

MACHINE LEARNING BASED CLASSIFIER FOR SERVICE FUNCTION CHAINS

Habib Allah KHOSRAVI, Yaghoub FARJAMI

*Department of Computer Engineering
Faculty of Engineering, University of Qom
Alqadir Boulevard, Qom, Iran*

e-mail: h.a.khosravi@stu.qom.ac.ir, farjami@qom.ac.ir

Abstract. Using service function chains, Internet Service Providers can customize the use of service functions that process the network flows belonging to their customers. Each network flow is injected into a service chain according to the flow features. Since most of the malicious applications try not to get the proper analysis by imitating some valid and famous applications, classification based on simple flow features may waste processing power by using inappropriate service chains for evasive flows. In this paper, we have explored an application-aware classification approach using machine learning methods. Using CatBoost as a machine learning method, a model is created and used for traffic classification. We have provided some statistical reports on how this approach is compared with simple flow feature-based approaches in malicious environments and how feature selection can impact classification correctness. Choosing the most suitable number of features at the right time can beat traditional approaches in classification quality and provide better results in the service function chaining environment.

Keywords: Service function chaining, classifier, machine learning, catboost

Mathematics Subject Classification 2010: 68M10

1 INTRODUCTION

The ability to classify network traffic is crucial in the operation and management of networks. Simple flow features like source and destination port numbers or layer 4 protocol are traditionally used for traffic classification, or mapping flows

into traffic classes. Since most modern applications use famous port numbers on their server-side and lots of malicious applications try to evade the proper analysis by imitating other applications features, traditional approaches have proven to be ineffective. The solution to inefficiencies of flow-based classification approaches is to harness application characteristics of the flows and making the network application-aware.

By leveraging service function chaining, Internet Service Providers are able to customize the services provided to their users, based on the type of traffic. Services are provided as predefined chains specific to special traffic types. In an application-aware service function chaining network, service chains are selected more accurately, the most related flows are steered into the chains and fewer resources are wasted. Classification of network flows in the service chaining classifier node can be based on flow characteristics or the type of traffic (Figure 1). Using the latter approach, the classifier must do a thorough analysis of the flow’s packets to detect the type of traffic. Legacy traffic classification techniques are famous as the most expensive task in the network and they cannot be effective in situations where applications get frequent updates and change their signatures.

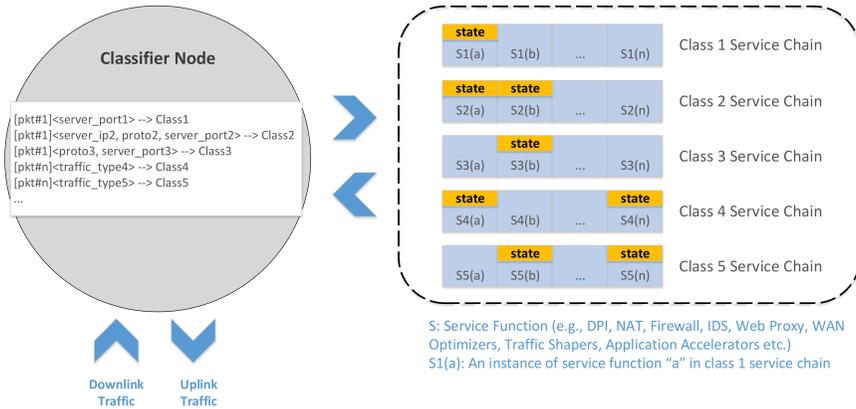


Figure 1. Service function chaining classifier node

Using machine learning algorithms in Internet traffic classification has recently received some attention [1, 2, 3, 4, 5, 6]. These approaches assume that applications send data in some patterns. These patterns can be used to classify traffic flows in different application classes. Flow features like the length of the flow, packet size, or total packet numbers can be used to find the application’s behavioral pattern. Flow features collected in various observation windows, lead to results with different accuracy. Finding the best observation window is a key factor in any classification scenario [7].

This paper explores the idea of using machine learning methods in service function chaining. Using CatBoost [8], decision tree models are created to examine

classification results in different conditions. An important point to consider when using any traffic classification method in service function chaining is that the classification should be performed at the early stages of a flow's life. Delayed traffic classification can impose an unnecessary burden on service functions for processing unrelated flows. By intelligently choosing the observation window and selecting the proper feature list, machine learning-based traffic classification at the beginning of flow is possible. Based on what was said, the motivation for the current work is to find a suitable solution for using machine learning methods in service function chaining. Therefore, the contributions of the current paper are the following:

- Use of a machine learning method in a service function chaining classifier node,
- Presenting an appropriate feature list extracted from the early stages of a traffic flow's life,
- Providing solutions for the special challenges that are present when machine learning methods are used in service function chaining environments,
- Proposing a machine learning-based early classification method with a high detection rate in service function chaining.

Different observations window sizes at the early stages of a flow's life are used for creating classification models. Based on the results, the best observation window size is selected and the desired feature list is detected. The results are compared with a signature-based approach to prove their effectiveness.

The rest of this paper is organized as follows. Section 2 provides the related work. Design and implementation details and challenges are described in Section 3. The results of the experiments accompanied by a detailed explanation are provided in Section 4. Finally, Section 5 concludes the paper.

2 RELATED WORKS

This section is dedicated to the works related to the topics that are discussed in this paper. We have divided all the topics into two categories of service function chaining and application-aware classification, and machine learning algorithms and tools for traffic classification. Each category is examined individually and some of the works done in that area are introduced. We start with service function chaining and application-aware classification which is the area where the problem was originally defined. We will start with the service function chaining problem statement and also introduce some of the solutions provided to solve those issues. Then, the notion of application-aware classification is stated and after that, some of the works done in this area are discussed. When we are done with the service function chaining part, we will introduce some of the works done in internet traffic classification using machine learning algorithms.

2.1 Service Function Chaining and Application Aware Classification

What serves as a service chain is the production of a related list of network functions and, consequently, the steering of network flows among them. The use of network service functions in traditional networks is accompanied by a set of constraints. As some of these constraints are described below, each of which will be briefly described [9, 10].

The first one described here is the topological dependency. The way network services are deployed in a computer network directly affects the order in which they are used for traffic processing and creates constraints when adding or changing network services. Another constraint present in the traditional networks is the configuration complexity which is a consequence of topological dependency. Considering service function chaining in this environment, simple actions like changing the order of service chains requires changes to the topology. Consistent ordering of service functions is another constraint which is the direct consequence of the dependency on topology. Many of the network functions work in a way that they need to be deployed in a specific order. Whereas changing the order in which service functions process network traffic is complex and difficult. Another important constraint is named the traffic selection criteria which refers to the problem that all traffic on a particular network segment traverses all service functions whether or not the traffic requires those services. Besides the aforementioned constraints, there are some others like constrained high availability, application of service policy, etc. that make the use of service function chaining inevitable. Service function chaining provides a set of solutions to resolve the problems existing in traditional networks and to make the network management more elastic and robust [10, 11].

The three solutions provided by the SFC working group at IETF to overcome the above constraints are service overlay, service classification, and SFC encapsulation. Service overlay is about creating service functions connectivity built on top of the existing network topology. It will be possible to create an arbitrary topology for connecting service functions in a required topology. Service classification and reclassification procedures are used to select which traffic enters an overlay and alter the sequence of service functions applied to traffic. Finally, using SFC encapsulation enables the exchange of metadata in the data plane between different components of the service chain [10, 12, 13, 15].

One of the challenges about traffic classification in service function chaining is using application data to select the service overlay where a network flow is going to be processed. In other words, instead of using the mere port numbers or the flow's network-level characteristics for traffic classification, information about applications involved in the generation of the traffic is used. Therefore, service overlays are selected more intelligently, and using service chaining's maximum potential becomes possible [14]. A deep packet inspection (DPI) function can do the task of application-aware service classification in a service function chaining scenario.

One of the works that has made use of the output of the traffic detection function (DPI service function) in creating service function chains is the *StEERING* [16]. The result of processing the traffic in DPI and the detected applications might be used in service chain selection for a network flow or it may alter the chain already selected for the flow. As the detection of applications in a DPI function might not be possible with the first received packet in flow, passing the packets to a new chain after the detection is finished might result in an incomplete analysis in some service functions.

SIMPLE is another solution that provides the definition of traffic routing policies between network services and translates these policies into traffic forwarding rules [17]. Different traffic classes along with their corresponding service function chains are defined by the network admin which are used for creating traffic forwarding rules based on the current state of the underlying network. Resource manager, dynamics handler, and rule generator are the three key components in *SIMPLE* system. Resource manager is in charge of providing the state of resources available in the network for the rule generator component which is the place where traffic forwarding rules are created. Dynamics handler detects the changes in traffic which are made by the service functions and provides the necessary information for the rule generator to create new forwarding rules. This component uses packet payload to find the connection between two distinct forms of one flow in the network.

