Serdica J. Computing **4** (2010), 385–416

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

COMPUTER NETWORKS SECURITY MODELS
A New Approach for Denial-of-Services Attacks Mitigation[*]

Tsvetomir Tsvetanov

Abstract. Computer networks are a critical factor for the performance of a modern company. Managing networks is as important as managing any other aspect of the company's performance and security. There are many tools and appliances for monitoring the traffic and analyzing the network flow security. They use different approaches and rely on a variety of characteristics of the network flows. Network researchers are still working on a common approach for security baselining that might enable early watch alerts. This research focuses on the network security models, particularly the Denial-of-Services (DoS) attacks mitigation, based on a network flow analysis using the flows measurements and the theory of Markov models. The content of the paper comprises the essentials of the author's doctoral thesis.

ACM Computing Classification System (1998): C.2.0.

Key words: network security, denial-of-services, pattern-based defense mechanism.

[*]This article presents the principal results of the doctoral thesis "Computers Networks Defending Models" by Tsvetomir Tsvetanov (St. K. Ohridski University of Sofia), successfully defended before the Specialised Academic Council for Informatics and Mathematical Modelling on 25 May, 2009.

1. Network Security—Issues and Challenges. In the last few years severe issues emerged in the information security field that had to be addressed immediately. Network researchers focused on developing new technologies which would be able to protect computer networks from the increasing number of security threats. Regardless of the great development in that area, today there is still more to do and the need for a breakthrough solution or technology is still on the table. Some computer network security researchers ([1], [2]) outlined four major issues of the global Internet security:

Issue 1: Internet security is highly interdependent. DoS attacks are commonly launched from systems that are subverted through security-related compromises. Regardless of how well secured the victim system may be, its susceptibility to DoS attacks depends on the state of the security in the rest of the Internet.

Issue 2: Internet resources are limited. Each Internet host has limited resources that can be consumed by a sufficient number of users.

Issue 3: The power of many is greater than the power of few. Coordinated and simultaneous malicious actions by some participants can always be detrimental to others, if the resources of the attackers are greater than the resources of the victims.

Issue 4: Intelligence and resources are not collocated. An end-to-end communication paradigm led to locating most of the intelligence needed for service guarantees with end hosts. At the same time, a desire for large throughput led to the design of high bandwidth pathways in the intermediate network. Thus, malicious clients can misuse the abundant resources of unwitting network for delivering numerous messages to a victim.

These days we have ended up with many concepts and solutions for protecting the networks. All these provide deployment of a protection system within a single computer host or distributed deployment via server and agents. There are, however, few solutions that are focused on ubiquitous protection including all computer workstations, network segments (Intranet and Extranet), as well as protection of the global network. The following are the known systems for informational security that cover a certain perimeter of the enterprise information systems and networks: antivirus systems, Intrusion Detection Systems, Intrusion Prevention Systems, and Traffic filtering systems. Every vendor emphasizes on a specific subset of vulnerabilities or on a few typical traffic parameters related to network security. However, none of these approaches is ubiquitous. Typically they would require configuring and using the individual security capabilities of the network appliance (routers, switches, etc.) in order to achieve robust network security. Again, the protection provided by the firmware (network operating system) is focused on certain traffic data requiring very good understanding on

part of the administrator about what does need protection and how it is to be protected.

1.1. Goals and Problems. The ultimate goal of the thesis is to research the existing methodologies in network security and to identify the issues with poor or no effective approach for providing an acceptable level of protection. As many of the security issues belong to the Denial-of-Services attacks, we focused our major goal in that area. In order to address the DoS problem, we defined the following tasks and problems for solving:

- Research the state-of-the-art methodology and approaches for predictive analysis of the computer networks; research the variety of DoS attacks and create taxonomies depending on the features of the attacks and defense mechanisms.
- Analyze, select, and develop a methodology for a critical subset of DoS attacks and apply that methodology using a software system based on that model.
- Develop a system prototype, run field tests in the network environment, analyze the results, and finally compare the results with similar existing methodology and approaches.

1.2. Methodology of computer networks analysis. The analysis center at CERT outlined a few challenges of predictive network analysis [3]. The analysts summarized five major problems for computer network traffic analysis for a successful prediction of threats:

- Establishing perspectives
- Establishing a base line profile of normal behavior
- Spotting and evaluating exceptional behavior
- Producing analysis results that support decision making
- Assessing the effectiveness of the actions

Analysts face the challenge of choosing among the perspectives from which to observe network behavior [3]. Four perspectives are listed below and shown at Figure 1.

Local perspective: Observing network behavior at the connection(s) between an organization's network and the Internet or another wide-area network, often a perimeter firewall. This is the easiest perspective to obtain. The advantage is that the threatening behavior identified is specific to the organization. The disadvantage is that there is little (if any) time for protective action, leaving only reactive strategies to the organization.

Proximate perspective: Arranging for observation at the wide-area net-

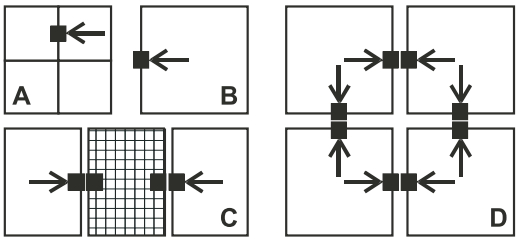


Fig. 1. Perspectives: (a) local, (b) proximate, (c) remote, (d) endemic

work point of presence. In general, this involves contract negotiations to establish what observations will be made and how the data will be transmitted. Advantages are that threatening behavior is still somewhat specific to the organization, and that a somewhat broader observation than the local perspective is possible. Disadvantages are the need for negotiation and the limited time for anything but reactive strategies.

Remote perspective: Arranging for a variety of points of observation at contracted points on the wide-area network, not necessarily related to a particular organization. The contract negotiations are likely to be more complicated than for the proximate perspective since the needs of a variety of organizations are involved. An advantage is the ability to generalize, along with a degree of analytical context. Disadvantages include varying amounts of data covering the same event and varying time lags; in addition, the task of organizing and coordinating the data is likely to be extensive.

Endemic perspective: Building a fostered cooperative network of allied network analysis groups. This allows a variety of perspectives on network behavior and a shared understanding of the analysis process and its results. Advantages are a fair amount of collaborative understanding and the ability to bring in the local expertise of a variety of organizations. It is also possible that even subtle threats would be detected with a maximum of time for protective action. Disadvantages include the difficulty of localizing threats to a specific organization and the possibility that differing assumptions of the analysis may lead to contradictory recommendations.

Along with establishing perspectives, analysts need to understand the normal behavior associated with these perspectives. They have to establish a baseline for later identification of exceptional behavior. Once some initial profiling and analysis has been done, analysts can move to more in-depth analysis. This involves looking for trends and cycles, incorporating data from multiple sources, and seeking triggering events or conditions for variations.

Regarding the spotting and evaluating of exceptional behavior, the fol-

lowing are among the factors that must be considered: time; source and victim; growth of incidents related to a specific rationale; and the perpetrators. The decision support role for predictive analysis involves a) understanding the various courses of action available in response to an alert along with their consequences, b) matching the threat to possible defensive actions, and c) providing those courses of action in a strategic context to decision makers.

To assess this effectiveness, analysts need to formulate criteria for a successful defensive action, collect observations to confirm or deny these hypotheses, test the hypotheses against the observations, and finally balance between the two general types of observational error: identifying a change that, in fact, does not exist and failing to identify a change that does exist.

1.3. Methodology for Detecting and Preventing Denial-of-Services Attacks. Searching the Web we would find hundreds of definitions of the denial-of-services (DoS) attack and their variations like distributed DoS (DDoS), distributed reflected DoS (DRDoS), permanent DoS (PDoS) and many others. However, none of those would be good enough to formally describe the whole variety of DoS attacks. In this paper we use an informal definition provided by US-CERT that basically addresses the end-users experience:

"In a *denial-of-service attack*, an attacker attempts to prevent legitimate users from accessing information or services. By targeting your computer and its network connection, or the computers and network of the sites you are trying to use, an attacker may be able to prevent you from accessing email, websites, online accounts (banking, etc.), or other services that rely on the affected computer." [4].

