SAJET

South Asian Journal of Engineering and Technology Vol.2, No.17 (2015) 223–234

ISSN No: 2454-9614

DSGA: Cloud Based Data Driven Detection of Masquerade Attacks for Multiple Users

M. Cilambarasu, L. Karthigadevi*, B. Vijayakumar, C. Vasuki

Department of Information Technology, Nandha Engineering College, Erode-52, Tamilnadu, India.

*Corresponding Author: M.Cilambarasu

E-mail: krishnakumarbtech@gmail.com

Received: 10/11/2015, Revised: 12/12/2015 and Accepted: 14/03/2016

Abstract

A masquerade attacker impersonates a legal user to utilize the user services and privileges. The semi-global alignment algorithm (SGA) is one of the most effective and efficient techniques to detect these attacks but it has not reached yet the accuracy and performance required by large scale, multiuser systems. To improve both the effectiveness and the performances of this algorithm, we propose the Data-Driven Semi-Global Alignment, DDSGA approach. From the security effectiveness view point, DDSGA improves the scoring systems by adopting distinct alignment parameters for each user. Furthermore, it tolerates small mutations in user command sequences by allowing small changes in the low-level representation of the commands functionality. It also adapts to changes in the user behaviour by updating the signature of a user according to its current behaviour. To optimize the runtime overhead, DDSGA minimizes the alignment overhead and parallelizes the detection and the update. After describing the DDSGA phases, we present the experimental results that show that DDSGA achieves a high hit ratio of 88.4 percent with a low false positive rate of S1.7 percent. It improves the hit ratio of the enhanced SGA by about 21.9 percent. Hence, DDSGA results in improving both the hit ratio and false positive rates with an acceptable computational overhead.

Reviewed by* **ICETSET'16 *organizing committee*

1. Introduction

A masquerader is an attacker who authenticates as a legal user by stealing its credentials or by violating the authentication service. An insider masquerader is a legal system user that misuses his/her privileges to access distinct accounts and perform unauthorized actions. An out-sider aims to utilize all the privileges of a legal user. Alternative implementations of this attack **[1]** do exist, such as duplication or ex-filtration of user password, installation of software with backdoors or malicious code, eavesdrop-ping and packet sniffing, spoofing and social engineering attacks. These attacks may leave some trail in log files that, after the fact, can be linked to some user. In this case, a log analysis by a host-based IDS remains the state-of-the art to detect these attacks. Attacks that do not leave an audit trail in the target system may be discovered by analyzing the user behaviours through masquerade detection. At first, mas-querade detection builds a profile for each user by gathering information such as login time, location, session duration, CPU time, commands issued, user ID and user IP address. Then, it compares these profiles against logs and signals as an attack any behaviour that does not match the profile. The current detection approaches have not achieved the level of accuracy and performance for practical deployment in spite of the large

amount of information they used to build a pro-file such as command line commands, system calls, mouse movements, opened files names, opened windows title, and network actions. Semi-global alignment (SGA) **[2]** is one of the most efficient detection algorithms and its accuracy was improved by Coull et al. **[3].** We refer to this new improvement as "Enhanced-SGA". This paper introduces the Data-Driven Semi-Global Alignment (DDSGA) approach, which improves both the detection accuracy and the computational performance of the Enhanced-SGA and of HSGAA that is also based upon SGA.

The main idea underlying DDSGA is to consider the best alignment of the active session sequence to the recorded sequences of the same user. After discovering the misalignment areas, we label them as anomalous and several anomalous areas are a strong indicator of a masquerade attack. DDSGA improves the security efficiency by using not only lexical matching such as string matching or longest common substring searches, but also by tolerating small mutations in the sequences with small changes in the low-level representation of the user commands. To this purpose, a command can be aligned with one that implements the same functionalities. To increase the hit ratio and reduce both false positive and false negative rates, DDSGA pairs each user with distinct gap insertion penalties according to the user behaviour. Furthermore, it improves both the alignment scoring system and the update phase of Enhanced-SGA to tolerate changes in behaviours without significantly reducing the alignment score. To reduce both the runtime overhead and the masquerader live time inside the system, DDSGA implements the detection and update operations in parallel threads and simplifies the alignment.

