BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **8**, No 3

Sofia • 2008

A Study on Network Flow Security

Tsvetomir Tsvetanov[1], *Stanislav Simeonov*[2]

[1] *Sofia University, Faculty of Mathematics and Computer Science, 1000 Sofia*
E-mail: ttsvetanov@gmail.com
[2] *Burgas Free University, Faculty of Computer Science and Engineering, 8000 Burgas*
E-mail: stan@bfu.bg

Abstract: *The computer networks are significantly critical factor in the performance of any modern enterprise. The network management is as important as managing any other aspect of a company's performance and security. Many tools and appliances for monitoring the traffic are already in place for analyzing the security aspects of the network flows. All they are using different approaches and they rely on different characteristics of the network flows. The network researchers are still working on a common approach for security base-lining that might enable early alerts. This paper is discovering a different view on the network security base-lining based on a simple flow analysis utilizing the flows measurements and the theory of the hidden Markov models.*

Keywords*: Computer network security, network traffic analysis, Markov models application.*

1. Introduction

The network security has always been one of the daily computers-related and computer networks issues. Two major approaches are utilized in the theory of computer network modeling: the queuing theory [2] and the hidden Markov models [3]. Basically, the queuing theory addresses the models for the ubiquitous service. The general issue of the contemporary online services is how we can assure that every customer would be satisfied in fairly short timeline. The seconds approach utilizes the Hidden Markov Models (HMM). In fact, the queuing theory also uses the Markov chains and models but we are addressing the network analysis based on the Markov models only. The hidden Markov models are stochastic systems that

have two parts: internal hidden part and a visible part that is manifesting the hidden behavior to the observer. The internal states of the model are hidden for the observer. We call them also *hidden states*. The system moves from state to state within the discrete time intervals. In the context of the computer networks we can think about the hidden states as internal processes or events occurring in the network environment. Besides the hidden part, the hidden Markov model has a "visible" part. The observer can "see" the hidden state manifestation in means of *observed symbols* emitted by the internal processes. The set of the observed symbols is the symbols *alphabet*. When we are talking about hidden Markov model application for network analysis, we can also call that alphabet of the observed symbols network alphabet. Should we pick up the hidden Markov model for network analysis application, we will need to define the set of the hidden states and the network alphabet, matching them to particular semantics in the context of the network analysis application.

In this paper we will focus on the network flow modeling for network behavior pattern recognition. In order to address this problem we are choosing the HMM approach. Once having built the model patterns we would be able to recognize if our network is handling a legitimate traffic or the traffic flow is offensive and malicious. The reason for choosing the HMM approach is also based on the following facts:

- There are certain internal processes running in the network and they are hidden for the end-users and even for the monitoring tools. Those processes are affecting the network resources and are changing the infrastructure behaviour.
- The comprehensive network devices are able to provide some predefined characteristics of the network flow to the network monitors. That information cannot be unlimited by quantity and by property, obviously because this is additional function for the network devices and it produces additional overhead on the device's resources. The device vendors addressed the network monitoring requirement by aggregation and export of the protocol headers in predefined data structures.

2. The network alphabet

Using a network tool for collecting the traffic tokens we can build up a network profile for certain discrete time intervals. The basic idea of this task is to receive two groups of observed *network words*:

- Words that were read in a time of normal network behavior, when there was no anomaly in the traffic flow or the anomalies were too weak to affect the network assets and performance.
- Words observed during long running intensive network flow anomaly when the there might be a Denial-of-Services or similar attack executed against the network resources.

The definition of the network alphabet that we are going to use is based on the relation between the communication peers from the previous traffic entry and the current one. The comparison of those two traffic entries depends on the network

and transport protocol header values. As we are focusing on IP network stack analysis, the following values are taking in account:

- Source IP address
- Source port
- Destination IP address
- Destination port
- Protocol code (e.g. ICMP=1, TCP=6, etc.)

The following table contains the network alphabet semantics.

Table 1. The network alphabet semantics

A	The communication peers are absolutely the same. The previous traffic flow entry and the current entry are absolutely the same, i.e. those are the same connection flows
B	The previous and the current traffic flow entry have one different port, either source or destination, for example: TCP / 10.10.167.154:2311 – 10.10.10.5:80 TCP / 10.10.167.154:2314 – 10.10.10.5:80
C	The traffic flow entries have one different IP address, either source or destination, for example: UDP / 10.10.10.8:53 – 10.10.16.4:53 UDP / 10.10.10.8:53 – 10.10.16.5:53
D	The traffic flow entries vary by the ports and the IP addresses are still the same, for instance: TCP / 10.10.10.3:162 – 10.10.10.8:53 TCP / 10.10.10.8:543 – 10.10.10.3:53
E	The traffic flow entries holds the same IP address and port pair for one of the peer, either source or destination, but the other peer IP address and port does not match. The most common reason for this kind of consequence could be that the matching peer is actually a service that was requested from different client as you can see in the flow entries example: TCP / 10.10.167.154:2311 – 10.10.10.5:80 TCP / 10.10.10.5:80 – 10.8.130.35:45319
F	The traffic flow entries, the previous and the current, hold the same IP address, either source or destination, but all other peer properties are different: TCP / 10.10.167.154:2311 – 10.10.10.5:443 TCP / 10.10.10.5:80 – 10.8.130.35:45319
G	The current and the previous traffic flows match only by one port, either source or destination: UDP / 10.11.101.230:21439 – 10.10.10.5:161 UDP / 10.16.116.159:19012 – 10.10.10.4:161
H	The traffic flow entries have absolutely nothing to deal each other

It is important to note that all the symbols except the symbol ‘H’ require the same transport protocol code. When the parameters source IP address, destination IP address, source port, and destination port, do not match between the traffic flows, the emitted symbol will be ‘H’ no matter the protocol is the same or not.

