Securing DBMS: Characterizing and Detecting Query Floods.

Elisa Bertino[1], Theodore Leggieri[2], and Evimaria Terzi[3]

[1] CERIAS and Department of Computer Sciences,
Purdue University, West Lafayette, IN, USA,
bertino@cerias.purdue.edu
[2] Department of Computer Sciences
University of Milano, Italy
[3] HIIT and Department of Computer Science
University of Helsinki
Helsinki, Finland
terzi@cs.helsinki.fi

Abstract. Current multi-tiered Web-based applications are very often characterized by the use of a database system. Database systems are thus not any longer confined to well protected environments. In this paper, we focus on a specific type of attacks, known as query flood. Under such an attack, a subject, or a colluding set of subjects, floods the database with a very large number of requests thus making the database unable to serve, with adequate response time, requests from honest subjects. The approach we propose in based on modeling access profiles to the database and using these profiles to detect unusual behaviours in the subjects accessing the database. Our approach supports varying granularities in that one can build a single profile for the entire database or build specialized profiles for each table in the database. We employ our techniques both in misuse and anomaly detection settings. An evaluation of the proposed approach has been carried out and results of the evaluation are reported in the paper.

1 Introduction

A database system is usually a core-component of modern web-based application architectures since it is in charge of storing the application-relevant data that need to be made available, acquired and managed through the Web. Apparenlty, the exposure of databases on the web increases the security risks to which data are exposed, which explains the large research effort towards developing sophisticated data protection techniques [1]. Whereas there is a variety of techniques addressing data confidentiality and integrity, the problem of database protection against denial-of-service (DoS) attacks has not been addressed. A typical DoS attack against a DBMS may occur through a *query flood*, that is, when a subject or a set of colluding users imposes a large number of queries and/or updates against the database with the intention to hinder other people from being serviced.

The goal of the work reported in this paper is to develop techniques and methodologies able to address query flood attacks against relational database systems. Our approach is characterized by three key aspects: (a) We address the problem of DoS attacks both from the point of view of misuse and anomaly detection. (b) We provide a characterization which is time-dependent and parameter-free. Some of the techniques we use are borrowed from standard time series and sequence analysis (see for example [2–5]). However we enhance them with features adequate to serve our purposes. (c) We provide different protection granularity so that the database administrator may decide to protect the entire database and using just a global access profile for the entire database or protect single relations by building a specific profile for each table.

Related work: Intrusion Detection (ID) has been a very active area of research. There are mainly two techniques towards ID, namely *anomaly detection* and *misuse detection*. A taxonomy and an overview of the existing ID Systems (IDS) is presented in [6]. In general, anomaly detection identifies activities that vary from established patterns for users or group of users. Anomaly detection typically involves the creation of knowledge bases (containing the profiles of the monitored activities), whereas misuse detection involves the comparison of the users activities with the (known) behaviors of attackers (attempting to penetrate a system). DoS attacks have been considered in many previous research efforts as for example in the work of [7–9]. Classes of models and tools that can become core components in intrusion detection systems have been extensively discussed in [10, 11]. However, those approaches mainly focus on network and operating system layer attacks. Additionally, the data mining and machine learning tools they use for constructing the models are completely disjoint with those proposed here. The only work that deals with DoS attacks at application layer is the work presented in [12]. In this work flooding-based denial-of-service attacks in Peer-to-Peer (P2P) systems have been investigated. The authors propose simple techniques to deal with the problem depending on the topology of the network where the distributed nodes that handle the arriving requests reside. However, such technique is very coarse and is not able to model access patterns also according to time.

Contribution: In this paper we focus on the detection of query floods within the database system at the application level. To the best of our knowledge this is the first comprehensive and articulated approach dealing with query flood attacks considering application layer log information. There are two main reasons why application layer attacks should be dealt with separately. First of all, actions malicious for an application are not necessarily malicious for the network as well. Additionally, network and application layer intrusion detection parts may cooperate with each other. Therefore, intrusion detection mechanisms that may be too time consuming and computationally expensive for network routers, for example, can very well be handled by an application layer mechanism. Database tables play a key role in our approach as they are considered as central entities around which the security model is built. The main tools used for handling the database log files are borrowed from information theory.

