Study on Masquerade User Detection Techniques using Schonlau's Data Set & Simple Cipher Substitution

Nitish V. Lad, Sagar S. Sawant
MCA, University of Mumbai

***Abstract* – Enforcing good security policies is an intricate job, but bypassing policies is not comparatively difficult; it is often required to share information among different organizations but the current security models are not effective enough to control the potential consequences of wrong-sharing; etc. Hence, it can be implied that access control systems are necessary measures, but not sufficient enough to deal with all the complexities posed by insider attacks.**

It is widely accepted that internal users poses most threat to the security of the computer by misusing their privileges. This may cause potential damages to the organization comparatively more than those by externals. The risks associated with internal security issues can be partially mitigated by the conventional access control models, but systems are much more complex.

In this paper, we discuss different algorithms for the detection of these masquerade attacks. In this paper, we leverage the pattern matching abilities of sequence alignment algorithms to discover masquerade attacks within sequences of information system audit data (e.g., command line entries).

***Keywords* - Masquerade user, Attack, Intrusion Detection System (IDS)**

I.INTRODUCTION

It has been long acknowledged that one of the worst threats to computer security is when internal users misuse their privileges for malicious purposes

What is masquerade attack?

The masquerade attack is a type of attacks, in which a user of a system illegitimately takes the identity of another legitimate user. Masquerade attacks are extremely serious, especially in the case of an insider who can cause considerable damage to an organization. Examples: Identity theft in financial transactionsystems.

Consider the following scenario that would explain the masquerade problem:

A legal user takes a tea break, leaving his/her terminal open and logged in. During the user's absence, an intruder takes the control of the keyboard, and enters commands, taking advantage of the legal user's right and access to programs and data. The intruder's commands may consist of read or write access to private data or he may acquire system right, or install malicious software in the system, etc. Because the intruder is behaving same as the legal user he/she is commonly known as a masquerader. There are many ways for a masquerader to gain access to legitimate user accounts, e.g., through a steal password or hack the system.

Masquerade Detection

Figure 1: shows a general architecture of a masquerade detection system. The essential part is to model user normal behaviour. Once such a model is constructed, it is relatively easy to evaluate test data. A good model must preserve the different characteristics of each user. In masquerade detection, users' history and operation performed are collected and stored, and then users' ongoing operations are examined based on their historical data.

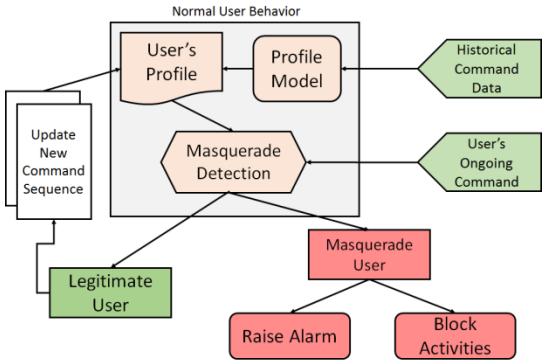

Fig.1: Basic Architecture of Masquerade Detection[5]

Intrusion Detection System (IDS)

An **intrusion-detection system** acquires the information about an information system to perform am identify on the security status of the system. The goal is to discover ***breaches*** of security, attempted ***breaches***, or open weakness that could lead to potential ***breaches***.

A typical intrusion-detection system is shown in Figure 2.

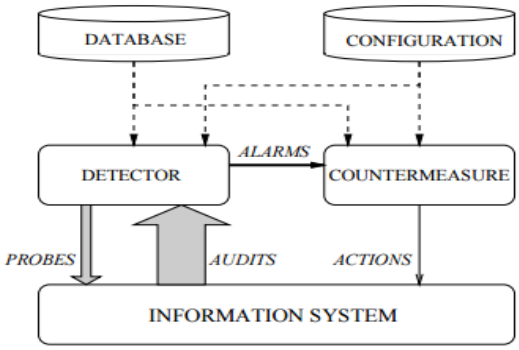

Fig.2: Simple Intrusion Detection System[6]

An intrusion-detection system can be described at a very large level as a detector that processes information coming from the system is to be protected (Fig. 1). This detector can

ISSN: 2278 – 1323

also launch probes to trigger the audit process, such as requesting version numbers for applications. The role of the detector is to eliminate unneeded information from the audit trail. It then presents either a true view of the security-related actions taken during normal usage of the system, or a true view of the current security state of the system. A decision is then taken to idea the probability that these actions or this state can be considered as reason of an intrusion or vulnerabilities. A countermeasure component can then take proper action to either prevent the actions from being executed or change the state of the system back to a secure state.

