Adaptive Network Intrusion Detection System using a Hybrid Approach

R Rangadurai Karthick
Department of Computer Science and Engineering
IIT Madras, India
ranga@cse.iitm.ac.in

Vipul P. Hattiwale
Department of Computer Science and Engineering
IIT Madras, India
vipul.hattiwale@gmail.com

Balaraman Ravindran
Department of Computer Science and Engineering
IIT Madras, India
ravi@cse.iitm.ac.in

***Abstract*—Any activity aimed at disrupting a service or making a resource unavailable or gaining unauthorized access can be termed as an intrusion. Examples include buffer overflow attacks, flooding attacks, system break-ins, etc. Intrusion detection systems (IDSs) play a key role in detecting such malicious activities and enable administrators in securing network systems. Two key criteria should be met by an IDS for it to be effective: (i) ability to detect unknown attack types, (ii) having very less miss classification rate.**

In this paper we describe an adaptive network intrusion detection system, that uses a two stage architecture. In the first stage a probabilistic classifier is used to detect potential anomalies in the traffic. In the second stage a HMM based traffic model is used to narrow down the potential attack IP addresses. Various design choices that were made to make this system practical and difficulties faced in integrating with existing models are also described. We show that this system achieves good performance empirically.

I. Introduction

Any attempt made to gain unauthorized access to a computer or disrupt the availability of a service/resource is termed as an intrusion. Intrusion Detection Systems (IDS) refers to a software or a system built to detect intrusions. In general, detection mechanism used by IDS can be classified into two major categories.

1) *Signature based detection*: Models built from well known attack types, i.e., from already known attack patterns.
2) *Anomaly based detection*: Modeled using normal traffic and deviation from this profile is considered anomalous.

Anomaly based techniques are preferred over signature based techniques owing to their ability to detect novel intrusions. Signature based techniques

The key aspects that we considered for building an anomaly based IDS are

- *Choice of attributes*: The model is proposed to be implemented in a web server or at network gateway, where the inflow of traffic is huge. We considered to use information available from a packet's header as features to build the model. This way we don't incur much overhead on the server and does not become a bottleneck.

978-1-4673-0298-2/12/$31.00 © 2012 IEEE

- *Handling infrequent patterns*: All normal network traffic do not follow uniform flow pattern. Any model proposed should be able to handle those normal traffic that are infrequent. Our model uses boosting techniques to learn over these infrequent patterns in order to classify them correctly.
- *False alarm rate*: The main drawback of an anomaly based detection is the high false alarm rate. Boosting technique used for the proposed model takes care of this problem and had very low false alarm rate.

We use Hidden Markov Model (HMM), a generative model, for modeling input data. The model is proposed to profile TCP based communication channel for intrusions. Any normal TCP connection would have three phases during their connection time, i.e., connection establishment, data transmission and connection termination phase. There is an inherent sequential nature in such mode of communication and makes it convenient for us to model them using HMM, which can exploit this nature of TCP traffic to build models.

A brief description of the HMM model is as follows. The first step in our approach is source separating traffic. It is performed on both training and testing traffic in order to preserve the sequence information of TCP traffic. HMM is used to profile source separated clean traffic and the model thus built is used to classify test traffic. This approach had high attack detection rate and also high false positive rate. High false positive rate corresponds to flagging legitimate traffic as attack and cannot be accepted when designing such systems. Hence various design choices, port based separation, cascading of HMMs, were considered for traffic profiling. These approaches increased the accuracy of the classifier with very low false positive rate. Intrusion detection data set released by DARPA [3] is used to train and test HMM models.

The HMM based model had few shortcomings when we tried to implementing it in real time. This lead us to look for alternative methods that could be compounded with HMMs to make it work in real time. Vijayasarathy et al. [2] had proposed a Naive Bayesian (NB) based model for profiling traffic. This approach handles the skewness in network traffic, i.e., the amount of anomalous traffic to a server is very low compared to that of clean traffic. NB based model runs faster,

close to line speed, and computes probability of occurrence of groups of incoming packets, windows.

