RESEARCH ARTICLE OPEN ACCESS

Packet Classification using Support Vector Machines with String Kernels

Sarthak Munshi
Department Of Computer Engineering Pune Institute Of Computer Technology Savitribai Phule Pune University (Formerly University Of Pune) Pune , India

ABSTRACT:
Since the inception of internet many methods have been devised to keep untrusted and malicious packets away from a user's system . The traffic / packet classification can be used
as an important tool to detect intrusion in the system. Using Machine Learning as an efficient statistical based approach for classifying packets is a novel method in practice today . This paper emphasizes upon using an advanced string kernel method within a support vector machine to classify packets .
There exists a paper related to a similar problem using Machine Learning [2]. But the researches mentioned in their paper are not up-to date and doesn't account for modern day
string kernels that are much more efficient . My work extends their research by introducing different approaches to classify encrypted / unencrypted traffic / packets .
Keywords: Support Vector Machines, Packets, Malicious Be- haviour, String Kernels, Intrusion Detection.

I. INTRODUCTION

The number of Internet users are increasing at an ever fast rate . This has increased the flow of packets over multiple networks . More than 10% of the packets are malicious, duplicates or datagram-free . Such packets also create a lot of stress on the network . Several internet applications such as peer-to-peer-based software and VOIP have also led to fast increments in Internet traffic.

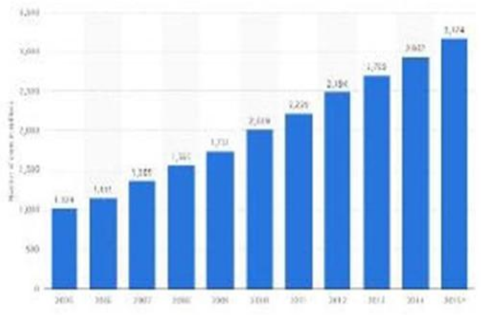


Fig. 1: Graph depicting yearly rise of internet users .

Such load creation on the network can be reduced to some extent if the redundant and malicious packets are eliminated from the flow . Given the ballistic amount of cyber attacks occurring , it becomes an issue of high significance to eliminate such packets from the system . Packets are the lowest abstrac-
tion of a network flow and analyzing it gives the best results [1]. Further, we will discuss about various existing packet

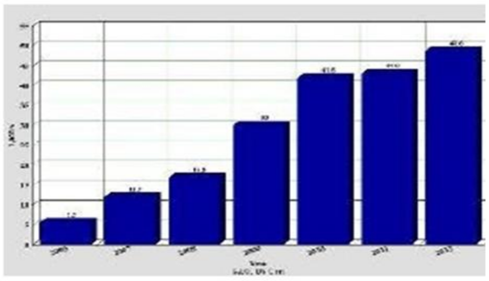


Fig. 2: Graph depicting yearly rise of cyber attacks .

classification techniques, how machine learning is useful , support vector machines, and kernels

II. PACKET CLASSIFICATION TECHNIQUES

A. Port-based Classification

Port based classification refers to classifying traffic based on TCP / UDP port numbers . For example, HTTP uses port 80, POP3 uses port 110, and SMTP uses port 25 and thus, we
can set up rules to classify the applications that are assigned to the port numbers. Moore and Papagiannaki claimed the accuracy of port-based classification is around 70% during their experiment [8]. Hav-ing static port numbers makes it easier to identify application specific traffic. Although this method worked successfully for quite some time, many issues came up in the long run. Applica-
tions tried to avoid firewalls by hiding their port numbers. Also, applications started using dynamic port numbers and servers on the same IP address would use port numbers not standardized by the IANA.

B. Payload-based Classification
Payload-based classification techniques classify the packets based on the signature in the packet payload. The signature is a unique string in the payload that distinguishes the target
packets from other traffic packets. We can set up rules to ana- lyze the packet payload to match the communication patterns in the payload in order to classify the application. Existing
methods classify the packets with a 100% accuracy rate , which is quite commendable . The issues with this method are that users may encrypt the payload to avoid detection, and also in
some countries doing payload inspection is forbidden in order to protect user privacy.
Furthermore, the classifier will experience heavy operational load because it needs to constantly update the application signature to make sure it contains the signature of all the latest applications.