Efficiency of intrusion-detection systems[6]:

To evaluate the efficiency of an intrusion-detection system, the following parameters are defined:

1. *Accuracy:* Accuracy deals with the proper detection. Inaccuracy occurs when an intrusion-detection system detects a legitimate action in the environment as anomalous or intrusive.
2. *Fault tolerance:* An IDS should itself be resistant to attacks, especially denial-of service-type attacks. This is very important because most intrusion-detection systems run above commercially available operating systems, which are most vulnerable to attacks.
3. *Timeliness:* An IDS has to perform as quickly as possible to ensure the security before much damage has been done.
4. *Performance:*The performance of an IDS is the rate at which events are processed. If the performance of the IDS is poor, then real-time detection is not possible.
5. *Completeness:* Completeness is the property of an IDS to detect all attacks. Incompleteness occurs when the intrusion-detection system fails to detect an attack. This measure is much more difficult to evaluate than the others because it is impossible to have a global knowledge about attacks or abuses of privileges.

II. IDS TECHNIQUES

1. Signature-based Detection and Anomaly based Detection

Signature-based detection systems depend on predetermined patterns that represent misuse. Such a pattern summarizes the distinct characteristics of an attack, often referred to as the signature of an attack. In the detection phase, the IDS records and inspects user activities. It then looks for events that match a predefined pattern. If a match is found, the detection system raises an intrusion alarm. Moreover, with the information associated with the signature, the IDS is able to give a concrete description of the threat when the alarm is raised [2]. Signature-based detection system cannot detect unknown attacks so it needs to be updated on regular intervals. So there is always a lag between updating the new patterns into system.

Advantages:

- Very effective at detecting attacks without generating large number of false alarms.
- Very accurate in detecting known attacks.

Disadvantages: Can only detect those attacks they know about and therefore they must be constantly updated with signatures of new attacks.

- Many misuse detectors are designed to use tightly defined signatures that prevent them from detecting variants of common attacks.

2. Anomaly-based detection systems

Anomaly-based detection systems model user behaviour to determine the characteristics of a user's normal behaviour [2]. During the detection phase, anomaly based systems record and analyses user activities and compares this against their normal behaviour model. A deviation from the established behaviour model is considered an anomaly, or an indication of a possible attack. Often a threshold value is used to define how much deviation will be required before an anomaly is considered an intrusion.

It has been challenging to model the normal behaviour of user as it may vary from time to time. To construct such a model, we must extract distinct characteristics of user behaviour. The user behaviour data must be collected under conditions where no intrusion is in progress [2]. A threshold value is needed to indicate how much deviation will be considered as an intrusion.

If this mechanism is not updated, the system will be obsolete and will result in false positive values. To overcome this problem, most anomaly-based IDSs will update to a new "normal" so that the model can adapt to changes. As it uses the normal user behaviour, it creates security issues as the normal behaviour changes periodically [2].

Advantages:

- Can detect unusual behaviour and thus have the ability to detect symptoms of attacks without specific knowledge of details.
- Can produce information that can in turn be used to define signatures for misuse detectors.
- This system can detect both known and unknown attacks if the attacker's behaviour is different from normal users.

Disadvantages:

- Usually produce a large number of false alarms due to the unpredictable behaviours of users and networks.
- Often require extensive "training sets" of system event records in order to characterize normal behaviour patterns.

A. Advantage of IDS

1. Constant Network Monitoring

- Intrusion detection systems constantly monitor a given computer network for invasion or abnormal activity.
- The advantage of this service is the "round-the-clock" aspect, i.e. the system is protected even when the user is away from any computer hooked up to the network.
- User information, access to the network, and firewall measures are all actively updated and looked after by intrusion detection systems.

ISSN: 2278 – 1323

2. *Versatility of the System*

- Intrusion detection systems are highly customizable to meet specific client needs.
- This allows users to custom-build network security to monitor individual activity from attacks to examining suspicious activity which may be a masquerade attempt to penetrate system security from outside the network.
- The system is able to monitor both the outside threats to a given network, and patterns of behaviour which may be threats operating within the system.

