Towards the Detection of Encrypted BitTorrent Traffic through Deep Packet Inspection

David A. Carvalho, Manuela Pereira, and Mário M. Freire

IT-Networks and Multimedia, Department of Computer Science, University of Beira Interior
Rua Marquês d'Ávila e Bolama, P-6201-001 Covilhã, Portugal
{david, mpereira, mario}@di.ubi.pt

Abstract. Nowadays, peer-to-peer file sharing applications are very popular, occupying the traffic volume generated by these applications a large percentage of the global network traffic. However, peer-to-peer traffic may compromise the performance of critical networked applications or network-based tasks in institutions, being need, in some cases, to block such traffic. However, this task may be particularly difficult, namely when that peer-to-peer traffic is encrypted and therefore being difficult to block. This paper presents a contribution towards the detection and blocking of encrypted peer-to-peer file sharing traffic generated by BitTorrent application. The proposed method is based on deep packet inspection and makes use of Snort, which is a popular open source network-based intrusion detection system. Experiments have been carried out to validate the proposed method as well as its accuracy.

Keywords: Peer-to-peer file-sharing applications, traffic identification, deep packet inspection, traffic monitoring, peer-to-peer content filtering and management.

1 Introduction

As the Transmission Control Protocol/Internet Protocol (TCP/IP) architecture becomes the dominant architecture for applications, contents and services, these are gradually migrating from the client-server paradigm to the peer-to-peer (P2P) paradigm. Recently, P2P systems have received a great amount of interest as a promising scalable, reliability and cost-effective solution for multimedia content sharing and distribution. Nowadays, P2P file sharing applications such as BitTorrent, Vuze, eMule, Limewire and GTK-Gnutella, among others, have achieved a tremendous success in the past few years. In this kind of systems, as more clients join, more resources are made available, which speeds up downloading and therefore may attract more clients. The traffic generated by P2P applications has become a dominating part of the global Internet traffic, being estimated to be around 60-70% [1], [2]. Without suitable network management, the traffic generated by this kind of applications may compromise the performance of critical networked applications or network-based tasks in institutions.

The traffic generated by first generation P2P applications was relatively easy to detect due to the fact that these applications used well-defined port numbers. However, nowadays, the traffic generated by P2P applications may be very difficult to detect because P2P applications may use different port numbers to escape the detection, do not use the default service port or use port 80 assigned for HTTP traffic [3], [4]. Besides, they may use obfuscation options where the traffic is encrypted, making therefore very difficult its detection. On the other side, link speeds in LANs are reaching speeds in the Gigabit per second range, which may become the detection infeasible since the processing speed cannot match the line speed and capturing every packet may pose severe requirements in terms of processing and memory capacities.

As detection and classification mechanisms evolve, P2P evasive techniques also evolve making traffic detection and classification a very difficult task. Recently, some approaches have been proposed to detect P2P applications. These techniques may be classified into two main categories [2], [5]: based on payload inspection or signature-based detection; and based on flow traffic behaviour. Deep packet inspection methods inspect the packet payload to locate specific string series, which are called signatures that identify a given characteristic, a given protocol or a given application, where as methods based on traffic behaviour attempt to detect and classify possible protocols or applications without looking into the payload contents.

Some approaches have been proposed for traffic identification using behaviour-based methods. The method based on transport-level connection patterns relies on two heuristics for P2P traffic classification: 1) it involves the simultaneous use of TCP and UDP by a pair of communicating peers and 2) regarding the connection patterns for (IP, port) pairs, the number of distinct ports communicating with a P2P application on a given peer will likely match the number of distinct IP addresses communicating with it [2]. The behavioural method based on entropy reported in [5] requires the evaluation of the entropy of the packet sizes in a given time window and works on-the-fly. Several approaches requiring the analysis of some fields of the header of TCP or IP packets for flow-based P2P traffic detection have been proposed based on machine learning [4], [6], support vector machines [7], [8], and neural networks [9]. This kind of methods may be used for high-speed and real-time communications with encrypted traffic or unknown P2P protocols. The main drawback is the possible lack of accuracy in the identification of P2P traffic.

