Using of Flow Statistics for Improvement of Protocol Detection

Pavel Piskač and Jiří Novotný

piskac@mail.muni.cz, novotny@ics.muni.cz

Institute of Computer Science
Masaryk University
Brno, Czech Republic

Abstract

Protocol detection can be made by using various techniques. First attempts were made through the use of protocol numbers but this method is insufficient now. Later approach analyses packet content, which needs high computation power and cannot be used for tunneled data or in high-speed networks. One of the newest methods is a protocol recognition, which uses flow statistics gathered from NetFlow data, and is the main focus of this work.

First attempts were made by using time characteristic of a flow. This kind of information consists of minimum, maximum, average and variation of the inter-packet gaps and it allows us to detect dictionary attacks on SSH protocol. All the data we had were manually inspected because this type method had been decided to be sufficient at this phase of development. In the first step, we discovered that time characteristic can be used but its analysis needs improvements so that it can detect more than one specific situation.

We experimentally discovered that the information about inter-packet gaps give us low detection accuracy, so extended flow statistics by packet sizes. This extension consists of minimum, maximum, average and variation of packet sizes in a flow. Our assumption is that the more information we have, the more details we can expose about protocols on the network. The amount of information which we have analyzed increased rapidly thus we had to implement QT (quality threshold) clustering algorithm. It can inspect all the data much more quickly and precisely than human examination.

In the future work, we are going to continue with discovering which combination of flow statics brings the best accuracy of protocol detection. These results will be summarized and implemented as a plugin for NfSen which is web based front end for the nfdump NetFlow tools which is used at Masaryk University.

Keywords: NetFlow, IPFIX, protocol detection, SSH, clustering, inter-packet gaps.

1 Introduction

Diversity of protocol detection methods brings different approaches with different results. Protocol detection can be used in various fields of interest. Computer network topology planning will achieve better results if the analysts know what applications will use it. Security teams need to use protocol detection for detecting botnets, malware, intrusions or other suspicious activities. Real-time data processing is crucial in many fields of interest.

Common approaches in protocol detection are based on port numbers and/or signatures. The signatures can be gathered either from payload or flows. This work is introducing an idea how to use information about inter-packet gaps and packet sizes for protocol detection which is able to process data on high-speed

networks. Similar experiments were done for example in [13] but we are more focusing on inter-packet gaps and packet sizes, we take into account only flow statistics without looking into packet content as well.

Our approach allows us to generate the statistics in hardware probes which are much more powerful than ordinary software tools. Another advantage of using hardware probes is in their preciseness. Common network card processes packets in chunks because they are buffered first and then sent to the operating system. This kind of packet handling increases transmission rate but it reduces the accuracy only to approximately 1 millisecond. On the other hand, hardware probes are able to process packets one-by-one and assign precise time-stamp to the packet; the accuracy in this case can be up to 1 nanosecond. Since the precise time-stamps are crucial for our work, implementation in hardware probes is necessary for more precise results.

In the previous part of research, we showed that inter-packet gaps analysis is suitable for dictionary attacks on SSH protocol detection. Now, we are working on making this method more general to be able to detect as many protocols as possible. Because of the current state of the research, we are not presenting the final work but mainly the idea and results which we reached so far.

2 Related Work

The first attempts in protocol detection were made using source and destination TCP and UDP port numbers assigned by Internet Assigned Numbers Authority (IANA) [1]. This method has insufficient accuracy now since servers can use different than the default port numbers, peer-to-peer (P2P) networks avoid using known port numbers in order to avoid their detection, and there are also protocols, such as FTP, which use randomly chosen ports.

The newer approach is called deep packet inspection (DPI), it analyses packets content [2] and looks for some predefined string patterns [3], [4], [5]. If the pattern is done properly, this method has the best accuracy. The main bottleneck of DPI presents high computational power needed for pattern searching inside the packets content, it inhibits from using on high-speed networks. If encrypted or tunneled communication is used, all the patterns are hidden and DPI cannot reveal any protocol. Investigating packet content can be interpreted as an attack upon users privacy, thus this method is prohibited in some countries due to their local law constraints.