The obvious purpose of this network alphabet semantics is to find out the following events and processes occurring in the network environment:

A long sequence of ‘A’ symbols would match to a communication between two network nodes over the same communication channel. The common scenario is when the communicating pairs are connected over TCP channel and they are exchanging data sporadically. In this case the network device would aggregate the traffic flows in individual entries as the flow cache at the network device memory would expire before the TCP channel is closed.

A long sequence of ‘B’-s then would be long running communication between a client and a server similar to that described for the symbol ‘A’ but with the difference that the client is using several communication channels to the server. As every channel might be able to finish its lifecycle within the traffic flow cache expiration at the network device, the communication data might be fully aggregated and exported as a single traffic flow entry.

If we find out a sequence of many ‘C’-s it would mean a service that is used by many different clients. However, the service itself requires the clients to use a specific port number rather than an arbitrary number. Examples for this kind of services are Domain Name Service (DNS) and Hot Stand-by Routing Protocol (HSRP).

Too many occurrences of ‘D’-s might be caused by an excessive communications between two nodes over different kind of protocols and services. The common case is when the peers are connecting each other on ad-hoc application ports.

Similarly to the symbol ‘C’, the occurrences of ‘E’ match a service used by different clients. However, unlike ‘C’, the service does not require the clients to connect from specific port and they can use arbitrary values. This is the most common scenario of service utilization.

If we observe many ‘F’ symbols, it might mean that different clients are accessing different services from the same server host.

Similarly to ‘F’ observations, if we see many ‘G’-s then it might be the case when different clients are accessing same kind of service running on a different host, for example when the service is provided in a high-availability mode by a server farm.

The occurrences of ‘H’ symbols match fully heterogeneous traffic flow. For the normal network behavior this is expected ‘t’ be the most common symbol in the observed network words.

3. Collecting the network symbols

In order to collect the network symbols from the traffic flow data, we will need a software system that interfaces the data and translates the flow entries into network words. The tool implements the following functions:

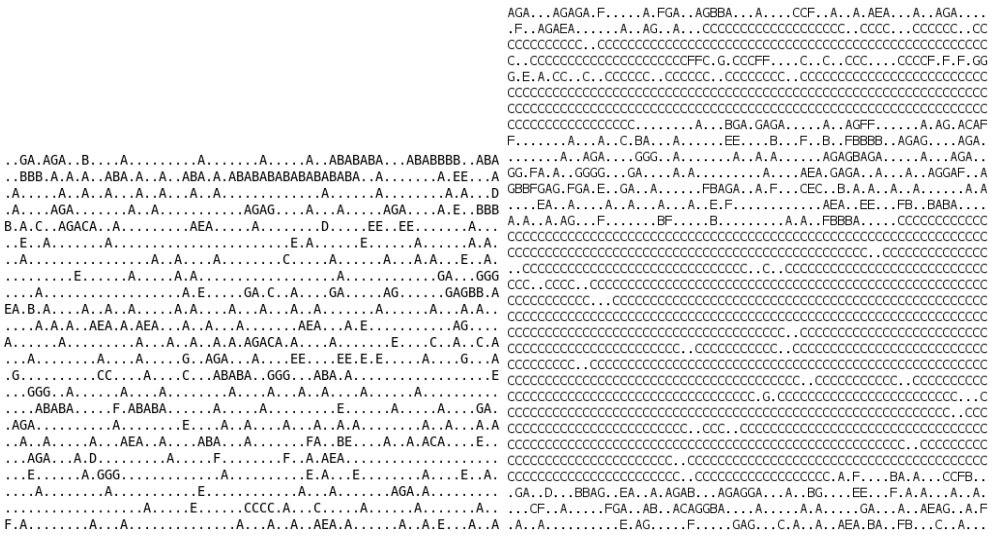
- Collects the traffic flow data from the network devices utilizing the standard NetFlow data format [1].
- Translates the traffic flow data into network symbols according to the network alphabet semantics. The results (the network words) then are exported into a file or set of files onto the file system.

