P2P ENTRAPMENT - INCRIMINATING PEER TO PEER NETWORK USERS

Written by have2Banonymous

27 September 2003

INTRODUCTION

If a user of peer to peer (P2P) networks is allegedly caught searching for, downloading, sharing or uploading contraband files such as copyright-protected music .mp3 files, they might mistakenly believe that their only option is to plea bargain with authorities.

However the P2P user, who the authorities are all too quick to brand as an offender, may actually be an innocent victim. It is possible for an attacker to exploit both the underlying design of P2P networks as well as implementation flaws in P2P applications in order to implicate another P2P user in behaviour deemed unacceptable by the authorities.

In the worst case scenario, an attacker can anonymously trick an innocent P2P user into downloading a contraband file from another user on the P2P network. If authorities participate in P2P networks in order to identify offenders, the innocent P2P user may have downloaded a contraband file from an authority.

This article will describe how a P2P user allegedly caught committing an offence relating to copyright violation, such as sharing/uploading/downloading/searching for contraband files, might not have been knowingly involved, or might not have been involved at all. A selection of the numerous ways to implicate P2P users in unacceptable behaviour will be provided in this article to illustrate that "evidence" of unacceptable behaviour obtained by the authorities can be extremely unreliable.

DEFINITIONS

The following definitions are provided for key words used throughout this article:

- *Authority* - an organisation with authority or pseudo-authority that has an interest in the activities of users on P2P networks.
- *Contraband* - files deemed inappropriate by an authority, for example copyright-protected music .mp3 files.
- *Apparent offender* - an innocent P2P user who appears to be engaging in behaviour deemed unacceptable by an authority.
- *Attacker* - a P2P user who turns an innocent P2P user into an apparent offender. The attacker could be someone with a vested interest or merely someone with too much spare time.

DISCUSSION

A peer to peer network places trust in each user to do the right thing. It is similar to a game of Chinese Whispers, where a message is passed from a source person to a destination person via other people in between who may or may not change the content of the message before propagating it to the next person in line.

In a typical (unencrypted) P2P network, the attacker would be located between the authority and the innocent P2P user, making it trivial for the attacker to modify packets on the P2P network in order to implicate the innocent P2P user as being an apparent offender.

In some cases however, there may be no P2P users between the authority and the apparent offender, who are connected directly via a TCP/IP connection. In these cases, the authority would place an even greater degree of trust in data (to be used as evidence) apparently originating from the apparent offender since there are no P2P users in between the authority and the apparent offender to tamper with the data. However, trusting the integrity of this data would be a mistake. Implementation flaws in P2P applications allow an attacker to falsify a range of "evidence" regardless of the network topology, including creating questionable data and making it appear to originate from another P2P user, thereby turning them into an apparent offender. Furthermore, an attacker can trick an innocent P2P user into downloading a contraband file from the attacker or a third party who may even be an authority. The easily abused trust placed in all P2P users to do the right thing combined with implementation flaws in P2P applications calls into question any "evidence" obtained by the authorities.

OVERVIEW OF EXAMPLES USED

To avoid legal issues involved with reverse engineering, this article will only provide examples from the Gnutella peer to peer (GP2P) network since the protocol is publicly documented, and a significant number of the applications used to interact with the GP2P network are open source. The selection of GP2P application flaws identified by the author of this article and presented here allow innocent GP2P users to be implicated in behaviour unacceptable to authorities.

The existence of these flaws could be considered of benefit to GP2P network users as a potential defence to prosecution by the authorities, since the integrity of the entire peer to peer system is called into question, and therefore so is any evidence obtained by the authorities. As a result, the author of this article will provide only the minimum number of example implementation flaws required to support statements made in the article.