Since DPI has to process enormous amount of data, privacy related problems and impossibility to classify encrypted communication, new methods have been researched. They use only aggregated subset of available information. It allows them to do protocol detection on high-speed networks without restrictions due to local law constraints.

The first payload independent method is monitoring the social behavior of hosts. It analyses for example number of hosts which communicated with the monitored host or communities of hosts that interact with the same set of hosts [5]. Other studies present profiling using Google search engine [6] or statistics about used peer numbers, port numbers and packet sizes [7]. These kinds of methods are reliable but divide communication only into groups (P2P, WWW, email, etc.). Furthermore, these methods require sufficient amount of data for their algorithms to work properly.

Another approach for payload independent traffic classification uses flow statistics. The statistics can consist of any subset of the following information: packet sizes, packet inter-arrival times, packet arrival order, flow duration, packet counts, etc. The first studies were able to detect traditional protocols such as HTTP, SMTP or FTP [8], [9]. Another approach uses similarity in flow properties and divides traffic into groups (bulk, WWW, P2P, etc.) [10], [11].

All the previous methods examine information from the whole flow. It was shown that only the first four packets from a connection are needed for successful classification of some traditional protocols both unencrypted and encrypted [12].

Combination of payload and flow-level statistics is possible as well. In the study [13] was presented an idea which uses over 30 unique fingerprints [14]. The fingerprints are used to measure various generic behaviors of application layer protocols.

3 System Architecture

At this part, we will introduce architecture of our system. Our work needs specialized network monitoring tools which are described in the first subsection. The other subsection is talking about information which we get from flows and how they are used for protocol detection.

3.1 Hardware Point of View

Data processing consists of two different layers. At first, we have to get time characteristics and statistics about packet sizes in a flow. It can be done in two ways, either using software tool or specialized hardware.

Software tool uses standard network interface card and time information from the operating system. This connection is suitable mainly for testing and development purposes because between packet arrival and its processing is a relatively long and non-constant time period. At first, packet enters the network interface card, is saved into first-in-first-out (FIFO) queue and card generates an interrupt. After the interrupt handler is run, data is moved into a system buffer in main memory, where the packet is placed on a receive queue and marked for system processing where time stamp is set by operating system.

Two buffers and unpredictable duration of the interrupt cause time characteristics distortion [15]. The conditions may vary but information more accurate than 1 millisecond cannot be taken into account. Such behavior does not represent sufficient environment for protocol detection but it is suitable for testing purposes. Computational power of such software tools is very limited by hardware and its throughput will always be lower than in case of hardware network probes.

Specialized hardware network probes are developed for precise packet handling and thus the time characteristics are not negatively influenced. Time stamps are set directly in the card. In this case we can talk about 1 nanosecond preciseness. Another advantage is much more throughput which it is able to achieve. Current probes are created for up to 10 Gbps bandwidth. The disadvantage is that there are no hardware probes customized for time characteristics creation thus the users need to use devices extensible by plugins.

Even if hardware probes offer better results, we use only software tools in our work. In respect to current state of development, it is easier to change software than hardware. The purchase cost of software is also much lower than hardware. As a time characteristics generating tool we use Flow Time Statistic (FTS) which is testing tool for the Liberouter project [16]. This tool is relatively easy to change for our purposes because it is written in C using common libraries. FTS is able to export data either into text files or in IPFIX format which is better for storing large amount of data and its processing. Text files provide easy to handle structure which is suitable for testing and development.

Protocol detection on a real network will consist of hardware probes which send data in IPFIX format to a data collector where is the data stored, processed and displayed to a user. There are various solutions for probes that support plugins ([22], [21], etc.) and others will come in the future. If there are customers for such probes, the developers will extend existing one by plugins support or time characteristics. There are also some tools for data visualization ([20], [25], etc.) or the users can create their own application

according to their current needs. Creating a new application is much easier than inventing a new hardware.

Connection of network monitoring tool FTS and the following data processing is described on Figure 1. There is no specialized data store in order to make it as simple as possible. At this phase of development, we want to concern on the protocol detection instead of data storing and visualization.

