A Real-Time Intrusion Detection System using Data Mining Technique

Fang-Yie Leu*
Department of Computer Science and Information Engineering, Tunghai University, Taiwan
leufy@thu.edu.tw

and

Kai-Wei Hu
Department of Computer Science and Information Engineering, Tunghai University, Taiwan

ABSTRACT

Presently, most computers authenticate user ID and password before users can login these systems. However, danger soon comes if the two items are known to hackers. In this paper, we propose a system, named Intrusion Detection and Identification System (IDIS), which builds a profile for each user in an intranet to keep track his/her usage habits as forensic features with which IDIS can identify who the underlying user in the intranet is. Our experimental results show that the recognition accuracy of students of computer science department is up to 98.99%.

Keywords: Forensic Features, Data Mining, Identifying Users, Intrusion Detection, Real-time System

1. INTERODUCTION

Being widely used and quickly developed in recent years, network technologies have provided us with new life and shopping experiences, particularly in the fields of e-business, e-learning and e-money. But along with network development, there has come a huge increase in network crime. It not only greatly affects our everyday life, which relies heavily on networks and Internet technologies, but also damages computer systems that serve our daily activities, including business, learning, entertainment and so on. Forty million user files of MasterCard and VISA were exposed in 2005 when the company cooperating with CardSystem Solutions was hacked [1, 2]. Many people were forced to renew their credit cards to avoid any financial loss. This event shows the importance of network security. Besides, internal hacking is difficult to detect because firewalls and IDSs usually only defend against outside attacks.

Currently, many systems can identify who the user logging into a system is by deploying biotechnical verifications [3-9]. Most current computers check UID and password as an authentication. But hackers may install Trojans to pilfer victims' security patterns, or issue a large scale of trials with the assistance of a dictionary to access users' passwords before they can "legally" log in to a system. When successful, hackers may access users' private files or even destroy system settings. Most host-based security systems can discover an intrusion from a user's logged history afterward. And most network-based systems can detect an intrusion online [10-12]. However, to identify who the attacker is in real-time is difficult since attack packets are often issued with forged IPs.

In this paper, we propose a security system, named the Intrusion Detection and Identification System (IDIS), which mines log data to identify commands and their sequences (together named command sequences (C-sequences in short)) that a user habitually submits and follows respectively as the user's forensic features. When an unknown user logs in to a computer, the IDIS starts monitoring the user's input commands to detect whether he/she is issuing an attack. In the following, we use "hacker", "attacker" and "intruder" interchangeably as the same terms are even defined differently by different authors.

The rest of this article is organized as follows. Section 2 introduces the related research. Section 3 describes the framework and details of the IDIS. Experimental results are shown in section 4. Section 5 concludes this paper and addresses our future work.

2. RELATED WORK

Computer Forensics, which views computer systems as scenes of a crime, is computer security technologies that analyze what attackers have done. Most of their applications focus on how to identify malicious network behaviors and the characteristics of attack packets, and the way to identify attack patterns based on their analyses. Abdullah et al. [13] used package dump tools, such as tcpdump and pcap, to collect and analyze network packets and to identify network attacks from different network states and packets' distribution.

Yu et al. [14] provided another example of integrating computer forensics with IDS. A knowledge-based system was deployed to collect forensic features from malicious network behaviors. This system performed excellently in improving the hit rate of intrusion alerts.

Yin et al. [15] proposed an approach that built a Markov chain to describe users' normal operations. A state of the chain records the probability of entering the next state. However, this approach focuses on system calls generated instead of commands submitted. Chau et al. [16] used a pattern extraction technique to identify particular crime data, such as segmenting and extracting a suspect from a picture on a security video. Cabrera et al. [17] deployed sequential pattern mining to identify attack patterns that hackers frequently submit, and classified the *modus operandi* that suspects used in the commission of crimes into predefined crime types.

These techniques and applications truly contribute to network security. However, they cannot easily authenticate remote-login users, and cannot detect specific types of intrusions, e.g., when an unauthorized user logs in to a system with a legal UID and password. Authentication based on the user's operation habits is what we propose. The IDIS uses data mining and forensic techniques to respectively analyze and identify user operation characteristics, which as a kind of biological characteristics are essential in identifying a user. This system can identify attack patterns that hackers often use as well. By long-term observation, user habits can be effectively identified.