2. Sea Dataset

Currently, five data sets may be used to evaluate masquerade detection techniques: SEA, Greenberg, Purdue, RUU, and our CIDD dataset **[5], [7], [8], [9],** and **[10].** The SEA dataset has become the de-facto standard because it has been used by many researchers working on masquerade detection. In principle, this simplifies the comparison of distinct approaches. SEA consists of commands and user names collected from UNIX acct audit data. The data describe 50 different users each issuing 15,000 commands. The first 5,000 commands of each user are assumed to be genuine. The remaining commands of a user are spitted into 100 blocks of 100 commands each. Each block maybe seeded with masquerade users, i.e. with commands of other users. There is a 1 percent probability that a block is a masquerader and, in this case, there is a probability of 80 percent that the next one is a masquerader as well. As a result, approximately 5 percent of the test data are masquerades. One of the most critical defects of SEA is that it neglects command arguments and parameters. Due to the way acct collects audit data, it is difficult to distinguish commands typed by human users from those generated by shell scripts. Due to the repetitive execution of shell scripts, for some users this may result in a very regular pattern with a few commands. To address some methodological shortcomings, two alternative configurations of SEA, namely 1v49 **[11]** and SEA-I **[12],** have been defined but they have not been widely used and do not truthfully represent real world masquerader activities.

3. Related Work in Masquerade Detection

We briefly outline some masquerade detection approaches. The uniqueness approach **[6]** assumes that commands that have not been seen in the training data indicate a masquerader. Moreover, the probability that a masquerader has issued a command is inversely related to the number of users that use such a command. While uniqueness has a relatively poor performance, it is one of the few approaches that target false alarm rate of 1 percent. Na€ıve Bayes One-step Markov **[13]** is based upon one-step transitions from a command to the next. It builds two transition matrices for each user from, respectively, the training data-base and the testing one and it triggers an alarm when these matrices noticeably differ. The false alarm rate of this method is not satisfactory. The Hybrid Multi-Step Markov method **[14]** is based on Markov chains. When a Markov model cannot be adopted because too many commands in the testing data have not been observed in the training, a simple independence model with probabilities estimated from a contingency table of users versus commands may be more appropriate. Schonlau et al. **[6]** toggled between a Markov model and the simple independence one. This approach achieves the best performance among the considered methods. The main idea underlying the compression approach **[6]** is that new and old data from the same user should compress at about the same ratio. Instead, data from a masquerading user will compress at a different ratio. Among the proposed methods, this results in the worst performance. Incremental Probabilistic Action Modeling (IPAM) **[15]** is based upon one-step command transition. It estimates the probability of each transition from the training data set and uses it to predict the sequence of user commands. Too many false predictions signal a masquerader. This method is in the lowest-per-forming group. Sequence-matching **[16]** computes a similarity match between the user profiles and the corresponding sequence of commands. Any score lower than a threshold signals a masquerader. Its performance on the SEA data set is not very high. Support Vector Machine (SVM) **[17]** denotes a set of machine learning algorithms for binary data classification. It exploits a set of support vectors in the training data that outlines a hyper-plane in feature space **[17].** SVM can potentially learn a large set of patterns but it results in high false alarm rates and a low detection rate. Furthermore, the user profile has to be updated to reduce false alarms. Szymanski and Zhang **[18]** propose a recursive data mining approach that discovers frequent patterns in the sequence of user commands, encodes them with unique symbols, and rewrites the sequence with the new coding. Then, a one-class SVM classifier detects masqueraders. This approach demands mixing user data and may not be ideal or easily implemented in real-world. It also suffers of some of the SVM shortcomings. Maxion and Townsend **[11]** applied a Na€ıve Bayes classifier widely used in text classification tasks and that classifies sequences of user-command data into either legitimate or masquerader. The method has not yet achieved the level of accuracy for practical deployment. Dash et al. **[19]** introduced an episode based Na€ıve Bayes technique that extracts meaningful episodes from a long sequence of commands. The Na€ıve Bayes algorithm identi-fies these episodes either as masquerade or normal accord-ing to the number of commands in masquerade blocks. The proposed technique significantly improves the hit ratio but it still has high false positive rates and it does not update the user profile. Alok et al. **[20]** integrates a Na€ıve Bayes approach with one based on a weighted radial

basis function, WRBF, similarity. The Na€ıve Bayes algorithm includes information on the probabilities of commands by one user over the other users. Instead, the WRBF similarity takes into account the similarity measure based on the frequency of commands, f, and the weight associated with the frequency vectors. Here, f is a similarity score between an input frequency vector and a frequency vector from the training data set. The experiments confirm that WRBF-NB significantly improves the hit ratio but, as the previous approach, it suffers from the high false positive rates. Furthermore, it increases the overall overhead by computing both the Na€ıve Bayes and the WRBF and integrating their results. Lastly, it does not update the user profile and neglects the low level representation of user commands.