Methods based on the payload inspection may be accurate, but may lead to network performance degradation under low latency or high-speed operations. Besides, they may not be useful when payload is encrypted or for new P2P protocols or in the cases where legal or privacy issues do not allow their use [2], [3].

Nowadays, one of the main challenges regarding P2P file-sharing traffic detection is concerned with the on-line detection of encrypted traffic under high-speed and real-time communications, where fast P2P traffic identification is required in order to avoid network performance degradation. To address this issue, we are developing an hybrid method for on-line P2P traffic identification, including encrypted traffic, based on a combination of the flow behaviour method using the evaluation of the entropy of the packet sizes in a given time window reported in [5] and deep packet inspection, to complement the first approach, in order to reduce false positives and false negatives. In this paper, we report the development of signatures for deep packet inspection for BitTorrent application as a piece of the puzzle.

2 Methodology for P2P Traffic Detection Using Signatures

The methodology used for the detection of P2P file sharing traffic makes use of an open source and widely used intrusion detection system, called Snort [10], which runs over Windows, Linux/Unix or MAC operating systems and is adapted from the methodology presented in [11]. The signatures of payloads of packets to identify are expressed in terms of Snort rules. The identification of Snort rules for detection of packets generated by a given P2P application, requires the execution of the following steps: i) identification of signatures associated with a given P2P protocol that can be revealed through the analysis of the payload of an IP packet; ii) writing Snort rules incorporating these signatures in order to detect all packets with that particular signatures.

Using this methodology, we developed Snort rules for the detection of P2P traffic generated by BitTorrent application, being paid particular attention to the detection of encrypted traffic. The identification of signatures associated with the packets generated by this application was made manually through the observation of repetitive patterns in the sequence of packets generated by BitTorrent, even with encryption of the payload. Those common patterns observed in the payload were then used to manually write the rules for the detection engine, which allow the identification of a given packet that contains that particular pattern in a particular position or after a given offset in its payload.

3 Experimental Setup

The experimental testbed includes several machines with different characteristics and running different operating systems at our research lab (NMCG Lab). All outgoing and incoming traffic for servers, workstations and laptops used at this lab is controlled by a computer running Smoothwall Express 3.0, which is a network administration specific Linux distribution, from SmoothWall Open Source Project [12], providing Internet security and Web filtering products. Although the SmoothWall Express 3.0 version has not the same capabilities as the commercial products, it enables powerful extended possibilities at very low cost, which was the main reason for its choice during the NMCG lab planning and deployment. This Lab has 24 8P8C sockets connecting to an Enterasys C2H128-48 switch through UTP Ethernet Enhanced Cat5 cabling. The switch then connects to the network backbone device of the Department of Computer Science building, an Enterasys E7 just one floor above, via an optical fibre uplink, which in turn, connects to the rest of University. All external communication with the University is made using an Enterasys SSR main router, located at the Center for Computer Science of the University.

To run P2P software, it is not usually necessary a great computing power. Usually, the most important feature is the size of the hard disk. When dealing with P2P file sharing programs, transferred files can easily reach a few gigabytes, since they are mostly movies, videos, music albums, games, etc. Real time network monitoring requires a lot more of memory and CPU. Therefore there were used more recent machines for the most critical applications, like the traffic classifier SNORT [10], or the analysis engine BASE [13] or even the packet analyzer Wireshark [14]. As for

Table 1. Characteristics of hardware and software used for P2P traffic detection

Type	Operating System	CPU	RAM	Software
Workstation	Fedora 9	Core 2 Duo 2.66 GHz	1 GB	Snort, Wireshark, BASE, Barnyard, Gtk-Gnutella, Livestation
Workstation	Windows XP SP3	Pentium III 800 MHz	512 MB	BitTorrent, eMule, aMule, Limewire, Livestation, TVU Player
Laptop	Windows Vista SP1 / Fedora 10	Core 2 Duo 2.4 GHz	3 GB	Wireshark, eMule, TVUPlayer, Livestation
Laptop	MAC OS X (10.5)	Power PC G4 1GHz	769 MB	Vuze, Livestation, TVUPlayer

running P2P software, pretty old machines were used, since they were mainly used for this purpose. Main characteristics of the hardware and the software used for the experiments are shown in Table 1.