Due to performance considerations and in order to make a flexible design we separated those two tasks in two different processes communicating via shared memory segment. The first process collects the data from the devices via UDP port. Then the collected NetFlow traffic data is preprocessed and put into an IPC message queue for handling by the other process. The second process takes the data from the message queue and compares the traffic entries. Depending on the relation between the neighbor traffic flow entries, the process decides on what network symbol is emitted. The decision to use a shared segment and two processes rather than a single process that reads and handles the data is based on the consideration that the single process might fail to read all data when it is busy with processing it. For example, if too many devices are forwarding the NetFlow data to the tool, the buffer could be quickly filled up as the process is spending more time in emitting the network symbols. In this situation the next traffic flow data might be missed and the output would not be relevant. Using a shared memory we are able to overwhelm and process promptly the input data. The second process can also continue emitting the symbols even if the first one was stopped if there are still data in the message queue.

The tool was developed and deployed into a real network environment for making the field tests. The adjacent devices were configured for traffic data export to the tool. In order to avoid the data duplicates the data was captured by the ingress routers only. The tool collected and translated the data into network symbols replacing the symbol 'H' with dot ('.') just for easy-to-read purpose as we expected to see much more 'H'-s for the normal traffic behavior. The network symbols were collected in fairly long period of time making sure different network states were captured. The results are shown in the following figures.

Fig. 1(a) shows mostly dots that in fact are 'H' symbols as we replaced them with dots. According to the network alphabet definition, as many 'H'-s occur in the network words as heterogeneous the traffic is. The heterogeneous traffic normally stands for normal network behavior. Some researchers are focusing exactly on the homogeneity flow measurements in order to detect network threads. According to the conclusions done, the homogeneous traffic is most likely produced by machine rather than by normal human interactions. That's why the homogeneous (machine-generated) traffic might be an alert for an offensive traffic flow running on the wire.

The next Fig. 1(b) comprises of parts with fairly heterogeneous symbols, however, there are some parts with mostly 'C'-s in a long sequences. According to the alphabet semantics, there adjacent traffic flow data contain the same peer host and port, either source or destination, while the peer is different between the flow entries. This might be the common case of using the same service from different clients. The service does not require specific client port number. Depending on the length of the 'C' sequences and depending on the real time interval (note that the symbols are matching a discrete time interval), this could be either legitimate traffic flow (legitimate clients are excessively using the service) or it could be a flooding targeted to the service provider host (a denial-of-service attack). We consider the example on Fig. 1(b) as quasi-homogeneous flow.



(a) (b)

Fig. 1. Networks words captured during a normal network behavior (a) and offensive traffic (b)

The example on Fig. 1(b) refers to a suspicious traffic flows to some specific service. This is the case when the ‘C’ symbol occurs in long sequences. The same traffic model would be observed when the server ports are scanned or when the service is excessively flooded with useless messages. Both cases are denial-of-services attacks with different goals: scanning and flooding. However, we have to take in consideration that the excessive packet flows to a particular service does not necessarily means the traffic flows are malicious. It is absolutely true that the offensive port scanners and flooders are generating those kinds of network words. However, the same words would be emitted even by the legitimate clients if in that particular moment the clients are connecting to the service. Usually this might be the typical case when in the morning business hours the employees are authenticating onto the Intranet and starting to access their common services. The other situation would be when a scheduled task runs and it produces a big bulk of packet flow onto the wire. Another example that comes in mind is the FTP protocol uploads and downloads. When huge files are transferred, the connection stays open a long time. Then the network monitors would observe long ‘A’-s and ‘B’-s, although it is absolutely legitimate network usage. Due to those reasons, when talking about a malicious traffic actually we are talking about a network behavior that might be exposed when a malicious traffic is going on. In order to be more precise we will call those traffic flows mostly suspicious rather then malicious. The suspicious traffic is a general definition that includes the real malicious and offensive traffic as well as the legitimate traffic that behaves the same way as the malicious does.

What would be then if too many long sequences of ‘A’-s and ‘B’-s were observed in the network words? As we already clarified, the occurrences of A, B, and C, stands for suspicious flows. The difference would be the target. In case of long A-sequences the target could be either source or destination but as we said it

might be excessive legitimate traffic. The B-sequences are likely to target a service flooded from the same source host. Also, the same sequence matches to the case when an attacker is scanning the destination ports in a long sequential order. Then the source side is using same port for packets generation. The C-sequence is excessive using of a service by multiple clients (legitimate case) or a denial-of-service flooding (malicious case).

4. Result analysis

After the network words are collected by the tool and stored into files, we are going to make a post-mortem analysis on the observed flows. Before proceeding to the task, we need to make few definitions used later in the paper.

Let the observed symbols set be $V = \{V_1, V_2 ..., V_M\}$. The set V is the network alphabet as previously defined in Table 1. As per the definitions the network alphabet size is 8, i.e. M=8. There are two major characteristics that we would be interested during the pre-modeling analysis. The first question to answer is how long the longest sequence of same symbols is, no matter the symbol itself. The next problem is what is the symbol share overall the whole observation discrete period of time. For those purposes we are introducing two definitions over the network alphabet set V.

Definition 1. The longest sequence $\rho_i(T)$ of same symbols V_i in the observed word O for the discrete period of time T, is called *density of the symbol V_i for the observed period T*. The set of all densities $\rho_i(T)$ for the corresponding symbols V_i, $\rho(T) = \{\rho_1(T), ..., \rho_M(T)\}$, is called density of the network alphabet V for the observation period T.

