IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 10, Issue 3 (Mar. - Apr. 2013), PP 01-10
www.iosrjournals.org

Analysis of service-oriented traffic classification with imperfect traffic classification methods

[1]Vivek.A, [2] Asst Prof. Balaji.S, [3] Dr.S.Anbu
[1]PG scholar P.B College of Engineering
[2]CSE Dept P.B College of Engineering
[3]HOD CSE Dept P.B College of Engineering

Abstract: *Typically the traffic through the network is heterogeneous and it flows from multiple utilities and applications Considering todays threats in network there is yet not a single solution to solve all the issues because the traditional methods of port-based and payload-based with machine learning algorithm suffers from dynamic ports and encrypted application.Many international network equipment manufactures like cisco, juniper also working to reduce these issues in the hardware side.Here this paper presents a new approach considering the idea based on SOTC.This method adapts the current approaches with new idea based on service-oriented traffic classification(SOTC) and it can be used as an efficient alternate to existing methods to reduce the false positive and false negative traffic and to reduce computation and memory requirements.By evaluating the results on real traffic it confirm that this method is effective in improving the accuracy of traffic classification considerably,and promise to suits for a large number of applications.Finally, it is also possible to adopt a service database built offline, possibly provided by a third party and modeled after the signature database of antivirus programs,which in term reduce the work of training procedure and overfitting of parameters in case of parameteric classifier of supervised traffic classification.*

Index Terms—*Network operations, traffic classification, security.*

I. Introduction

The major challenge for administrators of Intrusion Detection Systems is distinguishing between events that are genuine malicious activity and those that are false positives.. On the one side, network managers want to know precisely the type of traffic transmitted over their networks to enforce various polices such as for quality of service (QoS), security, management, and more. On the other side, an increasing number of applications tend to hide their behavior (through encryption, tunneling, etc.) trying to avoid limitations imposed by such policies. Many international network equipment manufactures like cisco, juniper also working to reduce these issues in the hardware side.

Traditionally, traffic classification relies on the port based method, which exploits transport layer information (source and destination TCP/UDP ports). However, this method has many limitations that make it quite imprecise and inefficient despite its extensive usage. Not all servers respect well-known ports conventions, malicious software can use well-known ports in order to let its traffic pass through port-based security restrictions, many peer-to-peer applications actively try to avoid classification using random ports, network tunnels can be instantiated using well known ports in order to avoid imposed traffic restrictions, IP payload encryption hides the port numbers.

An evolution of this approach relies on payload-based inspection that is used in most commercial devices and is declined in different flavors [4]. This technique shares some of the problems of port-based classification (encrypted protocols, tunneling) and is perceived as really expensive from the computational point of view. Other classification techniques that aim at identifying applications based on their behavior as inferred from observed traffic (statistic traffic analysis or heuristic analysis) are being studied,but are far from being ready for commercial deployment.

This paper presents a new classification technique that, in somerespect,is orthogonalto the above mentioned mechanisms.This approach *service-oriented traffic classification(**SOTC**),* called exploits information about services previously discovered in the network in order to classify traffic flows. Main advantages of this method are robustness, accuracy, a limited use of processing power, reduced memory requirements in classifying the false positive and false negative from the network traffic and the capability to use any classifier in the early stage of the classification (namely, the *service identification* phase).

II. Related Work

Currently deployed network classification algorithms generally fall in one of two categories: payload based algorithms and behavioral algorithms. This section provides a brief overview of the state of the art in network traffic classification focusing on some of the most relevant algorithms in each category.

