CIDD: A Cloud Intrusion Detection Dataset For Cloud Computing and Masquerade Attacks

Hisham A. Kholidy
Dipartimento di Informatica
Università di Pisa, Pisa, Italy
hkholidy@di.unipi.it, hisham_dev@yahoo.com

Fabrizio Baiardi
Dipartimento di Informatica
Università di Pisa, Pisa, Italy
baiardi@di.unipi.it

Abstract

Masquerade attacks pose a serious threat for cloud system due to the massive amount of resource of these systems. Lack of datasets for cloud computing hinders the building of efficient intrusion detection of these attacks. Current dataset cannot be used due to the heterogeneity of user requirements, the distinct operating systems installed in the VMs, and the data size of Cloud systems. This paper presents a Cloud Intrusion Detection Dataset (CIDD)that is the first one for cloud systems and that consists of both knowledge and behavior based audit data collected from both UNIX and Windows users. With respect to current datasets, CIDD has real instances of host and network based attacks and masquerades, and provides complete diverse audit parameters to build efficient detection techniques. The final statistic tables for each user are built by Log Analyzer and Correlator System (LACS) that parses and analyzes user's binary log files, and correlates audits data according to user IP address(es) and audit time. We describe in details the components and the architecture of LACS and CIDD, and the attacks distribution in CIDD.

Key Words: attacks, masquerade, dataset, cloud computing, security, intrusion detection

1. Introduction

Cloud computing presents different threats to an organization than traditional IT solutions [1]. Among those that applies to all cloud computing SPI models [2] we have the abuse and nefarious use of cloud computing and the malicious insiders threats. Both these threats are related to a masquerade attack, where an attacker poses as, or assumes the identity of another legal user to illegally use resources. This attack is a critical one because its exploitation is rather easy as it only requires the knowledge of a user password. The attack may result in trails left at the service location where the masquerader uses cloud resources. Current approaches to masquerade detection integrate anomaly and misuse detection. Anomaly detection foresees future behavior based on past actions. Misuse detection applies a statistical approach to determine if an action constitutes an attack. Several detection solutions are based upon host-based user profiling. To detect masquerade attacks in cloud systems, we consider the sequence alignment technique with a focus on the Semi-Global alignment technique (SGA) [3]. The most important advantages of this technique are its ability of exploiting distinct sequences of audit data, its low false positive and missing alarms rates Furthermore, it tolerates small deviations in the user behavior. To improve the performance and the security efficiency of SGA to fit to the cloud characteristics, we have developed a Heuristic Semi-Global Alignment Approach (HSGAA) [4] that computes the best alignment of the current session to the training sequences of the same user. We apply the HSGAA approach in our CIDS intrusion detection framework [4]. Masquerade detection techniques based on host-based user profiling use a dataset, i.e. a list of the previous used commands, as a source of audit data to build a profile of user signature patterns. Four datasets are currently used to evaluate masquerade detection techniques namely, SEA [5] (with 2 alternatives configurations SEA-I and 1v49), Greenberg dataset [6], Purdue or "PU" [7], and RUU [8]. All these datasets suffer from several deficiencies that hinder their usability in cloud environments. Cloud Intrusion Detection Dataset (CIDD) solves these deficiencies and includes audits parameters to detect not only masquerade attacks but also more than one hundred instances of attacks in distinct categories such as, Denial Of Service (DOS), User to Root (U2R), remote to user, surveillance probing, data attacks (altering and tampering of data), and anomalous user behavior. CIDD includes both knowledge and behavior based audit data. To build CIDD, we have implemented a Log Analyzer and Correlator System (LACS), to extract and correlate user audits from a group of log files in both host and network environments. LACS builds a statistical table for each user with information about the user behaviour. The remainder of this paper is organized as follows: Section 2 briefly reviews masquerade attacks and detection techniques for distinct user profiling environments. It also highlights the HSGAA approach that will use the CIDD dataset. After presenting current masquerade datasets, Section 3 discusses their adoption for cloud systems and the major challenges to build a cloud dataset. Section 4 introduces the architecture of LACS and the structure of CIDD. It also describes the distribution of the users to the VMs and the distribution of attacks and masquerades in CIDD. Lastly, Section 5 draws some conclusion and outlines future work.

