Probabilistic Identification for Hard to Classify Protocol

Elie Bursztein*
eb@lsv.ens-cachan.fr

LSV, ENS Cachan, CNRS, INRIA

Abstract. With the growing use of protocols obfuscation techniques, protocol identification for Q.O.S enforcement, traffic prohibition, and intrusion detection has became a complex task. This paper address this issue with a probabilistic identification analysis that combines multiples advanced identification techniques and returns an ordered list of probable protocols. It combines a payload analysis with a classifier based on several discriminators, including packet entropy and size. We show with its implementation, that it overcomes the limitations of traditional port-based protocol identification when dealing with hard to classify protocol such as peer to peer protocols. We also details how it deals with tunneled session and covert channel.

Keywords: payload analysis, header discriminator, p2p, traffic classification.

1 Introduction

The use of protocol identification as a defense against unwanted traffic such as P2P (peer to peer) and Malware, has received a lot of attention lately. It might appear that there is a simple identification technique to classify traffic : assuming that protocols will use well-known ports such as the one assigned by the IANA [18]. This is, however, not reliable anymore. A recent study [25] reports that in a large university about 40% of the traffic failed to be classified by this heuristic. In particular, many P2P (Peer to Peer) protocols use obfuscation techniques to avoid detection [21]. They do not use static well-known port numbers, but rather dynamically use available port numbers. They also masquerade themselves by using ports reserved for other applications and encrypt their packet payloads [29]. For example, an Edonkey node can use the port 80, typically reserved for HTTP for its own communication, allowing it to confuse firewalls, packet shaper and passive network monitors [21]. Moreover prohibited software and botnets try to deceive network security devices by tunneling their connections into other protocol such as ICMP or HTTP [7].

* PhD student, supported by a DGA grant.

In this paper, we shall show that our advanced protocol analysis allows to improve protocol identification and detect tunneled and covert session. In particular, we demonstrate through our passive network monitor prototype *NetAnalyzer* [6] evaluation, that our analysis if effective against two of the most populars P2P network: Edonkey and BitTorrent while remaining sufficiently efficient to be used online.

We choose to implement our technique in a passive network monitor because despite the great benefits provided by an advanced protocol analysis, every public passive network monitors we are aware of, including NTop [10] and Iptraf [19], still solely rely on a port-based heuristic for traffic classification. Additionally to traffic classification, our analysis reports valuable information for network assessment such as software products and protocol versions that can be used as contextual information for security evaluation.

The main contribution of this paper is a probabilistic identification analysis that combines a packet payload analysis, a packet classifier and the port heuristic. The payload analysis is based on signature. The classifier uses multiple packet discriminators: packet entropy, packet size, and packet intervals. To our knowledge, this is the first work that combines multiples identification techniques to return an ordered list of probable protocols for a given session. It seems to be also the first to add to signature and techniques a confidence value.

The rest of this paper is organized as follows. In Sect. 2, we will survey related work and in Sect. 3 we will give a straightforward example of how the analysis works. This example is used as a guideline for the rest of the paper. Sect. 4 presents our probabilistic identification analysis. Sect. 5 details how tunneled session are handheld. Sect. 6 shows how covert channel are detected. Sect. 7 details a specific application of the advanced analysis to passive monitoring: file masquerading detection. In Sect. 8 we evaluate the accuracy and the speed of the analysis against P2P traffic. Finally, we will conclude in Sect. 9.

We emphasize that even thought our discussion focus on passive network monitor, our analysis is not limited to this use. The ability to have a reliable protocol identification and session contextual information such as software version are also valuable for Firewall and NIDS (Network intrusion detection system) and traffic shaper.

2 Related Work

Ranking algorithms are a common for information retrieval [39, 15]. For instance Google use the well known Google Page rank algorithm [30]. Beside payload and packets discriminators, others techniques exists. They are either unusable for online analysis [42], or specific to a type of protocol such a P2P detection [9, 21, 16]. Protocol identification through payload analysis is used for Q.O.S enforcement in CISCO router [3] and Linux Netfilter [35]. It is also used in Bro NIDS [11] to instantiate appropriate decoders. The use of automatic signature generator has received a lot of attention [22, 37, 17, 41] because it removes the burden to create signature by hand. Tools such as Polygraph [28] and Hamsa [24] are used to create attack signature automatically. Attack against automatic signature generator have been studied in [8]. Some of *NetAnalyzer* signatures are taken from the Nmap [14], and L7 filter [35] databases. This technique uses packets headers to build a classifier [27] based on discriminators. Some discriminators work on packet payload for example packet entropy, [36], and character frequency [41]. Others, such as the packet size or time interval between packets [5] are payload independent and focus on packet headers. In [23], a combination of six discriminators is used to detect intrusion in HTTP CGI. Malwares and P2P clients, even Skype [4], use many techniques to avoid protocol identification. Packet padding [13] is used in Emule [34] to avoid traffic classifier. Popular BitTorrent clients use the "Message Stream Encryption" scheme [2] which is specifically designed to defeat protocol identification. Botnet use ICMP [7] and DNS [40] covert channel. Tunneling a protocol trough a proxy [12], is a popular technique to defeat protocol prohibition e.g IRC, Instant Messaging.