The NB model is used for online classification and HMM model is used for offline analysis of traffic. Traffic that were flagged as anomalous by the NB model were fed into an offline HMM that computed the probability of connections present in the window. Thus combining NB and HMM models we form a hybrid model, where in NB model computes probability of occurrence of windows and HMM computes the probability of each connection within the window. This way the output from the HMM, list of attacking IPs, can be used as an update to firewall on what IPs to block. HMM now can be used to generate IP blacklist and makes the hybrid model more efficient.

The rest of the paper is organized as follows. A brief description of HMM is presented in Section 2. Section 3 describes our proposed HMM model and preliminary results obtained are presented in Section 4. Section 5 explains the problems that we anticipated that could be faced while implementing this system in real time. Section 6 describes hybrid model and Section 7 describes various experiments and results obtained. Section 8 describes related work that has been done by research community in this area.

II. Hidden Markov Model

HMM is a generative model that can model data which is sequential in nature. It is used to model data where the assumption of i.i.d. is too restrictive, like speech processing applications. A detailed tutorial on HMM is available in [1].

Markov Property: Consider a system with *N* states and at discrete time intervals, there is transition among states. Let these instances be *t*, $t = 1, 2, 3, \cdots$. Any process is Markovian if the conditional probability of future states, given the present state and past states, depend only upon the present state. In order to predict future state, the process by which the current state is obtained does not matter, i.e.,

$$\begin{aligned} Pr[q_t = S_i | q_{t-1} = S_j, q_{t-2} = S_k, \cdots] \\ = Pr[q_t = S_i | q_{t-1} = S_j] \end{aligned} \tag{1}$$

We have used HMM that follows the above first order Markov property.

In a HMM, the states and their transitions are not visible. Instead an output symbol, from a discrete set of symbols, is emitted during every transition. This sequence of symbols are the observables used to train a HMM. The following figure explains this.

Definition of a HMM:
HMM [λ] is a five tuple, i.e., $\lambda = [N, M, A, B, \pi]$.
The parameters of the model are

N, number of states in the model, S = $\{S_1, S_2, \cdots, S_N\}$.

M, number of observation symbols, V = $\{V_1, V_2, \cdots, V_M\}$.

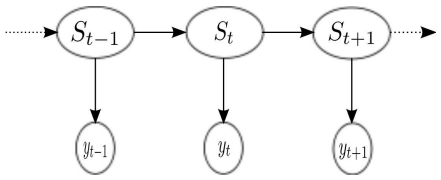

Fig. 1. HMM Architecture

A, state transition probability matrix, A=$\{a_{ij}\}$, where

$$a_{ij} = Pr[q_{t+1} = S_j | q_t = S_i] \quad 1 \leq i, j \leq N \tag{2}$$

It is a N*N matrix.

B, observation symbol probability matrix, B=$\{b_j(k)\}$, where

$$\begin{aligned} b_j(k) = Pr[v_k \ at \ t | q_t = S_j] \quad & 1 \leq j \leq N \\ & 1 \leq k \leq M \end{aligned} \tag{3}$$

It is a N*M matrix.

π, initial state probability matrix, $\pi = \{\pi_i\}$, where

$$\pi_i = Pr[q_1 = S_i] \quad 1 \leq i \leq N \tag{4}$$

It is a 1*N matrix.

Algorithms for HMM: The following two algorithms are used to model and use the HMM.

1) Baum-Welch algorithm is used to learn the parameters of the model, {A, B, π}, from input data.
2) Forward-Backward algorithm is used to learn the probability of occurrence of an observation sequence given the model, P[O|λ].

III. Design Choices

Web servers in general use Transmission Control Protocol (TCP) for communication between clients and server. TCP is a state based protocol, i.e., any TCP connection would progress through set of state transitions during its life time. This inherent stateful and temporal nature of TCP traffic could be captured well by using a HMM based classifier. This lead us to use HMM as our basic building block in our system design. In the remainder of this section, we describe parameters that were used to build our model and other design considerations that shaped our model design.

