IDDP: Novel Development of an Intrusion Detection System through Design Patterns

Rajshekhar M Patil
Research Scholar, CSE Dept., MGRU, Chennai-85, Tamil Nadu, India,
Assistant Professor, Information Science Department, Atria Institute of Technology, Bangalore-4, Karnataka, India.

Mamithar. R. Patil
Lecturer, T I T, Bhopal, Madhya Pradesh, India.

Dr. K V. Ramakrishnan
Professor, Dept. of CSE, CMRIT, White Field, Karanataka-78, India.

Dr. T.C.Manjunath
Ph.D. (IIT Bombay), Fellow IETE
PRINCIPAL, Atria Inst. of Tech., Bangalore, Karnataka, India

ABSTRACT
A novel method of design & development of an intrusion development system through design patterns is presented in this paper. Large scale use of computers and networking in various day to day businesses and individual communication applications has given rise to security issues. The process of monitoring the events occurring in a computer network and analyzing them for any sign of intrusion is known as IDS. Design pattern is a metric that measures how much of an object oriented design can be understood and represented as IDS. This paper presents a quantifiable and observable definition of metric for IDS. The IDS through design pattern is easier to implement compared to techniques like IDDM and IDS through UNIX system calls. The quantitative results shown in this paper projects the effectiveness of the proposed method that can be widely used in security systems.

General Terms
Knowledge Data Discovery 99, SNORT, Functional Points, Pattern.

Keywords
IDS - Intrusion Detection System, FP-Functional Points, IDDM - Intrusion Detection in Data Mining.

1. INTRODUCTION
Intrusion detection systems (IDS) were proposed to complement prevention-based security measures. An intrusion is defined to be a violation of the security policy of the system; intrusion detection thus refers to the mechanisms that are developed to detect violations of system security policy. Intrusion detection is based on the assumption that intrusive activities are noticeably different from normal system activities and thus detectable. Intrusion detection is not introduced to replace prevention-based techniques such as authentication and access control; instead, it is intended to complement existing security measures and detect actions that bypass the security monitoring and control component of the system. Intrusion detection is therefore considered as a second line of defense for computer and network systems. Generally, an intrusion would cause loss of integrity, confidentiality, denial of resources, or unauthorized use of resources [30].The process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusion is known as Intrusion Detection. The known events are called as functional points and should be carried on regularly. Computer network systems are known to be vulnerable to external attacks This has led us to carrying out the investigation of the problem of detecting misuse of computer network [1]. Many researchers around the world have worked on the current topic, viz., intrusion detection systems. The following paragraphs gives a brief survey about the same.

The duty of securing networks is very difficult due to their size, complexity, diversity and dynamic situation. The advantage of securing networks being to enhance security applications such as Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Adaptive Security Alliance (ASA), check points and firewalls and further guide to the security implementers [19].

PGNIDS (Pattern-Graph based Network Intrusion Detection System) generates the audit data that can estimate intrusion with the packets collected from network. An existing IDS (Intrusion Detection System), when it estimates an intrusion by reading all the incoming packets in network, takes more time than the proposed PGNIDS does. The PGNIDS not only classifies the audit data into alert and log through ADGM (Audit Data Generation Module) and stores them in the database, but also estimates the intrusion by using pattern graph that classifies IDPM (Intrusion Detection Pattern Module) and event type, Therefore, it takes less time to collect packets and analyze them than the existing IDS, and reacts about abnormal intrusion real time. In addition, it is possible for this to detect the devious intrusion detection by generating pattern graph [20], [26].

With the rapid growth of the internet, computer attacks are increasing at a fast pace and can easily cause millions of dollar in damage to an organization. Detection of these attacks is an important issue of computer security. Intrusion Detection Systems (IDS) technology is an effective approach in dealing with the problems of network security. In general, the techniques for Intrusion Detection (ID) fall into two major categories depending on the modeling methods used: misuse detection and anomaly detection. Misuse detection compares the usage patterns for knowing the techniques of compromising computer security [21], [26].

