AOCD: An Adaptive Outlier Based Coordinated Scan Detection Approach

Monowar H. Bhuyan[1], Dhruba Kr. Bhattacharyya[1], and Jugal K. Kalita[2]
(Corresponding author: Dhruba Kr Bhattacharyya)

Department of Computer Science and Engineering, Tezpur University, Tezpur-784028, Assam, India.[1]
Department of Computer Science, University of Colorado at Colorado Springs, CO 80918, USA.[2]
(Email: {mhb,dkb}@tezu.ernet.in, jkalita@uccs.edu)

(Received December 7, 2011; revised and accepted Jan. 30, 2012)

Abstract

Coordinated attacks are distributed in nature because they attempt to compromise a target machine from multiple sources. It is important for network defenders and administrators to detect these scans as possible preliminaries to more serious attacks. However, it is very difficult to detect malicious scans based on port specific behavior alone. In this paper, we present an Adaptive Outlier based approach for Coordinated scan Detection (AOCD) at an early stage with high accuracy. It is an outlier score based adaptive network anomaly detection approach that considers sets of normal instances during training. We use both normal and port scan instances for testing purpose. We achieve higher detection accuracy and low false positive rate on real-life and KDDcup99 probe datasets in comparison with existing techniques.

Keywords: Coordinated scans, outlier detection, port scan, principal component analysis

1 Introduction

During the last several decades, network defenders and researchers have developed approaches to detect malicious scans as well as coordinated port scans to keep enterprise networks secure. This is because cyber threats are becoming more sophisticated and more numerous, leading to more substantial damages to systems within short periods of time [7, 19]. Two types of correlations are used in a coordinated scan attack, Viz., *action correlation* and *task correlation* [5, 15]. Action correlation determines how actions performed by one user affect another user. For example, a particular action performed by one user may facilitate another user who performs the actual attack. In the other type of correlation, tasks divided among the multiple users are discovered. Here we focus mainly on task correlation.

Network administrators or defenders are interested in detecting coordinated scan attacks for a system in an enterprise network due to the following reasons.

- To detect coordinated scan attacks just like the detection of other attacks,
- To foil greater interest by the attacker who wants to remain undetected.
- To obviate the potential seriousness of the actual attacks.

A coordinated port scan is a part of a coordinated attack. Here, tasks are distributed among multiple hosts for their individual actions which may be synchronized. Such a port scan is an information gathering method used by an opponent to gain information about responding computers and open ports on a target network host. An opponent initiates the exploration of multiple hosts to scan a portion of the target network, with multiple sources focused on the portion of the target network which they want to compromise after getting relevant information from the target host. Intrusion Detection Systems (IDSs) are normally configured to recognize and report single source port scan activity. So, they cannot usually detect multiple source scans that collaborate with several hosts during scanning.

The detection of port scans, particularly stealthy or coordinated port scans, is important for early detection to enable action against potential intruders. The attackers or intruders are technically sophisticated enough to remain undetected while gathering information but the network defenders are usually out in the open. Single source scan detection is comparatively easy to detect because detection usually works better when a single source communicates with a single or multiple destinations. But the detection of a coordinated port scan is difficult due to the lack of relevant feature information at both packet and flow levels. Therefore, we are motivated to develop an adaptive outlier based detection mechanism for coordinated port scans known as AOCD. This paper makes the following key contributions.

- We formalize the problem of coordinated scan detection as a data mining problem and present an approach to transform network traffic data into a form where a classifier can be directly used. Specifically, we select random samples from the dataset and identify a set of features relevant for cluster detection for early detection of coordinated port scans.
- We exercise special care during labeling and use the labeled dataset for training as well as testing. The source is real network traffic data in our TU-IDS (Tezpur University Intrusion Detection System) testbed [3]. We demonstrate that our approach is capable of very early detection without significantly compromising the precision of the detection.
- We present extensive experiments on real-world network traffic data. The results show that the AOCD has substantially better performance than other state-of-the-art approaches in terms of accuracy and false positive rate.

The rest of the paper is organized as follows. Section 2 introduces the problem of coordinated port scan detection. Port scans and related concepts are introduced in Section 3. Section 4 provides related research and generic comparison of existing approaches. Our method for solving the problem is presented in Section 5. Section 6 describes empirical evaluation of AOCD. Finally, we present the concluding remarks and future work in Section 7.

2 Problem Statement

Coordinated or distributed port scans originate at multiple sources and focus on a single machine or multiple target machines. It is of special interest to large organizations with high level network situational awareness or military operations to detect coordinated port scans. The following are key problems.

- Coordinated scans compromise the victim machine earlier than single source port scans.
- Coordinated port scans are distributed in nature. So, intruders or attackers self-propagate the traffic and consume network bandwidth and resources quickly.

To overcome these problems, we develop an adaptive outlier based coordinated port scan detection approach. Let x be the captured, preprocessed current network traffic feature dataset, where $x_1, x_2, \cdots x_s$ are the training samples, randomly selected from dataset x that contain only normal instances. We apply the fuzzy c-means algorithm to cluster each sample individually into k number of clusters. Each cluster uses as a range based profile for detection. Let $x_1, x_2, \cdots x_t$ be the test instances to classify

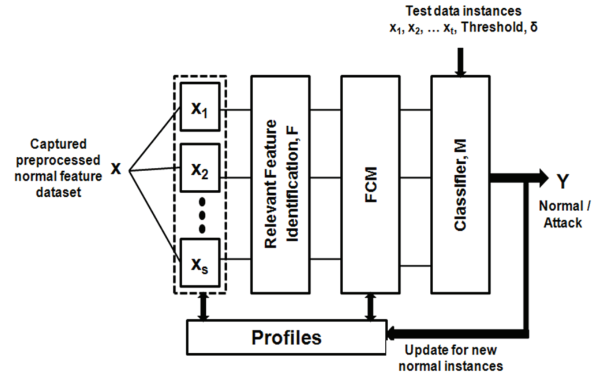

Figure 1: A framework for AOCD : FCM is the fuzzy c-means clustering algorithm for sample clustering and F is the PCA based feature selection technique for each sample as well as testing instances.

as attack or normal w.r.t. a threshold δ. The profile base is updated if any new distinct instances identified during testing. Thus, the AOCD adaptively updates its profile base for the new distinct instances. The framework for AOCD is given in Figure 1.

3 Port Scans and Related Concepts

In this section, we present preliminary discussions on port scans, outliers and network anomaly detection.