3 Identification Result Example

This example highlights the two most relevant benefits provided by our analysis namely: the accurate ranked protocol identification, and the session advanced information reporting.The example is a HTTP session reported by *NetAnalyzer* (figure 1). Each session report is composed of four parts: the summary (line 1), the traffic information (line 2), the advanced information (line 3 to 10) and the identification details (line 11 and 12);

3.1 Protocol Identification

This first major benefit of our identification analysis, is the ability to identify the correct protocol regardless of its and to provide the accuracy probability of its identification.

```
(1)http (94%): x:1052 -> y.:2080 F:0/2 R::2/0 R:0/2 E:0/0 TCP CLOSED RST
(2)[Traffic] I:1 Kb/s (3pkt) O:37 Kb/s (20pkt) [Distance] C:local S:7
(3)[Protocol]:http (1.1) HyperText Transfer Protocol - RFC 2616
(4)[File] request:www.xxx.org/vip.html Ref:"http://xxx"
(5)[File] content: extension:.html familly:text  (X)HTML
(6)[File] request:www.xxx.org/hello.gif Ref:"http://xxx"
(7)[File] content: extension:.jpg familly:image
(8)[Server] Apache httpd 2.0.52
(9)[Client] browser Internet Explorer 6.0 Windows XP
(10)[Client] proxy squid 2.5.STABLE4-20031106
(11)[Guessed protocol] http:94% Port:0% Class:100% Patt:100%
(12)[Guessed protocol] autodesk:9% Port:100% Class:n/a Patt:0%
```

Fig. 1. *NetAnalyzer* Session output example

The protocol identified with the highest probability is displayed on the left-most part of the first line. Here it is the *http* protocol with a probability of 94%. It is not 100% because the server port 2080 is not the standard one: 80. Hence there is a conflict between the port heuristic result and the payload analysis result. The list of all possible protocols for the session, is presented at the end of the report (line 11 and 12). Each line gives the protocol name, its probability and details the result of each technique used. Intuitively the probability of 94% for the HTTP protocol results of the six signatures positive match along with the port heuristic's negative result and the classifier 100% positive result. The classifier score is 100% positive because every 23 packets of the session match HTTP protocol profiles. Because each protocol classifier probability result is independent, the sum of all protocol probabilities might exceed 100% as here. Autodesk have a probability of 9% because it only has the port heuristic positive result. Autodesk classifier result report report *n/a* (not available) because the analyzer does not have profile for this protocol.

3.2 Advanced Data

Advanced data are displayed from line 3 to 10. These data are gathered when signatures are successfully matched. The signature language allows to extract data from the payload such as the filename requested in the HTTP request (line 4 and 6). This is done as in Perl regular expression by adding capture parenthesis to the signature and adding the corresponding capture variable to one of the signature template. In the case of the HTTP filename, the variable was added to the filename template.

To provide a set of suitable templates for each type of information gathered *NetAnalyzer* uses four types of signatures: protocol, file, software, and user. For example the software version data is irrelevant when dealing with file analysis.

4 Probabilistic Identification Analysis

As exemplified in the above section, the analysis combines the result of multiple signatures matches along with identification heuristics. The protocol identification is said continuous because for each new packet, the session protocol is re-evaluated. As demonstrated in [17], protocol identification based on signature matching is more accurate than the identification based on classifier. Both of them are of course more reliable than the identification based on port heuristic. Thus the analysis uses the weighted arithmetic mean to reflect these different levels of accuracy. Accordingly the probability of a protocol $\mathbb{P}_x$ is computed as follows:

$$\mathbb{P}_x = \frac{\alpha\mathbb{H}_x + \beta\mathbb{C}_x + \gamma\mathbb{S}_x}{\alpha + \beta + \gamma}$$

where $\mathbb{H}_x$ is the protocol probability according to the port heuristic and α is its confidence coefficient. $\mathbb{H}_x$ is either equal to 0 or 100. $\mathbb{C}_x$ is the protocol probability according to the classifier heuristic and β is its confidence coefficient. $\mathbb{C}_x$ range from 0 to 100. Finally $\mathbb{S}$ is the protocol probability according to the payload analysis and σ is its reliability coefficient. In *NetAnalyzer*, we use the following values $\alpha = 1, \beta = 5, \sigma = 10$. They are consistent with [17] and work well in practice. The HTTP protocol probability of the example 1 is therefore:

$$\mathbb{P}_{http} = \frac{1 \times 0 + 5 \times 100 + 10 \times 100}{16} = 93.75$$