2 Preliminaries

2.1 Minimum Description Length (MDL) Principle

MDL principle is a major tool that has been used to scale down the ideas behind the Kolmogorov complexity in order to make them applicable into practical settings. In brief MDL states the following: Assume two parties A and B, that want to communicate with each other. More specifically assume that A wants to send data D to B using as less bits as possible. In order for A to achieve this minimization of communication cost he/she has to select a model M, from a class of models $\mathcal{M}$, such that M best describes his/her data and then try to describe his/her data based on the chosen model M. The model M is selected from the class of models $\mathcal{M}$ with probability $P(M)$ while the data D (observations) have been generated by M with probability $P(D \mid M)$. Based on the Shannon's Information Theorem the number of bits needed to encode the data D is $L(D) = -log(P(D))$, where $P(D)$ is the probability of the data. Given that we have selected model M for our data then the encoding length in bits will be $L(D \mid M) = -log(P(D \mid M)P(M)) = -log(P(D \mid M)) - log(P(M))$. So the goal of the MDL principle is to find the model (hypothesis)$M \in \mathcal{M}$ such that the number of bits needed to encode the data (namely $L(D \mid M)$) is minimized. The set of models considered for modeling the data are the segmentation models, where a segmentation of a sequence of events is a set of successive non-overlapping intervals that cover the whole sequence. Each interval is characterized by its starting and ending points and its level that corresponds to the average intensity of the sequence in the segment.

2.2 Notation

For the rest of the paper we consider a database $U = R_1, R_2, .., R_n$ consisting of n relations. Each relation R_i consists of set of attributes A_i with $A(i) = \{a_1^i, a_2^i, ..., a_{n_i}^i\}$. Also consider a sequence of log files LF_τ ordered with respect to time τ and denote with LF_τ^r the series of log files that contain SQL commands that have included relation R_r. These log files have been separated from the global LF_τ after a preprocessing step. The set of usable SQL commands is denoted by Cmd, with $cmd_i \in Cmd$ to correspond to a single such command.

3 Misuse Detection of DoS attacks in DBMS

Assume a sequence of τ log files LF_τ consisting of commands of different types. Each such log file corresponds to logs over the same time-line T (say daily logs) and consists of a set of transaction log records. Each record contains the command used, the time this command was issued as well as other information.

Mining each such log file is a relatively easy task and can be done by defining sliding windows of length w along time-line T, and for each such sliding window counting the frequency of commands or combinations of commands of specific type. The frequency counting of commands and sets of commands is similar to the frequent episode mining problem. The mining algorithm we use is based on the frequent set mining algorithm (Apriori) [13]. For the sake of completeness we give the definitions of *k-command set(i)* and *frequent k-command set(i)* that are in complete agreement with the definitions of (frequent) itemesets and (frequent) episodes.

Definition 1. *A* **k-command_set** *is a set of k distinct commands that appear together in at least one sliding window in LF_τ.*

Definition 2. *A* **frequent k-command_set** *is a set of k distinct commands that appear together in at least* **freq** *sliding windows in LF_τ, where* **freq** *is the frequency threshold used in the mining process.*

In the context of misuse detection the goal is the following: Given the sequence of log files FL_τ we need to identify a segmentation S over τ with respect to the probability (or pattern) of appearence of a specific *command_set* $Cmd_i \subseteq Cmd$ throughout LF_τ.

For simplicity consider first the 1-command_set example. In this case the only parameter of the model is the probability of appearance (or average frequency) of each command within segment s, denoted by p_s. The goal is to find a segmentation S of the τ logs with respect to $p_s \forall s \in S$. More specifically, from the class of all possible segmentation models $\mathcal{M}$ over τ we need to identify a model $M \in \mathcal{M}$ such that $L(M) + L(D|M)$ (i.e. the length in bits of encoding the model and the data given the model) is minimized. The model itself is the value of p_s over the different segments s. If we assume that we have l segments for calculating the cost of the model we need to add up (a) the cost of the encoding the parameter p_s for each segment and (b) the cost of encoding the l block boundaries. Since p_s is a real number the cost for encoding it will be $\rho = \frac{1}{2}\log(|s_k|)$, with $|s_k|$ being the length of segment s_k. Calculating the cost of (b) is easy because each block boundary is a natural number between 1 and τ and therefore for each one of them we need $\log \tau$ bits. The total cost of the model in the case of 1-command_sets is:

$$L(M) = \sum_{k=1}^{l} \frac{1}{2} \log(|s_k|) + l \log \tau \tag{1}$$

The cost of encoding the data given the model $L(D|M)$ is given by:

$$L(D|M) = \sum_{s=1}^{l} -\log p_s \cdot P_{\mathcal{D}}(D(s)) \tag{2}$$

where $P_{\mathcal{D}}$ corresponds to the distribution that the patterns (here denoted by D) follow and $D(s)$ to the data in segment s.