"In a *distributed denial-of-service attack*, an attacker may use your computer to attack another computer. By taking advantage of security vulnerabilities or weaknesses, an attacker could take control of your computer. He or she could then force your computer to send huge amounts of data to a website or send spam to particular email addresses. The attack is distributed because the attacker is using multiple computers, including yours, to launch the denial-of-service attack." [4]. Figure 2 illustrates the DDoS network hierarchy. In DDoS the intruder engages other platforms as stepping stones for attacking the victim.

The DoS targets a victim in order to block access rather to steal and misuse information. Technically, the DoS could be used as a stepping stone for information theft and frauds; however, the DoS attack itself does not do that. The DoS could also be very destructive depending on the way it was performed and on its targets. This is the reason why DoS attacks are one of the most critical issues of the computer network security. Network analysts and researchers have developed many methodologies and systems in order to address the mitigation and prevention of the DoS problem. But the variety of this problem is so diverse

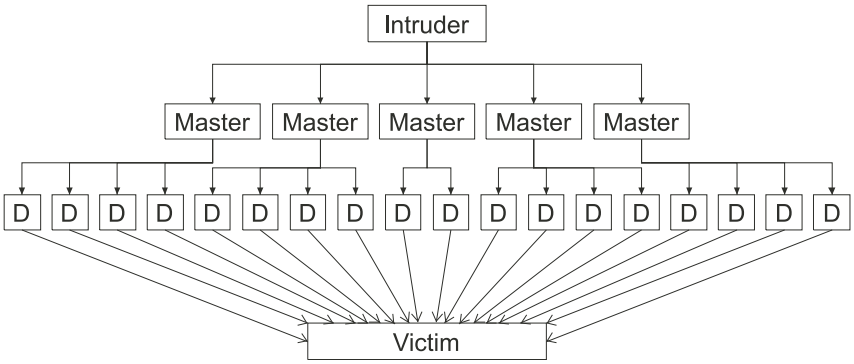


Fig. 2. Attacking network hierarchy

that no solution was able to span all over it.

The first set of problems defined in our thesis focused on the analysis of DoS attacks and the overview of the current methodology:

Develop a structural approach for DoS analysis based on the common taxonomy of DoS attacks and DoS defending mechanisms in order to get a clear understanding of the DoS problem [1]. Also, find out the most common characteristics of the DoS attacks with the greatest impact on the network.

Summarize the state-of-the-art mechanisms for defending against the DoS and the essentials of the approaches on which they are based.

The DoS taxonomy [1] addresses the following major issues and aids the analysts in the following directions:

- The taxonomy synthesizes the common characteristics of the DoS attacks and provides a common language and terminology for analysts.
- It leverages the communication and the cooperation between analysts for building up successful DoS defense mechanisms.
- It identifies the weak points of the DoS problem where many gaps are still not filled and where no solution has been found or even explored.

In our thesis we made a detailed overview of the most successful mechanisms for DoS defense. Most of them are research topics or prototypes in incubus phase or suggestions for design improvements of network protocols. However, they are all attempts to address the DoS phenomena globally: Pushback [5], Traceback [6], ICMP Traceback [7], D-WARD [8], NetBouncer [9], SOS [10], Proof-of-Work System [11], DefCOM [12], COSSACK [13], Pi [14], Stateless Internet Flow Filter (SIFF) [15], Hop-count Filtering system [16], IP Source Tracker [17], Locality and Entropy [18].

Summarizing the DoS characteristics together with the variety of solu-

tions in the context of the OSI network model and the deployment location, we presented graphically the information security systems as shown on Figure 3. The DoS problem, however, spans all over the OSI stack and has local impact on particular hosts and network segments in addition to being able to affect the global networks (Figure 4). This is why there is no easy technical approach that can address the whole impact area.

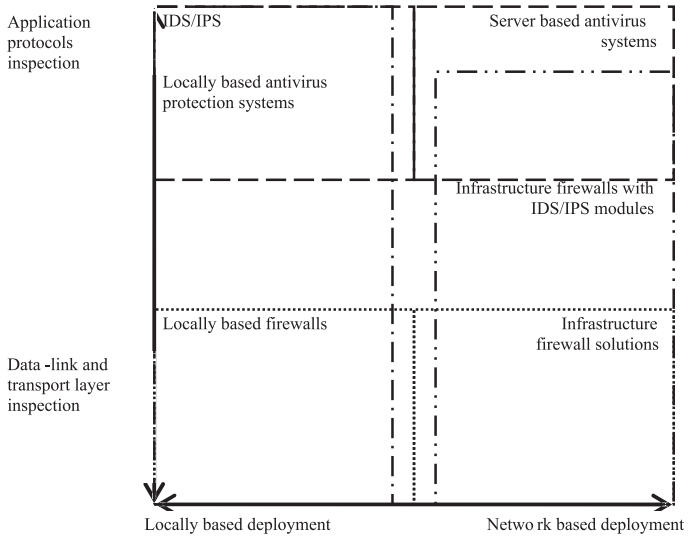


Fig. 3. Information security systems according to the deployment location and the protection level

2. DoS Attack Detection and Prevention—Analysis, Design, and Prototype. The major part of our thesis was focused on researching the opportunities for faster detection of DoS attacks based on a certain methodology, designing and prototyping a system that implements the methodology. On the basis of the analysis of the DoS problem as well as the findings about the attacks with the greatest impact, it was decided that the most important characteristics for detecting severe DoS were the traffic parameters of OSI layers 3 and 4 (network layer and transport layer, respectively). This is why our research and system design started with the following major traffic data: *transport protocol identifier (*$ICMP = 1$*,* $TCP = 6$*,* $UDP = 17$*), source IP address, source port number, destination IP address, destination port number*. In the case of ICMP the port numbers are calculated based on the ICMP type and code values: $p = 16t + c$,

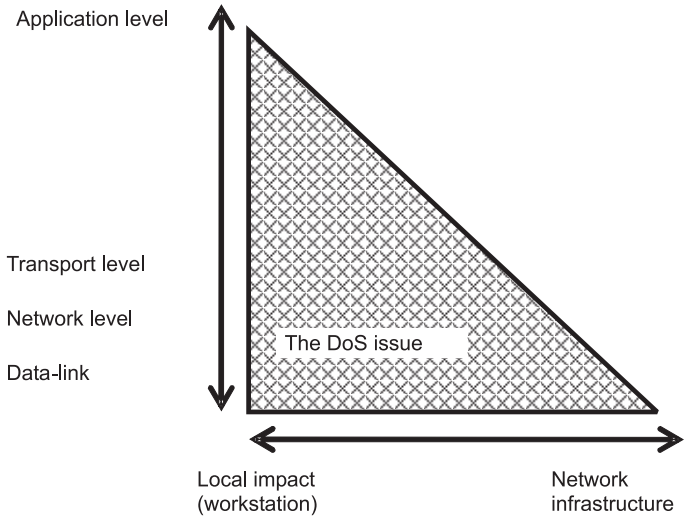


Fig. 4. DoS threats in the context of the OSI stack and the destination target

where t is the ICMP type and c is the ICMP code as per RFC792. Obviously the calculation in the program would be $\mathsf{p} = \mathsf{t} \ll 4| \mathsf{c}$;

These five traffic parameters are part of the IP header and the transport layer header. The IP network protocol was chosen because it is practically the sole network protocol in the global networks. For the purposes of our research it would be absolutely sufficient to focus on the headers data related to the peers over the IP communication.