B. Performance criteria

A threshold value is used to determine if the input is an original user or an intrusion user. For example, given a threshold value, if an input is greater than the threshold, this input will be categorized as normal data; otherwise, it will be categorized as intrusion data. The threshold value inversely affects the false positive rate and the false negative rate. That is greater the threshold value lower is false positive rate, while the false positive rate will increase.

The threshold value presents a trade-off between the false positive rate and false negative rate, since neither high false positive rate nor high false negative rate is desirable. High false negative rate leaves many intrusions uncaught, making IDSs useless. High false positive rate, on the other hand, floods IDSs with a large amount of false alarms, eventually causing administrators to ignore true intrusion alarms along with false alarms.

IV. SUBSTITUTION CRYPTANALYSIS TECHNIQUES

1. *Simple Substitution Cipher*

This is oldest cipher systems. Each letter of the plaintext is exchanged by other letter. One-to-one mapping is done between the letters in the plaintext and the cipher text.

Following table represents the simple substitution letter mapping. The conventions used are :

- **Plain Text (P.T):** Lower case
- **Cipher Text (C.T) :**Uppercase

P.T.	A	b	c	d	e	f	g	h	i	j	k	l	m
	N	o	p	q	r	s	t	u	v	w	x	y	z
C.T.	F	G	T	R	W	D	C	V	B	N	M	K	L
	Q	A	Z	X	S	E	P	O	I	U	Y	H	J

Table 1: Plain Text & Cipher Text

Example: By using the above mapping table

- Encryption Process :

Plain Text	hellofromcipher
Encrypted Text	VWKKADSALTBZVWS

- Decryption Process :

Encrypted Text	VWKKADSALTBZVWS
Plain Text	hellofromcipher

2. *Breaking Simple Substitution Ciphers*

Every permutation of the 26 letters can produce a simple substitution key which has a large key space of 26!.Even with such a huge key space, simple substitution is not secure. It can be relatively easy to manually break such a ciphertext by analyzing the letter frequencies and guessing the common words [2].

The nine most frequent letters in English are E, T, A, O, I, N, S, H, and R. After calculating and sorting the letter frequencies in the ciphertext message, an attacker can come up with pretty good guessing by substituting the most frequent letter in the ciphertext with "E", the second most frequent letter with "T", and so on.

Example, *happy* and *hello* have the same letter pattern of ABCCD. This is a manual schema to break simple substitution ciphers.

To evaluate how sensible a half-broken cipher text is an attacker should have good knowledge of English. For such an evaluation a Grading method is needed. To accomplish this task, a decipher system can use much information of the English language, such as the letter frequency counts, bigram frequencies, the most frequent used words, and English grammars. If the grading method is efficient, the decipher system can gradually adjust the key mapping to improve the score of the intermediate deciphered text. Eventually, the decipher system will output a candidate list of plaintexts with high scores.

V. MARKOV CHAIN

A Markov chain method uses state transition statistics for creating sequence of states. In Markov chain method, next state's property is depends on current state. In other word it is called "first order Markov chain". A Markov chain method also contains "higher order Markov chains", its next state property depends on both current state as well as previous states.

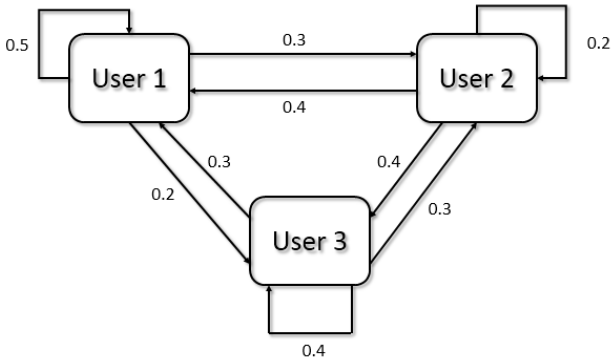

Fig 3: Sharing of single System

This Figure.3 represents Computer Sharing Pattern using Markov Chain. Consider there is single computer which is shared among multiple users. For example User1, User2 and User3. Every user can access the computer for 15 minutes.

ISSN: 2278 – 1323

Once the time slot of current user is expired, the next user will be able to access the computer.
The time slot which is accessed by each user is termed as state and the user is represented as rounded rectangle in the above figure which is followed by the users such as User1, User2 and User3.