4. The Enhanced-SGA

4.1 The Enhanced-SGA

Coull and Syzmanski modified the SGA algorithm to handle the problems of the traditional Smith-Waterman alignment algorithm from two perspectives. The first one considers that the usage patterns of legal users may change due to changes in their role or to new software. A static user signature is therefore prone to label as attacks some variations of legal users. To avoid these false positives, the signature is updated as new behaviour is encountered by exploiting the ability of SGA of discovering areas of similarity.

The command grouping and binary scoring systems, to set the alignment scores and the gap insertion penalties. The signature update scheme is applied with the binary scoring, their most efficient system. This scheme augments both the current signature sequence with information on the new behaviours and the user lexicon with the new commands the user invokes. The scheme also introduces a threshold for each user profile to ensure that both the signature sequence and user lexicon remain free of tainted commands from masquerade attacks. The threshold is used in both detection and update processes, and it is built through a snapshot of the user signatures. The other perspective considers that the Smith-Waterman algorithm is computationally expensive and impractical to detect masquerade attacks on multi-user systems. By selectively aligning only the portions of the user signature with the highest success probability, Heuristic Aligning **[3]** can significantly reduce the computational overhead with almost no loss of accuracy in detection. These modifications have been tested on the SEA data set to simplify the comparison with other approaches.

4.2 The Data-Driven Semi-Global Alignment Approach

DDSGA is a masquerade detection approach based upon Enhanced-SGA **[3].** It aligns the user active session sequence to the previous ones of the same user and it labels the misalignment areas as anomalous. A masquerade attack is signalled if the percentage of anomalous areas is larger than a dynamic, user dependent threshold. DDSGA can tolerate small mutations in the user sequences with small changes in the low level representation of user commands and it is decomposed into a configuration phase, a detection phase and an update one. The configuration phase, computes, for each user, the alignment parameters to be used by both the detection

and update phases. The detection phase aligns the user current session to the signature sequence. The computational performance of this phase is improved by two approaches namely the Top-Matching Based Overlapping (TMBO) and the parallelized approach. In the update phase, DDSGA extends both the user signatures and user lexicon list with the new patterns to reconfigure the system parameters.

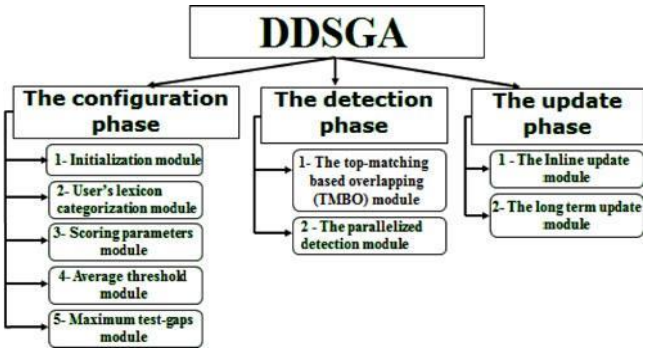


Fig:1 DDSGA Phases and modules

4.2.1 Maximum Test Gap Module

We recall that the Enhanced-SGA Heuristic Aligning decomposes the signature subsequence into 2n overlapped subsequences because if subsequences of length n are aligned, the maximum number of gaps that can be inserted into the test sequence is n for all users. By tracing the SGA algorithm, we have noticed that the maximum number of gaps is much lower than n, the length of the test sequence, and it differs for each user according to the level of similarity among the sub sequences in the user signature and to the length of the test sequence. Even if the test sequence is long enough, the number of gaps is at most half of the sequence length. This means that by partitioning the signature sequence into 2n overlapped sub sequences, the Maximum Test Gap module can divide it as in Eq. (5) to compute MFTG, the largest number of test gaps inserted into the user test sequences by the average threshold module. The Computational Enhancement (CE) for the session alignment of a user can be computed according to Eq. (6). We refer to the detection phase for an example that explains this section.

$$L \frac{1}{4} n \text{ þ } \text{Max } k \frac{1}{4} 1 \quad \frac{lts_k}{nt} \quad n ; \quad \frac{}{ntg_k}$$