Although misuse detection is effective against known intrusion types; it cannot detect new attacks that were not predefined. Anomaly detection, on the other hand, approaches the problem by attempting to find deviations from the established patterns of

usage. Anomaly detection may be able to detect new attacks. However, it may also cause a significant number of false alarms because the normal behavior varies widely and obtaining complete description of normal behavior is often difficult. Architecturally, an intrusion detection system can be categorized into three types host based IDS, network based IDS and hybrid IDS [21], [22], [26].

A host based intrusion detection system uses the audit trails of the operation system as a primary data source. A network based intrusion detection system, on the other hand, uses network traffic information as its main data source. Hybrid intrusion detection system uses both methods [23]. However, most available commercial IDS's use only misuse detection because most developed anomaly detector still cannot overcome the limitations (high false positive detection errors, the difficulty of handling gradual misbehavior and expensive computation[24]). This trend motivates many research efforts to build anomaly detectors for the purpose of ID [25], [26].

In [27], Vokorokos & Balaz presented an intrusion detection system which informs the system administrator about potential intrusion incidence in a system. This designed architecture employs statistical method of data evaluation, that allows detection based on the knowledge of user activity deviation in the computer system from learned profile representing standard user behavior. Srilatha *et.al*. proposed a new method of feature deduction and ensemble design of intrusion detection systems in their paper in [30].

A brief review about the intrusion detection systems follows.

Firewalls and other simple boundary devices lack some degree of intelligence when it comes to observing, recognizing, and identifying attack signatures that may be present in the traffic they monitor and the log files they collect. Without sounding critical of such other systems' capabilities, this deficiency explains why intrusion detection systems (often abbreviated IDS) are becoming increasingly important in helping to maintain proper network security. Whereas other boundary devices may collect all the information necessary to detect (and often, to foil) attacks that may be getting started or already underway, they haven't been programmed to inspect for and detect the kinds of traffic or network behavior patterns that match known attack signatures or that suggest potential unrecognized attacks may be incipient or in progress [28], [29].

In a nutshell, the simplest way to define an IDS might be to describe it as a specialized tool that knows how to read and interpret the contents of log files from routers, firewalls, servers, and other network devices. Furthermore, an IDS often stores a database of known attack signatures and can compare patterns of activity, traffic, or behavior it sees in the logs it's monitoring against those signatures to recognize when a close match between a signature and current or recent behavior occurs. At that point, the IDS can issue alarms or alerts, take various kinds of automatic action ranging from shutting down Internet links or specific servers to launching back-traces, and make other active attempts to identify attackers and actively collect evidence of their nefarious activities [28], [29].

By analogy, an IDS does for a network what an antivirus software package does for files that enter a system: It inspects the contents of network traffic to look for and deflect possible attacks, just as an antivirus software package inspects the contents of incoming files, e-mail attachments, active Web content, and so forth to look for virus signatures (patterns that match known malware) or for possible malicious actions (patterns of behavior that are at least suspicious, if not downright unacceptable) [28], [29].

To be more specific, intrusion detection means detecting unauthorized use of or attacks on a system or network. An IDS is designed and used to detect and then to deflect or deter (if possible) such attacks or unauthorized use of systems, networks, and related resources. Like firewalls, IDSs may be software-based or may combine hardware and software (in the form of preinstalled and preconfigured standalone IDS devices). Often, IDS software runs on the same devices or servers where firewalls, proxies, or other boundary services operate- an IDS not running on the same device or server where the firewall or other services are installed will monitor those devices closely and carefully. Although such devices tend to operate at network peripheries, IDS systems can detect and deal with insider attacks as well as external attacks [28], [29].