There is also another solution for using the result of traffic detection in software-defined networks that has utilized the idea of delayed traffic flows [18]. When a new traffic flow is received at the ingress to the network, the packets are delayed and they are mirrored to a DPI box for the traffic detection process. When the result is provided, the traffic scheduling module installs forwarding rules in the network and the flow packets are allowed to pass. Clearly, the solution of delayed network flows cannot be used in high-performance networks where a huge amount of network traffic is passing. A solution that has incorporated the traffic detection procedures in the network controller is also available, it cannot respond to scalability requirements in high-performance networks [19]. Other work that has mentioned the use of traffic detection tool results to reroute the traffic flows in the network has also ignored the importance of all packets being processed in the network functions [20, 21].

Using DPI as a service is another work done in this area. The idea is to remove the deep packet inspection function from service functions like IDS or Firewall and deploy it in a separate service function. Other services in a service chain can leverage the metadata provided by DPI service function without running deep packet inspection procedures themselves. Actually, what this work states is that deep packet inspection is done in multiple service functions in a single chain and it is the main reason for processing delay in each service. By using the idea proposed in this work, deep packet inspection procedures are only executed once in every service chain where other services use the results of deep packet inspection analysis in that service [22].

2.2 Machine Learning Algorithms and Tools for Traffic Classification

Machine learning refers to a set of techniques and algorithms that are able to learn from data and make predictions on data. Such algorithms overcome the process of following strict rules and instructions provided in a program by making data-driven predictions and decisions, through building a model from training inputs. Machine learning techniques can be split into three categories of unsupervised learning, supervised learning, and semi-supervised learning based on the amount of labeled data used in their training process. Since only the use of supervised learning techniques is present in the current work, we will only focus on the works which have used supervised learning techniques for traffic classification.

A supervised machine learning algorithm uses a labeled dataset for training. Every instance in the dataset contains the same set of features which is mapped to a given known label [23]. The result of the training phase is a model which can be used to predict the label for unknown set of data with the same feature list. To the best of our knowledge, there are no works that have specifically leveraged a supervised machine learning algorithm in a service function chaining classifier node. Because of that, we only discuss some works that have used a supervised machine learning for the general idea of traffic classification.

One of the works that rely on behavioral features of internet applications which are present at the start of the flow and do not use the packet payload to extract any features, claims to have achieved 99.8 percent of accuracy. Considering a set of 248 flow features from the beginning of individual network flows with different observation windows size, it leverages feature selection algorithms to find the best subset of features. Finally, the feature subset is used to train a model and the model is used for classifying unknown flows. The authors believe that if the traffic classification system is going to work near real-time with a considerable throughput, an appropriate and small set of features must be selected from a small number of packets in a limited duration. This work has used C4.5 decision tree since it had a low complexity and finally, they believe using features extracted from 5 or 6 packets in the network can achieve the highest accuracy [24]. The complete 248 features are detailed in [25].

A solution comprised of signature-based and machine learning methods is also proposed that claims 99.7 percent of accuracy in traffic classification [26]. After labeling the dataset using Snort and using multiple techniques for extracting the best set of traffic features, the model is created using a multi-classifier and it is used for traffic classification. Although the authors have mentioned that they have not placed a constraint on feature extraction window size, it seems like the final feature list is dependent on the full trace of a network flow.

There is another work that has considered the use of statistical features of a flow for traffic classification using machine learning. This work combines unsupervised and supervised machine learning algorithms to provide a method for internet traffic classification. It extracts traffic features and clusters them using an unsupervised machine learning algorithm. The result is used as the training data in a supervised

machine learning algorithm and the output model is finally used for classifying unseen traffic flows. They claim that the method has an accuracy of 90 percent for classifying unseen traffic flows. A set of five and eight features is selected for unidirectional and bidirectional flows, respectively. Both feature lists can be provided at the end of a flow [27].

Another solution that has proposed a two-phased machine learning approach by using an unsupervised machine learning method for feature extraction and using a supervised machine learning algorithm for traffic classification has provided a feature list completely dependent on the full flow trace [28].

Taking into account the related works in the two sections related to service function chaining and machine learning techniques, the need for doing a similar classification work in service function chaining environments is felt. This work is allocated to the idea of using machine learning tools in the classifier node in service function chaining.

3 DESIGN AND IMPLEMENTATION

In this section, the proposed approach for traffic classification in service chained environments is discussed. Starting from the nature of malicious traffic and inability of the traditional approaches to classify this type of traffic, continuing with presenting the architecture where the proposed approach is used and finishing with the implementation details.

3.1 Design Challenges and Requirements

Traditionally, applications used proprietary port numbers on their server-side. Network traffic was mostly not encrypted and protocols like HTTPS were not so commonly used. For example, port number 80 was reserved for HTTP protocol and the protocol's traffic was plain text. Nowadays, traffic is mostly encrypted whether the application is malicious or not and port hopping is a common technique to evade network analysis devices like firewalls or IDS/IPS. Malicious applications are increasing every day and can change their traffic pattern easily.

In this situation, using simple port numbers or pattern matching approaches for detecting applications is inappropriate and the decision made on such knowledge cannot be trusted. So, deep packet inspection devices that utilize signature-based approaches that need heavy processing are not suitable for some scenarios. Besides, the nature of applications' traffic is always changing and signatures used in the aforementioned approaches need to be updated constantly and cannot be sufficient in all cases. This makes it challenging for service function chaining scenarios to use application characteristics in traffic classification. Therefore, malicious network flows may find the chance to stay out of sight of appropriate analyzing services by imitating benign applications' characteristics. Besides, service resources are wasted analyzing the wrong imitating flows. By mentioning signature-based or pattern-based schemes, this paper refers to the approaches that utilize a set of predefined

rules and signatures for traffic classification. Signatures are created by following the strict rules and instructions provided in an application, after analyzing the application's traffic. The classification of a network flow is performed by comparing the details of multiple packets in the flow with the predefined set of rules and signatures. Examples of the commonly used signatures are the server-side port numbers used by Internet applications and string patterns present in the applications' data.

Recent research studies have shown that machine learning approaches can be leveraged in the process of network traffic classification. Machine learning can be used to construct models that learn to decide whether a network flow is malicious or not, directly from the data and without any predefined rules [29]. Each Internet application's network traffic has some characteristics and behavioral patterns that can be used to predict other unknown network flows from the same application. By extracting historical or application-specific features from labeled network flows and feeding them into a supervised learning algorithm, models can be created to classify network flows.

Results show that some approaches have achieved better than 95 percent of accuracy by using machine learning algorithms [2, 24, 26, 28]. In this work, we have used some historical and application-specific features of network flows and created a model to be used in a classifier node in service function chaining scenarios. By mentioning machine learning-based schemes, the paper refers to the category of approaches that use supervised machine learning methods for training a classifier based on an application's available network traces. Instead of comparing details of a flow with predefined signatures, multiple flow characteristics are extracted and used for making a data-driven prediction about the flow.

One of the main goals in applying service function chaining in a network is to differentiate the services provided for network flows. As most machine learning-based approaches proposed for traffic classification are based on features extracted from a full flow, using machine learning methods in a service function chaining classifier is challenging. Delayed traffic classification can cause unnecessary burdens on unrelated services and changing a flow's path in the middle of its life results in an incomplete state in some service functions. Therefore, early traffic classification is a necessity in a network applying service function chaining.

3.2 Proposed Solution

There are a lot of features that can be extracted from a total flow. Lots of them need the flow to be finished to become available [25]. But this cannot be achieved in service function chaining scenarios where real-time classification is needed. In this case, flows must be classified as soon as possible to be analyzed in the corresponding service chains. Otherwise, since some stateful services may need to inspect all the packets in a flow, the classifier has to enable multiple services that belong to different classes for packet inspection. Technically, some services need to store state for each flow processed by them. When a new flow enters a service chaining network while it has not been mapped into a class, it needs to be injected into a default chain of

multiple services belonging to different classes. This way, the state stored in those services is complete even after classification. The natural result of using a default service chain is that the resources on some services are wasted on unrelated packets. Figure 2 illustrates two stages of a flow’s life in the network, before and after the classification. The packets in the flow are steered into the default service chain until the successful classification results are prepared for the flow. After that, only the services corresponding to the flow specific service chain are present.

