GTVS: Boosting the Collection of Application Traffic Ground Truth*

Marco Canini[1,**], Wei Li[2], Andrew W. Moore[2], and Raffaele Bolla[1]

[1] DIST, University of Genoa, Italy
[2] Computer Laboratory, University of Cambridge, UK

Abstract. Interesting research in the areas of traffic classification, network monitoring, and application-oriented analysis can not proceed without real traffic traces, labeled with actual application information. However, hand-labeled traces are an extremely valuable but scarce resource in the traffic monitoring and analysis community, as a result of both privacy concerns and technical difficulties. Hardly any possibility exists for payloaded data to be released, while the impossibility of obtaining certain ground-truth application information from non-payloaded data has severely constrained the value of anonymized public traces.

The usual way to obtain the ground truth is fragile, inefficient and not directly comparable from one's work to another. This paper proposes a methodology and details the design of a technical framework that significantly boosts the efficiency in compiling the application traffic ground truth. Further, a case study on a 30 minute real data trace is presented. In contrast with past work, this is an easy hands-on tool suite dedicated to save user's time and labor and is freely available to the public.

1 Introduction

The collection of ground-truth application information of the Internet traffic is critical to both the research community and the industry:

- it is the basis to build and the only way to evaluate applications for network monitoring, information assurance, and traffic accounting,
- it facilitates the research on nearly every aspect related to applications, protocols, network modelling and data analysis, and
- it provides accurate knowledge of how people use the network which is increasingly important for network security and management.

However, due to privacy concerns, hardly any payloaded data can be released, while publicly accessible non-payloaded traces (e.g., LBNL and MAWI) are of limited value without the associated application information. A common practice becomes to obtain the ground truth from payloaded traces by hand.

* This work was supported by the Engineering and Physical Sciences Research Council through grant GR/T10510/02 `http://www.cl.cam.ac.uk/research/srg/netos/brasil/`

** This work was done while Marco Canini was visiting the Computer Laboratory, University of Cambridge.

M. Papadopouli, P. Owezarski, and A. Pras (Eds.): TMA 2009, LNCS 5537, pp. 54–63, 2009.

Many different, although inherently similar, approaches have been used in the past to obtain the ground truth: Moore and Papagiannaki [1] documented a fine-grained classification scheme comprising nine identification methods. The ground-truth labels used in [2] were based on "hand-classification", while the authors in [3] and [4] were using an "automated payload-based classification" as they described. The collection of many (if not all) of these ground-truth data was automated using extensible NIDSes such as Snort and Bro, or through homemade scripts. The efforts made to collect the ground-truth data were both significant and highly improvised, causing a lot of unnecessary, repeated labor, untrustworthy results (e.g., ground truth derived by signature matching alone) and inconsistency (e.g., different levels of completeness) between different works. Further, there is often a lack of verification mechanisms between multiple information sources, hence faults are inevitable and unrecoverable.

In this paper, we present GTVS (Ground Truth Verification System), a novel framework and methodology dedicated to boost the collection of application ground truth. It reduces the time and labor required in the process by automating the data manipulation and information retrieval processes as well as significantly increasing the efficiency of the hand-verification methodology. It facilitates validations among many information sources to improve the general quality of the ground truth. It works at a finest granularity of a bi-directional flow defined by the IP 5-tuple (IP addresses, transport ports and protocol) but provides aggregated views that allow for better recognition of the host behaviors and it uses heuristic rules based on multiple criteria that accelerate the verification. It is extensible to allow additional information sources and user-defined heuristics, and to achieve different goals. Finally, it provides a neat and handy platform to facilitate the management of hand-verification projects and to allow experiences and data to be shared among the research community.

The following section reviews and validates an important assumption used in our work. Section 3 presents an overview of the GTVS framework. Then, Section 4 presents a detailed case study on a 30 min trace to guide the readers through our ground-truth verification process. Related work is discussed in Section 5 and Section 6 concludes the paper.