Where:

- NTG is number of test gaps inserted to each test sequence,
- LTS is the length of the test sequence,
- NT is number of the test sequences of the user, fifty in case of SEA,

4.3 The Top-Matching Based Overlapping Module

To align the session patterns to a set of overlapped subsequence of the user signatures, this module uses the restricted permutation scoring system, Maximum Factor of Test Gaps (mftg) in Eq. (7), and the scoring parameters

of each user. As explained in Section 4.1, the TMBO improves the Heuritic Aligning of the Enhanced-SGA. After splitting the signature sequence into a set of over-lapped blocks of length L, see Eq. (5), it chooses the sub-sequence with the highest match to be used in the alignment process. In the worst case, all overlapped subsequence should be aligned as they have the same highest match value. We have verified that on average, the number of alignments is rather smaller because of the variation between the overlapped signature subsequence. The evaluation of the proposed TMBO approach mainly depends upon two parameters: (a) Number of average alignments for the detection process, (b) The effect of the TMBO on false alarm rates and hit ratio.

To evaluate our approach with respect to (a) we show through an example how it reduces the alignment computations. As far as concerns (b), we use the ROC curve and Maxion-Townsend. To evaluate TMBO, Fig. 10 shows the user session patterns to be aligned to the sig-nature patterns. The configuration phase returns these values: signature_gap_penalty¼9, test_gap_penalty¼5, optimal_score_sys ¼ "Restricted Permutation", detection_update_threshold ¼ 82.2%, and MFTG (maximum factor of test_gaps) ¼ 33%. The first step of TMBO computes the following length of the overlapped subsequence according to Eq. (5): ¼ ðn þ dmftgneÞ ¼ ð10 þ d33=100 10eÞ ¼ 10 þ d 3 :3e ¼ 10 þ 4 ¼ 14:

With respect to the one in the initialization module, the current overlapping runs with length L rather than 2n. Fig. 10 shows the resulting overlapped signature subsequence of size L¼14. The last subsequence, i.e. subsequence 15, may be shorter than L, but it is still longer than the test sequence.

The second step computes the match corresponding to each subsequence as shown in the front of each subse-quence in Fig. 10. We only consider the matching of subsequence 1 because those for other sub sequences are similar. If we denote by Match(X, s), the minimum between the occurrences of X in, respectively, a user session and in a subsequence.

The third step chooses the top match subsequence, e.g. sub sequences 2 and 15 in the example, as the best signature sub sequences to be aligned against the test session patterns of the user. To evaluate the reduction of the workload due to TMBO, consider the Number of Asymptotic Computations (NAC) computed by Eq. (11). In the previous example, TMBO reduces the number of alignments from 3000 to 280 with a saving of 90.66 percent.

$$NAC = Avg\_n\_align \; Sig\_len \; Test\_len \quad (11)$$

Where:

- Avg_n_align is the average number of alignments required for one detection session over all existing users.
- Sig_len is the length of the overlapped signature subsequence.
- Test_len is the length of the test sequence).

To determine if the session patterns of the current user contain a masquerader, the final step compares the highest scores of the previous two alignments against a detectio-n_update_threshold. As explained in the update phase, if atleast one of the previous eight alignments have a score larger than or equal to the detection_update_threshold, then an inline update process should be executed for the signature subsequence and the

user lexicon.

The evaluation using the SEA data set shows The evaluation using the SEA data set shows that TMBO reduces the maximum number of alignments from 49 to an average of 5.13 alignments per detection, a substantial improvement in detection scenarios. Table 6 shows the asymptotic computations for three detection approaches. The first is our TMBO without the inline update module. The second one is the Heuristic Aligning with signature update **[3].** Finally, the third one is the traditional SGA algo-rithm without the Heuristic Aligning or update feature. The NAC per one detection session can be computed as in Eq. (11). If we considered that each of the fifty users in SEA data set has one active session in a multi users system, then

Total_NAC ¼ NAC 50.

To evaluate false alarm rates and hit ratios, we have tested TMBO using the ROC curve and Maxion-Townsend score. Table 7 and Fig. 11 show that TMBO has a lower impact on the overall accuracy than other approaches. TMBO reduces the maximum number of alignments from 49 to an average of 5.13 alignments per detection, a substantial improvement in detection scenarios.