The following are the tasks defined in the corresponding chapter of our thesis:

- How we can obtain the necessary IP data?
- What methodology to choose for modeling the network traffic?
- What defense model should be preferred for prototyping and how to apply that methodology in our system?

2.1. Prerequisites and Input Data. The modern network solutions provide appliances for network data capture and analysis. There are basically two types, depending on the deployment: inline and passive. With the inline deployment the device would be able to modify the traffic; the passive deployment makes sense in the case of traffic monitoring, trending, and reporting. About ten years ago the traffic monitoring solutions were mostly software based. The administrator needed to set up a dedicated host running the monitoring software in a certain point of the network and collect the data. A few years ago the

network vendors agreed on a common standard for providing network monitoring data. NetFlow [19], which initially was Cisco proprietary, was finally published as industry standard under RFC3954.

NetFlow enables the network appliance to track unicast IP packets as they enter the router through an interface. As the name implies, NetFlow tracks IP packets on a "per flow" basis. A flow is made up of unidirectional flow of data having two endpoints as individually identified by a combination of the following seven criteria items:

– Source IP address
– Destination IP address
– Source port number (TCP, UDP)
– Destination port number (TCP, UDP)
– Layer 3 protocol type (TCP, UDP, ICMP)
– Type of Service (ToS) byte (0-7)
– Input logical interface

Any difference in these seven criteria distinguishes one flow from another.

The NetFlow standard was an answer of the first question—how we would capture the IP flow data. Next, we need to decide on the IP traffic modeling methodology.

2.2. Modeling the Network Traffic. Two major approaches are utilized in the theory of computer network modeling: queuing theory [20] and hidden Markov models [21]. Queuing theory addresses basically the models for the ubiquitous service. The general issue of the contemporary online services is how we can ensure that every customer would be served in a fairly short timeline. This issue is the main objective of the ubiquitous service. Apparently, queuing theory provides an approach for analysis of quality of service (QoS) issues. QoS was addressed in the early days of computer networking and it is still an ongoing issue for modern telecommunications. One aspect of the QoS problem is the quality measurement; the other aspect is related to the analysis of network processes. Queuing theory also aided analysts in discovering those internal network processes that allocate the major part of the network resources. Once knowing the processes, analysts might be able to decide on the policy of the allocation of resources in order to ensure the target level of the quality of service.

The seconds approach uses Hidden Markov Models (HMM). In fact, queuing theory also uses Markov chains and models, but we address network analysis based on Markov models only. Hidden Markov models are stochastic systems that have two parts: an internal hidden part and a visible part that manifests the hidden behavior to the observer. The internal states of the model are hidden from the observer. We call them also *hidden states*. The system moves from state

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

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

We will focus on modeling the network flow for network behavior pattern recognition. In order to address this problem we choose the HMM approach. Having built up the model patterns we would be able to recognize if our network is handling 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 from the end-users and even from the monitoring tools. Those processes affect the network resources and change the behavior of the infrastructure. Due to the internal processes the network moves to another state within certain discrete time interval. Every internal state transition emits external phenomena in terms of network traffic attributes like specific header values, payload content, a combination of both, or a specific sequence of network packets on the wire. All those manifestations may be different for particular infrastructures and environments and also may vary between 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 address 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.

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 obtain 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 a long running intensive network flow anomaly when the there might be Denial-of-Services or similar attacks 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 taken into 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 semantics of the network alphabet.

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.

Figure 5 shows the result of collecting the network words over the wire during normal and abnormal behavior. As expected, the symbol 'H' emerged mostly with heterogeneous traffic, which basically occurs as the normal network flow.

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

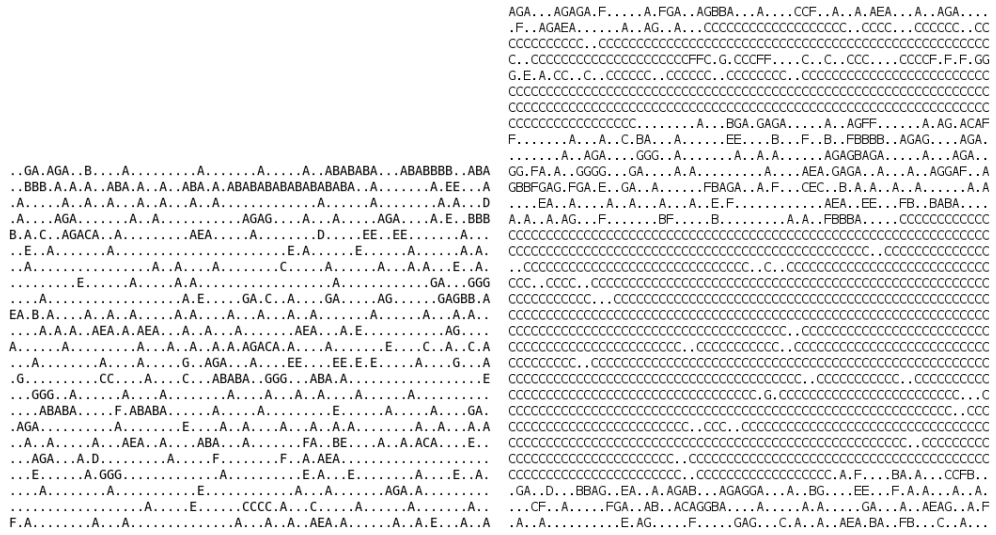
Let the set of observed symbols be $V = \{V_1, V_2, dots, V_M\}$. Set V is the network alphabet as previously defined in Table 1. As per the definitions the size

Table 1. Information security systems according to the deployment location and the protection level

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 hold the same IP address and port pair for one peer, either source or destination, but the other peer IP address and port do 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 a different client as you can see in this example of flow entries: **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 in 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 in common.

of the network alphabet is 8, i.e., $M = 8$. There are two major characteristics that we would be interested in during the pre-modeling analysis. The first question to answer is how long the longest sequence of same symbols is, no matter what the symbol itself. The next problem is what is the symbols share over the discrete period of time of the observation. For these reasons we introduce two definitions over the network alphabet set V.

Definition 1. *The longest sequence $\rho_i(T)$ of identical symbols V_i in the*



(a) (b)

Fig. 5. Networks words captured during a normal network behavior (a) and during suspicious traffic flows (b)

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, $\rho(T) = \{\rho_1(T), \ldots, \rho_M(T)\}$, *is called density of the network alphabet* V *for the observation period* T.

Definition 2 *The ratio* $\phi_i(T)$ *of the number of the occurrences of symbol* V_i *for the discrete period of time* T *to 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 symbols. The overall field tests were made over 24-hour periods of time. The results were collected in series of files and then we proceeded with the flow analysis.

Figures 8 and 9 summarize the results of the analysis. We processed the network words as follows. 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 hour of the day. Then we picked up the words and for each

of them we calculated the density and frequency for each of the network symbols. At the end we had 24 values of the arrays $\rho_i(t)$ and $\phi_i(t)$ 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 into a graphical format. These figures present the values of $\rho(T)$ and $\phi(T)$ during the normal flows and during the suspicious flows.

To what extent might the values of the density ρ and the frequency ϕ be helpful for the purposes of the traffic analysis? First, these parameters present the degree of entropy 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 accordingly for potential malicious network traffic. Table 2 summarizes the critical threshold values for the density and frequency. These values will be considered when deciding on the characteristics of the traffic flow.

Table 2. Critical values for density and frequency of the symbols A, B, and C

Symbol density	ρ	100
Symbol frequency	ϕ	20%

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

Beside the critical values of the symbol density and symbol frequency, we will also be interested in finding a probability model which will match most precisely the observed network words. Only the Hidden Markov Model can be such a stochastic model.