Consider the patterns which are observe during sharing the computer. For example, the probability of User3 accessing the system after User2 in next time slot is 40% and can be represented as a unidirectional arrow from User2 to User3 with probability value 0.4 as shown in the figure. The transition of the next state is dependent on current state irrespective of previous state.

To find the Sequence, "Start" and "End" State are added on both the end to Markov chain as shown in the Figure.

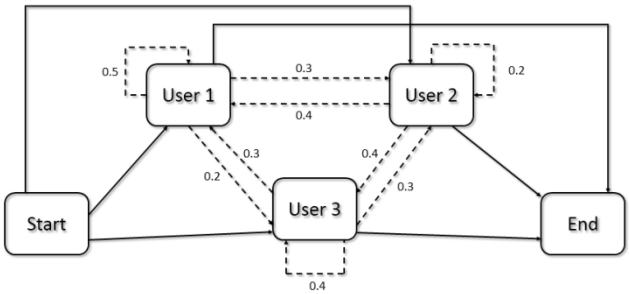

Fig. 4:Markov Chain with Start and End states.

VI. SCHONLAU DATA SET

Data Set of Masquerading user data is collected for Training and Testing purpose by Dr. Schonlau to perform Masquerade Detection and is also known as Schonlau's Data Set.

Figure 5 below illustrates the structure of Schonlau data set. Schonlau Data Set contains 50 individual separate data files, each per user.
Every file contains approximately 15,000 commands (collected using the UNIX audit tool [14]).
For training data purpose, first 5000 commands of legitimate user are used. The next 10000 commands filled as masquerade user commands used for testing data.

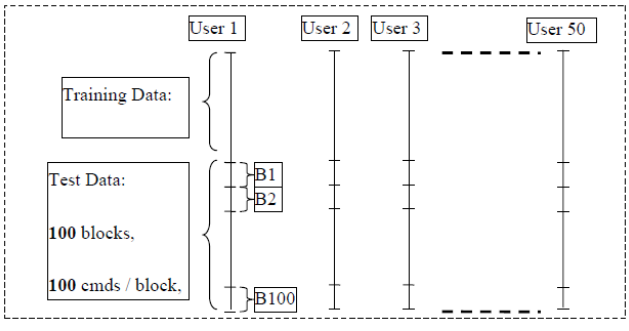

Fig. 5: Schonlau's Data Set

Each file has 100 rows and 50 columns where each column represents a single user and each row represents a test block. Output is either 0 or 1. 0 represents the commands which are not infected by masquerade and 1 represents the infected commands. Schonlau Data Set provides the training data which not infected by any intrusion. This is enough for masquerade detection as it is a particular case of anomaly-based techniques which is independent of signature of intrusion behaviours.

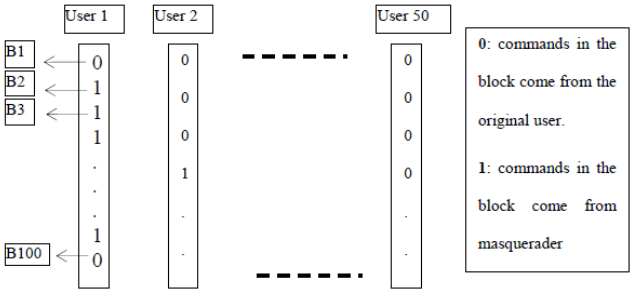

Fig. 6 :Location of the masquerades

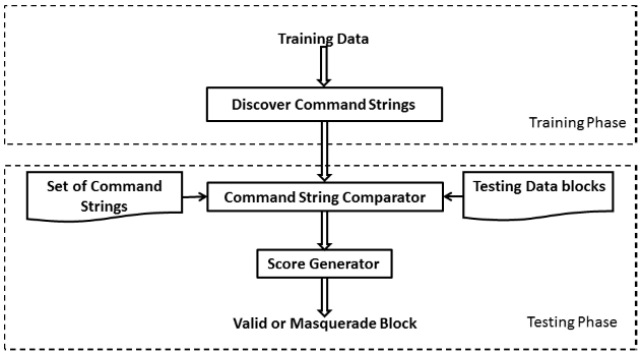

Fig. 7: Proposed approach

Proposed approach contains two phase procedure to masquerade. First phase include training phase which has the technique to discover unique command strings from training data. The Second phase includes identifying the user from testing data.