1.1 Characterizing Intrusion Detection Systems

IDS systems vary according to a number of criteria. By explaining those criteria, we can explain what kinds of IDSs you're likely to encounter and how they do their jobs. First and foremost, it's possible to distinguish IDSs on the basis of the kinds of activities, traffic, transactions, or systems they monitor. In this case, IDSs may be divided into network-based, host-based, and application-based IDS types. IDSs that monitor network backbones and look for attack signatures are called network-based IDSs, whereas those that operate on hosts defend and monitor the operating and file systems for signs of intrusion and are called host-based IDSs. Some IDSs monitor only specific applications and are called application-based IDSs. (This type of treatment is usually reserved for important applications such as database management systems, content management systems, accounting systems, and so forth.) Read on to learn more about these various types of IDS monitoring approaches [28], [29]:

1.1.1 Network-based IDS characteristics

Pros: Network-based IDSs can monitor an entire, large network with only a few well-situated nodes or devices and impose little overhead on a network. Network-based IDSs are mostly passive devices that monitor ongoing network activity without adding significant overhead or interfering with network operation. They are easy to secure against attack and may even be undetectable to attackers; they also require little effort to install and use on existing networks.

Cons: Network-based IDSs may not be able to monitor and analyze all traffic on large, busy networks and may therefore overlook attacks launched during peak traffic periods. Network-based IDSs may not be able to monitor switch-based (high-speed) networks effectively, either. Typically, network-based IDSs cannot analyze encrypted data, nor do they report whether or not attempted attacks succeed or fail. Thus, network-based IDSs

require a certain amount of active, manual involvement from network administrators to gauge the effects of reported attacks [28], [29].

1.1.2 Host-based IDS characteristics

Pros: Host-based IDS can analyze activities on the host it monitors at a high level of detail; it can often determine which processes and/or users are involved in malicious activities. Though they may each focus on a single host, many host-based IDS systems use an agent-console model where agents run on (and monitor) individual hosts but report to a single centralized console (so that a single console can configure, manage, and consolidate data from numerous hosts). Host-based IDSs can detect attacks undetectable to the network-based IDS and can gauge attack effects quite accurately. Host-based IDSs can use host-based encryption services to examine encrypted traffic, data, storage, and activity. Host-based IDSs have no difficulties operating on switch-based networks, either.

Cons: Data collection occurs on a per-host basis; writing to logs or reporting activity requires network traffic and can decrease network performance. Clever attackers who compromise a host can also attack and disable host-based IDSs. Host-based IDSs can be foiled by DoS attacks (since they may prevent any traffic from reaching the host where they're running or prevent reporting on such attacks to a console elsewhere on a network). Most significantly, a host-based IDS does consume processing time, storage, memory, and other resources on the hosts where such systems operate [28], [29].

1.1.3 Application-based IDS characteristics

Pros: An application-based IDS concentrates on events occurring within some specific application. They often detect attacks through analysis of application log files and can usually identify many types of attack or suspicious activity. Sometimes application-based IDS can even track unauthorized activity from individual users. They can also work with encrypted data, using application-based encryption/decryption services.

Cons: Application-based IDSs are sometimes more vulnerable to attack than the host-based IDS. They can also consume significant application (and host) resources.

In practice, most commercial environments use some combination of network- and host- and/or application-based IDS systems to observe what's happening on the network while also monitoring key hosts and applications more closely. IDSs may also be distinguished by their differing approaches to event analysis.

Some IDSs primarily use a technique called signature detection. This resembles the way many antivirus programs use virus signatures to recognize and block infected files, programs, or active Web content from entering a computer system, except that it uses a database of traffic or activity patterns related to known attacks, called attack signatures.

Indeed, signature detection is the most widely used approach in commercial IDS technology today. Another approach is called anomaly detection. It uses rules or predefined concepts about "normal" and "abnormal" system activity (called heuristics) to distinguish anomalies from normal system behavior and to monitor, report on, or block anomalies as they occur.

Some IDSs support limited types of anomaly detection; most experts believe this kind of capability will become part of how more IDSs operate in the future. Read on for more information about these two kinds of event analysis techniques [28], [29] :

1.1.4 Signature-based IDS characteristics

Pros: A signature-based IDS examines ongoing traffic, activity, transactions, or behavior for matches with known patterns of events specific to known attacks. As with antivirus software, a signature-based IDS requires access to a current database of attack signatures and some way to actively compare and match current behavior against a large collection of signatures. Except when entirely new, uncataloged attacks occur, this technique works extremely well.