Payload-based classification is applied by most commercial solutions for various purposes ranging from statistics to security, because it provides the best trade-off between the classification accuracy and the coverage in terms of number of recognizable protocols. A possibly deep inspection of data transported within packets is used to identify the flow packets belonging to and the application generating it. In fact, by inspecting the headers of the higher layer protocols, possibly up to the application layer payload, it is possible to precisely identify the protocol being used by the application possibly gather information on the type of traffic it generates. However, the correct identification of a protocol is not straightforward. One approach relies on searching for patterns or regular expressions that can uniquely identify each protocol; a database containing the description of each protocol is needed. Many payload based solutions have been proposed [2] [3], some coupled with an approach for describing network protocols in order to make classification code easy to reuse and update [5][6]: classification of additional protocols or new versions of existing protocols can be achieved by simply adding their description, without the necessity of any modification to the classification software itself. Known problems of payload based classification algorithms are (*i*) high sensitivity to packet loss and TCP/IP fragmentation and segmentation issues, (*ii*) hard and time- consuming task of creating protocol signatures, that are crucial to the effectiveness of the solution, (*iii*) encryption and/or tunneling that hinders access to data contained into application layer headers and payloads, and (*iv*) significant requirements in terms of computational and memory resources that actually make traffic classification at high line rates difficult. Due to the high computational requirements of deep packet inspection, payload based classification algorithms usually limit pattern searching to the initial packets of each flow. According to this method, named Packet Based – Flow State in [4], once the protocol transported by a flow has been recognized, the flow identifier (i.e., the 5-tuple including IP addresses, ports, and transport layer protocol) and the corresponding application-layer protocol are added to a data structure in memory, often called *session table*, that is maintained as long as the flow is active[1]. The main critic moved toward these methods is about the memory usage for maintaining flow state information; in case of large networks, the size of such per-flow state grows significantly and this might become an issue. Furthermore, additional memory is required because pattern matching usually relies on regular expressions, which are well-known for their memory consumption due to the necessity of maintaining graph-based structures representing Deterministic Finite Automata. On the other side, also processing requirements may be problematic due to regular expression matching and to session table management (lookup, insertion, deletion, etc.). These problems become even worse in the Message Based – Protocol State flavor [4] of the payload-based method (implemented in Binpac [6] and SML [7]), that needs to rebuild the entire application-layer message to enable the analysis of the entire data in order to achieve the precision required for security appliances. In this case, the amount of information to be maintained grows even more, as do processing requirements for session reconstruction and application-layer processing, although some smart method can be devised in order to decrease this complexity [18]. It is important to notice that [4] demonstrates that the simpler Packet Based – Flow State approach is in most practical cases sufficient for the vast majority of applications.

Another approach in traffic classification relies on behavioral techniques, whose main assumption is that each application is characterized by some specific behavior. Applications can then be identified by just gathering information at different levels (e.g., packet inter-arrival time, jitter, packet size, etc.) and analyzing it (e.g., from a statistical point of view), often without inspecting protocol headers and application data transported. Therefore behavioral algorithms are not affected by any of the shortcomings of payload based algorithms related to information hiding (e.g., by encryption) or camouflage (e.g., by using ports typically deployed by specific services). Specifically, behavioral algorithms work the same way independently of whether flows use encrypted payloads or not. Unfortunately, behavioral algorithms have some common limitations; first of all, most of them typically require a pre-classified traffic trace in order to train the classifier before it can start working. These pre-classified traces are usually classified using payload-based methods, manual inspections and human experience; although there are few guaranties about the actual precision of these pre- classified traces, all measurements are done starting from an imprecise base. Furthermore, a wide class of behavioral methods needs to be trained in exactly the same conditions of the environment where they are going to be deployed, which often prevents the training sets obtained in one site from being usable as a trainer set in other places. Additional problems are related to the limited temporal validity of the training set due to network reconfiguration and long term variations, and to the fact that these algorithms often need to observe a fairly large number of packets before they can work properly.

Behavioral algorithms can be further organized into three sub-categories. *Machine learning algorithms* [9] [10] [11] [12] [13] deploy advanced analysis techniques, such as clustering algorithms, to divide network flows in different classes based on information devised without inspecting application layer payload. *Statistical algorithms* [14] process statistical properties of network flows through mathematical function, like Bayesian filters, in order to derive a statistical "fingerprint" for each application. Typical data analyzed by these algorithms are round-trip-time, inter-arrival time, inter-arrival jitter, mean packet size. *Heuristic algorithms* evaluate how each host act within the network in order to identify the applications that hosts are running. Some examples of data analyzed by these algorthims are the order of requests/responses produced by a host, number of hosts contacted, number of ports deployed.

