many cases spurious results actually appear to be related to the original search keywords, since the malicious ultrapeers automatically rename the results, thus matching the submitted query. One problem with the existing keywords and IP address filters is that locally blocking search results will not prevent the query results from being propagated throughout the network. Instead, they are simply not displayed in the end host. Thus this approach will not prevent unwanted load, generated by junk traffic. Additionally, manually blocking hundreds of malicious nodes is definitively beyond the patience of most of users.

A possible improvement over the actual situation would be adding to the Gnutella netowrk distributed IP address-based blacklists. These can be are updated dynamically and in a distributed fashion (i.e. with the contribution of peers), but controlled and delivered by a central authority. This is similar to what currently happens for Emule network server lists. At startup, Gnutella servents connect to a server and download a simple text file. The file reports a list of IP addresses, which are automatically added to the local IP filter list. The Gnutella official website could easily host such small files, or, if their interests run contrary to it, a user community could set up a website for their distribution. The blacklist file would be downloaded before servents join the peer-network (i.e. before the handshake stage with any other peer): nodes only need to contact a remote server, for instance via http, thus they do not need to be connected to the P2P network for this purpose. In a scenario where many hosts acting as ultrapeers support filter lists, malicious hosts would not have the chance to join the system. This would effectively prevent unwanted traffic from being generated in the first place.

The lists need to be updated frequently, since bad peers may change their addresses. This can be done with the contribution of the whole Gnutella network, by using a report feature. Clients could implement a "report" button, to contribute to the building of blacklists. Ultimately, the control of such lists is central: this prevents abuses and at the same time does not represent a heavy burden for the central server, since the pattern of malicious ultrapeers is currently quite stable and easily recognizable. The fully decentralized structure of the system would not be tainted by such filtering feature.

Additionally, simple checks could be easily implemented in servents, to prevent peers from *inadvertently* sharing malicious content. In other cases, users *intentionally* inject bad files, by placing them in the shared folder, but they do not have the skills to modify the source code of the P2P program. Based on these assumptions, it would be possible to automatically prevent files that have a clearly spurious pat-

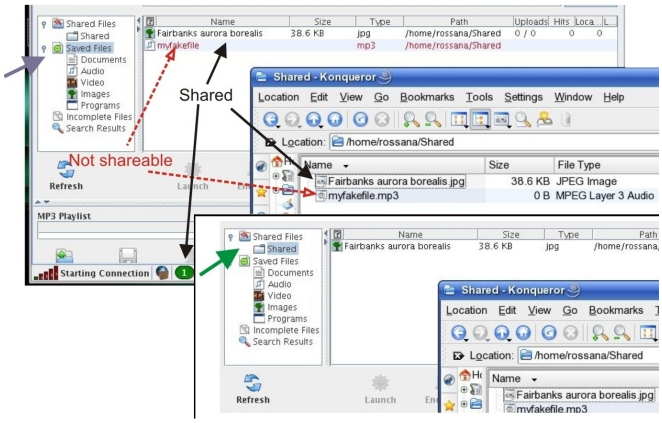

Figure 4: After modifying the source code, files showing an obvious spurious content pattern are automatically not shareable, even if placed in the shared folder. In the present case, myfakefile.mp3 has a suspicious size, namely it is too small to be a legitimate mp3.

tern from becoming shareable (unless the peer modifies the servent code). Different criteria can be adopted and customarily refined, e.g. based on the file size vs. file extension. Just to make an example, a 300 KB file claiming to be a movie cannot be legitimate. In many cases, advertisements spam and infected files are spread in tiny files, to make downloads faster and allow the malicious peer to upload a lot of them. A much more refined approach could even exploit something similar to a simple antivirus database, to run a quick scan through the file and search for specific suspicious patterns, before the files can be in the shareable pool. We have added a simple filtering features in Limewire code to prevent suspicious files from becoming shareable. As shown in Figure 4, even if a user attempts to share a file whose pattern matches a "junk" status, the servent will just not consider it and no query hits will be generated for that file. Of course the two improvements proposed above are far from representing robust and definitive solutions. However, their implementation is actually immediate and, in practice, it would mitigate the problem in a relevant number of cases. This strategy would also prevent junk traffic, such as query hits and query responses, to be generated in first place and thus it would improve not only security but also performance. Ultimately, the simplicity and ease of implementation of the many further possible improvements suggests that, if nothing has ever been done to address Gnutella problems, it may be the case that a security solution is not really sought by whoever is in charge of Gnutella specifications and development.

