Traffic Extraction and Classification in Network Forensics

Blind Copy and Blind Copy

Blind Copy Blind Copy, Blind Copy Blind Copy, Blind Copy
Blind Copy@Blind Copy.Blind Copy
BlindCopyBlindCopy
https://pluskal.github.io/AppIdent/

Abstract. Network traffic classification is essential for network monitoring, security analysis and also digital forensics. Accurate classification can reduce the amount of information that needs to be analyzed during the investigation. In this paper, we present a study that compares three different algorithms that according to the literature offer high accuracy and acceptable performance. These algorithms are evaluated on their ability to identify traffic classes at application protocol and also network application software levels. Based on experiments, Random forest algorithm gives promising results.

Key words: network forensics, network traffic classification, statistical protocol identification

1 Introduction

Network traffic classification is a useful technique for network monitoring, security analysis, and digital forensics. For digital forensics, correct classification of network traffic significantly reduces the volume of data that needs to be inspected manually.

In digital forensics, file types can be identified by the extension or by examining the beginning of the file searching for so-called magic numbers. Also, databases of hash values are used to identify known files. Identification of file types and filtering known files helps to reduce an amount of data that needs to be analyzed. Doing the same with network traffic is, however, more complicated as each data transfer contains specific and temporary characteristics. Traditional traffic classification identifies applications based on the used TCP or UDP ports. Port-based classification is of limited accuracy because many application use random or non standard ports. Advanced traffic classification employs payload analysis, statistical methods, machine learning algorithms, and hybrid approaches. Each of these techniques has advantages and drawbacks. For instance, payload-based analysis accuracy for encrypted communication is unacceptably low. Machine learning is the promising approach offering several benefits. Methods does not need to rely on the content of packets. It is possible to combine structural and behavioral features to increase detection accuracy. Unsupervised methods can identify unknown network traffic.

The ML approach to traffic classification has been widely studied by many researchers. Most of the works present methods that aim to classify the network traffic according to service protocol used. In this paper, we present results of experiments with the use of machine learning algorithms adapted to identify network applications. Network application identification gives more information about the network traffic comparing to what can be understood from identified application protocol. The HTTPS is often used for communication between clients and servers. For instance, Google Drive, iTunes, and OneDrive use HTTPS for data transfer. Providing information on recognized application instead of the protocol can be useful for various application domains.

1.1 Current State

Machine learning algorithms for network traffic classification have been studied since 1990's. The most utilized algorithms comprise of support vector machine [1], decision tree algorithms [2], and probabilistic [3] or statistical methods [4, 5] which are instances of supervised learning methods. In the case of unsupervised learning the clustering approach, namely K-means algorithm [6], enables to group the traffic by their significant properties. If the feature set is properly selected, then ML methods can exceed 90% accuracy [7].

A survey of classification methods based on ML can be found in [8] and another that includes more recent results in [7]. Methods of classification of encrypted traffic were reviewed in [9]. Most of the existing literature deals with traffic classification as a tool for intelligent network filtering or security monitoring. While the traffic classification for network forensics stems from the same ideas there are certain differences:

– Network forensics can be performed as an off-line analysis of captured data. In this case, the accuracy is more important than performance. Thus a combination of several methods or application of slower but more accurate methods can be considered.
– The investigator can deal and compensate incorrect outcomes by performing the additional manual inspection of results.
– Classification is expected to be deterministic as the forensic principles require that all results are verifiable.
– Classification methods can be tuned by the investigator and classification can be repeated with different parameters sets.

While traffic classification is under intensive research for applications in network monitoring or security analysis, significantly less research was done on traffic discrimination for network forensics. Recently, Al Khater and Overill [10] have proposed to investigate the use of different machine learning algorithms to improve traffic classification methods for digital forensics. Foroushani and Nur Zincir-Heywood [11] demonstrated the possibility to identify high-level application behaviors from encrypted communication of several network services.

Dai [12] and Miskovic et al. [13] described a method for fingerprinting mobile applications from their communication. Erman et al. [14] explored flow-based classification approach and proposed a semi-supervised classification method that can accommodate both known and unknown applications.

1.2 Contribution

In this paper, we overview the results of experiments with three feature-based classification algorithms for separation of the normal traffic to unknown (possible suspect) traffic. Based on the results available from the literature, we selected two machine learning algorithms, in particular, Bayesian Network Classifier and Random forests and compared them to a statistical protocol fingerprinting method. The contribution of the paper aims at showing how classification algorithms can provide valuable information for digital investigation by identifying application protocols and in some cases also communicating applications.