User's session patterns with length =10 (Test Sequence)

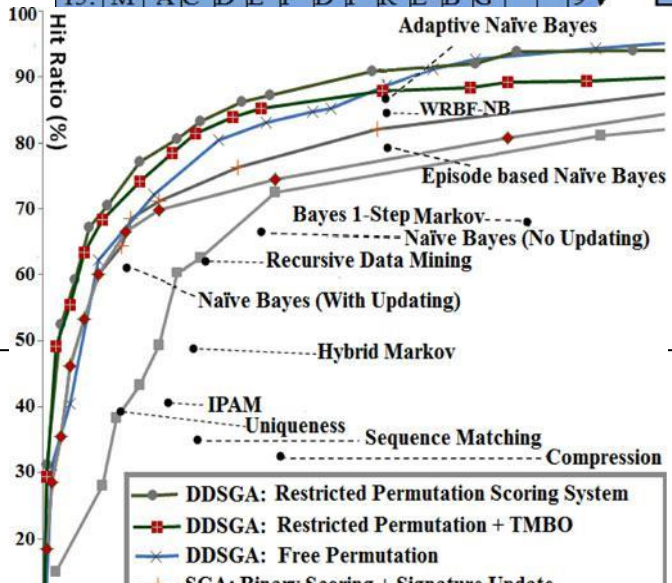
B	A	C	A	A	B	D	E	F	E

User's signature patterns with length =68 (Signature Sequence)

F	C	Y	D	D	B	A	E	F	F	K	E	C	B	G	F	A	V	E	F	M	G	I	C
H	H	N	D	F	R	C	A	A	B	V	F	E	C	G	G	H	K	Z	F	E	C	A	I
C	A	P	C	D	E	F	F	M	A	C	D	E	F	D	P	R	E	B	A				

No.	The Overlapped Subsequences														Match
1:	F	C	Y	D	D	B	A	E	F	F	K	E	C	B	8
2:	D	B	A	E	F	F	K	E	C	B	G	F	A	V	9
3:	F	F	K	E	C	B	G	F	A	V	E	F	M	G	6
4:	C	B	G	F	A	V	E	F	M	G	I	C	H	H	5
5:	A	V	E	F	M	G	I	C	H	H	N	D	F	R	5
6:	M	G	I	C	H	H	N	D	F	R	C	A	A	B	6
7:	H	H	N	D	F	R	C	A	A	B	V	F	E	C	7
8:	F	R	C	A	A	B	V	F	E	C	G	G	H	K	6
9:	A	B	V	F	E	C	G	G	H	K	Z	F	E	C	6
10:	E	C	G	G	H	K	Z	F	E	C	A	I	C	A	6
11:	H	K	Z	F	E	C	A	I	C	A	P	C	D	E	7
12:	E	C	A	I	C	A	P	C	D	E	F	F	M	A	7
13:	C	A	P	C	D	E	F	F	M	A	C	D	E	F	7
14:	D	E	F	F	M	A	C	D	E	F	D	P	R	E	6
15:	M	A	C	D	E	F	D	P	R	E	B	G			9