2.4. Maximizing the Hidden Markov Model against the Observed Network Words. In the next step of the analysis, before applying the HMM against the collected network words, let us summarize the HMM basics and statements. The *Markov Model* is a stochastic process—a process whose current state is relatively independent from the previous state:

$$(1)\quad P[x(t_n) \leq x_n | x(t) \forall t \leq t_{n-1}] = P[x(t_n) \leq x_n | x(t_{n-1})] \quad \text{for each } n \text{ and } t_1 < t_2 < \cdots < t.$$

The Hidden Markov Model is a statistical model assuming that the system has unknown parameters for its state transitions but emits a definite number of

visible symbols in each discrete time frame. The challenge of the HMM is how to discover the sequence of internal (hidden) states of the system knowing the external (visible) manifestations. In order to formalize the HMM we use the following definitions of the elements of the model [24]:

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

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

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

Every hidden Markov model has the following characteristics:

- N—the number of internal (hidden) system states. The sequence $Q = q_1 q_2 \ldots q_T$ is the sequence of the discrete states that the system was in during the observed period of time $[1;T]$. With q_t we will denote the hidden state that the system was in at the discrete moment of time t. Obviously q_t $(1 \leq t \leq T)$ belongs to the state set S.
- M—the number of symbols that the system emits during the observed period T. The sequence $O = O_1 O_2 \ldots O_T$ is the emitted word for the observed period $[1;T]$. In the case of a two-symbol alphabet $(M = 2)$, i.e., V is the Boolean alphabet, so the word O is considered a Boolean vector. Obviously the word O is a number in the interval $[0; M^T - 1]$.
- A—the matrix of the transition probability of the hidden states, $A = \{a_{ij}\}$, where

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

  In case the model allows any-to-any hidden state transitions, $a_{ij} > 0$, $\forall i, j$. In the real life examples, however, some of the state transitions are not possible, therefore $a_{ij} = 0$ for some pairs $\langle i, j \rangle$.
- B—the matrix of the symbol emission probability for observing the symbol V_k in case the system stays in the hidden state S_j, i.e. $B = \{b_j(k)\}$, where

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

- π—the initial vector of the state entry probability, $\pi = \{\pi_i\}$, where

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

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

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

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

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

In essence, the HMM maximizing procedure provided by Baum-Welch is used for discovering the optimal values of the model (A, B, π) for certain observations, or in other words it maximizes the probability $P[O|\lambda]$. The Baum-Welch algorithm is an iterative procedure for optimization of HMM values. On each iteration a new HMM $\bar{\lambda}(\bar{A}, \bar{B}, \bar{\pi})$ is calculated. Baum has proved that either the initial model λ is the critical point of the probability function, or the new model $\bar{\lambda}$ is the better one. So if the new model does not improve the observation probability, $P[O|\lambda] \geq P[O|\bar{\lambda}]$, then the algorithm stops. From the technical software perspective, we will rather not compare the probabilities but the difference

$$|P[O|\lambda] - P[O|\bar{\lambda}]| < \delta,$$

where δ is a very small constant (equality threshold).

The Baum-Welch procedure was applied over ten different models. The results published in [24] outlined the following two findings:

- The probability of observing the symbols is maximized over all the internal (hidden) states. No matter what the matrix of the values of symbol emission probability of the initial model λ, the values f the optimal model λ' are maximized for those symbols that were emitted most. This proves the criticality of what is observed at the traffic flows versus what actually happened internally in the network infrastructure.
- No matter what the HMM initial vector values π, the maximized model strongly depends on the input word. As a result, the maximized model has higher values $b_j(k)$ for those symbols V_k that had highest emissions in .

These findings basically ruled out the importance of the density and frequency variables before the characteristics of the hidden network processes. This would that mean the external manifestation of the internal processes is the important information we need to look at when analyzing the potential network threats. The next paragraph introduces the model and the approach we choose for system prototyping. Due to the importance of the IP headers data (the network flow), our model is based on recognition of IP flow patterns, hence, we named it *IP Flow Patterns*.

2.5. IP Flow Patterns. For addressing more precisely and adequately the analysis results above, we introduce the IP Flow Patterns (IPFP) concept as a potential solution for most DoS issues. The mechanism is based on administratively defined patterns following strict semantics. It implies a software tool that implements the pattern semantics. That software program would be started at the monitoring host where the enterprise traffic is accounted. The program receives the traffic characteristics, exports them in readable raw data format, and checks against the defined pattern. Since the algorithm is flexibly constructed it is easy to implement any traffic handler against the pattern check modules.

2.5.1. Pattern Definition, Rules, and Semantics. A pattern is a set of statements describing a characteristic of the network traffic:

P={ S_1, S_2, ..., S_n}

S=¡K,{ V_1, V_2, ..., V_k}¿

where P—pattern, S—statement, K—key, V—value

The statements are unique by their keys. If several statements are defined with the same key, the last one is the valid statement (the previous ones are ignored).

Rule 1: Network traffic matches a pattern if and only if it matches all the statements defined in the pattern. (Conjunction rule)

Rule 2: Network traffic matches a pattern statement if it matches one of the values defined under the statement key. (Disjunction rule)

Rule 3 (match-all rule): When a statement is not defined, it matches all network traffic entries.

The pattern semantics are defined by the following Extended Backus-Naur Form:

<IPFlow_pattern_config_file> ::= {¡DoS_pattern>}+ {¡comment_text>}*
<IPFlow_pattern> ::=
'Pattern' <pattern_number> '(' <EOL>
'proto:' <protocol_name> <endl>
{<protocol_specific> <endl>}?
{('src-addr:' | 'dst-addr:') <ip_address_set> <endl>}*

{'src-addr-seq' <endl>}?
{'dst-addr-seq' <endl>}?
{'redistribute' <endl>}?
{'frequency:' <freq> <endl>}?
{'description:' <text> <endl>}?
{'seqlen:' <uint8> <endl>}?
{<comment_text>}*
')' <EOL>
<comment_text> ::= '#'<text><EOL>
<pattern_number> ::= <uint8> (* number 1..255 *)
<protocol_name> ::= 'tcp' | 'udp' | 'icmp'
<freq> ::= <uint16> | 'no-limit'
<protocol_specific> ::= <icmp_specific> | <tcp_specific> | <udp_specific>
<icmp_specific> ::=
{'src-type-code:' <icmp_type_code> <endl>}?
{'dst-type-code:' <icmp_type_code> <endl>}?
<icmp_type_code> ::= 'any' | <icmp_type>'/'<icmp_code>
<icmp_type> ::= Integer (* 1..5 : refers to RFC 792 *)
<icmp_code> ::= Integer (* 1..20 : refers to RFC 792 *)
<tcp_specific> ::=
{<udp_specific> <endl>}? {'ttl:' <uint8> <endl>}? {'tos:' <uint8> <endl>}?
{'flags:' <tcp_flags> <endl>}?
<tcp_flags> ::= {'U'|'A'|'P'|'R'|'S'|'F'}+
<udp_specific> ::=
{('src-port:' | 'dst-port:') <port_enum_set> <endl> }+
(src-port-seq)? (dst-port-seq)? (* definitions for port sequence/scanning *)
<port_enum_set> ::= 'any' | {(<uint16> | <uint16>'-'<uint16>) {','<port_enum_set>}*

(* in the expression <uint16>'-'<uint16>
* the first number MUST be lower than the second *)
<ip_address_set> ::= ('any' | ('intranet' | 'internet' | <ip_net>) {','<ip_address_set>}+
<ip_net> ::= <uint8>'.'<uint8>'.'<uint8>'.'<uint8>'/'<cidr>
<cidr> ::= Integer (* 1..32 *)
<uint16> ::= Integer (* 0..65535 *)
<uint8> ::= Integer (* 0..255 *)
<endl> ::= ';' | <EOL>

As the match-all rule states, it does not matter if the pattern statement is defined with "any" value or it is missing at all. For instance, the following pattern descriptions are equivalent:

Pattern 121 (proto:icmp; src-addr:internet; dst-addr:any;)
Pattern 122 (proto:icmp; src-addr:internet;)

The pattern describes traffic characteristics of OSI Layer 3 and 4 attributes. Every pattern defines a specific protocol (ICMP, TCP, and UDP are implemented only) and it is expected that the protocol-specific data should be defined as well. Those data must match the protocol identifier or otherwise will be ignored. The current traffic flow is checked against the pattern(s) in the pattern configuration file. If the recognition matches, the process acts as follows:

- It increases the match counter and checks it against the occurrence frequency. If the latter is bigger than a specific threshold, the process raises an assigned action.
- The match process raises an assigned action immediately when no frequency is defined or the pattern is checking against a port or address sequence.