Cons: Signature databases must be constantly updated, and IDSs must be able to compare and match activities against large collections of attack signatures. If signature definitions are too specific, signature-based IDS may miss variations on known attacks. (A common technique for creating new attacks is to change existing, known attacks rather than to create entirely new ones from scratch.) Signature-based IDSs can also impose noticeable performance drags on systems when current behavior matches multiple (or numerous) attack signatures, either in whole or in part [28], [29].

1.1.5 Anomaly-based IDS characteristics

Pros: An anomaly-based IDS examines ongoing traffic, activity, transactions, or behavior for anomalies on networks or systems that may indicate attack. The underlying principle is the notion that "attack behavior" differs enough from "normal user behavior" that it can be detected by cataloging and identifying the differences involved. By creating baselines of normal behavior, anomaly-based IDS systems can observe when current behavior deviates statistically from the norm. This capability theoretically gives anomaly-based IDSs abilities to detect new attacks that are neither known nor for which signatures have been created.

Cons: Because normal behavior can change easily and readily, anomaly-based IDS systems are prone to false positives where attacks may be reported based on changes to the norm that are "normal," rather than representing real attacks. Their intensely analytical behavior can also impose sometimes-heavy processing overheads on systems where they're running. Furthermore, anomaly-based systems take a while to create statistically significant baselines (to separate normal behavior from anomalies); they're relatively open to attack during this period.

Today, many antivirus packages include both signature-based and anomaly-based detection characteristics, but only a few IDSs incorporate both approaches. Most experts expect anomaly-based detection to become more widespread in IDSs, but research and programming breakthroughs will be necessary to deliver the kind

of capability that anomaly-based detection should be, but is currently not, able to deliver.

Finally, some IDSs are capable of responding to attacks when they occur. This behavior is desirable from two points of view. For one thing, a computer system can track behavior and activity in near-real time and respond much more quickly and decisively during early stages of an attack. Since automation helps hackers mount attacks, it stands to reason that it should also help security professionals fend them off as they occur. For another thing, IDSs run 24/7, but network administrators may not be able to respond as quickly during off hours as they can during peak hours (even if the IDS can page them with an alarm that an attack has begun). By automating a response to block incoming traffic from one or more addresses from which an attack originates, the IDS can halt an attack in process and block future attacks from the same address [28], [29].

By implementing the following techniques, IDSs can fend off expert and novice hackers alike. Although experts are more difficult to block entirely, these techniques can slow them down considerably:

- Breaking TCP connections by injecting reset packets into attacker connections causes attacks to fall apart.
- Deploying automated packet filters to block routers or firewalls from forwarding attack packets to servers or hosts under attack stops most attacks cold-even DoS or DDoS attacks. This works for attacker addresses and for protocols or services under attack (by blocking traffic at different layers of the ARPA networking model, so to speak).
- Deploying automated disconnects for routers, firewalls, or servers can halt all activity when other measures fail to stop attackers (as in extreme DDoS attack situations, where filtering would only work effectively on the ISP side of an Internet link, if not higher up the ISP chain, as close to Internet backbones as possible).
- Actively pursuing reverse DNS lookups or other ways of attempting to establish hacker identity is a technique used by some IDSs, generating reports of malicious activity to all ISPs in the routes used between the attacker and the attackee. Because such responses may themselves raise legal issues, experts recommend obtaining legal advice before repaying hackers in kind [28], [29].

1.2 Definition of Metric

Our objective is to develop an overall framework for defending against attacks and threats to computer system. IDS design pattern metric is precise and explicit about what it is based on, namely collaborations. There are patterns that can't be neatly captured using collaborations.