2. Detection of masquerade attacks

A masquerader is an attacker who masquerades as a legal user after violating the user account. Security technologies such as firewalls or authentication protocols are useless because, after logging as a legal user, an attacker can exploit any user privilege. To understand the masquerader actions, we have to consider the distinct attacks in [9]. Among them: unauthorized extraction, tampering with data, destruction and deletion of critical assets, use of pirated software, eavesdropping and packet sniffing, spoofing and impersonating other users, and social engineering. Some of these attacks leave some trails in log files linked to the user and log analysis by a host based IDS remains the state-of-the art detection mechanism. To discover the attacks that do not leave an audit trail, masquerade detection techniques can be applied to analyze the user behaviors. Masquerade detection process gathers information about users and builds a profile for each user in terms of information such as login time, location, session duration, and commands issued. Then, user logs are compared against the profiles and a mismatch is signals as an attack. The detection of masquerade attacks is quite difficult because even the legitimate daily activity can easily become malicious according to its context. Furthermore, since cloud systems include a massive amount of resources and users can have different activities in several VMs, a profile can be defined only by correlating these activities. All the approaches described in the following analyze user behaviors according to the sequences of actions, such as user commands, system calls or network operations in distinct environments. In spite of the large number of proposals, the level of accuracy for practical deployment has not been achieved yet. This review highlights the detection techniques based on the analysis of user audits. The audits have been collected by several profiling methods in different environments i.e., UNIX, Windows and/or Network environments. We will see later, how CIDD data were collected through different profiling methods to support different detection techniques.

In UNIX environment, the source of audit data to build signature patterns is a list of UNIX user commands, programs, and system calls. Schonlau et al [12] presented 7 approaches based on sequence of UNIX commands. They compared these approaches using the same user dataset seeded with “simulated” masqueraders. We will highlight this dataset called "SEA" [5] later. Some other approaches, such as recursive data mining with support vector machines [13] and sequence alignment technique [3], consider a UNIX environment.

In Windows environments, there are three log sources: system, application and security. System and application log sources for use by the Windows operating system and Windows applications respectively. Only the Local Security Authority Subsystem Service (lsass.exe) can directly write to the Security log. Events that can be logged fall in several categories: account management, directory service access, logon events, object access, policy change, privilege use, process tracking, and system events. Less research work has addressed the Windows environment [14, 15, 16, and 17] than to the Unix one.

In Network Environments, masquerade detection considers the user network behaviour. The approach in [18] detects masquerade attacks using only basic network statistics because sometimes host data is not accessible or legal/ethical restrictions apply. The approach analyzes the server logs to tag network events with the associated user and build user network profiles by utilizing only anonymized summary data. This limits the privacy impact while avoiding the data accessibility issues associated with host-based approaches. The approach in [19] exploits the well-known Interval Algebra network [20]. The underlying idea is that the signature captures detectable patterns in a user sequence of commands. A user is modeled as a binary constraint network where each node represents an episode of commands. The relationship between a pair of episodes is encoded as the disjunction of interval relations [20]. Any new subsequence should be consistent with at least one user network.

3. Current masquerade datasets

In the following [10, 11] we briefly describe the four datasets currently used to evaluate masquerade detection techniques: SEA, Greenberg, Purdue, and RUU [5-8].

SEA dataset: Most papers about masquerader detection use this dataset [5] with its associated configuration. In principle, this supports the comparison of different detection approaches. SEA consists of commands collected from UNIX acct audit data. Among all the fields of audit data provided by acct only the username and the command were taken. The data describe 50 different users each issuing 15000 commands. The first 5000 commands are considered genuine. The remaining 10000 commands of each user are divided into 100 blocks of 100 commands each. These blocks are seeded with masquerade users, i.e. with data of further users. There is a 1% probability that a block is a masquerader and, in this case, there is a probability of 80% that the following one is a masquerader too. As a result, approximately 5% of the test data contain masquerades. One of most critical defects is that SEA neglects command arguments and parameters. Due to the way acct collects audit data, it is impossible to tell commands typed by human users from those run automatically from shell scripts. This can lead to a very regular pattern of few commands for some users due to the repetitive automated execution of shell scripts. There are two alternatives configurations of SEA, namely, 1v49 [21] and SEA-I [22], to address some methodological shortcomings. However, they have not been widely used and do not simulate masqueraders as in real world data.