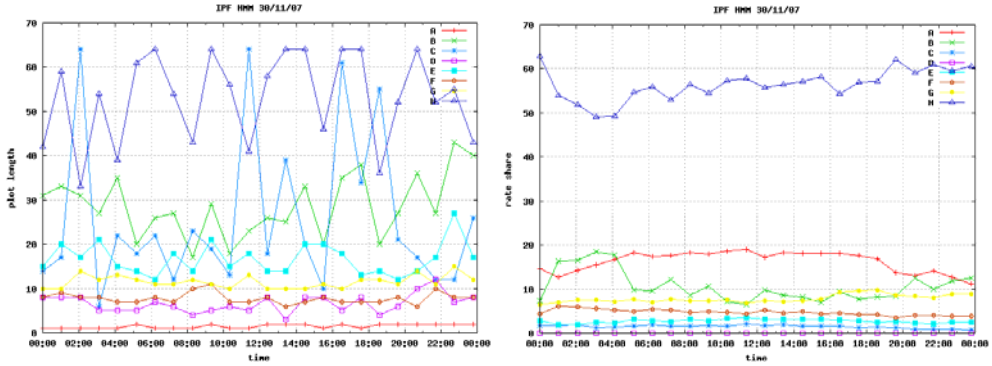
Definition 2. The relation $\phi_i(T)$ of the number of the occurrences of symbol V_i for the discrete period of time T and the length of the observed word O for the same period T is called *frequency of the symbol V_i for the observed period T*.

According to the second definition, the following relation immediately implies $\Sigma\, \phi_i(T) = 1$, $i \in [0; M]$.

Once collected, the network words are calculated against the density and the frequency values. If the words are observed during the normal behavior then we would get ρ and ϕ parameters for the normal traffic model. Using a test program we simulated flowing attack and collected the network symbol. The overall field tests were made over 24-hours period of time. The results were collected in series of files and then we proceeded with the flow analysis.

The next Figs. 2 and 3 summarize the analysis result. We have processed the network words as follow. The overall period of time was separated in series of time intervals. As the tests ran for 24-hours, we divided the results in 24 output words, one for every day hour. Then we picked up the words and for each of them we calculated the density and frequency for every of the network symbols. Finally, we have had 24 values of the arrays $\rho_i(t)$ и $\phi_i(t)$ for each time interval $1 \le t \le T$. The output results were generated as double arrays $\rho(T)$ and $\phi(T)$ and the values were translated in a graphical format.

The four figures are presenting the values of $\rho(T)$ and $\phi(T)$ during the normal flows and during the suspicious flows.



(a) (b)

Fig. 2. Density of the network symbols during the normal network behavior

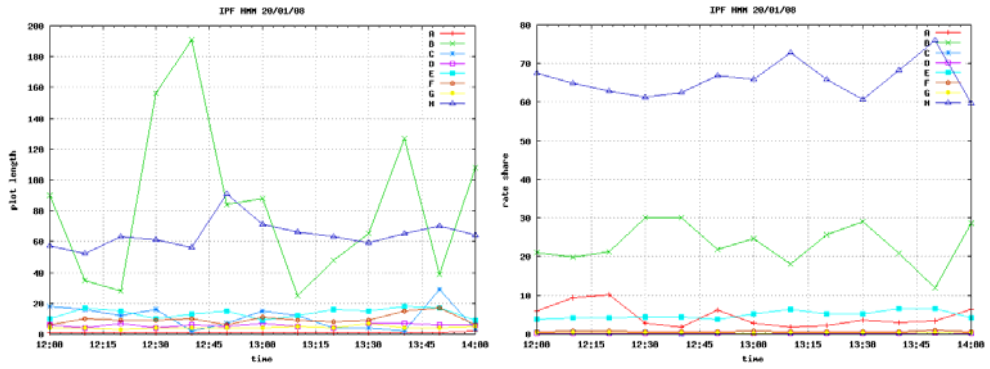
Based on the results on Fig. 2(a) we can make the following conclusions:

- During the normal behavior the highest density belongs to the 'H' symbol. This means fairly high flow heterogeneity.
- The densities of 'B' and 'C' might also be relatively high compare to the other symbols' densities. As we can see 'C' was burst in some of the discrete time intervals, while the 'B' was keeping a plateau rates. As we discussed earlier, the bursts on 'B' and 'C' are acceptable during the normal traffic flows due to an excessive usage of a specific service by one and more clients.

The graphic on Fig. 2(b) shows the values of ϕ_i (T) captured for 24-hours period. The values are percentage of the symbol occurrences over the word length. The 'H' symbol again has the highest share rate. It is over 50% which means it has higher share even than the sum of the other symbols shares. The next share rates are those of the 'A'-s and 'B'-s. The occurrences of 'A'-s might be a long running connection between two particular nodes. Similarly, 'B' symbols mean short term usage of a connection and then opening a new one by the same client but using different client port. That's the way some of the application protocols work. For example, the old HTTP/1.0 does not support keep-alive TCP connections. The client opens a channel, sends the request, retrieves the reply, and finally closes the channel. For each individual request-reply transaction, the protocol requires different TCP channel. Another example could be the DNS queries. Every individual query is provided in a different UDP datagram. Then an individual host might produce too many 'A'-s in a short time if the host is resolving too many names. The common case is the network monitoring software. It scans the network on regular basis in order to build up-to-date topology.

