Network Flow Security Baselining

Tsvetomir Tsvetanov[1], Assoc.Prof.Dr. Stanislav Simeonov[2]

[1]Sofia University
James Boucher Blvd 5
Sofia, Bulgaria
[2]Burgas Free University
San Stefano Blvd 62
Burgas, Bulgaria
ttsvetanov@gmail.com
stan@bfu.bg

Abstract: Networks are a critical factor in the performance of a modern company. Managing a network is as important as managing any other aspect of a company's performance and security. There are many tools and appliances for monitoring the traffic and analyzing the security aspects of the network flows. 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 baselining that might enable early alerts. This paper is focusing on the security baselining based on a simple flow analysis utilizing the flows measurements and the theory of the Markov models.

1 Introduction

Two major approaches are utilized in the theory of computer network modeling: the queuing theory [Da05] and the hidden Markov models [Ra89]. The queuing theory addresses basically 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. This issue is the main objective of the ubiquitous service. Apparently, the queuing theory is providing an approach for analysis of the quality of service (QoS) issues. QoS was addressed in the early days of the computer networking and it is still ongoing issue for the modern telecommunications. The one aspect of the QoS problem is the quality measurement. The quality assessment might be done in different ways: measuring the delay of the system access, the time frame for the data access, taking an account the quantity of the lost information, the voice signal distinction or the data purity. Usually the quantity measurements are random variables and the results are the average rates of the variable distributions. The major issue regarding the QoS delivery is the price for the quality that the

end user would agree to pay. The price generally depends on the resources needed for providing the service quality level. The QoS analysts are focused on that problem, building up the optimal share between the quality rate and the price of the service.

For more than three decades there was a trend against ubiquitous service in the packet-switching networks. The main reason for that was the price decrease with simply increasing the capacity of the shared resources. On the other hand, there was no mechanism in place that might provide certain service quality level on a reasonable price. Then the queuing theory has been adopted as the main approach for solving that problem for building the optimal rate between the shared resources value and the quality of the service that the end user achieve. The queuing theory also aided the analysts for discovering those internal network processes that allocate the significant part of the network resources. Once knowing the processes, the analysts might be able to decide on the policy of the resources allocation in order to ensure the target level of the service quality.

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.

Unlike the queuing theory approaches, the network analysis based on the Markov models are not addressing the quality of service issues; it rather solves the problem for pattern recognition onto the network flows. Using the HMM the analysts are building different kind of network profiles for the different network environments. Those profiles aid the analysts in abnormal network behavior detection as well as for finding specific trends in the network environment utilization. The profiles also might be applied in network security management. When building a set of profiles within certain timeline, the analysts are able to determine the differences and then detecting potential threads.

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. Due to the internal processes the network is moving to another state within certain discrete time interval. Every internal state transition emits external phenomena in means of network traffic attributes like specific header values, payload content, combination of both, or specific sequence of network packets on the wire. All those manifestations might be different for the particular infrastructures, environments and also might vary between the network segments within an autonomous system.

The comprehensive network devices are able to provide some predefined characteristics of the network flow to the network monitors. That information, of course, 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. Using that information the network monitors are able to collect traffic data, i.e. the network symbols emitted by the internal processes.

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.

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.

Table 2.1: The network alphabet semantics

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 [Cl04].
- 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.

Figure 3.1 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.

Figure 3.1: Networks words captured during a normal network behavior

The next figure 3.2 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 figure 3.2 as quasi-homogeneous flow.