A. Choosing Parameters

The key aspect in building a HMM is to decide the states and symbols that are to be used to build the model. Choosing right set of attributes for a model is very important as this step would ensure effective usage of available data. For our experiments, we use TCP header information present in packets as features.

States of the model are called hidden or latent variables and are used to describe the underlying distribution generating the data. In our approach, states of the HMM do not correspond to actual TCP states. They are used to model the HMM to best explain the traffic. They do not have direct physical significance. For example, network traffic can be assumed to consist of traffic from legitimate and malicious users. Transition from one state to another can be considered equivalent to switch from traffic between malicious and legitimate users.

Next we had to decide upon what could be used to represent symbols in our HMM model. We use TCP flags as symbols for the HMM model, following Vijayasarathy et al. [2]. The other parameters of the HMM model - π, A, and B are estimated using Baum-Welch algorithm.

B. Initial Approach

Building anomaly based classifier involves two phases - training and testing. During the training phase, the classifier is made to profile over clean traffic, i.e., traffic stream which is devoid of any malicious traffic stream. During the testing phase, traffic which were not used during training are used to measure the performance of the model built. The classifier flags any traffic that deviates from clean traffic profile as suspicious. The intuition behind this approach is that clean traffic and malicious traffic are not generated from the same distribution.

Training phase of our algorithm begins with source separating training traffic into separate streams. All packets between a unique source/destination IP pair constitute a stream. Each stream consist of series of TCP flags that were used in the packets throughout the connection. Then a single HMM model is used to learn the characteristics of all streams to the server.

The HMM model takes these TCP flags as observables and other parameters of the model can be computed from them. Upon analyzing the traffic data, we found that only few flags were used in general for most TCP communication. We associated a number with each flag and a connection with sequence of flags is converted into a sequence of numbers. HMM model is trained over this sequence of numbers. The frequently used TCP flags and the unique ID which we used for our modeling are as follows.

- SYN - 0
- SYN/ACK - 1
- ACK - 2
- PUSH/ACK -3
- FIN/ACK - 4
- RST - 5
- other TCP flags - 6

The same procedure is followed in the testing phase. The TCP flag sequence is converted into a sequence of numbers and the probability of occurrence of this sequence is tested over the model. Since states of the model does not correspond to actual TCP states, the number of states can be chosen empirically.

The testing phase of the above said approach is depicted in Figure 2.

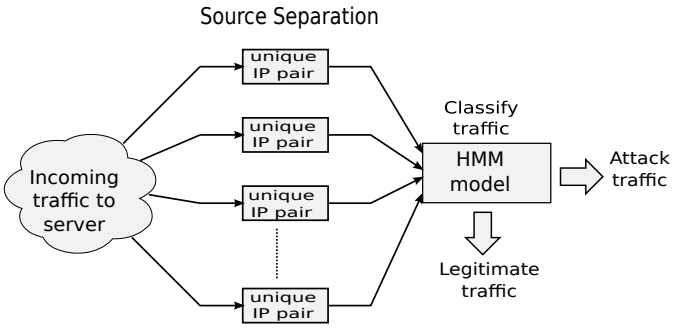

Fig. 2. Initial Approach

DARPA data set for intrusion detection [3] is used for training and testing our HMM model. Preliminary results obtained for the above said approach were not satisfactory. The model had very high false positive rate, i.e., clean traffic stream were also being flagged as attack. The classifier did not succeed in discriminating between good traffic and bad traffic. This low performance might be attributed to using just one HMM to learn all clean traffic profile. A single HMM could not capture all the characteristics of clean traffic that were used for training.

C. Alternate Design

In order to overcome the above said shortcoming, we performed source separation on training/testing traffic according to destination ports of the server and then upon source/destination IP address. Instead of using a single HMM to lean all traffic coming to a server, we used separate HMMs for each frequently occurring server port. The reasoning behind such an approach is that not all traffic belonging to different applications behave in the same way. For instance, different traffic streams belonging to a particular application port, say port 25 (SMTP), have similar characteristics than to traffic at port 20 (FTP). This approach improved the results drastically, i.e., the model had higher accuracy and lower false positive rate compared to the single HMM approach.

