Design of Anomaly Detection System for Outlier Detection in Hardware Profile Using PCA

Hari Om, Tanmoy Hazra
Department of Computer Science & Engineering
Indian School of Mines, Dhanbad 826004, India
[1]hari.om.cse@ismdhanbad.ac.in, [2]tanmoyhazra316@gmail.com

Abstract
In this paper, we design an Anomaly Detection System for Outlier Detection in Hardware Profile by using Principal Component Analysis (PCA) that helps reduce the dimension of data. Anomaly detection methods can detect new intrusions, but they suffer from false alarms. Another approach is misuse detection that identifies only known attacks by matching with the previous patterns. Host based Intrusion Detection Systems (HIDSs) use anomaly detection approach to identify malicious attacks i.e. intrusion. Data being of large dimensional generates features in terms of large set of dimensions and hence the system takes considerable time for processing the huge amount of data. The PCA is used to reduce the dimensionality of the host based data without any loss of useful information such as non-redundant data. We experimentally show that the proposed intrusion detection system has detection rate in the range of 90% - 97.5% and false alarm rate in the range of 2.5% - 7.5% depending upon the major and minor principal components.

Keywords: Anomaly Detection, Outlier Detection, PCA, Mahalanobis Distance, False alarm rate

1. Introduction

With the explosive rapid expansion of computers in last decade and so, their security has become an important issue. The process of monitoring the events occurring in a computer system and analyzing them for identifying intrusions is known as intrusion detection technique and the system is known as intrusion detection system (IDS). An intrusion is defined as an attack in a network or system by an intruder that compromises the security parameters such as integrity, confidentiality, and authentication of the system. The attacks can be external attacks, internal penetrations, and misfeasors. An intruder tries to get access into a system for which he/she is not authorized. An Intrusion Detection System (IDS) is a program that analyzes the events that have taken place or those happen during an execution and it tries to find indications of misuse of the computer. Host based Intrusion Detection Systems (HIDSs) monitor suspicious activities that take place in the system. The HIDSs can be either anomaly detection that is based on statistical measure or misuse detection that is based on signature. Anomaly detection is used to capture the changes in behavior that are not normal. These methods use as input the training data to build normal system behavior models that signal alarms when there is any abnormal activity which deviates from the normal model. These models may be generated using different approaches such as statistical analysis, data mining algorithms, genetic algorithms, artificial neural network, fuzzy logic, rough set. Anomaly detection methods have problems of false positive and false negative. Since the numbers of new attacks are increasing and the variations of known attacks cannot be recognized by misuse detection. Therefore, we develop an intrusion detection system using Principal Component Analysis (PCA) that detects the outlier data.

2. Related Work

There are several works related to intrusion detection in literature [1-6]. The principal component analysis (PCA) is one of the important approaches that are used to reduce the data size and also detect errors in multivariate data [3]. The Chi-square distribution is also very useful statistical approach in detecting anomalies. Shyu discusses an intrusion predictive model that uses PCA and Chi-square distribution for KDD1999 dataset [1]. For detecting anomaly in a system, monitoring of its behaviour is required. If there is abnormal behaviour in the system, one can suspect some security violation. In [2], an intrusion detection model based on security violations that is capable of detecting break-ins, penetrations and other types of computer attacks is discussed. Ye uses Chi–square statistic to develop an anomaly detection technique that has 0% false alarm rate and 100% detection rate [4]. Puketza provides a comparative study of detection rate and false alarm rate by using Hotelling's T^2 test and Chi-squared distance test. He has reported experimentally that the Chi-square distribution has better performance than the Hotelling's T^2 test [14]. Chen et. al discuss an efficient filtering scheme that requires only 0.3% of the original traffic volume for anomaly [17]. Casas et. al discuss an unsupervised network intrusion detection system that can detect unknown network attacks without using any kind of signatures, labeled traffic, or training [18]. In this paper, we use PCA methodology to detect intrusion and our proposed system has detection rate in the range of 90% - 97.5% and false alarm rate in the range of 2.5% - 7.5% depending upon the major and minor principal components. The rest of the paper is organized as follows: section 3 discusses the proposed work. Experimental methodology has been discussed in section 4, Results and Discussions are given in section 5. Finally, the conclusion is given in section 6.

