# Traffic Classification and Application Identification in Network Forensics

Jan Pluskal, Ondrej Lichtner, Ondrej Rysavy

Chapter 10

# TRAFFIC CLASSIFICATION AND APPLICATION IDENTIFICATION IN NETWORK FORENSICS

Jan Pluskal, Ondrej Lichtner and Ondrej Rysavy

**Abstract** Network traffic classification is an absolute necessity for network monitoring, security analyses and digital forensics. Without accurate traffic classification, the computational demands imposed by analyzing all the IP traffic flows are enormous. Classification can also reduce the number of flows that need to be examined and prioritized for analysis in forensic investigations.

This chapter presents an automated feature elimination method based on a feature correlation matrix. Additionally, it proposes an enhanced statistical protocol identification method, which is compared against Bayesian network and random forests classification methods that offer high accuracy and acceptable performance. Each classification method is used with a subset of features that best suit the method. The methods are evaluated based on their ability to identify the application layer protocols and the applications themselves. Experiments demonstrate that the random forests classifier yields the most promising results whereas the proposed enhanced statistical protocol identification method provides an interesting trade-off between higher performance and slightly lower accuracy.

**Keywords:** Protocol identification, application identification, machine learning

## 1. Introduction

Network traffic classification is an important technique used in network monitoring, security analyses and digital forensics. In digital forensics, file types can be identified by file extensions or by searching for magic numbers at the beginning of files; known files can be identified using databases of hash values. The identification of file types and filtering of known files reduce the amount of data that needs to be analyzed. Do-

ing the same with network traffic is much more complicated because each data transfer contains specific and temporary characteristics that depend on the network state, network utilization and locations of communications endpoints. The correct classification of network traffic enables an automated analyzer to determine which application protocol parser to use to extract information carried by an IP flow (a packet sequence identified by the same source and destination IP addresses, transport protocol ports and transport protocol type). This, in turn, helps speed up a forensic investigation by reducing the number of unclassified IP flows.

Traditional traffic classification methods identify applications based on the TCP or UDP ports that are used. This provides only limited accuracy (60–80%) because many applications use random or non-standard ports [3, 24], for example, peer-to-peer applications, multimedia streaming applications, computer games and tunneled traffic. Advanced traffic classification utilizes supervised machine learning methods based on payload analysis, statistical methods and hybrid approaches [17, 19, 26, 27, 29]. Each technique has its advantages and disadvantages. For example, payload analysis of encrypted communications is unacceptably inaccurate. Statistical and hybrid approaches demonstrate that it is not necessary to rely exclusively on packet content [5, 12, 21], but that it is possible to combine structural and behavioral features to increase detection accuracy [16].

Unsupervised machine learning methods can classify unknown network traffic [9] into unlabeled clusters based on their similarity. An expert investigator, upon inspection of a few samples of a cluster, can label the entire cluster.

Several researchers have investigated machine learning approaches for traffic classification. Most of the research has focused on classifying network traffic to identify the application layer protocol in order to support intelligent network filtering and security monitoring. While traffic classification for network forensics stems from the same ideas, there are some notable differences. Network forensics analysis can be performed off-line on captured data. In this case, accuracy is more important than speed. Thus, a combination of several methods or applications that are slower, but more accurate, can be considered.

In network forensics, an investigator can compensate for incorrect results by performing additional manual inspections of results. For example, some methods return a probability vector that can be inspected to consider different results.

Additionally, in network forensics, classification must be deterministic because forensic principles require that all results be verifiable.

Also, classification methods can be tuned by an investigator and can be repeated with different parameter sets to increase sensitivity while decreasing specificity.

Machine learning algorithms for network traffic classification have been studied since the 1990s. The most common algorithms include support vector machines [12], decision tree algorithms [21] and probabilistic [5] and statistical methods [16, 19], all of which involve supervised learning. The unsupervised $k$-means clustering algorithm [9] groups traffic based on its significant features. If the feature set is selected properly, a machine learning method can exceed 90% accuracy [26].

Surveys of classification methods by Nguyen and Armitage [27] and Namdev et al. [26] discuss protocol identification. Classification methods for encrypted traffic are reviewed in [29]. Al Khater and Overill [2] have proposed the use of machine learning algorithms to improve traffic classification methods for digital forensic applications. Foroushani and Zincir-Heywood [10] have demonstrated the possibility of identifying high-level application behaviors from encrypted network service communications. Dai et al. [6] and Miskovic et al. [23] have described methods for fingerprinting mobile applications based on their communications. Erman et al. [8] have explored flow-based classification and have proposed a semi-supervised classification method that can accommodate known and unknown applications.

While traffic classification has been applied extensively to network monitoring and security analysis, significantly less research has focused on traffic discrimination for network forensics. This research makes some key contributions to the field of network forensics. The first is the creation of a dataset that provides a means to reliably acquire ground truth for experiments. Typical datasets use information inferred from `l7-filter` [28] or `nmap` [1] and, therefore, offer only approximations of the real information. Shang and Huang [28] have shown that the precision of these techniques is always one (no false positives), but the recall varies between 0.67 and 0.87. This means that 13–33% of the samples are not labeled and the researchers would have excluded them from the datasets because they lacked labels [1, 12]. Therefore, the remaining dataset is already classifiable via deep packet inspection and is less relevant to finding better classification methods. In other cases, researchers do not include information about the data used in their experiments, or the descriptions are vague and not reproducible [28], or they do not describe how to annotate data with labels without errors [5].