On the other hand, the final output of our work cannot store data in text files because amount of data is much bigger than while testing (Figure 2). A database presents an effective way how to store and browse huge amount of data.

Visualization in our case is always done through the NfSen which is a graphical web based front end for the NFDUMP NetFlow tools. NfSen can be extended by plugins so that we do not have to invent any new technology but we use existing infrastructure instead. This approach saves much time and it is also much more advantageous because the less lines of code are written the fewer mistakes can be made.

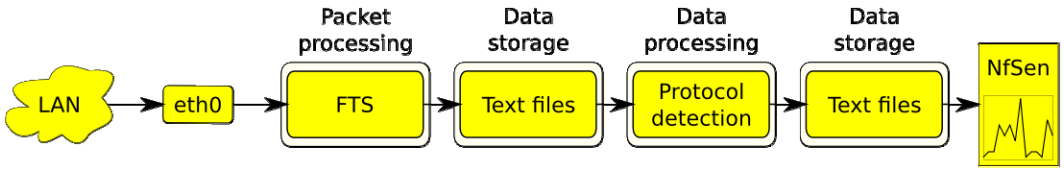

Figure 1: Packet processing using FTS.

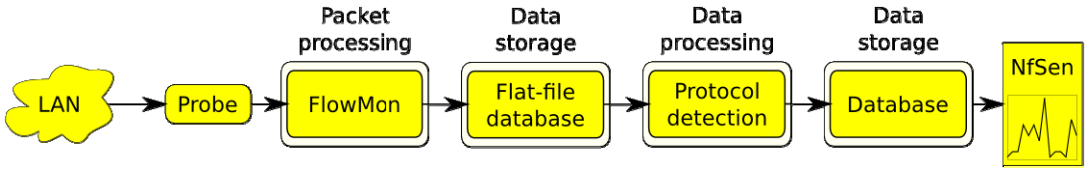

Figure 2: Packet processing using a hardware network probe.

3.2 Protocol Detection

As it was mentioned in the beginning, protocol detection is based on time characteristics and other flow statistics analysis. Time characteristics are statistical information about inter-packet gaps and they consist of:

- accurate time stamp of a flow begin,
- accurate time stamp of a flow end,
- minimal inter-packet gap in a flow,
- maximal inter-packet gap in a flow,
- average inter-packet gap in a flow,
- standard deviation of inter-packet gaps in a flow.

Protocol detection based on these characteristics was the goal for Pavel Piskač's diploma thesis and it was shown that we are able to detect protocols with lower accuracy than we expected. So that we extended information about a flow by statistics about packet sizes:

- minimal packet size in a flow,
- maximal packet size in a flow,

- average packet size in a flow,
- standard deviation of packet sizes in a flow.

This data is creating a vector and the detection is done using vector comparison methods. Since the detection is under research now, content of the vector can change in order to get better results. We are making experiments with various combinations of vector components and also taking advices from available related work.

At the first step, we used a pattern vector which was compared to unknown connections vectors and according to their similarity it was decided whether the unknown vector can represent the protocol from the pattern vector.

Such approach is very limited because there has to be created a pattern for each situation which can appear on the network. For example simple SSH connection is able to create many different situations where each of them has to have its own pattern. It means that for simple password authentication, dictionary attack on password, file transfer, etc. is needed the pattern to be created. Such approach is ineffective and also very difficult. The final detection method will be slow and able to recognize only predefined set of protocols.

Our goal is to create as easiest to deploy detection method as possible thus we moved into the other part where clustering algorithm is used. Clustering has the advantage in automatized data processing where the flows are divided into groups and then the groups are compared to patterns. The improvement is obvious, there is no need to compare each flow to a pattern but if the clustering is used there are compared the whole groups.

3.3 Process of Protocol Detection

Gathered flow statistics are sent from probes (Figure 2) or FTS (Figure 1) to a collector where they are stored. This action does not take place continually but it depends on the source configuration. In a regular scenario, there is set activity and inactivity time-out. Activity time-out is used for decision whether a flow was finished according to the time of last received packet. If the limit is reached, the flow is marked as finished. Inactivity time-out serves for ending very long connections.