C. Statistical-based Classification
This approach uses a set of sample traffic trace to train the classification engine to identify future traffic based on the application flow behaviors, such as packet length, inter- packet arrival time, TCP and IP flags, and checksum. The main aim here is to classify traffic with similar patterns into groups or application specific data. Accuracy of classifying encrypted traffic using a statistical-based approach is relatively low, varying from 76% to 86% with false positive rate between 0% and 8% when considering certain parameters and settings [9, 10]. Here, machine learning is preferred because it can automatically create the signatures for the applications and identify the application in the future traffic flow. Another reason is that machine learning has the ability to automatically select the most appropriate features to create the signature. A set of procedures in a typical machine learning problem are as follows :
1) Establish the features significant to the traffic .
2) Shortlist an application type based on data .
3) Use application traces to train the classification engine to generate rules, and uses the machine learning algorithms to classify future traffic. Machine learning based approaches can give the best results if the right technique and right parameters are taken into account. Approaches discussed in this paper substantiate this claim even further .

Support Vector Machines And Kernels
A. Support Vector Machines (SVM)
Support vector machine (SVM) is known as one of the most successful classification algorithms for data mining. It is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we
plot each data item as a point in n-dimensional space (where n is number of features we have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes.
Support Vectors are simply the co-ordinates ofindividual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane / line).

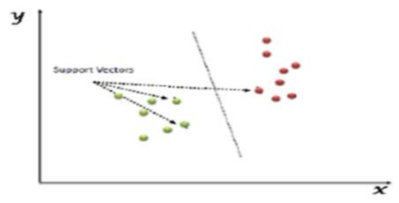


Fig. 3: Support Vectors on basic dataset segregation

In Fig.3, we see that two different sets of data points are linearly separable in one-dimensional space . In cases, where the data points are not linearly separable, the concept of kernel pops in .

B. Kernels
A kernel is a function that enables the support vector machine to linearly separate the data in a higher-dimensional space . Using a kernel function is similar to adding a trivial feature to the already existing set of features only except no feature is added or removed. Kernel function K(x, y) can be expressed as a dot product in a high dimensional space. If the arguments to the kernel are in a measurable space X,
and if the kernel is positive semi-definite for any finite subset
{x1, ..., xn} of X and subset {c1, ..., cn} of objects
K(xi, xj)cicj ≥ 0,
i,j

then there must exist a function φ(x) whose range is in an inner product space of possibly high dimension, such that K(x, y) = φ(x)φ(y). Note that the kernel computes this inner product by implicitly mapping the examples to the feature space. The explicit extraction of features in a feature space generally has very high computational cost but a kernel function provides a way to handle this problem.
This non-linear algorithm is equivalent to the linear algorithm operating in the range space of φ. However, because kernels are used, the φ function is never explicitly computed. The kernel
representation of data amounts to a nonlinear projection of data into a high-dimensional space where it is easier to separate into classes [11].

Some of the kernels used with SVM are Polynomial Kernel, Gaussian Radial Basis Kernel, Hyberbolic Tangent Kernel. All of these kernels operate with numerical
data. For our purpose is necessary to use string kernels which are described in following section.

IV. STRING KERNELS

Wikipedia defines a string kernel as, ”a kernel function that operates on strings, i.e. finite sequences of symbols that need not be of the same length” .
A string x is defined as x ∈ Σ∗, where Σ is the Kleene closure over all symbols from the finite alphabet Σ [12]. Using this, we can define similarity measures for using kernel machines on strings. There are two major types of string kernels known. The first ones are directly defined on strings and second ones
are defined on generative models [12]. Some of string kernels are explained as follows .