For these reasons, this research captured one week's worth of packet data in an environment with eight hosts, which translates to roughly

20 GB. The data was automatically tagged with complete information about the origin application.

This research has also developed an enhanced statistical protocol identification (ESPI) method that leverages a machine-learning-based classifier. Upon evaluating the results of related studies, two additional classifiers, a Bayesian network classifier and a random forests classifier, were selected for comparison. This chapter describes all three methods and shows that they can be used to identify application layer protocols and even the applications that used the protocols. This is important because application identification provides more information about network traffic compared with what can be gleaned from the identified application layer protocols. Consider a situation where HTTPS is used to create an encrypted tunnel. A tool capable of recognizing applications (e.g., Google Drive, iTunes and OneDrive) in network traffic instead of merely the application layer protocol (e.g., HTTPS) is useful in several domains. Notably, in forensic analysis, application identification could significantly reduce the amount of data to be analyzed compared with conventional approaches.

## 2. Data Collection and Preprocessing

Network traffic classification takes a network traffic capture file as input, typically in the PCAP format. The captured traffic is then split into a collection of layer 4 conversations represented by one or two IP flows for one-way or two-way communications, respectively. The experiments described in this chapter employed an annotated dataset captured by Microsoft Network Monitor, which provides application labels for almost all conversations. The dataset contains regular network traffic generated by eight user workstations running the Windows operating system. The final capture file has the following characteristics:

- **PCAP File Size:** 19.5 GB.
- **PCAP Format:** Microsoft NetMon 2.x.
- **Capture Duration:** 119 hours.
- **Number of Packets:** 27,616,138.
- **Number of Layer 7 Conversations:** 269,459.
- **Number of Application Protocols:** 58.
- **Number of Communicating Applications:** 93.

Information about the dataset is available at `pluskal.github.io/AppIdent` and the dataset itself can be downloaded from `nes.fit.vutbr.cz/AppIdent`.

Before the capture file could be used, additional post-processing steps from previous work [22] were applied to enhance data extraction. The final post-processing step used a round of experiments with the enhanced statistical protocol identification method. Based on these initial results, a second instance of the dataset was created that contained ground truth about the application protocols. The ground truth supported manual hierarchical clustering analysis of the results.

The post-processing steps improved the traffic classification accuracy by reducing the noise in the extracted features caused by the following items:

- Important TCP session control information, such as synchronization segments and finalization segments, may be missing.
- Sequence numbers may overflow in long-running TCP conversations. This can result in incorrect interpretation, causing single conversations to be split or two unrelated IP flows to be joined into a single conversation.
- The joining of capture files from multiple probes must address issues related to possible packet duplication and the proper ordering of packets belonging to the same conversation.
- Some IP packets may be missing or be duplicated (e.g., in the case of TCP retransmission).
- Finally, associated IP flows in bidirectional conversations must be paired correctly.

Matousek et al. [22] have shown that other network forensic solutions do not effectively address these issues. This implies that adopting the proposed additional steps would also be beneficial in the context of network traffic classification. To address these issues, Netfox Detective (`github.com/nesfit/NetfoxDetective`), a custom tool created for these use cases, was used to process the captured PCAP files.

## 2.1 Application Conversations and Messages

In addition to addressing the basic issues related to processing layer 4 conversations, Netfox Detective also enabled the dataset to be processed to track layer 7 conversations and to approximate individual application messages. This increased the classification accuracy by identifying application communications patterns. It also eliminated remnants of network packet fragmentation in the Internet layer and TCP retransmission in the transport layer. Packet fragmentation and TCP retransmission are

independent of application communications patterns and, thus, can negatively impact classification.

An application message was identified in the reassembled stream based on the transport protocol. The following rules were used for identification:

- If a stream uses the UDP transport protocol, then the entire payload of each UDP datagram is considered to be a single application message.
- In the case of the TCP transport protocol, segments are separated into application messages based on packets with PSH, RST or FIN flags, or based on timeouts.

These rules are simple to implement and yield accurate approximations of application messages in most cases.

## 3. Classification Methods

Using machine learning algorithms to classify traffic is by no means a new concept in the field of network forensics. However, the typical use case is to identify the application protocol [27, 29]. In this research, the approach was expanded to also identify the application that created the traffic. This provides more information that can be used by a forensic investigator for easier and more precise analysis.

This section describes revisions to the commonly-used feature sets [16, 19, 25] to address the task at hand and presents a feature elimination method based on feature correlation to improve the accuracy of the created classifiers. Finally, the proposed enhanced statistical protocol identification method is described along with two other classification methods from the literature that have yielded promising traffic identification results.

### 3.1 Feature Set

The quality of a feature set directly influences classification accuracy [32]. Common features used for traffic classification are related to key aspects of packet communications and network architecture. These include port numbers, transport protocol type, starting sequence of payload bytes, pattern occurrence, message length and message timing. Researchers have identified a list of possible features comprising 92 items that are invariant to network line characteristics [16, 19, 25]. The list is available at `github.com/pluskal/AppIdent`.