In all practical experiences reported here, Snort was forced to analyse other network traffic than P2P, like HTTP, Windows Remote Desktop Connection (RDC), SSH, etc. In fact, this was quite worthy, since it enabled the testbed to run in similar circumstances of those of deployed P2P classifiers, which also have to deal with network traffic generated by a vast number of applications and then to correctly identify P2P among it.

4 BitTorrent Application

4.1 Application Details

BitTorrent application version 6.1.2 was configured so that it would only allow bi-directional encrypted connections, i.e., both outgoing and incoming traffic had to be encrypted, so that communication was possible with other BitTorrent clients (applications). Nowadays, users tend to use these settings to avoid being throttled or blocked by their ISPs. As a consequence, there are not so many sources available to download if one does not use the *Forced* setting for outgoing encrypted traffic, since other clients are mostly configured to deny *legacy connections*, thus not allowing unencrypted connections. These settings are configured under the menu Options → Preferences → BitTorrent → Protocol Encryption. To only use encrypted connections, the Outgoing combo box must be set with the value *Forced* and *Allow incoming legacy connections* must be unchecked.

In all of the following tests, the setting *Ask the tracker scrape information*, also under Options → Preferences → BitTorrent → was always checked. This enables the client to obtain newer peers and provide statistics about their availability. Although it is not mandatory, specially if other mechanisms are used to obtain peer information like the DHT, it can be useful to maintain updated records about resource availability. It is important to notice that if this setting is unchecked, there is no traffic for *BitTorrent tracker request* and, consequently, the rules for detecting it are never triggered. In this work, it was kept checked for studying the frequency of communications to the tracker.

4.2 SNORT Rules and Experiments with Encrypted Traffic

The first two tests were conducted with the previous mentioned settings and with DHT disabled, so that BitTorrent would not generate too much control traffic, making it harder to detect. The following rules were triggered and corresponding test results are provided in Table 2.

Snort Rule 1000301. Rule for detection of traffic generated by BitTorrent application.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P BitTorrent
outbound announce request"; flow:to_server,established; content:"GET";
offset:0;depth:4; content:"/announce"; distance:1; content:"info_hash=";
offset:4; content:"event=started";offset:4; classtype:policyviolation;
sid:1000301; rev:1;)
```

Snort Rule 1000305. Rule for detection of traffic generated by BitTorrent application.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"LocalRule:P2P
BitTorrent outbound - tracker request"; flow:to_server,established;
content:"GET"; offset:0; depth:4; content:"/scrape"; distance:1;
content:"info_hash="; offset:12; content:"User-Agent:";
offset:80;classtype:policy-violation; sid:1000305; rev:1;)
```

Table 2. Characteristics of experiences and detection results for BitTorrent application

Date	Start	End	Packets	Bytes	P2P Downloaded	P2P Uploaded	Rule-Count
17-01-2009	20:34	21:58	280791	107825488	22 MB	18.4 MB	1000301-1 1000305-1
27-01-2009	21:31	21:44	23175	10546443	1.2 MB	3.0 MB	1000301-1 1000305-1

So, even with DHT disabled, the above two snort rules for TCP traffic are frequently triggered. In this case, it happened only once, due in part to the small amount of BitTorrent traffic. In the following tests, one can confirm a greater occurrence of them. Once again it is important to emphasize, that if the *Ask the tracker scrape information* was unchecked, rule 1000305 would never be triggered at all. For the next tests (see Table 3), four rules were introduced. They refer to DHT traffic, and use the UDP unlike the previous ones. Each of the next sets of the first two and last two rules could be combined into a single one. The only advantage in specifying them independently is that it allows distinguishing incoming and outgoing traffic. As one can easily see, enabling the useful DHT feature allows the successfully identification of UDP traffic for trackerless requests and trackerless responses.