3. Proposed Work

The PCA is a common technique to find the patterns in the data of high dimension. It basically reduces the number of dimensions in an input data set without losing its useful information. In PCA technique, a set of principal components are obtained that constitute an orthogonal set of eigenvalue and eigenvector pairs. The set of principal components, also called axes, best suits the data. In our proposed scheme, these set of axes represent features' normal data. Outlier detection occurs by mapping the used data to these normal axes in order to find the distance from the axes. If the distance is greater than a certain threshold, it is assumed that there is an attack i.e. outlier detection. The principal components are linear combinations of *m* random variables (features of used data), denoting them as $X_1, X_2, \ldots, X_m$, that have two important properties:

- They are uncorrelated, and sorted in descending order.
- Their total variance, denoted by R, is the summation of variances of each variable $X_1, X_2, \ldots, X_m$, i.e.,

$$R = \sum_{i=1}^{m} R_i \text{ , where } R_i \text{ is variance of } X_i$$

Assume that the original data is represented in matrix form with *n* observations, each observation has *m* attributes i.e. X_{nxm}. Let ρ_{mxm} and $\sum_{mxm}$ be the symmetric correlation and variance-covariance matrices of X_1, $X_2,\ldots,X_m$, respectively. $X= [X_1, X_2, \ldots, X_m]^T$ denotes the observation data matrix. Let the correlation matrix be the *mxm* symmetric matrix as given below:

$$\rho = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1m} \\ \rho_{12} & \rho_{22} & \cdots & \rho_{2m} \\ . & . & & . \\ . & . & & . \\ \rho_{1m} & \rho_{2m} & & \rho_{mm} \end{pmatrix}$$

where the correlation coefficient ρ_{ik} measures the amount of linear association between X_i and X_k that is defined in terms of covariance σ_{ik} and variances σ_{ii} and σ_{kk} as follows:

$$\rho_{ik} = \sigma_{ik} /(\sqrt{\sigma_{ii}} \sqrt{\sigma_{kk}})$$

Thus, the correlation matrix ρ can be defined as follows:

$$\rho = \begin{pmatrix} \sigma_{11} / (\sqrt{\sigma_{11}} \sqrt{\sigma_{11}}) & \sigma_{12} / (\sqrt{\sigma_{11}} \sqrt{\sigma_{22}}) & \ldots & \sigma_{1m} / (\sqrt{\sigma_{11}} \sqrt{\sigma_{mm}}) \\ \sigma_{12} / (\sqrt{\sigma_{11}} \sqrt{\sigma_{22}}) & \sigma_{22} / (\sqrt{\sigma_{22}} \sqrt{\sigma_{22}}) & \ldots & \sigma_{2m} / (\sqrt{\sigma_{22}} \sqrt{\sigma_{mm}}) \\ . & . & & . \\ . & . & & . \\ \sigma_{1m} / (\sqrt{\sigma_{11}} \sqrt{\sigma_{mm}}) & \sigma_{2m} / (\sqrt{\sigma_{22}} \sqrt{\sigma_{mm}}) & \ldots & \sigma_{mm} / (\sqrt{\sigma_{mm}} \sqrt{\sigma_{mm}}) \end{pmatrix}$$

$$\sum = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1m} \\ \sigma_{12} & \sigma_{22} & \ldots & \sigma_{2m} \\ . & . & & . \\ . & . & & . \\ \sigma_{1m} & \sigma_{2m} & \ldots & \sigma_{mm} \end{pmatrix}$$

where σ_{ik} represents the covariance between i^{th} and k^{th} attributes defined below:

$$\sigma_{ik} = 1/(n-1) \sum_{i=1}^{n} (X_i - \overline{X_i})(X_k - \overline{X_k})$$

and σ_{ii} represents the variance of i^{th} attribute.

and let V_{mxm} be standard deviation matrix that is defined below:

$$V^{1/2} = \begin{pmatrix} \sqrt{\sigma_{11}} & 0 & \dots & 0 \\ 0 & \sqrt{\sigma_{22}} & \dots & 0 \\ . & . & & . \\ . & . & & . \\ 0 & 0 & \dots & \sqrt{\sigma_{mm}} \end{pmatrix}$$

Then, it can be easily verified that

$$V^{1/2}\rho V^{1/2} = \Sigma$$

We can also write

$$\rho = (V^{1/2})^{-1}\Sigma(V^{1/2})^{-1}$$

The principal components may also be obtained for the standardized variables: $Z_1, Z_2,...,Z_m$ using the following equation:

$$Z_i = (X_i - \overline{X_i})/\sqrt{\sigma_{ii}} \qquad \text{for } i=1, 2, \dots, m.$$

$\overline{X} = [\overline{X}_1, \overline{X}_2, ..., \overline{X}_m]^T$ is the mean vector of X which is having *m* attributes/components i.e. $X= [X_1, X_2, \dots, X_m]$. We can also represent it in a matrix form of dimension *mxm*: $Z = (V^{1/2})^{-1}(X- \overline{X})$, where $Z = [Z_1, Z_2, \dots, Z_m]^T$ the column vector of the standardized observation data X. The principal components of Z are obtained from the eigenvectors of the correlation matrix ρ. Let Y_i be the i^{th} principal component of Z and (λ_i, e_i) represent the i^{th} eigenvalue/eigenvector pairs among *m* eigenvalues from ρ. If (λ_1, e_1), (λ_2, e_2), ..., (λ_m, e_m) are *m* eigenvalue-eigenvector pairs, where $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m \geq 0$, the i^{th} principal component is given by

