**International Journal of Computer Science and Engineering (IJCSE)**
**ISSN(P): 2278-9960; ISSN(E): 2278-9979**
**Vol. 2, Issue 5, Nov 2013, 129-134**
**© IASET**

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

# INTELLIGENT NETWORK SNIFFER USING DATA MINING

**MITESH JAIN & ARUNA GAWADE**

Department of Computer Engineering, Dwarkadas J. Sanghvi College of Engineering,

University of Mumbai, Mumbai, Maharashtra, India

## ABSTRACT

A packet sniffer is a computer program or a piece of computer that can intercept and log traffic passing over a digital network or part of a network. The goal of a network sniffer is to detect malicious traffic over the network. The sniffer monitors all the incoming and outgoing traffic. However, the flaw with the traditional Intrusion Detection Systems (IDS) [1] is that they can't detect the newly emerging cyber threats. In this paper, we propose a novel technique of an Intelligent Network Sniffer (INS) [1] using the data mining algorithms for measuring the deviation of malicious traffic from normal profile. We use the Iterative Dichotomizer-3 (ID3) [3] algorithm for this purpose.

**KEYWORDS:** Data Mining, Entropy, IDS, INS, IDS, Network Sniffing

## INTRODUCTION

As network technologies have been developed quickly, they have brought us a whole new experience of a new life and shopping. Business transactions over the internet have attracted enterprises' and users' attention. Since network-based computer systems today play a vital role in modern society they also have become the target of intrusions by our enemies and criminals.

Network Sniffers usually act as network probes or snoops, examining network traffic, however not intercepting or altering it. A Network Sniffer program works by asking the Network Interface Card (NIC) [3] of a computer to stop ignoring all the traffic headed to other computers on the Network and pay attention to them. This is achieved when the Network Sniffer program places the NIC in the promiscuous mode. The NIC, in a promiscuous mode [2] enables a machine to be able to see all the data transmitted in the network.

The data being transmitted over the network has very strict formatting as the data travels in the form of packets which have specific formats of their respective protocols. A Network Sniffer program penetrates these heavy layers of data encapsulation in order to gain key information such as the identity of the source and destination along with all the data that is being exchanged between them.

## RELATED WORK

As we can learn from the definition, a network sniffer can also be used as a network monitor to monitor the traffic in network. Thus in a network we can also detect any malicious attacks from outside as well as inside the network. Sniffer functionality also includes Intrusion Detection System (IDS). An IDS monitors network and/or system activities for malicious activities or policy violations and produces reports to a Management Station. Although there are various IDS already available there exist following loop holes in these. They are:

- The type of user i.e. normal/genuine user, spy, intruder, unauthorized user etc. has to be detected manually from the network data traffic.

- Due to this, at times, even a genuine user may be classified as a threat and may be denied the appropriate services. This is called as occurrence of a False Positive.
- Also, many a times, the concerned IDS or Sniffer may not alert the administrator when there is actually an attack or intrusion taking place. This is called as occurrence of a False Negative.

## PROPOSED PROBLEM DEFINITION

So now we propose the idea of an Intelligent Network Sniffer (INS). The Sniffer is Intelligent enough to automatically identify the type of user from the analysis of network traffic and classify it.

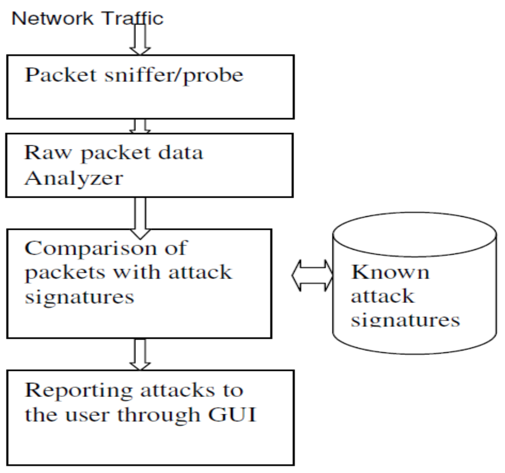
Network Traffic
Packet sniffer/probe
Raw packet data Analyzer
Comparison of packets with attack signatures
Known attack signatures
Reporting attacks to the user through GUI

**Figure 1: Problem Definition**

Firewall (software or hardware) [12] is a mechanism that filters data that enters our network from the Internet and protects the network from malicious content. However it has a limitation. As shown in the above diagram, the firewall prevents the hacker from the ‘Outside Attack’ but it can’t detect the ‘Inside Attack’. This can be achieved using out technique.