During the network symbols collection we simulated an offensive traffic flow. Fig. 3 shows how that traffic emerges in the network words parameters. The figures present graphically the results within the observed time interval 12:00-14:00. They are respectively addressing the density and the frequency. Note that the suspicious traffic flows are running for relatively short time period and usually with fluctuating

frequency. That's why in order to get the clear picture of the network symbols emission we need to narrow down the observed period within few hours. The results then are made in a shorter time slots aggregation. In this example on the Fig. 3 the observed period of 120 minutes is broken down in 12 timeslots each of 10 minutes. Thus, for the interval 12:30-12:40 we calculated density index 192 for the symbol 'B', which means there was a sequence of 192 'B'-s in the network words for that particular 10 minutes interval.



(a) (b)

Fig. 3. Network symbol density during a suspicious flows

The network alphabet was defined in way to semantically represent the homogeneity grade of the network flows. As many top alphabet symbols (A, B, C) we observe, as more homogeneous the traffic flows are. The opposite is also true. As many symbols from the end of the alphabet we see, as more heterogeneous the traffic is. As we already emphasized, the homogeneous traffic is likely to be machine-generated. Hence, the first alphabet symbols are tokens for suspicious traffic, while the last alphabet symbols are tokens for normal network behavior.

The fairly high density values for the symbol 'B' at Fig. 3(a) are warnings for suspicious traffic running on the wire. Comparing the values against those captured during the normal traffic flows at Fig. 2(a) we can see that the densities are fluctuating in some of the time intervals for the symbol 'B'. However, the symbol 'H' again has also high density values, even though the traffic flows were fairly homogeneous. The frequency graphic on Fig. 3(b) addresses the symbol share rates during the suspicious traffic. The interesting fact is that the 'B'-s went beyond 20%; however, again the 'H' is still the top symbol with frequency rates over 70%. Again, comparing to the corresponding graphic for the normal behavior at Fig. 2(b) we can easily notice that all the frequencies are below the 20% threshold expect the one of 'H'. Nevertheless in some of the discrete time intervals the frequencies of 'A' and 'B' are fluctuating most probably due to the application level protocol specifics, we could be quite convenient that the critical frequency threshold between the machine generated and the regular traffic flow is around 20% share rate. Saying more simple, if we observe some of the first symbols in the network alphabet (say, 'A', 'B', or 'C') having 20% frequency, the traffic is likely to be machine-generated.

To what extend might be helpful the values of the density ρ and the frequency ϕ for the purposes of the traffic analysis? First, those parameters are presenting the entropy grade in a measurable way. The average values for the density of 100 counts for the symbols 'A', 'B', and 'C' together with the frequency rates over 20% will be a token for machine-generated flows and correspondingly for potential malicious network traffic. Table 2 summarizes the critical threshold values for the density and the frequency. Those values might be considered when deciding on the traffic flow characteristics.

Table 2. Critical values for ρ and ϕ of the symbols 'A', 'B', and 'C'

Symbol density	ρ	100
Symbol frequency	ϕ	20%

Once we build and utilize the traffic flow collector together with the analytic engine, we can make network security probes on short time interval basis. Using those probes the tool would be able to generate early alerts for the ongoing suspicious flows. Even though we already defined the critical threshold values for the symbol density and the symbol frequency, it would be even better if the values are tuned up according to the specific network environment.

5. Maximizing the HMM against the observed network words

Prior to apply the Hidden Markov Model (HMM) against the collected network words, let's summarize the Markov Model basics and statements as well as the main algorithms for solving the HMM issues. The Markov Model is a stochastic process, i.e. a process which current state is relatively independent from the previous state:

(1) $P[x(t_n) \le x_n | x(t) \; \forall \; t \le t_{n-1}] = P\,[x(t_n) \le x_n | x(t_{n-1})]$ for each n and $t_1 < t_2 < \ldots < t_n$.

The Hidden Markov Model is a statistical model assuming that the system has unknown parameters for its states transitions but it emits definite number of visible symbol on each discrete time frame. The challenge of the HMM is about how to discover the internal (hidden) states sequence of the system knowing the external (visible) manifestations. In order to formalize the HMM we are using the following definitions of the model elements:

$S = \{S_1, S_2, ..., S_N\}$ the set of the hidden system states

$V = \{V_1, V_2, ..., V_M\}$ the set of the visible symbols emerged by the system state change.

The elements in the set V are called *observed symbols set* or *symbols*. The set V itself is called *a discrete alphabet*. For a particular discrete time period T we will observe T number of symbols. That symbol sequence is also called observed word.

Every hidden Markov model has the following characteristics:

1. N – the number of the internal (hidden) system states. The sequence $Q = q_1 q_2 \ldots q_T$ is the sequence of the discrete states the system has been during the observed period of time [1; T]. With q_t we will define the hidden state the system

has been in the discrete moment of time t. Obviously, q_t ($1 \le t \le T$) belongs to the state set S.