Multiple Command Types: In the multiple command case we need to determine segmentation points over τ with respect to the probability or pattern of appearance of a *k-command_set*. For the description of each of the *k-command_sets* we need to define the probability of appearance of each one of the k commands in the set and therefore our model consists of these k probabilities. These commands may not neccesarily be dependent, and therefore the space of models that cover all possible dependencies grows exponentially. For example in the 3-command case there are $7 = \sum_{i=1}^{3} \binom{3}{i}$ possibilities of combinations of dependencies between the corresponding 1-command_sets. However notice that once one has found the best model for the case of 2-command_sets then one does not have to search the space of all the possible models for the 3-command_set case but it is enough to consider only the subset of models that are compatible with the one selected for the 2-command_set case. This observation proves useful for prunning the space of models.

Algorithm for finding the best segmentation The best segmentation can be found using dynamic programming with the following set of equations: Let $s = [a, b]$ be a temporal interval. The cost associated with interval s is:

$$f(a, b) = -\log P_{\mathcal{D}}(D(s) + \frac{1}{2}\log(|s|) + \log \tau$$

Therefore, the optimal segmentation S_{opt} is the one that minimizes such cost, that is, we require that:

$$L(S_{opt}) = min_S \sum_{[a,b]\in S} f(a, b)$$

where S includes all possible segmentations in $[1, \tau]$. If additionally we denote with $F(k)$ the cost of the optimal segmentation from 1 up to k, then the total dynamic programming equation is as follows.

$$F(k) = min_{1\leq i\leq k}(F(i-1) + f(i, k))$$

where $F(0) = 0$.

Reporting DoS attacks: For separating the surprising patterns that may be indicators of attacks we need to define measures for quantifying "suspiciousness" of a pattern. Some natural definitions of the suspicion **suspicion factor** of a pattern are the following:

Definition 3. *The* **"diff" suspicion factor** *gives an estimate of how suspicious the fluctuation of a pattern, or in our case a command_set, based on the L_1 metric and thus is defined as:*

$$SF_{diff}(cmd\_set) = |p(cmd\_set)^{obs} - p(cmd\_set)^{est}|$$

Alternatively, one can consider the ratio of estimated probabilities. This ends up in the "ratio" suspicion factor defined below.

Definition 4. *The* **"'ratio" suspicion factor** *gives an estimate of how suspicious the fluctuation of a pattern, by considering the ratio of observed versus expected probabilities of appearance of commands. Thus:*

$$SF_{ratio}(cmd\_set) = \frac{p(cmd\_set)^{obs}}{p(cmd\_set)^{est}}$$

What the above definitions say is that the difference (definition 3) and the ratio (definition 4) of the observed probability of the patterns and the estimated probability based on our MDL approach give an estimate of how suspicious a segment is. When $SF(cmd\_set) \geq sT$, where sT is a pre-specified suspicion threshold, then the cmd_set is said to show a suspicious fluctuation.

Determining the threshold is crucial. For this have employed a randomization test. More specifically we have generated a large number of random pairs of p parameters. For each one of these pairs we have taken their difference and ratio and we have sorted them. Values of the differences and thresholds occurring in the large extreme (very large values) are a good candidates for thresholds that will guarantee high accuracy.