The top match Subsequences



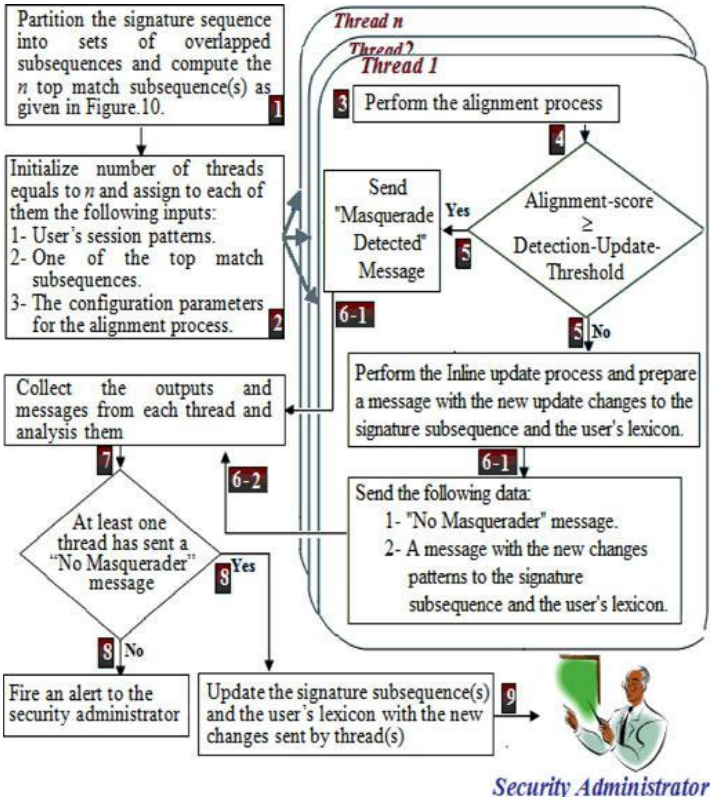


Fig. 2 processes of the parallelized detection module.

4.4 The Update Phase

The update of the user signature patterns is critical because any IDS should be automatically updated to the new legal behaviours of a user. This update is implemented by two modules: the inline update module and the long term update one.

4.4.1 The Inline Update Module

This module has two main tasks:

1) Finding areas in user signature sub sequences to be updated and augmented with the new user behaviour patterns.
2) Update the user lexicon by inserting new commands. In the detection phase, after each alignment, each parallel thread may update both the user signature subsequence and the user lexicon. Three cases are possible in the TBA, see Fig. 15:

1) The test sequence pattern matches the corresponding signature subsequence pattern,
2) A gap is inserted into either or both sequences
3) There is at least a mismatch between the patterns in the two sequences.

In case (a), no update is required because the alignment has properly used the symbol in the proper subsequence to find the optimal alignment. Also case (b) does not require an update because symbols that are aligned with gaps are not similar and should be neglected. In case (c), we consider all the mismatches within the current test sequence. Then, both the signature subsequence and the user lexicon are updated under two conditions.

The first one states that we can insert into the user signatures only those patterns that are free of masquerading records. This happens anytime the overall-alignment-score for the current test sequence is larger than or equal to the detection-update-threshold. The second condition states that the current test pattern should have previously appeared in the user lexicon or belongs to the same functional group of the corresponding signature pattern. The two conditions imply that the inline module updates the user lexicon with the new pattern if it does not belong to the lexicon. It also extends the pattern with the current signature subsequence and adds the resulting subsequence to the user signatures without changing the original one. In other words, if a pat-tern in the user lexicon or in the same functional group of its corresponding signature pattern has participated in a con-served alignment, then a new permutation of the behavior of the user has been created. For instance, if the alignment score of the test sequence in Fig. 16 is larger than or equal to the detection_update_threshold, then the pattern ‘E’ at the end of this test sequence has a mismatch with ‘A’ at the end of the signature subsequence. If ‘E’ exists in the user lexicon or belongs to the same functional group of ‘A’, the signature subsequence is augmented so that the position with ‘A’ matches with both ‘A’ or ‘E’. If ‘E’ does not belong to the lexicon, it is also inserted into it. This simply embeds observed variations in the signature sequence without destroying any information it encodes. In this case, a new augmented sub-sequence (HEE) is inserted into the user signature subsequence.

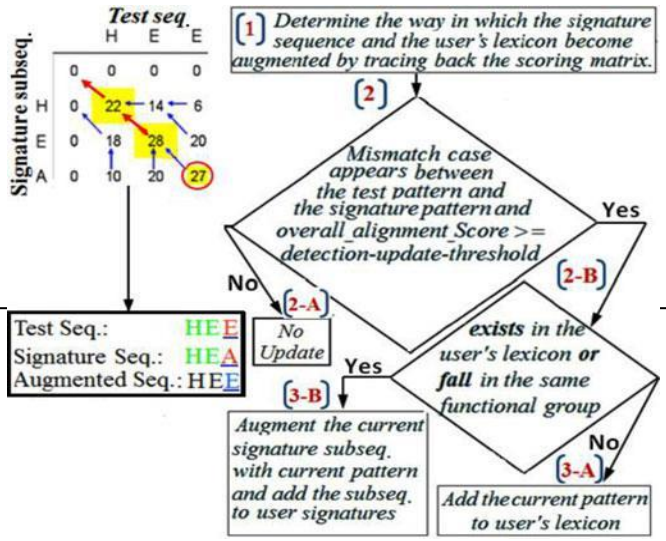
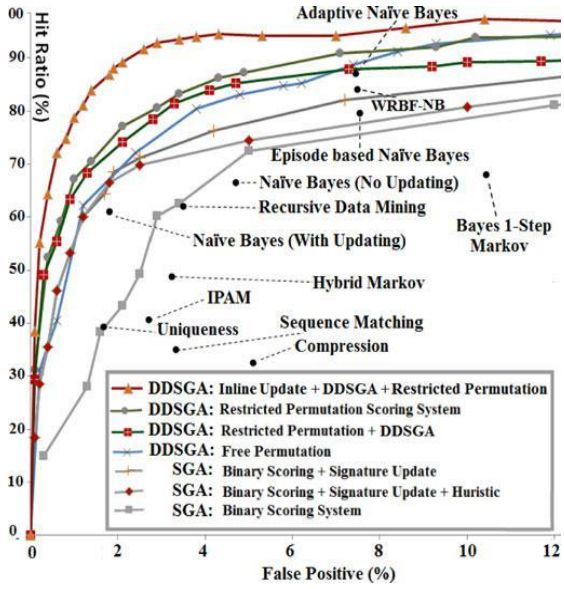