2. M – the number of the symbols that the system emits during the observed period T. The sequence $O = O_1O_2 \ldots O_T$ is the emitted word for the observed period [1; T]. In case of two symbol alphabet (M=2), i.e., V is a Boolean alphabet, we can consider the word O as a Boolean vector. Obviously, the word O is a number in the interval [0 ; M^T– 1].

3. A – the matrix of the hidden states transition probability, $A = \{a_{ij}\}$, where

$$a_{ij} = P\,[\, q_{t+1} = S_j \mid q_t = S_i \,], \qquad 1 \le i, j \le N. \tag{2}$$

In case the model allows any to any hidden state transitions, then $a_{ij} > 0 \;\forall\; i,j$. In the real life examples, however, some of the state transitions are not possible, that's why $a_{ij} = 0$ for some pairs $<i, j>$.

4. B – the matrix of the symbols emission probability for observing the symbol V_k in case the system is staying in the hidden state S_j, i.e. $B = \{b_j(k)\}$, where

$$b_j(k) = P\,[\, V_k \text{ in time } t \mid q_t = S_j \,], \qquad 1 \le k \le M,\ 1 \le j \le N. \tag{3}$$

5. π – the initial vector of the state entry probability, $\pi = \{\, \pi_i \,\}$, where

$$\pi_i = P\,[\, q_1 = S_i \,], \qquad 1 \le i \le N. \tag{4}$$

The hidden Markov model is the triple $\lambda = (A, B, \pi)$. Based on (2), (3) and (4), hence the HMM fulfills the following stochastic constants:

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall\, i,\ 1 \le i \le N, \tag{5}$$

$$\sum_{k=1}^{M} b_j(k) = 1 \qquad \forall\, j,\ 1 \le j \le N, \tag{6}$$

$$\sum_{i=1}^{N} \pi_i = 1. \tag{7}$$

Given the form of HMM, there are three basic problems of interest that must be solved for the model to be useful in the real-world applications. These problems are the following:

1) Given the observation sequence $O = O_1O_2\ldots O_T$ and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(O|\,\lambda)$, the probability of the observation sequence, given the model?

2) Given the observation sequence $O = O_1O_2\ldots O_T$ and a model $\lambda = (A, B, \pi)$, how to choose a corresponding state sequence $Q = q_1q_2\ldots q_T$ which is optimal in some meaningful sense, i.e., best explains the observation O?

3) How do we adjust the model parameters = (A, B, π) in order to maximize the probability $P(O|\,\lambda)$?

The solutions of these three problems are already provided respectively by the Forward-Backward procedure, the Viterbi algorithm, and the Baum-Welch algorithm. The algorithms are well described in the literature and their details are out of the current scope. However, we will need to implement the Baum-Welch algorithm in order to find out the HMM values $\lambda = (A, B, \pi)$ that are maximizing the

observation probability $P(O|\ \lambda)$. The goal is to find out those values that are maximizing the normal network words and those that are maximizing the offensive network words.

5.1. The Baum-Welch algorithm

Leonard Baum and Lloyd Welch have solved the self-evaluating task of the HMM, i.e. the Problem 3 described above. Given the observation sequence O and the initial model λ the goal is to find a new model $\bar{\lambda}$ that is locally maximizing the probability $P\ (O|\lambda)$. In order to describe the algorithm steps, first we will need to define the variables:

$\alpha_t(i)$ – given the model λ and the observation O, the probability to observe the subsequence $O_1O_2\ldots O_t$ and to reach out state S_i at time t;

$\beta_t(i)$ – given the model λ and the observation O, the probability to observe the subsequence $O_{t+1}O_{t+2}\ldots O_T$ in case the system has been in state S_i at time t;

$\gamma_t(i)$ – given the model λ and the observation O, the probability the system to be in state S_i at time t;

$\xi_t(i,j)$ – given the model λ and the observation O, the probability of being in state S_i in the discrete moment t an in state S_j in the moment t+1 in case of sequence O and model λ,

(8) $$\alpha_t(i) = P\ [O_1O_2\ldots O_t\ ,\ q_t = S_i \mid \lambda\ ],$$

(9) $$\beta_t(i) = P\ [\ O_{t+1}O_{t+2}\ldots O_T\ ,\ q_t = S_i \mid \lambda\ ],$$

(10) $$\gamma_t(i) = P\ [\ q_t = S_i \mid O, \lambda\ ] = \frac{\alpha_t(i)\beta_t(i)}{P[O \mid \lambda]} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_i(i)\beta_t(i)},$$

(11) $$\xi_t(i,j) = P\ [\ q_t = S_i\ ,\ q_{t+1} = S_j \mid O,\ \lambda\ ].$$

The event sequence that leads to the conditions in (11) is shown on the Fig. 4. This would mean that we can easily conclude that

(12) $$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P[O \mid \lambda]} = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}.$$