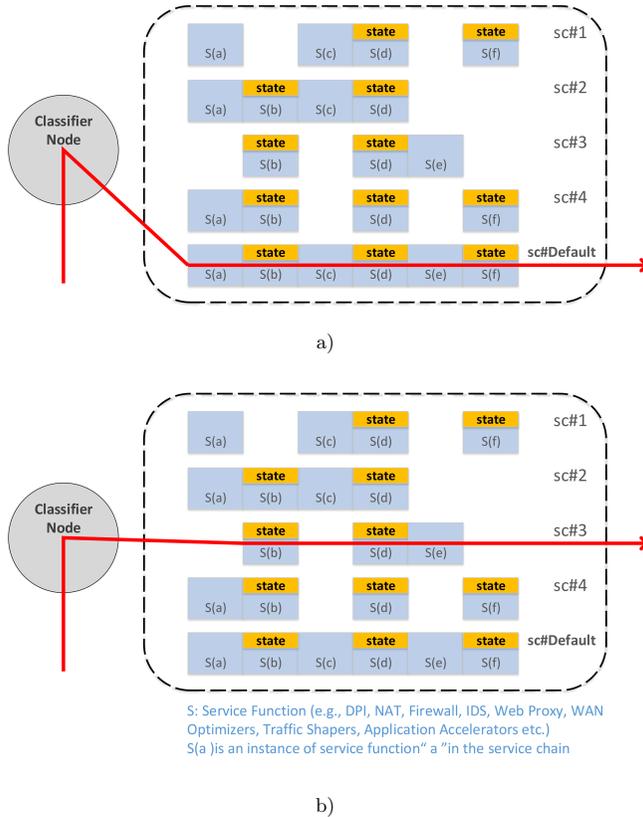


Figure 2. Illustration of a flow’s life in a network a) before and b) after flow classification

Some previous work has mentioned that considering a total of 5 or 6 packets from the start of flow can achieve the highest possible accuracy for traffic classification [24]. In service function chaining scenarios there is a tradeoff between the number of packets used for feature extraction and the load on the services belonging to each chain (as depicted in Figure 3). So, finding the best point for flow classification (feature extraction windows size) is the key to using machine learning algorithms in such environments.

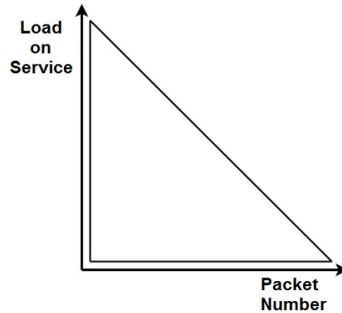


Figure 3. The tradeoff between the packet number and the load on services

In this work, the list of features used for traffic classification is categorized into historical and application-specific features. The list of historical features contains the 5-tuple characteristics of each flow which are the source and destination IP address, transport layer protocol, source, and destination port numbers. By historical it refers to the fact that network flows from specific client locations to specific server locations may happen again. It is true most of the time that a triad of server IP address, protocol, and the port number corresponds to a specific application server on the Internet. If we consider the habits of the clients in using the internet, the 5-tuple feature list may lead to a good prediction result for unknown flows.

In the application-specific feature list, besides the list of historical features, the features related to the application traffic patterns have been considered. Because of this, only the packets containing at least one byte of data have been used for feature extraction. By configuration, features from up to three data packets are used for traffic classification. The list of features extracted from each data packet is as follows:

- the size of transport layer payload (application data),
- the size of the captured data from network,
- the packet direction (from client or reverse),
- packet number in flow,
- TCP header push flag,
- packet entropy.

The size of the application data in the first packets may contain important information about the application. For example, this data is normally small for the first packet in an SSL flow. Besides, some malicious applications may try to segment data packets to a specific size or create messages of a specific size that can be considered as a pattern. To consider the size of other network layers' headers, the size of the captured data is also considered. In most client/server applications, the client sends the first data packet but this may be different in some other applications.

Also, it may be done by some malicious applications. This is the reason that data packet direction is considered. To consider the number of packets without data, the number of packets in the flow is also considered. TCP header push flag is also set in some applications' data packets which is also considered. Finally, packet entropy is also used as a feature that can specify whether the data is encrypted or not. This is an important feature for detecting some applications.

3.3 Technological and Implementation Details

The purpose of the experiments is to show how machine learning can help to classify network flows in a malicious environment where a good percent of the flows try to evade detection or analysis by imitating some famous applications' patterns. Also, to specify the best way to use machine learning algorithms in a service chaining environment. For this reason, nine application classes of Unknown, HTTP, DNS, HTTPS, FTP, Telnet, SSH, SMTP, and QUIC are considered. There are lots of malicious applications that try to evade detection by leveraging some of these famous protocols' characteristics (using their default port number for example). The classes have been chosen to somehow simulate the diverse nature of the Internet. All the experiments done by using machine learning algorithms are also compared with experiments done using a signature-based scheme where only port numbers are used for the classification of flows. In the end, some other experiments using pattern matching techniques are also presented.

CatBoost is a machine learning method based on gradient boosting over decision trees. Gradient boosting is a machine learning technique that can be used in classification problems by creating an ensemble of classifiers, typically decision trees. Gradient boosting trees have some properties that can perform excellently in network traffic classification [30]. In this work we have used CatBoost to create the model used for traffic classification. CatBoost performs well in the classification of traffic into several classes and it can be a great choice to be used in this work [31].

30 Gigabytes of network traffic has been captured from a company's network where a vast amount of malicious applications are active. The company provides Internet access for a group of people comprising scientists, students, and other people. The data was captured with a rate of 21 Mbps for more than four hours at peak time. After cleaning the data by removing TCP flows without SYN packet and removing flows with no data packets and also, removing duplicate flows, about 450 thousand flows are left. These flows are fed into two available DPI modules (one of them is a commercial DPI and the other one is open source nDPI [32]) and they are all labeled with classes mentioned above. Traffic flows in this data set belong to all of the classes with different proportions. This labeled data set is used as the train data to be fed into CatBoost and to create the model.

Two smaller traffic captures have also been prepared for running the experiments. One of them is captured in a network where a mix of multiple traffic classes is active and some malicious flows exist in it. The other one is captured in the same network where the training data was captured. They are named mixed traffic and

malicious traffic, respectively. By malicious, it refers to the fact that the rate of using deceptive techniques by applications present in this network is higher.

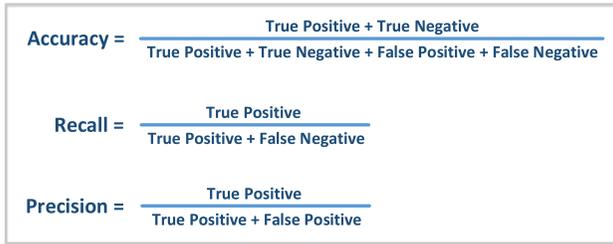
For each traffic capture the following set of experiments is performed. At first, the traffic is fed into a signature-based classifier where only port numbers are used to detect flows (the port number is used as a representative for signature-based methods). For each traffic class, its default port number(s) is used as the signature. After the classification of a flow, packets are sent into a service chain specific to that class where they are processed. Besides the signature-based classification, multiple tests are done using machine learning algorithms. One test is done only using historical features. After that, three tests are done with application-specific features extracted from one, two, or three data packets.

There is a difference between a flow classified in Unknown class and an unclassified flow. For a flow that is classified as unknown, we can say for sure that it does not belong to any other class. But, for an unclassified flow, nothing can be said about its class. Therefore, for flows classified to Unknown class no services are used and the traffic is passed unprocessed. The results of the tests are compared to two main parameters. The first one is the quality of classification of flows and the second one is the load on each service chain and consequently, on services.

4 EVALUATION RESULTS

In this section, we investigate the results of the tests mentioned above for the three metrics of accuracy, recall, and precision. Accuracy is about the number of flows predicted correctly in each test. Technically, it specifies the number of flows correctly predicted in a class and correctly predicted not in a class. Recall considers the number of flows in each class which is predicted correctly. In other words, it specifies the number of flows that are actually in a class and have been correctly predicted. Finally, precision specifies what percent of the positive predictions are correct. Figure 4 presents the formulas for calculating accuracy, recall and precision. As mentioned above, two data sets are prepared to run the tests. Before we run the tests, each data set is examined using the same DPI modules that we used for labeling the training data and these results are used for evaluating each metric. After that, each data set is once tested in a signature-based scenario and then in multiple machine learning scenarios. The average value of accuracy, recall, and precision are calculated for each type of test and when the result of all tests is examined, the average values are compared to find the best approach for traffic classification. In the end, the amount of traffic load on service chains is also investigated.

Figure 5 presents the results of the signature-based tests on two mixed and malicious data sets. Mixed data set refers to the traffic captured in an environment where a mix of benign and malicious applications are active with a broad list of applications and the malicious data set refers to the traffic captured in an environment where a vast amount of active applications are malicious. The results for mixed and



$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Figure 4. Formulas for calculating accuracy, recall, and precision

malicious traffics are distinguished using their colors. Results of accuracy, recall, and precision are all depicted in Figure 5.

As can be seen in Figure 5, all application classes except HTTPS and Unknown result a high accuracy of more than 90 percent. This means that with respect to those classes, most traffic flows are predicted correctly in a signature-based approach. The reason for a lower accuracy in HTTPS class is the fact that this port number is the most popular port number between malicious applications and all the applications that need to hide themselves from the sight of network analysis devices. The accuracy for the Unknown class is also low because the failure in classifying HTTPS traffic results in decreased accuracy for the Unknown class. A large number of Unknown flows must be classified in HTTPS class which is the result of evasive techniques employed in malicious applications.