1.3 Paper Organization

The paper is organized as follows: Section 2 shortly describes how traffic data is collected and prepared for classification. Issues related to correct identification of conversations and application messages are discussed. Section 3 presents proposed features and the method for elimination statistically dependent features. Three classification methods are then briefly outlined. Section 4 details the experimental environment, data sources and reviews the results. Paper is concluded in section 5 by summing the results and suggesting directions for future work.

2 Data Collection and Preprocessing

The input to classification is a collection of conversations that are composed of packets read from PCAP files. A conversation is defined by a 5-tuple consisting of the IP address of the pair of hosts, the (transport) protocol type and the port numbers used by the two hosts. For all conversations, we apply validation steps to ensure that the available information is correct. Also, we attempt to identify application messages to improve the accuracy of the classification.

2.1 Data Preparation and Cleaning

Classifying communication at the conversation level usually provides more information to the classifier. While the reassembling procedure is quite straightforward, some subtle issues can negatively affect the classification results:

- For TCP session some important control information may be missing, e.g., synchronization segments or finalization segments.
- For long running TCP conversation, the sequence number overflow problem may arise.

– Capture file may contain incorrect or inaccurate frame timestamps.
– Some TCP segments may be duplicated or missing.

Ignoring some of these issues may lead to missing conversations or inaccurate traffic classification [15].

2.2 Application Message Segmentation

The segmentation provides for each conversation a group of application messages. While this represents an additional processing step, it can help to increase the accuracy of classification by identification application communication patterns. It also eliminates remnants of network packet fragmentation at the Internet layer and TCP retransmission at the transport layer. Packet fragmentation and TCP retransmission are independent of application communication patterns and thus can negatively affect the classification.

An application message is identified in the reassembled stream depending on the transport protocol by the following rules:

– If a stream uses UDP transport protocol, then the entire payload of each UDP datagram is considered as a single application message.
– For TCP transport protocol, messages are separated by segments that contain PSH, RST, or FIN flags.

The rules are simple to implement, but for most of the applications, they give an accurate approximation of the applications message identification.

2.3 Traffic Class Levels

In general, traffic classification can be with different granularity. According to Khalife et al. [16] traffic classes range from coarse-grained to fine-grained levels as follows:

1. Application type level represents a group of traffic of the same kind, e.g., game, browsing or VoIP.
2. Application protocol level identifies a concrete protocol, e.g., HTTP, SMTP, or FTP;
3. Application software level determines a specific client or server program related to the traffic, e.g., HTTP-Firefox, HTTPs-Chrome.
4. Message level detects a specific message exchange within application communication, e.g., e-mail delivery, Google search, or Skype call.

Fine-grained level classification represents the most valuable information. However, it is also the most difficult to get this classification accurate. Our study focuses on classification at application protocol and application software levels.

3 Classification Methods

Based on the literature survey, we identified three different classification methods that give promising results for traffic identification. As we aim at identifying applications at conversation level, we have revised the usually utilized feature sets [17, 5] to include additional information that characterizes conversations and application message abstractions, e.g., message timing vs. packet timing, response patterns, etc. In this section, we describe the selected feature set, the elimination of redundant features, and briefly recall the principles of considered classification algorithms.

3.1 Feature Set

The quality of a set of features influences the classification accuracy [18]. Commonly used features of traffic classification are related to significant aspects of packet communication and network architecture. Port numbers, transport protocol type, starting the sequence of payload bytes, pattern occurrence, packet length, and packet timing are often used. We identified a list of possible features. The proposed list consisting of 92 items can be found in the reference implementation [1]. To develop the optimal set of features for the classifier, we analyzed the original list and selected statistically independent features. Table 1 enumerates a reduced collection of features that were finally used in the experiments. Most of the features describe the characteristics of flows rather than individual packets. Not relying on the signature or some specific patterns in the content of the packets gives better results for encrypted or less structured traffic. Also, statistical representation of the application traffic can help to discriminate between different applications utilizing the same application protocol.

3.2 Correlated Features Elimination

Machine learning methods are very sensitive to the feature selection. Best performance is obtained when features are orthogonal, i.e., when there is no correlation among features [19]. There are several approaches how to calculate a correlation between features, e.g., Pearson, Spearman, Kendall correlation formulas [20] or covariance matrix [21]. The covariance matrix shows a correlation value between the pair of features.