Greenberg dataset: This dataset [6] contains data from 168 UNIX users using csh (C shell) as command line shell. Users are classified into four groups: novice programmers, experienced programmers, computer scientists and non-programmers. Data is stored in plain text files that record the following information: session start and end times, the command line as entered by the user, the current working directory, any alias expansion of the previous command, an indication whether the line entered has a history expansion or not, and any error detected in the command line. With respect to SEA, this dataset includes additional information for each command that increases the effectiveness of masquerader detection.

Purdue University dataset: Purdue or just “PU" dataset [7] consists of the UNIX shell command histories of 4 users of the Purdue Millennium Lab, collected in four months. From each user from 7769 to 22530, commands have been collected with an average of about 16500 commands. Commands names, arguments and options are preserved but filenames are omitted. A few works use this dataset and this may be due to its low number of users.

RUU dataset: This dataset was collected by Columbia IDS group [8] to evaluate masquerader attack detection techniques. To this purpose, they built a Windows host sensor to audit process registry behavior and user window touches. The audit program created three types of records: registry actions, process execution, and window touches. The group has built and published only a Windows dataset even if it has also built a Linux host sensor. The dataset was collected from 34 normal volunteer users and 14 masquerade users in a red team exercise. They model how normal users typically search a file system and use these models to detect unusual searches that may be considered as masquerades. The dataset includes records from about 15 minutes of computer use by each masquerader who should perform a specific task to find any data useful for financial gain on a previously unknown file system.

3.1. Deficiencies of current datasets: The previous datasets suffer partially or fully from several deficiencies which prevent their adoption for cloud environments. In this perspective, their most significant weakness is the lack of real masquerade and attack data. No command sequences were issued by attackers, only the RUU dataset includes real masquerades but in a predefined limited scenario. Also the SEA dataset simulates masquerade attacks by randomly inserting a command sequence from one user into those issued by another one. Further problems of these datasets are:

(1) Clouds systems are heterogeneous and user audits may be distributed among VMs running distinct operating systems. Furthermore cloud user runs a larger set of applications than those in the datasets. Current datasets are based on host-based user profiling parameters and lack network parameters such as user IP address, network services and protocols names.
(2) The absence of command arguments and other useful details such as when the user commands were issued and the duration of each user's session.
(3) They are not suitable for training or testing any cloud IDS because of their small size.
(4) Any efficient Cloud dataset should include both behavior based and knowledge based data for better training and coverage for attacks in all cloud service models (SaaS, PaaS, and IaaS).

3.2. Challenges to build a cloud dataset

Building a real intrusion dataset is a hard task, because it takes a long time to collect the training audits and to prepare the scenarios for both training and testing phases. Furthermore, data collection requires special tools to access and monitor the cloud infrastructure and system that requires proper authorization. These are major challenges in cloud systems for several reasons:

(1) Lack of real data to study available solutions and models. Data are out of reach and controlled under the rules of evidence, rather than being a source of valuable information for research purposes. Most cloud systems are commercial and the control of their infrastructures is very difficult if it is not impossible. Private cloud systems are not also suitable because they are not available to external users and thus hinder the building of complete attack scenarios.
(2) It is difficult to collect real data about a malicious or a legal user if audits are distributed across different environments (e.g., Windows, Linux, and network). The heterogeneity of the audit parameters increases the complexity of audit correlation. The complexity is even larger for low level formats e.g., “.bsm” in UNIX, “.evt” in Windows, and “.tcpdump” in network. It is also difficult to build a summarized statistical profile for each user, because categorizing a set of UNIX commands is different from categorizing Windows events and applications.
(3) The huge size of the audit data for cloud systems (more than 20GB for CIDD dataset) and the high number of users require huge computing resources.