The implementation details of this model are as follows. Training traffic to the server is first separated based upon destination port number of packets. Traffic to particular ports are then source separated and trained by separate models for each port. Ports which have higher traffic, like ports for HTTP, telnet, FTP, etc., were modeled with separate HMM models. Traffic to other infrequent ports were modeled by a separate model. The testing phase proceeds the same way. Testing traffic is first separated based upon ports and then source

separated and tested by corresponding HMM model for the port. Figure 3 describes this approach.

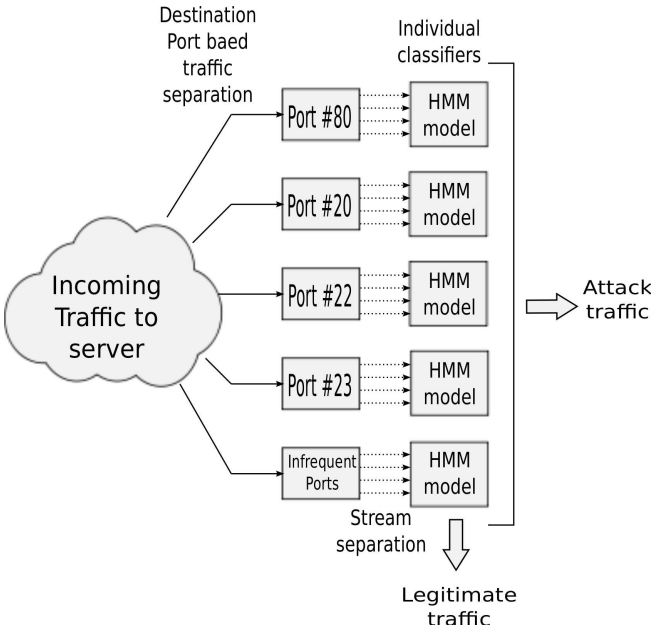

Fig. 3. Layered Model

Even though port wise separation approach had better results that single model approach, the false positive rate were still high, almost 10% of training traffic were flagged as attack. Any practical system designed to detect intrusions should have low false positive rate, i.e., rate at which a legitimate user is wrongly classified as attack should be very low. It is able to classify most of the frequently occurring positive traffic correctly but it is not able to correctly classify positive traffic that were infrequent. Infrequent traffic that were clean or positive were also flagged as attack. We made this model as our basic classifier model and it required us explore other strategies that would improve the performance of our base classifier.

D. Cascaded HMMs

The positive traffic that were wrongly classified by the above approach were traffic streams that were not so frequent. This can be attributed to those traffic streams which had very low probabilities in the training phase of the above approach. In order to overcome this high false positive rate, we employed multi-stage combination of models to improve the base classifier's performance. We employed cascading of base classifiers into several layers to improve performance. Figure 4 describes the cascading of models.

Implementation details of this approach: Low probability legitimate streams that were flagged suspicious by all the base classifiers are fed as input to a separate HMM model. This HMM trains on all the infrequently occurring streams and builds a model. Traffic streams that have low probabilities in this model are fed into next layer of HMM model for training.

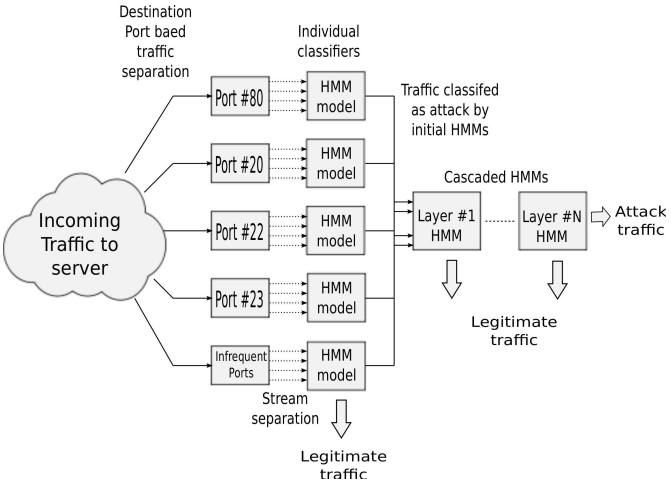

Fig. 4. Cascaded HMM design

The above process of adding new HMMs, i.e., cascading HMMs, can be continued until addition of a new model makes no improvement to the accuracy of the model.