Implementation flaws exist in the range of open source GP2P applications, though only the popular Gnucleus GP2P application will be used as an example throughout this article since the source code is relatively well laid out and easy to read. Proprietary GP2P and P2P applications and the undocumented protocol used by some P2P applications will not be covered in this article, though they have their share of serious flaws that taint "evidence". For example, vulnerabilities in the network protocol used by the KaZaA, Grokster, iMesh and Morpheus P2P applications have been identified which resulted in identity spoofing [1], denial of service [2], and more recently allowed an attacker to execute arbitrary code on a victim's computer [3].

SCENARIOS

The following scenarios describe where an authority (mistakenly) believes they have obtained evidence of an apparent offender engaging in unacceptable behaviour. The scenario description details what an observer would notice rather than what actually happened:

- An apparent offender performs a search on a keyword such as *metallica*, or returns a search result declaring that they are sharing files with *metallica* in their filename, available for anyone to download.
- An apparent offender downloads a file such as *metallica_enter_sandman.mp3* directly from another GP2P user who happens to be an attacker.
- An apparent offender downloads a file such as *metallica_enter_sandman.mp3* directly from another GP2P user who happens to be an authority.

These attack scenarios are listed roughly in order of increasing attack sophistication and increasing degree of incrimination of the apparent offender.

In the majority of the attack scenarios, the attacker does not need to store contraband files on their own hard disk or transfer contraband files to another GP2P user in order to incriminate them. This means that the attacker is not infringing copyright, and that the attacker requires negligible bandwidth per attack. Also, if the attacker has at least one GP2P user insulating them from both the targeted innocent GP2P user and the authority, the attacker can remain anonymous since they do not need to make a direct connection (thereby revealing their IP address) to either the innocent GP2P user or the authority.

SCENARIO 1: MODIFYING SEARCH REQUESTS AND SEARCH RESULTS IN TRANSIT

In a typical GP2P network configuration, the attacker is located between other GP2P users and the authority. The attacker can trivially breach the implicit trust placed in every user of the GP2P network to "do the right thing" by modifying data passing through them to turn innocent GP2P users into apparent offenders.

For example, if an innocent GP2P user performs a search on the string *clipart*, the attacker can modify the search string to *metallica* before propagating the search request for it to eventually reach an authority. The authority may mistakenly believe that a GP2P user x hops away from them performed a search for contraband files and is therefore an apparent offender.

An attacker can similarly modify a search result in order to implicate an innocent GP2P user, though to further incriminate the innocent GP2P user, the attacker can also place the innocent GP2P user's IP address in the search result packet. An authority is more likely to take action on search results containing contraband files rather than search requests for contraband files, since search results contain the IP address of the apparent offender who apparently has the contraband files on their hard disk.

SCENARIO 2: SPOOFING THE ORIGINATOR OF SEARCH REQUESTS AND SEARCH RESULTS

The hop count field inside a search request is the number of GP2P users the search request has propagated through. The originator of a search request sets the hop count to be 0 before sending it onto the GP2P network. Each GP2P user increments the hop count field before further propagating the search request.

If an authority receives a search request for *metallica* from another GP2P user and the hop count is set to 0, it is logical for the authority to assume that the directly connected GP2P user who passed the search request to the authority is an apparent offender because they originated the search request for a contraband file. Furthermore, it is logical for the authority to assume that the search string had not been modified (e.g. from *clipart* to *metallica*) since there are no GP2P users between the authority and the search request originator. The authority even knows the apparent offender's IP address due to the direct TCP/IP connection. However, these assumptions are false due to implementation flaws in GP2P applications.

An attacker can incriminate an innocent GP2P user as being the originator of a search request for a contraband file by passing them a search request for *metallica* with the hop count field set to 255. If the innocent user is using a flawed GP2P application, the hop count value is incremented from 255 to 0 and the search request is propagated.

An attacker can similarly generate a search result in order to implicate an innocent GP2P user, though to further incriminate the innocent GP2P user, the attacker can also place the innocent GP2P user's IP address in the search result packet. An authority is more likely to take action on search results containing contraband files rather than search requests for contraband files, since search results contain the IP address of the apparent offender who apparently has the contraband files on their hard disk.