The main phases of the pattern match process are: *Read, validate, and activate patterns → Get traffic → Map against the pattern(s) → Reaction: log, iterate, reconfigure, etc.* The program logic is implemented in the third phase. The pattern definition is stored in a data structure. The pattern API provides functions for parsing and filling up data structures and outputting them in a byte stream. On the other hand, the NetFlow API implements traffic collection and store. Both APIs are compliant to a "check algorithm" against the pattern semantics. During the check, several additional handler operations are performed. They are used for process optimization as well as handling complex tasks, e.g., checking against address and port sequences. The following operations are done during pattern-to-traffic validation:

- If the Layer 4 protocol (ICMP, TCP, and UDP) does not match, the check returns false.
- If the sequences (address and/or port) are defined, the process behavior depends on:
  - Checking the IP address (source or destination, depending on the source or destination sequence definitions) against the address set definitions (network segments, Intranet, Internet). If the address matches at least one of the sets, the traffic entry is inserted in the Sequence Handler Module (SqHM). Otherwise it is not.
  - If no corresponding (source or destination) IP address set is defined, the traffic entry is inserted in SqHM (matches all).
  - SqHM is queried for address and/or port sequences described in the pattern. If a sequence was found, SqHM returns the address for it. The IP address is the last from the address sequence or the unique

address for a port sequence.

  - The pattern matching process compares the address returned by SqHM against the corresponding pattern address set definitions. If there is no address set definition (match-all rule) the process returns true. If the address returned by SqHM matches the address set definition the process returns true as well.
  - If there are no protocol-specific statements defined, the process returns the result from the invocation of the Frequency Handler Module (FqHM).

- When a port statement is defined, the process checks against the port set. If the corresponding port matches the set, the process returns the result from FqHM.
- When TCP protocol-specific characteristics are defined in pattern statement(s), the process check against them (TTL, ToS, flags) and returns the result from FqHM, if the check operation matched.
- FqHM invocation iterates the matches and calculates the matches per second. The latter is checked against the frequency defined in the corresponding pattern. If no frequency value is defined or the frequency statement has a value of "any", FqHM returns true. Otherwise returns false.
- The checking function follows the signature (pseudo C code): `BOOL check (pattern*, ip_traffic*);`. The recommendation is to implement a function as universal as possible in order to manipulate different network traffic output interfaces and APIs.

The last step of the process is the reaction to the event. While the IP Flow Patterns core algorithm is dedicated to the anomaly and DoS detection, the reaction step is executed externally. The IPFP configuration provides an ability to declare a path to an external executable that is responsible to the event reaction. The executable receives the following information as command line parameters: the victim IP address, the peer IP address (attacker), the source port, the destination port, the last NetFlow entry to trigger the event. For example, should the following string be configured for external execution '/root/tools/suspicious.sh -x', the execution line would look like this:

```
/root/tools/suspicious.sh -x 10.10.10.123 172.17.17.17 80 12341 \
tcp/10.10.10.123:80:172.17.17.17:12399
```

2.5.2. Handler Modules. The IPFP implementation uses two handler modules: Sequence Handler Module (SqHM) and Frequency Handler Module (FqHM). These are additional modules in the check mechanism for providing a pluggable infrastructure. The modules are tightly connected to the pattern

semantics via the statements src-addr-seq, dst-addr-seq, src-port-seq, and dst-port-seq for the SqHM and frequency for the FqHM.

2.5.2.1. Sequence Handler Module. This module is designed for caching network data. It is size-restricted and the cache expiration period is practically throughput dependent. SqHM is a table with strictly defined length. It stores minimal network traffic entries. The entries themselves contain four additional columns for storing entry pointers in order to implement two bidirectional link lists. The data remain in a process image allocated in the memory. Depending on the traffic rates this cache table must be configured with a proper size. For low-rate IP traffic, e.g., less than 10 Mbps, a size of 65536 is good enough. If the patterns using the SqHM are defined with both source and destination IP address sets, SqHM will be able to store network traffic entries for larger periods of time.

Here is a sample output snippet from the cache table (< 2 MB):

idx	spr	snx	dpr	dnx	pt	src_ip	sprt	dst_ip	dprt	sec	nanosec
0000	001f	0017	0011	000a	06	010b0ad9	0002	010be605	0002	1b1b2299	1bae83a2
0001	0006	0003	0013	0004	06	01000006	0001	01040c0d	0001	09ca4fe9	69868278
0002	0018	000c	000d	0013	06	4e2d0201	0002	d30635db	0002	03f70e08	3a2389bf
0003	0001	0005	0019	000c	06	01000007	0003	01130002	0002	451d1318	46fce92e
0004	0007	0006	0001	0007	06	01000004	0004	01040c0d	0004	6e124ee7	0c9d1752
0005	0003	0008	0012	0010	06	01070013	0005	015801cd	0005	461c25a3	4f30854d
....	....	....	....	....	..	....	....	....	....	....	....

The cache table is made of homogeneous data structures. Every entry is 29 bytes long. If we allocate 2^{16} rows, then the whole allocated memory will be 29*65536 = 1,9 MB. The structure we are using can represent up to 65536 sorted entries. If there is a requirement for a bigger cache, then the four indexes must be changed to 4-byte values. In that case we are able to address up to 2^{32} rows and the total size of the cache is up to 4294967296*37 = 158,9 GB. Using a big table is not effective because this kind of cache uses a linear search.

An SqHM must be instantiated per pattern. Every pattern that contains a sequence statement must use a different SqHM cache. Otherwise the match can fail or produce unexpected result. The SqHM is a sorted bidirectional link lists table and following the links the mechanism can detect address and port sequences. When an insert operation is performed, the entry must find its place in order to keep the sorted cycle list consistent. The algorithm itself must provide robust consistency for both source and destination values. When a new value is added, the IP Flow Pattern detect process performs a scan for source and destination socket sequences. The complexity of the algorithm can be calculated in two steps. First, insert action is a linear search function which performs a maximum of N iterations until finding the place for the new entry. After that the sequence detection function again performs a linear search in exactly N steps.

Finally, insert and sequence search ability costs $O(2n)$ for each direction (source and destination). Since the table has fixed length (e.g., 65356), we can expected a constant cost behavior of the cache table utility.