The first point to be raised is some of these design patterns are not in a design level but rather on an architecture or programming level. Data generated from network tends to have very high volume, dimensionality and heterogeneity. More important is it does not seem sensible to mix (largely) orthogonal design aspects into one metric. A better approach might be to have a pattern density metric for each major type of aspect in a given system. We can make progress towards automated calculation of the metric design pattern density.

1.3 Intrusion pattern & collaborations

Here, in this section, we define the making of object collaborations as the atomic unit of functionality with which to measure the number of design pattern instances in a given frame work. "IDS design pattern" is defined as a percentage of a frame works functionality. that can be explained as design pattern instances. For this we need a measure of functionality on the level of granularity of design patterns so that we can measure and represent that functionality.

1.4 ATOMICITY AND PATTERNS

Here, in this section, we focus on the atomicity of the classic design pattern and we ignore aspects like simultaneous existence. The variation of patterns can be explained in its granularity (from architectural styles to intrusion programming idioms) as well as in design and address.

Atomicity level is addressed by classic design pattern and its class method level. It is a refined atomicity level in comparison with pipes and filters [6]. It is finer atomicity to identify similarity of statements and create a dissimilar one or take suitable decision later on which is the responsibility of Intrusion Prevention System. Specific purpose of a design pattern is distributed across its objects and collaborations. Hence each object is acting as an agent in a network system [11].

Responsibilities are distributed across object classes & are done by configuring their instances for a specific purpose. The focus is on the object collaboration rather than class structure. Flexibility of the design is recalled through inheritance. However, all design patterns are not about collaborations some are about how architectural and structural design aspects of intrusion detection system behavior can be connected. In the next section we explain about such situations collaborations and its design.

Based on the work done by various researchers so far as described earlier, there were lot of drawbacks & disadvantages in the network security & in the intrusion detection systems such as the network being prone to some of the deadly viruses & could not detect when there is huge amount of data. Some of the drawbacks of the above mentioned works were considered in our work, rectified, improvised some of the concepts, developed & proposed a new framework of network security with a sophisticated intrusion detection system.

The paper is organized in the following sequence. Firstly, a brief introduction about the research work was presented in the previous paragraphs in the introductory section. Secondly, the development of the JAVA frameworks is dealt with in the section II. The section III deals with the design using advanced UML's. Object communication & inheritance is briefly dealt with in section IV. Section V deals with the detailed design of the IDS design patterns. Case studies are dealt with in section VI. In the section VII, future work is presented. Conclusions are presented in the last section, i.e., in section VIII. This is followed by the references & the author biographies.

2. JAVA FRAME WORK

Junit is a widely used unit testing frame work for Java. The central abstraction in the frame is observed in various patterns. Pictures of mature object design's show this same pattern density. The star of the design has a rich set of relationships with the supporting players [3].The mature frame works "exhibit a high design pattern density". This paper attempts to utilize design pattern density so that we can track its value in the evolution of a given frame work. The metric is applied to various case studies which are then interpreted based on the results.

An enhanced definition of collaboration based design is used to define a quantitative measure of functionality in a class model. Object collaborations are used as atomic unit of functionality. This makes it easy to assess the number of design patterns instances. Thus the calculation of a framework's design pattern density becomes the percentage of collaboration instances. An enhanced definition of collaboration based design can easily cope with inheritance interfaces design pattern density is a quantitative and measurable entity.

3. DESIGN USING ADVANCED UML

A detailed design using advanced UML technology is presented in this section. It is related to the class responsibility collaboration [CRC] but very much independent of IDS design. Collaboration based design has made its way into UML [12].

Table 1: Data from the collaboration design view of the Junit 3.8.