III. Service-Oriented Traffic Classification

Service-Oriented Traffi Classification is a surprisingly simple idea that relies on the observation of how hosts usually interact and on the assumption that certain hosts, usually called *servers*, perform similar interactions, usually offering a *service*, with multiple other hosts over a certain time span. This assumption, which provides the foundation of our method, will be verified through experiments on real network data.

According to the classic client-server paradigm, a potentially large number of hosts connect to a single one to obtain a service. In this situation it is easy to identify the server as a main actor with a long lasting role as it usually offers the same service at the same "network coordinates" (IP address and TCP/UDP port) for a long time. The basic assumption in service-based classification is that knowing

which service is offered at an IP address/port pair, a classifier can infer that all sessions directed toward that pair will access such service. For example, if the classifier knows that host www.polito.itis running a web server on TCP port 80, it can classify all sessions established to this IP address/port pair as HTTP traffic. It is important to notice that such a classifier does not work like a port based classifier. While the latter assumes that a session is transporting HTTP because it is connected to TCP port 80, a service-based classifier *knows* that www.polito.it is running a web server on TCP port 80. When the classifier discovers a service, it stores the triple identifying it — i.e., IP address (of the server), TCP/UDP port (at the server), and transport protocol in an appropriate structure in memory called *Service Table*.

The same principle can be applied to hosts running peer-to- peer applications. In this case the application has a client part and a server part running simultaneously: the client part of a peer establishes sessions to the server part of other peers awaiting for connections at a specific port. How this port is assigned and communicated to the other peers depends on the specific application and protocol, but the key point is that the port used to receive connections from other peers usually does not vary very frequently and is reused many times for the same instance of the peer-to-peer application. So when the client part of a peer connects to the server part of another peer to transfer information, the service-based classifier identifies the server part of such session as a service and stores the associated triple in the service table. Also peer-to-peer applications that use the same port for both the server part and the client part, such as Skype for example, are handled properly. After a peer A has received a connection to its server part, a triple containing its IP address and port is created in the service table as a service. When its client part connects to another peer B, the service-based classifier classifies the corresponding packets according to either A's service entry or B's service entry. Although classification based on A's service entry is in principle mistaken as packets are being exchanged as part of a session whose server side is B, the packets are anyway correctly classified as belonging to the peer-to-peer application at hand. When an application shows such behavior (which is not uncommon among P2P software) our approach can be extended by adding also the client-side of a session to the service table, which will become the server part in a later data exchange, for all traffic belonging to that application.

It is important to notice that finding out which service is running at a certain IP address/port pair (i.e., *service identification*) is orthogonal to the service-based approach: in principle, any method can be used to perform service

Table-1 List of false positivies seen on an internal network

Event	Parameters	Possible cause
HTTP port probe	port = 80	Attempted web connection to a machine without web services running.
NetBIOS port probe	port = 139	Attempted drive mapping to a machine without NetBIOS services running
SNMP port probe	port = 161	Enterprise management & monitoring systems.
TCP/UDP port probes	port = varies (e.g. TCP 443 TCP 445)	Monitoring agents failing to connect to busy collection servers. Browsers failing to connect to busy HTTPS services. Windows 2000 machines attempting direct host connections to their peers.
TCP/UDP port scans	port = varies (e.g. port1|port2|port3..)	Network backup agents communicating with busy backup servers

SMB empty password	account = varies
	share = varies
		Connection to non-password protected shares

SNMP backdoor	community = varies	Network using default SNMP Passwords
ICMP subnet mask request, Ping sweep	(null)	Network management and Monitoring.
TCP SYN flood	percent from intruder = varies	Offline services returns online (e.g. email services).
ICMP subnet mask request	(null)	Network management and monitoring tools

identification (payload-based, heuristic, or even manual inspection, and more). The service-oriented approach assumes to know precisely the service associated to an IP address-port couple and from that point on it will guarantee a precise identification of that traffic. Obviously, service identification is not straightforward and its effectiveness has an impact on the outcome of service based classification, as discussed later.