Snort Rule 1000306. Rule for detection of traffic generated by BitTorrent application.

```
alert udp $HOME_NET any -> $EXTERNAL_NET any (msg:"LocalRule:P2P
BitTorrent UDP - BitTorrent outgoing DHT for trackerless comunication
request (d1:ad2:id20)"; content:"d1:ad2:id20";
nocase;depth:11;classtype:policy-violation; sid:1000306; rev:2;)
```

Snort Rule 1000307. Rule for detection of traffic generated by BitTorrent application.

```
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"LocalRule:P2P
BitTorrent UDP - BitTorrent incoming DHT for trackerless comunication
request (d1:ad2:id20)"; content:"d1:ad2:id20";
nocase;depth:11;classtype:policy-violation; sid:1000307; rev:3;)
```

Snort Rule 1000308. Rule for detection of traffic generated by BitTorrent application.

```
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"LocalRule:P2P
BitTorrent UDP - BitTorrent incoming DHT for trackerless comunication
response (d1:rd2:id20)"; content:"d1:rd2:id20";
nocase;depth:11;classtype:policy-violation; sid:1000308; rev:3;)
```

Snort Rule 1000309. Rule for detection of traffic generated by BitTorrent application.

```
alert udp $HOME_NET any -> $EXTERNAL_NET any (msg:"LocalRule:P2P
BitTorrent UDP - BitTorrent outgoing DHT for trackerless comunication
response (d1:rd2:id20)"; content:"d1:rd2:id20";
nocase;depth:11;classtype:policy-violation; sid:1000309; rev:3;)
```

Table 3. Characteristics of an experience and its detection results for BitTorrent application

Date	Start	End	Packets	Bytes	P2P Downloaded	P2P Uploaded	Rule-Count
01-02-2009	23:01	23:21	71783	46023309	15 MB	6.1 MB	1000301-3 1000305-2 1000306-1562 1000307-689 1000308-24 1000309-30

Two additional rules were triggered during the tests on the BitTorrent application. They are available at [15] and are listed bellow.

Snort Rule 2008581. Rule for detection of traffic generated by BitTorrent application [15].

```
#http://www.emergingthreats.net/rules/emerging-p2p.rules
#By David Bianco
alert udp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET P2P BitTorrent DHT
ping request"; content:"d1n:ad2n:id20n:"; depth:12; nocase; threshold:
type both, count 1, seconds 300, track by_src; classtype:policy-
violation; reference:url,wiki.theory.org/BitTorrentDraftDHTProtocol;
sid:2008581; rev:1;)
```

Snort Rule 2008584. Rule for detection of traffic generated by BitTorrent application [15].

```
alert udp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET P2P BitTorrent DHT
get_peers request"; content:"d1n:ad2n:id20n:"; nocase; depth:12;
content:"9n:info_hash20n:"; nocase; distance:20; depth:14;
content:"e1n:q9n:get_peers1n:"; nocase; distance:20; depth:17;
threshold: type both, count 1, seconds 300, track by_src;
classtype:policy-violation;
reference:url,wiki.theory.org/BitTorrentDraftDHTProtocol; sid:2008584;
rev:1;)
```

Rule 2008581 is identical to the locally developed 1000306. They share some of their content, more exactly *d1:ad2:id20*. Even though, rule 1000306 triggered 614 times against a single one of 2008581. With these additional rules included and also enabling the DHT features, the results presented in Table 4 were obtained.

Table 4. Characteristics of an experience and its detection results for BitTorrent application

Date	Start	End	Packets	Bytes	P2P Downloaded	P2P Uploaded	Rule-Count
03-02-2009	20:47	20:59	20434	8642013	147.1 KB	3.4 MB	1000301-3 1000305-3 1000306-614 1000307-222 1000308-17 1000309-11 2008581-1 2008584-1

Another test was conducted in the same circumstances than the previous, but generating a bit more traffic. For this, it was select a torrent file for a drama movie released in 2008. The results obtained are listed in Table 5.