The following extract of relevant code from an older version of Gnucleus illustrates that it would increment and wraparound the hop count from 255 to 0, and then test to determine if the hop count was too large before propagating the packet. This implementation flaw has already been corrected in Gnucleus by testing the hop count before incrementing it.

Note that `[....]` represents irrelevant code which has been deleted for clarity.

From an old version of the Gnucleus source code file Packet.h:

```
struct packet_Header          // Size 23
{
     [....]

     BYTE  Hops;                          // 18

     [....]
};
```

From an old version of the Gnucleus source code file GnuNode.cpp:

```
bool CGnuNode::InspectPacket(packet_Header* packet)
{
     [....]

     // Increment hops
     packet->Hops++;

     [....]

     // If packet has hopped more than 7 times kill it
     if(packet->Hops > MAX_TTL)
          return false;

     [....]

     return true;
}
```

SCENARIO 3: RENAMING A CONTRABAND FILE TO MATCH INCOMING SEARCH REQUESTS

This scenario is included only for the sake of completeness.

An attacker can rename a contraband file stored on their hard disk to match search requests performed by other GP2P users. For example, if an innocent GP2P user performs a search on the string *church choir hymns*, an attacker could return a search result stating that they have a file called *church_choir_hymns_highest_quality.mp3*, rename the contraband file to this name, and hope that the innocent GP2P user downloads the misnamed contraband file thereby turning them into an apparent offender. GP2P applications typically automatically share the contents of their download directory for other GP2P users (including an authority) to download, so the introduction of the contraband file and sharing the file may occur without the apparent offender's knowledge. The apparent offender may listen to the song at a later stage and determine that it is actually a contraband file, however the damage has already been done and the apparent offender may be subject to the authorities' wrath.

There are several drawbacks to this attack scenario. The attacker loses their anonymity since file transfers are performed via a direct TCP connection, so the apparent offender knows the attacker's IP address. Also, the attacker has to host the contraband file on their computer, leaving them open to prosecution by the authorities. Finally, it is not a scalable attack since the attacker's bandwidth is consumed by the attack.

SCENARIO 4: IMPERSONATING ANOTHER GP2P USER

Each user on the GP2P network has a self-assigned randomly generated unique identifier known as a GUID. Failing to also use more appropriate identification such as IP addresses allows an attacker to impersonate another GP2P user on the network.

For example, an innocent GP2P user might perform a search, receive a search result from a second innocent GP2P user, and attempt to connect to the GP2P user with the desired file in order to download the file. The download attempt will fail if the user with the file is behind a firewall. In this case, the user who wants the file sends a push request through the GP2P network asking the user with the file to establish a connection to them and push (transfer) the file.

When an attacker receives a push request, instead of propagating it, they can impersonate the user who the push request was intended for and transfer a contraband file (potentially even with a contraband-sounding filename....) to the originator of the push request thereby turning them into an apparent offender. GP2P applications typically automatically share the contents of their download directory for other GP2P users (including an authority) to download, so the introduction of the contraband file and sharing the file may occur without the apparent offender's knowledge.

Instead of merely relying on a self-assigned "unique" ID value for identifying users on the GP2P network, GP2P applications could perhaps also verify that the IP address of the GP2P user pushing them a file is the same IP address of the GP2P user who originally claimed to have the desired file (i.e. who the push request was actually intended for). Identification using IP address is more accurate than using a self-assigned value, though problems may arise when users are behind a firewall performing network address translation.

The following extract of relevant code from the current version of Gnucleus illustrates that the self-assigned "unique" GUID value is used to determine if the GP2P user who is pushing the file is the same GP2P user who originally claimed to have the desired file.