3.1 Port scans and types

There are several forms of reconnaissance activity, which often precedes an attack. When an adversary uses an effective mechanism to remotely probe a network, it is known as *port scanning*. System administrators and other network defenders also use this mechanism to detect port scans as precursors to serious attacks [4]. A port scan can be defined as sending packets to a particular IP or port to get a response from an active host in the network indicating services it offers. A port scan is useful to an attacker who wants to gain substantial information about the target host. Thus, it is of considerable interest to attackers to determine whether or not the defenders of a network scan ports regularly. Attackers hide their identity during port scanning whereas the network defender do not. Vivo et al. [9] describe a port scan as being composed of hostile Internet searches for open "doors" or "ports", through which intruders gain access to computers. Generally, there are several hosts available on a network and they run many services that commonly use TCP or UDP ports for communication with each other. A computer contains 65536 standardly defined ports [18]. They can be classified into three large ranges: (a) well known ports $(0 - 1023)$, (b) registered ports $(1024 - 49151)$ and (c) dynamic and/or private ports $(49152 - 65535)$. Normally, a port scan helps the attacker in finding those ports that are available to launch attacks, but it does not directly harm the system. Essentially, a port scan sends a packet with a message to the target host one at a time and listens for an answer. The response indicates whether the port is being used. This is a probe for weaknesses to launch future attacks. TCP and UDP ports are usually used for port scanning but TCP port scanning returns good feedback to the attacker because it is a connection-oriented protocol. UDP port scanning may not readily give relevant information to the attacker because it is a connectionless protocol. Also, a UDP port may be easily blocked by network defenders or network administrators. Following are the various types of port scans [4] which are used to probe weaknesses from a networked host.

1) *Stealth scan*: Auditing tools cannot detect this type of scanning because of their complicated design architecture. Such a scan sends TCP packets to the destination host with stealth flags. Some of the flags are SYN, FIN and NULL.

2) *SOCKS port probe*: It allows sharing of Internet connections on multiple hosts. Attackers scan these ports because a large percentage of users misconfigure SOCKS ports, potentially permitting arbitrarily chosen sources and destinations to communicate. It also allows the attackers to access other Internet hosts while hiding their true location.

3) *Bounce scan*: An FTP bounce scan attack takes advantage of a vulnerability of the FTP protocol itself. Email servers and HTTP Proxies are the common applications that allow bounce scans.

4) *TCP scan*: This type of scanning is used by a smart attacker because it never establishes a connection permanently. The attacker can launch an attack immediately if a remote port is accepting the connection request. Normally, this type of connection request cannot be logged by a server's logging system due to its smart connection attempt. Some TCP scans are TCP Connect(), reverse identification, Internet protocol (IP) header dump scan, SYN, FIN, ACK, XMAS, NULL and TCP fragment.

5) *UDP scan*: A UDP scan attempts to discover open ports related to the UDP protocol. However, UDP is a connectionless protocol and, thus, it is not often used by attackers since it can be easily blocked.

The list of port scan types discussed above along with firewall detection possibilities during the scanning process is given in Table 1. We can see from the table that most scans are not detected in firewall level.

The task of distributed information gathering is accomplished using either a many-to-one or a many-to-many model [14, 11]. The attacker utilizes multiple hosts to execute information-gathering techniques in two ways: rate-limited, and random or non-linear. In a rate-limited information-gathering technique, the number of packets sent by a host to scan is limited [10, 28]. This is based on the FreeBSD (BSD-Berkeley Software Distribution) implementation of UNIX where separate rate limits are maintained for open ports as well as closed ports. For example, TCP RST is rate limited. "ICMP port unreachable" is also rate limited. On the other hand, a random or non-linear gathering technique refers to randomization of the destination IP-port pairs among the sources, as well as randomization of the time delay for each probe packet. A coordinated attack has a more generic form of a distributed scan than the ones described by Staniford-Chen et al. [6]. It is defined as multi-step exploitation using parallel sessions with the objective of obscuring the unified nature of the attack, allowing the attackers to proceed more quickly. We present a general architecture (see Figure 2) for coordinated port scans, which are used during launching of various scans in the TUIDS testbed [3]. Each handler accepts the connection request from the attacker and sends it to the agents. The agents directly interact with the victim host.

Table 1: Port scan types and firewall level detection possibilities

Port scanning technique	Protocol	TCP flag	Target reply (open port)	Target reply (closed port)	Firewall level detection possibility
TCP *Connect()*	TCP	SYN	ACK	RST	Yes
Reverse Ident	TCP	No	No	No	No
SYN Scan	TCP	SYN	ACK	RST	Yes
IP Header Dump Scan	TCP	No	No	No	No
SYN|ACK Scan	TCP	SYN|ACK	RST	RST	Yes
FIN Scan	TCP	FIN	No	RST	No
ACK Scan	TCP	ACK	No	RST	No
NULL Scan	TCP	No	No	RST	No
XMAS Scan	TCP	All flags	No	RST	No
TCP Fragment	TCP	No	No	No	No
UDP Scan	UDP	No	No	Port Unreachable	No
FTP Bounce Scan	FTP	Arbitrary Flag Set	No	No	No
Ping Scan	ICMP	No	Echo Reply	No	Yes
List Scan	TCP	No	No	No	No
Protocol Scan	IP	No	-	-	No
TCP window scan	TCP	ACK	RST	RST	No

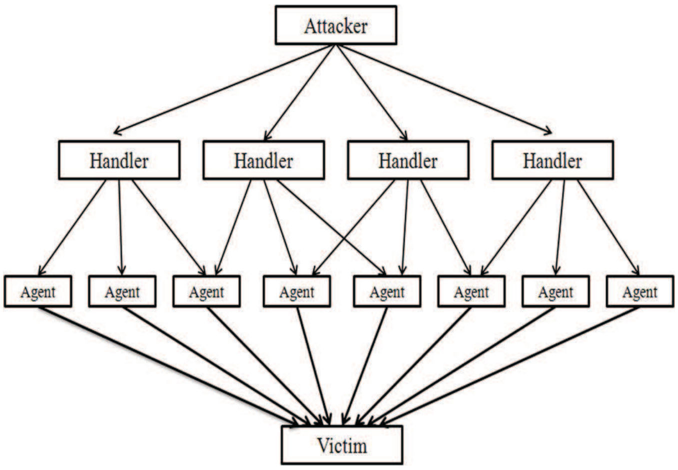

Figure 2: General architecture for generating coordinated port scans. It is used during launching of various scans in the TUIDS testbed for real-world coordinated scan dataset preparation.

4 Related Research

There are various methods for detecting coordinated attacks. We describe some of them in brief next.

Gates [12] describes a model of potential adversaries based on the information they wish to obtain, where each adversary is mapped to a particular scan footprint pattern. The adversary model forms the basis of an approach to detect forms of coordinated scans, employing an algorithm that is inspired by heuristics for the set covering problem. The model also provides a framework for comparing various types of adversaries that different coordinated scan detection approaches might identify. Both the detection and false positive rates gathered from the experiments are modeled using regression equations. Whyte [26] describes the design, implementation and evaluation of fully functional prototypes to detect internal and external scanning activity in an enterprise network. These techniques offer the possibility of identifying local scanning systems within an enterprise network after the observation of only a few scanning attempts with low false positive and negative rates. To detect external scanning activity directed at a network, it makes use of the concept of exposure maps that are identified by passively characterizing the connectivity behavior of internal hosts in a network as they respond to both legitimate connection attempts and scanning attempts. The exposure map technique enables: (a) active response options to be safely focused exclusively on those systems that directly threaten the network, (b) the ability to rapidly characterize and group hosts in a network into different exposure profiles based on the services they offer, and (c) the ability to perform a reconnaissance activity assessment that determines what specific information was returned to an adversary as a result of a directed scanning campaign. Finally, the author experiments with real-life scan activity as well as offline datasets. Singh and Chun [25] implement a TCP based port scanner in the OMNeT++ simulator. The authors describe two modules: simple and compound, and both modules are implemented using C++. They claim that their approach can detect TCP connect(), TCP SYN (half-open), TCP FIN (stealth), Xmas, NULL, ACK, Window and Reset (RST) scans at the router level.