4. Cloud Intrusion Detection Dataset (CIDD)

To overcome the problems previously outlined, we built a Log Analyzer and Correlator System (LACS) to parse and correlate user audits from low level log files. We have applied LACS to logs from the DARPA Intrusion Detection Evaluation Group of MIT Lincoln Laboratory [23]. The logs and the TCP dump data are from the Eyrie Air Force Base network that consists of two segments, representing the inside and outside of a government installation. The outside segment consists of 10 machines running Solaris, Linux, and SunOS, 1 SNMP monitor, 1 gateway, and 1 web server. The inside segment consists of 35 machines with Windows NT, Solaris, Linux, and SunOS, 1 inside gateway, and 1 inside sniffer. The network has also 2 routers and 2 hubs. The log files focus mainly on the network audit data. However, we have extracted the host and network audits by parsing the log files collected from, respectively, one Windows NT machine, one Unix Solaris machine, and the raw packet data collected through TCP-dump. CIDD considers both network and host audit data. These data are correlated according to user IP address and audit times. The following section describes the architecture of LACS.

4.1 LACS System

LACS parses the binary log files collected by Unix Basic Security Module (BSM), the security, application and service log files of the Windows event log system, and the raw packet data. In the following, we briefly describe the component of LACS in Figure.1:

- **Parsers components:** LACS has 3 parser components for each independent environment:

1- Solaris parser: The Solaris C2 audit daemon (the auditing capability of BSM) writes binary event data to the local file system. Our parser converts the audit events of this file into a readable text format and stores its output in a local file in the same order entered by each user. This file can be analyzed later by the log analyzer and correlator component. The parser extracts the following parameters: user id, user name, day, time, system calls, path (for commands, files, or directory), login source (IP address or URL), session id, effective uid, attributes and arguments for system call, and return value (call success or failure).

2- Windows parser: It converts to a human readable format the primary binary/encoded Windows security event, and application and service log files. It stores its output in a local file to be analyzed by the log analyzer and correlator component. The parser extracts from the security event log files the following parameters: type (audit success or fail), day, time, event id, source (security log in this case), audit category (e.g., system event, object access, detailed tracking, privilege use, logon/logoff, account management…etc.), user id, user name, audit action, audit parameters (e.g., object name, handle id, privileges, etc…), and description. The parser extracts form the application and service log files the following details: source machine (IP address or URL), user name, Day, time, service or application name, source port, destination port, and target. Most of these applications and services are web applications and mail services.

3- Network parser: It extracts user audits from the raw packets data files collected by TCP-dump. The data contains information about the activities of the user source machine. The parser extracts from the TCP-dump files the following details: day, time, duration, service/protocol name, source port, destination port, source IP, destination IP, a flag value (1 if the packet contaminated with an attack and 0 otherwise), and attack description, if it is clear or stealthy, and if it has a new signature or not.

- **Log analyzer and correlator component:** This is the core component and performs the following tasks:

(1) It correlates the user audits in host and network environments using user IP and audit time. Then it links each audit to the corresponding user.

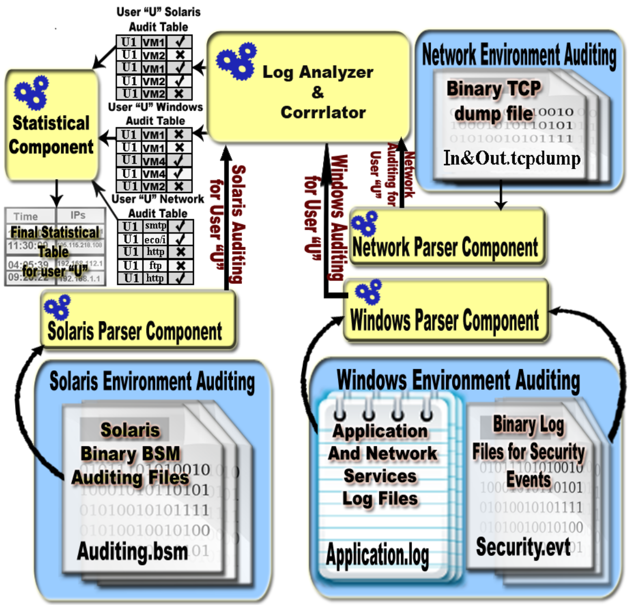
User "U" Solaris Audit Table
U1 VM1 ✓
U1 VM2 ✗
U1 VM1 ✓
U1 VM2 ✓
U1 VM2 ✗
User "U" Windows Audit Table
U1 VM1 ✗
U1 VM1 ✗
U1 VM4 ✓
U1 VM4 ✓
U1 VM2 ✗
User "U" Network Audit Table
U1 smtp ✓
U1 eco/i ✓
U1 http ✗
U1 ftp ✗
U1 http ✓
Statistical Component
Log Analyzer & Corrrlator
Final Statistical Table for user "U"
Time
IPs
Solaris Auditing for User "U"
Windows Auditing for User "U"
Network Auditing for User "U"
Network Environment Auditing
Binary TCP dump file
In&Out.tcpdump
Network Parser Component
Windows Parser Component
Solaris Parser Component
Solaris Environment Auditing
Solaris Binary BSM Auditing Files
Auditing.bsm
Windows Environment Auditing
Application And Network Services Log Files
Application.log
Binary Log Files for Security Events
Security.evt