Using correlation matrix, we can automate feature selection process to increase accuracy and eliminate features which demonstrate a significant correlation. This process is parameterized by maximally allowed correlation that is acceptable. By our measurements, more than 80% of the features that we originally proposed showed 0.5 and higher correlation. Such features were eliminated. In Table 1, the first column shows the correlation value for a group of features.

[1] https://github.com/pluskal/AppIdent/blob/gh-pages/Pages/allFeatures.pdf

Correlation	Feature for 0.1 t/v ratio	Feature for 0.2 t/v ratio
	BytePairsReoccuringDownFlow	
	DirectionChanges	
	First3BytesEqualDownFlow	First3BytesEqualDownFlow
	FirstBitPositionUpFlow	FirstBitPositionUpFlow
	FirstPayloadSize	
	MinInterArrivalTimeDownFlow	
	MinInterArrivalTimePacketsUpAnd DownFlow	MinInterArrivalTimePacketsUpAnd DownFlow
	MinPacketLengthDownFlow	MinPacketLengthDownFlow
	NumberOfBytesDownFlow	
	NumberOfPacketsUpFlow	
	PacketLengthDistributionDownFlow	PacketLengthDistributionDownFlow
	PacketLengthDistributionUpFlow	
		ThirdQuartileInterArrivalTimeUp
		ByteFrequencyUpFlow
		MaxSegmentSizeDown
		MaxSegmentSizeUp
		MinInterArrivalTimePacketsUpFlow
		NumberOfBytesUpFlow
		ThirdQuartileInterArrivalTimeDown
<0.25	PUSHPacketsDown	PUSHPacketsDown
	ThirdQuartileInterArrivalTimeDown	
		NumberOfBytesUpFlow
<0.3		FirstPayloadSize
	ByteFrequencyUpFlow	
	MinPacketLengthUpFlow	MinPacketLengthUpFlow
	NumberOfPacketsPerTimeUp	
		DirectionChanges
		BytePairsReoccuringDownFlow
<0.4		MeanPacketLengthUpFlow
<0.5	MeanPacketLengthUpFlow	

Table 1: Enumeration of features remained after feature elimination on sample data with 0.1 and 0.2 training to verification ratios (t/v). These feature sets are used for classification with Bayesian and Random forests classifiers.

3.3 Statistical Protocol Identification

Statistical Protocol Identification (SPID) was originally developed by Erik Hjemvik to be used with NetworkMiner tool[2]. The method performs application protocol classification based on a database of protocol fingerprints. Protocol fingerprints are computed in the learning phase. A feature set utilized by SPID is called protocol attribute meters, and each item may represent a different kind of information. Some items can be scalar values representing, e.g., payload data size, the number of packets in the session, or port number. Other items can be composite values, for instance, a tuple consisting of packet direction, packet order, packet size, byte value frequency.

The SPID classifier computes the distance of the attribute to the actual data of the analyzed conversation using Kullback-Leibler divergence. The best matching protocol model is the one that has the smallest sum of KL-divergences of all attributes. Kohnen et al. [5] developed a new version of the SPID algorithm by incorporating other attributes to increase the accuracy of this method. The method, utilized in our experiments referred as ESPI, is a version of Kohnen's

[2] http://www.netresec.com/?page=NetworkMiner

SPID. Comparing to Kohen's SPID methods, we use a different set of 92 features and the algorithm for computing similarity of measured values to protocol fingerprint.

Each feature is associated with function f to compute the divergence of a measured value to the model values, the function g to compute the normalized feature value from the actual measured value, the function w to define a weight of the feature for the protocol fingerprint. Similarity is based on the computation of Euclidian distance [22] of weighted divergences for individual features. The classification stands for computing a distance of the actual flow represented by attribute values $x_1, ..., x_n$ to protocol fingerprint c, see Equation 1.

$$d_{x,c} = \sqrt{\sum_{i=0}^{n} \left(w \cdot f_i \left(g(x_i) - c_i\right)\right)^2} \tag{1}$$

A difference d_{x,c^j} for each protocol fingerprint c^j is computed. The class of the flow corresponds to the protocol k such that $d_{x,c^k} = min(d_{x,c^1} \dots d_{x,c^m})$.

In comparison with ML methods, ESPI does not suffer overfitting related to the use of correlated features because ESPI assigns weights on per feature basis. This property makes ESPI easily extendable for classification of new protocols and introduction of new features that are unique for those protocols but may be correlated with other features for other protocols.