Robertson et al. [22] define a distributed port scan as a set of port scans that originate from source IP addresses that are located close together. In other words, they assume that a scanner is likely to use several IP addresses on the same subnet. This implies that if a particular IP address scans a network, IP addresses near this IP address, rather than those far away, are more likely to have also scanned the network. Yegneswaran et al. [27] can detect coordinated port scans where a distributed port scan is defined as a set of scans from multiple sources (i.e., five or more) aimed at a particular port of destinations within an 1-hour window. On the basis of this definition, the authors find that a large proportion of daily scans are coordinated in nature, with coordinated scans being roughly as common as vertical and horizontal scans. The system looks to see if different sources start and stop scanning either at the same time, or in very similar temporal patterns. There is little locality in the IP space for these coordinated scanning sources. The authors do not discuss characteristics of the target hosts.

Several approaches have been used for visualizing network traffic to detect whether the flow of network packets is an attack or normal behavior. One such commonly found approach is proposed by Conti and Abdullah [8]. The approach attempts to detect distributed scans against a background of normal traffic based on visualization. Due to the lack of details, it is difficult to understand how a distributed scan would use this tool. Also, it is not clear how much traffic can be viewed at one time without obscuring features of interest.

Most distributed port scan detection approaches analyze packet level information. They can detect port scan attacks based on the IP addresses (source IP, destination IP), connection information, and port (source ports, destination ports) fields in the IP header. A general comparison of the distributed scan detection approaches discussed in this section is given in Table 2. We see in column 3 of the table that most of these approaches are non-real time.

Table 2: Comparing distributed port scan detection approaches.

Detection approach	Year of publication	Real-time (R)/ Non-real time(N)	Packet(P)/ Flow(F)
SysD [22]	2003	N	P
Pattern Based [27]	2003	R	P
Visual [8]	2004	N	P
Set Theoretic [12]	2006	N	P
Exposure Map [26]	2008	R	P
PCF [25]	2010	N	P

5 AOCD: The Proposed Approach

We describe the required concepts first and then the AOCD algorithm to detect coordinated port scans.

5.1 Outliers and Anomaly Detection

An outlier is an abnormal or infrequent event or object that varies significantly from the normal event or object in terms of a distance measure. A network administrator needs to define the abnormal event based on the normal statistics [30]. Outlier detection discovers exceptional events from small or large datasets [17]. Example of outliers in a two dimensional dataset are illustrated in Figure 3.

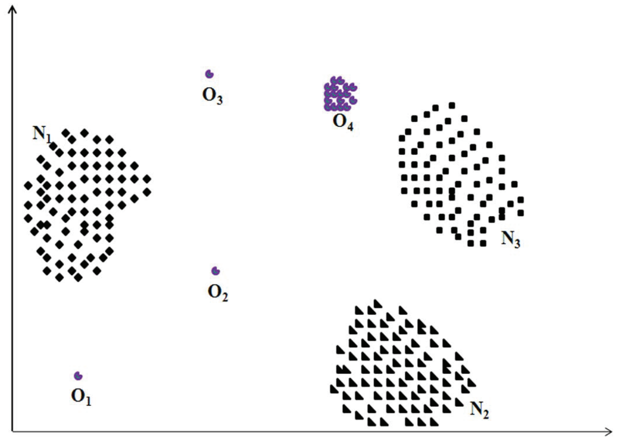

Figure 3: Outliers in two dimensional dataset: N_1, N_2, and N_3 are the three normal regions. Points that are sufficiently far away from the normal region (e.g., points O_1, O_2, O_3 and points in O_4 regions) are outliers.

5.1.1 Outlier Score and Its Importance

A large number of outlier detection techniques have been proposed in the literature but only some of them have been applied to anomaly detection [21, 29]. An outlier score is a summarized value based on distance, density or other statistical measures. A reference based outlier score is presented by Pei and Zaiane [20] for detecting outliers in large datasets. The authors estimate outlier score based on distance and a degree of nearest neighbor density. The authors define the outlier score as:

$$ROS(x) = 1 - \frac{D^p(x,k)}{\underset{1 \leq i \leq n}{max} D^p(x_i,k)} \quad (1)$$

where $D^p(x,k)$ is $\underset{1 \leq r \leq R}{min} D(x,k,p_r)$. $D^p(x,k)$ is the degree of neighborhood density of the candidate data point x with respect to the set of reference points p, n is the total number of data points, k is a reference based nearest neighbor, and R is the number of reference points. $D(x,k,p)$ is the relative degree of density for x in the one dimensional data space x^p and defined as:

$$D(x,k,p) = \frac{1}{\frac{1}{k}\sum_{j=1}^{k} |d(x_j,p) - d(x,p)|} \quad (2)$$

where $d(x,p)$ is the distance of x from the reference point p. p_r is the closest reference point to p. The candidate data points are ranked according to their relative degrees of density computed on a set of reference points. Outliers are those with high scores. This scheme can discover multiple outliers in larger datasets. However three main limitations of this scheme [20] are: (a) The score does not always vary with the change of candidate data points, (b) Summarizing the data points in terms of scores may not be effective for some attacks, and (c) It does not work in high dimensional datasets.

5.1.2 Anomaly Detection

Anomaly detection refers to the problem of finding nonconforming patterns in data. These patterns are often known as anomalies, outliers, exceptions, surprises, or peculiarities in different application domains. Anomalies and outliers are two terms used most commonly in the context of anomaly detection, sometimes interchangeably. The importance of anomaly detection is due to the fact that anomalies in data translate to significant, and often critical, actionable information in a wide variety of application domains. For example, an anomalous traffic pattern in a computer network could mean that a hacked computer is sending out sensitive data to an unauthorized destination.

5.2 Feature Selection Using PCA

Principal Component Analysis (PCA) is often used to reduce the number of dimensions in data for cost-sensitive analysis [24]. Let $x_1, x_2, x_3, \cdots x_p$ and $y_1, y_2, y_3, \cdots y_p$ be two p dimensional observations. PCA is concerned with explaining the variance-covariance structure of a set of variables through a few new variables which are functions of the original variables. Principal components are particular linear combinations of the p random variables $x_1, x_2, x_3, \cdots x_p$ with three important properties: (i) The principal components are uncorrelated, (ii) The first principal component has the highest variance, the second principal component has the second highest variance, and so on, and (iii) The total variation in all the principal components combined is equal to the total variation in the original variables $x_1, x_2, x_3, \cdots x_p$. They are easily obtained from an eigen analysis of the covariance matrix or the correlation matrix of $x_1, x_2, x_3, \cdots x_p$.

Let dataset x be denoted as $\{x_1, x_2, x_3 \cdots x_n\}$ with n objects, where each x_i can be a numeric or categorical attribute represented by a d-dimensional vector, i.e., $x = \{x_{i,1}, x_{i,2}, x_{i,3} \cdots x_{i,d}\}$.