2 Assumption and Validation

We observe that flows belonging to the same service or application often share a subset of the IP 5-tuple, notably, the {dst IP, dst port} and {src IP, dst IP} sub-tuples (where dst refers to the server side). This leads to an assumption that underpins our approach: *flows of the same sub-tuples are associated to the same service or application.* With this, the data can be labeled at high-level aggregations. Similar assumptions are implicitly used in BLINC [3] where traffic is classified using graphlets which essentially resolve into sub-tuples.

The consistency between the application label and the two sub-tuples was validated using two day-long traces [1] which had been previously hand-classified and several segments of most recent data. This consistency assumption holds for most cases with few exceptions as separately discussed below.

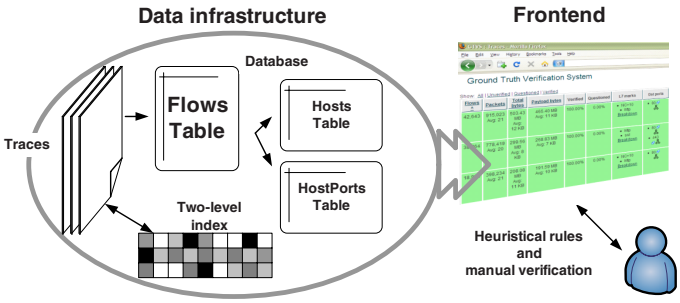


Fig. 1. GTVS: A structure overview

The {dst IP, dst port} sub-tuple. Exceptions are different application encapsulated in VPNs and SOCKS proxies. In our settings, this traffic is currently categorized into the "remote access" class. Others have discussed further mechanisms that can be applied to identify the encapsulated applications [5].

The {src IP, dst IP} sub-tuple. Exceptions include VPNs and SOCKS proxies as well as circumstances where there are multiple applications between a server and a client, e.g., a server operating both an ssh and a web[1] server. However, in such circumstances, the server is usually operating traditional services (e.g., web, ftp, mail or ssh). This sub-tuple effectively complements the one above in classifying applications on multiple ports (e.g., ftp transfers, or P2P).

3 Overview

GTVS can be described as a user-oriented design in a layered structure, as shown in Figure 1. It is composed of **(i)** a basic infrastructure layer for data management including packet traces and flows database, **(ii)** an information-rich frontend from which the user can retrieve all information related to the flows at different aggregations, and **(iii)** a verification process accelerated by flexible heuristic rules. A detailed description for each layer is presented below.

3.1 Data Infrastructure

The data infrastructure processes payloaded traces to collect information on different levels of aggregation (flows, {IP, port} tuples, and hosts).

The trace is organized into files of relatively small size (e.g., 512MB) and is indexed by the timestamp of the first packet contained in each file and by the IP addresses and protocol of each packet therein. The two indexes enable fast queries of the payload content for each flow.

The packets in the trace are aggregated into bi-directional flows and matched against known protocol signatures. For each flow, a number of statistics are

[1] In this paper, web-browsing refers to services using a web interface: including web sites and web-based applications such as gmail or youtube.

collected. This information along with the signature-matching results is stored in the *Flows* table. Based on this table, two further tables are created: namely the *Hosts* table and *HostPorts* table, to support aggregated views of the traffic grouped by server or client IPs. These views enable the user to browse the general behavior on a higher aggregation and also to verify the traffic at this level.

3.2 The Verification Frontend

This second layer consists of a frontend that includes a graphical interface which presents abundant information at various levels of aggregation, and supports the use of different kinds of heuristic rules, all to facilitate the verification process.

Combining the merits of many traffic classification works, the information presented in the frontend is collected from a broad set of sources, including:

- Flow statistics (e.g., duration, number of packets, total number of bytes and payload bytes in each direction and TCP flags) as in [2, 6].
- Payload information from a fine-tuned set of protocol signatures as in [1].
- Host name resolution for all the IP addresses appearing in the trace as in [3].[2]
- Port-based mapping using a comprehensive list of well known port numbers.
- Packets payload content (e.g., tcpdump of a flow).
- Host-level connection statistics and transport-layer behavior of the P2P overlay networks as in [7, 3].

Additionally, further information may be available as an extension, such as data mined from the payload of flows, flow-behavior features as used in [6], or from external modules (e.g., IDSes, specific traffic analyzers, traffic classifiers).