3.1 Identifying DoS attacks at multiple degrees of granularity

The method described in the previous subsection allows for the identification of suspicious patterns given a set of log files LF_τ. Depending on the granularity at which the DB Administrator (DBA) wants to enforce the misuse detection method he/she can follow one of the two approaches proposed below:

- *Schismatic Approach:* In this case, each relation R_r is considered separately and the corresponding sequence of its logs LF_τ^r is evaluated independently. Information that in the same command a user may refer to more than one relation is not considered. Commands of the form *select* a_1^1 *from* R_1, and *select* a_2^1 *from* R_1 are considered similar since they are both selections and they refer to the same relation R_1. In the above setting this means that we consider a sequence of τ log files each corresponding to commands referring relation R_i.
- *Catholic Approach:* In this case the database log LF_τ is considered as a whole. Given two commands **cmd1:** cmd_1 on R_i and **cmd2:** cmd_2 on R_j appearing in the log files one can proceed using one of the following two alternatives:
  - **cmd1** and **cmd2** are considered the same if $cmd_1 = cmd_2$ and different otherwise.
  - **cmd1** and **cmd2** are considered the same if $cmd_1 = cmd_2$ and $R_i = R_j$.

4 Anomaly Detection of DoS Attacks

In the anomaly detection we need: (a) to provide models and methods to describe the normal profiles; (b) to define of measures of deviation from the given profile in order to identify behaviors that are suspicious.

4.1 Building the model of normal behaviors

Assume again a series of τ training log files LF_τ, each one logging events on time-line T consisting of $|T|$ time units. Further assume that none of these log files contain an intrusion [4]. Time-line T can correspond to any real-life time-line (e.g. a single day, a week, a month etc.). with respect to which events, which in our case are query commands, occur. The goal here is to build a model that best describes the activities of normal users captured in all the τ log files. From experience and common sense we know that users' activity usually follows some patterns. For example, we can expect their activity to be similar in all mornings as well as in all evenings of the weekdays and to differentiate during the mornings and evenings of the weekends. Additionally, probably all nights, no matter the days, show similar behavior. However, these observations do not generalize so well. For example, Sunday evenings are expected to be much different from Saturday evenings, and also Friday evenings much different from other weekday evenings. Furthermore, the observed patterns may be completely different during holiday days. Therefore, forming a framework that models the set of acceptable behaviors of the users is a rather challenging task. We need a model that really captures all those variations, without being very tolerable so as to ignore possible occurrences of attacks.

We again adopt a segmentation-based model. The goal here is to find groups of log files with identical or similar segmentation points. The input in this case consists of τ sequences of observations of length $|T|$. Therefore the data can be represented as a matrix of dimension $\tau \times |T|$, namely $LF^r_{\tau \times T}$, with $LF^r(i,j)$ corresponding to observations in log file LF^r_i at time point j. We assume that the data follow a generic data distribution $\mathcal{D}$ with probability function of this distribution as $P_{\mathcal{D}}$. The segmentations of the normal log files are build using one of the following alternatives:

Loose Group Segmentation (LGS): In this case we consider the τ log files corresponding to the same time line T and the same relation R_r. The goal is to divide the set of time lines LF^r into segments in such a way that time lines of events with similar segmentation points are going to be assigned to the same cluster. The algorithmic steps for performing LGS are the following:

1. For each time line i that corresponds to log file LF^r_i compute a segmentation S_i using the method described in section 3.
2. Cluster the τ available time lines using the following similarity measure between segmentations S_i and S_j:

$$\mathrm{sim}(S_i, S_j) = \sum_{p \in S_i} \min_{q \in S_j}(|p-q|) + \sum_{q \in S_j} \min_{p \in S_i}(|q-p|) \tag{3}$$

[4] Since there is no way one can guarantee in practice that log files not to contain intrusions, we assume that LF_τ is a sequence of log files obtained by observing the user accesses and from which the suspicious elements of the sequences have been removed by following the approach that we proposed in the previous section.

where p, q run all the segmentation points of segmentations S_i and S_j respectively.

3. For each cluster randomly obtain l_c sequences to be considered as the most representative behaviors of the cluster. Behaviors that deviate from the representative descriptions are not well described by the cluster and therefore correspond to suspicious actions.