According to the definitions (10) и (12) we can conclude that

(13) $$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j)\ .$$

When we sum $\gamma_t(i)$ over the time variable t, $t \in [0;\ T-1]$, we will find out the probability the system to go through the hidden state S_i except the final stage T as it is not a transitional moment (i.e. it is a final). In same way when we calculate the sum of $\xi_t(i,j)$ over t, $t \in [0;\ T-1]$, then we will receive the probability of making a transitions from S_i to S_j:

$$\sum_{t=1}^{T-1}\gamma_t(i)\ = \text{expected number of transitions from } S_i,$$

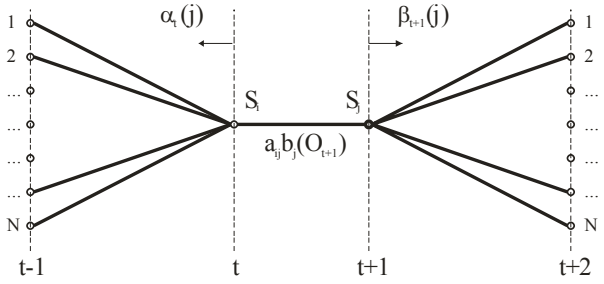


Fig. 4. State transition from S_i to S_j

$\sum_{t=1}^{T-1} \xi_t(i,j)$ = expected number of transitions from S_i to S_j.

Now the question is how we can estimate the initial model λ based on the observation sequence O? In order to address this problem, we have to define a new model $\bar{\lambda}$ ($\bar{\pi}$, $\bar{A}$, $\bar{B}$) in the following way:

(14) $$\bar{\pi}_i = \gamma_i(t),$$

(15) $$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)},$$

(16) $$\bar{b}_j(k) = \frac{\sum_{t=1, O_t=V_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}.$$

Let's emphasize that the sum over $\gamma_t(j)$ at (16) takes in account only those $\gamma_t(j)$ values, where the emitted symbol in time t was exactly V_k.

Initially, we define a HMM with certain parameters λ (A, B, π). Then we use that model for calculating the right side of the equations (14), (15), and (16). After that we can define a new HMM $\bar{\lambda}$($\bar{A}$, $\bar{B}$, $\bar{\pi}$) based on the results of the calculations (25), (26), (27). On the next step we will need to check if the new model $\bar{\lambda}$ can emit the observation O with higher probability, i.e., if $P[O|\lambda] < P[O|\ \bar{\lambda}]$. Baum proofed that either the initial model λ is the critical point of the probability function, or the new model $\bar{\lambda}$ is the better one. Using that iteration on every step we will generate a new model $\bar{\lambda}$ that better generates the observation O. The criterion for the end of the loop will be $P[O|\lambda] = P[O|\ \bar{\lambda}]$. In the real-world implementation, however, we have to look for an acceptable difference between the probabilities $P[O|\lambda]$ and $P[O|\ \bar{\lambda}]$ rather than a full match. Hence, we will define a very small constant δ which would be the equality threshold; if $|P[O|\lambda] - P[O|\ \bar{\lambda}]| < \delta$, then we assume the probabilities are equal. Leonard Baum also proofed that the new model $\bar{\lambda}$ keeps the stochastic constants (5), (6), and (7).

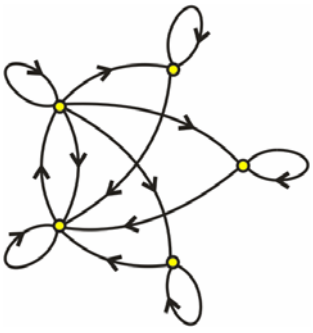
5.2. Implementation issues and a numeric example

According to the network alphabet description as per Table 1 the number of the symbols is eight. Now the question is how to pick up the number of the internal (hidden) states? If we choose some very small number, e.g., 2 or 3, it would be very far from the reality as the networks are very complex systems. If we, however, choose some large number if hidden states, that would be closer to the reality but very difficult for the analysis. Using big 'N' and 'M' values requires higher number of calculations. The problem is not only of the iterations themselves. Nevertheless the algorithm complexity is $O(N^2T)$, we might be able to handle the number of the iterations in some acceptable timeframe. Unfortunately the calculation precision is also an issue using large numbers as the probability values become very close to 0. That's why we would choose the number of the hidden state of 5. Then the number of the iterations is 500. In order to make the difference between the abstraction and the reality, we can imagine the internal states as some set of network phenomena that have the same influence onto the computer networks. For example, the Denial-of-Services attacks are affecting the network traffic in similar way no matter what kind of attack was deployed.

Given the HMM abstraction of the network traffic, we have developed a tool for collecting the traffic data and converting the data into the network symbols. Once we collect the network words, we have made another tool that implemented the Baum-Welch algorithm. The tool is taking as an input an initial HMM values and the observed network word, i.e. O and λ. Based on O and λ the tool calculates a new model $\bar{\lambda}$ and checks if it "better" that the initial. The iteration stops when the new model does not emit the observation O with higher probability.

We have driven the HMM calculation onto 10 different models and over 20 input word with length of 40 symbols. The words were collected during a normal traffic flow and during suspicious flows. Here is a numeric example of the tool results.