3.4 Bayesian Network Classifier

The algorithm that powers the Bayesian network classifier [23] stems from the Bayes' theorem, which defines the probability of an event considering conditions related to the occurrence of the event. The classifier is composed of Bayesian belief networks that are built during the learning phase. Recall that Bayesian network is a directed acyclic graph and a set of conditional probability tables. Nodes represent feature variables, and edges represent conditional dependencies. Probability tables provide probability functions for nodes. The application protocol is identified by finding a node (or a set of nodes) that has for the given input feature values the highest probability. The advantage of this classifier is that it also computes the probability of which the conversation belongs to the identified class. This information may be useful for the examiner to decide whether to analyze the traffic.

3.5 Random Forests Classifier

Random forest is, in a context of this paper, an ensemble method constructing multiple C4.5 decision trees in *training phase*, used for classification in *verification phase* where a mode of partial results is selected as the resultant class [24]. This makes Random forests prone to overfitting [25]. Random forests are parametrized by multiple variables, e.g., forest count, join, and training to verification ratio. Optimal values used for these parameters can be found by a cross-validation and computation of an out-of-bag-error (OOB) to estimate the performance of particular parameter combination. Because the algorithm computes OOB error, there is no need to have separate verification data. Therefore, the algorithm can be trained on the whole data set. Although, this approach was proven to be computationally very expensive.

4 Experiments And Results

A set of experiments was conducted to evaluate properties of three classification methods. An annotated data set was prepared using Microsoft Network Monitor which provides application labels for most of the conversations. The data set comprise of a regular network traffic of eight user stations running Windows operating system. The captured traffic has the following characteristics[3]:

- PCAP file size: $19.5GB$
- PCAP Format: *Microsoft NetMon 2.x*
- Capture duration: $119h$
- Number of packets: 27616138
- Number of L7 conversations: 269459

The PCAP file was processed using an engine of Netfox Detective [15]. This engine tracked application conversations (L7), reassembled transport protocol streams to create an approximation of application messages, and served as a middleware to access captured data. Features were extracted and stored as a separate binary file. ESPID method was implemented from scratch. Random Forest and Bayesian network classifier were implemented using Accord.NET library[4] of machine learning algorithms.

4.1 The Scheme of Experiments

Experiments with all classification algorithms followed the same procedure. This procedure consists of the following steps:

1. The data source was split into two disjoint data sets according to the particular case. One data set was used for learning and the other was used for evaluation.
2. All feature vectors were annotated with label depending on the level of classification. The label was represented as:
   a) a tuple containing a *transport protocol type*, *destination transport layer port* or *manually assigned label*, and *application process info*, for *identification of applications*, e.g., *tcp_http_skypeexe*.
   b) a tuple containing an destination *transport protocol type*, *transport layer port* or *manually assigned label*, for *identification of application protocols*, e.g., *tcp_http*.

[3] https://github.com/pluskal/AppIdent/blob/gh-pages/Pages/captureStats.pdf
[4] http://accord-framework.net/

3. [Training phase] Classifier was created and trained using the learning data set:
   a) *Enhance Statistical Protocol Identification* – For each group, an application protocol model was computed using the g function.
   b) *Bayesian network classifier* – For each group, a Bayesian network classifier was trained.
   c) *Random forests* – Optimal parameters leading to the most accurate model were computed. A cross-validation phase was used to determine the best model.
4. [Verification phase] Created classifiers were used to identify a class label for each conversation from the validation data set:
   a) Multi-label results
      i. Enhance Statistical Protocol Identification - Each feature vector symbolizing an L7 verification conversation was compared to each application protocol model using a Euclidean distance.
      ii. Bayesian network classifier – Each Bayesian classifier yielded a probability of which an L7 verification conversation belonged to the class represented by that classifier.
      iii. The classification resulted in a set of distances or probabilities. The set was ordered, and the protocol fingerprint with the smallest distance or the highest probability was selected as a label.
   b) Single-label results
      i. Random forests – Classifier yielded a resulting label.
5. The label was compared to the annotation, and statistical properties of the classification were computed.

4.2 The Summary of Results

The first goal was to compare results yielded by machine learning and statistical methods that are based on completely different bases but share the same feature set. The second goal was to observe a dependency between a size of the training set and a feature elimination ratio to find the best-suited combination for classification of application protocols and network applications. The third goal was to prove that using application classifiers we are also able to classify the traffic with respect to network applications.