4.1 Classifier Probability

The classifier probability for a given protocol is computed by comparing each packet with a set of profiles. In its current implementation, *NetAnalyzer* only reports TCP protocol solely identified by profile if and only if there is no protocol identified by the payload analysis. ICMP and UDP identified protocols are always reported. The classifier probability is the ratio between successful matched packets and the total of packets:

$$\mathbb{C} = \frac{success}{total} * 100$$

In the above example (Figure 1), the 23 packets were successfully matched against HTTP profiles.

As in [41], server and client stream profiles are separated to improve classifier accuracy. This makes sense because often one stream is used to request data that are sent back by the other e.g the HTTP client request a file that is sent by the server. The classifier uses four discriminators: the packet size, the time elapsed since the last packet from the same stream, the time elapsed since the last packet from the other stream, and finally the entropy of the packet. The entropy use the Paninsky estimator [31] which is known to be efficient even against a small amount of data. As noted in [5], the set of profiles matched by a packet is related to it position in the stream: the stream first packets often contains request and authentication data. For example a POP3 session starts with a hello exchange followed by an authentication. That is why we keep a separate set of profiles for the first 10 packets of each stream and an aggregate one for the remaining packets. Thus each protocol has 22 profiles sets.

```
Ping:ICMP:2:2:64:64:995229:1004962:::7.54564:7.65728:
```

Fig. 2. ICMP ping packet 1 profile

For each discriminator, an upper bound and a lower bound are determined from the average mean and the standard deviation. If the packet value is between these bounds then it matches the discriminator. A packet matches a profile if it matches the four discriminators. We use a set of profiles rather than a single profile with very large bound to improve detection accuracy. We generate automatically profiles with smaller bound by using the standard k-means clustering [32] algorithm. We prevent profile over-fitness by limiting the number of profiles to 5 by set. During the clustering process, meaningless discriminators are removed. A discriminator is meaningless, if its standard deviation is too wide despite the clustering process. For example the entropy discriminator is removed if it has a standard deviation above 3. Figure 4.1 is an example of profile used in *NetAnalyzer*. The first field is the protocol name: Ping. The second field is the layer 3 protocol: ICMP. The third is the targeted stream: the number 1 is used for the server stream and 2 for client stream. The fourth field is the stream packet number. The exemple is a profile for the stream second packet. The rest of the profile are the four discriminators lower and upper bound: First the packet size (64 bytes), Secondly the time elapsed since the last packet from the same stream in microsecond (around 1 second here). Thirdly the time elapsed since the last packet from the opposite stream. Here it has been removed because it was meaningless.

Finally the packet entropy, which has to be between 7,54 and 7,65. Profiles are very efficient for covert channel detection as we will shown in Sect. 5

4.2 Payload Probability

Intuitively the payload analysis can be misled by tunneled session because the first signature matches belong to the tunnel protocol not the tunneled one. Hence the first thing to detect tunneled session is to perform continuous identification to identify the tunneled protocol. This is however not sufficient because the analyzer has also to figure out which one is the real protocol. This is done by taking into account the matches time line . Relying on the last match is not an option because it opens the door to injection attack that add an irrelevant payload at the end of the session. That is why the payload analysis gives to the latest matched signature a more significant weight than any previous match. To do so the technique uses a *weighted moving average* to compute each protocol probability:

$$\mathbb{P} = \frac{D_{x_i} \times n + D_{x-1} \times (n-1) + ... + D_{x_1} \times (1)}{n + (n-1) + ... + 1}$$

Where D_{x_i} is the confidence value associated to the signature matched in position i and n the global number of matches. The confidence value associated to a signature is by default 100. It can be however tweaked by the signature language option *confidence*. This is useful to decrease the confidence of a signature that is known to produce false positive. *NetAnalyzer* uses a specialized signature language to perform the payload analysis that allows to specify the signature confidence. For instance some of the signatures used to detect edonkey p2p traffic are known to produce from time to time false positive because they are quite short. However due to the edonkey protocol format they cant be improved, hence the only way to mitigate false positive is to reduce their confidence value.