Machine learning algorithms achieve the best performance when the selected features are orthogonal (i.e., no correlation exists between the

features) [14]. Several approaches have been proposed for calculating feature correlations, including the Pearson, Spearman, Kendall correlation formulas [31] and covariance matrix [13]. This research opted for the covariance matrix method due to its ease of implementation.

The covariance matrix provides a correlation value for each pair of features. This matrix was used to design an automated two-step procedure for eliminating features. In the first step, a covariance matrix was calculated based on a chosen ratio of training data to verification data ($t/v$). In the second step, based on a maximum allowed correlation value, feature pairs with higher correlation values were identified and features that were, on average, more correlated with all the other features, were iteratively removed from the feature set. The resulting feature set was used by the selected classification method and could be evaluated to find the optimal set.

In the experiments, more than 80% of the feature pairs had correlation values of 0.5 or higher. Table 1 lists the features that remained after feature elimination was performed on sample data with training to verification ratios of 0.1 and 0.2, based on accepted correlation values up to 0.5. Note that the correlation column shows the maximal-allowed correlation values of features listed on the corresponding line and higher. These feature sets were used by the Bayesian network and random forests classifiers.

Most of the features describe flow characteristics instead of individual packet characteristics. This confirms the assumption that relying on a signature or some pattern in packet content gives better results for encrypted or less-structured traffic.

## 3.2 Enhanced Statistical Protocol Identification

Hjelmvik [16] developed the statistical protocol identification (SPID) method for use with the NetworkMiner tool. The learning phase of the method creates a database of protocol fingerprints for identifying application protocols. The features utilized by the statistical protocol identification method are called "protocol attribute meters," each conveying different information. Some items are scalar values representing payload data size, number of packets in a session or port number. Other items are composite values, such as a tuple comprising packet direction, packet ordering, packet size and byte value frequency.

The original implementation uses about 35 protocol attribute meters and extracts information from the first few packets of IP flows to achieve better speed compared with other classification methods that analyze entire IP flows. The distance between the analyzed data to a

*Table 1.* Features remaining after elimination based on $t/v$ ratios of 0.1 and 0.2.

| **Correlation** | **Feature (t/v = 0.1)** | **Feature (t/v = 0.2)** |
|---|---|---|
| | BytePairsReoccuringDownFlow | |
| | DirectionChanges | |
| | First3BytesEqualDownFlow | First3BytesEqualDownFlow |
| | FirstBitPositionUpFlow | FirstBitPositionUpFlow |
| | FirstPayloadSize | |
| | MinInterArrivalTimeDownFlow | |
| | MinInterArrivalTimePackets | MinInterArrivalTimePackets |
| | UpAndDownFlow | UpAndDownFlow |
| | MinPacketLengthDownFlow | MinPacketLengthDownFlow |
| | NumberOfBytesDownFlow | |
| | NumberOfPacketsUpFlow | |
| | PacketLengthDistribution DownFlow | PacketLengthDistribution DownFlow |
| | PacketLengthDistribution UpFlow | |
| | | ThirdQuartileInterArrival TimeUp |
| | | ByteFrequencyUpFlow |
| | | MaxSegmentSizeDown |
| | | MaxSegmentSizeUp |
| | | MinInterArrivalTimePackets UpFlow |
| | | NumberOfBytesUpFlow |
| | | ThirdQuartileInterArrival TimeDown |
| <0.25 | PUSHPacketsDown | PUSHPacketsDown |
| | ThirdQuartileInterArrival TimeDown | |
| | | NumberOfBytesUpFlow |
| <0.3 | | FirstPayloadSize |
| | ByteFrequencyUpFlow | |
| | MinPacketLengthUpFlow | MinPacketLengthUpFlow |
| | NumberOfPacketsPerTimeUp | |
| | | DirectionChanges |
| | | BytePairsReoccuringDownFlow |
| <0.4 | | MeanPacketLengthUpFlow |
| <0.5 | MeanPacketLengthUpFlow | |

known protocol fingerprint is computed using the Kullback-Leibler divergence and the best matching protocol fingerprint has the smallest sum of Kullback-Leibler divergences over all the attributes. Kohnen et al. [19] have developed a new version of the statistical protocol iden-

tification method by adding support for UDP and handling streaming protocols using a different set of protocol attribute meters.

The research described here has drawn on this work in creating the enhanced statistical protocol identification method. The research was motivated by the fact that a forensic investigator is more interested in the precision of identification than its speed (although quicker identification is important); therefore, completed conversations are analyzed instead of just the first few packets. Additionally, as mentioned above, the intent is to identify application protocols as well as the applications themselves; therefore, approximated application messages instead of individual packets are analyzed. The enhanced statistical protocol identification method also uses a different set of features (92 features selected as described in Section 3.1) and a different method for computing the distances between measured values and learned protocol fingerprints.

The following three functions are employed:

- Function $f$ computes the divergence of a measured value to a fingerprint value.
- Function $g$ returns a normalized feature value for an actual measured value.
- Function $w$ returns the weight of a feature for a protocol fingerprint.