Figure 1 and Figure 2 summarize best results of classification. Only a top 20 most accurately identified classes were selected for presentation in this paper. Other classes are presented as aggregated statistics in Table 3[5]. Results achieved by the different methods are evaluated using the F-Measure, which is also known as the balanced F-score [19]. This single score stands for the harmonic mean of precision and recall, and is computed by Equation 2.

[5] The complete results for protocol classification can be found at https://github.com/pluskal/AppIdent/blob/gh-pages/Pages/comparisonPA.pdf or for *application identification* at https://github.com/pluskal/AppIdent/blob/gh-pages/Pages/comparisonPAAppTags.pdf

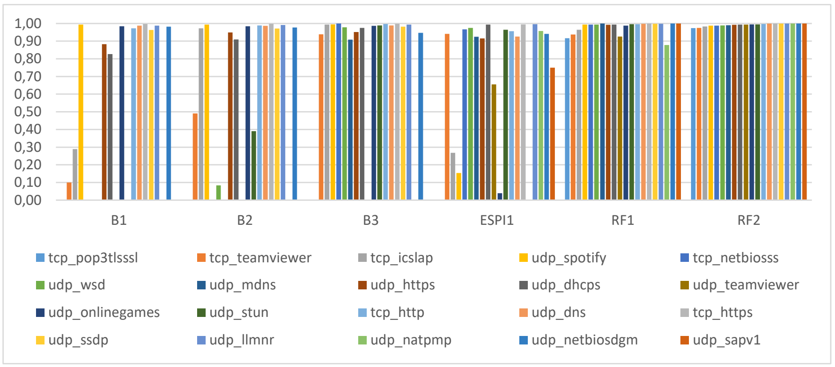
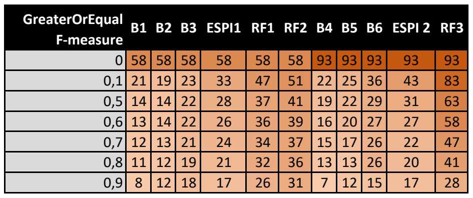


Fig. 1: Performance of application protocol classifiers using F-measure. Abbreviations explained in Table 2.

Method	Abbreviation	Training to verification ration	Feature selection ratio
Bayesian classifier	B1	0.1	0.3
	B2	0.2	0.5
	B3	0.5	0.5
	B4	0.1	0.2
	B5	0.2	0.25
	B6	0.5	0.25
ESPI	ESPI 1	0.7	1
	ESPI 2	0.2	1
Random forests	RF1	0.1	0.4
	RF2	0.2	0.4
	RF3	0.2	0.5

Table 2: Configuration of classification methods.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2}$$



GreaterOrEqual F-measure	B1	B2	B3	ESPI1	RF1	RF2	B4	B5	B6	ESPI 2	RF3
0	58	58	58	58	58	58	93	93	93	93	93
0,1	21	19	23	33	47	51	22	25	36	43	83
0,5	14	14	22	28	37	41	19	22	29	31	63
0,6	13	14	22	26	36	39	16	20	27	27	58
0,7	12	13	21	24	34	37	15	17	26	22	47
0,8	11	12	19	21	32	36	13	13	26	20	41
0,9	8	12	18	17	26	31	7	12	15	17	28

Table 3: Summary of classification methods' performance. Numbers show, how many classes were classified with equal or greater F-measure.

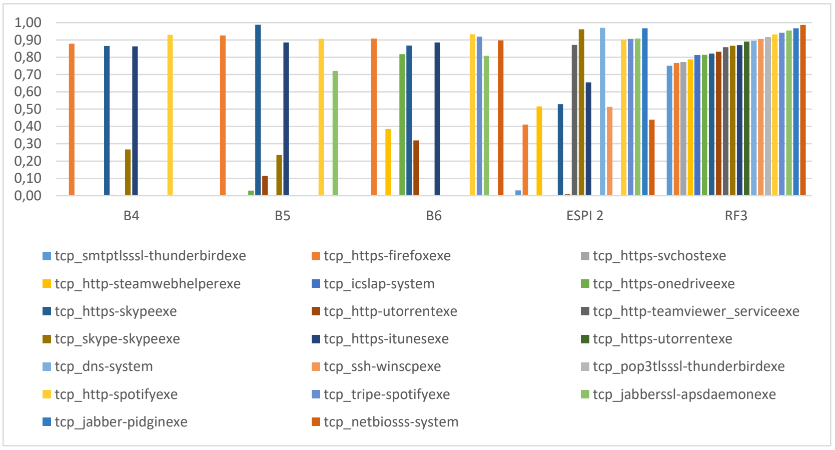


Fig. 2: Performance of application classifiers using F-measure. Abbreviations explained in Table 2.