5 Tunneled Session Detection

Tunneling a protocol into an another is a popular technique to bypass firewall restriction, hence a tunneled session is often a violation of the security policy. As explained above (Sect. 4.2), the WMA *Weighted Moving Average* used to compute the payload analysis probability assigns a heavier weight to the most recent match. This is consistent with the fact that matches from the tunneled protocol will be reported after tunnel protocol ones.

```
(1) CONNECT irc.********.org:6667 HTTP/1.0
(2) HTTP/1.0 200 Connection established
(3) USER 0 0 0 ::
(4) NICK ****
(5) :irc.********.org 001 **** :Welcome to the **  **!0@xxx
```

Fig. 3. An exemple of a tunneled IRC session in a HTTP one

Figure 3 example is an IRC session tunneled into an HTTP one. The first two matches will identify the sessions as a HTTP one (line 1 and 2). If the identification process stops after the first match then the session is incorrectly identified as HTTP. With the continuous inspection, two matches for the IRC protocol will be reported (line 3 and 5). Hence the analysis ends up with four matches: two for HTTP and two for IRC. Because the analysis use a WMA, the two IRC matches will have an heavier weight, and therefore the session will be correctly identified as an IRC one. Payload probability scores are:

$$\mathbb{S}_{http} = \frac{100 + 200}{100 + 200 + 300 + 400} = 30\% \qquad \mathbb{S}_{irc} = \frac{300 + 400}{1000} = 70\%$$

Thereafter identification scores are:

$$\mathbb{HTTP} = \frac{100 \times 1 + 30 \times 10}{11} = 36,3\% \qquad \mathbb{IRC} = \frac{0 \times 1 + 70 \times 10}{11} = 66,6\%$$

Let's take a step further and imagine that the session is not a tunneled session but instead the download of the IRC RFC. In this context, the protocol identification has been misled. However, the previous computation does not take into account the probability given by the classifier. HTTP session and an IRC session exhibit very different profiles.

The figure 4 presents the packet size evolution for an IRC server stream whereas the figure 5 shows the packet size evolution for a HTTP server stream. In the HTTP stream, the packet size is constant after the first packet because when a file is sent to a client, the standard behavior is to maximize the throughput by sending the largest packet possible. This behavior is often referred as a TCP bulk transfers. Conversely because IRC is an interactive protocol the message size transmitted by the server to the client greatly fluctuate and never reach the maximum segment size. Hence every packet will be recognized by the classifier as HTTP one. Thereafter the complete protocol identification probability is:

$$\mathbb{HTTP} = \frac{100 \times 1 + 100 \times 5 + 30 \times 10}{16} = 56,25\% \qquad \mathbb{IRC} = \frac{0 \times 1 + 0 \times 5 + 70 \times 10}{11} = 43,75\%$$