From the Gnucleus source code file GnuSock.cpp (CVS) version 1.18 dated 10 June 2003:

```
void CGnuSock::OnReceive(int nErrorCode)
{

     [....]

     // Server Requesting to Push a file to us

          [....]

          // Get Server ID of client giving us the file
          int Front = m_Handshake.Find(":") + 1;
          int End   = m_Handshake.Find("/");
          CString PushGuid  = m_Handshake.Mid(Front, End - Front);
          PushGuid.MakeUpper();

          [....]

          // Find the download that requested the push
          for(int i = 0; i < m_pTrans->m_DownloadList.size(); i++)
          {

               [....]

               if(EncodeBase16((byte*) &p->m_Queue[j].PushID, 16) == PushGuid)
               {
                    Found = true;
                    break;
               }

          [....]

}
```

SCENARIO 5: TRICKING AN INNOCENT USER INTO DOWNLOADING CONTRABAND FROM AN AUTHORITY

Readers of this article will realise by now that attempting to identify offenders via searches, search requests, push requests, and the sharing of contraband files is fraught with error and inaccuracy. When authorities realise this, they might decide to host contraband files themselves and identify offenders as the users who download the contraband files - the authority even knows the apparent offender's IP address due to the direct TCP/IP connection used to download files. This might appear to be the most accurate method for authorities to identify offenders, however due to implementation flaws in GP2P applications, this method is also fraught with error and inaccuracy.

An attacker can incriminate an innocent GP2P user by tricking them into downloading contraband files from an authority - or if there are no authorities sharing contraband files, from another GP2P user in order for the innocent GP2P user to automatically share the unknowingly downloaded contraband files to be subsequently found by authorities searching the network for contraband files.

In this attack scenario, the attacker can remain anonymous since they do not need to make a direct connection to either the innocent GP2P user or the authority. Also, the attacker does not need to store any contraband files on their own hard disk or transfer them to the innocent GP2P user, resulting in the attacker requiring negligible bandwidth per attack.

The GP2P protocol specifies that files referred to in search results, push requests and direct downloads/uploads should be represented by a numerical index which is another way of addressing a file with a given file name. The developers of GP2P applications evidently thought that using both a file name and a file index to refer to exactly the same file was somewhat redundant, which results in the flaw described in this attack scenario.

The attacker could perform a search such as *metallica enter sandman*, and make a note of the file name, file index, and IP address of GP2P users who returned a search result indicating they had a matching contraband file. Considering Metallica's history [4], most if not all of these GP2P users could represent an authority (assuming the authorities decide to share contraband files in order to identify offenders). An example search result would contain the information *The GP2P user with IP address 123.123.123.123 has a file with index number 17 and a name of metallica_enter_sandman.mp3*.

The attacker then waits until an innocent GP2P user performs a search, for example on the innocent search string *church choir hymns*. The attacker sends them a search result such as *The GP2P user with IP address 123.123.123.123 has a file with index number 17 and a name of church_choir_hymns_HIGHEST_QUALITY.mp3*. The attacker would not propagate the search request thereby limiting the number of results returned, and increasing the likelihood of the file in the search result provided by the attacker being downloaded.

The innocent GP2P user decides they want to download the file *church_choir_hymns_HIGHEST_QUALITY.mp3* so they select the file name displayed by their GP2P application and click the download button. Their GP2P application connects to the GP2P user with IP address 123.123.123.123 (e.g. the authority) and requests the desired file by both file index number and file name. If the authority is using one of the several GP2P applications that consider having both a file index and file name is redundant, and therefore ignore the requested file name, the authority's GP2P application will send the file *metallica_enter_sandman.mp3* to the innocent GP2P user, turning them into an apparent offender.

The file index represents the actual file, but the file index is transparent to both the apparent offender and the authority. Therefore, as far as the apparent offender is concerned, they are downloading an innocent file called *church_choir_hymns_HIGHEST_QUALITY.mp3*. As far as the authority is concerned, the apparent offender requested and is downloading the contraband file *metallica_enter_sandman.mp3* from the authority. The authority may believe with certainty that they could successfully prosecute the apparent offender for downloading contraband files. The direct connection used to transfer the file results in the apparent offender revealing their IP address and losing all anonymity. The authority's GP2P application will show them the apparent offender's IP address and the file that the authority sent to the apparent offender.