Let A be a $n \times p$ covariance matrix of n observations in p dimensional space, i.e., p random variables

$$ROS'(x) = \frac{\max_{1\leq i\leq k'} S_i}{k'} \times \left(\frac{\left(1 - \min_{1\leq i\leq k'} dist(x_{i,j}, R_{i,j})\right) \times \left(\sum_{i=1}^{k'} \min_{1\leq i\leq k'} dist(x_{i,j}, R_{i,j})\right)}{\sum_{i=1}^{k'} \max_{1\leq i\leq k'} dist(x_{i,j}, R_{i,j})} \right) \quad (3)$$

$x_1, x_2, x_3, \cdots x_p$. If $(\lambda_1, e_1), (\lambda_2, e_2), (\lambda_3, e_3), \cdots, (\lambda_p, e_p)$ are the p eigenvalue-eigenvector pairs of A, $\lambda_1 \geq \lambda_2 \geq \lambda_3, \cdots, \lambda_p \geq 0$, the i^{th} sample principal component of an observation vector, $x = (x_1, x_2, x_3, \cdots x_p)'$ is

$$y_i = e_i' z = [e_{i1}' z_1, e_{i2}' z_2, e_{i3}' z_3, \cdots, e_{ip}' z_p] \quad (4)$$

where '$'$' represents the transpose of the matrix, $e_i = (e_{i1}, e_{i2}, e_{i3}, \cdots, e_{ip})$ is the i^{th} eigen-vector and $z = (z_1, z_2, z_3, \cdots, z_p)'$ is the vector of standardized observations defined as $z_k = \frac{x_k - \overline{x_k}}{\sqrt{s_k}}$ where $\overline{x_k}$ and s_k are the sample mean and sample variance of the variable x_k. The features are selected based on the eigenvectors with highest eigenvalues from p dimensional space. Therefore, our approach works on reduced feature spaces given by PCAF, which is based on PCA.

Algorithm 1 FCM $(x, k, m, l, \varepsilon)$

Input: x_i is the i^{th} data instance and u_{ij} represents the whole data matrix, k is the number of clusters, m is a real number greater than 1, l is the number of iterations, ε is the termination criteria between 0 and 1.
Output: Generate cluster, $C_1, C_2, C_3, \cdots C_k$.
Initialize $U = [u_{ij}]$, $U^{(0)}$.
Compute the center vectors $k^{(l)} = [k_j]$ with $U^{(l)}$: $k_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}$
Update $U^{(l)}, U^{(l+1)}$: $u_{ij} = \frac{1}{\sum_{l=1}^{k} \left(\frac{\|x_i - k_j\|}{\|x_i - k_l\|}\right)^{\frac{2}{m-1}}}$ w.r.t. PCAF module.
if $\|U^{l+1} - U^l\| < \varepsilon$ **then**
  Stop.
else
  Return to Step 2.
end if

5.3 The Proposed Approach

AOCD aims to detect anomalous patterns, i.e., coordinated port scans using an adaptive outlier based approach with reference to profiles. Initially, we select random samples, $x_1, x_2, \cdots x_s$ using a linear congruential generator from the dataset x for training purpose. It is a maximum length pseudo random sequence generator [23] and can be defined as $x_n = (ax_{n-1} + b) \bmod m$, where x_n is the n^{th} number of sequence, x_{n-1} is the previous number of the sequence. a, b, and m are secrets, a is the multiplier, b is the increment, and m is the modulus.

We cluster each sample into k classes by using the Fuzzy C-means [2] clustering technique. We receive the following clusters from all samples: $C_{11}, C_{12}, C_{13}, \cdots C_{1k}$, $C_{21}, C_{22}, C_{23}, \cdots C_{2k}$, $\cdots$ $C_{s1}, C_{s2}, C_{s3}, \cdots C_{sk}$. It estimates the range based profiles for each cluster and matches each profile with others to remove redundancy. These profiles are used as reference during score computation. Finally, it computes score for each candidate object and reports as normal or outliers (i.e., attack) w.r.t. a threshold, δ. We present the Fuzzy C-means clustering technique for cluster formation in Algorithm 1.

Let S_i be the number of classes to which each of k' nearest neighbor data objects belongs, where k' is fixed for a particular dataset. Let $x_{i,j}$ be a data object in x and $dist(x_{i,j}, R_{i,j})$ be the distance from the reference point $R_{i,j}$ to the data object $x_{i,j}$, where $dist$ is a proximity measure and x represents the whole dataset. The proposed approach is independent of the use of any particular proximity measure. However, in our experiments, we use Euclidean distance in computing proximity.

The formula for the outlier score ROS' is given in Equation (3). In this formula, $\frac{\max_{1\leq i\leq k'} S_i}{k'}$ is the maximum probability that a data object belongs to a particular class; the remaining part is the summarized value of similarity measure within k' nearest neighbors. The candidate data objects are ranked based on the score. Objects with scores higher than a user defined threshold δ are considered anomalous or outliers. δ is determined by a heuristic method. To test effectiveness, we consider seven different cases (illustrated in Figure 4 [3]) and the proposed algorithm is capable of identifying all these seven cases.

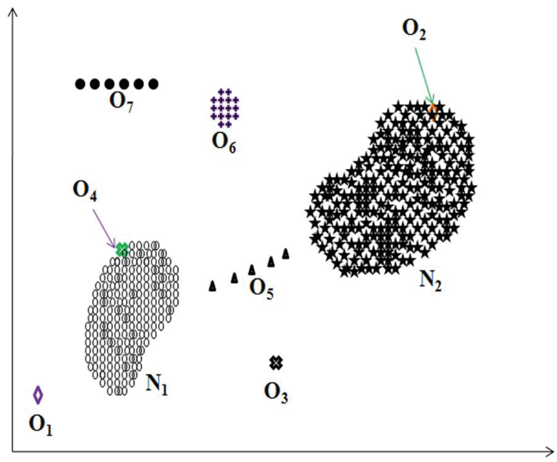

Figure 4: Illustration of seven different cases: N_1 and N_2 are two normal clusters, O_1 is the distinct outlier, O_2, the distinct inlier, O_3, the equidistance outlier, O_4, the border inlier, O_5, a chain of outliers, O_6 is another set of outlier objects with higher compactness among the objects and O_7 is an outlier case of "*stay together*".

A heuristic identification of k' values for our own flow level dataset vs. accuracy is given in Figure 5. We now present a few definitions before we present our algorithm.

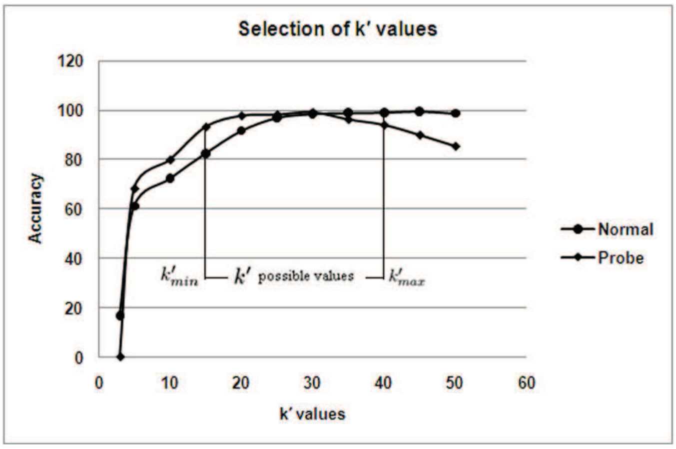

Figure 5: k' values vs accuracy in our own flow level dataset.