$$Y_i = e_i^T Z$$

$$= e_{i1}Z_1 + e_{i2}Z_2 + ... + e_{im}Z_m, \quad i = 1, 2, ..., m, \quad \text{where } e_i \text{ is given by}$$

$$e_i = \begin{pmatrix} e_{i1} \\ e_{i2} \\ . \\ . \\ e_{im} \end{pmatrix}$$

Each eigenvalue of a principal component corresponds to the amount of variation it has. The larger eigenvalues are more significant and correspond to their projected eigenvectors. The points which lie at a far distance from these axes would exhibit abnormal behavior that can easily be identified. Using a suitable threshold value, the normal system generated data with Mahalanobis distance greater than the threshold is considered as an outlier and it is an attack. If the data is in the threshold boundary, sometimes it alerts as intrusion. The sum of squares of the partial principal component values equals to the principal component value that is given as follows:

$$\sum_{i=1}^{m} Y_i^2/\lambda_i = Y_1^2/\lambda_1 + Y_2^2/\lambda_2 + \dots + Y_m^2/\lambda_m$$

This sum is nothing but Mahalanobis distance of the dataset X from the mean of the normal sample dataset [9]. In general, Mahalanobis distance between two vectors x and y is calculated by

$$d^2(x, y) = (x - y)^T \rho (x - y), \text{ where } \rho \text{ is the sample correlation matrix.}$$

Here, the major principal components value is used to detect extreme deviations with large values and minor principal components value is used to detect slight deviations on the normal dataset. Thus, two thresholds are needed to detect attacks. Let q & r be the most significant principal components and least significant principal components and T_q & T_r be the thresholds for the major principal component and minor principal component. We say that an attack occurs for any observation of X if any one of the following condition is satisfied:

$$\sum_{i=1}^{q} Y_i^2/\lambda_i > T_q \qquad \text{or} \qquad \sum_{i=m-r+1}^{m} Y_i^2/\lambda_i > T_r$$

These inequalities contain square of projections on the axes normalized by corresponding eigen values. The first inequality contains the sum of squares of first *q* principal component values (projections on first *q* axes) and the second one contains sum of squares of last *r* principal component values (projections on last *r* axes). If the first sum is greater than the threshold value T_q or the second sum is greater than the threshold value T_r, then there is large deviation and such deviations are termed as abnormal behaviour of the system, i.e., an attack. Now we discuss confusion matrix that helps computing recall, precision, detection rates and false alarm rates.

Confusion matrix:

False alarm rate and detection rate can be calculated using the confusion matrix that is given below.

Actual Class \ Predicted Class	C	NC
C	TN	FP
NC	FN	TP

Fig. 1 Confusion matrix

C – Anomaly class
NC – Normal class
TN – True Negative
FN – False Negative
TP – True Positive
FP – False Positive

Recall (R) = TP / (TP+ FN)
Precision (P) = TP / (TP+FP)
F-measure = 2*R*P/(R+P)
= $((1+\beta^2).R.P) / (\beta^2.R+P)$
where β is the relative importance of precision vs recall and it is usually set to 1.

First, we calculate the mean vector for all the attributes that have been used for our experimental datasets. Then, we calculate the correlation matrix followed by the eigenvalues and eigenvectors from the correlation matrix. In order to calculate the principal components – major or minor- we sort the eigenvalues and their corresponding eigenvectors. We compute the summation of major and minor principal components and determine corresponding suitable threshold values from the normal dataset and compare with each observation of the mixed dataset. In order to evaluate the detection rate and false alarm rate accurately, we have used confusion matrix. The flow chart of the entire process is shown in the following Fig. 2.