Name of collaboration	Total collaborations	Roles in each collaborations	All methods included in each collaboration	Pattern name if any else nil
Test Case	1	2	4	-
Test suit Test creation	1	2	4	-
Test Run	1	2	1	command
Test case Test Run	1	2	2	-
Test suite Test Run	1	2	1	-
Test Hierarchy	1	3	11	composite
Test Result	1	3	7	Collecting parameter
Test Result Controller	1	2	2	-
Test Result Observer	1	2	5	Observer
Collecting Test Run	1	2	4	Command
Test Run Method	1	2	4	Template Method
Assertions	1	2	34	-
Test Failure	1	2	4	-
Comparison Failure	1	2	3	-
Compact Method	1	2	4	Composed Method
Total	16	40	94	7

This paper uses the Junit frame work as a running example [14]. Junit is a frame work for writing unit tests in Java. It is available in source code form.

We focus on the Junit frame work classes only. The discussion in this paper is based on our own method of Junit 3.8 using IDS collaboration [11][12].

This paper also uses UML concept of interface to represent a role and the UML concept of package to scope collaboration. Compared with UML2.x as well as our work this is a simplified metric definition. In design based collaboration objects play important roles.

"A role is a type that defines the behavior of an object within collaboration and collaboration is grouping of roles that defines how objects behind these roles are allowed to interact."

Table 1 explains the test result observed between two agents from Junit 3.8. each agent defines how a test result object allows for registration and un-registration of test listener objects interpreted in an object. For this purpose test listener objects provide call back methods that the test result objects can invoke.

It will happen only when test run starts, the time it ends failure occurs. This process is an application of observer patterns; similarly the other classes are defined. One more class test result is shown in Table 2 and its roles and collaboration are shown in Table 3. One can notice here using UML interfaces to represent a role doesn't imply that on the code level any such interface exists. Methods defined by roles are directly embedded in Junit interface Table 2.

4. OBJECT COMMUNICATION AND INHERITANCE

Communication and collaboration between objects is an important aspect, dealt through inheritance. The inheritance interface that super classes define as a contact between subclasses is also important. Collaboration based IDS design with the way to specify and using inheritance interfaces without such enhancement is difficult to explain white box or gray box functional points.

Like Junit explains key concept is object may play several roles in class like observing, listening etc. We can explain intra object communication with the same approach as inter object communication. As an example of the above we use template method in this pattern. Figure4 shows an application of the template method using textual notations for communication and collaboration.

5. IDS DESIGN PATTERN

We now provide here a quantitative and measurable definition of IDS design pattern using pattern density. The IDS design pattern density of an object oriented IDS is the percentage of its collaboration between agents that are design pattern instances.

For example, as table 2 shows the core agent Junit frame work classes are composed from 15 agent collaborations, of these 6 are instances of design pattern. Hence as in table 3 the design pattern of Junit 3.8, 6/15 or 48%. The 'IDS metric design pattern density' is simple precise complete and measurable, it is simple and precise because only a basic collaboration is needed. It is complete because of its enhanced collaboration based design method. After evaluation of this metric it is found that agent collaboration attacks detected is less than 0.5. It is apparent that this method gives slightly different results than the method with other metric. Still this is considered as most successful IDS design pattern density.

Table 2 : Detection for Detecting Pattern Using metrics

Method	Evaluation Metrics			Detection Rate
	Collaboration Or Test 1	Test 2	Test 3	
IDDM	3/3 (100%)	7/11 (63.6%)	2/3 (66.7%)	13/19 (68.4%)
SNORT IDS	3/3 (100%)	10/11 (90.1%)	2/3 (66.7%)	16/19 (84.2%)

Table 3: Summary data from Junit 3.8 analysis

Junit 3.8 case study :	
No of class interfaces	09
No of collaborations	15
No of pattern instances	06
No of roles in total	32
No of ratio roles	2.8
IDS design pattern	48 %

The interface architecture of core collaborations of IDS design pattern if any one is interested in they can do it as

```
Public collaboration Test run {
Free role client {
------- }
Role template method {
Public void runb ( ) throws throwable;
}
Role primitive method {
Protected void runtest( ) throws throwable;
Protected void setup ( ) throws exception;
Protected void teardown ( ) throws exception;
}
}
```

6. CASE STUDIES

In addition to Junit 3.8 design pattern we used the metric to gather data from two other functional points.