A. N-Gram Kernel

N-grams transform documents into high dimensional fea- ture vectors where each feature corresponds to a sequence of n consecutive characters [13].
Example : 3 − grams = ios jko ioo i o io The feature space associated with n-gram kernel is defined as

$$K(S,T) = < \phi^n(S), \phi^n(T) > = \sum_{u \in \Sigma^n} \Phi_{nu}(S)\Phi_{nu}(T)$$

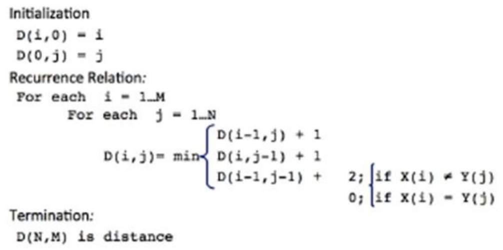
where Φ represents the surroundings of substring u in S(string) such that v1uv2 ∈ S and S ∈ Σn .

B. Bag of Words Kernel

In Bag of Words kernel, strings (documents) are mapped into very high dimensional feature vectors, where dimension-ality of the feature space is equal to the number of words in a corpus. Each entry of the vector represents the occurrence or non-occurrence of a word by a number. Kernel represents
the inner product between mapped sequences which gives sum over all common (weighted) words.

C. Edit Distance Kernel

The edit distance between two strings is number of minimum editing operations, Insertion, Deletion, Substitution needed to transform one into the other. A dynamic program- ming and rather intuitive approach is explained below .



```
Initialization
 D(i,0) = i
 D(0,j) = j
Recurrence Relation:
 For each  i = 1…M
       For each  j = 1…N
                          ⎧D(i-1,j) + 1
              D(i,j)= min ⎨D(i,j-1) + 1
                          ⎩D(i-1,j-1) +   2; ⎧if X(i) ≠ Y(j)
                                          0; ⎩if X(i) = Y(j)
Termination:
 D(N,M) is distance
```

Fig. 4

Here, D represents a 2D-matrix used as a dynamic programming table . All the string kernels described above have been previously tested for a similar purpose [2]. However, in this paper we would discuss about the string kernels that are in common use today and handle higher order dimensionality in a more
efficient manner .

V. STRING KERNELS UNDER CONSIDERATION

A. (k,m) Mismatch Kernel

K-spectrum kernel was designed, in particular, for the application of SVM protein classification. These kernels use counts of common occurrences of short k-length sub- sequences, called k-mers, rather than notions of pairwise se- quence alignment. The k-mer idea still captures a biologically- motivated model of sequence similarity, in that sequences that diverge through evolution are still likely to contain short sub-sequences that match or almost match [15].

Using a mismatch tree data structure, the complexity of the kernel calculation was shown to be $O(c_K(|x| + |y|))$, with $c_K = k^{m+1}|\Sigma|^m$ for k-grams with up to m mismatches from alphabet Σ. Hence, a (k,m) mistmatch kernel is given by,

$$K_{(k,m)}(x, y) =< \Phi_{(k,m)}(x), \Phi_{(k,m)}(y) >$$

Observing carefully, the representation mentioned above, we can conclude that if the number of mismatches is zero, i.e., m = 0, then this kernel becomes a n(k)-gram kernel described earlier .

B. Restricted Gappy Kernel

For restricted gappy kernel or (g, k)-gappy string kernel,where g represents the number of gaps or g-ers and g ≥ k, we use the same |Σ|k-dimensional feature space, indexed by the set of k-mers from Σ, but we define our feature map based on gappy matches of g-mers to k-mer features. For a fixed g-mer α = a1a2...ag(each ai ∈ Σ), let G(g,k)(α) be the set of all the k-length sub-sequences occurring in α (with up to g–k gaps). Then we define the gappy feature map on α as

$$\Phi^{Gap}_{(g,k)}(x) = \sum_{q\text{-mers } \alpha \in x} \Phi^{Gap}_{(g,k)}(\alpha).$$

VI. EXPERIMENT

In this experiment, TCP network traffic produced by com- mon network applications is utilised. All network data is cap- tured, and sorted by Wireshark into separated files according to protocols and then split into individual flows (connections) by tcpflow.