Service-oriented classification features interesting advantages over other classification methods. *Encrypted traffic* at application layer can be properly classified provided that the corresponding service has been previously identified, i.e., it has an entry in the service table. It offers *pattern segmentation transparency*, i.e., a flow can be properly classified even though protocol identifying patterns are split across multiple packets, avoiding the complexity of reassembling application data units. A service-oriented classifier needs to maintain only information about services (i.e., IP address, port, transport protocol and service offered) independently of the number of traffic flows actually using such services; hence it has *limited memory requirements*.The limited amount of state information kept by a service-orinted classifier impacts (*i*) *scalability*, performance (*ii*) *lookup time* and (*iii*) *hardware implementations* that can rely on faster on-chip memory. Classification of a packet belonging to a known service requires a single lookup on three fields (IP address, port and transport protocol) in a relatively small lookup table, therefore with *low computational cost*. Moreover, service identification, which might have higher computational cost, is expected to be performed only on a small fraction of the packets and it can be even performed offline; in any case, service identification is orthogonal to the service-based method. Finally, as we said, service-oriented classification is among the few methods that guarantees *early classification*, i.e. being able to classify even the first packet (e.g., a TCP SYN) within each session, while other methods need to process at least the first few packets within each session.Table-1 shows the list of possible false positives which occurs on a network.

Service-oriented classification also has some potentially critical issues.Its effectiveness, in terms of minimizing both misses and wrong matches, and also its performance heavily depends on identification of network services that must be as accurate as possible. A wrong entry in the service table leads to wrongly classifying a potentially large number of flows, while a missing entry possibly leads to both a failing classification of a large number of flows and deploying significant amount of computational resources in an effort to identify the service being used, e.g., by deeply inspecting the corresponding packets. Consequently, a successful service-oriented classification is tightly coupled to a robust and effective service identification solution, which as we said, is orthogonal to service-oriented classification.

In addition, not keeping information about individual sessions, service-oriented traffic classification is not suitable for applications that require such granularity level, such as, for example, per-session enforcement of QoS policies. A service-oriented classifier can be customized for such applications to keep an additional session table for those services requiring so, which is a simple extension that can be added to any implementation.

IV. Implementing *Sotc*

Although the service-oriented traffic classifier looks simple and elegant, some issues need to be addressed to make it working properly. This section presents such issues and gives some insight in how they havebeen addressed in our implementation. Given the generality of the service-oriented method, other implementation strategies can be adopted.

A. Service identification

Given the expertise and previous work of the authors, a payload-based implementation of a service identification module has been an obvious choice. In particular, an existing packet processing engine based on the Network Packet Description Language (*NetPDL*) [1] [5] has been reused in the implementation of the service identification module. NetPDL is an application-independent packet format description language that enables the creation of a generic protocol description database: the NetPDL database, in fact. Although it includes only packet header formats and does not support the description of protocol temporal behavior (e.g., a protocol state machine), it has proved being extremely effective and robust with respect to traffic classification [4], thanks to an extension that enables management of lookup tables, originally used to maintain transport-level sessions [5]. The high flexibility of NetPDL makes the engine suitable for the implementation of the service-based classifier as well, in addition to the payload-based service identification module.

The main modification made to the NetPDL engine is the addition of some new tables, such as the service table that contains information about services. The process starts with an empty service table, while traffic is processed by extracting IP addresses and ports from each arriving packet. Since the server side of the communication cannot be inferred on a packet-basis, the service table is looked up twice: once with the source identification (source IP/port) and once with the destination identification. If one of these lookups is successful, the packet is classified through the service-oriented method. Otherwise, as depicted in Figure 1, the service identification module performs a payload-based classification to possibly introduce a new entry in the service table containing the IP address and the transport layer port used by the server side of the session and the application protocol associated. Any new packet toward this “known service” can subsequently be classified directly through the information kept in the service table as described above without any further processing (e.g., payload inspection). Please note that the identification of the server side of the connection is not straightforward and will be discussed.

As time passes, more and more traffic will be classified by the service- based method since the service table will include an increasing number, possibly most, of the services active in the network.