VII.PROPOSED APPROACH

A. Training Phase

- *Command Strings Generation*

We only combine those pairs of commands which have maximum correlation coefficient. This step takes the training data as input and finds the set of unique commands. It then creates a frequency matrix for these unique commands in all users for calculating Pearson correlation coefficient between each pair of unique command sets.

Proposed approach of discovering command strings has the following main steps.

Step 1: Construct a frequency matrix F

Let us consider m users and uses n distinct commands. These are represented as a frequency matrix of size $m \times n$. This frequency matrix F has elements f_{ij} and they indicate the number of times the command C_j is used by the user U_i.

Step 2: Calculate the correlation coefficient for all pair of commands

This aspect is represented as a correlation matrix R of size $n \times n$. This correlation matrix R has elements r_{ij}which indicates the correlation value between the command pair C_i and C_jThe Pearson's Correlation coefficient is calculated as:

ISSN: 2278 – 1323

$$R(i,j) = \frac{\sum_{k=1}^{m} f_{i,k} f_{j,k} - \frac{\sum_{k=1}^{m} f_{i,k} \sum_{k=1}^{m} f_{j,k}}{m}}{\sqrt{\left(\sum_{k=1}^{m} f_{i,k}^2 - \frac{\left(\sum_{k=1}^{m} f_{i,k}\right)^2}{m}\right)} \sqrt{\left(\sum_{k=1}^{m} f_{j,k}^2 - \frac{\left(\sum_{k=1}^{m} f_{j,k}\right)^2}{m}\right)}}$$

Step 3: Combine the pair having maximum correlation coefficient

From all pair of unique commands we combine those pair of commands who have maximum correlation coefficient out of correlation value.

Algorithm 1: find_max_correlation()

I	**: R:** Correlation Matrix
O	**:** Maximumcorrelation in **R**
1.	Let ***n*** is number of unique commands
2.	**Do for***i* = 1 to *n*-1
3.	*maxcoff* = r_{11}
4.	**Do for***j*= *i*+1 **to**
5.	**If**(*maxcoff*<r_{ij}) **Then**
6.	*maxcoff* = r_{ij}
7.	**Endif**
8.	**EndDo**
9.	**EndDo**

Algorithm 2: Generation of command strings from training dataset

Algorithm-2 uses Algorithm-1 which finds maximum correlation among elements in correlation matrix ***R*** to generate

I	**:** Training data
O	**:**Correlation Coefficient Matrix R, Command String set *S*
1.	*U* : set of unique commands in training data
2.	*n* : number of unique commands
3.	Create frequency matrix *F*
4.	**Do** for all
5.	**Do** for each $u_i \in U$, $i = 1,2, \dots, n-1$
6.	**Do** for each $u_j \in U$, $j = i+1, \dots, n$
7.	r_{ij} = *Cal*_Correlation(*F*, *i*, *j*)
8.	*maxcorr* = find_max_correlation()
9.	**If** (*stringcount*<*threshold*) **Then**
10.	add u_i *and* u_j *to* S_k
11.	*Stringcount*++
12.	**Else**
13.	K++
14.	**EndIf**
15.	**EndDo**
16.	**EndDo**
17.	**EndDo**

B. Testing phase

1. String Comparator

1. As input from Training phase it takes testing data and the set of command strings.
2. From a user's testing data a block of 100 commands is extracted and unique commands are found in this block.
3. This set of unique command is compared with all command strings generated by Training phase.
4. It stores count of commands in a block which is generated in command string.

2. Score generator

1. From string comparator it takes an array of counts and calculates the percentage of count between number of unique commands in a block and the array of count for command strings.
2. If (percentage count > threshold) then the block is valid block otherwise a masquerade.