The divergence from a learned fingerprint is computed as the Euclidean distance [7] of the weighted divergences for individual features:

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

where $x_1, \ldots, x_n$ denote the current flow protocol feature values; $c_1, ..., c_n$ denote the normalized feature values in the protocol fingerprint; and $w_i(c)$ denotes the weight of the $i^{th}$ feature in protocol fingerprint $c$.

Equation (1) is used to compute the difference $d_{x,c^j}$ for each protocol fingerprint $c^j$. The identified protocol or application $k$ is the one such that $d_{x,c^k} = min(d_{x,c^1} \ldots d_{x,c^m})$.

Compared with other machine learning methods, the enhanced statistical protocol identification method does not suffer from overfitting due to the use of correlated features because it assigns weights on a per-feature basis. This property renders the enhanced statistical protocol identification method readily extensible to classifying new protocols and incorporating features unique to the new protocols, which could be correlated with features of other protocols.

### 3.3 Bayesian Network Classifier

The Bayesian network classifier [11] relies on Bayes' theorem, which defines the probability of an event based on prior knowledge about the conditions related to the occurrence of the event. The classifier incorporates Bayesian belief networks that are constructed during the learning phase. A Bayesian network is a directed acyclic graph and a set of conditional probability tables. Nodes in the network represent feature variables and edges represent conditional dependencies. The probability tables provide probability functions for the nodes.

A Bayesian network classifier identifies the application protocol by determining the node (or set of nodes) with the highest probability for the given input feature values. The advantage of the Bayesian network classifier is that it also computes the probability that the conversation belongs to the identified protocol. This information enables a forensic investigator to decide whether or not to analyze the conversation.

### 3.4 Random Forests Classifier

Random forests is an ensemble method that constructs multiple C4.5 decision trees during the training phase; the trees are used for classification in the verification phase, where the mode of the partial results is selected as the resulting class [4]. This makes the random forests classifier prone to overfitting [15]. Random forests are parametrized by multiple variables such as the forest count, join, and training to verification ratio. Optimal values for the parameters are determined by cross-validation and computation of an out-of-bag error that estimates the performance of specific parameter combinations. Because the classifier computes the out-of-bag error, there is no need to employ a separate data verification phase. Therefore, the random forests classifier can be trained on the entire dataset, although this approach can be computationally expensive.

## 4. Experimental Procedures and Results

This section presents the experimental procedures and the results obtained using the three classification methods. The experiments were designed with three goals in mind. The first goal was to compare results yielded by machine learning and statistical methods that share the same base feature set, but involve fundamentally-different approaches to classification. The second goal was to observe how the training set size and feature elimination ratio impact the accuracy of application protocol and application classification. The third goal was to prove (or

disprove) that application classifiers can identify network traffic based on the applications that generated the traffic.

The Netfox Detective tool was employed as middleware for parsing and processing the captured traffic into application conversations and messages. The feature elimination algorithm and classification methods were implemented as modules in Netfox Detective for easy integration with input data. A standalone application was used to automate the experimental procedure with different parameters. The enhanced statistical protocol identification method was implemented from scratch. The Bayesian network and random forests classifiers were implemented using the Accord.NET library of machine learning algorithms.

## 4.1 Experimental Procedures

As mentioned above, Netfox Detective was used to parse and process the captured traffic and to extract the full set of feature values for the resulting conversations (feature vectors). Each feature vector was annotated with a label that identified the level of classification using the ground truth from the original capture file. The following labels were used:

- **Application Protocol:** Each application protocol was labeled using a tuple with the components: (i) transport protocol type; and (ii) destination transport layer port or manually assigned label (e.g., `TCP_http`).
- **Application:** Each application was labeled using a tuple with the components: (i) transport protocol type; (ii) destination transport layer port or manually assigned label; and (iii) application process information (e.g., `tcp_http_skypeexe`).

Because this task was time-intensive, but only had to be done once, the results were saved in a separate binary file. A custom application was developed to automatically execute the same experiment, but with different configuration parameter values (classification method, training to verification ratio and accepted correlation value for feature elimination).

All the experiments involved the following five steps:

- **Step 1 (Dataset Generation):** The available data was split into two disjoint datasets based on the training to verification ratio. The first dataset was used for training and the second for verification.
- **Step 2: (Feature Elimination):** The experiments using the Bayesian network and random forests classifiers used the training

dataset created in Step 1 with the feature elimination algorithm described in Section 3.1. The experiments using the enhanced statistical protocol identification method employed the accepted correlation value of one to include all the features; this is because, as explained in Section 3.2, the enhanced statistical protocol identification method does not require feature elimination.