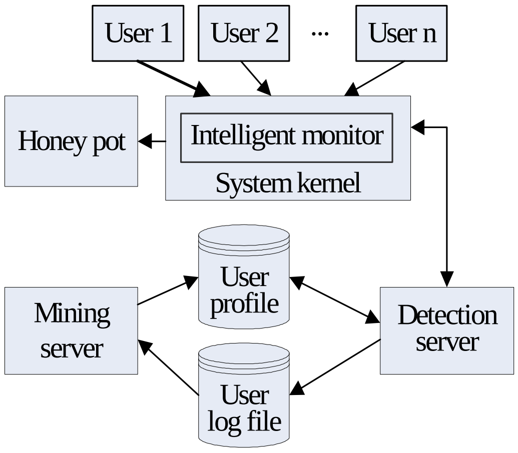


Fig. 1 The IDIS system architecture

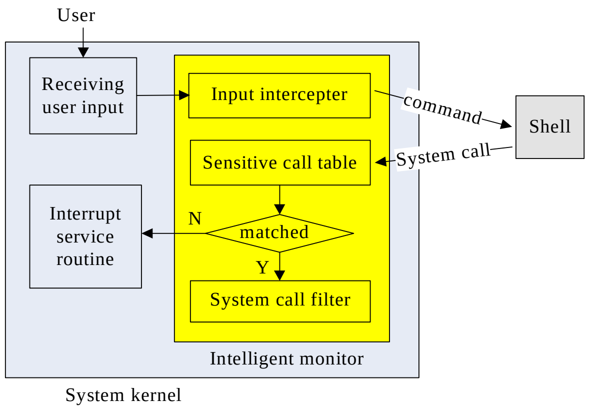


Fig. 2 Control flow of intelligent monitor on intercepting user inputs

3. SYSTEM FRAMWORK

The IDIS framework, as shown in Fig.1, consists of intelligent monitor, detection server and mining server. Intelligent monitor collects input commands from underlying user and transfers the command sequences to detection server which compares these commands with attack patterns real-time to discover attacks. In IDIS, attack patterns are represented by a reverse tree, a tree of which commands are organized in the reverse order of their arrival from the root. If matched, detection server notifies intelligent monitor to disconnect the session established for the user. Mining server analyzes log data with data mining techniques to identify user habits. The IDIS can discriminate who a underlying user is in a concerned intranet by comparing the user's current inputs with all others' habits.

Intelligent Monitor
As an extended portion of an operating system, intelligent monitor comprises input interceptor and system call filter (see Fig. 2). When a user submits a command, at least one system call will be generated. A system call generated by shell in executing a command is compared with sensitive call table, a table holding all sensitive calls. Once matched, the system call will be transferred to system call filter to check to see whether the call is safe or not. Unsafe system calls will be retained for a further analysis. A safe one will be sent to system kernel to perform its corresponding service.

Besides, we divide users into groups according to their occupations. Each group G has its corresponding inhibited commands, named class-limited command list (G) [18]. Intelligent monitor denies a user's input command immediately if the command is in the user's class-limited command list.

After the IDIS starts up, input interceptor sends a command, request-cmd-list as shown in Fig. 3, to request monitored command list which consists of last commands of all attack patterns. As an input command is in the monitored command list, this command will be held by input interceptor until detection server replies with "safe" or "unsafe".

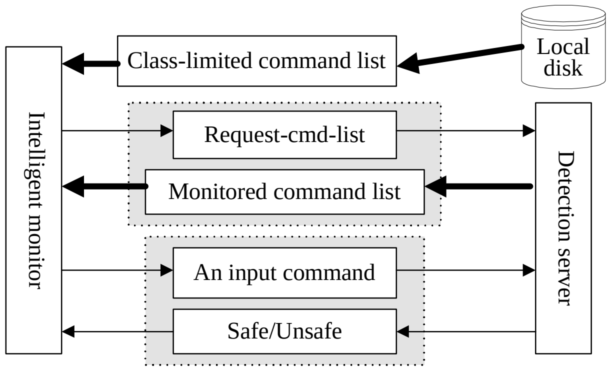


Fig. 3 Communication between intelligent monitor and detection server.

Mining Server
Mining server extracts commands that a user has habitually used from his/her log file, counts the frequency each command appears in the log file, and stores the result in the user's habit file. After that, users' habit files are mutually compared with each other to identify common and user specific command sequences, with which user profiles [18,20] can be then created.