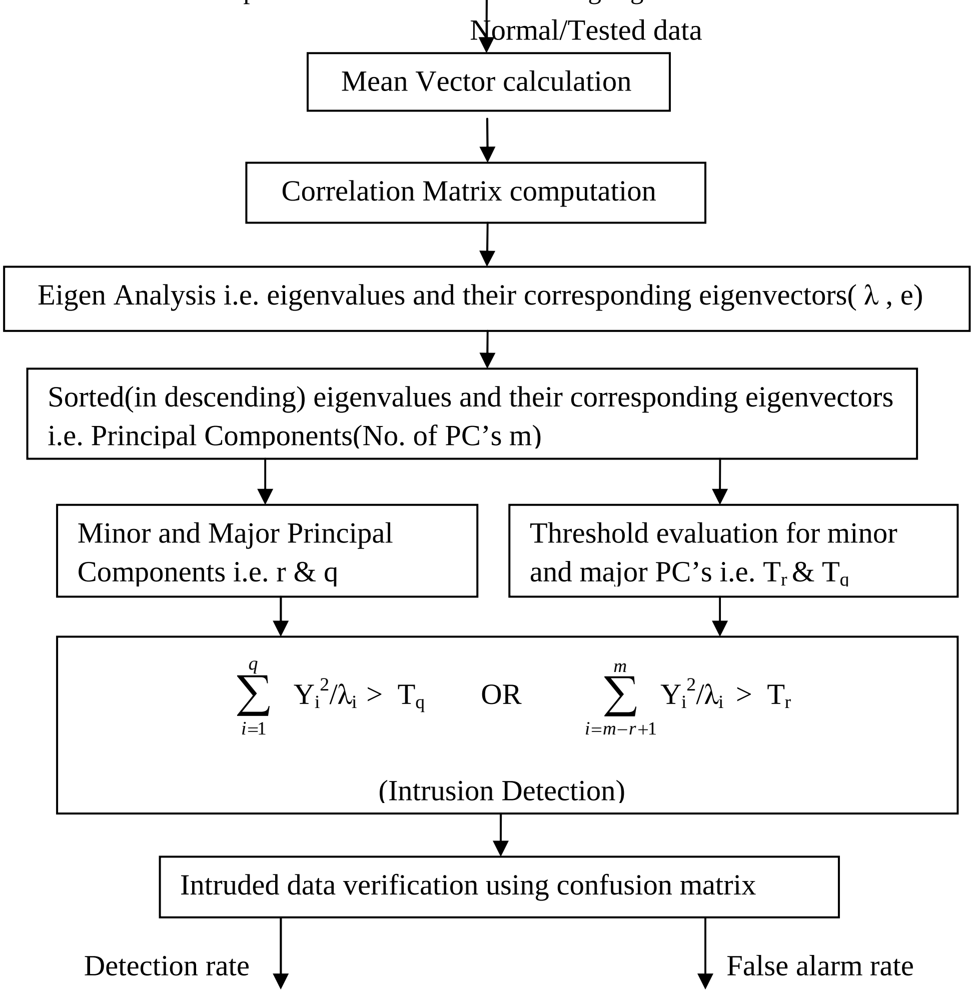

Fig 2. Various steps for verification of intrusion detection using PCA

4. Performance Log Analysis

We generate a log file of patterns with errors and without errors and then use PCA to analyze the results.

a. Performance Log

As PCA has wide area of applications, one area of application is HIDS. The analysis of the paper uses a host-based anomaly detection scheme to identify abnormal system behavior. Normal behavior of the system is created based on the processes running in the system. Then abnormal behavior is generated by creating problems in the system. Performance log are generated by taking some of the process attributes for the normal and abnormal behavior of the system. The performance of the personal computer can be measured by using the performance log. The hardware profile of the system that has been used for the experiment is as follows:

- Intel(R) Core 2 Duo CPU 1.60 GHz
- 1.99 GB RAM
- Microsoft Windows XP Professional Service Pack 2

b. Attributes used in performance log

Different attributes considered for Performance Log analysis are as follows:

- Committed byte in use (%): This is the ratio of memory committed bytes to memory commit limit. Here committed memory is physical memory in use for which space has been reserved in the paging file and should be written to the disk. The commit limit is determined by the size of the paging file. If the paging file is enlarged, the commit limit increases, and the ratio is reduced. This counter displays the current percentage value only (not an average).
- Available Mbytes: This is the amount of physical memory in Megabytes available to processes running in the computer. It is calculated by summing up the space of the Zeroed, Free, and Standby memory lists. Free memory is ready for use. Zeroed memory is pages of memory filled with zeros to prevent later processes from seeing data used by a previous process. Standby memory is memory removed from a process' working set (physical memory) on route to disk, but is still available to be recalled. This counter displays the last observed value only (not an average).
- Cache faults/sec: It is the rate at which faults occur when a page sought in the file system cache is not found and must be retrieved from elsewhere in memory (a soft fault) or from the disk (a hard fault). The file system cache is an area of physical memory that stores recently used pages of data for applications. Cache activity is a reliable indicator of most application I/O operations. This counter shows the number of faults, without regard for the number of pages faulted in each operation.
- Page faults/sec: It is the average number of pages faulted per second. It is measured in number of pages faulted per second because only one page is faulted in each fault operation; hence this is also equal to the number of page fault operations. This counter includes both hard faults (those that require disk access) and soft faults (where the faulted page is found elsewhere in physical memory.) Most processors can handle large numbers of soft faults without significant consequence. However, hard faults, which require disk access, can cause significant delays.
- Page writes/sec: It is the rate at which pages are written to disk to free up space in physical memory. Pages are written to disk only if they are changed while in physical memory, so they are likely to hold data, not code. This counter shows write operations, without regard to the number of pages written in each operation. This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.
- Page op/sec: It is the rate at which pages are written to disk to free up space in physical memory. Pages are written back to disk only if they are changed in physical memory, so they are likely to hold data, not code. A high rate of pages output might indicate a memory shortage. Windows writes more pages back to disk to free up space when physical memory is in short supply. This counter shows the number of pages, and can be compared to other counts of pages, without conversion.
- Pool non-paged allocs: is the number of calls to allocate space in the non-paged pool. The non-paged pool is an area of system memory for objects that cannot be written to disk, and must remain in the physical memory as long as they are allocated. It is measured in numbers of calls to allocate space, regardless of the amount of space allocated in each call. This counter displays the last observed value only; it is not an average.