1. The IDDM architecture based implementation of distributed object[14].
2. The IDS system to design IDS design pattern using unix system calls[15].

The IDDM and IDS design are the result of a major revision and hence is not analyzed now but the major functional points are analyzed and deduced for further utilization.

6.1 Case study data

Table 4 below shows summary data and the design pattern from the two case studies excluding the Junit case study.

The two new case studies are assessed an interface architecture level so the numbers given in table 4 are interface architecture design pattern.

Table 4 : Summary data from two case studies

Case study	[1]	[2]
No of interfaces and interface classes	15	11
No of collaborations	30	18
No of pattern instances	16	09
No of roles assigned	72	42
Ratio per class/interface	3.4	2.4
Design pattern density (interface architecture)	56%	60%
[1] IDDM frame work. [2] IDS design pattern frame work		

Table 5 summarizes the pattern densities and assigns a maturity level to each frame work. The maturity level is a simple integer value 1-3, where 1 represents "new", 2 shows "revised" and 3 shows "mature"

Table 5: the maturity level, pattern, roles per collaboration, and data of the case studies.

Case study	Maturity level (1- 3)	Design pattern density	No of roles collaboration
Assessed on the interface architecture level			
IDDM	02	58%	2.12
IDS for unix system calls	02	60%	2.10
Assured on the complete design			
junit	2.8	45%	2.05

7. FUTURE WORK

Several IDS design pattern schemes for designing network intrusion patterns are proposed in this paper. which is applied to various data sets. The work presented in this paper makes a lot of assumptions that restrict the applicability of the metric.

Currently applying neural networks in intrusion detection may also be used, which will give a robust approach to ensure security in the network system. Further, neural networks are also alternatives to other approaches in the area of intrusion detection.

8. CONCLUSION

This paper presents a unique metric for IDS design pattern. Several IDS schemes designing and detecting network intrusions are proposed. If an extension collaboration based design as the instrument to calculate the metric value in a given frame work. The paper makes IDS design metric more precise and also measurable. To do so we show collaboration based design are not only used to capture inter object collaboration, but also can be extended to capture class inheritance interfaces. The metric is applied to two case studies followed by discussion of their assessment & further justified by using case studies.

REFERENCES

[1] Dorothy Denning "An intrusion detection model" IEEE Trans. on software engineering, No. 2, pp. 272-280, Feb 1987.

[2] J M Bradshaw, "An introduction to software agents" in software agents, Bradshaw J M (ED), Cambridge M A : MIT press, 1997.

[3] Kent Beek and Erich Gamma Junit: A cook's tour available from http:// Junit://Junit.source .net/doc/cook'stour/cookstour.html.

[4] Fayyad.U.Piatesky-Shapiro, G and Smyth . P.1996 The KDD process of extracting useful knowledge from volumes of data communication ACM 39,11, 27-34.

[5] Dirk Riechale *et. al.* "Design pattern validated".

[6] Rajashekaran.S: efficient parallel hierarchical clustering algorithms . IEEE Trans. in parallel and distributed systems 16(6), 497-502 (2005).

[7] Dirk Rachele, Roger Bradman,Thomas Gross and matzel "pattern density and role modeling of an object transport service" ACM Computing surveys 32(Mar. 2000), No. 10.

[8] James O coplin Advanced C++ programming styles and idioms.Addison Wesley, 1991.

[9] Helmer G. S K Wang "Host based IDS" lowa state university International conference on applications" Maths (Jul. 2002) 4 97 – 501.

[10] Erich Gamma, Richord helm, Ralph Johnson .Design patterns: Elements of reusable object oriented software, Addison Wesley – 1995.

[11] Paul Dokas ,Vipin kumar "network Intrusion Detection",200 UNION STREET SE 192,CSE building UOM,MINNEPOLIS, MN 55455 U S

[12] Dirk Rachele junit 3.8 documented using collaborations. In S E notes, Vol. 33, No. 2, Art. 5, Mar. 2008, ACM 2008.