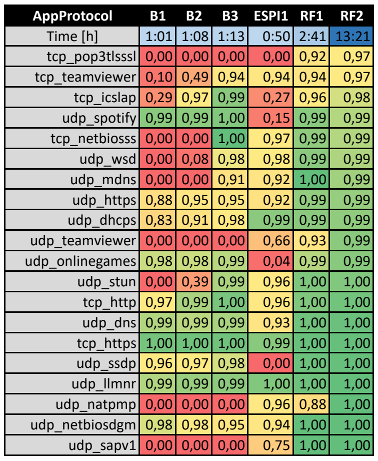
Figure 1 visualizes the identification of application protocols. We can observe that both Random forest classifiers (RF1, RF2) are very accurate. Bayesian classifier (B3) also performs very well but it requires a larger training set with ratio 0.5 and more features, see Table 2. The time needed for various steps, such as feature selection, data loading, training, and classification, was also measured and is presented in Figure 3 along with a performance comparison of each evaluated method.

Figure 2 provides a visual representation of results for the classification of applications. It can be seen that Random Forest classifier again provides the best results. In this case, Bayesian classifier was outperformed by ESPI which also shown the best trade-off between performance and accuracy.

To resolve an issue with an application protocol ambiguity when classifying traffic that cannot be classified on network applications level, i.e., *tcp_http_skypeexe* is classified a *tcp_http_firefoxexe*, and vice-versa. We propose to apply a hierarchical clustering based on ESPI protocol fingerprints. The investigator can infer the actual application protocol classes by visual cluster analysis. This method was also used to perform manual corrections in the dataset.

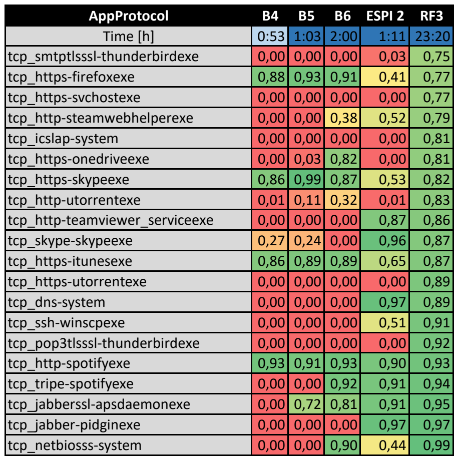
5 Conclusions

We have presented a study on the usage of protocol classifiers for separating regular traffic generated by known networking applications and unknown possibly suspect traffic. The approach described in this paper focuses on the identification of network applications in addition to the classification of application protocols. Because of this, the input to the classification contains features that characterize



AppProtocol	B1	B2	B3	ESPI1	RF1	RF2
Time [h]	1:01	1:08	1:13	0:50	2:41	13:21
tcp_pop3tlsssl	0,00	0,00	0,00	0,00	0,92	0,97
tcp_teamviewer	0,10	0,49	0,94	0,94	0,94	0,97
tcp_icslap	0,29	0,97	0,99	0,27	0,96	0,98
udp_spotify	0,99	0,99	1,00	0,15	0,99	0,99
tcp_netbiosss	0,00	0,00	1,00	0,97	0,99	0,99
udp_wsd	0,00	0,08	0,98	0,98	0,99	0,99
udp_mdns	0,00	0,00	0,91	0,92	1,00	0,99
udp_https	0,88	0,95	0,95	0,92	0,99	0,99
udp_dhcps	0,83	0,91	0,98	0,99	0,99	0,99
udp_teamviewer	0,00	0,00	0,00	0,66	0,93	0,99
udp_onlinegames	0,98	0,98	0,99	0,04	0,99	0,99
udp_stun	0,00	0,39	0,99	0,96	1,00	1,00
tcp_http	0,97	0,99	1,00	0,96	1,00	1,00
udp_dns	0,99	0,99	0,99	0,93	1,00	1,00
tcp_https	1,00	1,00	1,00	0,99	1,00	1,00
udp_ssdp	0,96	0,97	0,98	0,00	1,00	1,00
udp_llmnr	0,99	0,99	0,99	1,00	1,00	1,00
udp_natpmp	0,00	0,00	0,00	0,96	0,88	1,00
udp_netbiosdgm	0,98	0,98	0,95	0,94	1,00	1,00
udp_sapv1	0,00	0,00	0,00	0,75	1,00	1,00