- Pool paged allocs: is the number of calls to allocate space in the paged pool. The paged pool is an area of system memory (physical memory used by the operating system) for objects that can be written to disk when they are not being used. It is measured in numbers of calls to allocate space, regardless of the amount of space allocated in each call. This counter displays the last observed value only; it is not an average.
- System driver total byte: It is the size, in bytes, of the pageable virtual memory currently being used by device drivers. Pageable memory can be written to disk when it is not being used. It includes physical memory (Memory\\System Driver Resident Bytes) and code and data paged to disk. It is a component of Memory\\System Code Total Bytes. This counter displays the last observed value only; it is not an average.
- Write copies/sec: It is the rate at which page faults are caused by attempts to write that have been satisfied by coping of the page from elsewhere in the physical memory. This is an economical way of sharing data since pages are only copied when they are written to; otherwise, the page is shared. This counter shows the number of copies, without regard for the number of pages copied in each operation.

5. Experiment Methodology

To carry out the experiment, the performance logs are generated. The steps for generating the performance logs are as follows [16]:

- On the start menu, point to settings, point to Control Panel, double click Administrative Tools, and double click Computer Management.
- Explore performance Logs and Alerts, right click Counter Logs, and then click New Log Settings.
- Type a name for the counter log and then click OK.
- Click Add Counters.
- In the Performance object box, select a performance object that need to be monitored.
- Counters added for experiment.
- On the General tab under Sample data, every sampling interval of 15 seconds is configured.
- On the Log Files tab, log files properties are configured as Comma delimited files that can be viewed later in reporting tools such as Microsoft Excel.

After the performance log has been generated each day, the log is divided into 4 groups, and the average values for each column of the table are calculated. These values are used as our normal data set. In the meanwhile, for one day the system is left to work when the graphics driver, audio driver, and USB driver have been disabled. This generates the logs for system performance that have been considered as intruded data. We have taken the same number and same type of attributes in our experiment. For our experiment, we have taken the normal dataset and the testing dataset i.e. mixture dataset (normal and intrusion), which are given Tables 1 and 2, respectively. We have also shown how our proposed methodology detects and verifies the true intrusion in data flow diagram (DFD) (ref. Fig. 2).

Table 1 : Normal dataset with some selective attributes

Committed byte in use	Available Mbytes	Cache faults/sec	Page faults/sec	Page writes/sec	Page op/sec	Pool Non-paged Allocs	Pool Paged Allocs	System driver total byte	Write copies/sec
3.824418508	1724.572519	101.6124671	295.3630973	0.068574005	1.097184087	26682	42251.76336	7503152.855	3.335821308
3.641453572	1736.848485	57.15040864	256.0914499	0.245342241	3.925475853	23133.42424	32497.9697	7503872	4.445429107
5.111144298	1680.819718	79.47708971	334.4868715	0.143723968	2.299583495	37059.20563	52290.02817	7607771.944	1.918616972
5.638546946	1654.261628	36.1708794	162.9619512	0.036869153	0.589906451	32614.11047	45698.55233	7503872	2.133367554
4.578404615	1702.971429	51.11998798	280.3527608	0.158585618	2.537369884	27955.8	37290.37143	7503872	5.541007434
4.779657979	1696	49.93331807	128.9745175	0.168400988	2.694415815	33171.61811	48146.60236	7503872	1.321704214
6.23580037	1640.778065	94.47212975	308.6162728	0.132309061	2.11694497	54970.75613	71052.61742	7503872	3.689727265
5.61765	1648.30	46.0269	219.40	0.01555	0.24880	33495.5	43982.1	750368	7.98484