The example model describes is shown in Fig. 5.



$$\pi_i = 0.2 \ \forall \ i,\ 1 \le i \le N,$$

$$a_{ij} = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0 & 0.5 & 0 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0 & 0 & 0.5 \end{bmatrix},$$

$$\beta_j(k) = 0.125,\ \forall\ j,\ 1 \le j \le N,\ \forall\ k,\ ,\ 1 \le k \le M$$

Fig. 5. Example HMM

The result of Baum-Welch maximization procedure is shown on the Tables 3 and 4. They present the HMM parameter maximized with observations during a normal traffic and during an offensive traffic flow. As we can see the state transition matrix is slightly changed, however, the symbol generation matrix is

much more fit against the observation. This would mean that the observations (the emitted characters) are the one to express the normal vs. suspicious traffic difference.

Table 3. Normal traffic maximization

Word	HFHHHHHHHHHGHHAHEHAHHHHHBBHHHHHAHEHAHHHH							
A	0.21189	0.22446	0.22446	0.22446	0.11473			
	0.00000	0.42990	0.00000	0.00000	0.57010			
	0.00000	0.00000	0.42990	0.00000	0.57010			
	0.00000	0.00000	0.00000	0.42990	0.57010			
	0.52715	0.00000	0.00000	0.00000	0.47285			
B	0.09500	0.04750	0.00000	0.00000	0.04750	0.01805	0.02375	0.76820
	0.09974	0.04987	0.00000	0.00000	0.04987	0.04738	0.02494	0.72821
	0.09974	0.04987	0.00000	0.00000	0.04987	0.04738	0.02494	0.72821
	0.09974	0.04987	0.00000	0.00000	0.04987	0.04738	0.02494	0.72821
	0.10361	0.05181	0.00000	0.00000	0.05181	0.01230	0.02590	0.75457
Π	0.20000	0.20000	0.20000	0.20000	0.20000			
$P\,[O \mid \lambda]$	3.80×10^{-11}							
State sequence	44444444444444444444444444444444444444							

Table 4. Offensive traffic maximization

word	BBBBABABBBBABFCFBBBGGBFHGHHHHBBBBBBHHBAB							
A	0.20038	0.19996	0.19996	0.19996	0.19975			
	0.00000	0.49960	0.00000	0.00000	0.50040			
	0.00000	0.00000	0.49960	0.00000	0.50040			
	0.00000	0.00000	0.00000	0.50011	0.49989			
	0.49874	0.00000	0.00000	0.00000	0.50126			
B	0.09628	0.57122	0.02375	0.00000	0.00000	0.07125	0.07125	0.16625
	0.09839	0.55253	0.02493	0.00000	0.00000	0.07480	0.07480	0.17454
	0.09839	0.55253	0.02493	0.00000	0.00000	0.07480	0.07480	0.17454
	0.09839	0.55253	0.02493	0.00000	0.00000	0.07480	0.07480	0.17454
	0.10379	0.53356	0.02590	0.00000	0.00000	0.07771	0.07771	0.18132
π	0.20000	0.20000	0.20000	0.20000	0.20000			
$P\,[O \mid \lambda]$	2.17×10^{-13}							
State sequence	14444444444444444444444444444444444444							

6. Conclusion

The state-of-the-art network devices are providing network flow export to the network monitoring systems. The vendors have already implemented the standards for the exported data. Besides the basic network measurements and flow baselines we can utilize that data in a way to build also security analysis. In this paper we have discovered a simple approach how to “extract” the security characteristics of

the network flows. We used a Markov model definition for creating a flow model. The model was based on the relation between the previous and the current flow entry exported by the network device. Once the observed symbol definition was made we developed a tool for collecting the data and emitting the network symbols according to the alphabet definitions. The collected network words were analyzed against two characteristics: the symbol density and the symbol frequency previously defined. The results we receive clearly emerged the thresholds between the normal flows and the machine generated flows.

On the next step we have picked up a hidden Markov model that would be convenient for describing the network flow behavior. Even though the model is far from the complex network organism, we can use it as a high-level network process modeling. We have applied the Baum-Welch (BW) algorithm against the model and two types of observation: normal flow and suspicious flow observations. We showed the network words were the one to control the results of the BW procedure. The maximized model has changed mainly in the B matrix that typically proofed the observed symbols meaning rather than the actual internal states (network processes) semantics.

There are two major benefits of that simple flow analysis. First, the network security analyst might be able to easily apply early alerts on the network segments. After the analysis tool is deployed in the critical network segments it can be used for notifying the administrators or the network monitor stations for a suspicious network behavior. The second benefit is the security profiling of the corporate network. The network words could be collected over longer time period and stored for further profile analysis. If we span the observation time over a week, we will be able to profile the flows against the business hours. Respectively, running the observation over a month, quarter, or even a year would build not only the flow profile but also will establish trends measurements for a longer period.

References

1. C l a i s e , B. Cisco Systems NetFlow Services Export Version 9: RFC 3954. October 2004.
2. D a i g l e, J. Queuing Theory with Applications to Packet Telecommunication. Springer, 2005.
3. R a b i n e r, L. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. – In: Proc. of the IEEE, **77**, 1989, No 2, 257-286.