3.3 Heuristic Rules

The verification frontend also supports the use of heuristic rules. The main idea is to leverage a core set of automated procedures to verify subsets of similar flows with very high confidence, while resorting to human discernment when not enough clues are available to recognize the nature of certain flows.

The heuristics can either be derived empirically or built using a combination of signature matching results and *a priori* knowledge about known applications, port numbers and host names. The user can flexibly build his own heuristics, blending his own site and application-specific knowledge to facilitate desired tasks. To validate the heuristics, a specific dry-run mode is available for previewing the results of an action before actually modifying the database. On applying heuristic rules, GTVS will search for potential candidate flows and verify those which satisfy the conditions given in the heuristics.

In our experience, the use of heuristic rules has allowed us to drastically reduce the time needed to verify the ground truth.

[2] Ideally, the IP addresses should be resolved at the time when the trace is collected. However, for previously collected traces, host names can be mined from the DNS traffic in the original trace as well as the Host header field in the HTTP requests, or, in the worst case, resolved when the trace is being verified.

4 Accelerating the Verification: Experiences with GTVS

The use of GTVS does not replace the manual verification process but is dedicated to accelerating it. Here we suggest two principles, namely those of efficiency and accuracy which we apply to the use of GTVS. The efficiency principle is to always try to work upon higher aggregations (e.g., services rather than individual flows) whenever possible. For example, large numbers of well-known, traditional service traffics on a specific host can be verified in the first instance. The accuracy principle is to make decisions only with very high confidence, e.g., when strong evidence from two or more mutually-independent information sources match with each other.

Normally, the hand verification of an hour-long trace on a 1 Gigabit/s link would take more than a hundred man-hours[3]. With GTVS, we hope an experienced user would be able to verify an initial data trace within days.

In this section, we use the case study for a 30 min trace as an example to introduce the heuristic rules and show how they are exploited to accelerate the verification process. The trace was collected in mid December 2007 from the link to the Internet of a research facility. There were several thousands of users on site, mainly researchers, administrators and technical staff. Table 1 lists the working dimensions of our data set.

Table 1. Working dimensions of our data set

Distinct IPs	Server IPs	Server IP:port pairs	Client IPs	Flows	Packets	Bytes
25,631	11,517	12,198	14,474	250,403	10.9 M	7.4 GB

Because of page limit, we focus on describing how we verified the complete TCP flows, i.e., the flows that are captured entirely from triple handshake to tear down. As for the rest, the UDP flows are verified in a much similar way, except that they are defined using a configurable timeout value. The incomplete TCP flows are typically composed of various kinds of scans and unsuccessful connection attempts. Most of this traffic has distinguishable patterns upon which custom heuristic rules can be built up.

4.1 Initial Application Clues

A set of payload signatures is used in GTVS to help collect the clue of an application from packet payload. Our signature set is capable of identifying 35 most popular protocols. These signatures are derived from databases available on the Web, (e.g., l7-filter[4]). We tested the signatures on previously hand-classified data and several segments of new data. The underspecified signatures which create many false positives (e.g., eDonkey, Skype) have either been changed or

[3] An indication from the authors' previous experiences in hand-classification [1].

[4] `http://l7-filter.sourceforge.net/`

excluded, while the undermatching ones (e.g., BitTorrent) have been improved. Of course, the signatures are still far from being able to identify the totality of the traffic. However, they can be regarded as a useful clue, especially when the results they provide can be co-validated with different evidence.

4.2 The Verification Process (in Iterations)

Our approach is based on a number of successive iterations, each refining the result. Each successive verification iteration focuses upon the remaining unclassified flows about which we have the most confidence. In this way, we can accumulate knowledge based on the highest level of confidence and use this to assist in the classification of data about which we have lower levels of confidence.

Our approach requires the grouping of the heuristics to each iteration and then ordering of the iterations based upon the level of confidence we are able to place in the classification outcome.

We have derived a set of heuristics of which a core subset is presented here. We consider this subset contains those heuristics that provide sufficient generality to be useful for a wide range of applications across different physical sites.