Figure.1: The architecture of LACS system

(2) It assigns user audits to a set of VMs according to their login sessions time and the characteristic of the user task. During audit collection, each user logs into the network in one or two different time shifts, one in the morning and the other in afternoon or evening and sometimes both. It also assigns user morning sessions to one VM and the other sessions to another VM. Section.4.2 describes the distribution of the users to the VMs.

(3) It marks malicious audit records according to attacks and masquerades tables given by MIT group [23]. The marking is done according to attack time, date,

destination IP/URL and the name of victim user. It also marks some audit records in a session with different time and/or different source IP than the training audit data stored for the user.

(4) It produces the final tables that store the marked audits for each individual user with its assigned VMs. This step produces tree tables namely, Solaris, Windows, and network audit tables, each with a different sequence of audits. The Solaris table contains a sequence of user actions. The Windows table stores a sequence of user actions e.g., security actions and opened applications and services. The Network table stores valuable information on the sequence of machines, network services and protocols accessed by the user, and normal times and dates of accesses. These tables enable any IDS to follow the sequence of user audits in different environments. The following equation correlates the audit in the three tables:

$$P_{Cmasq}(U_i) = \sum_{j=1}^{m} \left(\frac{P(U_i) * P(IP_j)}{\sum_{k=1}^{n} P(U_k)} \right) + P(U_i)$$

Where:

- $P_{Cmasq}(U_i)$: Probability that U_i is a masquerader according to U_i behaviors in any cloud node. It includes the probability that the masquerader can be detected by the login IP(s).
- $P(U_i)$: Probability that U_i is a masquerader according to U_i behaviors in any cloud node. It does not include user IP behaviors.
- *m:* Number of IP(s) that U_i uses to login to the cloud.
- *n:* Number of cloud users who share the same IP_j of U_j
- *k:* An index for the current user who shares the same IP of U_i.
- *j:* An index for the current IP address of U_i.
- $P(IP_j)$: Probability that IP_j reveals to be a masquerader.

Consider, as an example, a simple case where U_1, U_2 and U_3 share the same login IPs, IP_1 *and* IP_2. Also suppose that the probabilities that IP_1, *and* IP_2 could be used by a masquerader are: $P(IP_1) = 0.4$, and $P(IP_2) = 0.5$, and the probabilities that U_1, U_2, and U_3 reveal to be masqueraders according to their behaviors in all the cloud nodes are: $P(U_1)=0.4$, $P(U_2)=0.3$, and $P(U_3)= 0.6$, and the detection threshold $\theta=0.75$. We use the previous equation to compute $P_{Cmasq}(U_i)$ for each U_i to determine which one is a real masquerader according to both the corresponding host and network audits.

$P_{Cmasq}(U_1)$= ((0.4*0.4) / (0.4+0.3+0.6) + (0.4*0.5) / (0.4+0.3+0.6)) + 0.4 = 0.6769 < θ (not masquerader)

$P_{Cmasq}(U_2)$= ((0.3*0.4) / (0.4+0.3+0.6) + (0.3*0.5) / (0.4+0.3+0.6)) + 0.3 = 0.5076 < θ (not masquerader)

$P_{Cmasq}(U_3)$= ((0.6*0.4) / (0.4+0.3+0.6) + (0.6*0.5) / (0.4+0.3+0.6)) + 0.6 = 1.0153 > θ (masquerader)

- **The statistical component:** It builds host and network based statistics according to the three previous tables. Host based statistics include: number of login failures, logging times (morning, afternoon, evening, and nights), logging source address(es), a list with common commands and system calls used by the user (in case of Unix Solaris system), a list of common services, applications, and security actions(in case of Windows NT), and VMs names used by each user. Network based statistics are based on the IP address and include: list of network services and protocols used, list of machines accessed, hours and days at which the IP becomes active, and list of failures.