Mining User Habits and Attack Patterns
A log file consists of many sessions. Each comprises commands a user submitted within the period of time between his/her login and the corresponding logout. Given a user's log file, the IDIS processes the commands with a sliding window of size 10, named Log-sliding window (L-window in short), to partition the commands along their submitted sequence into k-grams where k is the number of consecutive commands, k =2, 3, 4....10. Besides, another sliding window of 10 commands, named Compared-sliding window (C-window in short), is also deployed on another concerned session. This time, k' consecutive commands, preserving their submitted sequence, are extracted from C-window generating a total of (10 – k' + 1) k'-grams, k'=2, 3, 4,...10. Mining server invokes algorithm 1 to compare each of

$\sum_{K=2}^{10}(10-k+1)$ k-grams with $\sum_{K'=2}^{10}(10-k'+1)$ k'-grams by using the longest common sequence algorithm. After that, C-window shifts one command right. The procedure repeats until the last session of the log file is involved. Then Log-window shifts one input command right. The whole procedure repeats until last ten or all (if less than ten) commands of the last second session are encountered by the L-window.

Algorithm 1: **generating a habit file**
Input: a log file with *r* sections
Output: a habit file

```
{
1.  Let x=0, $0 \le x \le r$;
2.  while ( $x++ \le r-1$ )      /* from the first to the last
                                   second session */
      {y=x;
     for (each of u L-windows in session x , where
              $u = |session\ x| - 9$ if $|session\ x| \ge 10$,
              otherwise $u=1$)
       {for (each of $\sum_{k=2}^{10}(10-k+1)$ k-grams, e.g., p-gram, in
           current L-window)
         {while ( $y++ \le r$ ) /* from x++ to the last session */
            for (each of w C-windows in session y, where
             $w = |session\ y| - 9$  if  $|session\ y| \ge 10$ ,
             otherwise w=1)
             {for (each of $\sum_{k'=2}^{10}(10-k'+1)$
                k'-grams, e.g., g-gram, in C-window)
               {compare the p-gram and g-gram with the
                  longest common sequence algorithm;
                 if (the result, a common sequence,
                    does not exist in habit file)
                   insert the common sequence into
                     habit file with count=1;
                  else increase the count of the common
                       sequence by one;}
               shift C-window one command right as a
                  new C-window;}
           y=x+1;}
         shift L-window one command right as a new
           L-window;}}}
```

In a habit file, a line is a habit, also a common sequence, ended by its appearance frequency. The more frequently a common sequence appears, the higher probability the sequence is the user's habit. After the habit file is constructed, each time when the user logins and logouts later, algorithm 1 will be invoked under the situation that current session is treated as a L-window and log file sessions are processed by C-window to generate new habits and to increase habit counts.

Furthermore, we can apply algorithm 1 to known attackers' log files to extract their usage habits as attack patterns.

Create User Profiles

A user's profile is a habit file, but each habit is ended by a discrimination score instead of appearance frequency. Let DSc_{ij} be the discrimination score of command sequence j, a usage habit, submitted by a user i.

$$DSc_{ij} = \frac{H_{ij}}{\sum_{t=0}^{n-1} H_{tj}} \qquad (1)$$

Where n is the number of users in the intranet concerned, and H_{ij} is command sequence j's appearance frequency in user i's habit file. DSc_{ij} is a floating number ranging from 0 to 1 for all i and j. A user's habitual command sequence, that rarely appears or has not appeared in others' habit files, will obtain a high score. Those given low scores are commonly used command sequences.

Similarity Scores

We deploy Eq. (2),

$$W_{ij} = \frac{\mathrm{sf}_{ij}}{\mathrm{sf}_{ij} + 0.5 + 1.5\frac{ns_j}{AVG(ns)}} \times \frac{\log\left(\frac{N+0.5}{Mi}\right)}{\log(\mathrm{N}+1)},\ \text{i=1,2,...m}_i\text{, j=1,2,...N} \quad [21]$$