Table 5. Characteristics of an experience and its detection results for BitTorrent application

Date	Start	End	Packets	Bytes	P2P Downloaded	P2P Uploaded	Rule-Count
07-02-2009	19:53	22:57	231536	134571450	63.5 MB	46.7 MB	1000301-2 1000305-2 1000306-8423 1000307-4258 1000308-57 1000309-31

As one can see, rules 1000306, 1000307, 1000308 and 1000309 are triggered much often than 1000301 and 1000305. This is because when DHT is enabled, peers communicate frequently with each other to check for data and peer availability. As for rule 1000301, it is only triggered when a peer tells another that it is interested in some file shared by it and this usually occurs only just before beginning the download of another chunk. If the scrape feature is disabled, through the Ask the tracker scrape information option, rule 1000305 is not triggered at all, since communication with the tracker with the scrape content does not occur.

5 Conclusions

This paper describes the use of a deep packet inspection method based on signatures coded as SNORT rules for detection of encrypted P2P traffic generated by BitTorrent application. Several lab experiments were carried out to validate the proposed method and to evaluate its accuracy. As a future work, we intend to integrate this method as part of a hybrid method that also considers a flow-based method based on entropy [5] to work under high speed (1-10 Gbps) and low latency.

Acknowledgments. This work has been partially funded by Portuguese Fundação para a Ciência e a Tecnologia through TRAMANET Project contract PTDC/EIA/73072/2006.

References

1. PeerApp: Comparing P2P Solutions (2007), `http://www.peerapp.com/docs/ComparingP2P.pdf`
2. Madhukar, A., Williamson, C.: A Longitudinal Study of P2P Traffic Classification. In: 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006), pp. 179–188. IEEE Press, New York (2006)
3. Guo, Z., Qiu, Z.: Identification Peer-to-Peer Traffic for High Speed Networks Using Packet Sampling and Application Signatures. In: 9th International Conference on Signal Processing (ICSP 2008), pp. 2013–2019. IEEE Press, New York (2008)
4. Liu, H., Feng, W., Huang, Y., Li, X.: A Peer-To-Peer Traffic Identification Method Using Machine Learning. In: International Conference on Networking, Architecture, and Storage (NAS 2007), pp. 155–160. IEEE Press, New York (2007)
5. Gomes, J., Inacio, P., Freire, M., Pereira, M., Monteiro, P.: Analysis of Peer-to-Peer Traffic Using a Behavioural Method Based on Entropy. In: IEEE International Performance, Computing and Communications Conference (IPCCC 2008), pp. 201–208. IEEE Press, New York (2008)
6. Soysal, M., Schmidt, E.G.: An accurate evaluation of machine learning algorithms for flow-based P2P traffic detection. In: 22nd International International Symposium on Computer and Information Sciences (ISCIS 2007), pp. 1–6. IEEE Press, New York (2007)
7. Gonzalez-Castano, F.J., Rodriguez-Hernandez, P.S., Martinez-Alvarez, R.P., Gomez, A., Lopez-Cabido, I., Villasuso-Barreiro, J.: Support Vector Machine Detection of Peer-to-Peer Traffic. In: IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, pp. 103–108. IEEE Press, New York (2006)
8. Gao, Z., Lu, G., Gu, D.: A Novel P2P Traffic Identification Scheme Based on Support Vector Machine Fuzzy Network. In: Second International Workshop on Knowledge Discovery and Data Mining (WKDD 2009), pp. 909–912. IEEE Press, New York (2009)
9. Raahemi, B., Kouznetsov, A., Hayajneh, A., Rabinovitch, P.: Classification of Peer-to-Peer traffic using incremental neural networks (Fuzzy ARTMAP). In: Canadian Conference on Electrical and Computer Engineering (CCECE 2008), pp. 719–724. IEEE Press, New York (2008)
10. Snort, `http://www.snort.org`
11. Spognardi, A., Lucarelli, A., Di Pietro, R.: A Methodology for P2P File-sharing Traffic Detection. In: Second International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P 2005), pp. 52–61. IEEE Press, New York (2005)
12. Smoothwall open source project, `http://www.smoothwall.org`
13. Basic analysis and security engine (base), `http://base.secureideas.net`
14. Wireshark, `http://www.wireshark.org`
15. Emerging threats, `http://www.emergingthreats.net/rules/emerging-p2p.rules`