Like the accuracy, we can see a high value of recall for most classes except the Unknown class. A low value of recall for class Unknown implies a high percentage of unknown flows which are incorrectly classified in other classes. The value presents the percentage of correctly predicted Unknown flows. Aside from the Unknown class, we see a lower recall for class HTTP compared with other classes. This implies that also for HTTP traffic there are some flows that have not used their default port number which is port number 80. This must be true since there are many HTTP servers running on port numbers 8080 or some other ports similar to this one. For HTTPS class we see a very high value of recall. The reason for this one is that almost all HTTPS traffic has used its default port number and nearly all of these flows are predicted correctly. Since some of the traffic classes were not present in the malicious traffic data, they are depicted without the value of recall.

By looking at the precision results in Figure 5, we will come up with more interesting conclusions. A low value of precision for a specific class is the result of some other flows incorrectly predicted in this class. As we can see, the class Unknown has a high value of precision while classes like HTTPS, SSH, or QUIC have resulted in lower values of precision. This means that a large number of Unknown flows have incorrectly been predicted as other classes. This is where we can see the track of malicious applications trying to mitigate other application properties. Classes

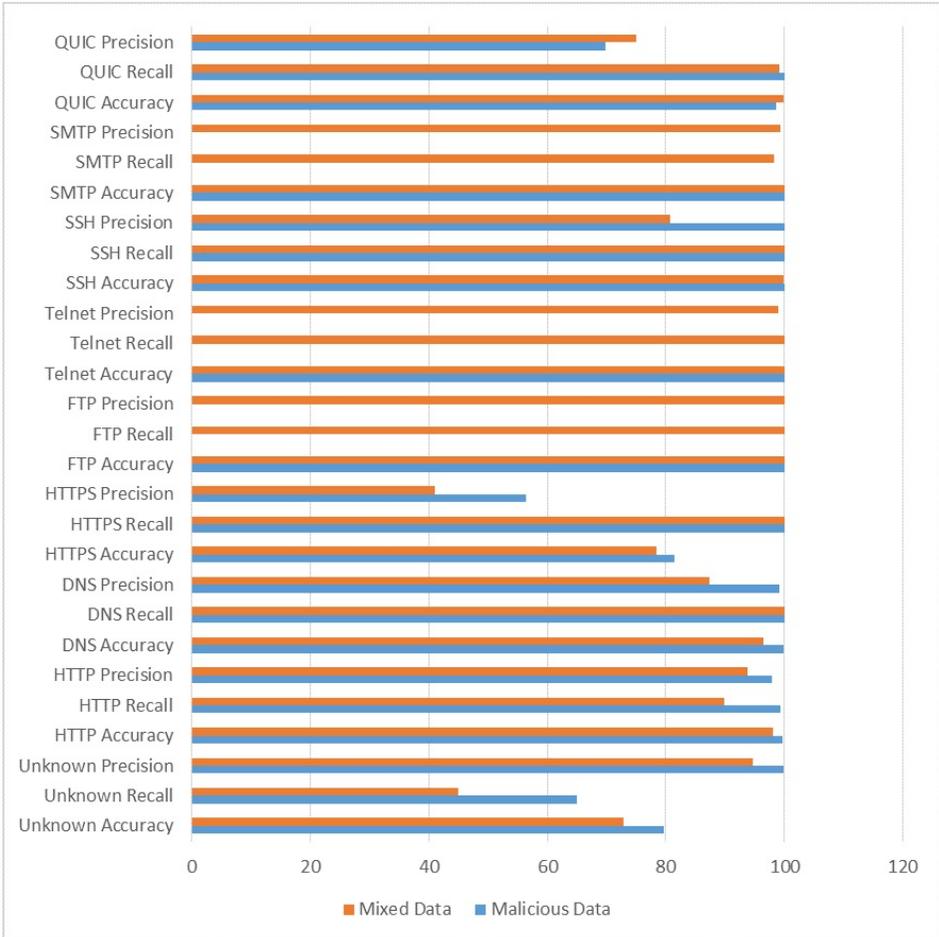


Figure 5. Results for signature-based scenario

without a precision value are also present since no traffic flow has used their default port number.

Besides the above discussion, we can see lower values in almost all metrics for the mixed data compared to the malicious data. This can emphasize the fact that the same trend of using evasive techniques is active in both networks.

After discussing the signature-based detection scheme, it is time to consider the results of machine learning-based experiments. Machine learning-based detection using a historical feature list (flow characteristics as mentioned before) is considered first. When speaking about a historical feature list, tracking the type of flows from specific clients to specific servers on the Internet is considered. When the model is created based on the type of flows originated from some clients to servers on

the Internet, it can predict unseen flows from the same clients to the same servers based on what it has seen historically. Based on the similarities of the new flows' characteristics with the flows used in model creation, new flows can be classified.

As mentioned in the previous section, the historical feature list is the 5-tuple of source/destination IP address, protocol, and source/destination port numbers extracted for each flow. Figure 6 compares the results for the machine learning-based classification using historical features in both mixed data and malicious data.

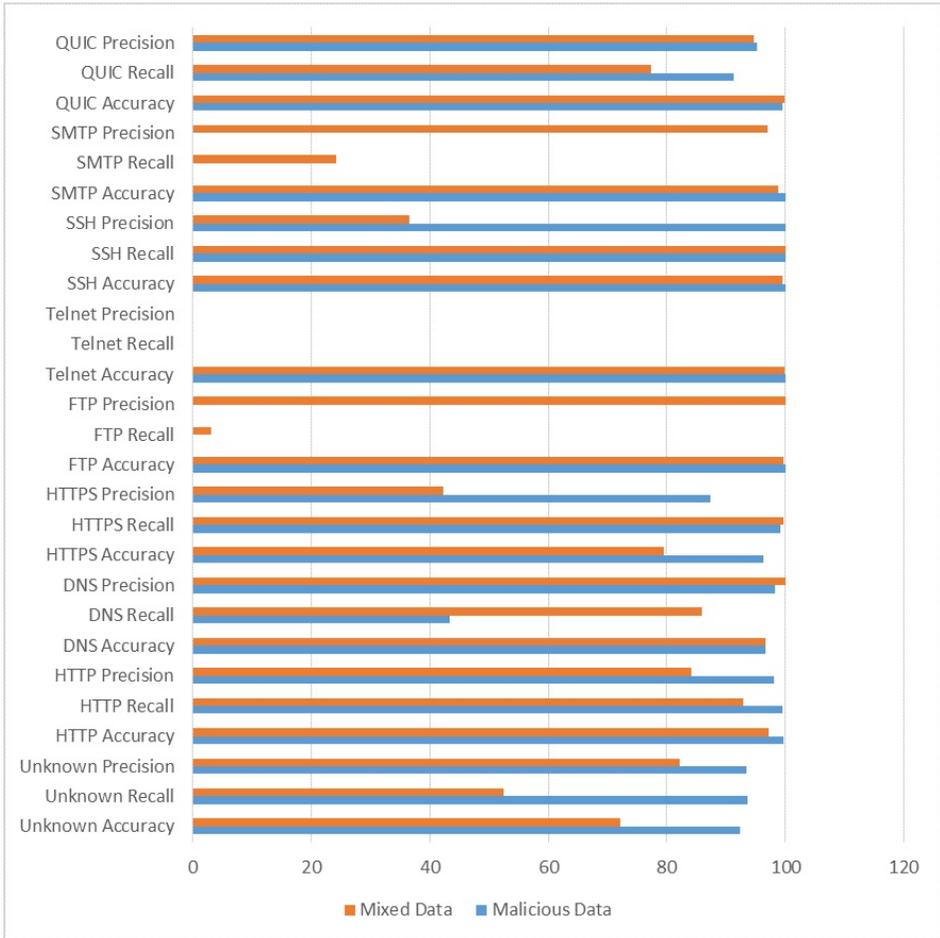


Figure 6. Results for historical features based scenario

Before we speak about the results presented in Figure 6, it should be reminded that the data set used to create the model for all the experiments was provided in a network with malicious traffic where lots of flows on famous port numbers are of

other unknown protocols. So, we can look for the effects of such a phenomenon in all experiments.

As it can be seen in Figure 6, the accuracy results for all traffic classes in malicious data are more than 90 percent while we see lower values of accuracy for some classes in mixed data. Since this experiment only considers the historical data for creating the model and the malicious data set was captured in the same networks as the training data, the higher accuracy for malicious data is justifiable. A high value of accuracy for the malicious traffic implies that the use of historical information in a network may be effective in application classification.

We can see a similar result comparing recall between mixed and malicious traffics. For most of the classes the value of recall for mixed traffic is less than the value of recall for malicious traffic. We believe this must also be the result of the difference between networks where training data and the mixed data are captured. As an exception, we can see that the DNS class has resulted a better recall for the mixed traffic data. Clearly, the list of DNS servers in all the networks all over the world is similar. This is the reason the amount of recall is not lower for mixed data. Also, the presence of a lot of malicious applications in the malicious network may cause a lower recall for that case since there may be some traffic flows trying to mitigate DNS traffic while they are not actually DNS.