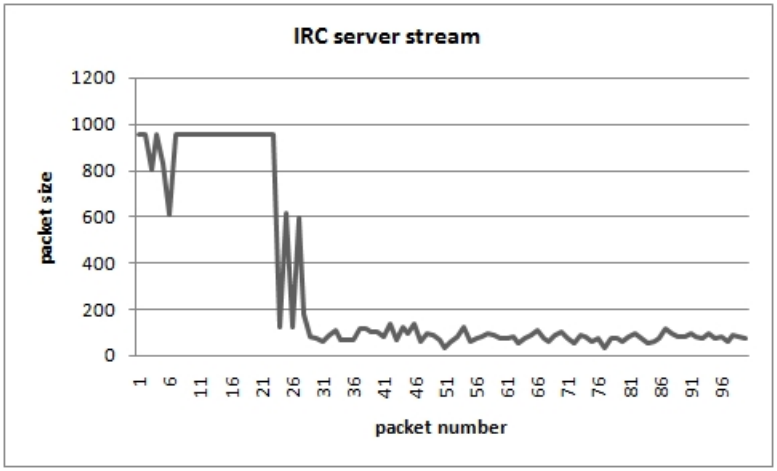

Fig. 4. An IRC server stream packet size evolution

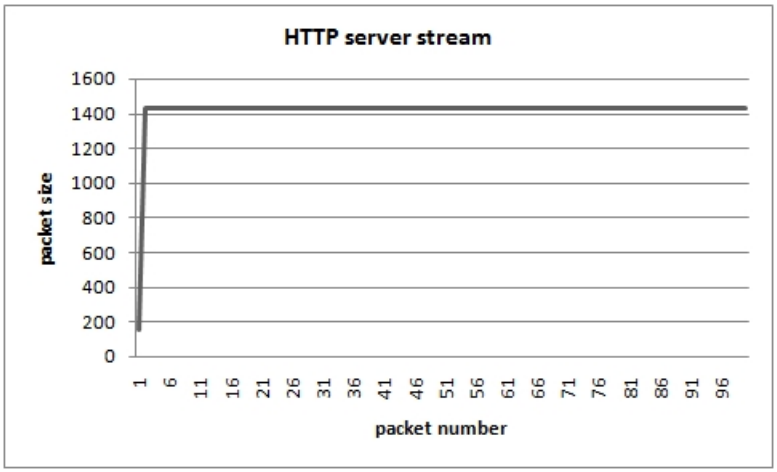

Fig. 5. An HTTP server stream packet size evolution

6 Covert Channel Detection

An other important problem is the detection of covert channel. Botnets and malware use them to hide there presence and bypass firewall restriction. The DDOS tool Stacheldraht [7] use an ICMP covert channel, and the backdoor Spotcom [40] use, to some extent a DNS one. ICMP covert channel are hard to detect because the RFC [33] state that the packet payload can be anything. DNS channel are also difficult because tunneled traffic hide in TXT record or other arbitrary length field. That is why covert channel are mainly uncovered by the traffic classifier. We present here how a ICMP tunnel is detected by the traffic classifier. For testing purpose, we have used the popular software PTunnel [38] to tunnel a SSH session into an ICMP covert channel. *NetAnalyzer* was able to detect it because the packets generated by the covert channel does not match ICMP ping profiles. Two discriminators exhibit very different behavior when it is a legitimate ping and when it is a covert channel. The first discriminator is the packet size: for a legitimate ping, the packet size is constant (diagram 6) whereas it fluctuates greatly for the covert channel (diagram 7)

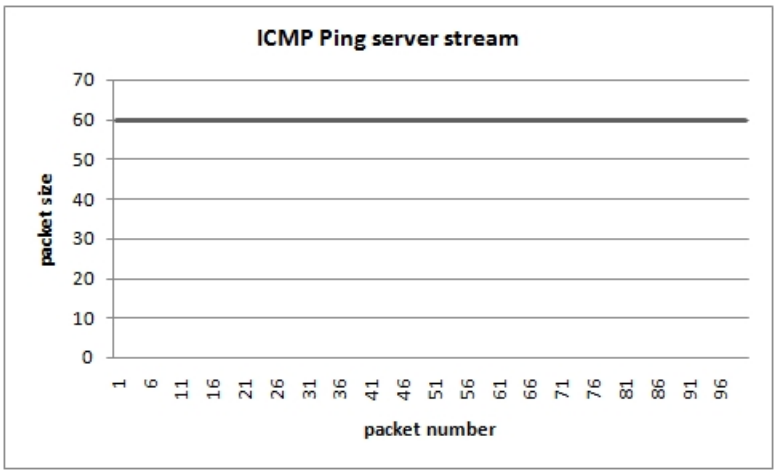

Fig. 6. An Icmp echo reply stream packet size evolution

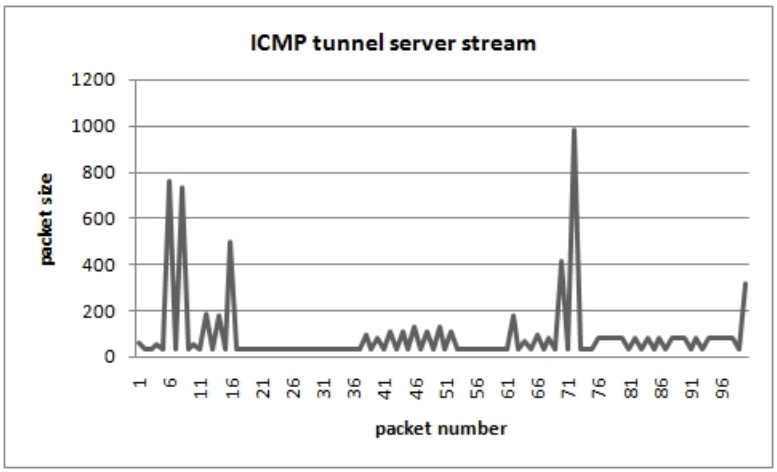

Fig. 7. A Ptunnel server stream packet size evolution

The second discriminator is the interval between packets of the same stream. Values are very stable (See diagram 8) for a legitimate ping because requests are send on regular basis, whereas the interval is totally unpredictable for the covert channel (See diagram 9).

Due to the lack of space, we do not detail how a DNS channel is detected, but as noted in [20], the entropy discriminator is effective against it. Intuitively this is because the legitimate data uses a limited character set [26], that induce a low entropy whereas tunneled session have a high entropy. Traffic classifier is effective against covert channel, because it is very difficult to impersonate successfully every aspect of a given protocol. However some obfuscation techniques such as packet padding [13] or a combination of them [2] can be use to deceive it. In this case the only option is to find an other discriminator that will be not confused until a new obfuscation method is introduced.