- **Step 3: (Training):** The training dataset created in Step 1 was used to train the three classifiers:
  - **Bayesian Network Classifier:** A classifier was trained for each group of feature vectors with the same label.
  - **Random Forests:** The optimal parameters specified in Section 3.4 corresponded to the most accurate classifier.
  - **Enhanced Statistical Protocol Identification Classifier:** For each group of feature vectors with the same label, an application protocol fingerprint was computed using function $g$.
- **Step 4 (Verification):** A cross-validation phase was used to determine the best classifiers created in Step 3. Specifically, the classifiers were used to classify each conversation from the verification dataset. They returned either: (i) multiple labels; or (ii) single labels:
  - **Multiple Labels:** Multiple labels were returned as a set of probabilities or distances. The set was ordered and the label with the highest probability or shortest distance was selected. In the case of the Bayesian network classifier, each Bayesian classifier yielded a probability of the current conversation belonging to the class of interest (application protocol or application) represented by the classifier. In the case of the enhanced statistical protocol identification classifier, the Euclidean distance between the specific conversation to each application protocol or application fingerprint was returned.
  - **Single Label:** The random forests classifier returned a single label.
- **Step 5 (Label Comparison):** In each case, the label was compared against the annotation and the statistical properties of each classification method were computed.

*Table 2.* Configurations of the classification methods.

| Classification Method | Experiment ID | Training to Verification Ratio | Highest Feature Correlation Used |
|---|---|---|---|
| Bayesian Network | 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 | ESPI1 | 0.7 | 1 |
| | ESPI2 | 0.2 | 1 |
| Random Forests | RF1 | 0.1 | 0.4 |
| | RF2 | 0.2 | 0.4 |
| | RF3 | 0.1 | 0.5 |
| | RF4 | 0.2 | 0.5 |

## 4.2 Experimental Results

The automated application ran many experiments with various configurations of parameters with the goal of identifying the configurations that yielded the best results. The experiments were organized based on the classification methods. For better comparisons, the most successful experiments for each method with various training to verification ratios were employed.

Table 2 lists the configurations of the classification methods with the best results. The last column specifies the highest feature correlation values used for feature elimination. The experiments were split into two categories. Experiments B1, B2, B3, ESPI1, RF1 and RF2 used classifiers for application protocol identification, for which the complete dataset contained 58 application protocol tags. The remaining experiments B4, B5, B6, ESPI2, RF3 and RF4 used classifiers for application identification, for which the complete dataset contained 93 application tags. All the experimental results are available at `pluskal.github.io/AppIdent`. The figures and tables in this section show the truncated results of the experiments. The truncation was performed by selecting the best experiment in each category as a baseline. The 20 most accurately identified labels are shown for all the experiments in a category.

The labels returned by the classification methods were compared with the ground truth from the original captured data and separated into four categories defined by the confusion matrix in Table 3. Note that a classi-

*Table 3.* Confusion matrix for a single label (application protocol or application).

| **Classification Result / Ground Truth** | **Positive** | **Negative** | **Total** |
|---|---|---|---|
| Positive | True Positive ($TP$) | False Positive ($FP$) | $P$ |
| Negative | False Negative ($FN$) | True Negative ($TN$) | $N$ |
| Total | $P'$ | $N'$ | $P+N$ |

fication result is positive when the classifier returns that the conversation can be labeled with the label and negative when it cannot. The ground truth is positive when the conversation in the dataset is actually labeled with the label and negative when it is not.

The F-measure, also referred to as the balanced F-score [14], was used to compare the classification methods. This single score is computed as the harmonic mean of the precision and recall using the equation:

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

where the precision and recall are computed from the corresponding confusion matrix values using the equations:

$$precision = \frac{TP}{TP + FP} \tag{3}$$

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{4}$$

Figure 1 presents the visualization of the application protocol identification results. The two random forest classifiers (RF1 and RF2) were very accurate. The Bayesian network classifier (B3) also performed very well, but it required a larger training set, a training to verification ratio of 0.5 and more features (see Table 2).

Figure 2 presents the visualization of the application identification results. The two random forest classifiers again yielded the best results. However, in this case, the Bayesian network classifiers were outperformed by the enhanced statistical protocol identification classifier, which also provided the best trade-off between performance and accuracy.

Figure 3 provides the aggregate statistics for all the classes. The number in each cell corresponds to the number of labels that were classified with F-measures greater than or equal to the F-measure value. Note that the size of the shaded area in a cell is proportional to the number of labels classified in the cell.

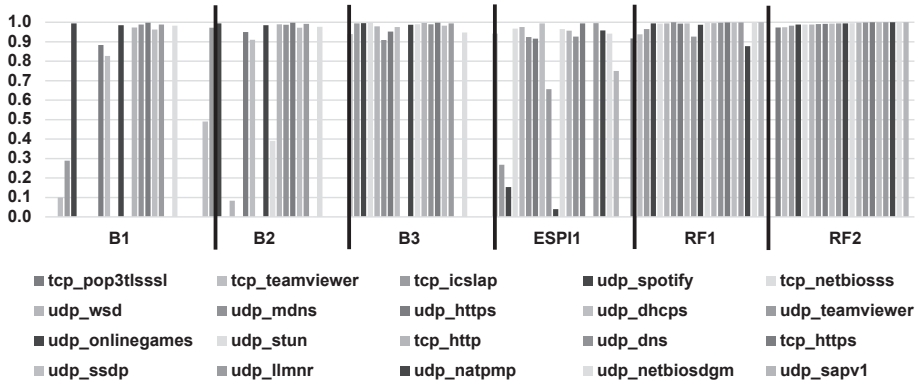


*Figure 1.* Performance of application protocol classifiers using the F-measure.