The usage of traffic streams from different protocols for the first layer of cascaded HMM might be counter-intuitive, since we perform protocol based traffic separation in the first step before feeding traffic into HMM models for each protocol. We observed that most of training traffic connections to frequently occurring ports were correctly classified by their respective HMM model. The number of connections that were wrongly flagged as anomalous were very less. But this was not the case with the HMM model for infrequent port traffic. The traffic connections that occur infrequently were the ones wrongly flagged by initial HMMs. Since the connections were anyway infrequent in their respective protocol, combining them together did not reduce the performance of the model. Instead, it improved the accuracy of the HMM model.

The HMM model can be extended to having separate levels of cascading for each protocol. Since the data available for training and testing were limited, we performed a combined layer of cascading for all protocols.

IV. Preliminary Results

Building any classifier involves two phases, i.e., training and testing phases. Training phase in our approach involves learning the parameters of the model from a clean traffic trace. HMM profiles this data and uses this information to test incoming traffic. During the testing phase, traffic that were not used for training are tested against the model learnt. To build a classifier we need to have labeled data for training and testing. Data sets released by DARPA[3] were used to train and test our classifier.

Experiments

The experiments that were conducted are described as follows.

# states	Connection Separation	Separate Models for Protocols	Boosting	Accuracy (%)	False Alarm Rate (%)
5	Just IP	No	No	81.75	19.63
9	Just IP	No	No	85.14	15.05
5	IP & Port	Yes	No	91.49	9.49
9	IP & Port	Yes	No	92.27	8.49
5	IP & Port	Yes	Yes	96.96	2.89
9	IP & Port	Yes	Yes	97.1	2.71

TABLE I
RESULTS ON DARPA DATA SET

1) Single HMM model: Training traffic is separated according to source/destination IP pair and trained with a single HMM model. In the testing phase, source separated connections were tested against the learnt model. The performance of the model is bad since it had very high false positive rate. Probable reason for the failure of the model could be that a single HMM could not capture all possible traffic characteristics. High false positive rate can be alleviated by the following approach.

2) Multiple HMM models: We performed source separation both on IP and port information of source and destination. Separate HMMs were used to train/test connections pertaining to different protocols. Protocols with large amount of incoming traffic were trained separately, while other infrequent ports were trained separately. This approach reduced the false positive rate and we made this type of source separation as our basic step for building HMM.

3) Cascading of HMMs: In order to improve the performance of the above approach, we employed boosting. HMM models were cascaded into several layers to model low probability traffic. The results reported for our experiments are using two layers of HMM model for cascading.

We used two days of clean traffic data from DARPA data set for training and the rest of the traffic from other days were used for testing the learnt model. This way we don't overfit the training process. Table I describe the performance of our model on a particular server in DARPA data.

Number of states for the model

The number of states to be used for HMM could be determined experimentally. Using 9 or 10 states for the model gave us good results for DARPA data set. We tried using higher number of states for HMM and the results obtained were similar and did not improve the performance any further. Hence we have reported the results on using 9 states for HMM model.

Attacks detected by HMM

The following attacks present in the DARPA data set were detected by HMM model.

- neptune - Syn flood denial of service attack on one or more ports.
- ipsweep - Surveillance sweep performing ping on multiple host addresses.
- portsweep - Surveillance sweep through many ports to determine which services are active on a single host.
- satan - Network probing tool to exploiting well-known weaknesses.
- nmap - Network mapping using the nmap tool.