$$(2)$$

which is frequently used to assign a weight to a term in information retrieval domain, to calculate a weight for a command sequence. Given a set of user habit files $D = \{UP_1, UP_2, \ldots, UP_N\}$ where *N* is the number of users in an intranet. Let $T = \{CS_1, CS_2 \ldots, CS_{Mi}\}$ be the set of the sequences retrieved from *D* where M_i is the number of sequences. The weight W_{ij} of CS_i in UP_j is defined as where sf_{ij} is appearance frequency of CS_i in UP_j, ns_j total number of sequences in UP_j, *AVG(ns)* the average number of C-sequences a *UP* has, and $\log((N+0.5)/M_i)/\log(N+1)$ is the ICPF (inverse characteristics profile frequency) [21]. Given an unknown user x's current input commands *CMD*s ($1 \leqq x \leqq N$), the similarity score between *CMD*s and UP_j is defined as $SimS_{xj} = \sum_{i=1}^{p} W_{ij}$ where p is the number of sequences appearing in both of *CMD*s and UP_j.

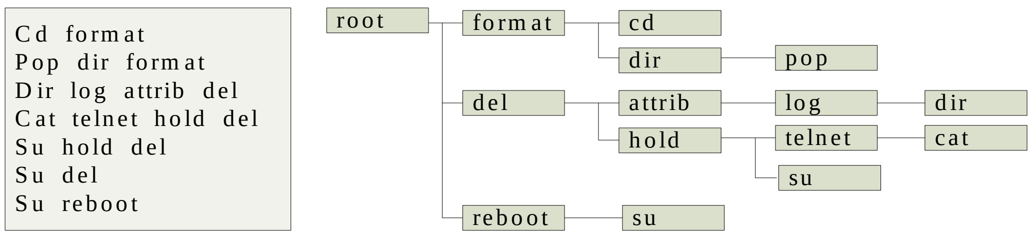
Ranking	Profile	Similarity score
Rank 0	sXX2845.pro	1267.878
Rank 1	sXX2816.pro	1256.459
<u>Rank 2</u>	<u>sXX2819.pro</u>	940.535
...		…
Rank 174	sXX2852.pro	0.0
a total of 655 user profiles, @ rank 2, decisive rate: 99.695%, cost 1906ms		

Fig. 4 The identification of user x given sXX2819's input commands as an unknown user's current inputs. x is ranked the second.

Fig. 4 lists experimental results given a user x (=sXX2819)'s input commands as an unknown user's current inputs. The higher a similarity score, the higher probability x is the user who submits these input commands.

Detection Server

As stated above, attackers' common behaviors are represented by a reverse tree, named common reverse tree.



```
Cd format
Pop dir format
Dir log attrib del
Cat telnet hold del
Su hold del
Su del
Su reboot
```

(A) Attack patterns **(B)** The corresponding common reverse tree

Fig. 5 Representation of attackers' common behaviors

Fig. 5 gives an example. Nodes of the first level are last commands of attackers' common behaviors. Monitored command list consists of all commands of the first levels of the common reverse tree. Assume that user has entered m commands $\{C_1, C_2, \ldots C_m\}$ after his/her login where C_i is submitted prior to C_{i+1}, i = 1, 2, … , m-1. To detect intrusion on-line, the underlying input command C_i, $1 \leqq i \leqq m$, is compared with monitored command list first. If matched, a modified deep-first search traversal algorithm will start traversing the sub-tree of the common reverse tree which is rooted at C_i to see whether the user's inputs, C_j to C_{i-1}, j = 1, 2,….or i-2, in the reverse order of their arrival can finally reach a leaf node or not.

In this algorithm, when one input command C_j matches node N_k, a node at level k, in the common reverse tree, and C_{j-q}, q = 1, 2, …, or j-1, with the smallest q (if there are several such C_{j-q}) also matches one of N_k's immediate child nodes, e.g., N_{k+1}, then let $C_j = C_{j-q}$ and try to find another C_{j-q} that matches one of N_{k+1}'s immediate child nodes. Each time when no such C_{j-q} can lead the algorithm to arrive at a leaf, the algorithm backtracks to the last second matched node and tries to find $C_{j-q'}$ that matches N_k's another immediate child where $C_{j-q'}$ locates between C_1 and C_{j-q} (since on last match on N_k, the smallest q is chosen as the next command that matches N_{k+1}, therefore, it should be that $q' \geqq q$), and q' is the smallest integer if several q's exist. When one is found (at level k+1), the algorithm looks for a matched node at level k+2. The procedure repeats until no such C_{j-q} can match N_k's immediate child or the algorithm reaches a leaf node. The latter is considered as an attack. Detection server replies intelligent monitor with an "unsafe" (see Fig. 3).The former means it is safe.