7 File Masquerading Detection

The payload analysis allows *NetAnalyzer* to use content information to detect *File masquerading*. A file masquerading occurs when the extension of file does not reflect the content of the file *e.g* an *avi* file with a *.html* extension. This masquerading can mainly occurs for two reasons that deserve attention. The most common one, is when the masquerading is used to hide litigious or illegal file.

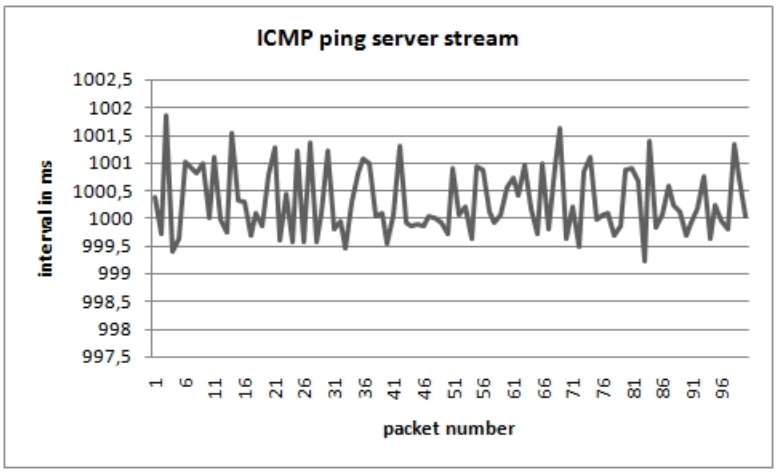

Fig. 8. An Echo reply stream packet interval evolution

This a common practice used by many pornographic websites to deceive free hosting policy: they rename their photos or videos with textual extension such as .txt, to evade server log analysis that only rely on file extension. This technique is also commonly used by hackers to hide their illegal files on a compromised server. The second reason that leads to file masquerading is when a legitimate user makes a mistake. In this case detecting file masquerading is also important because it can prevents legitimate application to open properly the masqueraded file. A real world example of such masquerading is visible in figure 1: The first requested file is a gif masquerades as a jpeg file. This is not a serious issue as the browser will handle it, but nevertheless gif format is still proprietary.

8 Evaluation

To evaluate the effectiveness of our identification analysis presented in Sect 4, we run *NetAnalyzer* against a 8Gb trace of a residential network traffic. The goal was to evaluate the analysis ability to identify P2P sessions. We run two analysis : the first with only the port heuristic activated and the second with advanced analysis.

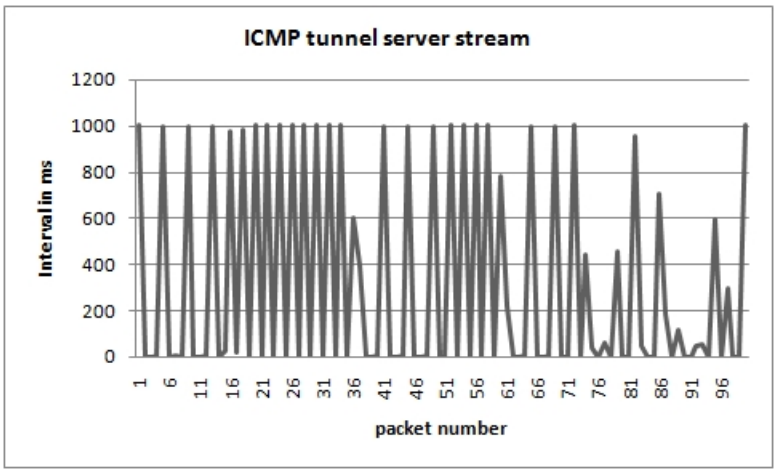

Fig. 9. A PTunnel server stream interval size evolution

Analysis results are summarized in the following table:

Protocol	Port heuristic	Advanced inspection	Difference
HTTP	8589	9512	10,7%
BitTorrent	0	1504	n/a
Edonkey	1694	11249	564%
Unknown	16604	7564	-54.4 %
Others	3748	806	- 78,5%

When the identification use only the port heuristic, 54% of the traffic is reported as unknown which is coherent with the study [25]. Only 6% of the traffic is detected as Edonkey and none as BitTorrent (See figure 10). The traffic reported as ”Other” include email and instant messaging traffic but also improbable protocols. This not surprising that port heuristic performs so poorly against Edonkey and BitTorrent traffic. Both of them use many obfuscation techniques. The popular client for the Edonkey network Emule [34] enable by default obfuscation techniques, such as padding, since version 0.47b . Popular BitTorrent client such Azureus [1] recommends to not use standard port and can use a very effective obfuscation scheme called ”Message Encryption Stream” [2] based en public key cryptography and payload randomization. This scheme, also used by μTorrent, is designed ”*to provide a completely random-looking header and (optionally) payload to avoid passive protocol identification and traffic shaping.*” In particular it can use RC4 and DH to encrypt the packet payload.