B. Understanding the clients and servers

The server side of a TCP session can be easily identified by observing the SYN and ACK flags during in the three-way handshake of the TCP protocol. In our implementation we use an additional lookup table, called *Candidate Service Table*, in which a new entry is added with the IP address and port of a host that accepted an unclassified TCP session by generating a TCP packet with both the SYN and ACK flag enabled. The Candidate Service Table is required to keep track of the server side of a session because the service is possibly identified, e.g., through payload inspection, once the session has been opened, i.e., when the SYN/ACK flags, used only during the initial handshake phase, are not available to enable the identification of the server side. When the service is finally identified, the server information is moved from the candidate service table the service table.

Entries of the Candidate Service Table are subject to a very fast ageing (about ten seconds [19]) in order to avoid their number to explode over time due to sessions opened by unidentified services, unsuccessful handshakes, or unused opened sessions, as in cases of malicious activity such as SYN flooding and port scanning.

With UDP services identifying the server is different Since explicit information like the SYN flag in the TCP case is not available. Although, especially with the growing adoption of broadband multimedia applications, UDP is expected to significantly increase its traffic share, possibly becoming predominant, this paper focuses on TCP traffic, which as of today accounts for the vast majority of data. UDP traffic classification, that requires a non-straightforward extension of what is proposed in this work, is left to a companion future paper.

C. Managing the service table for false positive and false negative determination

Besides properly populating the service table, an important issue is the prompt elimination of service entries once the corresponding service is no longer provided. This is important in order to avoid the explosion of the number of service entries and that a service offered only temporarily leads to classification errors. One possible approach is to purge an entry that does not make a hit for a certain amount of time, hereafter referred to as *service inactivity timeout.* As a further refinement, the service inactivity timeout can be differentiated for different service classes. For example, some services are offered over a long time period, possibly permanently, even with a low connection rate, and their entries are given a long service inactivity timeout. A typical example of this service class is an SMTP server contacted only few times in a day, but providing its service over a very long time period. Vice versa, other services have a naturally short life and the inactivity timeout associated to their entry may be shorter. Typical examples are peer-to-peer applications.

Assigning distinct service inactivity timeouts to different classes of services, although not strictly necessary, is useful in avoiding multiple re-identification of long-term services, e.g., through costly deep packet inspection. On the other hand, assigning an entry to the long-term service category is critical because if the service is not actually long-term or it has been wrongly identified, the entry can lead to persisting classification errors. Consequently, there should be a certain level of certainty about service before categorizing it as a long-term one. One possible policy is to set any newly identified service "under observation": its entry is categorized as short-term and some additional checks are performed on packets classified according to the entry. For example, payload inspection can be executed on randomly chosen new sessions. After a certain period of observation confirming the initial identification, hence the long-term nature of the service, the corresponding entry can be categorized as long-term. Another policy can be to categorize services as long-term only through explicit (e.g., manual) configuration.

With respect to the scalability of service-oriented classification, it is worth noticing that the management of the service table is independent of the classification process and

can be implemented as a distinct process running separately from the core classification process.And invoking this service-oriented result to determine the false positive and false negative to classify the network traffic.

V. Experimental Evaluation

This Section provides an experimental evaluation of service-based classification, including some problems that arise in its implementation. The next section first devises the benefits expected by the deployment of service-based classification from an analysis of network traffic itself — i.e., not based on the results of particular classification experiments — which provides a more general assessment of the potential of service based classification. Then, the results of specific classification experiments are reported to substantiate such general assessment.

A. General Assessment

Before implementing our service-oriented traffic classifier we collected a set of session-related statistics on the link that connects our University to the Internet to assess the potential benefits of sotc classification in terms of memory occupancy to distinguish false positive and false negative, i.e., if the number of services was really smaller than the number of sessions. These measurements, done using *Tstat* [15] and lasting several days, wanted to determine the maximum number of service entries required to classify all the traffic with a *sotc* approach, compared to the number of session entries required by a classifier based on session identification. The obtained results must be intended as a lower bound of the session/service table size since they account for the session/services present and actually active at any given time. A TCP session is considered closed when a FIN or RST packet is observed; in case of abnormal termination, a 10-minutes session inactivity timeout is used to declare a session terminated, as suggested in [22] and 0. Analogously, services are considered closed if no traffic is observed in an idle period of the same duration.