4. EXPERIMENTS

We collect 253 computer-science students' log files and 206 non-computer-science students' log files from computer center of TungHai University as the experimental data.

All commands extracted from a computer-science student's log file are saved in the user's habit file in accordance with their original submitted sequence. Others parameters, e.g., time and date, are removed to simplify the scope of the following experiments. To perform privacy preserving experiments [22], we hid user IDs for all users in the data set involved.

Identifying Unknown Users

The first experiment, to identify an unknown user, is performed ten times, each time we select different 75% commands from each log file as the training data to generate his/her profile. The remaining 25% are the test data (a test file). Given an unknown user's test data, if a user's similarity score, the average of its ten-fold values, is within first x% of all users, we say the decisive rate is x (=(|UPs|-avg_rank)/|UPs|*100%)%, $0 \leq x \leq 100$, where |UPs| is the number of user profiles, and avg_rank, average of all users' ranks, is defined as

$$\frac{\sum_{i=1}^{|UPs|}\left(\sum_{j=1}^{q} rank_{ij} / q\right)}{|UPs|}$$

where q (=10) is the times of experiment and $rank_{ij}$ is user i's ten-fold ranks. The experimental result shown in Fig. 6 depicts that avg_rank=2.56, and the average decisive rate x of all users is 98.99% (= (253-2.56)/253*100%). "Rank P" and "Cost Y ms" respectively represent that a user profile is ranked P and the time required to compare the user's test file (i.e., 25%) with 253 profiles (i.e., 75%) is Y ms.

	User's test file	Rank	Decisive rate (%)	Cost (ms)
1	sXX2808.tst	0	100.00	31
2	sXX2959.tst	75	88.55	375
3	sXX2811.tst	160	75.57	328
…	…	…	…	…
108	sXX2849.tst	5	99.24	62
…	…	…	…	…
253	sXX2902.tst	2	99.69	828

A total of 253 user profiles (75% portion)
A total of 253 test files (25% portion)
Total cost = 20108 msec
Average rank = 2.56
Average decisive rate = 98.99 %
Average number of commands per user profile = 165.96
Average number of commands per test file = 56.8

Fig. 6 A part of experimental results generated by a cross comparison by comparing each user's test data (25% of a profile) with 253 user profiles (75%). (sXX28XX and sXX29XX are computer science students)

Students of non-computer-science departments often use simple and common commands, and of their log sessions are often short and highly similar. Thus, the average decisive rate of 206 non-computer-science department students goes down to 93.33%.

Detection of Single Command Attack Pattern

To detect single command attack pattern, we randomly inserted twenty different attack commands AC1 to AC10 to a legal user's log file. Before retrieving the user's habits, we ripped off three types of patterns including those that are legal and safe, that contain more than one attack command, and that are not ended by an attack command. We deleted the third because those located after an attack command are safe according to the definition of single command attack pattern. Fig. 7 shows several habits of a user. Each is ended by a single command attack pattern.

```
cat cat ls cd ls AC9
cat cat ls ls AC3
cat cd cd AC2
cd vi AC6
ls cd quota AC1
ls clear ls w AC8
…
```

Fig. 7 User habits, each line is ended by a single command attack pattern

After that, we selected a% of user log contents from each log file and b% of collected attack patterns, and mixed them to simulate attackers' inputs, which will be described below.

From 459 user log files, we extracted 76,341 commands, from which 1423 attack patterns were extracted. Let a = b = 15 and a sliding window of size ten is used to mix an attack pattern and legal commands. Table 1 shows the mixed patterns and the detection results. The field "ID of an inserted pattern (inserted location)" lists IDs of single command attack patterns followed by their locations. They are generated by the following procedure. Given an attack pattern ($C_1, C_2, \dots, C_x$), which were mixed with 10-x consuetude log file commands R={ $C'_{q-(10-x)}$, $C'_{q-(10-x)+1}, \dots, C'_{q-1}$} in the underlying session by randomly inserting C_i, i = 1, 2, 3, …x-1, to any position among the elements of R under the constraint that C_x is located at C'_q position and C_i should be prior to C_{i+1} where C_i is ith command of the pattern. After insertion, C_i might be adjacent to C_{i+1} or separated by several commands, and now the location of C_x became q+x-1. For example, #793(11) in the first record of Table 1 represents that the last command of attack pattern #793 (cd cd ls AC2), x=|#793| = 4, is inserted into the 8th (i.e., qth) position of log 01982. Original C'_8 becomes C'_9. Commands cd, cd and ls are mixed with those in { $C'_2, C'_3, \dots, C'_7$ }. After that, C_x becomes the 11th command.