6. CONCLUSIONS

Due to the size reached by the Gnutella network, we believe that security solutions should be sought, to counteract the threat represented by malicious nodes, acting as ultrapeers and spreading polluted content. Additionally, the waste of bandwidth, due to such malicious traffic, represents a relevant performance issue. Drastic changes, such as switching to hybrid P2P architectures, are hard to achieve and even harder to be accepted by Gnutella community. Several reputation and incentive models have been proposed, but never really taken into account for official releases or even for alpha testing. However, other more immediate strategies are possible and fairly easy to integrate. We propose two slight modifications, one in the Gnutella protocol and the other directly implementable in any servents, to decrease the quantity of malicious nodes and of shared polluted content. Although such modifications represent only a slight improvement and not a real solution, they may still provide visible benefits from a security and performance point of view. We believe that a non-trivial point in the actual Gnutella system is whether there are commercial or strategic interests in maintaining the present situation unchanged, as suggested by the presence of corporation and Government entities voluntarily trying to pollute the Gnutella network.

7. REFERENCES

[1] http://www.limewire.org/forum/showthread.php?t=93.
[2] http://research.sunbelt-software.com/threatdisplay.aspx?threatid=55129.
[3] http://www.limewire.org/forum/showthread.php?t=1071.
[4] http://insecure.org/nmap.
[5] http://www.honeynet.org/papers/bots.
[6] http://www.limewire.org/forum/showthread.php?t=1168.
[7] http://forums.phoenixlabs.org/showthread.php?t=8938.
[8] Anonymously launching a ddos attack via the gnutella network. http://www.auscert.org.au/render.html?it=2404.
[9] Choosing reputable servents in a p2p network. http://seclab.dti.unimi.it/Papers/www02.ps.
[10] Exploiting the security weaknesses of the gnutella protocol. http://www.cs.ucr.edu/ csyiazti/courses/cs260-2/project/gnutella.pdf.
[11] Gnutella viruses weaker than email bugs, experts say. http://news.com.com/2100-1023_3-241440.html.
[12] http://forums.whirlpool.net.au/forum-replies-archive.cfm/533720.html.
[13] P2p: Is big brother watching you? http://www1.cs.ucr.edu/store/techreports/UCR-CS-2006-06201.pdf.
[14] Quantitative analysis of the gnutella network traffic. http://www1.cs.ucr.edu/store/techreports/UCR-CS-2004-04089.pdf.
[15] Query-flood dos attacks in gnutella. http://infolab.stanford.edu/ daswani/papers/p115-daswani.pdf.
[16] Security problems in p2p networks: A case study on the gnutella network. http://wwwcsif.cs.ucdavis.edu/ andrei/ECS235/ecs235_report.pdf.
[17] F. Cornelli, E. Damiani, S. D. Capitani, S. Paraboschi, and P. Samarati. *Implementing a Reputation-Aware Gnutella Servent*, volume 2376. Lecture Notes In Computer Science, Springer-Verlag, London, UK, 2002.
[18] D. Ferguson. *Trends and Statistics in Peer-to-Peer*. VP Engineering CacheLogic, 2006.
[19] S. H. Kwok, K. Y. Chan, and Y. M. Cheung. A server-mediated peer-to-peer system. *ACM SIGecom Exchanges*, pages 38–47, 2005.
[20] S. M. Lui, K. R. Lang, and S. H. Kwok. Participation incentive mechanisms in peer-to-peer subscription systems. *HICSS 2002*, pages 3925–3931, 2002.
[21] L. Sroura, A. Kayssia, and A. Chehab. Reputation-based algorithm for managing trust in file sharing networks. *Securecomm and Workshops*, pages 1–10, 2006.
[22] D. Stutzbach and R. Rejaie. *Capturing accurate snapshots of the Gnutella network*, volume 4. INFOCOM 2005, Proc. IEEE, 2005.