Figure 2 shows, for each minute, the number of active traffic sessions and the corresponding number of services on the uplink (100 Mbps) of our university network (about 6,000 hosts) over a 7-day period. The average number of active traffic sessions is 80,000 with peaks of 180,000, while the total number of services never exceeds 10,000. Figure 3 shows the same figures for a traffic trace[2] from the MAWI wide traffic archive [21]. The average number of active session is 120,000 with a peak of 380,000, while the total number of services never exceeds 10,000. The average on the whole observation period of the *session to service* ratio is about 20 for both traces, which means that a service table requires roughly 20 times fewer entries than a session table.

Furthermore, a service entry is smaller than a session entry, thanks to the smaller number of information that has to be stored. This is beneficial in terms of memory requirements as well as both processing requirements and performance for session/service information look-up.

Observed sessions	40503
Observed services	21675
Observed applications	81
Services in which sessions are classified univocally as belonging to the same	21042

Table-2 Observation results

The tool has been installed on 11 hosts (with Linux, Windows and MacOS-X operating systems, running several applications; among the other Skype, Emule, Joost, uTorrent), the traffic produced has been captured for 4 days and the traffic traces have been analyzed by a payload-based classifier.

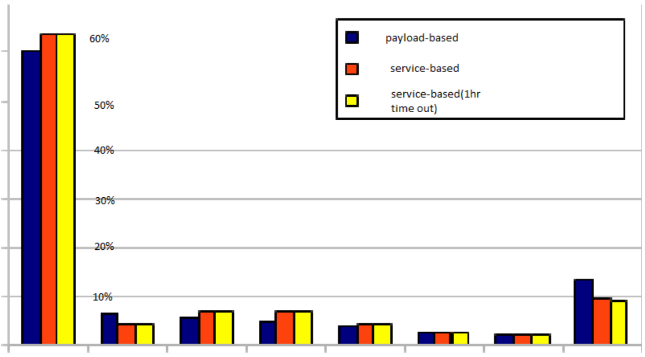


Fig-1 Comparision

An important observation is that simply increasing the service inactivity timeout may not be a good idea, since we may end up filling the service table with entries related to one- shot services or services that are anyway not any longer active, which will never appear again in the future. This is evident in Figure 2 that shows an almost four-fold increase of the service table size when changing the service inactivity timeout from 10 to 60 minutes— without any appreciable advantage in terms of classified traffic, Therefore, a 10 minute service inactivity timeout has been used in the experiments producing all the results presented in this discussion.

B. Accuracy

Our tests show that service-based classification offers an improvement in classification accuracy over results obtained with the original payload-based classifier. For example, trace *Weekend* contains a significant amount of traffic generated by eDonkey that hinders payload-based classification when application-layer data is encrypted. The payload-based classifier recognized only a small percentage of the flows generated by these applications, e.g., some sessions that are occasionally sent in clear and that represent special cases. For example, Skype sometimes produces packets that are only partially encrypted and consequently can be properly inspected and classified; similarly, not all eDonkey messages are encrypted. In all the other cases, the payload-based classifier is unable to identify the protocol transported and it marks flows as unknown, as it is shown by the high percentage of unknown traffic in Figure 1. Experimental evaluation also showed another problem related to the completeness of the pattern database used by the payload- based method. In fact, some unknown traffic is related to flows that use particularly rare or undocumented application level messages that are not part of the pattern database of the payload-based classifier. Service-oriented traffic classification does not have this problem, because once a service has been identified thanks to the presence of some known signatures in application-level messages, following sessions are classified based on the network coordinates they are related to. This is confirmed by Figure 1 where the service-oriented traffic classification leaves a much smaller amount of traffic as unknown, while classifies as eDonkey a much larger percentage of traffic than payload-based classification. Results reported in Figure 2 are referred to the percentage of packet classified; results are slightly worse in terms of bytes, in which the percentage of the unknown traffic is 11%.