Definition 1. ***Pattern Similarity*****:** Two data objects x_1 and x_2 are defined as similar iff (a) $dist(x_1, x_2) < \delta$ and (b) $dist(x_1, x_2) = 0$, if $x_1 = x_2$.

Definition 2. ***Profile*****:** A profile of a cluster C_i is a range value, $\mu(x_{\mu,1}, x_{\mu,2} \cdots x_{\mu,d})$ of dataset x, where each $x_{\mu,j}$ is the range of the j^{th} column of the respective cluster C_i.

Definition 3. ***Outliers*****:** Two data objects, O_i and O_j are defined as outliers w.r.t a cluster C_i iff (a) $ROS'(O_i, \mu_i) \geq \delta$ where μ_i is the profile of C_i and, (b) for any other data object O_j in C_i, $dist(O_i, O_j) > \delta$.

The symbols used to define the score based network anomaly detection algorithm are given in Table 3.

Table 3: Symbols used

Term	Definition
x	dataset
n	number of data objects in x
C	set of clusters
R_i	i^{th} reference point
S_i	occurrences belonging to a class within k^{th} nearest neighbors
$dist$	similarity based on *Euclidean* distance
δ	threshold value for the outlier score
x_c	candidate data objects
μ	mean based profile value w.r.t a cluster
k'	number of nearest neighbors
m	number of large clusters
k	number of clusters
F	selected feature set
α	random subset selection using maximum length pseudo random sequence generator

Clustering is initiated based on a random selection of k centroids. We assign each $x_{i,j}$ object to a particular cluster based on the cluster membership value w.r.t. a proximity measure, i.e., $dist(x, y)$. We use Euclidean distance as proximity measure. $dist$ is defined as

$$dist(x,y) = \begin{cases} 0 & \text{if } x = y \\ \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} & \text{otherwise.} \end{cases}$$

5.4 AOCD: The Algorithm

The AOCD algorithm is based on the NADO [3] approach. AOCD differs from NADO technique in the following key points.

- We use principal component analysis (PCA) [24] based feature reduction technique to identify the relevant feature set. These feature sets are used during cluster formation (Algorithm 1).
- AOCD uses a variant of the Fuzzy C-means clustering algorithm to cluster formation.
- We test AOCD using real life coordinated datasets.
- AOCD adaptively updates the profiles for new test instances.

AOCD works as follows. $C_{11}, C_{12}, C_{13}, \cdots C_{1k}$, $C_{21}, C_{22}, C_{23}, \cdots C_{2k}, \cdots$ $C_{s1}, C_{s2}, C_{s3}, \cdots C_{sk}$ are the set of clusters with cardinality sk. It generates the profiles, $\mu_{s1}, \mu_{s2}, \mu_{s3}, \cdots \mu_{sk}$ for the clusters $C_{s1}, C_{s2}, C_{s3}, \cdots C_{sk}$ obtained from the dataset x. Then it detects coordinated scans based on the outlier score ROS' from the testing datasets. The major steps of AOCD are given in Algorithm 2.

Algorithm 2 AOCD (x, δ)

Input: x is the dataset, δ is the threshold
Output: $O_{i,j}$'s are the anomalous objects
Select random sample, $x_1, x_2, \cdots, x_s$ from the dataset x using α.
Find clusters $C_{s1}, C_{s2}, C_{s3}, \cdots C_{sk}$ for for each sample x_s based on a variant of Fuzzy C-means clustering (Algorithm 1) technique w.r.t. relevant feature set F.
Compute range based profile μ_{sk} for each of those sk number of clusters w.r.t. F.
Calculate outlier score ROS' for each candidate data object, $X_{c_{i,j}}$ w.r.t. F and μ_{sk}.
Rank the candidate data objects according to their score values.
Sort the data objects based on score values and report the anomalies or outliers, $O_{i,j}$'s w.r.t. the threshold δ.
if new test instances found **then**
    Update range based profiles, μ_{sk}.
    Return to Step 4.
end if

6 Experimental Results

The main goal of the experiments is to apply AOCD to coordinated scan detection as well as to evaluate its capability in detecting outliers or anomalies or scans and compare it to the current best performing algorithms. To achieve this goal, we have implemented our algorithm and tested it with various real world datasets and datasets prepared by us on our TUIDS testbed in both packet and flow level. It has been used during attack generation in our TUIDS testbed for labeled coordinated dataset preparation. The network laboratory layout where we capture network traffic for coordinated port scans data is shown in Figure 6. The network has 32 subnets including a wireless network, 4 routers, 3 wireless controllers, 8 L3 switches, 15 L2 switches and 300 hosts. The DHCP server is set up inside the main network for wireless network. During attack generation, we use 3 subnets as handlers, 15 subnets as agents and one wireless subnet is used to launch the attack.

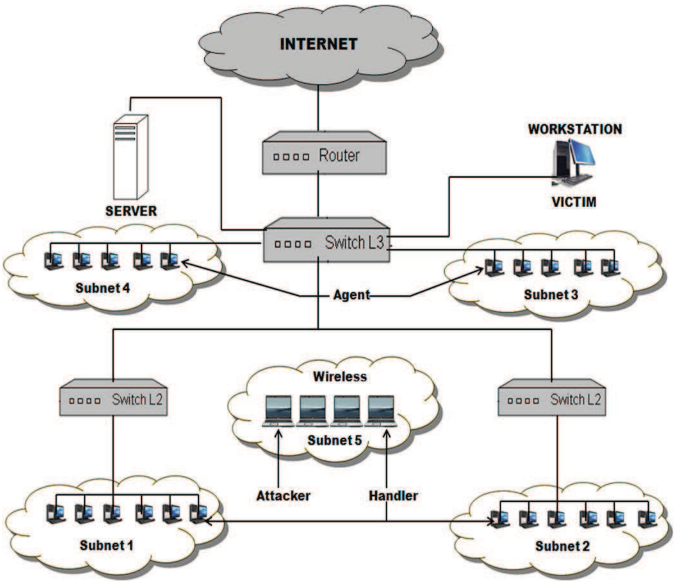

Figure 6: Coordinated port scan TUIDS testbed setup with 15 agents, 3 handler and a /32 subnet.

6.1 Environment Used

AOCD is implemented on an HP *xw*6600 workstation, Intel Xeon Processor (3.00 Ghz) with 4GB RAM. *Java* 1.6.0 version is used for the implementation in Ubuntu 10.10 (Linux) platform. Java is used to facilitate the visualization and reusability of code for further experimentation.

6.2 Datasets Used

To evaluate the performance of AOCD, we use several real life datasets for experimentation. We use three datasets: our own datasets that are *packet* and *flow* based and *KDDcup99 probe* [16] dataset. The characteristics of our own packet and flow level coordinated port scan datasets are presented in Table 4. The characteristics of the KDDcup99 probe datasets used in this experiments are given in Table 5.