The following extract of relevant code from the current version of Gnucleus illustrates that it is one of the several GP2P applications that ignore the file name of a requested file, and select the file to send the requesting user based on the requested file index.

From the Gnucleus source code file GnuUploadShell.cpp (CVS) version 1.18 dated 10 June 2003:

```
void CGnuUploadShell::ParseRequest(CString Handshake)
{

     [....]

     else if (LowRequestURI.Left(5) == "/get/")
     {
          CString IndexString = m_RequestURI.Mid(5,
m_RequestURI.Find("/", 5) - 5);

          m_Index = atoi(IndexString);
          m_Name  = m_RequestURI.Mid(5 + IndexString.GetLength() + 1);
     }

     [....]

     // Get Handle to requested file, now that all needed data is collected
     if(m_Index)
     {
          CString UploadPath = m_pShare->GetFilePath(m_Index);
          m_Name = m_pShare->GetFileName(m_Index);

     [....]

}
```

It can be seen that *ParseRequest* retrieves the file name (stored as *m_Name*) and the file index (stored as *m_Index*) from the user's request, and then proceeds to overwrite *m_Name* with the name of the file stored at index *m_Index*. A suggested fix for this example implementation flaw in Gnucleus (and other similarly affected GP2P applications) is to check that the file index and file name in file transfer requests actually refer to the same file.

This type of implementation flaw is one of the more interesting flaws which the author of this article has identified across a range of GP2P applications, since the flaw is in the server component of the GP2P applications. A GP2P user is vulnerable to this attack not because their software has a flaw, but because another user on the network is using flawed software. The author of this article has informed the programmer of the GP2P application used in examples throughout this article, and anticipates a fix to be released by the time this article is publicly released. However, GP2P users using **any** GP2P application will still be vulnerable to the attack described here as long as there is at least one user on the GP2P network still running an unpatched version of Gnucleus or one of the other similarly flawed GP2P applications.

A GP2P user (using **any** GP2P application) who does not want to become a victim to this attack vector could stop using the GP2P network until every GP2P user with flawed software has upgraded to a patched version. Alternatively, perhaps a possible solution would be to incorporate code into every GP2P application so that when a file is about to be downloaded, the version of the GP2P application with the file is checked (via the HTTP Server field) and the file transfer aborted if it is found to be a flawed version of one of the affected GP2P applications.

While reviewing this article, the developer of Gnucleus kindly contributed the observation that instead of using the index/filename combination to identify a file, a cryptographic hash (typically using the SHA1 algorithm) may be used, and searches and download requests can be performed using a hash value as the identifying criteria. Furthermore, an attacker can incriminate an innocent GP2P user by providing them with a search result that references a contraband file on an authority's computer. The search result would contain the hash value of the file, but a non-contraband sounding filename in order to entice the innocent GP2P user into downloading the file. When the innocent GP2P user downloads (and subsequently shares) the file from the authority, the download request is based on the hash value, and the (mismatching) file names are ignored. Even if the innocent GP2P user downloaded the contraband file from a GP2P user who was not an authority, they will still become an apparent offender if authorities search the GP2P network for hash values of known contraband files, since the contraband file on the innocent GP2P user's computer will match regardless of its filename.

CONCLUSION

Authorities are increasingly claiming that P2P users are infringing copyright laws. The authorities are handling the alleged infringements by threatening the apparent offender's Internet Service Provider (ISP) with a subpoena forcing the ISP to provide the apparent offender's details to the authority [5], and threatening the apparent offender with a lawsuit [6] [7] [8]. New legislation is being drafted to allow for fines of up to $250000 and even to send apparent offenders to prison for up to five years [9]. Furthermore, authorities have clearly indicated a desire to attack [10] and destroy [11] computers belonging to apparent offenders.