SqHM is able to detect address and/or port based sequences since the cache flow table (the sequence cache) is not dated. When high traffic occurs, SqHM will be able to find a sequence flowed within shorter time, while in case of low congestion SqHM could detect such a DoS address/port scanning for a longer periods. When SqHM is active (the pattern is defined for sequence detection) we have implemented additional programmed logic for checking the peer socket. In order to elaborate further, let us assume that we have activated the following pattern:

```
pattern 6 (
descr:TCP sequence detection
proto:TCP
dst-addr:10.32.0.0/16
dst-port-seq
seqlen:30 )
```

The pattern will be able to discover port scanning of the host from a wide intranet (10.32.0.0/16). Most probably we will be able to find a port sequence for some host or hosts within very short time:

```
tcp 192.168.10.11:80 -> 10.32.24.10:1035
tcp 192.168.10.18:80 -> 10.32.24.10:1033
tcp 192.168.10.14:81 -> 10.32.24.10:1037
tcp 192.168.10.10:80 -> 10.32.24.10:1034
tcp 192.168.10.21:81 -> 10.32.24.10:1036
tcp 192.168.10.12:80 -> 10.32.24.10:1038
.....
```

The destination socket in fact changes sequentially but when looking at the peer socket (the source) we can easily find that actually the source host was the target of malicious service scanning; in this example it was over well-known HTTP ports. This is why, when checking against port permutation scanning, we implicitly make additional checks against service(s) scanning of the peer. Here is some typical output of SqHM when a service discovery attempt is detected:

```
2006-03-09.19-41-44| WARN| 31775|212|
max is c0a80a00->30 * 100.00% from 30 top * 8 ports are 80 81 0 0 0 0 0 0
2006-03-09.19-41-44| INFO| 31775|212|
Peer host(s) are the real victim [net:192.168.10.0, percent:100.00].
10.32.24.10 runs port scanner.
```

2.5.2.2. Frequency Handler Module. Frequency Handle Module (FqHM)

implements the steps for iteration and check when the pattern frequency statement is defined. When the check function detects traffic-to-pattern match, it calls the FqHM in order to iterate the matches. FqHM calculates the matches per second (mps) and returns true if the mps value is larger than the pattern frequency statement value and if there is no pattern frequency statement. Otherwise it returns true.

FqHM stores a pointer to the pattern definition, startup time values, and matches. FqHM is used on per-pattern basis. Every pattern has its own FqHM structure. FqHM could be enhanced in order to store the last N matches for further analysis instead of issuing directly an entry match. During the experiments we observed that the high traffic with specific characteristics as defined by the DoS pattern could be a normal or abnormal flow depending on the network services installed. It depends on the administrator's knowledge of the characteristics of the specific network services as to how to define the DoS patterns in order to exclude that kind of flow.

2.6. A Prototype System Architecture. A prototype was developed and deployed in the university campus network of the Faculty of Mathematics and Computer Science in Sofia. The system, running on a Linux platform, received the NetFlow data from the backbone campus router, so the characteristics of the entire campus traffic were analyzed by the system.

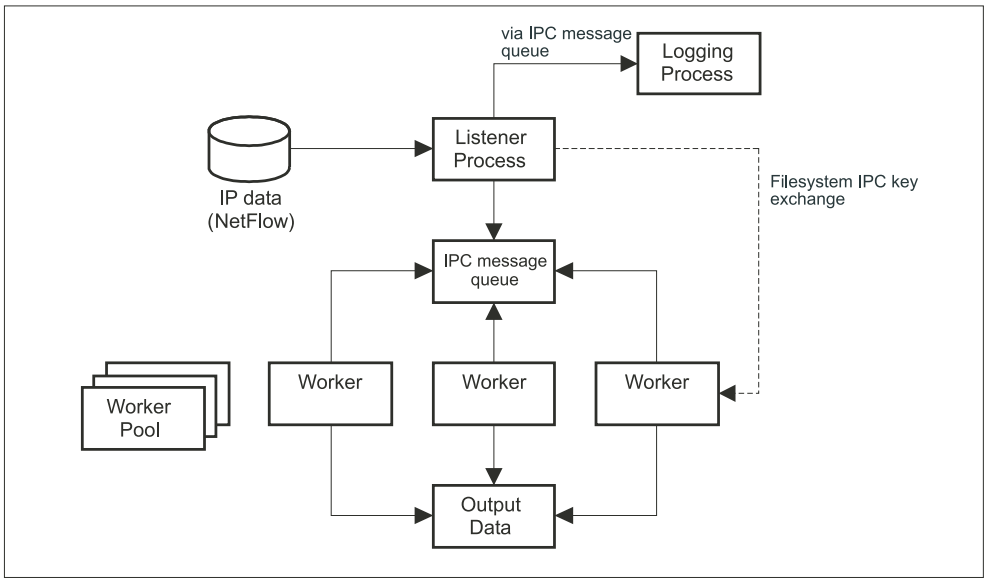


Fig. 6. IPFP Collector

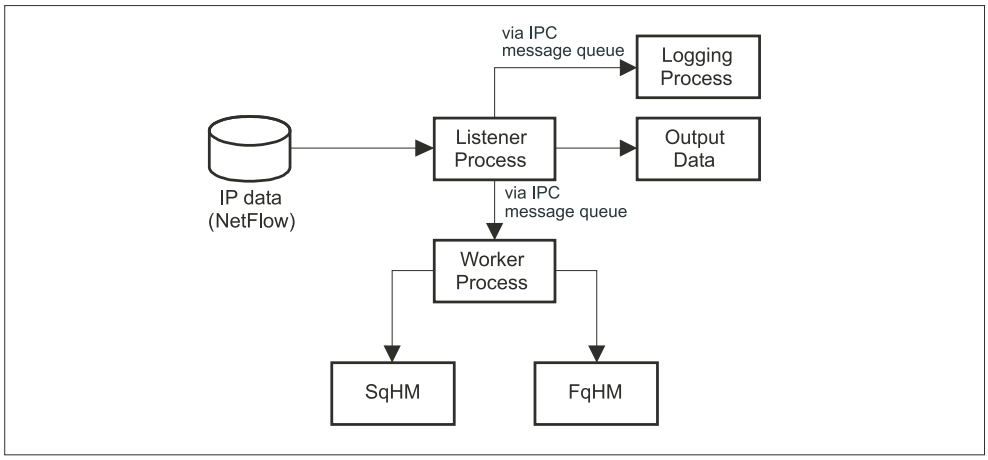


Fig. 7. IPFP Inspector

The system comprises two major components: *IP Flow Collector* and *IP Flow Inspector*:

- The IPFP Collector (Fig. 6) collects the traffic data from the routers via NetFlow, generates human-readable interpretation of the data, and also does data aggregation for the flow statistics. The output format is controllable and the administrator could expose it via an HTTP server.
- The IPFP Inspector (Fig. 7) is the actual core module of the IPFP prototype system. It is responsible for matching the IPF patterns against the traffic flow in order to discover the DoS threats. The Inspector basically matches the NetFlow data against the predefined patterns. The match results are printed out in log files; however, the administrator is also able to run an external process (program) that receives a short info about the suspicious traffic via input parameters. That external program would be able to take further actions for the attack prevention, for example, sending system notifications and early watch alerts, reconfiguring automatically the network devices, and other related activities for DoS prevention that are specifically needed for the network infrastructure.

The IPFP basically addresses the network anomalies based on the IP header data. IPFP does not take into account the application level data as was decided in the design phase (Section 2.5). The solutions we described in Section 1.3 as well as the modern IDS/IPS components do not process deeply the IP level peer communication; they rather utilize the traffic inspection through the whole OSI stack emphasizing at the application protocol level as it might

exploit specific, usually well-known, vulnerabilities. In order to compare the IPFP functionality and performance towards another system or software, we had to find one that is similar to IPFP or at least one that claims to be able to provide permutation scanning on the addresses and ports together with port scanning (sweeping) detection. This was the reason why we chose SNORT for the comparison.

2.7. Comparing IPFP with SNORT. In the last chapter of the thesis we defined a methodology for comparing IPFP and SNORT [22] and then applied that approach to these systems. The methodology is based on two aspects: (1) functional tests and (2) assessment indexes. The functional tests were provided in terms of simulation programs that can replicate the following network flow anomalies:

- UDP port sweeping—sending a UDP datagram towards certain port range(s).
- UDP datagram flooding—sending UDP messages to a specific port.

2.7.1. Additional comparison indexes. Beside the functional testing we defined a series of additional boolean indexes for comparing the systems IPFP and SNORT. The indexes were measured on the basis of on the functional capabilities of the systems (Protection indexes group), the system design features (System indexes group), as well as the resource consumption (Runtime group). Table 5 summarizes the indexes and whether the systems match them.