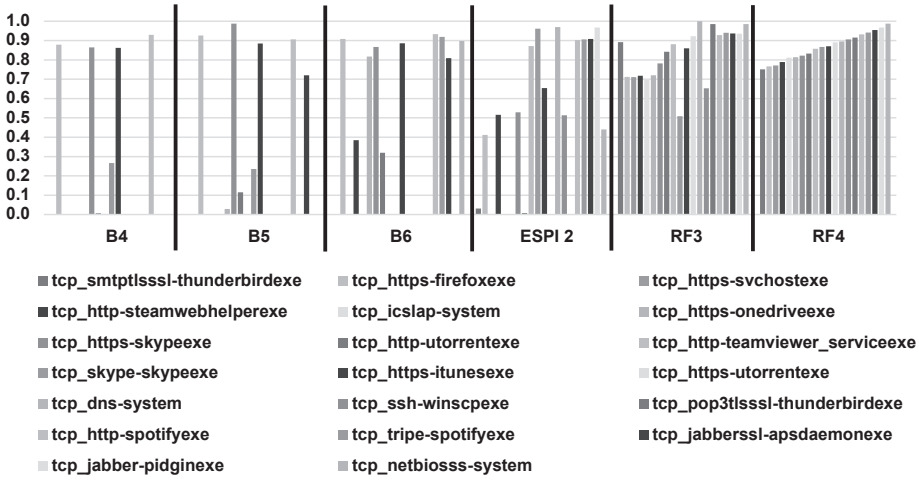


*Figure 2.* Performance of application classifiers using the F-measure.

Figure 4 presents the results of the performance comparison of application protocol classifiers. The first row shows the times required to complete all the steps involved in the experiments. The remaining rows show the F-measure scores of each evaluated method for the top 20 labels based on the most successful experiment in the category.

Figure 5 presents the results of the performance comparison of application classifiers. Once again, the first row shows the times required to complete all the steps involved in the experiments. The remaining rows show the F-measure scores of each evaluated method for the top 20 labels based on the most successful experiment in the category.

| GreaterOrEqual F-Measure | B1 | B2 | B3 | ESPI1 | RF1 | RF2 | B4 | B5 | B6 | ESPI2 | RF3 | RF4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.0 | 58 | 58 | 58 | 58 | 58 | 58 | 93 | 93 | 93 | 93 | 93 | 93 |
| 0.1 | 21 | 19 | 23 | 33 | 47 | 51 | 22 | 25 | 36 | 43 | 83 | 83 |
| 0.2 | 16 | 18 | 23 | 31 | 45 | 47 | 22 | 23 | 34 | 40 | 77 | 77 |
| 0.3 | 14 | 18 | 22 | 29 | 41 | 45 | 20 | 22 | 34 | 37 | 74 | 75 |
| 0.4 | 14 | 16 | 22 | 29 | 40 | 43 | 19 | 22 | 30 | 36 | 68 | 70 |
| 0.5 | 14 | 14 | 22 | 28 | 37 | 41 | 19 | 22 | 29 | 31 | 63 | 63 |
| 0.6 | 13 | 14 | 22 | 26 | 36 | 39 | 16 | 20 | 27 | 27 | 54 | 58 |
| 0.7 | 12 | 13 | 21 | 24 | 34 | 37 | 15 | 17 | 26 | 22 | 45 | 47 |
| 0.8 | 11 | 12 | 19 | 21 | 32 | 36 | 13 | 13 | 26 | 20 | 38 | 41 |
| 0.9 | 8 | 12 | 18 | 17 | 26 | 31 | 7 | 12 | 15 | 17 | 25 | 28 |

*Figure 3.* Summary of classification method performance.

| 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 |

*Figure 4.* Performance comparison of application protocol classifiers.

## 5. Conclusions

This research has focused on the important network forensics problem of identifying network applications in addition to just application protocols in network traffic flows. It has studied various aspects of applying machine learning methods and the selection of features that characterize application behavior, such as message timing, content length and TCP flags instead of features related to network line characteristics. An automated feature elimination method based on the feature correlation