Some connections can be active for a very long time and only the activity time-out guarantees that we will know about them in a reasonable short time. The connections are marked as finished after some constant time. The other effect of such behavior is automatized freeing buffers.

The most common configuration for inactivity time-out is 5 minutes. Then the data is sent to a collector and prepared for the protocol detection algorithm. Since the data is sent each 5 minutes the algorithm cannot run longer and it is very important requirement on its effectiveness.

Results of the detection are saved into database or files in order to be available on request. This approach is more effective from the user point of view because the data is always available and the users do not have to wait until it is computed.

The disadvantage is that the information about protocols is not available immediately. The user has to wait 10 minutes where the first 5 minutes takes gathering statistics about flows and the other 5 minutes is for protocol detection. In case of deep packet inspection, the protocol information is available almost immediately after the communication began.

4 Practical Results

This part is delegated to results which we got so far and it is divided into two parts. At first, results of dictionary attacks on SSH will be presented. Current state of work will then be described.

4.1 Dictionary Attacks on SSH Protocol

Since there was hardly any work relating to time characteristics analysis, we had to show that protocol detection can be made using this type of information. The first steps were done during autumn 2009 as the Pavel Piskač's diploma thesis.

4.1.1 Protocol Selection

We decided to use SSH protocol because it presents possible vulnerability on a computer network if a computer is compromised through this protocol. The other advantage of SSH is that it is a standard described in many RFCs ([23], [24], etc.) so that we know its behavior on the network.

We could also make experiments with for example BitTorrent or Skype protocols but they use obfuscation and do not have known specification. We decided to start with some important but easy to detect protocol. Skype either BitTorrent does not belong into this group.

The assumption that SSH is an easy to describe and easy to detect protocol has been decided to be false. It is true that SSH is well known but there are many implementations and versions thus many various situations can happen and make the detection more complicated.

The differences in communication between hosts are created by operating system as well. TCP connection handling in the Linux kernel can be modified in order to achieve better transfer efficiency and bandwidth. These tweaks influence maximal number of received packets before the ACK packet is sent and maximal interval before the ACK packet is sent. As the result, there are changing positions and counts of ACK packets which influences time characteristics.

The third problematic SSH characteristic behavior is the count of situations which may occur. For example for authentication can be used password, public key, etc. Each of the authentication methods can be configured in many different ways. Since each situation has to have its own pattern vector, SSH protocol detection would require more than 10 different pattern vectors.

4.1.2 Protocol Detection

Our first attempts were aimed to detect user authentication even if many pattern vectors had to be created and tested. During the testing, we discovered a complication with password based authentication because when a user enters the password the inter-packet gaps are always different. It is caused by user's speed of typing and as a result we were not able to use pattern vectors for password based authentication.

Our servers are almost always under dictionary attacks so that we decided to detect them. Dictionary attacks are generated by foreign computers without any user interaction. It implies that we are able to detect them using our method. There are simpler and more accurate methods than our approach but we wanted to find out if the time characteristics are suitable for protocol detection.

That time has been decided to use vector matching algorithms. The vector was consisting of the following information about a flow:

- number of transferred packets,
- number of transferred bytes,
- network layer protocol number,

- transport layer protocol number,
- minimal inter-packet gap,
- maximal inter-packet gap,
- average inter-packet gap,
- standard deviation of inter-packet gaps.

These vectors were compared using four different methods:

- average distance between vectors (Equation 1),
- root-mean-square distance (Equation 2),
- Euclidean distance (Equation 3),
- angle between vectors (Equation 4).

$$d(p,q) = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N}$$

Equation 1: Average distance between vectors.

$$d(p,q) = \sqrt{\sum_{i=1}^{N} \frac{(p_i - q_i)^2}{N}}$$

Equation 2: Root-mean-square distance.

$$d(p,q) = \sqrt{\sum_{i=1}^{N} (p_i - q_i)^2}$$

Equation 3: Euclidean distance.

$$d(p,q) = \frac{\sum_{i=1}^{N} (p_i \times q_i)}{\sqrt{\sum_{i=1}^{N} (p_i^2)} \sqrt{\sum_{i=1}^{N} (q_i^2)}}$$

Equation 4: Angle between vectors.