1849	0971	8896	03424	0526	8412	1262	0097	9.072	3091
4.111348119	1720.170732	132.1210622	525.6392525	0.065672177	1.05075484	26849.53659	37876.33537	7500175.61	7.747301195
9.084746823	1619.290476	68.32869399	458.027867	0.601359321	9.62174913	42900.96667	47518.92857	7470655.39	4.443562404
10.70494185	1580.088785	76.97290274	512.524161	1.165441987	18.64707179	49948.01402	62041.12617	7470663.776	10.85759362
10.84215109	1563.674455	102.997176	469.8837442	0.465373406	7.445974504	86950.07321	100841.3692	7470944.498	8.171717948
8.920870039	1650.936255	35.11830951	189.8569296	0.290242311	4.643876977	36718.68127	45366.95219	7470916.335	7.522112596
9.019445732	1629.359551	123.1521235	660.7778565	1.237642668	19.80228269	37276.39326	44230.85393	7470045.483	9.026770954
8.15211776	1669.168831	46.88692477	185.9397726	0.321431061	5.142896979	36697.48312	42494.61299	7470859.304	2.604372686
9.789734979	1564.333333	108.4193887	688.6777162	1.01201838	16.19229409	39289.68571	42273.74286	7469309.562	11.22624646
8.821370882	1635.538462	70.41303081	401.7980469	0.526899445	8.430391119	36366.85	41617.46154	7470379.323	7.551446472
9.226012337	1605.618421	82.65173496	478.153206	0.812458579	12.99933727	44724.5	48669.67763	7470484.211	5.782761903
13.13818528	1567.827759	60.97717436	335.7265847	0.801764537	12.8282326	49766.02174	58386.99164	7470946.462	6.014783825
10.23761708	1621.770221	107.500139	639.6973908	0.618029752	9.888476038	49423.07537	56112.78125	7577140.706	3.400912746

Table 2 : Testing dataset with some selective attributes

Committed byte in use	Available Mbytes	Cache faults/sec	Page faults/sec	Page writes/sec	Page op/sec	Pool Nonpaged Allocs	Pool Paged Allocs	System driver total byte	Write copies/sec
4.601387552	1700.022388	26.04043396	132.9967699	0.087610478	1.401767647	29212.34328	39788.48507	7533002.507	2.940184286
4.94660698	1690.179012	85.35408783	371.7505326	0.282849872	4.525597951	32166.55556	43517.9321	7503581.235	7.655012219
9.251920893	1632.754655	220.4284527	905.3287534	0.37617982	6.018877122	63023.91566	72219.42935	7497443.119	8.420612447
9.873066812	1584.642612	64.19829632	766.8079994	0.424827439	6.797239025	48472.48454	57402.49828	7470780.261	4.506649959
7.567823945	1692.206349	68.28457374	324.1501421	0.94456778	15.11308448	35413.93651	38624.19048	7470356.317	5.48903716
8.831054122	1646.637537	54.31647998	264.2389394	0.325504227	5.208067639	39974.58491	44790.88183	7471010.447	5.517903309
9.11075792	1635.158042	51.96754351	234.469843	0.429925858	6.87881372	49530.98042	54235.6993	7470840.481	3.021982506
8.038268296	1653.070313	71.37037658	399.8283644	0.942087728	15.07340364	35642.40625	39465.27344	7470368	5.881560678
9.811394241	1581.92963	87.2102098	427.90725	0.707488741	11.31981986	43191.15556	52279.08889	7470755.081	10.64523952
13.2414429	1585.845865	64.64190334	316.8836163	0.675352729	10.80564366	53580.36842	61298.03383	7470985.945	4.572603869

8.742937931	1644.183962	32.31156115	222.8207108	0.296411962	4.742591388	36625.95755	41421.07547	7470881.811	4.255707607
8.213684021	1634.324786	62.82807225	336.1020724	0.534603814	8.553661019	38311.66239	44574.31197	7470298.803	3.907110265
12.96123187	1484.299335	62.83706428	446.2232714	0.559429696	8.950875138	39367.74279	50919.15965	7470895.113	5.646072064
9.599338354	1608.684211	83.17642964	498.3845806	0.96693393	15.47094288	40983.42915	46584.31579	7470341.182	9.075043219
12.42302175	1585.587703	73.91782982	383.0054702	1.114807342	17.83691747	52191.46293	57598.28571	7470933.642	4.484665132
8.301144161	1657.94197	225.5267105	470.6244796	0.305879715	4.894075434	40272.80297	51679.34143	7470849.727	3.655737869
11.65648745	1522.172414	189.6987689	1189.393533	2.624174933	41.98679892	48923.3908	55990.70115	7468938.299	14.63283051
13.84320911	1499.341053	64.76622069	314.7177163	0.876485557	14.02376892	58342.58105	66071.59158	7470707.335	4.62282402
9.739266809	1581.827451	68.70486069	415.7679794	0.548092356	8.769477698	40609.1098	47562.47059	7470734.557	7.130239303
11.23444288	1540.498941	99.73592101	500.4337987	0.328273963	5.252383416	58790.27542	63676.78072	7509421.559	3.99335076
3.19629803	1740	1602.592741	4193.397458	1.865528638	29.84845821	22480	28030	7278592	123.2285306
7.098092873	1613	3708.629397	48843.19072	4.600624828	73.60999725	26335	31848	7278592	504.4256508
7.63337616	1600	2531.466836	29764.09948	6.082982197	97.32771516	29131	37788	7372800	287.2519371
7.648479754	1603	1006.769784	1956.87085	6.000217242	96.00347587	29534	40935	7372800	27.80100656
8.262957542	1590	461.1478738	2469.611203	5.200163966	83.20262346	31214	45784	7372800	200.4729878
3.324678577	1738	1762.764486	4682.115926	1.678077508	26.84924012	22296	28074	7278592	141.0144466
6.191877248	1643	3732.799556	49552.13253	4.666082767	74.65732428	25054	31251	7278592	492.6716819
5.875397339	1651	2049.189776	21404.14929	3.522962366	56.36739785	31232	31145	7376896	297.9699201
7.646194342	1600	2791.575869	31989.23505	2.533795515	40.54072824	34482	35446	7376896	211.7052831
7.893018861	1602	295.9433757	2465.416994	6.000203608	96.00325773	31029	44443	7372800	34.80118093
7.84651569	1601	524.727077	1533.889331	5.993836392	95.90138227	36112	36112	7471104	48.55007478
7.96336981	1600	1156.744234	2285.154868	6.000056544	96.0009047	36439	42058	7471104	44.60042031
5.936308543	1649	1982.852701	20715.51759	3.392345214	54.27752342	31382	31160	7376896	292.8186234
7.633276794	1598	2817.941408	32296.54016	2.530706249	40.49129998	34623	35585	7376896	196.9955022
7.843634084	1601	480.2046148	1513.814548	6.000057662	96.00092258	36190	37699	7471104	48.60046706
5.747215523	1653	1971.068567	20561.84666	3.393332869	54.29332591	31316	31052	7376896	284.2050934
7.656230282	1602	2656.199819	31600.57928	2.532888158	40.52621053	34369	35563	7376896	220.1613049