When the advanced analysis is enabled, the protocol classification accuracy improves drastically (See figure 11). This time the traffic classification shows that more than 60% of the traffic is in fact P2P traffic. It also reduce the number of unknown traffic by 54.4%, and to increase the number of Edonkey session identified by 564%. The advanced analysis is able to uncover BitTorrent traffic, something that the heuristic based analysis was unable to achieve. The number of others protocol also shrinks down by 78,5%: the advanced analysis was able to reduce the number of sessions incorrectly identified. Even if the protocol classification is still not perfect, as 25% of the traffic remains unclassified, the advanced analysis as been proved effective to improve significantly the protocol classification when hard to classify protocols are used.

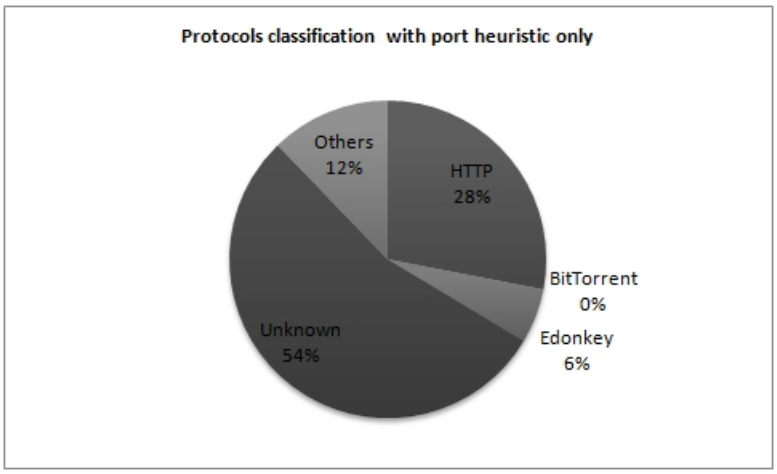

Fig. 10. Protocols classification based only on the port heuristic

9 Conclusion

The main contribution of this paper was a probabilistic identification analysis and its implementation in a passive network monitor called *NetAnalyzer*. We have shown that it overcomes the limitations of traditional port-based protocol identification when dealing with hard to classify protocol. We also have shown that the analysis is able to deal with tunneled and covert channel. A Future work is to introduce specific discriminator for P2P to improve further more the

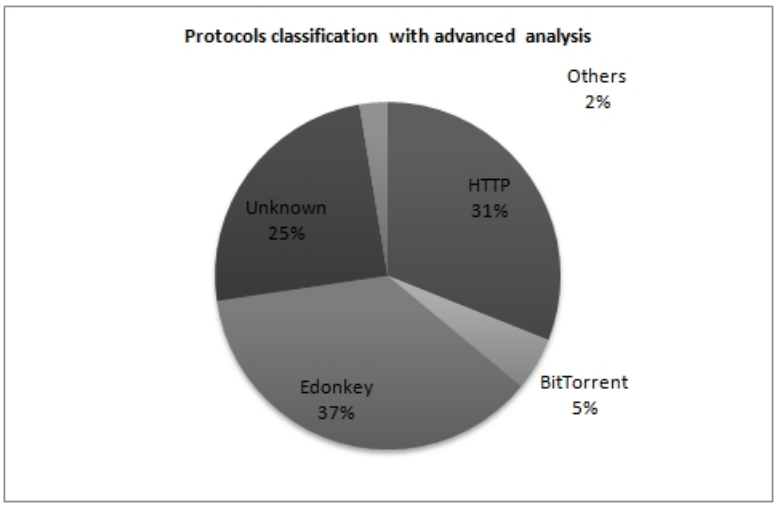

Fig. 11. Protocols classification that uses the advanced analysis

identification. The UDP and TCP connection pairing discriminator proposed in [21] seems promising.