Auckland Data Set

We tried our cascaded HMM experiments on Auckland IV[4] data set. In the training phase, HMM model is trained with clean HTTP traffic from DARPA data. For testing purpose, HTTP traffic to various servers in Auckland data set were considered. Auckland data set is not a labeled data set. Hence the testing results had to be cross checked manually. HTTP sequences that were flagged as anomalous were of the following types.

- *Reset Attacks*
- *Short Connections*

Connections that were too short were flagged as anomalous by the model. The reason for very short connection length could be abrupt end of connection. HMM model with just 5 states is sufficient for classifying Auckland data set.

V. MOTIVATION FOR HYBRID APPROACH

The goal of the work is to implement suitable models that can function effectively in real time. When implementing the above model into a real-time system and it in turn had the following pitfalls [6].

Source separation of incoming traffic is the first and foremost step in our design. This way, the model keeps track of all incoming IP addresses. But then, the problem of IP address spoofing could tax our proposed model. Assume an incoming packet to have spoofed IP address. The server replies to it and allocates resource for this IP address. It is highly unlikely that the connection established by a spoofed IP address would proceed any further. This would make the server to wait until time out period and to reclaim allocated resource. The above scenario could be repeated by attackers and result in exhausting the resources of a server.

The second issue to consider is the typical length of a connection. The DARPA data used for training and testing our model had information about entire connections. But in reality, we have no way of telling when a connection would end. The computations performed had complete end to end connection data, which is quite impossible in reality. If this model were to be implemented in a server, then the server has to have separate buffers for each incoming new connection. This again would end up in using all of server's available buffer to store packets. We cannot decide on how much buffer

space to allocate, since we do not know the length of each connection.

Aforementioned drawbacks prevent HMM based model from being implemented as a stand alone device for a server's security. In the next section, we will describe another approach where HMM model coupled with Naive Bayesian based model would overcome these issues.

VI. Hybrid Model

We propose a hybrid model combining our HMM based model with Naive Bayesian (NB) based approach proposed by Vijayasarathy et al. [2] to address the issues. Hybrid model would have NB model for online learning and HMM model for offline learning. The online classifier (NB model) would monitor incoming traffic and flag traffic blocks that are suspicious. The offline classifier (HMM model) would be fed with the traffic flagged by NB model. HMM model would then perform source separation for the connections present in the flagged traffic and classifies the connections as either attack or normal. The addition of HMM model to NB model is intended to narrow down on the attacking IPs present in flagged traffic rather than to improve the performance of it.

Figure 5 depicts the proposed hybrid model.

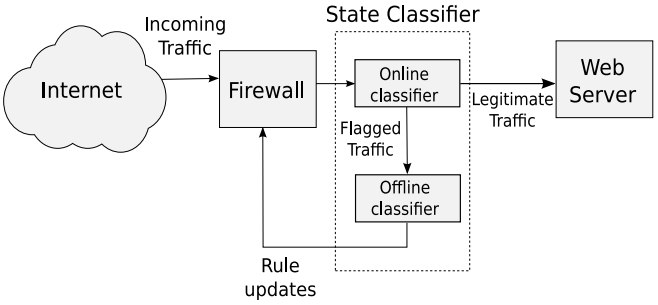

Fig. 5. Flow diagram

A. Working of NB model

Incoming traffic to NB model is split up into logical units called windows. Each window is a group of packets and modeling is based on TCP flags set in packets. Based on experiments, packets in a window can belong to any one of the following depending upon the TCP flag that is set.

1) RST - packets with reset bit set
2) SYN - syn packets
3) ACK - ack packets
4) FIN/ACK - fin/ack packets
5) PSH/ACK - push/ack packets
6) others - packets with other TCP flags set.