Following the same pattern, we see lower precision for the test with the mixed data. Lower precision shows a lower percentage of correct positive predictions which can be justified considering the different networks mixed and malicious traffic where captured.

Considering the mixed data traffic, we can see that the precision for HTTPS and SSH classes is lower than the other classes. Comparing the result of precision in these classes in the mixed data with the same classes in the malicious traffic, we see much better results for the malicious data. This emphasizes the fact that the characteristics of these application classes are desirable with malicious applications and the lack of historical information about the traffic can result in lower precision when predicting application classes.

We saw the effects of using historical features in flow classification both for mixed and malicious traffic. Now we want to enter some features related to the actual application payload into the experiments. These features are listed in the previous section and they are categorized based on the number of data packets where the flow classification takes place. First, we look at the results where only the first data packet is considered. Figure 7 presents the results for this experiment.

It can be seen that adding some application payload features to the model can do the magic where the accuracy results for all classes in both data sets are more than 90 percent. This result presents the power of a good machine learning scheme when application-specific features are considered for classification. We can see that when application payload characteristics are added to the model, the difference between the accuracy for mixed data and malicious data is reduced.

Now that we have seen the result of using application-specific features in flow classification, it is time to increase the number of features by using more data

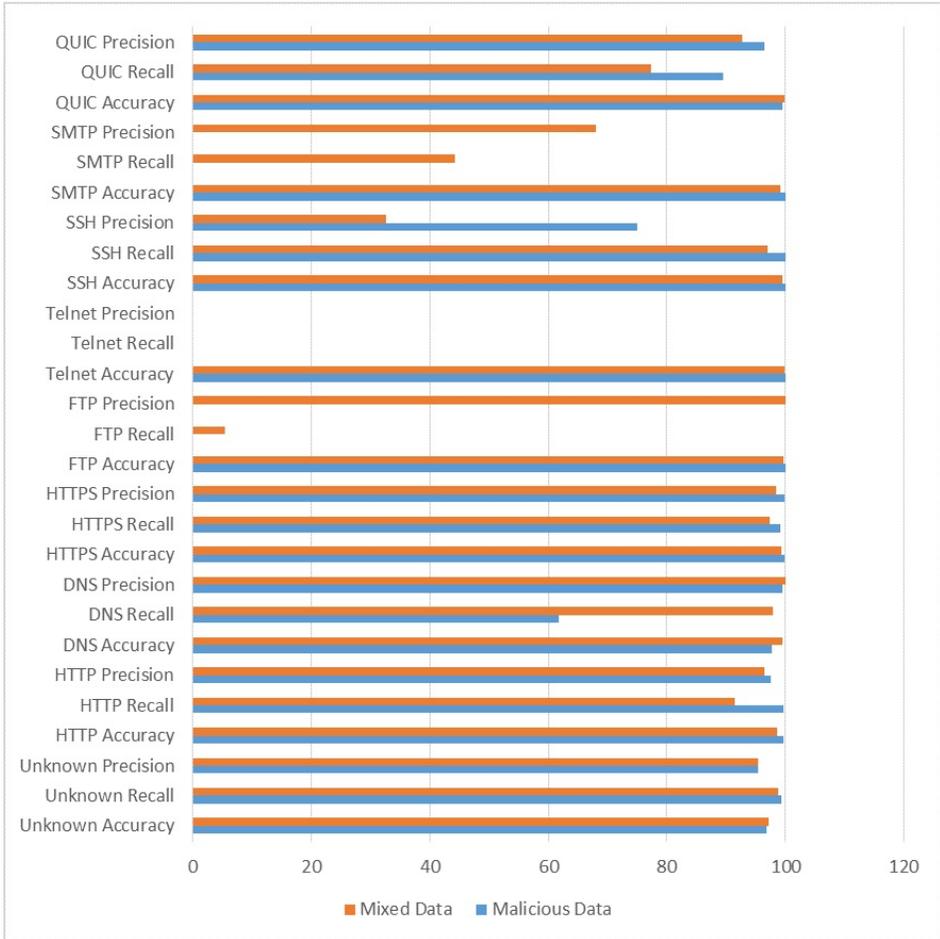


Figure 7. Results for application-specific features based scenario with one packet

packets in flow classification. Figure 8 presents the classification results when the classification is done as the second data packet is received in the flow and features related to the application payload are from the first and the second data packets.

Before speaking about the results presented in Figure 8, it should be noted that in any traffic data set, there may be some flows with only one data packet. So, when considering the second data packet for flow classification, some of the flows remain unclassified until they finish. The results presented in the above table are only considering the classified flows and unclassified ones are ignored. So, if for example it says that 10 percent of the flows are in HTTP class, it is referring to the set of classified flows and not the unclassified ones.

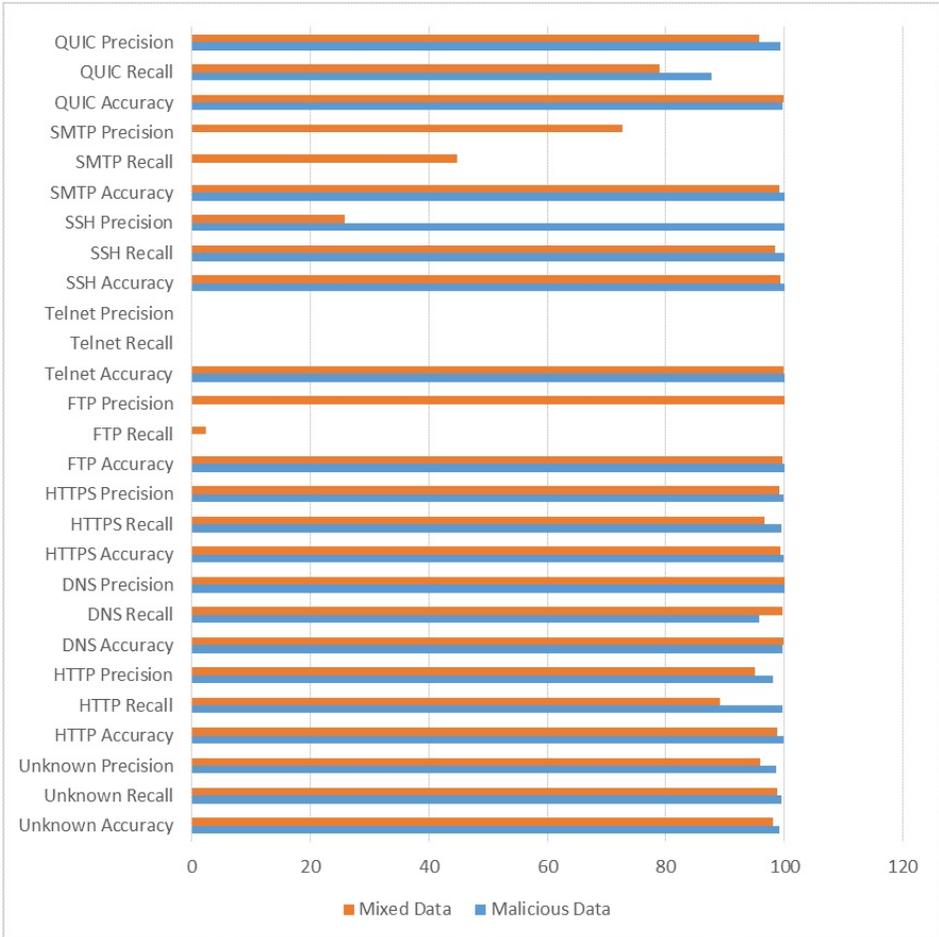


Figure 8. Results for application-specific features based scenario with two packets

Considering the results presented in Figure 8, we see the accuracy of more than 95 percent in all application classes. Compared to the results of experimenting with only one data packet, we can also see an increase in the value of recall for almost all classes. This result can also be observed for precision values. Some incompatible results for some of the classes (like the reduced recall for SSH) may be judged by the one packet flows which are omitted from the results in the current figure.

Finally, we want to consider one more data packet in the flow classification procedure. As the third data packet is added to the procedure, more features specific to that packet are added to the model. Besides, as mentioned above, the natural consequence of waiting for more packet until traffic classification, is the reduction in

the number of classified flows. Figure 9 presents the result of the experiment using three data packets where only the classified flows are considered to calculate the values.