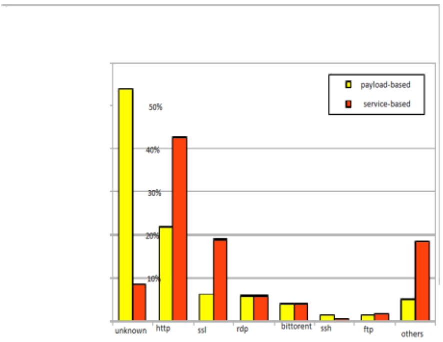


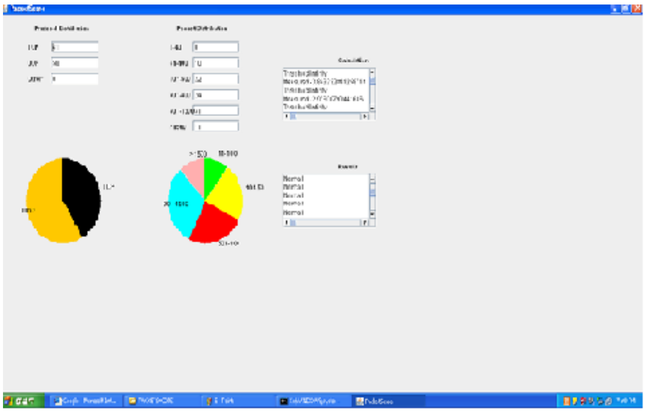
Fig-2 Comparision of both service and payload-based classification

Payload based classification on trace *WorkingDay* results in a low percentage of unknown traffic because the trace includes mainly HTTP traffic. However, service-based classification results in improved accuracy also on the *WorkingDay* trace. Figure 1 focuses only on the unclassified traffic of Figure 2 and shows how this traffic has been classified by the service-based classifier. For instance, among the 54% of unclassified traffic of the *Weekend* trace, about
18% was eDonkey, 14% RPC (which is included in the "others" bin in Figure 2), and more. Manual investigations on a randomly chosen subset of classified flows confirm that the outcome of the service-based classifier is correct.

C. Scalability

Scalability must be assessed in terms of memory and processing requirements.
From the processing side, the computational complexity of a classification solution is an important index of its scalability. Profiling done on our classification code (written in C/C++) confirmed that the cost for a lookup in the service table (i.e. the main cost associated to each packet by the service-based method) is 37 times lower than the cost for a pattern matching on the payload (9700 clock ticks against 260^7). Fig-3 shows a GUI screen which produces the resultsof service-oriented classification to find the false positive and false negative traffics in the network. Although the asymptotic processing cost remains the same in both service-oriented and payload-based classifier (in the unfortunate case in which each service is associated to a single session), in practical terms our method guarantees a speed-up of more than an order of magnitude at best.

In summary, the performance and scalability improvements of service-oriented classification over payload-based classification is directly proportional to the percentage of traffic classified by the service table, i.e., without performing payload inspection.



Normal
Normal
Normal
Normal
Normal

Fig-3 Classification of traffic

VI. Conclusions

The major challenge for administrators of Intrusion Detection Systems is distinguishing between events that are genuine malicious activity and those that are false positives.This paper presents a new idea for traffic classification, named *service-oriented traffic classification*, that is, in some respect, orthogonal to the other classification techniques. This method introduces also in the traffic classification arena the concept of *fast path*, through which the vast majority of the traffic is processed with a limited use of processing and memory resources —ultimately in a short time— and a *slow path* that is invoked in a limited number of cases. Experimental data confirm that services are very stable even over long periods, making this method extremely simple,efficient and robust to classify the false positive and false negative netwrork traffic. Particularly, robustness is achieved because this method does not require the analysis of all sessions: provided that a service has been previously recognized, sessions accessing it can be classified even if encrypted at application-layer or data flow is observed only in one direction. Results in terms of efficiency are impressive, leading to a 37x reduction in processing cost, and a 20x reduction in the number of entries in data structures compared to session based classifiers at least in the traffic trace examined; furthermore each entry being half the size. Real-time measurements on the actual traffic transmitted on the upstream link of our University show that roughly 81% of the packets and 93% of the traffic (in terms of bytes) is successfully classified with the proposed method.Furthermore,service based classification is among the few methods that guarantee early classification, including the initial TCP handshake of a session. Among the few drawbacks of this method is the impossibility to classify IPsec traffic. It is worthy noticing that the precision of the service identification process is crucial for obtaining high-quality results, since a mismatch inservice identification will impair the classification of all the sessions related to that service.