First iteration. Based on the assumption introduced and justified in Section 2, we derive some simple heuristics below.

If the signature matching results for a specific server:port endpoint appear to be strongly consistent, we can reasonably assume that we have identified a particular service on that endpoint. Several criteria are used to quantitatively justify the consistency: thresholds are specified to guarantee that at least a certain percentage of flows as well as a minimum number of flows have matched a specific signature. In addition, only a given subset of signatures is allowed to have matched the flows. For example, it is known that HTTP might appear in BitTorrent[5] traffic, but BitTorrent should not appear in the flows toward a Web server. This constraint is expressed by defining a subset of possible signatures (which does not include BitTorrent when the heuristic is used for HTTP traffic). The thresholds are initially set in a conservative way (e.g., at least 90% and 10 flows), and will be tuned in the third iteration. We apply this heuristic for most of the protocols, especially for those with a well-established signature.

The next heuristics are based on the assumption that flows between the same IP addresses pair are likely due to the same application. For example, FTP traffic between two hosts can be easily verified by considering a host that has an already verified FTP service (e.g., using the first heuristic) and a number of flows each going to a different high port number on the same destination in an incremental fashion. As another example, consider the HTTPS protocol. In many cases a web server is running both standard and secure HTTP services. If a standard HTTP service has been verified on port 80, and a certain fraction of flows to port 443 matches the SSL signature, then the flows between a hosts pair can be heuristically verified. Other similar heuristics can be derived for streaming and

[5] BitTorrent clients use HTTP in some of their communications.

Table 2. Traffic breakdown by class and evolution of completeness by iteration

Class	Total			Number of flows by iterations				
	Flows	Packets	Bytes [MB]	1st	2nd	3rd	4th	5th
email	10,871	808,272	470.55	7,225	8,743	9,439	10,420	10,871
ftp	555	894,805	838.22	555	555	555	555	555
gaming	150	2,882	0.47	0	108	108	108	150
im	506	21,036	4.18	65	503	505	505	506
malicious	4,008	62,259	5.30	0	0	0	0	4,008
p2p	17,851	1,125,766	685.43	12,046	12,046	12,728	17,708	17,851
remote	317	135,735	109.26	254	254	254	254	317
services	618	16,675	9.22	466	604	610	610	618
streaming	11	17,815	16.33	0	2	8	8	11
voip	1,043	52,020	11.92	0	121	121	1,042	1,043
web-browsing	212,432	7,630,649	4,889.80	207,888	208,313	211,522	211,522	212,432
unknown	2,041	112,840	52.66	21,904	19,154	14,553	7,671	2,041

VoIP applications: for example, RTSP appear within a TCP control channel while the data are relayed on a unidirectional UDP stream; instead a VoIP application may use a SIP session and a bi-directional UDP stream.

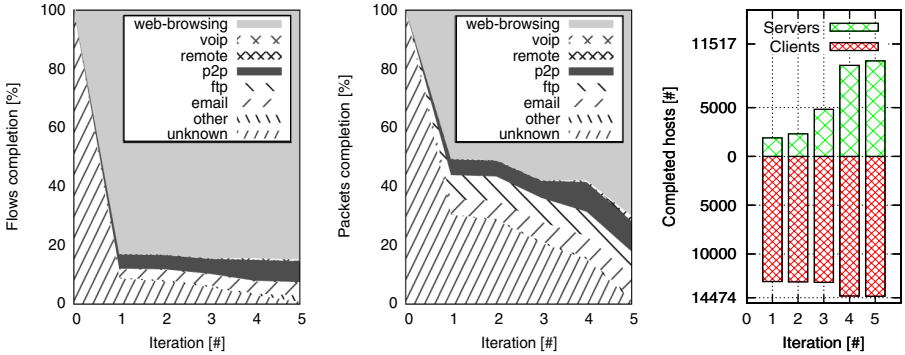
Second iteration. A great amount of information can be derived from the host names. For very popular services (e.g., Google, MSN, eBay), heuristics based on domain names can be easily defined: e.g., HTTPS traffic to MSN servers is due to MSN messenger instead of browsers as well as traffic on port 1863. Further, assuming the trace was captured at the edge of a certain network, specific site information about the internal services and traffic policies can be used to efficiently verify a part of the traffic.