This six tuple describes the mix of traffic present in a window. For example, for a window of size 100 with <3 RSTs, 8 SYNs, 48 ACKs, 1 FIN/ACKs, 40 PSH/ACKs, 0 other packets> could be considered normal, but <0 RSTs, 100 SYNs, 0 ACKs, 0 FIN/ACKs, 0 PSH/ACKs, 0 other packets> would represent a syn flooding scenario. Windows that are similar are grouped to form bands, so as to reduce the number of different events to model. During the training phase, the model computes the probability of occurrence of various event types. In the testing phase, traffic windows with very low probability of occurrence are flagged as attack.

For a detailed description of NB based approach, refer to the original paper [2]

B. Combining the models

We used the same clean data set for training both NB and HMM models. In the testing phase, for every window, the NB model would classify if it were normal or abnormal. In our implementation, if there were five consecutive attack flags raised by the NB model, and incoming traffic from then on would be buffered and fed as input to the HMM model. Attack flag would be on until there are five consecutive normal windows to the server. Buffering of data is carried on between the raise and fall of flag. The number of windows to consider during time out mechanism is implementation specific, depending upon traffic characteristics of a server.

The windows that were buffered when the attack flag is on, would be fed into HMM for further processing. Source separation is then performed on the IP addresses present in the flagged traffic. Individual streams, thus obtained are tested using HMM model and probability of occurrence of each sequence could be calculated. IP address of connections that were anomalous could be added to firewall black list. This way, HMM model could be used to blacklist IP addresses that have suspicious traffic characteristics. Experiments conducted using this approach are described in the next section.

VII. Experiments and Results

We used CAIDA [5] data set for testing our hybrid model. It is an hour long Distributed Denial of Service (DDoS) attack data at a server. Since the data set consist only of attack traffic, we mixed it with normal traffic and used hybrid model to classify it. We considered traffic to port 80 (HTTP protocol) for training and testing. We used same clean traffic trace from DARPA data set for training both NB and HMM models.

For testing the classifier, we mixed traffic traces from CAIDA and DARPA together. The IP address of destination server of CAIDA data set traffic is changed to the same destination server used by DARPA traffic. This mixed traffic is used for testing our hybrid model. NB model processes incoming traffic online, and feeds offline HMM with flagged traffic. When an attack flag is raised, we start buffering from ten windows prior to the window where the attack flag was actually raised. This is done to ensure that we don't erroneously classify connections that began recently, just before attack flag was raised, as attack. Buffering continues until we receive five consecutive clean windows or upto the end of testing data file. Figure 6 describes this process.

HMM was trained throughout with equal prior probability to all states. This is done to ensure not to flag any clean traffic stream that was buffered from middle as attack. When an attack flag is raised, there would connections that would have

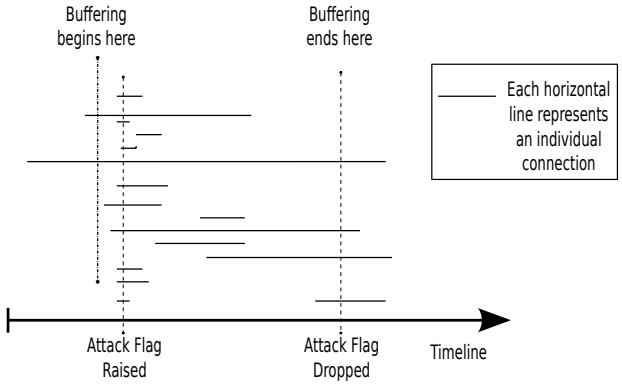

Fig. 6. Buffering Procedure

started early and half way through. The beginning of such a stream would not necessarily have three way handshake.

We perform source separation on the buffered data and use HMM to classify it. HMM was successful in finding out IPs from CAIDA traffic on almost all cases. HMM with more than five states had 100% accuracy classifying all IPs from CAIDA as attack and IPs from DARPA as clean. When tried with HMM with less than five states, few of the attacking streams were classified as clean traffic. This is due to the inability of HMM with very few states to model the data accurately. Using HMM with larger states gave us exact results and the number of states to be chosen for a server can be computed empirically.