References

[1] Cisco: a network device manufacturer. http://www.cisco.com

[2] J. Bellardo and S. Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *Proceedings of the 11th USENIX Security Symposium*, pages 15–28, Washington D.C, USA, 2003.

[3] A.Patwardhan, J.Parker, M.Iorga, A. Joshi, T.Karygiannis and Y.Yesha "Threshold-based Intrusion Detection in Adhoc Networks and Secure AODV"Ad Hoc Networks Journal (ADHOCNET), June 2008.

[4] S. Sen, O. Spatscheck, D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. *Proceedings of World Wide Web Conference*, pp. 512-521 NY, USA, May 2004.

[5] P. Haffner, S. Sen, O. Spatscheck, D. Wang, D. 2005. ACAS: automated construction of application signatures. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data*, pp. 197-202, Philadelphia, USA, August 2005.

[6] F. Risso, A. Baldini, M. Baldi, P. Monclus, O. Morandi. Lightweight, Session-Based Traffic Classification. *Proceedings of the IEEE International Conference on Communications* (ICC 2008) - Advances in Networks & Internet Symposium, Beijing, China, May 2008.

[7] F. Risso, A. Baldini, F. Bonomi. Extending the NetPDL Language to Support Traffic Classification. In *Proceedings of IEEE Globecom 2007*, Washington, D.C, USA, November 2007.

[8] G.Varghese, J.A. Fingerhut, F. Bonomi. Detecting Evasion Attacks at High Speeds without Reassembly. *Proceedings of ACM SIGCOMM 2006*, Pisa, Italy, September 2006.

[9] J. Erman, A. Mahanti, M. Arlitt. Traffic Classification using Clustering Algoritms. *Proceedings ACM SIGCOMM Workshop on Mining Network Data (MineNet 06)*, Pisa, Italy, September 2006.

[10] J. Erman, A. Mahanti, M. Arlitt, C. Williamson. Identifying and Discriminating Between Web and Peer-to-Peer traffic in the Network Core. *Proceedings of the 16th International World Wide Web Conference (WWW)*, pp. 883-892, Banff, Canada, May 2007.

[11] N. Williams, S. Zander, G. Armitage, Evaluating Machine Learning Algorithms for Automated Network Application Identification. *CAIDA Technical Report* 060410B, April 2006.

[12] R. Pang, V. Paxson, R. Sommer, L. Peterson. Binpac: a yacc for writing application protocol parsers. In *Proceedings of the 6th ACM SIGCOMM on Internet Measurement*, pages 289-300, Rio de Janeiro, Brazil, October 2006.

[13] O. Reviv. Inside network programming with SML. *EE Times*, August.

[14] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *Proceedings of ACM SIGCOMM*, pages 229–240, Philadelphia, PA, August, 2005.

[15] A. Este, F. Gringoli, L. Salgarelli, Machine Learning techniques for traffic classification: an approach based on Support Vector Machines. *Technical Report*, November 2007.

[16] T.T.T. Nguyen, G. Armitage. A Survey of Techniques for Internet Traffic Classification using Machine Learning. To appear in *IEEE Communications Surveys & Tutorials*, (4th edition 2008).

[17] M. Crotti, M. Dusi, F. Gringoli, L. Salgarelli. Traffic Classification through Simple Statistical Fingerprinting. *ACM SIGCOMM Computer Communication Review*, Vol. 37, No. 1, pp. 5-16, Jan. 2007.

[18] M. Mellia, A. Carpani, R. Lo Cigno. TStat: TCP STatistic and Analisys Tool. *Proceedings of the 2nd International Workshop on Quality of Service in Multiservice IP Networks* (QoSIP2003) - LNCS2601, Milano, Italy, February 2003.

[19] Measurement and Analysis on the WIDE Internet Working group traffic archive, *http://tracer.csl.sony.co.jp/mawi/*