Third iteration. Now we try to lower the thresholds of the previous heuristics. We focus on particular hosts where certain flows are matched by a specific signature, while a part of flows are not matched. If also these flows correspond to the same application, the thresholds can be lowered for those hosts.

Fourth iteration. In this iteration we consider behavioral characteristics of hosts in regard to overlay networks, mainly for the identification of P2P traffic. A typical example, however, is the SMTP traffic which has a strong P2P-like behavior in that SMTP servers act as both the recipient and the sender of emails. The assumption is that if a host is an SMTP server, all the flows generated from this host toward port 25 are mail traffic. In general, this heuristic is applicable for P2P traffic as long as the information about the port number can be utilized[6] and the assumption of the heuristic can be validated. In our experience, for example, there is a large number of eDonkey flows which can be identified using port 4662 and for BitTorrent on port 6881. This heuristic can re-iterate through the data set until no new flows are discovered.

Additionally, for P2P applications that use dynamic port numbers, we resort to a heuristic that considers the host activities and their relationship with a

[6] We observe that many P2P nodes still use the well-known port numbers.



(a) Breakdown of flows (b) Breakdown of packets (c) Breakdown of hosts

Fig. 2. Verification completeness against successive iterations

certain overlay network. We select an initial set of peers which are known to run a particular P2P application: some P2P nodes are already identified in the first iteration, and some Skype clients are identified in the second iteration using the information of login servers.[7] Then, for each identified peer, we consider the set of hosts that it communicates with. We select the subset of hosts corresponding to the intersection of all these host sets. Lastly, we identify hosts that are likely peers by applying a threshold (e.g., ≥ 3) on the host's connection degree (i.e., how many peers are connected to this host) and selecting those hosts that do not have other conflicting activities (e.g., if they run multiple P2P applications).

Fifth iteration. From our experience, at this iteration only a small number of payloaded flows remain. User-defined heuristics can be derived according to the specific applications in the analyzed trace, or to the particular behaviors that might be found through manual inspection. Also, based on information of the already verified hosts, one can start to label the incomplete TCP flows and unknown UDP flows, using the assumption on the sub-tuple consistency.

So far, the heuristic rules have greatly reduced the time to verify the flows, although manual verification of a small amount of remaining traffic is still necessary, especially for the identification of new applications.

Table 2 summarizes the total traffic breakdown as we verified, and shows the partial results measured at the end of each iteration, which are graphically presented in Figures 2a and 2b for two metrics: flows and packets. Finally, Figure 2c reports the evolution of completeness for clients and servers during each iteration. As can be seen, a very small number of clients are responsible for the 2,041 unknown flows toward 1,736 servers. In this case, these hosts happen to simultaneously run several P2P applications and we are not able to determine a final conclusion on the specific application.

[7] For identifying Skype clients we also use another heuristic based on the peculiarity of this application receiving TCP connection on ports 80 and 443 plus a high number chosen at random.

Table 3. Evaluation of l7-filter's per-class accuracy

Class	False negatives [%]			False positives [%]		
	Flows	Packets	Bytes	Flows	Packets	Bytes
email	25.99	22.33	21.75	0.00	0.00	0.00
ftp	81.26	99.53	99.97	0.00	0.00	0.00
gaming	100.00	100.00	100.00	0.00	0.00	0.00
im	74.90	79.60	81.78	0.00	0.00	0.00
malicious	100.00	100.00	100.00	0.00	0.00	0.00
p2p	16.75	16.65	14.67	0.34	1.85	2.10
remote	18.93	0.52	0.04	0.00	0.00	0.00
services	98.54	99.48	99.94	0.00	0.00	0.00
streaming	27.27	0.15	0.02	0.00	0.00	0.00
voip	100.00	100.00	100.00	0.00	0.00	0.00
web-browsing	0.29	0.42	0.40	0.42	0.52	0.27

Finally, we evaluate l7-filter's accuracy based on the obtained ground truth. Table 3 shows per-class false negatives and positives. Its signatures do not significantly over match, yielding to very few false positives. However, with the exception of web-browsing class, all classes exhibit many false negatives. This is due to two major factors: underspecified signatures and obfuscated traffic. In both cases, our method can exploit information about traffic aggregates to derive the actual application and produce accurate ground truth.