| AppProtocol | B4 | B5 | B6 | ESPI 2 | RF3 | RF4 |
|---|---|---|---|---|---|---|
| Time [h] | 0:53 | 1:03 | 2:00 | 1:11 | 20:13 | 23:20 |
| tcp_smtptlsssl-thunderbirdexe | 0.00 | 0.00 | 0.00 | 0.03 | 0.89 | 0.75 |
| tcp_https-firefoxexe | 0.88 | 0.93 | 0.91 | 0.41 | 0.71 | 0.77 |
| tcp_https-svchostexe | 0.00 | 0.00 | 0.00 | 0.00 | 0.71 | 0.77 |
| tcp_http-steamwebhelperexe | 0.00 | 0.00 | 0.38 | 0.52 | 0.72 | 0.79 |
| tcp_icslap-system | 0.00 | 0.00 | 0.00 | 0.00 | 0.70 | 0.81 |
| tcp_https-onedriveexe | 0.00 | 0.03 | 0.82 | 0.00 | 0.72 | 0.81 |
| tcp_https-skypeexe | 0.86 | 0.99 | 0.87 | 0.53 | 0.78 | 0.82 |
| tcp_http-utorrentexe | 0.01 | 0.11 | 0.32 | 0.01 | 0.84 | 0.83 |
| tcp_http-teamviewer_serviceexe | 0.00 | 0.00 | 0.00 | 0.87 | 0.88 | 0.86 |
| tcp_skype-skypeexe | 0.27 | 0.24 | 0.00 | 0.96 | 0.51 | 0.87 |
| tcp_https-itunesexe | 0.86 | 0.89 | 0.89 | 0.65 | 0.86 | 0.87 |
| tcp_https-utorrentexe | 0.00 | 0.00 | 0.00 | 0.00 | 0.92 | 0.89 |
| tcp_dns-system | 0.00 | 0.00 | 0.00 | 0.97 | 1.00 | 0.89 |
| tcp_ssh-winscpexe | 0.00 | 0.00 | 0.00 | 0.51 | 0.65 | 0.91 |
| tcp_pop3tlsssl-thunderbirdexe | 0.00 | 0.00 | 0.00 | 0.00 | 0.98 | 0.92 |
| tcp_http-spotifyexe | 0.93 | 0.91 | 0.93 | 0.90 | 0.93 | 0.93 |
| tcp_tripe-spotifyexe | 0.00 | 0.00 | 0.92 | 0.91 | 0.94 | 0.94 |
| tcp_jabberssl-apsdaemonexe | 0.00 | 0.72 | 0.81 | 0.91 | 0.94 | 0.95 |
| tcp_jabber-pidginexe | 0.00 | 0.00 | 0.00 | 0.97 | 0.94 | 0.97 |
| tcp_netbiosss-system | 0.00 | 0.00 | 0.90 | 0.44 | 0.98 | 0.99 |

*Figure 5.* Performance comparison of application classifiers.

matrix was employed to improve the classification results. Additionally, this research has developed the enhanced statistical protocol identification method, which was compared against the Bayesian network and random forests classification methods from the literature that offer high accuracy and acceptable performance.

The experimental results confirm that application protocols as well as the applications that generate network traffic can be classified with high confidence. For example, NetBIOS service and DNS were identified accurately and several common applications that use the HTTP(S) application protocol were identified with high accuracy. Similarly, it was possible to distinguish between communications traces of OneDrive, Skype, iTunes, Spotify, Steam and $\mu$Torrent clients, although all of them use the same application protocol (HTTPS).

The random forests classifier achieved the best results, confirming the results obtained by other researchers [20, 30] who experimented with machine learning approaches for traffic classification. The enhanced statistical protocol identification classifier yielded better results than the Bayesian network classifier and was much faster than the Bayesian network and random forests classifiers.

Classification accuracy is mainly determined by the quality of the selected features. This research has employed features based on previous observations and intuition. Future research should focus on the systematic analysis and selection of feature sets that could improve classification accuracy and robustness.

To improve the identification of applications that employ the same application protocol (e.g., removing errors when `tcp_http_skypeexe` is classified as `tcp_http_firefoxexe`, or vice-versa), future research should focus on hierarchical classification methods. An example is hierarchical clustering based on enhanced statistical protocol identification fingerprints. A forensic investigator could then infer the actual application classes by visual cluster analysis. This approach could also be extended to other levels such as application message level.

Future research should also consider combining multiple classifiers [18] to increase the confidence in the results. Research should also focus on semi-supervised classification methods [8] that enable the creation of models from partially-labeled data.

Finally, experiments should be conducted to extend the classification models and evaluate the properties of other datasets. The classification methods considered in this work require accurate models. Creating such models requires the analysis of large numbers of traffic samples. Experimenting with different datasets could provide more accurate classification models and valuable insights into the properties of individual classification methods.

A reference implementation is available under an MIT license from GitHub at `pluskal.github.io/AppIdent`. This includes the framework for parsing captured data, extracting features and eliminating features, along with the three classifiers described in this chapter and the standalone application that automated the experiments. The dataset is available at `nes.fit.vutbr.cz/AppIdent` to facilitate the reproducibility of the experiments and to serve as a benchmarking platform for testing other machine-learning-based application identification methods.

## References

[1] S. Alcock and R. Nelson, Libprotoident: Traffic Classification Using Lightweight Packet Inspection, Technical Report, WAND Network Research Group, Computer Science Department, University of Waikato, Hamilton, New Zealand, 2012.

[2] N. Al Khater and R. Overill, Forensic network traffic analysis, *Proceedings of the Second International Conference on Digital Security and Forensics*, 2015.

[3] T. Auld, A. Moore and S. Gull, Bayesian neural networks for Internet traffic classification, *IEEE Transactions on Neural Networks*, vol. 18(1), pp. 223–239, 2007.

[4] L. Breiman, Random forests, *Machine Learning*, vol. 45(1), pp. 5–32, 2001.

[5] E. Bursztein, Probabilistic identification of hard to classify protocols, *Proceedings of the Second IFIP WG 11.2 International Conference on Information Security Theory and Practices: Smart Devices, Convergence and Next Generation Networks*, pp. 49–63, 2008.