Table 4: Distribution of Normal and Attack connection instances in TUIDS real-life packet and flow level intrusion datasets.

Connection type	Dataset type: Training dataset		Testing dataset	
Packet level				
Normal	71785	100%	47895	75.78%
Probe			15307	24.22%
Total	71785	-	63202	-
Flow level				
Normal	23120	100%	16770	48.56%
Probe			17762	51.44%
Total	23120	-	34532	-

Table 5: Distribution of Normal and Attack connection instances in KDDcup99 probe datasets.

Connection type	Dataset type: Training dataset (10% corrected)		Testing dataset (Corrected)	
Normal.	97278	100%	60593	87.98%
Probe.			8273	12.01%
Total.	97278	-	68866	-

6.3 Results and Discussion

We use our feature datasets for experimentation in both packet and flow levels. The datasets are generated in our network security laboratory as discussed earlier. At packet level, we extract basic features, content based features, time based features and window based features (see in Table 6). At flow level, we extract basic features, time based features, and window based features (see in Table 7). We convert all categorical attributes into numeric form and then compute the $log_z(a_{i,j})$ of larger values to normalize data objects, where z depends on the attribute values and $a_{i,j}$ represents the larger attribute values.

We have generated sixteen types of attacks (see Table 1) for coordinated scans. However, in this experiment we consider only four types of scans (i.e., TCP SYN, window, XMAS, and NULL) in coordinated mode during testing in both packet and flow level datasets. The PCAF module selects the relevance feature set in both packet and flow level datasets (see in Table 8). PCAF reduces the dataset in dimension based on their feature relevance. Hence, a continuous feature IDs is seen in Table 8. This reduced dataset is used by cluster formation

Table 6: List of packet level features in our own TUIDS intrusion dataset.

Label/feature name	Type*	Description
Basic features		
1. Duration	C	Length (number of seconds) of the connection
2. Protocol-type	D	Type of protocol, e.g., tcp, udp, etc.
3. Src-ip	C	Source host IP address
4. Dest-ip	C	Destination IP address
5. Src-port	C	Source host port number
6. Dest-port	C	Destination host port number
7. Service	D	Network service on the destination e.g., http, telnet etc.
8. num-bytes-src-dst	C	The number of data bytes flowing from source to destination
9. num-bytes-dst-src	C	The number of data bytes flowing from destination to source
10. Fr-no.	C	Frame number
11. Fr-len.	C	Frame length
12. Cap-len.	C	Captured frame length
13. Head-len.	C	Header length of the packet
14. Frag-off	D	Fragment offset '1' for the second packet overwrite everything '0' otherwise
15. Ttl	C	Time to live '0' discards the packet
16. Seq-no.	C	Sequence number of the packet
17. CWR	D	Congestion window record
18. ECN	D	Explicit congestion notification
19. URG	D	Urgent TCP flag
20. ACK	D	Acknowledgement flag value
21. PSH	D	Push TCP flag
22. RST	D	Reset TCP flag
23. SYN	D	Syn TCP flag
24. FIN	D	Fin TCP flag
25. Land	D	1 If connection is from/to the same host/port; 0 otherwise
Content-based features		
26. Mss-src-dest-requested	C	Maximum segment size from source to destination requested
27. Mss-dest-src-requested	C	Maximum segment size from destination to source requested
28. Ttt-len-src-dst	C	Time to live length from source to destination
29. Ttt-len-dst-src	C	Time to live length from destination to source
30. Conn-status	C	Status of the connection (e.g., '1' for complete, '0' for reset)
Time-based features		
31. count-fr-dest	C	Number of frames received by unique destination in the last T seconds from the same source
32. count-fr-src	C	Number of frames received by unique source in the last T seconds to the same destination
33. count-serv-src	C	Number of frames from the source to the same destination port in the last T seconds
34. count-serv-dest	C	Number of frames from destination to the same source port in the last T seconds
35. num-pushed-src-dst	C	Number of pushed packets flowing from source to destination
36. num-pushed-dst-src	C	Number of pushed packets flowing from destination to source
37. num-SYN-FIN-src-dst	C	Number of SYN/FIN packets flowing from source to destination
38. num-SYN-FIN-dst-src	C	Number of SYN/FIN packets flowing from destination to source
39. num-FIN-src-dst	C	Number of FIN packets flowing from source to destination
40. num-FIN-dst-src	C	Number of FIN packets flowing from destination to source
Connection-based features		
41. count-dest-conn	C	Number of frames to unique destination in the last N packets from the same source
42. count-src-conn	C	Number of frames from unique source in the last N packets to the same destination
43. count-serv-srcconn	C	Number of frames from the source to the same destination port in the last N packets
44. count-serv-destconn	C	Number of frames from the destination to the same source port in the last N packets
45. num-packets-src-dst	C	Number of packets flowing from source to destination
46. num-packets-dst-src	C	Number of packets flowing from destination to source
47. num-acks-src-dst	C	Number of acknowledgement packets flowing from source to destination
48. num-acks-dst-src	C	Number of acknowledgement packets flowing from destination to source
49. num-retransmit-src-dst	C	Number of retransmitted packets flowing from source to destination
50. num-retransmit-dst-src	C	Number of retransmitted packets flowing from destination to source

Note: *(C-Continuous, D-Discrete)

and coordinated scan detection module. AOCD is evaluated in terms of accuracy and false positive rate (FPR). The evaluation metrics are described below.

- **True Positive (TP)** represents the number of suspicious activities correctly detected as true attacks.
- **False Positive (FP)** represents the number legitimate activities misdetected as attacks.
- **False Negative (FN)** denotes the number of suspicious activities not detected by the model.
- **True Negative (TN)** represents the number of normal activities correctly detected as legitimate activities.

Finally, we summarize the measures in terms of detection accuracy and false positive rate as follows.

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- $False\ Positive\ Rate\ (FPR) = \frac{FP}{FP+TN}$

Details of performance of AOCD for the real life TUIDS packet and flow level coordinated scan datasets are given in Table 9 and shown in Figure 7. Our results are better than the results in Singh and Chun [25]. They obtained greater than 90% accuracy using their method. The performance of the AOCD algorithm is quite satisfactory in case of probe class for both packet and flow dataset. We seen in table 9 that the average accuracy for SYN, window, XMAS and NULL classes in packet level is 99.02% and in flow level it is 98.50%. Also, we test AOCD on four coordinated scan datasets but Singh and Chun [25] method was tested only on TCP SYN scan.

Table 7: List of flow level features in our own TUIDS intrusion dataset.