7.8977 88417	1599	866.33 54261	2194.6 76453	5.9940 1817	95.904 29071	36073	37570	747110 4	37.895 51487
12.938 51347	1429	233.86 71284	1094.2 68827	0	0	55611	63459	747110 4	67.800 13385
13.700 15193	1405	282.06 85421	2013.0 80051	0	0	58759	68319	747110 4	76.800 51064

Table 3: Detection rate and false alarm rate using different number of major and minor principal components

Used Dataset	q (Major)	r (Minor)	Detection Rate	False Alarm Rate
Testing Dataset (Table 2)	4	3	92.5%	5%
	4	4	97.5%	5%
	3	3	87.5%	7.5%
	5	2	97.5%	2.5%
	3	4	90%	7.5%

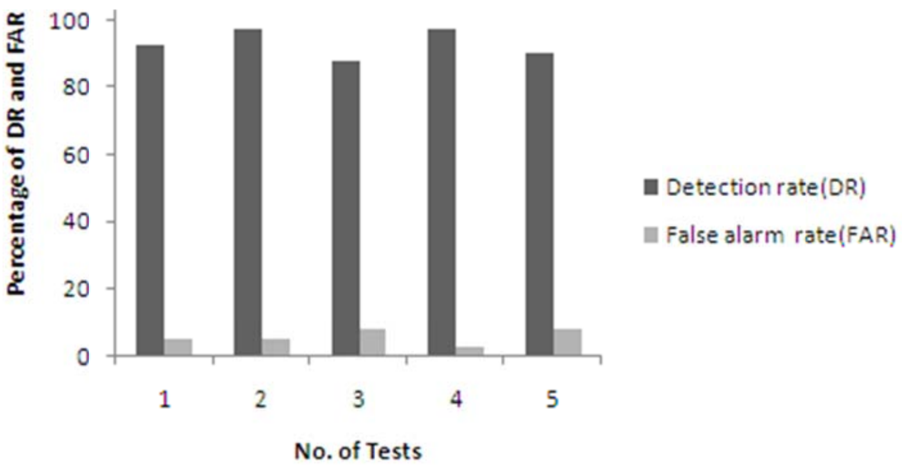

Fig 3. Detection rate and false alarm rate

6. Results and Discussion

In our experimental results, we have evaluated the detection rate and false alarm rate of the used tested data (ref. Table 2) for different number of major and minor principal components. We observe from our experimental results that the detection rate varies from 90% to 97.5% and the false alarm rate from 2.5% to 7.5% (ref. Fig. 1 & Table 3) by taking five different numbers of major and minor principal components for the used dataset. The above results of detection rate and false alarm rate have also been displayed in the bar chart (ref. Fig. 3). Furthermore, the detection rate and false alarm rate depend on the number of major principal components and minor principal components. It has been observed that the major principal components are more effective than the minor principal components because the major principal component specifies the sharp deviation of the value and the minor principal component specifies the slight deviation of the value. We have calculated the threshold value for the major principal component and minor principal component for the normal dataset and have been compared with all the data of the tested dataset. When the number of major principal components are greater than or equal to the number of minor principal components, the detection rate is high and false alarm rate is low (ref. Fig. 1 & Table 3).