4.2 CIDD Architecture

CIDD audit data consists of two parts. The first one is a collection of Unix Solaris audits and their corresponding TCP dump data. The second part includes Windows NT audits and their corresponding TCP dump data. As any intrusion dataset, CIDD consists of training and testing data for both parts 1 and 2. In training data of part 1, CIDD has 7 weeks (35 days) of Unix Solaris audits and TCP dump data with labeled attacks which can be used to train any IDS with a set of attack signatures. Week 6 contains 21 real masquerade attacks in UNIX audits which can be used to test any anomaly detection based IDS. Figure 3 shows the distribution of these masquerade attacks in week 6 data. Most of audits of CIDD users are distributed across several VMs. Users with less than 5 login sessions have been deleted. CIDD has 84 Solaris users distributed into 4 VMs. Users are categorized according to the applications of their host machines into:

- 2 programmers sharing VM1 and VM2,
- 1 secretary using VM3 and VM4,
- 1 system administrator using VM3 and VM4,
- 56 normal users using VM3 and VM4 to issue UNIX commands, exchange mails, and internet navigation,
- 22 advanced users that access VM1 and VM2 to run advanced applications for some special tasks.

In testing data of part 1, CIDD has 2 weeks (10 days) of Unix Solaris audits and their corresponding TCP dump data for testing purpose. The data include more than one hundred instances of attacks categorized to distinct categories such as Denial Of Service (DOS), User to Root (U2R), remote to user, surveillance probing and anomaly attacks. In training data of part 2, CIDD has 3 weeks (15 days) of Windows NT audits and their corresponding TCP dump data with labeled attacks only in the second week. CIDD has 44 Windows NT users with a complete windows audit data. Some of these users exist in part1 audits with the same names. Users are distributed among VMs as following: 5 in VM1, 32 in VM2, and 7 that share both VM1 and VM2. In testing data of part 2, CIDD has 2 weeks (10 days) of Windows NT audits and their corresponding TCP dump data for testing purpose. Part 2 testing data contains 38 real masquerade attacks in Windows NT audits. Some of these attacks result from one or several U2R attacks, while others are implemented through human masquerader action. One user of the inside network segment implements masquerade attacks, while "outsider" are due to either users of the outside network or outside the simulation network. Part 2 testing data has the same attack categories of part 1 and a further category, data attacks.

The CIDD webpage [24] describes further details such as user statistic tables, masquerade distributions, the simulated network of the auditing experiments, user distribution to VMs, attacks database and their categories and snapshots for both training and testing data. Table 1 compares CIDD against publicly available datasets.

Table 1: Comparison of publicly available datasets

Dataset	Normal Users	Audit Format	Audit Parameters	Real Masquerades?	Audit Environment	Sessions	Attack Signatures
SEA	50	Unix commands	No	Simulated	Unix hosts	No	No
Greenberg	168	Unix commands	yes	Simulated	Unix hosts	Yes	No
PU	4	Unix commands	yes	Simulated	Unix hosts	Yes	No
RUU	34	Windows audits	yes	Real but with predefined scenarios	Windows hosts	Yes	No
CIDD	128 (84 in UNIX + 44 in Windows)	Complete Unix audits, Windows audits, TCP dumps	yes	Real time masquerades and more than 100 instances of real attacks	Hosts(Unix, Windows), Network, Grid and Cloud	Yes and assigned to VMs	Yes

5. Conclusion and future work

Masquerade attacks are critical for cloud systems, as they enable the attacker to control cloud resources. Current datasets are not suitable to train and evaluate cloud IDSs, because they neglect the typical behaviours of a cloud user. CIDD can solve these deficiencies and provides the complete audit parameters that help in detecting more than hundred instances of attacks and masquerades that exist in CIDD. To build CIDD, we have developed a log analyzer and correlator system to parse and analyze the host based log files and network packets. CIDD has different audits from different environments, to increase its usability in different systems and thus simplify the support of distinct detection techniques.

For further refinements of CIDD, we plan to deploy CIDD audits in a real cloud and use our HSGAA approach [4] to detect masquerade attacks.