Strict Group Segmentation (SGS): A formal specification of the SGS problem is the following. Given a set of τ log files $LF^r_{\tau \times |T|}$ referring to a specific relation $R_r \in U$ identify a common segmentation $S = \{1, 2, ..., I\}$ of the τ time-lines of length $|T|$, consisting of I non-overlapping time-intervals in such a way that for each one of these non-overlapping intervals the set of all τ observations in this interval can be adequately described by a small set of classes. Again MDL is used for finding the best segmentation of the sequences. The way we proceed here is somehow different, due to the inherent difference of the setting. We need to simultaneously segment τ sequences and we require them to have the same segmentation points. Therefore, we are trying to find a global segmentation $S = \{[b_1, e_1][b_2, e_2]...[b_I, e_I]\}$ consisting of I non-overlapping segments, and each segment $s \in S$ is characterized by its beginning and ending point b_s and e_s respectively. Additionally, for each segment $s \in S$ there are n_s centers that can adequately describe the data appearing in all the τ sequences in segment s. Each such center is a vector with $e_s - b_s + 1$ elements where each element corresponds to the parameter mean value of the observations at this location. Given the above segmentation S the cost, in bits, of encoding this segmentation is determined by: (a) the cost for specifying the block boundaries ($I \log(|T|)$); (b) the cost for specifying the number of representatives per segment ($I \log \tau$); (c) the cost for coding each such representative which for segment $s \in S$ will be $n_s \rho(e_s - b_s + 1)$, with parameter ρ being the cost of encoding a real number. Sufficient accuracy is obtained for $\rho = log(\tau)/2$. Therefore the total cost of the model of segmentation S in bits is given by the following expression.

$$L(S) = I \log(|T|) + I \log \tau + \sum_{s=1}^{I} n_s \rho(e_s - b_s + 1) \tag{4}$$

Consider again a single segment $s = [b_s, e_s]$ and the time-line l with $1 \leq l \leq \tau$, LF^r_l whose values within s are denoted by $LF^r_l(s, j)$ with $b_s \leq j \leq e_s$. Additionally, consider that segment s has n_s representative centers forming the set C_s. Assume a single such center $c_s \in C_s$ and its values $c_s(j)$ with $b_s \leq j \leq e_s$. Then the probability of the data $LF^r_l(s)$ is given by $P(LF_l(s)|c_s) = \prod_{j=b_s}^{e_s} P_{\mathcal{D}}(LF_l(s, j))$, under the assumption that neighboring events are independent.

Therefore the cost of coding the data is equal to the sum of $-\log(P(LF_l(s)|c_s))$ for all segments $s \in S$ and for all time-lines $l \in [1, \tau]$, and of the cost for identifying for each segment the center that best describes the data, which is $\log(n_s)$. The way to proceed in order to construct the segmentation is the same as the one described in the previous section using dynamic programming. The main difference between the case we have considered in Section 3 and the case we consider here is that we additionally need to specify for each segment s the set of segment centers C_s that best describe the data in this segment. To this purpose we employ the k-means clustering algorithm for k varying from 1 to 5.

Combining LGS and SGS in an iterative fusion we obtain the alternative algorithm of *Combined Group Segmentation (CGS)*.

4.2 Detecting Anomalous Behaviors

Assume now that a new set of events e out of the possible set of observations $\mathcal{E}$, has been logged involving relation R_r. The goal here is to identify whether the occurrence of the set e or of any subset of it is enough raising an alarm. The set of events in e defines a time interval $i_e = [B_e, E_e]$, which has a position over the time line T. In order to be able to identify whether e contains a query flood attack or not we need to define a class of "deviation" functions $\mathcal{F}$ that allow the comparison between $e \in \mathcal{E}$ and the segmentation model $S \in \mathcal{S}$ that has already been specified from the set of all possible segmentations $\mathcal{S}$. Therefore each function $f \in \mathcal{F}$ is defined as $f : [\mathcal{E}, \mathcal{S}] \rightarrow \Re$. If there exists $f \in \mathcal{F}$ such that $f(e, S) \geq iT$, where iT is a pre-specified intrusion threshold, then a query flood attack is highly probable to exist within e.

Definition 5. *The* **deviation function** f_1 *is a mapping from* $[\mathcal{E}, \mathcal{S}] \rightarrow \Re$, *from the set of possible observed sequences of events* $\mathcal{E}$ *and segmentation models* $\mathcal{S}$ *to the set of real numbers. Given an event* $e \in \mathcal{E}$ *and a segmentation model* $S \in \mathcal{S}$, $f_1(e, S)$ *is defined as follows:*

$$f_1(e, S) = \sum_{i=1}^{E_e - B_e + 1} (e(i) - min\_c_s)$$

where min_c_s *is the value of the segment center* c *of segment* $s \in S$ *that overlaps with* s *and which has the smallest partial distance for the subset of events in* e *that overlap with* s.