[6] S. Dai, A. Tongaonkar, X. Wang, A. Nucci and D. Song, NetworkProfiler: Towards automatic fingerprinting of Android apps, *Proceedings of the IEEE International Conference on Computer Communications*, pp. 809–817, 2013.

[7] M. Deza and E. Deza, *Encyclopedia of Distances*, Springer-Verlag, Berlin Heidelberg, Germany, 2009.

[8] J. Erman, A. Mahanti, M. Arlitt, I. Cohen and C. Williamson, Offline/real-time traffic classification using semi-supervised learning, *Performance Evaluation*, vol. 64(9-12), pp. 1194–1213, 2007.

[9] A. Finamore, M. Mellia and M. Meo, Mining unclassified traffic using automatic clustering techniques, *Proceedings of the Third International Conference on Traffic Monitoring and Analysis*, pp. 150–163, 2011.

[10] V. Foroushani and A. Zincir-Heywood, Investigating application behavior in network traffic traces, *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 72–79, 2013.

[11] N. Friedman, D. Geiger and M. Goldszmidt, Bayesian network classifiers, *Machine Learning*, vol. 29(2-3), pp. 131–163, 1997.

[12] G. Gomez Sena and P. Belzarena, Early traffic classification using support vector machines, *Proceedings of the Fifth International Latin American Networking Conference*, pp. 60–66, 2009.

[13] I. Guyon and A. Elisseeff, An introduction to variable and feature selection, *Journal of Machine Learning Research*, vol. 3(March), pp. 1157–1182, 2003.

[14] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, Waltham, Massachusetts, 2012.

[15] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning – Data Mining, Inference and Prediction*, Springer, New York, 2009.

[16] E. Hjelmvik, The SPID Algorithm – Statistical Protocol Identification, Gavle, Sweden (`www.iis.se/docs/The_SPID_Algorithm_-_Statistical_Protocol_IDentification.pdf`), 2008.

[17] 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(2), pp. 101–120, 2014.

[18] J. Kittler, Combining classifiers: A theoretical framework, *Pattern Analysis and Applications*, vol. 1(1), pp. 18–27, 1998.

[19] C. Kohnen, C. Uberall, F. Adamsky, V. Rakocevic, M. Rajarajan and R. Jager, Enhancements to statistical protocol identification (SPID) for self-organized QoS in LANs, *Proceedings of the Nineteenth International Conference on Computer Communications and Networks*, 2010.

[20] J. Li, S. Zhang, Y. Xuan and Y. Sun, Identifying Skype traffic by random forests, *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 2841–2844, 2007.

[21] Y. Luo, K. Xiang and S. Li, Acceleration of decision tree searching for IP traffic classification, *Proceedings of the Fourth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pp. 40–49, 2008.

[22] P. Matousek, J. Pluskal, O. Rysavy, V. Vesely, M. Kmet, F. Karpisek and M. Vymlatil, Advanced techniques for reconstruction of incomplete network data, *Proceedings of the Seventh International Conference on Digital Forensics and Cyber Crime*, pp. 69–84, 2015.

[23] S. Miskovic, G. Lee, Y. Liao and M. Baldi, AppPrint: Automatic fingerprinting of mobile applications in network traffic, *Proceedings of the Sixteenth International Conference on Passive and Active Measurement*, pp. 57–69, 2015.

[24] A. Moore and K. Papagiannaki, Toward the accurate identification of network applications, *Proceedings of the Sixth International Workshop on Passive and Active Network Measurement*, pp. 41–54, 2005.

[25] A. Moore, D. Zuev and M. Crogan, Discriminators for Use in Flow-Based Classification, Technical Report RR-05-13, Department of Computer Science, Queen Mary, University of London, London, United Kingdom, 2013.

[26] N. Namdev, S. Agrawal and S. Silkari, Recent advancements in machine learning based Internet traffic classification, *Procedia Computer Science*, vol. 60, pp. 784–791, 2015.

[27] T. Nguyen and G. Armitage, A survey of techniques for Internet traffic classification using machine learning, *IEEE Communications Surveys and Tutorials*, vol. 10(4), pp. 56–76, 2008.

[28] C. Shen and L. Huang, On the detection accuracy of the `l7-filter` and OpenDPI, *Proceedings of the Third International Conference on Networking and Distributed Computing*, pp. 119–123, 2012.

[29] P. Velan, M. Cermak, P. Celeda and M. Drasar, A survey of methods for encrypted traffic classification and analysis, *International Journal of Network Management*, vol. 25(5), pp. 355–374, 2015.

[30] Y. Wang and S. Yu, Machine learned real-time traffic classifiers, *Proceedings of the Second International Symposium on Intelligent Information Technology Applications*, vol. 3, pp. 449–454, 2008.

[31] I. Zezula, On multivariate Gaussian copulas, *Journal of Statistical Planning and Inference*, vol. 139(11), pp. 3942–3946, 2009.

[32] L. Zhen and L. Qiong, A new feature selection method for Internet traffic classification using ML, *Physics Procedia*, vol. 33, pp. 1338–1345, 2012.