6. References

[1] "Top Threats to Cloud Computing", Cloud Security Alliance, http://www.cloudsecurityalliance.org/csaguide.pdf, V. 1.0 (2010)

[2] I., Foster, Z.Yong , I. Raicu, S. Lu, , "Cloud Computing and Grid Computing 360-Degree Compared", Grid Computing Environments Workshop, GCE '08, pp.1-10, Nov. 2008

[3] S. E. Coull, J. W. Branch, B. K. Szymanski, E.A. Breimer. 2008. "Sequence alignment for masquerade detection". Journal of Computational Statistics & Data Analysis. 52, 8 (April 2008),pp4116-4131.

[4] H. A. Kholidy, F. Baiardi, "CIDS: A framework for Intrusion Detection in Cloud Systems", in the 9th Int. Conf. on Information Technology: New Generations ITNG 2012, April 16-18, Las Vegas, Nevada, USA.

[5] http://www.schonlau.net/intrusion.html

[6] S.Greenberg,: Using unix: Collected traces of 168 users. Report 88/333/45, , Univ. of Calgary, 1988.

[7] T.Lane , C.E. Brodley,: An application of machine learning to anomaly detection. In Proc. of the 20th National Information Systems Security Conference. (1997) 366-380

[8] RUU dataset: http://sneakers.cs.columbia.edu/ids/RUU/data/"

[9] A. H. Phyo, S. M. Furnell. A detection-oriented classification of insider and misuse. In Proc. of the Third Security Conf., 2004.

[10] M. Bertacchini and P. Fierens. "A Survey on Masquerader Detection Approaches", In Proc. of 2008 Congreso Iberoamericano de Seguridad Informatica, pages 46-60,.

[11] M. B. Salem, S. Hershkop, S. J. Stolfo. "A Survey of Insider Attack Detection Research" in Insider Attack and Cyber Security: Beyond the Hacker, Springer, 2008

[12] M. Schonlau, ,W. DuMouchel, W. Ju, A.F. Karr, M. Theus, , and Y. Vardi, "Computer Intrusion: Detecting Masquerades", Statistical Science 16(1), 58-74, 2001.

[13] B. Szymanski, B., Y. Zhang, Y.. Recursive Data Mining for Masquerade Detection and Author Identification. In Proc. of the 5th IEEE System, Man and Cybernetics Information Assurance Workshop, West Point, June 2004, 424-431.

[14] A. Garg, R. Rahalkar, S. Upadhyaya, K. Kwiat, "Profiling Users in GUI Based Systems for Masquerade Detection", in the 2006 IEEE Workshop on Information Assurance, pp.48-54.

[15] E.Imsand, J., Hamilton, "GUI Usage Analysis for Masquerade Detection",. in the 2007 IEEE Workshop on Information Assurance. June 20-22, West Point, NY, USA. Pages 270-277

[16] A. Wespi, M. Dacier, and H. Debar, "Intrusion Detection Using Variable-Length Audit Trail Patterns,", LNCS., vol. 1907, pp. 110-129, Springer-Verlag, 2000.

[17] C. Marceau, "Characterizing the behavior of a program using multiple-length N-grams," in the 2000 workshop on New security paradigms, County Cork, Ireland, pp. 101-110.

[18] C. Strasburg, S. Krishnan, K. Dorman, S. Basu, J. Wong, "Masquerade Detection in Network Environments", the 2010 10th IEEE/IPSJ Int. Symp.. on Applications and the Internet.

[19] A. Kumar, S. Rawat, G. Vijaya Kumari, ,A. K. Pujari, "Masquerade Detection Using IA Network", CPSec 2005.

[20] J. Allen,.: Maintaining knowledge about temporal intervals. Communications of the ACM, 26, (1983) 832-843

[21] R.A.Maxion, T.N. Townsend,. Masquerade Detection Using Truncated Command Lines. in the Int. Conf. on Dependable Systems and Networks, Washington, D.C., June 2002,

[22] R.Posadas, J.C. Mex-Perera, R. Monroy, J.A. Nolazco-Flores, Hybrid method for detecting masqueraders using session folding and hidden markov models. In Gelbukh, Garcia, eds.: MICAI. LNCS 4293., Springer (2006) 622-631

[23] "DARPA IDS Group", http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html

[24] http://www.di.unipi.it/~hkholidy/projects/cidd/