Four methods were implemented in order to test their accuracy and also we wanted to avoid mistakes caused by wrongly chosen algorithm. There are invented and tested more sophisticated algorithms such as K-Means [21] or DBSCAN [22]. But our goal was to show the possibility of protocol detection using time characteristics and the complexity of such advanced algorithms was inadvisable at this phase.

We used two types of environments. The first one served for testing of SSH properties and FTS testing. When FTS was installed on our server, the real environment was set. The proper pattern vector was chosen by hand from observations.

4.1.3 Protocol Detection Results

Results of detection dictionary attacks on SSH protocol are shown in Table 1. There was a choice if we want to have as good true acceptance rate as possible or as low false acceptance rate as possible. Since we

use statistical methods, it is impossible to have 100% true acceptance rate and 0% false acceptance rate. The decision whether the flow contains an attack was done through the use of a threshold which was also found manually.

Table 1 also shows that Angle between vectors method in combination with the pattern vector from real environment has the worst results. The other methods are almost independent on the pattern vector origin.

Pattern	Average distance		Root-mean-square distance		Euclidean metric		Angle between vectors	
	TAR	FAR	TAR	FAR	TAR	FAR	TAR	FAR
	%	%	%	%	%	%	%	%
Testing	91	8	91	10	91	10	94	25
Real	88	3	88	3	87	2	78	19

Table 1: Detection methods and their accuracy.

The main goal for this part of work was to verify whether time characteristics are suitable for protocol detection. We showed that the goal was fulfilled but there are many limitations such as a pattern vector for each situation, low accuracy, etc.

4.2 Current State of Research

The following step of our work is dealing with automatized dividing of flows into groups using a clustering algorithm. The main benefit of this feature is the ability to detect all the protocols without needing to compare each single flow with the pattern. Predefined pattern is required in order to choose which group represents which protocol.

There are invented some algorithms for data clustering such as K-Means [21], DBSCAN [22] or Quality Threshold (QT) [26]. The two first methods are widely used for protocol detection and the last one was invented for genome research. K-Means and DBSCAN offer better results than QT but there has to be set the number of clusters in advance and they do not provide always the same results. QT algorithm does not provide such computational speed and accuracy but it gives always the same results and there is no need to set number of clusters. Recently we implemented QT algorithm and we are making experiments with the automatized division of flows into groups.

We wanted to find out whether we are using the best vectors. At first sight it seems like an easy task but the true is different. The vector can consist of:

- flow duration,
- number of transferred packets,
- number of transferred bytes,
- time characteristics (4),
- packet size statistics (4),
- any other new information which was not discovered so far.

In contrast to our previous research, there is no information about network and transfer layer protocol numbers because we want to make a detection method based only on protocol behavior. It causes versatility which allows the users to detect ordinary traffic even if it is encrypted or tunneled. The only difficulty is with obfuscated protocols (for example BitTorrent, Skype, etc. [13]).

We have a set of 11 currently available flow statistics which we can observe. In order to get as precise results as possible it is important to find out which subset gives the best results. The naïve approach is brute force where are tested all possible combinations. This number can be represented as (Equation 5).

$$vector\ combinations = 2^{number\ of\ vector\ components} - 1$$

$$vector\ combinations = 2^{11} - 1$$

$$vector\ combinations = 2047$$

Equation 5: Number of vector combinations.

Such a huge number of possible vector combinations cannot be explored and tested by hand. Moreover new characteristics can be invented and it significantly increases the observed field of interest. Automatized searching over the results will solve our current problem but is not implemented yet.

Since we could not use brute force method, we made more sophisticated tests where we thought about impact on the protocol behavior. For example if we consider duration of a flow, the result vector will be similar only to flows with similar length. It holds for number of transferred packets and bytes as well. It induces that there has to be some kind of making these numbers independent on their absolute size. The possible answer is to make a vector from the values and convert them into a unit vector. As a result, we will lose information about their sizes but the ratio among them will stay the same. This information can be used for protocol detection if we presume that this ratio is protocol dependent.