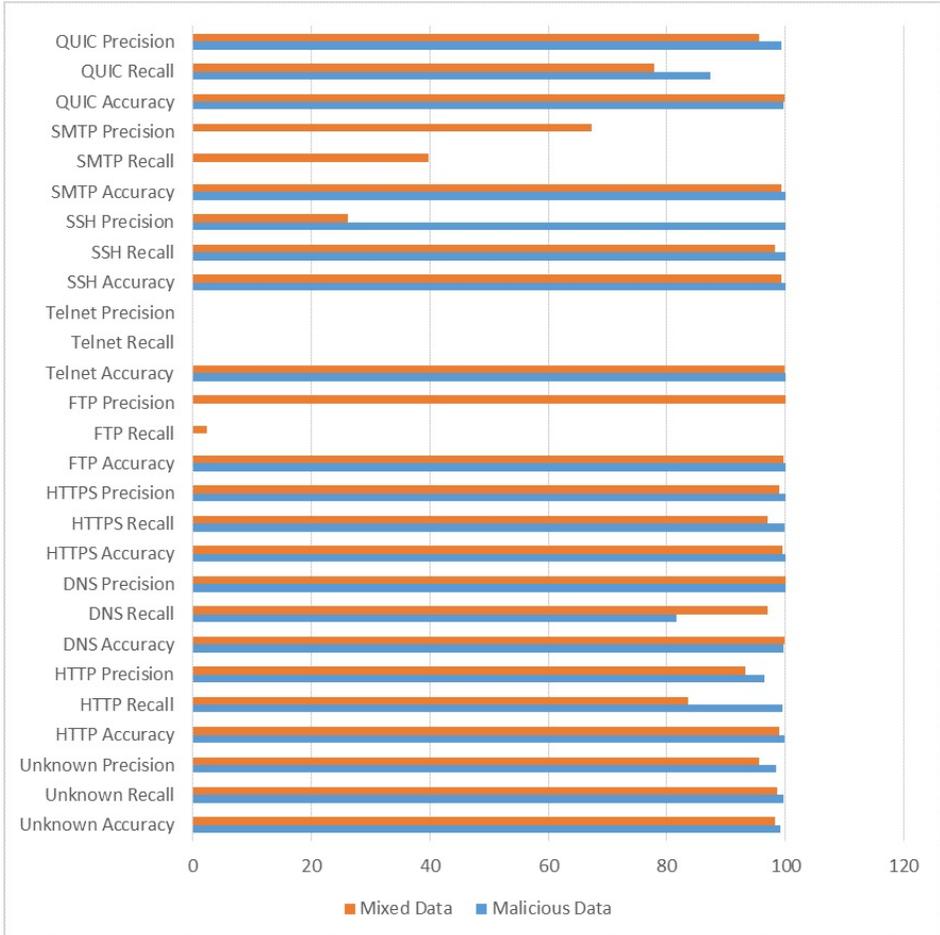


Figure 9. Results for application-specific features based scenario with three packets

Surprisingly, we see a reduction in calculated metrics for different classes compared to the previous tests. Although the reason may be that some valid flows with less than two data packets have been omitted from the results and the amount of true positive results has been reduced, it seems like using two data packets for feature extraction cannot result as good as approaches using fewer data packets. It should also be mentioned that commonly DNS flows only have two data packets consisting of a request packet and a response packet. Also, the existence of HTTP

flows with only one request packet and only one response packet is common. These sentences are mentioned to emphasize the previous statement.

Table 1 provides a numerical representation of all the results. A result of 100 percent for accuracy refers to the fact that all the corresponding flows in test data were correctly classified and no false positive or false negative cases were present. For recall, a result of 100 percent means that there were no false negative cases present and a value of 100 percent for precision means that there were no false positives. For recall and precision, a value of 0 percent means that no true positive cases were present and none of the positive cases were correctly classified.

		Signature-Based		Historical Features		Application Features #1		Application Features #2		Application Features #3	
		Malicious	Mixed	Malicious	Mixed	Malicious	Mixed	Malicious	Mixed	Malicious	Mixed
QUIC	Precision	69.87	75	95.22	94.68	96.53	92.71	99.34	95.74	99.29	95.65
	Recall	100	99.13	91.28	77.39	89.45	77.39	87.72	78.95	87.42	77.88
	Accuracy	98.54	99.92	99.55	99.93	99.53	99.93	99.66	99.94	99.67	99.94
SMTP	Precision	0	99.42	0	97.06	0	68.01	0	72.61	0	67.35
	Recall	0	98.27	0	24.19	0	44.28	0	44.81	0	39.76
	Accuracy	100	99.97	100	98.87	100	99.12	100	99.23	100	99.28
SSH	Precision	100	80.75	100	36.62	75	32.64	100	25.74	100	26.15
	Recall	100	100	100	100	100	96.92	100	98.37	100	98.28
	Accuracy	100	99.93	100	99.51	99.98	99.43	100	99.24	100	99.3
Telnet	Precision	0	98.89	0	0	0	0	0	0	0	0
	Recall	0	100	0	0	0	0	0	0	0	0
	Accuracy	100	100	100	99.81	100	99.81	100	99.81	100	99.81
FTP	Precision	0	100	0	100	0	100	0	100	0	100
	Recall	0	100	0	3.1	0	5.43	0	2.34	0	2.34
	Accuracy	100	100	100	99.73	100	99.74	100	99.73	100	99.73
HTTPS	Precision	56.32	41.09	87.42	42.27	99.93	98.46	99.93	99.07	100	98.97
	Recall	100	100	99.16	99.6	99.16	97.41	99.54	96.68	99.93	97.05
	Accuracy	81.39	78.4	96.37	79.44	99.78	99.38	99.88	99.4	99.98	99.45
DNS	Precision	99.2	87.29	98.22	99.97	99.58	99.97	100	99.99	100	100
	Recall	100	99.98	43.34	85.9	61.84	97.94	95.83	99.67	81.55	97.06
	Accuracy	99.95	96.45	96.58	96.56	97.73	99.49	99.77	99.94	99.7	99.95
HTTP	Precision	97.98	93.8	97.99	84.15	97.5	96.53	98.04	95.05	96.53	93.22
	Recall	99.32	89.76	99.49	92.93	99.66	91.43	99.64	89.22	99.49	83.53
	Accuracy	99.75	98.1	99.77	97.12	99.74	98.62	99.8	98.85	99.88	98.97
Unknown	Precision	99.83	94.59	93.47	82.12	95.37	95.48	98.58	95.9	98.4	95.48
	Recall	64.95	44.94	93.64	52.48	99.34	98.73	99.58	98.73	99.68	98.66
	Accuracy	79.63	72.8	92.27	72.08	96.76	97.17	99.1	98.09	99.24	98.23

Table 1. Results of precision, recall, and accuracy for all scenarios

After discussing the results of traffic prediction in each test for different application classes, it is time to compare different approaches for traffic classification and choosing the most suitable one. The average value for each metric is calculated for all tests and a separate diagram is created for each metric. Figure 10, Figure 11 and Figure 12 present the results of accuracy, recall and precision, respectively.

We can see in Figure 10 that for the malicious traffic, accuracy has been increased by each new test where the lowest accuracy has been experienced in the signature-based test and the highest one is experienced when three data packets are used for feature extraction. The results for the mixed data are similar to the results of malicious traffic as the highest accuracy is achieved when three data packets are used for feature extraction but they have an interesting difference. As we can see, the accuracy has been reduced when using the historical machine learning approach for mixed data. As mentioned previously, the reason for this phenomenon is that the network where malicious traffic has been captured is the same as the network where the training data was captured. Therefore, the flow characteristics may not follow the same pattern historically. It should also be mentioned that the accuracy for all tests using the application-specific features has reached 99 percent and the difference between tests with two data packets and three data packets is negligible.

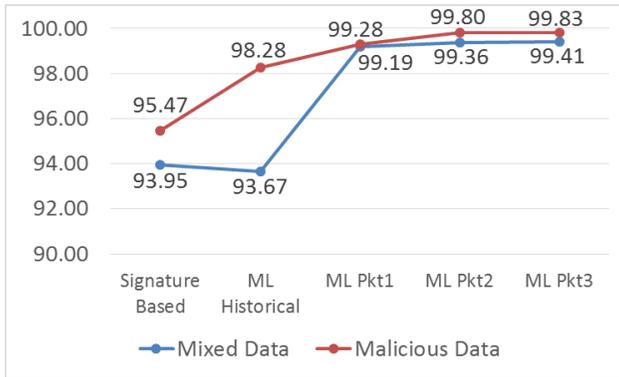


Figure 10. Comparing accuracy for malicious and mixed data

As we can see in Figure 11, the amount of recall is almost the same for both data sets when the signature based approach is utilized at first. For the mixed data we see a reduction in recall value as the tests proceed with the lowest value experienced with historical features approach. For the malicious data, the highest recall value is experienced when two data packets are used for feature extraction and it reduces when the number of packets is increased to three. We believe the reason that the value of recall for malicious data is more than the recall for mixed data is the choice of feature list and the fact that malicious data is captured in the same network as the training data.