Label/feature name	Type*	Description
Basic features		
1. Duration	C	Length (number of seconds) of the flow
2. Protocol-type	D	Type of protocol e.g., TCP, UDP, ICMP
3. Src-ip	C	Source host IP address
4. Dest-ip	C	Destination IP address
5. Src-port	C	Source host port number
6. Dest-port	C	Destination host port number
7. ToS	D	Type of service
8. URG	D	TCP urgent flag
9. ACK	D	TCP acknowledgement flag
10. PSH	D	TCP push flag
11. RST	D	TCP reset flag
12. SYN	D	TCP SYN flag
13. FIN	D	TCP FIN flag
14. Src-bytes	C	Number of data byte transfer from source to destination
15. Dest-bytes	C	Number of data byte transfer from destination to source
16. Land	D	1 If connection is from/to the same host/port; 0 otherwise
Time-based features		
17. count-dest	C	Number of flows to unique destination IP in the last T seconds from the same source
18. count-src	C	Number of flows from unique source IP in the last T seconds to the same destination
19. count-serv-src	C	Number of flows from the source to the same destination port in the last T seconds
20. count-serv-dest	C	Number of flows from the destination to the same source port in the last T seconds
Connection-based features		
21. count-dest-conn	C	Number of flows to unique destination IP in the last N flows from the same source
22. count-src-conn	C	Number of flows from unique source IP in the last N flows to the same destination
24. count-serv-srcconn	C	Number of flows from the source IP to the same destination port in the last N flows
25. count-serv-destconn	C	Number of flows to the destination IP to the same source port in the last N flows

Note: *(C-Continuous, D-Discrete)

Table 8: TUIDS packet and flow level intrusion datasets - selected feature set.

Method	#Features	Selected features
Packet level		
PCAF	19	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
Flow level		
PCAF	24	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24

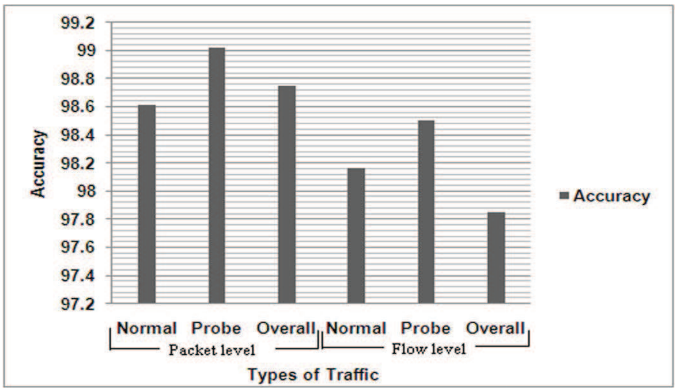

Figure 7: The performance of AOCD on the packet and flow level TUIDS intrusion datasets. The performance of flow level dataset is a bit less than packet level dataset due to non availability of packet specific information. But it is faster.

Table 9: The performance of AOCD on the packet and flow level TUIDS intrusion datasets.

Type of traffic	Correctly detected	Miss detected	Accuracy (%)
Packet level			
Normal.	47257	638	98.61
Probe.	15158	149	99.02
Overall.	62415	787	98.75
Flow level			
Normal.	16358	412	98.16
Probe.	14496	266	98.50
Overall.	30854	678	97.85

In another set of experiments, we use the KDDcup99 probe [16] dataset. Like the TUIDS datasets, we convert all categorical attributes to numeric and normalize them. We use KDDcup99 10% corrected normal dataset for training purpose and KDDcup99 corrected and 10% corrected probe datasets for testing purpose during performance analysis. The testing dataset contains six attacks, i.e., portsweep, ipsweep, satan, nmap, mscan and saint. The feature set selected by PCAF module for normal and probe classes is given in Table 10. Here, we see a continuous sequence of feature IDs in Table 10 because of PCAF reduces the feature dimension. Performance details of this datasets are given in Table 11. Figure 8 reports the comparison of AOCD using the intrusion dataset with other similar algorithms, where the false positive rate is multiplied by 100 to highlight the efficiency of our approach in the graph. In our experiment, better results are obtained in KDDcup99 probe dataset with δ values in the range of (0.8 - 1.35) over for normal records and (0.4 - 1.15) for attack records.

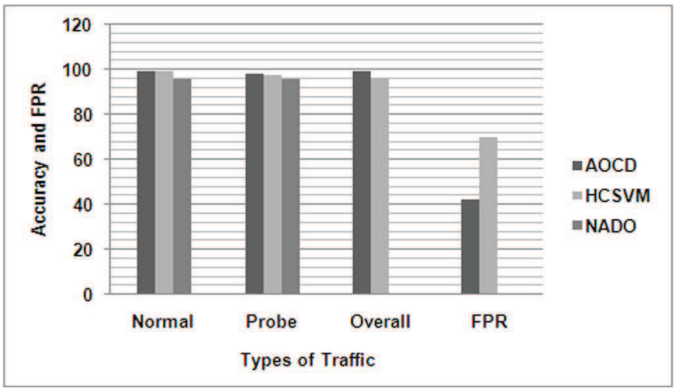

Figure 8: Comparison of the AOCD with other techniques over KDDcup99 probe dataset. The AOCD performs better than other two recent competing algorithms, HCSVM [13] and NADO [3] in terms of accuracy and false positive rate.

Table 10: KDDcup99 dataset - selected features set

Method	#Features	Selected features
Normal class		
PCAF	18	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18
FFSA [1]	6	5, 3, 1, 4, 34, 6
MMIFS [1]	6	5, 23, 3, 6, 35, 1
LCFS [1]	15	12, 34, 33, 3, 23, 27, 29, 40, 39, 28, 2, 41, 26, 35, 10
Probe class		
PCAF	25	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
FFSA [1]	24	40, 5, 41, 11, 2, 22, 9, 27, 37, 28, 14, 19, 31, 18, 1, 17, 16, 13, 25, 39, 26, 6, 30, 32

Table 11: The performance of AOCD over the KDDcup99 probe dataset

Type of traffic	Correctly detected	Miss detected	Accuracy (%)
Normal.	60189	404	99.38
Probe.	8114	159	98.08
Overall.	68303	563	99.18

7 Concluding Remarks

In this paper, we present an adaptive outlier based approach for coordinated port scan detection [3]. Unlike previous approaches which have been based on clustering and manual analysis, AOCD uses random sample selection using a linear congruential generator for distinct profile generation. It uses an outlier based approach for scoring each feature traffic data object and reporting as malicious or anomaly or outlier. AOCD is capable of detecting coordinated scans that have a stealthy and horizontal or strobe footprint across a contiguous network address space. We have tested this algorithm using different real-life datasets (i.e., TUIDS datasets and KDDcup99 probe datasets). Coordinated scans are performed in an isolated environment, combining the network traffic traces with those collected from live networks. We extract various features from network packet as well as flow traffic data by developing our own modules for feature extraction. This approach achieves high detection accuracy and low false positive rate on various real life datasets in comparison to existing coordinated scan detection approaches.

We are in the process of generating coordinated port scan feature datasets for the rest of the attacks.

Acknowledgments

This work is an outcome of a research project "DDoS attack detection" funded by DIT, Ministry of IT and supported by Council of Scientific & Industrial Research (CSIR), Government of India. It is also partially supported by NSF grants CNS-0958516 and CNS-0851783. The authors are thankful to the funding agencies and also gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] F Amiri, M M R Yousefi, C Lucas, A Shakery, and N Yazdani, "Mutual information-based feature selection for intrusion detection systems," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1184–1199, 2011.

[2] James C Bezdek, Robert Ehrlich, and William Full, "FCM: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191 – 203, 1984.