Algorithm 3: To examine a testing data block as valid or masquerade

I	**:** command string set *S*, testing data
O	**:** Boolean Vector *c(i)* = true if block *b(i)* is masquerade block
1.	*c(i)* = false for all *i*
2.	*k* = number of command strings in *S*
3.	**Do** for all users
4.	*scount* = 0
5.	**Do** for block *i* = 1 **to** 100
6.	find unique commands in block b_i
7.	**Do** for command string *j* = 1 **to***k*
8.	count number of unique commands of b_i present in S_j
9.	*scount* = *scount* + *count*
10.	**EndDo**
11.	**If** (*scount* < threshold) **Then**
12.	c(i) = true
13.	**EndIf**
14.	**EndDo**
15.	**EndDo**

VIII. FREQUENCY STATISTICS

1. N-group: Unigroup, Bigroup, Trigroup, and N-group

An N-group is a subsequence of n items from a given sequence. An Ngroup frequency is the number of the occurrence for an N-group unit. The 1-group, 2-group, and 3-group are often referred to as unigroup, bigroup, and trigroup, respectively. In the example of Section 2.2, the key mapping guessing is based the English letter frequencies, which is an instant of unigroup. Bigroup frequency of English letters is used in [1]. Following is various n-group units generated from the command sequence: **"sh xrdb mkpts env csh csh sh csh kill".**

Command Sequence:
sh xrdb mkpts env csh csh sh csh kill

Unigroup:
Sh, xrdb, mkpts, env, csh, csh, sh, csh, kill

Bigroup:
sh xrdb, xrdb mkpts, mkpts env, env csh, csh csh, csh sh, sh csh, csh kill

ISSN: 2278 – 1323

Trigroup:
sh xrdb mkpts, xrdb mkpts env, mkpts env csh, env csh csh, csh csh sh, csh sh csh, sh csh kill

4-group:
sh xrdb mkpts env, xrdb mkpts env csh, mkpts env csh csh, env csh csh sh, csh csh sh csh, csh sh csh kill

IX. CONCLUSION

Now a days not only the research community but also commercial companies having interest in finding Intrusion. Research products have become noticeable and commercial products are present based on recent research. To detect masquerade users we studied Pearson Correlation Coefficient method.

To find command string we represented simple way that command string contains unique commands which having maximum correlation coefficient from user's training set. We have learned Schonlau's dataset for analysis of proposed method.

We have also studied the detection results of the simple substitution cipher and the uniqueness weighted N-Group models using Schonlau data set. We have found that the simple substitution cipher does not yield better performance than the N-Group since Schonlau data set lack position information required for the simple substitution cipher.Hence we have done the comparative studies of the present algorithms.

In other words, once the simple substitution cipher are constructed using the training data, these models are not updated per user's new behaviors. Other studies on the same data set have yielded better detection results by dynamically updating user profiles. Therefore, future research can be conducted to study how much performance gain can be obtained by exploring the updated algorithms.

X.REFERENCE

[1]. http://www.cs.rpi.edu/~szymansk/papers/csda.08.pdf
[2]. http://www-users.cs.york.ac.uk/~jac/PublishedPapers/InformationTheoreticDetectionMimicry.pdf
[3].http://ids.cs.columbia.edu/sites/default/files/RAID_2011_0.pdf
[4].http://www.cs.cmu.edu/~maxion/pubs/MaxionDSN03.pdf
[5].http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.186.9171&rep=rep1&type=pdf
[6].http://www.syros.aegean.gr/users/tsp/citations_dnl/Debar00a.pdf
[7].http://www.cis.syr.edu/~wedu/Teaching/cis758/LectureNotes/Intrusion_Detection.pdf
[8]. http://www.ehow.com/list_6076479_advantages-disadvantages-intrusion-detection-system_.html
[9] Thomas Jacobsen, A Fast method for the Cryptanalysis of Substitution Ciphers, 1995
[10] Mark Stamp, Information security: principles and practice, Wiley 2005
[11] M Schonlau, W DuMouchel, Computer Intrusion: Detecting Masquerades.
[12] Intrusion Detection Systems - INTRODUCTION, DETECTION METHODOLOGIES:
[13] M Schonlau, Masquerading User Data, http://www.schonlau.net/intrusion.html
[14]. The GNU Accounting Utilities, http://www.gnu.org/software/acct/

About Authors

Nitish Vijay Lad, currently pursuing MCA Final year from ASM's Institute of Management & Computer Studies,IMCOST, Thane which belongs to University of Mumbai, has an interest in learning new technologies and has a unique hobby.

Sagar Sushil Sawant, currently pursuing MCA Final year from ASM's Institute of Management & Computer Studies(IMCOST), Thane, which belongs to University of Mumbai, has an interest in Software Development.

ISSN: 2278 – 1323