Similar situation will occur with average size of packets and average inter-packet gaps. They are also influenced by the protocol duration and number of transferred packets. At this time is not their grouping and converting into a unit vector obvious because their values can change nonlinearly.

The other protocol statistics can be minimal size of packets which is often 66 bytes. Exactly 66 bytes have packets that have ACK flag set. Since almost every flow has some ACK packets, the minimal size of packet seems to be meaningless information.

The absolute values of vector components will create patterns which are able to detect connections with the same values. Converting into unit vectors will make the patterns same for each protocol. As a result we will lose information about absolute values but they are not important anyway.

Grouping and deleting some vector components decreases the number of possible situations and allows us to test some of them by hand. We had made several tests and the results were not as good as we want them to be. It seems that the world is not so simple as we predicted and some other network protocol behavior influences the results.

The protocols which we are able to group at this time are HTTPS (TCP/443), DNS (UDP/53) and Nagios (TCP/5666) with probability around 80 %. The testing data were captured on our testing server where is only very limited set of protocols and hosts. In order to make the test more serious, we need to data from the real environment.

5 Future Work

We have two types of planning the future work – short and long time planning. They have different sub goals but they lead to the common one.

5.1 Short Time Goals

At first we have to find what is influencing the results. There is some hidden protocol behavior that causes our tests to be inaccurate. This part of our work is one of the most complicated because creating the proper patterns from flow statistics is the new what our work would like to bring to the world. More tests have to be done and we believe that the results will appear soon.

QT clustering algorithm was implemented for our testing purposes because of its manner of data processing. The final output of our work will use some verified algorithm with a better speed of clustering. We cannot say the name of the algorithm because tests among them have to be done.

After the previous steps are done, we can develop a plugin for NfSen which will display the data in human-readable form. NfSen is widely used among universities and companies mainly in Europe. This final step is the most important from users' point of view because if there is no tool which will apply results of our work, our research would be incomplete. Plugins for other visualization tools can be made as well but there is no reason to do them at this time.

5.2 Long Time Goals

When the plugin is finished, there has to be a network monitoring device which is able to generate input data for the detection algorithm. This task is much more difficult because it needs a lot of effort and often money. Currently there are only few devices with an interface which is able to be extended by plugins. It presents the first investment.

Another complication may appear in data processing because NetFlow version 5 is standard in many companies but the data from the network probe will be in IPFIX format. Converting the existing infrastructure from NetFlow to IPFIX is not trivial and it will need lot of human effort.

We decided to use IPFIX because it allows us to include flow characteristics there and it supports more accurate time stamps with comparison to NetFlow version 5. NetFlow version 5 has minimal time resolution only 1 millisecond but IPFIX is able to measure even 1 nanosecond. The main advantage of IPFIX is its ability to be extended by any new information. The other advantage is support for IPv6 which is only in IPFIX and NetFlow version 9.

Masaryk University will take advantages of protocol detection based on time characteristics analysis. The university is using FlowMon programmable hardware probes built in COMBO cards. We will extend them in order to generate data in IPFIX format with precise time stamps.

Creating the plugin is only up to us but its final implementation in real world is up to the network administrators and there starts the real challenge.

6 Conclusion

This paper described network protocol detection using flow statistics. Our approach is suitable for high-speed networks since there is no need to process content of each single packet as deep packet inspection does. We are measuring statistics about flows which consist of inter-packet gaps and packet sizes. The advantage is that this information can be computed in hardware network probes which are much faster

and precise in comparison with software probes. We are presenting mainly the idea in respect to the current state of research.

At this time we are able to detect dictionary attacks on SSH protocol. This first step was very important for us because we checked there that time characteristics can be used for protocol detection. The other result of this first step was decision to extend time characteristics by statistics about packet sizes. These two flow dependent characteristics will help us in detecting the whole protocols instead of one specific situation.

Currently we are working on making the detection more precise. This goal will be achieved in three different steps. We have to find the core of behavior where are no negative influences which can corrupt the results. Then the best set of vector components has to be isolated, it increases speed of data processing since no unnecessary information will be processed. At last, we have to find a clustering algorithm which will provide the best results in combination with the highest speed of data processing.