Table 1 Patterns obtained by mixing log contents with single command attack pattern and the detection results

	Mixed patterns		Detected attack patterns	
User log file	No. of inserted patterns	ID of an inserted pattern (inserted location)	No. of detected patterns	ID of a detected pattern (detected location)
log01982	2	#632(4), #793(11)	2	#2000(4), #1257(11)
log02864	19	#7(5), #1203(12), …	19	#1(5), #100(12), #1270(20),…
log02871	9	#1382(6), #1409(15), …	9	#810(6), #1406(15), …
…	…	…	…	…

Nevertheless, the commands coming from both of #793 and R individually follow their original submitted sequence.

The field "ID of a detected pattern (detected location)" represents the ID of a detected pattern followed by a detected location which is the position of the pattern's last command. For example, #1257(11) in the first record shows that attack pattern #1257 (ls AC2), ended at position 11 of log01982, is detected. Basically, a shorter pattern may be contained in a longer one. Hence, the ID of a detected pattern may be different from that of the inserted since #1257 is a proper subset of #793. From Table 1, we can conclude that the number of patterns inserted is equal to that of discovered, i.e., precision = recall = 100%. The detection process costs 348 ms.

Detection of Multi-stage Attack Pattern

A log file consisting of multi-stage attack patterns. Each line has several stages. Each stage in turn is a user habit ended by a single command attack pattern. For detecting multi-stage attack patterns, again, let a = b = 15 given nineteen stages, S1 to S10, from which 58 different attack commands are retrieved, i.e.,

$|\bigcup_{i=1}^{10}\{C_r \mid C_r \textit{ is a command}, C_r \in S_i\}| = 58$ where

$\{C_p \mid C_p \textit{ is a command}, C_p \in S_i\}$ $\cap$

$\{C_q \mid C_q \textit{ is a command}, C_q \in S_j\} = \phi$, i, j = 1, 2, …9, and i $\neq$ j.

That means, S1 to S10 are non-cross-reference patterns so that all the IDs of detected patterns are exact the same as those of the inserted ones. The common reverse tree is six levels in height, and 76,341 commands extracted from 459 user log files were compared, costing 210 ms. Also, the precision = recall = 1.

5. CONCLUSIONS

In this article, we bring up an approach to find out users' habits by deploying data mining and forensic techniques. To identify the representative C-sequences for a user, the frequency that a habitual command sequence appears in the user's log file is counted and its discrimination score is calculated so that the user's profile can be established. By comparing a user's current input commands with all others' profiles, the IDIS can identify who the user is. The accuracy is high enough to make the IDIS be a valuable auxiliary subsystem in a closed environment to assist the identification of an internal hacker. Of course, a new user whose user profile has not been established will not be a candidate to be identified. Meanwhile, a user's input commands are compared with the common reverse tree in which all commands of an attack pattern are organized in their reverse order so as to real time detect whether underlying inputs are an attack or not. Employing the common reverse tree can lightweight IDIS and lower the load of detection server.

Moreover, accurately and completely collecting user behaviors on much more basic operations, such as system calls instead of commands, is much more helpful in detecting hackers and identifying a user. Such will also help us to collect intrusion behaviors in a system that employs GUI interface.

However, how to process and mine such a huge volume of data may be the first challenge. Several papers have addressed this topic [15,23]. But many systems have not been implemented, and many did not describe their implementation. Additionally, to detect an attack and respond real time, we need a fast algorithm and a distributed computing environment to speedup data processing since the time complexity of algorithm 1 is high. Cluster and/or Grid computing should be the candidates. Besides, a mathematical analysis on the IDIE's behaviors so as to build its formal performance and cost models is interesting. Those are our future research topics.