References

1. Azureus. Increase download speed recommandation: http://www.azureuswiki.com/index.php/increase_download_speed.
2. Azureus. Message stream encryption: http://www.azureuswiki.com/index.php/message_stream_encryption.
3. Babachiz. Managing peer-to-peer traffic with cisco service control technology. Technical report, CISCO, Febuary 2005.
4. S. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *IEEE infocom 2006*, 2006.
5. L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic classification on the fly. *SIGCOMM Comput. Commun. Rev.*, 36(2):23–26, 2006.
6. E. Bursztein. Netanalyzer homepage: http://www.netqi.org, 2006.
7. G. Cheng. Analysis on ddos tool stacheldraht v1.666 : http://www.sans.org/resources/malwarefaq/stacheldraht.php.
8. S. P. Chung and A. K. Mok. Allergy attack against automatic signature generatio. In *RAID Recent Advances in Intrusion Detection*, 2006.
9. F. Constantinou and P. Mavrommatis. Identifying known and unknown peer-to-peer traffic. In *NCA '06: Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications*, pages 93–102, Washington, DC, USA, 2006. IEEE Computer Society.
10. L. Deri and S. Suin. Ntop: Beyond ping and traceroute. In *DSOM*, pages 271–284, 1999.
11. H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. Sommer. Dynamic application-layer protocol analysis for network intrusion detection. In *Usenix*, 2006.
12. D. Dvoinikov. Htthost (http proxy):http://www.htthost.com/htthost.boa.

13. X. Fu, B. Graham, R. Bettati, and W. Zhao. On effectiveness of link padding for statistical traffic analysis attacks. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 340, Washington, DC, USA, 2003. IEEE Computer Society.
14. Fyodor. Nmap : free open source utility for network exploration or security auditing http://www.insecure.org/nmap/.
15. F. R. G. M. Del Corso. Ranking a stream of news. In *14th international conference on World Wide Web*, page 97, 2005.
16. P. K. Gummadi, S. Saroiu, and S. D. Gribble. A measurement study of napster and gnutella as examples of peer-to-peer file sharing systems. *SIGCOMM Comput. Commun. Rev.*, 32(1):82–82, 2002.
17. P. Haffner, S. Sen, O. Spatscheck, and D. Wang. Acas: automated construction of application signatures. In *MineNet '05: Proceeding of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 197–202, New York, NY, USA, 2005. ACM Press.
18. IANA. Matrix for protocol parameter assignment/registration procedures http://www.iana.org/numbers.html.
19. J. Java. Iptraf (ip network monitoring software): http://iptraf.seul.org/.
20. D. Kaminsky. Attacking distributed systems the dns case study. In *Black Hat Europe*, 2005.
21. T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy. Transport layer identification of p2p traffic. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 121–134, New York, NY, USA, 2004. ACM Press.
22. C. Kreibich and J. Crowncroft. Creating intrusion detection signatures using honeypot. In *Second Workshop on Hot Topics in Networks*, Nov. 2003.
23. C. Kruegel and G. Vigna. Anomaly Detection of Web-based Attacks. In *Proceedings of the* 10^{th} *ACM Conference on Computer and Communication Security (CCS '03)*, pages 251–261, Washington, DC, October 2003. ACM Press.
24. Z. Li, M. Sanghi, B. Chavez, Y. Chen, and M.-Y. Kao. Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilience. In *IEEE Symposium on Security and Privacy*, May 2006.
25. A. Madhukar and C. Williamson. A longitudinal study of p2p traffic classification. In *MASCOTS '06: Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation*, pages 179–188, Washington, DC, USA, 2006. IEEE Computer Society.
26. P. Mockapetris. Rfc1035: Domain names - implementation and specification. Technical report, Network Working Group, 1987.
27. A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 50–60, New York, NY, USA, 2005. ACM Press.
28. J. Newsome, B. Karp, and D. Song. Polygraph: Automatically generating signatures for polymorphic worms. In *IEEE Symposium on Security and Privacy*, May 2005.
29. S. Ohzahata, Y. Hagiwara, M. Terada, and K. Kawashima1. A traffic identification method and evaluations for a pure p2p application. In Springer, editor, *Passive and Active Network Measurement*, pages 55–68, 2005.
30. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
31. L. Paninski. Estimation of entropy and mutual information. *Neural Comput.*, 15(6):1191–1253, 2003.
32. D. Pelleg and A. W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.

33. J. Postel. Rfc792: Internet control message protocol. Technical report, Network Working Group, 1981.
34. E. Project. http://www.emule-project.net/.
35. Quadong. Linux layer 7 packet classifier http://sourceforge.net/projects/l7-filter/.
36. C. E. Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, 1951.
37. S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *Operating Systems Design and Implementation*, Dec. 2004.
38. D. Stødle. Ping tunnel.
39. P. Tsaparas. *Link Analysis Ranking*. PhD thesis, University of Toronto, 2004.
40. J. Turkulainen. Backdoor spotcom analysis http://www.securiteam.com/securityreviews/6z00n208uc.html.
41. K. Wang, G. Cretu, and S. J. Stolfo. Anomalous payload-based worm detection and signature generation. In *Recent Advance in Intrusion Detection*, 2005.
42. C. Wright, F. Monrose, and G. M. Masson. Hmm profiles for network traffic classification. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 9–15, New York, NY, USA, 2004. ACM Press.