7. Conclusions

In this paper, we have developed an intrusion detection system using principal component analysis (PCA) which has been implemented for HIDS on the basis of performance log. Our experimental results show that our proposed system provides detection rate in the range of 90% t0 97.5% and false alarm rate in the range of 2.5% to 7.5%, which are supposed to fairly good detection and false alarm rates.

8. Acknowledgement

The authors express their sincere thanks to Prof. S. Chand for his invaluable comments and suggestions.

References

[1] M.L . Shyu, S.C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," IEEE Foundations and New Directions of Data Mining Workshop, Nov. 2003, pp. 172–179.
[2] D.E. Denning, "An Intrusion Detection Model," IEEE Trans. on Software Engineering, SE-13, No. 2, 1987, pp. 222-232.
[3] D.M. Hawkins, "The Detection of Errors in Multivariate Data Using Principal Components," Journal of the American Statistical Association, Vol. 69, No. 346, 1974, pp. 340-344.
[4] N.Ye and Q. Chen, "An Anomaly Detection Technique Based on a Chi –Square Statistic for Detecting Intrusions into Information Systems," Quality and Reliability Eng. Int'l, Vol. 17, No. 2, 2001, pp. 105-112.
[5] J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory," ACM Transactions on Information and System Security, Vol. 3, No. 4, 2000, pp. 262-294.
[6] N.J. Puketza, K. Zhang, B. Mukherjee and R.A. Olsson, "Testing Intrusion Detection Systems: Design Methodologies and Results from an Early Prototype," Proc. 17th Natrional Computer Security Conference, Vol. 1, Oct. 1994, pp.1-10.
[7] F.N.M. Sabri, Md. Norwawi, and K. Seman, "Identifying False Alarm Rates for Intrusion Detection System with Data Mining," International Journal of Computer Science and Network Security, Vol.11, No.4, Apr. 2011.
[8] Z.K. Baker and V.K. Prasanna, "Efficient Hardware Data Mining with the Apriori Algorithm on FPGAs," In Proceedings of the Thirteenth Annual IEEE Sym. on Field Programmable Custom Computing Machines 2005.
[9] J.D. Jobson, "Applied Multivariate Data Analysis," Volume II: Categorical and Multivariate Methods. Springer- Verlag, N Y, 1992.
[10] I. T. Jolliffe, "Principal Component Analysis," Springer- Verlag, N Y, 2002.
[11] Jungsuk Song, Hiroki Takakura and Yasuo Okabe, "A proposal of new benchmark data to evaluate mining algorithms for intrusion detection," In 23rd Asia Pacific Advanced Networking Meeting, 2007.
[12] N. Athanasiades, R. Abler, J. Levine, H. Owen and G. Riley, "Intrusion detection testing and benchmarking methodologies," In IEEE International Information Assurance Workshop, 2003.
[13] H. Song and J.W. Lockwood, "Efficient packet classification for network intrusion detection using FPGA," Intl. Symp. On Field Programmable Gate Arrays, Feb. 2005.
[14] N. Ye, S.M. Emran, Q. Chen, and S. Vilbert, "Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection," IEEE Trans. on Computers, Vol-51, No.7, 2002.
[15] R.A. Johnson, D.W. Wichern, "Applied Multivariate Data Analysis," 3rd Edition, Prentice-Hall, Inc., Englewood Cliffs, N.J., U.S.A., 1992.
[16] H. Om and T.K. Sarkar, "Neural network based intrusion detection system for detecting changes in hardware profile," Journal of Discrete Mathematical Sciences & Cryptography, Vol. 12, No. 4, 2009, pp. 451-466.
[17] C.M. Chen, Y.L. Chen, H.C. Lin, "An efficient network intrusion detection," Computer Communications, Vol. 33, 2010, pp 477-484.
[18] P. Casas, J. Mazel, P. Owezarski, "Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge," Computer Communications, Vol. 35, 2012, pp. 772-783.

Authors:

Hari Om is presently working as Assistant Professor in the department of Computer Science & Engineering at Indian School of Mines, Dhanbad, India. He earned his Ph.D. in Computer Science from Jawaharlal Nehru University, New Delhi, India. He has published more than 50 research papers in International and National Journals including various Trans. of IEEE, Springer, Elsevier etc., International and National conferences of high repute. His research areas are Video-on-Demand, Cryptography, Network security, Data Mining, and Image Processing.

Tanmoy Hazra completed his M.Tech in Computer Applications, department of Computer Science and Engineering from Indian School of Mines, Dhanbad in May, 2012. He is presently working as Assistant Professor in the department of Computer Engineering at ISB & M School of Technology, Pune. His research interest is mainly on Intrusion Detection Systems.