VIII. Related Work

Lee et al. [7] proposed a system using combined misuse and anomaly detection approaches to generate rules for IDS. For improving efficiency, multiple model cost based approaches are applied. These analyze and detect models with high accuracy but low cost. A distributed architecture is proposed for evaluating models in real time. To improve usability adaptive learning algorithms are used for incremental updates. To reduce reliance on the labeled data unsupervised learning is studied.

Ourston et al. [8] have used a HMM based model to detect complex network attacks that happens in various stages. They use HMM to model alert sequences that were raised between every source/destination IP pair. The hidden states of their model correspond to various attack stages. For example, a generic network intrusion would undergo the following states, i.e., probe, consolidate, exploit and compromise. They use separate HMMs for every attack type to detect such multistage network attacks.

Chu et al. [9] use an Frequent Pattern Tree (FP-Tree) based approach to construct a network intrusion detection system. The architecture used for on the updates and detection is outlined. FP Tree is constructed for normal traffic as well as attacks so as to increase the detection rate and decrease false alarm rate.

Cai et al. [10] have explored a TCP rule based TCP anomaly detection system. TCP header information is used to generate different clusters of normal traffic. These clusters then represent unique patterns in normal network traffic. A connection which is dissimilar to all these clusters is termed as an attack.

In [11], Estevez et al. have used a Markov model to model incoming HTTP requests to a server. The key assumption is that HTTP protocol requests have highly structured payloads.

Xu et al. [12] have considered monitoring the influx of new IP address during DDoS scenario. IP address of incoming traffic is categorized into two types, i.e., those which have been observed already and those which are new. HMM is used to model this sequence of IP address. The models are placed at distributed points in the network and Reinforcement Learning (RL) algorithms are used for efficient message passing among these distributed network points.

IX. Conclusion

In this paper, we have proposed a hybrid approach for adaptive network intrusion detection. We started off with HMM for network intrusion detection and it performed good empirically on DARPA data set. The difficulties that might arise when implementing HMM model in real time were described. We incorporated HMM model along with NB model into a hybrid model for intrusion detection. The proposed hybrid model also performed well in detecting intrusions and the experiments and results also reported. As an extension to HMM model, we would like to look at characterizing the diurnal variation characteristics of traffic to web server. It would involve learning the nature of traffic at various instances of the day.

References

[1] Lawrence R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, 1989.

[2] Vijayasarathy, R., Ravindran, B.and Raghavan, S.V., *A system approach to network modeling for DDoS detection using a Naive Bayesian classifier*, COMSNETS, 2011.

[3] *DARPA intrusion detection evaluation dataset*, http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html

[4] *Auckland IV Dataset*, http://www.wand.net.nz/wits/auck/4/

[5] Paul Hick, Emile Aben, kc claffy, Josh Polterock, *The CAIDA "DDoS Attack 2007" Dataset*, http://www.caida.org/data/passive/ddos-20070804_dataset.xml

[6] Vijayasarathy, R - personal communication.

[7] Wenke Lee and Salvatore J. Stolfo and Philip K. Chan and Eleazar Eskin and Wei Fan and Matthew Miller and Shlomo Hershkop and Junxin Zhang, *Real Time Data Mining-based Intrusion Detection*, IEEE, 2001.

[8] Ourston, Dirk and Matzner, Sara and Stump, William and Hopkins, Bryan, *Applications of Hidden Markov Models to Detecting Multi-stage Network Attacks*, HICSS, 2003.

[9] Nelson CN Chu, Adepele Williams, Reda Alhajj, Ken Barker, *Data stream mining architecture for network intrusion detection*, IEEE, 2004.

[10] Weijie Cai , Li Li, *Network Traffic Anomaly Detection Using TCP Header Information*, IEEE, 2004.

[11] Estevez-Tapiador, Juan M. and Diaz-Verdejo, Jesus E, *Detection of Web-based Attacks through Markovian Protocol Parsing*, ISCC, 2005.

[12] Xu, Xin and Sun, Yongqiang and Huang, Zunguo, *Defending DDoS Attacks Using Hidden Markov Models and Cooperative Reinforcement Learning*, PAISI, 2007.