2.7.2. SNORT vs. IPFP—summary. The functional tests and the comparison indexes showed the level of maturity of the IPFP. The fact that SNORT is commercial IPS software with a long background was also taken in account. Keeping in mind that IPFP focuses on the IP flow level rather than the whole OSI stack, we still proved that the IPFP concept is as powerful as the full-blown IPS concept. The specific focus on the transport level definitely resulted in a higher level of attack detection as we saw with the functional tests. The features of SNORT and IPFP are summarized in Table 6.

3. Conclusion. At the very beginning we outlined the ultimate goal—to find an effective mitigation method for the major severe DoS issues. In order to address this goal we defined three problems to solve in our paper:

- Research the current state-of-the-art approaches and methodologies for analysis of network flow behavior; research and analyze the DoS attacks, their characteristics and taxonomies based on common features.

[1]SANS Internet Storm Center—Cooperative Network Security Community—Query for a threat level report on the TCP port number 2967
`http://isc.sans.org/port.html?start_month=12&start_day=21&start_year=2006&end_month=12&end_day=31&end_year=2006&port=2967`

Table 3. Discovering UDP port sweeping: horizontal: 1, 2, 5—number of the repeats of the scanning; S, R—sequential scanning and random scanning, respectively vertical: 1200, 4000, ..., 62000—starting port number for the corresponding scanning region; 50, 200, ..., 2500—port region length

UDP port	scanning length	SNORT (7 matches)						IPFP (222 matches)					
		1		2		5		1		2		5	
		S	R	S	R	S	R	S	R	S	R	S	R
40000	50	X										X	X
	200							X	X				
	500				X			X	X	X	X		
	1000										X	X	
	2500			X		X			X	X	X	X	
35000	50											X	X
	200					X		X	X				
	500						X	X	X	X	X		
	1000										X	X	
	2500								X	X	X	X	
25000	50											X	X
	200							X	X				
	500							X	X	X	X		
	1000										X	X	X
	2500								X				
20000	50											X	X
	200							X	X				
	500							X	X	X	X		
	1000									X	X		
	2500								X	X	X	X	
18000	50											X	X
	200							X	X			X	X
	500							X	X	X	X		
	1000			X							X	X	
	2500				X				X	X	X		
10000	50											X	X
	200							X	X				
	500							X	X	X	X		
	1000									X	X		
	2500								X	X	X	X	
4000	50											X	X
	200							X	X				
	500							X	X	X	X		
	1000										X	X	X
	2500								X	X	X	X	
1200	50											X	X
	200							X	X				
	500							X	X	X	X		
	1000									X	X		X
	2500								X	X	X	X	

– Select an appropriate methodology for DoS attacks detection and develop a system model for applying that methodology.

Table 4. UDP flooding detection: Horizontal: 50, 100, 200, and 500—number of the datagrams in the traffic flooding; Vertical: 1000, 3000, . . . , 62000—the victim's UDP port numbers

UDP port / # of datagrams	SNORT (1 match)				IPFP (27 matches)			
	50	100	200	500	50	100	200	500
1000				X	X	X	X	XX
3000				X	X		X	X
5000				X	X		X	X
20000				X	X	X	X	X
35000				X	X	X	X	X
50000				X	X	X	X	X
62000				X	X	X	X	X

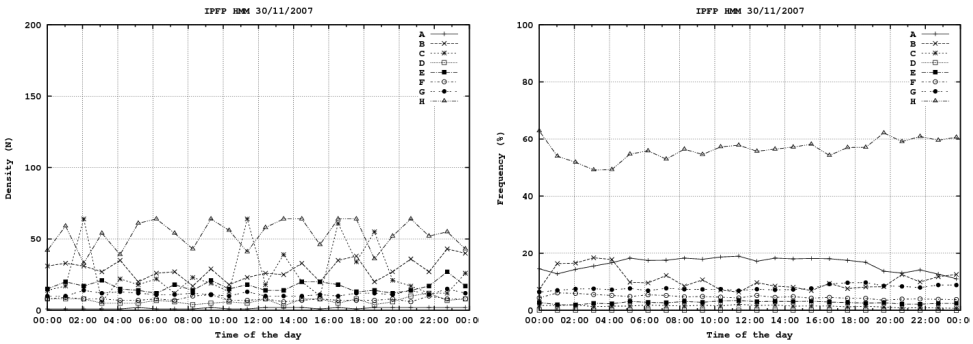


Fig. 8. Densities and frequencies of the network symbols during the normal network behavior

– Implement the model with software technology, test the product in the network environment, analyze the output results, and compare the effectiveness with existing systems, products, and approaches.

These problems were successfully addressed in the thesis and as a result we ended up with a prototype system called IP Flow Patterns (IPFP). The IPFP was a clear proof of the concept defended in the thesis. This is why we may outline the following major contributions of our solution:

– Network security issues were investigated, emphasizing on the DoS attacks issues. Research of the network flow analysis methodology and the approaches for estimate and analysis of the normal network behavior is followed. The DoS attacks taxonomies and the defense mechanisms were made in dependence of different grouping standards.
– Research of the network flow probing methods was pursued and a model

Table 5. IPFP vs SNORT—a summary of the additional comparison indexes

		IPFP	SNORT
Protection indexes	Passive protection	X	X
	Active protection	X	X
	Permutation scanning detection	X	X
	High-frequency traffic detection ("port or address sweeping")	X	–
	Attacks on the application protocol	–	X
	Infrastructure level protection	X	– *
	Local host-based protection	– *	X
System indexes	Support of a standard input data and interfaces	X	X
	Flexibility	X	X
	Scalability	–	X
Runtime indexes	CPU consumption	–	–
	Memory consumption	X	X
	Usability and maintenance	X	–
* some exceptions apply depending on the infrastructure			

Table 6. SNORT vs. IPFP—features summary

SNORT	IPFP
Complex fully functional IDS.	Simplified model for network defense.
A rule-based detection engine checks the traffic against traffic characteristics.	Uses text definitions to check the network traffic against specific patterns.
Takes a predefined operation (LOG, ACTION) when at least one packet matches with the rule.	Takes an action (a command string) when a set of packets matched the IP Flow pattern.
The SNORT rules analyses the traffic through the whole OSI stack—from data-link to application layer.	The IPFP works on the network and transport layer; the packets payload is not inspected.
Using SNORT rules the network administrator can detect attack attempts to some specific application protocol's vulnerability. The network and transport layer-related threats are handled by a dedicated plug-in module.	Using IPFP the network administrator can detect attack attempts for permutation scanning of network segments, hosts, and services. The IP Flow patterns discover high-frequent requests to sets of segments, hosts, and services.

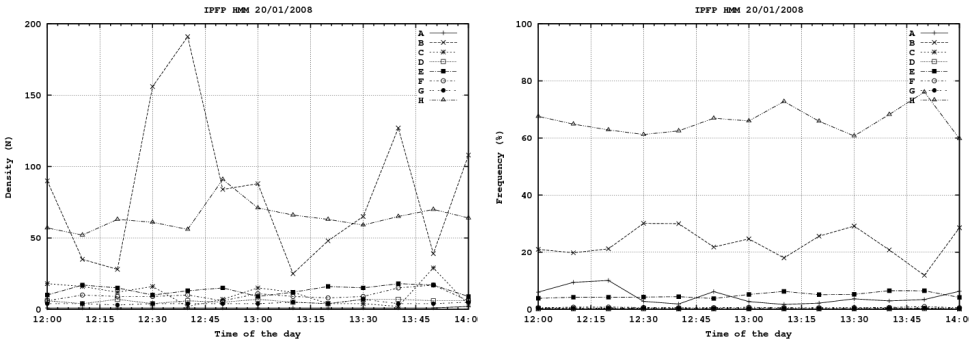


Fig. 9. Densities and frequencies of the network symbols during suspicious traffic flows