Alternatively, given the segmentation S we determine for each segment $s \in S$ the upper threshold for this segment. More specifically:

Definition 6. *The* **upper threshold** $(Upper\_T(s))$ **of a segment** $s \in S$, *containing* n_s *centers that define the set* C_s *is defined as*

$$Upper\_T(s) = max_{c_s \in C_S} c_s$$

Definition 7. *The* **upper threshold** $(Upper\_T(S))$ **of a segmentation** $S \in \mathcal{S}$, *containing* I *segments* $(\{s_1, s_2, ..., s_I\})$, *is defined to be:*

$$Upper\_T(S) = \sum_{j=1}^{I} Upper\_T(s_j)$$

5 Experimental Evaluation

Here we present experiments on simulated data where accuracy is measured in terms of false positives and false negatives. A false positive occurs when the system raises an alarm for a DoS attack while there was not such an attack in the sequence. A false negative refers occurs when the system does not raise an alarm while it should have done so. Misuse and anomaly detection have been studied separately.

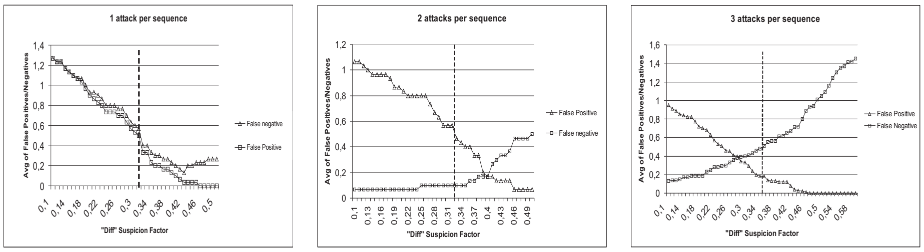

Fig. 1. Accuracy results for "Diff" suspicion factor

5.1 Misuse Detection

For the misuse detection experiments we have generated a set of sequences, each consisting of segments of data following Bernoulli distributions with varying parameters p. The parameter p used for generating the data in each segment was selected to be below a random threshold, when the threshold was containing no attack and above the random threshold otherwise . Within each sequence we have incorporated one, two or three attacks.

The results for the "diff" suspicion factor are reported in Figure 1. The plots summarize the accuracy results (the values are averages over 100 runs) of our method when using the "diff" and the "ratio" suspicion factors. The $y-$axis corresponds to the average number of false positives/negatives (over the 100 runs) while the $x-$axis to the different values of the suspicion factor. In all cases we achieve high accuracy results. The lines parallel to the y-axis in the figures correspond to the optimal thresholds for the suspicion factor ("diff" or "ratio") as output by the randomization test. A general observations from the plots is that the model selection is a tradeoff between false positives and false negatives. Depending on the application the DBA needs to decide which of the two wants to avoid (favor) most. However, in any case the accuracy of our method is satisfactory enough to provide some accuracy guarantees for a quite wide range of suspicion factor values. The results for the "ratio" suspicion factor are similar, but omitted due to space limitations.

5.2 Anomaly Detection

For the anomaly detection experimental evaluation we indicatively present the results of for LGS and SSG. The accuracy is measured by the average number of false positives and false negatives over a number of runs. For the experiments

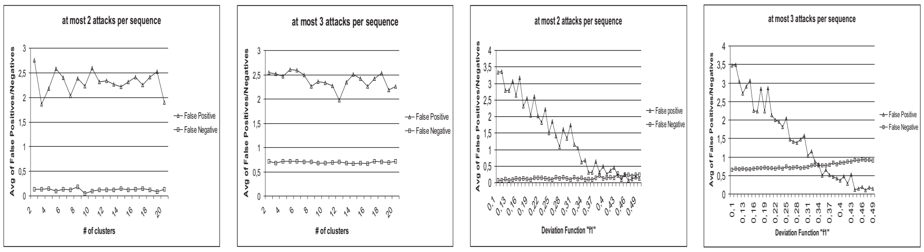

Fig. 2. LGS accuracy results wrt the number of clusters and deviation function f_1

we generated 100 normal sequences for training the model. For those sequences the probability of occurrence of an event was below a certain threshold. For testing 100 normal data sequences and 100 sequences containing artificial attacks were generated. Figure 2 summarizes the accuracy results of the LGS-related