(a) Application protocol classifiers



AppProtocol	B4	B5	B6	ESPI 2	RF3
Time [h]	0:53	1:03	2:00	1:11	23:20
tcp_smtptlsssl-thunderbirdexe	0,00	0,00	0,00	0,03	0,75
tcp_https-firefoxexe	0,88	0,93	0,91	0,41	0,77
tcp_https-svchostexe	0,00	0,00	0,00	0,00	0,77
tcp_http-steamwebhelperexe	0,00	0,00	0,38	0,52	0,79
tcp_icslap-system	0,00	0,00	0,00	0,00	0,81
tcp_https-onedriveexe	0,00	0,03	0,82	0,00	0,81
tcp_https-skypeexe	0,86	0,99	0,87	0,53	0,82
tcp_http-utorrentexe	0,01	0,11	0,32	0,01	0,83
tcp_http-teamviewer_serviceexe	0,00	0,00	0,00	0,87	0,86
tcp_skype-skypeexe	0,27	0,24	0,00	0,96	0,87
tcp_https-itunesexe	0,86	0,89	0,89	0,65	0,87
tcp_https-utorrentexe	0,00	0,00	0,00	0,00	0,89
tcp_dns-system	0,00	0,00	0,00	0,97	0,89
tcp_ssh-winscpexe	0,00	0,00	0,00	0,51	0,91
tcp_pop3tlsssl-thunderbirdexe	0,00	0,00	0,00	0,00	0,92
tcp_http-spotifyexe	0,93	0,91	0,93	0,90	0,93
tcp_tripe-spotifyexe	0,00	0,00	0,92	0,91	0,94
tcp_jabberssl-apsdaemonexe	0,00	0,72	0,81	0,91	0,95
tcp_jabber-pidginexe	0,00	0,00	0,00	0,97	0,97
tcp_netbiosss-system	0,00	0,00	0,90	0,44	0,99

(b) Application classifiers

Fig. 3: Comparison of top 20 classifier's performance. Abbreviations explained in Table 2.

application behavior, such as message timing, content length, TCP's PUSH flags usage, etc. The presented results suggest the following conclusions:

- The particular type of applications can be classified with high confidence. For instance, NetBIOS service or DNS were identified accurately. Also, several common applications that use HTTP(s) application protocol were identified. Similarly, it is possible to distinguish the communication traces of OneDrive, Skype, iTunes, Spotify, Steam or μTorrent clients although all of them use the same application protocol (HTTPS).
- In our experiments, Random forest classifier gives the best results. It is not surprising and it agrees with conclusions of other researchers experimenting with different ML algorithms for traffic classification, e.g. [26] or [27]. In addition to traffic classification, we focused on distinguishing communicating applications using the same protocol. In this case, Random forest method demonstrated its ability to identify most of the applications correctly.

Further research should be conducted in different directions, namely:

- Systematical analysis of feature sets to improve the accuracy and robustness of classification methods. Classification accuracy is mainly determined by the quality of selected features. In the presented work, we have proposed features based on previous observations and intuition rather than applying a systematic approach.
- Further research on hierarchical classification. Hierarchical classifiers may provide viable information on the traffic being examined. Sometimes it is not possible to accurately determine the network application, but the classifiers can identify application protocol with high confidence. Also, extending this

approach to other levels, e.g., message level, represents an added value for network forensics.
- The combination of the different classifiers [28] to increase the confidence of the results. While a single method can better fit the classification of a certain kind of application traffic, by combining several classifiers, we may get more accurate overall classification score.
- Adaptation of semi-supervised classification method [29] that enables to create models from partially labeled data. Because network forensics includes a significant amount of human labor, this can be used to guide the classifier during the model construction stage.
- Further experimentation to extend the classification models of the classifiers and evaluate the properties of other data sets. Considered classification methods depend on the accurately created models. These models require analyzing an amount of traffic that contains enough samples to construct application protocol fingerprints correctly.

Reference implementation of classification algorithms is available under the MIT license on GitHub: https://pluskal.github.io/AppIdent/.