However, this article has indicated that in order for the authority's claim to have merit and for their retaliatory action to be justified, they would need to prove that the apparent offender was an actual offender. This would require the authority proving that there were no malicious users on the P2P network at the time, that the apparent offender's P2P application had no implementation flaws, and (as shown in Scenario 5) that none of the numerous other P2P applications that could have been connected to the network at the time had implementation flaws.

The author of this article is not a lawyer, though it appears that implementation flaws in P2P applications (such as the small sample of flaws detailed in this article, along with other flaws not in the public domain), coupled with the inherent trust placed in a network which is untrustworthy (if not downright hostile when users such as the author of this article are involved), make it nearly impossible for authorities to prove that a P2P user knowingly committed an offence, or even committed an offence at all. Obviously however, if the apparent offender's computer is seized and inspected by an authority and there are contraband files with contraband-sounding filenames in the directory *c:\pirated_mp3s*, there is an indication of a guilty mind and/or intent and it may be time for the (not-so-apparent) offender to consider a plea bargain.

Authorities recently filed a lawsuit against a 66 year old woman accusing her of illegally sharing hundreds of songs including rap music via the KaZaA P2P network, and threatened fines of up to $150000 for each song [12]. Instead of succumbing to the scare tactics of the authorities and settling out of court, she disputed the claim and questioned the evidence of her alleged misbehaviour. The authorities subsequently dropped the lawsuit.

A GP2P user who the authorities claim is an apparent offender should also consider disputing the authority's "evidence". If the user allegedly searched for or returned matches for a contraband file, they could claim that an attacker generated or manipulated network communication data in order to incriminate them. If they allegedly downloaded a contraband file (e.g. from an authority), they could claim plausible deniability - an attacker must have tricked them into doing it in order to incriminate them. Furthermore, their GP2P application automatically shared the contraband files which they didn't even know they had, for other users (including an authority) to download.

In a nutshell, alleged offenders could show this article to the authorities and claim that they were victimised by an anonymous user on the network - and they probably were.

REFERENCES

[1] *FastTrack P2P Technology Message Service Identity Spoofing Vulnerability*, 17 Feb 2002, http://www.securityfocus.com/bid/4121/discussion/

[2] *FastTrack P2P Technology Message Service Denial Of Service Vulnerability*, 17 Feb 2002, http://www.securityfocus.com/bid/4122/discussion/

[3] *FastTrack P2P Supernode Packet Handler Buffer Overflow Vulnerability*, 26 May 2003, http://www.symantec.com/avcenter/security/Content/7680.html

[4] *Napster, universities sued by Metallica*, 13 Apr 2000, http://news.com.com/2100-1023-239263.html?legacy=cnet

[5] *RIAA wins battle to ID Kazaa user*, 21 Jan 2003, http://news.com.com/2100-1023-981449.html?tag=nl

[6] *Campus file swappers to pay RIAA*, 1 May 2003, http://news.com.com/2100-1027-999332.html

[7] *Line'em up! RIAA to sue thousands*, 25 Jun 2003, http://www.theregister.co.uk/content/6/31434.html

[8] *Some Defendants Stunned After RIAA Files Lawsuits*, 8 Sep 2003, http://www.local6.com/technology/2463549/detail.html

[9] *House proposal targets file swappers*, 17 Jul 2003, http://news.com.com/2100-1028-1026715.html

[10] *Hollywood hacking bill hits House*, 25 Jul 2002, http://news.com.com/2100-1023_3-946316.html

[11] *Senator endorses destroying computers of illegal downloaders*, 17 Jun 2003, http://www.securityfocus.com/printable/news/5865

[12] *File-sharing suit against 66-year-old sculptor dropped*, 24 Sep 2003, http://www.cnn.com/2003/LAW/09/24/tech.lawsuit.ap/index.html