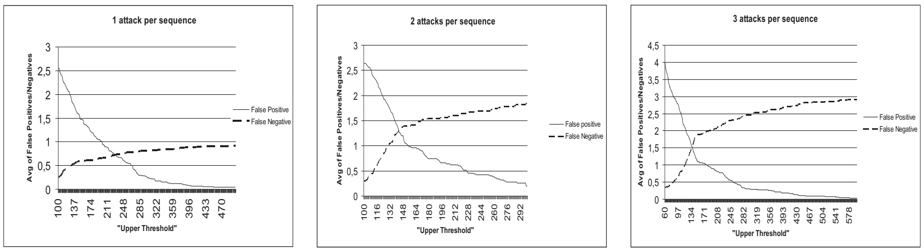

Fig. 3. SGS accuracy results with respect to the value of the "upper threshold"

experiments. The attack sequences in this case contain at most two and three attacks. The accuracy results are again averages over 100 runs. One can easily conclude that the number of clusters hardly plays any role in the overall accuracy of the method. On the other hand, as the values of the f_1 deviation function becomes larger the obtained accuracy increases, since both the number of false positives and false negatives is low. Figure 3 provides a summary of the results of the SGS experiment. Again the threshold selection is critical, and constitutes a tradeoff between the number of false negatives and false positives. Notice though that for high values of the threshold value the false positives almost tend to zero whatever the number of attacks (Figure 3).

6 Conclusions and Future Work

In this paper we have addressed the problem of query floods attacks to relational database systems. Some relevant features of our approach include access profiles incorporating time and variable protection granularity, in that one can associate a general access profile with the entire database or associate specific access profiles for specific tables. We have also evaluated our approach through some simulation and results show that our method is effective.

Indeed our methods are general and their applicability can also go beyond the DBMS. We are currently extending our work in various directions. In particular, we plan to develop a tool assisting the administrator in managing the profiles and the various parameters required in order by our approach. A second direction is related to investigating how different DBMS react with respect to query flood attacks and identify which features of such systems that make them more vulnerable to such attacks. Finally, we plan to extend our approach to protect other types of systems, namely LDAP directories and UDDI registries for Web services.

References

1. Bertino, E., Ferrari, E.: Information security. Practical Handbook of Internet Computing, M.Singh ed., CRC Press (2003)
2. Li, W.: Dna segmentation as a model selection process. In: RECOMB. (2001) 204 – 210
3. Bellman, R.: On the approximation of curves by line segments using dynamic programming. Communications of the ACM **4** (1961)
4. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: ICDM. (2001) 289 – 296
5. Cantoni, A.: Optimal curve fitting with piecewise linear functions. IEEE Transactions on Computers **C-20** (1971) 59–67
6. Axelsson, S.: Intrusion detection systems: A survey and taxonomy. In: Technical Report No 99-15, Dept. of Computer Engineering, Chalmers Univeristy of Technology, Sweden. (2000)
7. Moore, D., Voelker, G., Savage, S.: Infering internet denial of service activity. In: USENIX Security Symposium. (2001)
8. Schuba, C., Krsul, I., Kuhn, M., Spafford, E., Sundaram, A., Zamboni, D.: Analysis of a denial of service attack on tcp. In: IEEE Symposium on Security and Privacy. (1997) 208–223
9. Yu, C., Gligor, V.: A formal specification and verification method for the prevention of denial of service. In: IEEE Symposium on Security and Privacy. (1988) 187–202
10. Lane, T., Brodley, C.E.: Temporal sequence learning and data reduction for anomaly detection. ACM Trans. Inf. Syst. Secur. **2** (1999) 295–331
11. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM Trans. Inf. Syst. Secur. **3** (2000) 227–261
12. Daswani, N., Garcia-Molina, H.: Query-flood attacks in gnutella. Technical Report, Stanford University (2002)
13. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD. (1993)