Fig.3 The inline update steps

If only the first condition is satisfied, only the user lexicon is updated so that the following checks use the new pattern. In fact, it is highly likely that a pattern that has appeared within a conserved, high scoring alignment has been created by the legitimate user. Besides improving the computational performance of system update, the module also improves the signature update scheme of the Enhanced-SGA **[3]** as following:

1) It uses the parameters, the threshold, and the scoring system returned by the configuration phase.
2) It runs in parallel with the detection phase & starts as soon as the alignment score is computed. Instead, the sign update scheme runs independently after each detection process and it repeats the backward tracing step.
3) It improves flexibility in the signature update by considering any occurrence of commands permutations or functionality matching.



To evaluate how the inline update module reduces the false alarm rates and improves the hit ratio, we have used the ROC curve and the Maxion-Townsend score after applying the inline update module. Fig. 16 and Table 8 show that the inline update module reduces the false alarm rates and increases the hit ratio. Therefore, it significantly improves the accuracy with respect to other approaches.

4.4.2 The Long Term Update Module

This module reconfigures the system parameters through the outputs of the inline update module. There are

three strategies to run the module: periodic, idle time, and thresh-old. The proper one is selected according to the characteristic and requirements of the monitored system.

The periodic strategy runs the reconfiguration step with a fixed frequency, i.e. 3 days or 1 week. To reduce the over-head, the idle time strategy runs the reconfiguration step anytime the system is idle. This solution is appropriate in highly overloaded systems that require an efficient use of the network and computational resources. The threshold strategy runs the reconfiguration step as soon as the number of test patterns inserted into the signature sequences reaches a threshold that is distinct for each user and frequently updated. This approach is highly efficient because it runs the module only if the signature sequence is changed.

5. Conclusion and Future Work

Masquerading is by far one of the most critical attacks because an attacker that can successfully logs to a system can also maliciously control it. The semi-global alignments (SGA) are based upon sequence alignment and it is one of the most effective detection techniques that can be applied to distinct sequences of audit data. While SGA may result in low false positive and missing alarms rates, even its enhanced version has not yet achieved the level of accu-racy and performance for practical deployment. This is the reason underlying the design of the Data-Driven Semi-Global Alignment Approach, DDSGA.

From the security efficiency perspective, DDSGA models more accu-rately the consistency of the behaviour of distinct users by introducing distinct parameters. Furthemore, it offers two scoring systems that tolerate changes in the low-level representation of the commands functionality by categorizing user commands and aligning commands in the same class without reducing the alignment score. The scoring systems also tolerate both permutations of its commands and changes in the user behaviour over time. All these features strongly reduce false positive and missing alarm rates and improve the detection hit ratio. In the experiments using the SEA data set, the performance of DDSGA is always better than the one of SGA. From the computational perspective, the Top-Matching Based Over-lapping approach reduces the computational load of alignment by decomposing the signature sequence into a smaller set of overlapped subsequence. Furthermore, the detection and the update processes can be parallelized with no loss of accuracy.

For future work, we plan to apply our approach to detect masquerade attacks in cloud environment by improving our CIDS framework **[4].** As a first step, we have developed a new data set, CIDD **[10]** that includes distinct audit data from distinct host operating systems and physical network environment. This will supports an evaluation of DDSGA that can use different kinds of audit sequences.