6. REFERENCES

[1] B. Schneier, "CardSystems Exposes 40 Million Identities," http://www.schneier.com/blog/archives/2005/06/cardsystems_exp.html

[2] A.P. Mitchell, “40 million,” http://www.theinternetpatrol.com/cardsystems-compromises-data-of-40-million-mastercard-and-visa-cardholders

[3] Y. Sheng, V. V. Phoha, and S. M. Rovnyak, “A Parallel Decision Tree-Based Method for User Authentication Based on Keystroke Patterns,” IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, vol. 35, no. 4, August 2005, pp.826-833.

[4] M. Ray, P. Meenen, and R. Adhami, “A Novel Approach to Fingerprint Pore Extraction,” IEEE Southeastern Symposium on System Theory, March 2005, pp.205-208.

[5] D. Krašnjak, and V. Krivec, “Fingerprint Classification Using a Homogeneity Structure of Fingerprint’s Orientation Field and Neural Net,” International Symposium on Image and Signal Processing and Analysis, 2005.

[6] W. Yuan, Z. Lin, and L. Xu, “A Rapid Iris Location Method Based on the Structure of Human Eyes,” Engineering in Medicine and Biology Annual Conference, September 2005.

[7] P. Meenen, and R. Adhami, “Approaches to Image Binarization in Current Automated Fingerprint Identification Systems,” IEEE Southeastern Symposium on System Theory, March 2005.

[8] L. Nanni, and A. Lumini, "A Deformation-invariant Image-based Fingerprint Verification System," NeuroComputing, vol.69, no.16, October 2006, pp.2336-2339.

[9] A. Antonelli, R. Cappelli, D. Maio, and D. Maltoni, "Fake Finger Detection by Skin Distortion Analysis," IEEE Transactions on Information Forensics and Security, vol.1, no.3, September 2006, pp.360-373.

[10] www.snort.org

[11] http://en.wikipedia.org/wiki/Snort_%28software%29

[12] F.Y. Leu, J.C. Lin, M.C. Li, and C.T. Yang, “A Performance-Based Grid Intrusion Detection System,” International Computer Software and Applications Conference, July 2005.

[13] K. Abdullah, C. Lee, G. Conti, J.A. Copeland, “Visualizing Network Data for Intrusion Detection,” the IEEE Workshop on Information Assurance Workshop, June 2005.

[14] J.Q. Yu, Y.V.R. Reddy, S. Selliah, S. Kankanahalli, S. Reddy and V. Bharadwaj, “TRINETR: An Intrusion Detection Alert Management System,” SIPLab, Concurrent Engineering Research Center Lane Department of Computer Science and Electrical Engineering, June 2004, pp. 235-240.

[15] Q. Yin, L. Shen, R. Zhang, and X. Li, “A New Intrusion Detection Method Based on a Behavioral Model,” World Congress on Intelligent Control and Automation, June 2004, pp.4370-4374.

[16] M. Chau, J.J. Xu, and H. Chen, “Extracting Meaningful Entities from Police Narrative Reports,” National Conference on Digital Government Research, 2002, pp. 271-275.

[17] J.B.D. Cabrera, L. Lewis, and R.K. Mehra, “Detection and Classification of Intrusion and Faults Using Sentences of System Calls,” SIGMOD Record, vol.30, no.4, 2001, pp.25-34.

[18] F. Dridi, B. Muschall, and G. Pernul, “Administration of an RBAC system," International Conference on System Sciences, 2004, pp.1-6. http://csdl2.computer.org/comp/proceedings/hicss/2004/2056/07/205670187b.pdf

[19] Y. Okazaki, I. Sato and S. Goto, “A new Intrusion Detection Method Based on Process Profiling,”. Symposium on Applications and the Internet, Feb. 2002, pp. 82 -90.

[20] J.E. Dickerson and J.A. Dickerson, “Fuzzy Network Profiling for Intrusion Detection,” International Conference of the North American on Fuzzy Information Processing Society, July 2000, pp. 301 -306.

[21] A. Leuski, “Evaluating Document Clustering of Interactive Information Retrieval,” ACM CIKM, Nov. 2001, pp. 33-40.

[22] V.S. Verykios et al., “State-of-the-Art in Privacy Preserving Data Mining,” SIGMOD Record, vol.33, no.1, March 2004, pp.50-57.

[23] K. Lu, Z. Chen, Z. Jin, and Jichang Guo, “An Adaptive Real-Time Intrusion Detection System Using Sequence of System Call, ” IEEE Electrical and Computer Engineering, May/mai 2003, pp.789-792.