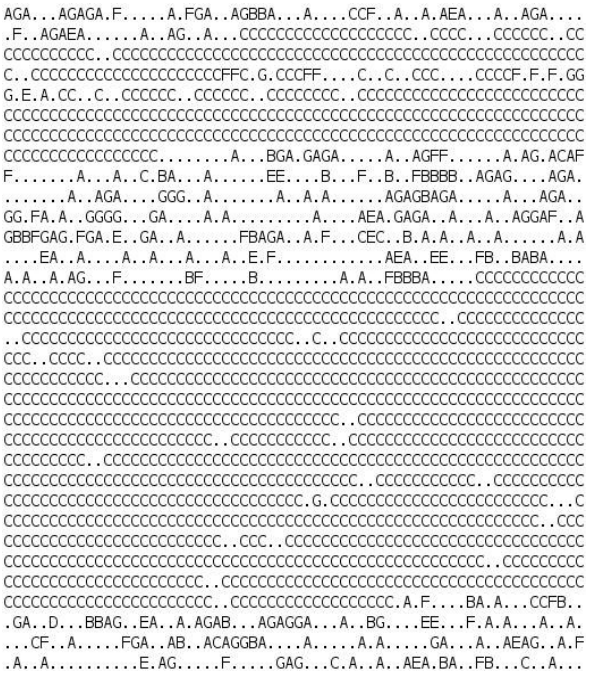

Figure 3.2: Network words emitted during suspicious traffic flows

The example on figure 3.2 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 2.1. 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 figures 4.1, 4.2, 4.3, and 4.4 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 $1 \leq t \leq T$. 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.

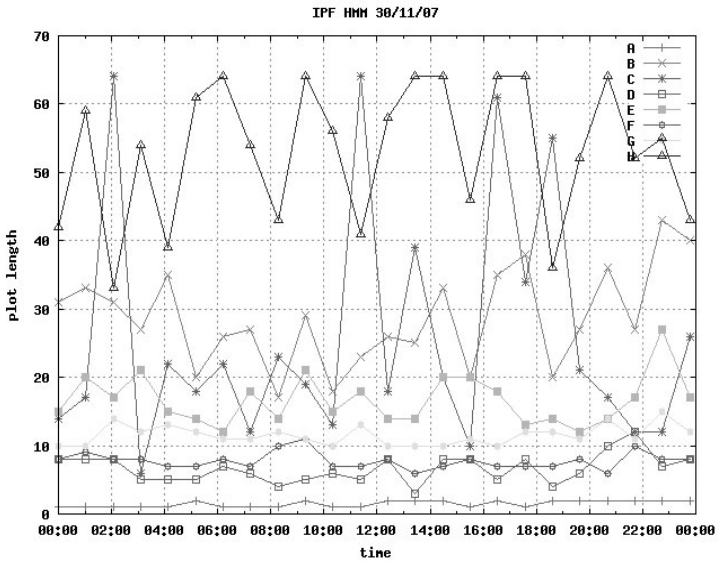

Figure 4.1: Density of the network symbols during the normal network behavior.

Based on the results on figure 4.1 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 figure 4.2 shows the values of ϕ_i (T) captured for 24h 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.

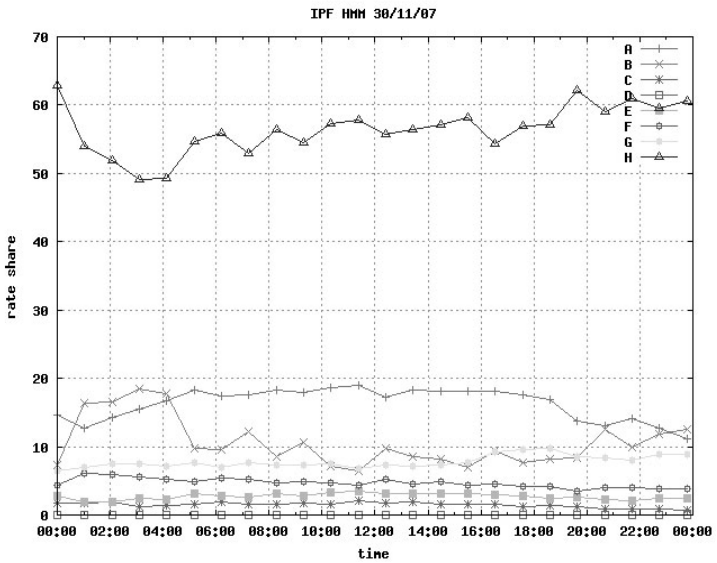

Figure 4.2: Frequency of the network symbols during the normal traffic flow.