References

1. G. Gómez Sena and P. Belzarena, “Early traffic classification using support vector machines,” in *Proceedings of the 5th International Latin American Networking Conference*, pp. 60–66, ACM, 2009.
2. Y. Luo, K. Xiang, and S. Li, “Acceleration of decision tree searching for ip traffic classification,” in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pp. 40–49, ACM, 2008.
3. E. Bursztein, “Probabilistic identification for hard to classify protocol,” in *IFIP International Workshop on Information Security Theory and Practices*, pp. 49–63, Springer, 2008.
4. E. Hjelmvik, “The spid algorithm-statistical protocol identification,” *Gävle, Sweden, October*, 2008.
5. C. Köhnen, C. Überall, F. Adamsky, V. Rakocevic, M. Rajarajan, and R. Jäger, “Enhancements to statistical protocol identification (spid) for self-organised qos in lans.,” in *ICCCN*, pp. 1–6, 2010.
6. A. Finamore, M. Mellia, and M. Meo, “Mining unclassified traffic using automatic clustering techniques,” in *Lecture Notes in Computer Science*, vol. 6613 LNCS, pp. 150–163, 2011.
7. N. Namdev, S. Agrawal, and S. Silkari, “Recent advancement in machine learning based internet traffic classification,” *Procedia Computer Science*, vol. 60, pp. 784–791, 2015.
8. T. T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.
9. P. Velan, M. Čermák, P. Čeleda, and M. Drašar, “A survey of methods for encrypted traffic classification and analysis,” *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

10. N. Al Khater and R. E. Overill, “Forensic network traffic analysis,” in *Proceedings of The Second International Conference on Digital Security and Forensics, Cape Town, South Africa*, 2015.
11. V. A. Foroushani and A. N. Zincir-Heywood, “Investigating application behavior in network traffic traces,” in *Computational Intelligence for Security and Defense Applications (CISDA), 2013 IEEE Symposium on*, pp. 72–79, IEEE, 2013.
12. S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, “NetworkProfiler: Towards automatic fingerprinting of Android apps,” *Proceedings - IEEE INFOCOM*, pp. 809–817, 2013.
13. S. Miskovic, G. M. Lee, Y. Liao, and M. Baldi, “AppPrint: Automatic fingerprinting of mobile applications in network traffic,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8995, pp. 57–69, Springer Verlag, 2015.
14. J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, “Offline/realtime traffic classification using semi-supervised learning,” *Performance Evaluation*, vol. 64, no. 9-12, pp. 1194–1213, 2007.
15. P. Matoušek, J. Pluskal, O. Ryšavý, V. Veselý, M. Kmeť, F. Karpíšek, and M. Vymlátil, “Advanced techniques for reconstruction of incomplete network data,” *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 2015, no. 157, pp. 69–84, 2015.
16. J. Khalife, A. Hajjar, and J. Diaz-Verdejo, “A multilevel taxonomy and requirements for an optimal traffic-classification model,” *International Journal of Network Management*, vol. 24, no. 2, pp. 101–120, 2014.
17. A. Moore, D. Zuev, and M. Crogan, “Discriminators for use in flow-based classification,” tech. rep., 2013.
18. L. Zhen and L. Qiong, “A new feature selection method for internet traffic classification using ml,” *Physics Procedia*, vol. 33, pp. 1338–1345, 2012.
19. J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 3rd ed., 2011.
20. I. Žežula, “On multivariate gaussian copulas,” *Journal of Statistical Planning and Inference*, vol. 139, no. 11, pp. 3942–3946, 2009.
21. I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.
22. M. M. Deza and E. Deza, “Encyclopedia of distances,” in *Encyclopedia of Distances*, pp. 1–583, Springer, 2009.
23. N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth, “Bayesian Network Classifiers,” *Machine Learning*, vol. 29, pp. 131–163, 1997.
24. L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
25. J. Friedman, T. Hastie, and R. Tibshirani, “The elements of statistical learning: Data mining, inference, and prediction,” *Springer Series in Statistics*, 2009.
26. J. Li, S. Zhang, Y. Xuan, and Y. Sun, “Identifying skype traffic by random forest,” in *2007 International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2007*, pp. 2841–2844, 2007.
27. Y. Wang and S. Z. Yu, “Machine learned real-time traffic classifiers,” in *Proceedings - 2008 2nd International Symposium on Intelligent Information Technology Application, IITA 2008*, vol. 3, pp. 449–454, 2008.
28. J. Kittler, “Combining classifiers: A theoretical framework,” *Pattern analysis and Applications*, vol. 1, no. 1, pp. 18–27, 1998.
29. J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, “Offline/realtime traffic classification using semi-supervised learning,” *Performance Evaluation*, vol. 64, no. 9, pp. 1194–1213, 2007.