Acknowledgment

This work was supported by the Czech Ministry of Defence under Contract No. SMO02008PR980-OVMASUN200801.

References

[1] Internet Assigned Numbers Authority - port numbers, http://www.iana.org/assignments/port-numbers.

[2] Rehman, R. U.: Intrusion Detection with SNORT: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID, Prentice Hall PTR, New Jersey, 2003.

[3] Moore, A. W., and Papagiannaki, K.: Toward the Accurate Identification of Network Applications, Springer Berlin / Heidelberg, pp. 41–54, 2005.

[4] Dreger, H., Feldmann, A., Mai, M., Paxson, V., and Sommer, R.: Dynamic application-layer protocol analysis for network intrusion detection, in USENIX Security Symposium, 2006.

[5] Karagiannis, T., Papagiannaki, K., and Faloutsos, M.: Blinc multilevel traffic classification in the dark, SIGCOMM, 2005.

[6] Trestian, I., Ranjan, S., Kuzmanovic, A., and Nucci, A.: Googling the internet: Profiling internet endpoints via the world wide web. IEEE/ACM Trans. Netw., 18(2), pp. 666–679, 2010.

[7] Dewaele, G., Himura, Y., Borgnat, P., Fukuda, K., Abry, P., Michel, O., Fontugne, R., Cho, K., and Esaki, H.: Unsupervised Host Behavior Classification from Connection Patterns. Wiley IJNM: International Journal on Network Management, 2010.

[8] Erman, J., Arlitt, M., and Mahanti, A.: Traffic classification using clustering algorithms, SIGCOMM, 2006.

[9] McGregor, A., Hall, M., Lorier, P., and Brunskill, J.: Flow Clustering Using Machine Learning Techniques, in Passive and Active Measurement Conference (PAM), Antibes Juan-les-Pins, France, 2004.

[10] Moore, A. and Zuev, D: Internet traffic classification using bayesian analysis techniques, In Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp. 50–60, 2005.

[11] Zuev, D. and Moore, A: Traffic classification using a statistical approach., Passive and Active Network Measurement, pp. 321–324, 2005.

[12] Bernaille, L., Teixeira, R., and Salamatian, K: Early Application Identification, in ADETTI/ISCTE CoNEXT Conference, Lisboa, Portugal, 2006.

[13] Hjelmvik, E., John W.: Breaking and Improving Protocol Obfuscation, Technical Report No: 2010-05, Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden, 2010.

[14] Hjelmvik, E.: Documentation of spid attribute-meters, http://sourceforge.net/apps/mediawiki/spid/index.php?title=AttributeMeters, 2009.

[15] Claffy, K., Polyzos, G. C., and Braun, H.: Measurement Considerations for Assessing Unidirectional Latencies, Internetworking: Research and Experience, pp. 121–132, 1993.

[16] Liberouter, http://www.liberouter.org/.

[17] Brownlee, N.: RFC 2722, The University of Auckland, http://www.ietf.org/rfc/rfc2722.txt, 1999.

[18] NfSen, http://nfsen.sourceforge.net/.

[19] Nprobe, http://www.ntop.org/nProbe.html.

[20] FlowMon, http://www.invea-tech.com/products/flowmon.

[21] Hamerly, G., and Elkan, C.: Learning the K in K-Means, In Neural Information Processing Systems, MIT Press, 2003.

[22] Ester, M., Kriegel, H., Sander, J., and Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise, AAAI Press, pp. 226-231, 1996.

[23] RFC 4251, http://www.ietf.org/rfc/rfc4251.txt.

[24] RFC 4252, http://www.ietf.org/rfc/rfc4252.txthttp://www.ietf.org/rfc/rfc4252.txt.

[25] iSiLK, http://tools.netsa.cert.org/isilkhttp://tools.netsa.cert.org/isilk.

[26] Heyer, L. J., Kruglyak, S., and Yooseph, S.: Exploring expression data: Identification and analysis of coexpressed genes, in Genome Research, 9(11):1106-1115, 1999.