[3] M H Bhuyan, D K Bhattacharyya, and J K Kalita, "NADO: Network anomaly detection using outlier approach," in *Proceedings of the International Conference on Communication, Computing and Security*, pp. 531–536. ACM, February 2011.

[4] M H Bhuyan, D K Bhattacharyya, and J K Kalita, "Surveying port scans and their detection methodologies," *The Computer Journal*, vol. 54, no. 10, pp. 1565–1581, 2011.

[5] Sviatoslav Braynov and Murtuza Jadliwala, "Detecting Malicious Groups of Agents," in *Proceedings of the 1st IEEE Symposium on Multi-Agent Security and Survivability*, pp. 90–99. IEEE CS, 2004.

[6] S Staniford Chen, S Cheung, R Crawford, M Dilger, J Frank, J Hoagland, K Levitt, C Wee, R Yip, and D Zerkle, "Grids: a graph based intrusion detection system for large networks," in *Proceedings of the 19th National Information Systems Security Conference*, pp. 361–370, USA, October 1996.

[7] Kim Kwang Raymond Choo, "The cyber threat landscape: Challenges and future research directions," *Computers & Security*, vol. 30, no. 8, pp. 719–731, 2011.

[8] Gregory Conti and Kulsoom Abdullah, "Passive visual fingerprinting of network attack tools," in *Proceedings of the CCS Workshop on Visualization and Data Mining for Computer Security*, pp. 45–54, Washington, USA, October 2004.

[9] Marco De-Vivo, Eddy Carrasco, Germinal Isern, and Gabriela O de Vivo, "A review of port scanning techniques," *SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp. 41–48, 1999.

[10] Roya Ensafi, Jong Chun Park, Deepak Kapur, and Jedidiah R. Crandall, "Idle port scanning and non-interference analysis of network protocol stacks using model checking," in *Proceedings of the 19th USENIX conference on Security*, pp. 1–17, Berkeley, USA, 2010.

[11] Vincenzo Falletta and Fabio Ricciato, "Detecting scanners: Empirical assessment on 3g network," *International Journal of Network Security*, vol. 9, pp. 143–155, September 2009.

[12] Carrie Gates. *Co-ordinated Port Scans: A Model, A Detector and An Evaluation Methodology.* PhD thesis, Dalhousie University, Halifax, Nova Scotia, February 2006.

[13] Shi Jinn Horng, Ming Yang Su, Yuan Hsin Chen, Tzong Wann Kao, Rong Jian Chen, Jui Lin Lai, and Citra Dwi Perkasa, "A novel intrusion detection system based on hierarchical clustering and support vector machines," *Expert Systems with Applications*, vol. 38, pp. 306–313, January 2011.

[14] hybrid@hotmail.com, "Distributed information gathering," *Phrack Magazine, Article 9*, vol. 9, no. 55, 1999.

[15] Jaeyeon Jung, Vern Paxson, Arthur W Berger, and Hari Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.

[16] KDDCUP99. "Winning strategy in kdd99,". http://kdd.ics.uci.edu/databases/kddcup99/ kddcup99.html, October 28 1999.

[17] Zaiqiang Liu, Dongdai Lin, and Fengdeng Guo, "A method for locating digital evidences with outlier detection using support vector machine," *International Journal of Network Security*, vol. 6, pp. 301–308, May 2008.

[18] Prabhaker Mateti. "Internet security,". http://www.cs.wright.edu/ pmateti/ Internet-Security/Lectures/Probing/index.html. Wright State University.

[19] Bimal Kumar Mishra and Gholam Mursalin Ansari, "Differential epidemic model of virus and worms in computer network," *International Journal of Network Security*, vol. 14, pp. 149–155, May 2012.

[20] Yaling Pei, Osmar R Zaiane, and Yong Gao, "An efficient reference-based approach to outlier detection in large datasets," in *Proceedings of the Sixth International Conference on Data Mining*, pp. 478–487, Washington, USA, 2006. IEEE CS.

[21] Komsit Prakobphol and Justin Zhan, "A novel outlier detection scheme for network intrusion detection systems," in *Proceedings of the International Conference on Information Security and Assurance*, pp. 555–560, Washington, USA, 2008. IEEE CS.

[22] Seth Robertson, Eric V Siegel, Matt Miller, and Salvatore J Stolfo, "Surveillance detection in high bandwidth environments," in *Proceedings of the DARPA DISCEX III Conference*, pp. 130–139, Washington, April 2003.

[23] Bruce Schneier, *Applied cryptography : protocols, algorithms, and source code in C.* New York, USA: John Wiley & Sons, 2nd edition, 1995.

[24] M L Shyu, S C Chen, K Sarinnapakorn, and L Chang, "A novel anomaly detection scheme based on principal component classifier," in *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with ICDM'03*, pp. 171–179, 2003.

[25] Himanshu Singh and Robert Chun. "Distributed port scan detection,". in *Handbook of Information and Communication Security*, pp. 221–234. 2010.

[26] David Whyte. *Network Scanning Detection Strategies for Enterprise Networks.* PhD thesis, School of Computer Science, Carleton University, September 2008.

[27] Vinod Yegneswaran, Paul Barford, and Johannes Ullrich, "Internet intrusions: Global characteristics and prevalence," in *Proceedings of the ACM Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 138–147, California, USA, June 2003.

[28] Hao Lan Zhang. *Agent-based open connectivity for decision support systems.* PhD thesis, School of Computer Science and Mathematics, Victoria University, 2007.

[29] Jiong Zhang and Mohammad Zulkernine, "Anomaly based network intrusion detection with unsupervised outlier detection," in *Proceedings of the IEEE International Conference on Communications*, vol. 5, pp. 2388–2393, June 2006.

[30] Zonghua Zhang, Hong Shen, and Yingpeng Sang, "An observation-centric analysis on the modeling of anomaly-based intrusion detection," *International Journal of Network Security*, vol. 4, pp. 292–305, May 2007.

Monowar Hussain Bhuyan received his M.Tech. in Information Technology from Department of Computer Science & Engineering, Tezpur University, Assam in 2009. Currently, he is pursuing his Ph.D. in Computer Science & Engineering from the same university. He is a life member of IETE, India. His research areas include biometric authentication, data mining, and network security. He has published eight papers in international journals and referred conference proceedings.

Dhruba Kr Bhattacharyya received his Ph.D. in Computer Science from Tezpur University in 1999. Currently, he is a Professor in the Computer Science & Engineering Department at Tezpur University. His research areas include data mining, network security and bioinformatics. Prof. Bhattacharyya has published more than 140 research papers in leading international journals and conference proceedings. Dr. Bhattacharyya also has written/edited 8 books. He is on the editorial boards of several international journals and also on the programme committees/advisory bodies of several international conferences/workshops.

Jugal K. Kalita is a professor of Computer Science at the University of Colorado at Colorado Springs. He received his Ph.D. from the University of Pennsylvania in 1990. His research interests are in natural language processing, machine learning, artificial intelligence, bioinformatics and applications of AI techniques to computer and network security. He has published more than 100 papers in international journals and referred conference proceedings and has written a book. Professor Kalita is a frequent visitor of Tezpur University where he collaborates on research projects with faculty and students.