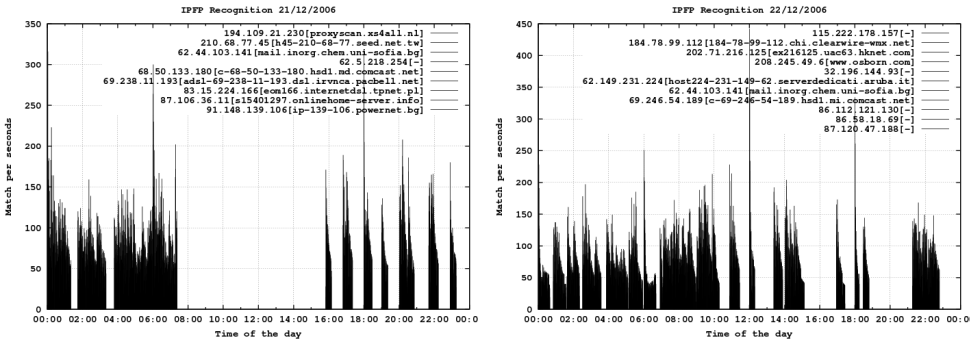


Fig. 10. Occurrences of a suspect network flow. The graphic on the right presents flooding over application protocol ssc-agent (tcp/2967) that was apparently confirmed to be a severe threat in Dec 2006 by SANS Internet Storm Center[1]

for drilling and analysis was chosen.

- There was invented a new approach for DoS mitigation, based on the analyzed DoS defense mechanisms. Using a mathematical model we discovered the essential network flow parameters. These parameters were built into the new defense model.
- There was developed a methodology for functional quality assessment of the DoS defense mechanisms.

In addition to the research we also developed a prototype for proving the new concept for the DoS defense. The IPFP contributed basically to the applied aspect of the thesis. The following is the summary of the achievements of the application level:

- Software tools for network traffic data were developed.

– Based on the new model for DoS protection the IPFP was implemented. The IPFP was tested in real life scenarios and the output was analyzed. The analysis results were compared with the corresponding online data centers for transport level vulnerabilities (Fig. 10).
– IPFP was compared with SNORT based on the methodology for assessment of the effectiveness of the DoS defense system.

The IP Flow Patterns model was also submitted for a patent at Sunnyvale, California, U.S.A. [23]. The innovations that were nominated for patenting were the pattern defense model and the pattern syntax, the methodology for the network flow data processing, the methodology and the concept for permutation network scanning detection, the frequency calculation of the traffic flowing rates and others. Some parts of the thesis were probed and presented at conferences, conventions, and a scientific journal ([24], [25], [26], [27], [28]).

REFERENCES

[1] Mirkovic J., J. Martin, P. Reiher. A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms. Computer Science Department, University of California, Los Angeles, Technical report #020018.

[2] Dittrich D. Books on DDoS. http://staff.washington.edu/dittrich/misc/ddos/

[3] Shimeall T., C. Dunlevy, L. Pesante. Challenges of Predictive Analysis for Networks, CERT®Analysis Center.

[4] US-CERT, Understanding Denial-of-Service Attacks, National Cyber Alert System. http://www.us-cert.gov/cas/tips/ST04-015.html

[5] Floyd S., S. Bellovin, J. Ioannidis, K. Kompella, R. Mahajan, V. Paxson. Pushback Messages for Controlling aggregates in the Network. Internet draft, Work in progress. http://search.ietf.org/internet-drafts/draft-floyd-pushbackmessages-00.txt, July 2001.

[6] Bellovin S. ICMP traceback messages. Internet draft. http://lasr.cs.ucla.edu/save/rfc/draft-bellovin-itrace-00.txt, October, 2001.

[7] Bellovin S., M. Leech, T. Taylor. ICMP Traceback Messages. http://www.ietf.org/proceedings/02mar/I-D/draft-ietf-itrace-01.txt

[8] Mirkovic J. D-WARD: Source-End Defense Against Distributed Denial-of-Service Attacks. http://lasr.cs.ucla.edu/ddos/dward-thesis.pdf

[9] O'Brien E. NetBouncer: A practical client-legitimacy-based DDoS defense via ingress filtering,
http://www.isso.sparta.com/documents/netbouncer.pdf

[10] Keromytis A., V. Misra, D. Rubenstein. SOS: Secure Overlay Services. http://www.sigcomm.org/sigcomm2002/papers/sos.pdf

[11] Proof-of-Work system. (definition from Wikipedia).
http://en.wikipedia.org/wiki/Proof-of-work_system

[12] Mirkovic J., M. Robinson, P. Reiher, G. Kuenning. Forming Alliance for DDoS Defenses. In: Proceedings of the New Security Paradigms Workshop (NSPW 2003), ACM Press, 11–18.

[13] Papdopoulos C., R. Lindell, J. Mehringer. A. Hussain, R. Govindan. Cossack: Coordinated Suppression of Simultaneous Attacks. In: Proceedings of 3-rd DARPA Information Survivability Conference and Exposition (DISCEX 2003), Vol. **2** (April, 2003), 94–96.

[14] Yaar A., A. Perrig, D. Song. Pi: A Path Identification Mechanism to Defend against DDoS Attacks. In: Proceedings of the IEEE Symposium on Security and Privacy, May, 2003, 93–107.

[15] Yaar A., A. Perrig, D. Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In: Proceedings of the IEEE Symposium on Security and Privacy, May, 2004, 130–143.

[16] Jin C., H. Wang, K. G. Shin. Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic. In: Proceedings of the 10-th ACM Conference on Computer and Communication Security, ACM Press, October, 2003, 30–41.

[17] Cisco IP Source Tracker.
http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/ipst.html

[18] Kulkarni A., S. Bush, S. Evans. Detecting Distributed Denial-of-Service Attacks Using Kolmogorov Complexity Metrics. Technical Report 2001, CRD176, General Electric Research and Development Center, December, 2001.

[19] NetFlow standard. RFC 3954, October, 2004.
http://www.ietf.org/rfc/rfc3954.txt

[20] Daigle J. Queueing Theory with Applications to Packet Telecommunication. Springer, 2005.

[21] Rabiner L. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: Proceedings of the IEEE, **77** (1989), No 2, 257–286.

[22] SNORT. The free Intrusion Detection System. www.snort.org

[23] Tsvetanov T. Pattern-based Network Defense Mechanism. Patent application, US ID 20080295173, EFS ID 2083257, August, 2007.

[24] Tsvetanov T., S. Simeonov. A Study on Network Flow Security. *Cybernetics and Information Technologies*, **8** (2008), No 3, IIT—BAS, Bulgaria, 32–47.

[25] Tsvetanov T., S. Simeonov. Pattern-Based Security Algorithm for DoS Detection. In: Proceedings of the 20-th International Conference on Systems for Automation of Engineering and Research SAER 2006, St. Konstantin, Varna, Bulgaria, September, 2006, ISBN-10: 954-438-575-4, 80–88.

[26] Tsvetanov T., S. Simeonov. Applying Pattern Detection Network Security against Denial-Of-Service Attacks. In: Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics WMSCI 2006, Orlando, Florida, USA, July, 2006, ISBN 980-6560-65-8, 314–319.

[27] Tsvetanov T., S. Simeonov. Securing a Campus Network. In: Proceedings of the 29th International Convention MiPro 2006, Opatija, Croatia, May, 2006, ISBN 953-233-022-4, 141–146.

[28] Tsvetanov T., N. Manolov. Cisco IOS Features used for QoS Analysis and Implementation. In: Proceedings of the XXXIX International Scientific Conference on Information, Communication and Energy Systems and Technologies ICEST 2004, Bitola, Macedonia, June, 2004, ISBN 9989-786-38-0, 99–102.

Tsvetomir Tsvetanov
St. Kl. Ohridski University of Sofia
e-mail: ttsvetanov@gmail.com

Received January 17, 2010
Final Accepted July 22, 2010