4.3 Discussion

Here we have focused on describing the verification of application traffic. The verification processes of malicious and unwanted traffic (left out due to page limit) are also in progressive development, based on their specific patterns.

One can see that the first-time use of GTVS on any given trace will often require inspection of small segments of data throughout the process, in customizing and testing new heuristics, dry-runs, tuning thresholds, and final manual decisions on hard objects. However, if one is carrying out a continuous ground truth collection work on a specific site or on many sites simultaneously, time would be further saved as we expect only limited tuning and validation are needed.

Since this framework will become publicly available, it is also easier to share the knowledge within the community: not only the string signatures but also the heuristics and application-specific knowledge would become a public resource and can be constructed and validated by any user of this framework.

We also note that the confidence of the ground truth verified by GTVS relies mainly on its user. Therefore to collect good ground truth requires sufficient user interactions and dry-runs to double-confirm the user's judgments.

5 Related Work

On the technical aspects, our work can be seen as a cumulative progress, with lots of inspirations from previous traffic classification works, including [2, 3, 6, 7, 8, 9].

Each of these works made use of a different set of information sources, which are combined in our framework.

A content-based classification scheme comprising of nine identification methods was presented in [1]. Despite their highly accurate and complete results, there was not a systematic infrastructure or an indication of how the procedure can be organized. Thus a barrier exists preventing other people from repeating their method. Further, GTVS uses a broader set of information sources.

In [10], the authors suggested a technique based on active measurements to cover the shortage of ground-truth data. This work is tackling a similar problem to ours. However, we argue that this technique is incapable of delivering the variety and fidelity of real traffic. In contrast, we focus on maximally reducing the time and labor necessary to obtain accurate ground truth from real traffic.

6 Conclusions

In this paper, we presented the novel Ground Truth Verification System (GTVS). A detailed guide is shown on how to use GTVS to accelerate the verification process, as well as the results by iterations from a case study of real traffic. Further, we are publicly releasing this system and our rule sets. It is hoped that it will substantially save the time and labor for individual researchers, and more public data with ground-truth labels may subsequently become available to the community in the near future.

References

1. Moore, A.W., Papagiannaki, D.: Toward the accurate identification of network applications. In: Dovrolis, C. (ed.) PAM 2005. LNCS, vol. 3431, pp. 41–54. Springer, Heidelberg (2005)
2. Moore, A.W., Zuev, D.: Internet traffic classification using bayesian analysis techniques. In: Proceedings of ACM SIGMETRICS 2005, pp. 50–60 (2005)
3. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: Blinc: multilevel traffic classification in the dark. In: Proceedings of ACM SIGCOMM 2005, pp. 229–240 (2005)
4. Erman, J., et al.: Traffic classification using clustering algorithms. In: Proceedings of the SIGCOMM workshop on mining network data, MineNet 2006 (2006)
5. Dusi, M., et al.: Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting. Computer Networks 53(1), 81–97 (2009)
6. Li, W., Moore, A.W.: A machine learning approach for efficient traffic classification. In: Proceedings of IEEE MASCOTS 2007 (October 2007)
7. Karagiannis, T., Broido, A., Faloutsos, M., Claffy, K.: Transport layer identification of P2P traffic. In: Proceedings of Internet Measurement Conference (2004)
8. Trestian, I., Ranjan, S., Kuzmanovi, A., Nucci, A.: Unconstrained endpoint profiling (googling the internet). In: Proceedings of ACM SIGCOMM 2008, pp. 279–290 (2008)
9. Dreger, H., et al.: Dynamic application-layer protocol analysis for network intrusion detection. In: 15th USENIX Security Symposium (2006)
10. Szabó, G., et al.: On the validation of traffic classification algorithms. In: Claypool, M., Uhlig, S. (eds.) PAM 2008. LNCS, vol. 4979, pp. 72–81. Springer, Heidelberg (2008)