References:

[1] E. A. Breimer, “Intrusion detection: A bioinformatics approach,” in Proc. 19th Annu. Comput. Security Appl. Conf., Las Vegas, NV, USA, Dec. 2003, pp. 24–33.

[2] S. E. Coulla and B. K. Szymanski, "Sequence alignment for masquerade detection," J. Comput. Statist. Data Anal., vol. 52, no. 8, pp. 4116–4131, Apr. 2008.
[3] Hisham A. Kholidy and FabrizioBaiardi, "CIDS: A framework for intrusion detection in cloud systems," in Proc. 9th Int. Conf. Inf. Technol.: New Generations, Las Vegas, Nevada, USA, Apr. 2012, pp. 16–18. (2001) [Online]. Available: http://www.schonlau.net/intrusion. html
[4] M. Schonlau, W. DuMouchel, W. Ju, A. F. Karr, M. Theus, and Y. Vardi, "Computer intrusion: Detecting masquerades," Statist. Sci. vol. 16, no. 1, pp. 58–74, 2001.
[5] "Greenberg: Using unix: Collected traces of 168 users," Dept. Comput. Sci., Univ. Calgary, Calgary, Canada, Res. Rep. 88/333/ 45, 1988.
[6] T. Lane and C. E. Brodley, "An application of machine learning to anomaly detection," in Proc. 20th Nat. Inf. Syst. Security Conf., 1997, pp. 366–380.
[7] RUU data set: (2008) [Online] Available: http://sneakers.cs. columbia.edu/ids/RUU/data/.
[8] Hisham. A. Kholidy and FabrizioBaiardi, "CIDD: A cloud intru-sion detection data set for cloud computing and masquerade attacks," in Proc. 9th Int. Conf. Inf. Technol.: New Generations, Las Vegas, NV, USA, Apr. 2012, pp. 16–18.
[9] R. A. Maxion and T. N. Townsend, "Masquerade detection using truncated command lines," in Proc. Int. Conf. Dependable Syst. Netw., Washington, DC, USA, Jun. 2002, pp. 219–228.
[10] R. Posadas, J. C. Mex-Perera, R. Monroy, and J. A. Nolazco-Flores, "Hybrid method for detecting masqueraders using session fold-ing and hidden markov models," in Proc. 5th Mexican Int. Conf. Artif. Intell., 2006, pp. 622–631.
[11] W. Dumouchel. (1999). "Computer intrusion detection based on Bayes Factors for comparing command transition probabilities". Technical report 91, National Institute of Statistical Sciences, [Online]. Available: www.niss.org/downloadabletechreports. html
[12] W. Ju and Y. Vardi. (1999). "A hybrid high-order Markov chain model for computer intrusion detection". Nat. Inst. Statist. Sci. Research Triangle Park, NC, USA, Tech. Rep. 92. [Online]. Avail-able: www.niss.org/downloadabletechreports.html
[13] Brian D. Davison and Haym Hirsh, "Predicting sequences of user actions," in Proc. Joint Workshop Predicting Future: AI Approaches Time Ser. Anal., 1998, pp. 5–12.
[14] T. Lane and C. E. Brodley, "Approaches to online learning and concept drift for user identification in computer security," in Proc 4th Int. Conf. Knowl. Discovery Data Mining, New York, NY, USA, Aug. 1998, pp. 259–263.

[15] B. Christopher, "A tutorial on support vector machines for pattern recognition," Data Mining Knowl. Discovery, vol. 2, no. 2, pp. 121– 167, 1998.
[16] B. Szymanski and Y. Zhang, "Recursive data mining for masquer-ade detection and author identification," in Proc. IEEE 5th Syst., Man .Cybern. Inf. Assurance Workshop, West Point, NY, USA, Jun. 2004, pp. 424–431.
[17] S. K. Dash, K. S. Reddy, and A. K. Pujari, "Episode based mas-querade detection," in Proc. 1st Int. Conf. Inf. Syst. Security, 2005, pp. 251–262.
[18] A. Sharma and K. K. Paliwal, "Detecting masquerades using a combination of Na€ıve Bayes and weighted RBF approach," J. Com-put. Virology, vol. 3, no. 3, pp, 237–245, 2007.
[19] A. Sharma and K. K. Paliwal, "Detecting masquerades using a combination of Na€ıve Bayes and weighted RBF approach," J. Com-put. Virology, vol. 3, no. 3, pp, 237–245, 2007.