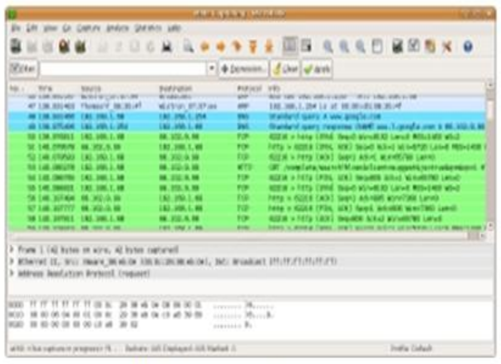


Fig. 5: Data collected from Wireshark.

Kernel Function	Substring Length	Connection Length	Lambda	Gen Accuracy
(k, m) Mismatch Kernel	4	750 bytes	0.5	99.61%
Restricted Gappy Kernel	4	750 bytes	0.5	99.55%

TABLE I: Classification results for each string kernel function.

Collected data was further filtered by removing traffic from unreliable protocols and normalizing string content to 450 bytes and 750 bytes per flow. Michlovsky et al. in their research deduced that gap-weighted subsequence kernel and N-gram kernel functions give 99.58% general accuracy, which follows
that only 2 of total 472 connections are misclassified. Our results using the mismatch kernel and restricted gappy kernel are summarized in the TABLE 1 .
Michlovsky et al. in their paper stated that the general accuracy decreases over the length of substring . However , using mis- match kernel and gappy kernel function we obtain an almost similar or higher level of general accuracy while maintaining a similar substring length and using 750 bytes length . N-gram kernel delivers a slightly lesser accuracy rate with 450 bytes length of data .
Analysis was carried out in Python using scikit-learn, numpy, pandas and matplotlib .

VII. CONCLUSION

Hence, this paper would ultimately like to state that (k, m) mismatch kernel function and restricted gappy kernel functions are also good options for classifying traffic . Moreover, they reduce the overhead of normalizing the data to 450 bytes and provide a slight better general accuracy than the already
implemented string kernels . Future work can probably consist of developing new string kernels and discovering new machine learning techniques for this purpose .

REFERENCES

[1]. Yu Liu, A Survey of Machine Learning Based Packet Classification, Vancouver, BC , 2012.
[2]. Michlovsky Z., Pang S. , Kasabov N., Ban T., and Kadobayashi Y., String Kernel Based SVM for Internet Security Implementation, Proc. Of ICONIP 2009, pp. 530-539, 2009.
[3]. Lodhi H., Saunders C., Taylor J. and Watkins C., Text Classification using String Kernels, Journal of Machine Learning Research 2 , pp. 419-444, 2002.
[4]. Jean-Philippe Vert, Practical session: String kernels, Notes - Centre for Computational Biology, Paris - 2010 .
[5]. Sunil Ray, Understanding Support Vector Machine algorithms, Analytics Vidya - (goo.gl/1iFnfz) .
[6]. M. Baldi, N. Cascarano, F. Risso, Service-Based Traffic Classification, IEEE - Netgroup at Politecnico di Torino .
[7]. Internet Assigned Numbers Authority (IANA) - (goo.gl/5dNIAR) (Last accessed June 2016) .
[8]. Moore, Andrew W. and Papagiannaki, Konstantina, Toward the Accurate Identification of Network Applications, Passive and Active Measurement Workshop (PAM 2005), March 2005.
[9]. C. V. Wright, F. Monrose, G. M. Masson, On inferring application protocol behaviors in encrypted network traffic, J. Mach. Learn. Res. 7 (2006) 27452769.
[10]. C. Wright, F. Monrose, G. M. Masson, HMM profiles for network traffic classification, Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, ACM Press, New York, NY, USA, 2004, pp. 915.

[11]. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Ma- chines and Other Kernel-based Learning Methods, Cambridge University Press, Cambridge (2000).
[12]. Sonnenburg S., Ratsch G. and Rieck K., Large Scale Learning with String Kernels - 2007 .
[13]. Lodhi H., String Kernels, University of Sheffield Lectures -(Jurafsky D., Stanford University Lectures, class/cs124/lec/med.pdf)
[14]. Leslie C. and Kuang R., Fast String Kernels using Inexact Matching for Protein Sequences, Journal of Machine Learning Research 5 , pp. 1435-1455, 2004.