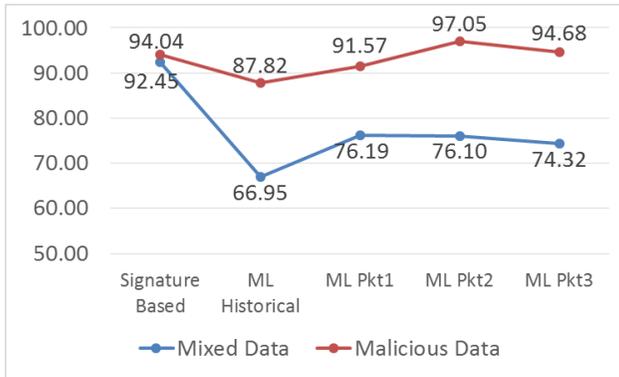


Figure 11. Comparing recall for malicious and mixed data

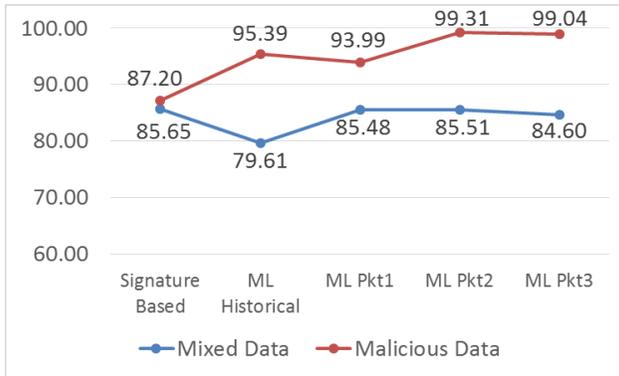


Figure 12. Comparing precision for malicious and mixed data

Like the results presented for recall, the precision value starts almost the same for both data types when the signature-based approach is tested, the lowest value for the mixed data is when historical features machine learning approach is tested and the highest value for the malicious data is seen when two data packets are used for testing the application-specific machine learning approach. Again, it seems like the fact that malicious data is captured in the same network as the training data has had a positive effect on the results.

The reason that signature-based tests for both malicious and mixed data are almost the same is that no feature related to the network environment is used in those tests. Based on the above experiments, we can see that when the training data and the test data are both captured in the same network, we can get the best results with the application-specific features approach when two data packets are used for feature extraction. Considering the case of service function chaining, this theory

may not be the best choice. To make sure, we need to investigate how different approaches affect the load in the network.

Now that we have discussed the classification results in multiple experiments, it is time to discuss about the load imposed on service chains prepared for application classes. When the classification of a flow is done on one of its packets, that packet and the packets after that will all be injected into the chain prepared for the corresponding class. Flows classified in Unknown protocols class are not processed in any chain and their packets are passed unprocessed. If classification is postponed to data packets after the first one, packets in that flow must be processed in all services (in all chains) so the flow receives a complete service and the state created in those services is not incomplete.

Figure 13 presents the amount of load on all service chains in all experiments for mixed and malicious data sets. The load is calculated as the percentage of all data packets (where the actual processing is done) that have entered all service chains over the total number of data packets.

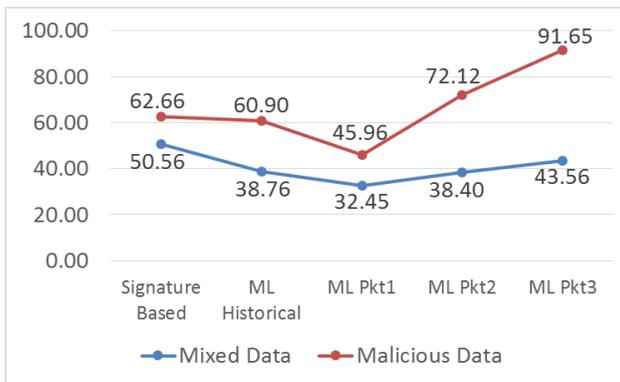


Figure 13. Comparison of load on service chains for mixed and malicious data

What can be seen in the above chart is that as we track the experiments until the one using the application-specific features on the first data packet, the amount of load on all service chains is decreased and gets more meaningful. The reason is that flow classification is becoming more accurate and the resources are not wasted for processing unrelated packets. But as we go on to the experiments using the second and the third data packets, we see an increase in the load imposed on service chains. The reason for this increase is that as the classification is postponed to future data packets, unclassified flows must be processed in all service chains so that the services in the correct service chain do not miss any packets. So, while adding delay to classification time may help in better classification, it results in more load on service chains which is contrary to the main purpose of service chaining idea.

The total trend of the diagram is the same for mixed and malicious traffic data but the load on service chains is higher when testing with the malicious data in contrast with the mixed. The reason for the higher load when using malicious data is not necessarily the result of correctness in classification. A greater proportion of Unknown traffic may be present in the mixed data and when a flow is classified as Unknown, it will not be processed in any of the service chains. So, the variation of traffic classes in a traffic data set can impact the load on service chains.

Before we finish this part and in order to make more powerful results, we have made another comparison between another signature-based classification method and the machine learning methods. This signature-based method is pattern matching. We have labeled the malicious data set with three classes of Unknown, Google, and Instagram and used the same data set in multiple experiments to see how machine learning can be compared with other signature-based methods. Figure 14 presents the results of accuracy, recall, and precision from five experiments using pattern matching and machine learning methods. As we can see, the accuracy is around 95 percent in all the experiments which means that if we consider the correctness of all predictions on network flows, a very good result has been provided. Even if we consider recall, we can see that pattern matching has achieved a value as good as the best machine learning methods.

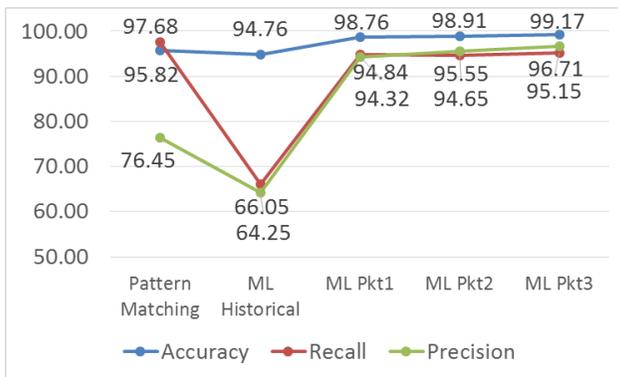


Figure 14. Comparing accuracy, recall and precision when using machine learning methods against pattern matching

The most interesting part of the result is about precision where the pattern matching method presents weaker results compared to some other methods. The DPI module used for labeling the data set leveraged the same pattern matching procedure, but after the application protocol is totally parsed. In other words, the module has intelligently compared each signature pattern with the most appropriate location in packet payload. There are some applications that try to evade detection by using signatures of famous applications in their payload. For example,

a malicious application which uses HTTPS protocol can mention Google in subject alternative name section in its certificate. To avoid the complexities of protocol parsing in DPIs, the pattern matching method used in current experiment has blindly searched for each pattern in the whole packet payload. The reason for a low precision is that there are some flows that have been classified in one of these classes by mistake.

5 CONCLUSIONS

The current work focused on using machine learning techniques in a service function chaining classifier node by leveraging a method named CatBoost. Taking into account the new evasive techniques used by malicious applications active on the Internet, the traditional classification methods like signature-based detection are believed to be ineffective. There needs to be a way for classifying the unknown application flows without following the strict instructions provided by the signatures for known applications. Machine learning can help to classify unknown application flows only by considering the data itself and without the use of application signatures.

By selecting a list of historical and application-specific features and using a labeled data set, we used CatBoost to create models that are used for classifications of unseen network flows in a service function chaining environment. The classified flows were forwarded into a predefined service chain composed of specific services to the selected class where they are processed according to the actual application type. Considering the quality of traffic classification and the load imposed on services, there exists a challenge for selecting the number of packets for extracting the application-specific features. As more packets are considered before the classification of a network flow, more features are available, and possibly, higher quality is achieved.

In our experiments, we have found out that taking into account the service function chaining problem statement, the best number of data packets considered for feature extraction is one. Although selecting one more data packet can slightly increase the classification quality, we believe that the amount of load saved by using only one packet can have more benefits in a service chaining environment. For example, a lower load on service chains can make space for more throughput in the network. This is while the difference in classification quality between methods using one and two data packets is not much at least in respect to accuracy. Also, based on the selection of features extracted from each flow, it is clear that the classification quality will increase if training data and test data are captured in the same network.

Finally, it can be said that using a machine learning method with proper selection of its properties can be a good replacement for traditional signature-based classification and to avoid the expensive, complex procedures of deep packet inspection modules.