[13] SNORT Intrusion Detection System.www.snort.org.

[14] TCP TRACE software tool,www.tcptrace.org.

[15] P C Mahalanobis, on tests and measures of groups Divergence ,IJNS of Bengal,1930.

[16] J Mc Hugh,1998 Lincoln laboratory IDS evaluation (A critique),proceedings of the recent advances in intrusions detection system 145 – 161, France 2000.

[17] Li X: Parallel algorithms for hierarchical clustering and clustering validity. IEEE trans, pattern analysis and machine intelligence,12,1088 – 1092(1990).

[18] R K Cunnigham ,R P Lippman , results of the 1999 darpa off line IDS ,RAID- 99 IN 1999.

[19] Iftikhar Ahmad, Azween B Abdulah and Abdullah S Alghamdi, "Towards the Designing of a Robust Intrusion Detection System through an Optimized Advancement of Neural Networks", Advances in Computer Science and Information Technology, Lecture Notes in Computer Science, Volume 6059, 2010, 597-602, DOI: 10.1007/978-3-642-13577-4_53.

[20] Byung-kwan Lee, Seung-hae Yang, Dong-Hyuck Kwon and Dai-Youn Kim, "PGNIDS", Computational Science and Its Applications - ICCSA 2006, LNCS, 2006, Volume 3982 / 2006, pp. 38-47, DOI: 10.1007/11751595_5

[21] J., Muna. M. and Mehrotra M., "*Intrusion Detection System : A design perspective"*, 2rd Int.Conf. On Data Management, IMT Ghaziabad, India. 2009.

[22] M. Panda, and M. Patra, "*Building an efficient network intrusion detection model using Self Organizing Maps*", proceeding of world academy of science, engineering and technology,Vol. 38. 2009.

[23] M. Khattab Ali, W. Venus, and M. Suleiman Al Rababaa, "*The Affect of Fuzzification on Neural Networks Intrusion Detection System"*, IEEE computer society.2009.

[24] B. Mykerjee, L. Heberlein T., and K. Levitt N., *"Network Intrusion Detection"*, IEEE Networks, Vol. 8, No.3, PP.14-26. 1994.

[25] W. Jung K., "Integration Artificial Immune Algorithms for Intrusion Detection", dissertation in University of London, pp. 1-5.2002.

[26] Muna Mhammad T. Jawhar & Monica Mehrotra, "Design Network Intrusion Detection System using hybrid Fuzzy-Neural Network International" Journal of Computer Science and Security, Int. J. of Comp. Sci. and Security, Volume (4), Issue (3) pp. 285-294, 2010.

[27] Vokorokos, L. Balaz, A., "Host-based intrusion detection system", Technical University of KoÂice, 14th Int. Conf. on Intelligent Engg. Systems (INES-2010), Las Palmas, Spain, pp. 43 – 47, 5-7 May 2010.

[28] http://www.windowsecurity.com/articles/What_You_Need_to_Know_About_Intrusion_Detection_Systems.html

[29] Robert J. Shimonski, "Security+ Study Guide and DVD Training System", Published: Nov 18, 2002, Updated: Jul 23, 2004.

[30] Srilatha Chebrolu, Ajith Abraham, Johnson P. Thomas, ature deduction and ensemble design of intrusion detection systems", Journal of Computers & Security, Volume 24, Issue 4, pp. 295-30, June 2005.

Mr. Rajashekar Patil, currently, is working as Assistant Professor in the Department of Information Sciences & Engg. Dept., Atria Institute of Technology, Bangalore, Karnataka, India. He is also a research scholar in the prestigious Dr. MGR Deemed University & simultaneously doing his research work & progressing towards his Ph.D. in the computer science field. He has also published a number of research papers in various national & international journals & conferences. He has conducted a number of seminars, workshops, conferences, summer courses in various fields of computer science & engineering. His research interests are Data Mining, Computer Networks, Parallel computing, Java based programming, Intrusion detection systems, etc.