During the network symbols collection we simulated an offensive traffic flow. The figures 4.3 and 4.4 show 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 figures 4.3 and 4.4 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.

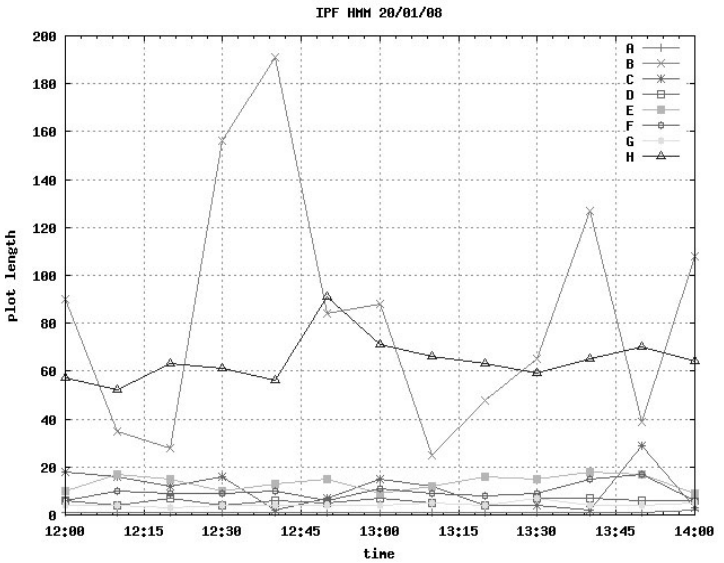

Figure 4.3: Network symbol density during a suspicious flows.

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 figure 4.3 are warnings for suspicious traffic running on the wire. Comparing the values against those captured during the normal traffic flows at figure 4.1 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 figure 4.4 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 figure 4.2 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.

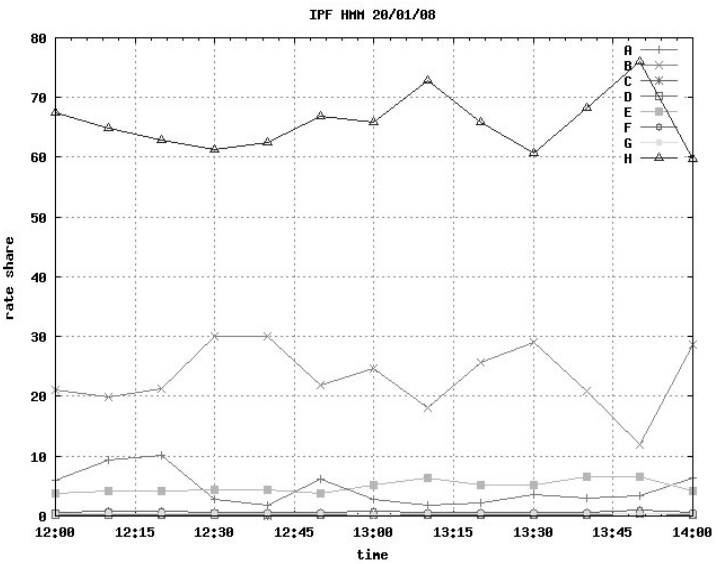

Figure 4.4: Network symbol frequency during a suspicious flows.

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 4.5 summarizes the critical threshold values for the density and the frequency. Those values might be considered when deciding on the traffic flow characteristics.

Symbol density	ρ	100
Symbol frequency	ϕ	20%

Table 4.5: Critical values for ρ and ϕ of the symbols A, B, and C

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 Conclusion

The contemporary network appliance is 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.

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.

Bibliography

[Cl04] Claise, B.: Cisco Systems NetFlow Services Export Version 9: RFC 3954. Oct 2004.

[Da05] Daigle, J.: Queuing Theory with Applications to Packet Telecommunication. Springer, 2005.

[Ra89] Rabiner, L.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 77 (2), 1989; p. 257–286