REFERENCES

- [1] MCGREGOR, A.—HALL, M.—LORIER, P.—BRUNSKILL, J.: Flow Clustering Using Machine Learning Techniques. In: Barakat, C., Pratt, I. (Eds.): *Passive and Active Network Measurement (PAM 2004)*. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3015, 2004, pp. 205–214, doi: 10.1007/978-3-540-24668-8_21.
- [2] MOORE, A. W.—ZUEV, D.: Internet Traffic Classification Using Bayesian Analysis Techniques. 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'05), Banff, Alberta, Canada, June 2005. ACM SIGMETRICS Performance Evaluation Review, Vol. 33, 2005, No. 1, pp. 50–60, doi: 10.1145/1071690.1064220.
- [3] ERTAM, F.—AVCI, E.: A New Approach for Internet Traffic Classification: GA-WK-ELM. *Measurement*, Vol. 95, 2017, pp. 135–142, doi: 10.1016/j.measurement.2016.10.001.
- [4] ACETO, G.—CIUNZO, D.—MONTIERI, A.—PESCAPÉ, A.: Multi-Classification Approaches for Classifying Mobile App Traffic. *Journal of Network and Computer Applications*, Vol. 103, 2018, pp. 131–145, doi: 10.1016/j.jnca.2017.11.007.
- [5] ACETO, G.—CIUNZO, D.—MONTIERI, A.—PESCAPÉ, A.: Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges. *IEEE Transactions on Network and Service Management*, Vol. 16, 2019, No. 2, pp. 445–458, doi: 10.1109/TNSM.2019.2899085.
- [6] LOTFOLLAHI, M.—SIAVOSHANI, M. J.—SHIRALI HOSSEIN ZADE, R.—SABERIAN, M.: Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning. *Software Computing*, Vol. 24, 2020, No. 3, pp. 1999–2012, doi: 10.1007/s00500-019-04030-2.
- [7] ERMAN, J.—MAHANTI, A.—ARLITT, M.: QRP05-4: Internet Traffic Identification Using Machine Learning. *IEEE Globecom 2006*, San Francisco, CA, USA, 2006, pp. 1–6, doi: 10.1109/GLOCOM.2006.443.
- [8] CatBoost Web Site. Available at: <https://tech.yandex.com/catboost/>.
- [9] GHAZNAVI, M.—SHAHRIAR, N.—KAMALI, S.—AHMED, R.—BOUTABA, R.: Distributed Service Function Chaining. *IEEE Journal on Selected Areas in Communications*, Vol. 35, 2017, No. 11, pp. 2479–2489, doi: 10.1109/JSAC.2017.2760178.
- [10] QUINN, P.—NADEAU, T.: Problem Statement for Service Function Chaining. IETF, 2015, available at: <http://tools.ietf.org/html/rfc7498.html>.
- [11] BHAMARE, D.—JAIN, R.—SAMAKA, M.—ERBAD, A.: A Survey on Service Function Chaining. *Journal of Network and Computer Applications*, Vol. 75, 2016, pp. 138–155, doi: 10.1016/j.jnca.2016.09.001.
- [12] HALPERN, J.—PIGNATARO, C.: Service Function Chaining (SFC) Architecture. IETF, 2015, available at: <https://www.rfc-editor.org/rfc/pdf/rfc/rfc7665.txt.pdf>.
- [13] LI, H.—WU, Q.—HUANG, O.—BOUCADAIR, M. et al.: Service Function Chaining (SFC) Control Plane Components and Requirements. IETF, 2016, available at: <https://tools.ietf.org/pdf/draft-ietf-sfc-control-plane-03.pdf>.

- [14] Using Service Classification to Build an Application-Aware NFV Infrastructure for Virtual CPE Services. 2015, available at: <https://embedded.communities.intel.com/docs/D0C-8603>.
- [15] HANTOUTI, H.—BENAMAR, N.: Analysis of Service Function Chaining Forwarding Methods, Advances and Drawbacks. The Fourth International Workshop on RFID and Adaptive Wireless Sensor Networks (RAWSN 2016), Marrakesh, Morocco, May 2016.
- [16] ZHANG, Y.—BEHESHTI, N.—BELIVEAU, L.—LEFEBVRE, G.—MANGHIRMALANI, R.—MISHRA, R.—PATNEYT, R.—SHIRAZIPOUR, M.—SUBRAHMANNIAM, R.—TRUCHAN, C.—TATIPAMULA, M.: StEERING: A Software-Defined Networking for Inline Service Chaining. 2013 21st IEEE International Conference on Network Protocols (ICNP), Goettingen, Germany, 2013, pp. 1–10, doi: 10.1109/ICNP.2013.6733615.
- [17] QAZI, Z. A.—TU, C.-C.—CHIANG, L.—MIAO, R.—SEKAR, V.—YU, M.: SIMPLE-fying Middlebox Policy Enforcement Using SDN. ACM SIGCOMM Computer Communication Review, Vol. 43, 2013, No. 4, pp. 27–38, doi: 10.1145/2534169.2486022.
- [18] JEONG, S.—LEE, D.—HYUN, J.—LI, J.—HONG, J. W.-K.: Application-Aware Traffic Engineering in Software-Defined Network. 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), Seoul, South Korea, 2017, pp. 315–318, doi: 10.1109/APNOMS.2017.8094144.
- [19] LI, G.—DONG, M.—OTA, K.—WU, J.—LI, J.—YE, T.: Deep Packet Inspection Based Application-Aware Traffic Control for Software Defined Networks. 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 2016, pp. 1–6, doi: 10.1109/GLOCOM.2016.7841721.
- [20] LI, G.—ZHOU, H.—LI, G.—FENG, B.: Application-Aware and Dynamic Security Function Chaining for Mobile Networks. Journal of Internet Services and Information Security (JISIS), Vol. 7, 2017, No. 4, pp. 21–34, doi: 10.22667/JISIS.2017.11.30.021.
- [21] LI, G.—ZHOU, H.—FENG, B.—LI, G.: Context-Aware Service Function Chaining and Its Cost-Effective Orchestration in Multi-Domain Networks. IEEE Access, Vol. 6, 2018, pp. 34976–34991, doi: 10.1109/ACCESS.2018.2848266.
- [22] BREMLER-BARR, A.—HARCHOL, Y.—HAY, D.—KORAL, Y.: Deep Packet Inspection as a Service. Proceedings of the 10th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT '14), Sydney, Australia, 2014, pp. 271–282, doi: 10.1145/2674005.2674984.
- [23] KOTSIANTIS, S. B.: Supervised Machine Learning: A Review of Classification Techniques. Informatica, Vol. 31, 2007, No. 3, pp. 249–268.
- [24] LI, W.—MOORE, A. W.: A Machine Learning Approach for Efficient Traffic Classification. 2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Istanbul, Turkey, 2007, pp. 310–317, doi: 10.1109/MASCOTS.2007.2.

- [25] MOORE, A. W.—ZUEV, D.—CROGAN, M.: Discriminators for Use in Flow-Based Classification. September 2005, available at: <https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/5050/RR-05-13.pdf>.
- [26] JAMIL, H. A.—ALI, B. M.—HAMDAN, M.—OSMAN, A. E.: Online P2P Internet Traffic Classification and Mitigation Based on Snort and ML. *European Journal of Engineering Research and Science*, Vol. 4, 2019, No. 10, pp. 131–137, doi: 10.24018/ejers.2019.4.10.1534.
- [27] VLĂDUȚU, A.—COMĂNECI, D.—DOBRE, C.: Internet Traffic Classification Based on Flows' Statistical Properties with Machine Learning. *International Journal of Network Management*, Vol. 27, 2017, No. 3, Art.No. e1929, 14 pp., doi: 10.1002/nem.1929.
- [28] BAKHSHI, T.—GHITA, B.: On Internet Traffic Classification: A Two-Phased Machine Learning Approach. *Journal of Computer Networks and Communications*, Vol. 2016, 2016, Art.No. 2048302, 21 pp., doi: 10.1155/2016/2048302.
- [29] WANG, M.—CUI, Y.—WANG, X.—XIAO, S.—JIANG, J.: Machine Learning for Networking: Workflow, Advances and Opportunities. *IEEE Network*, Vol. 32, 2018, No. 2, pp. 92–99, doi: 10.1109/MNET.2017.1700200.
- [30] BAGUI, S.—FANG, X.—KALAIMANNAN, E.—BAGUI, S. C.—SHEEHAN, J.: Comparison of Machine-Learning Algorithms for Classification of VPN Network Traffic Flow Using Time-Related Features. *Journal of Cyber Security Technology*, Vol. 1, 2017, No. 2, pp. 108–126, doi: 10.1080/23742917.2017.1321891.
- [31] BAKHAREVA, N.—SHUKHMAN, A.—MATVEEV, A.—POLEZHAEV, P.—USHAKOV, Y.—LEGASHEV, L.: Attack Detection in Enterprise Networks by Machine Learning Methods. 2019 International Russian Automation Conference (RusAutoCon), Sochi, Russia, 2019, pp. 1–6, doi: 10.1109/RUSAUTOCON.2019.8867696.
- [32] nDPI: Open and Extensible LGPLv3 Deep Packet Inspection Library. Ntop, available at: <https://www.ntop.org/products/deep-packet-inspection/ndpi/>.



Habib Allah KHOSRAVI received his B.Sc. degree in information technology engineering from the University of Mazandaran in 2012. He received his M.Sc. degree in computer networks from the University of Isfahan in 2014 and is currently a Ph.D. student in the University of Qom. His research interests include computer networks, network security and distributed systems.



Yaghoub FARJAMI is Professor of computer science in the University of Qom. He received his Ph.D. degree from the Sharif University of Technology in Tehran, Iran. His research interests include computer security, artificial intelligence and machine learning, cryptocurrencies, etc.