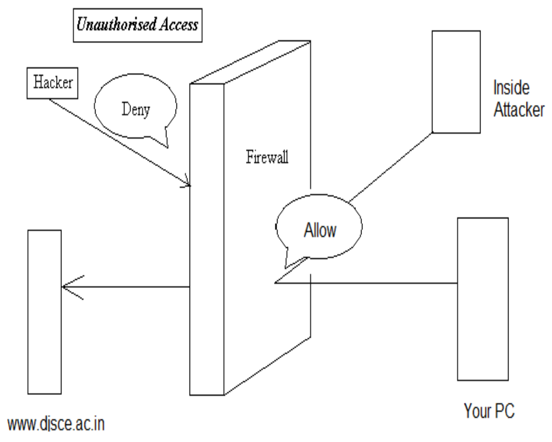

**Figure 2: The Proposed Architecture of Our College LAN**

## DATA MINING TECHNIQUES

We use data mining [6] for detecting frequent patterns of data packets on the network. Association Rules [6] in data mining is an effective technique for the same. This recognition of patterns is called as Knowledge Discovery [6].

We then create a rules table that classifies the users from the patterns recognised using association rules above. Further we need to classify the users as per the rules table. So we construct the decision tree using the ID3 algorithm [1].

## ID3 Algorithm

ID3 stands for Iterative Dichotomizer-3, which is a mathematical algorithm for building fastest and shortest decision trees. The resultant tree is used to classify the users.

The ID3 [4] algorithm can be summarized as follows:

- Take all unused attributes and count their entropy concerning test samples
- Choose attribute for which entropy is minimum (or, equivalently, information gain is maximum)
- Make node containing that attribute

The algorithm is as follows:

ID3 (Examples, Target Attribute, Attributes)

- Create a root node for the tree
- If all examples are positive, Return the single-node tree Root, with label = +.
- If all examples are negative, Return the single-node tree Root, with label = -.
- If number of predicting attributes is empty, then Return the single node tree Root, with label = most common value of the target attribute in the examples.
- Otherwise Begin
  - A = The Attribute that best classifies examples.
  - Decision Tree attribute for Root = A.
  - For each possible value, $v_i$, of A,
    - Add a new tree branch below Root, corresponding to the test A = $v_i$.
    - Let Examples($v_i$) be the subset of examples that have the value $v_i$ for A
    - If Examples($v_i$) is empty
      - Then below this new branch add a leaf node with label = most common target value in the examples
    - Else below this new branch add the sub tree ID3 (Examples($v_i$), Target Attribute, Attributes – {A})
  - End
  - Return Root

## ID3 Metrics

The algorithm is based on Occam's razor: it prefers smaller decision trees (simpler theories) over larger ones. However, it does not always produce the smallest tree, and is therefore a heuristic. Occam's razor is formalized using the concept of information entropy:

## Entropy

$$E(S) = -\sum_{j=1}^{n} f_S(j) \log_2 f_S(j) \qquad (1)$$

Where:

- E(S) is the information entropy of the set S;
- n is the number of different values of the attribute in S (entropy is computed for one chosen attribute)
- $f_S(j)$ is the frequency (proportion) of the value j in the set S
- $\log_2$ is the binary logarithm

Entropy is used to determine which node to split next in the algorithm. The higher the entropy, the higher is the potential to improve the classification here.

Gain: Gain is computed to estimate the gain produced by a split over an attribute:

$$G(S,A) = E(S) - \sum_{i=1}^{m} f_S(A_i)E(S_{A_i}) \tag{2}$$

Where:

- G(S,A) is the gain of the set S after a split over the A attribute
- E(S) is the information entropy of the set S
- m is the number of different values of the attribute A in S
- $f_S(A_i)$ is the frequency (proportion) of the items possessing $A_i$ as value for A in S
- $A_i$ is $i^{th}$ possible value of A
- $S_{A_i}$ is a subset of S containing all items where the value of A is $A_i$

Gain quantifies the entropy improvement by splitting over an attribute: higher is better.

Entropy is the randomness in a system. A completely homogenous system has entropy of 0i.e there is no randomness in the system. A completely heterogeneous system has entropy of 1 i.e. total random system. Intruder may use the unused port numbers and IP address as resources to hack the system. So we create a rule table to prevent this, as follows:

**Table 1: Rule Table**

| Sr No | IP Address | Port No | Protocol Type | Intruder Result |
|---|---|---|---|---|
| 1 | Known | Known | HTTP | False |
| 2 | Known | Unknown | TCP/IP | True |
| 3 | Unknown | Known | FTP | True |
| 4 | Known | Known | UDP | False |
| 5 | Unknown | Unknown | SMTP | True |
| 6 | Unknown | Known | HTTP | True |
| 7 | Known | Known | FTP | False |
| 8 | Known | Unknown | UDP | True |
| 9 | Known | Known | SMTP | False |

In the above table, total no of entries in the rules table are 9. Total no of entries having True values are 5 and that of False are 4.

Positive entropy = No of false values/ Total no of entries in the rule table.

= 5/9

Negative Entropy = No of false values/ Total no of entries in the rule table

= 4/9

Entropy is calculated as:

Entropy (S) = $-\sum P \log_2 P$

Hence, Entropy(S) = - (5/9) log (5/9) – (4/9) log (4/9) = 0.2983

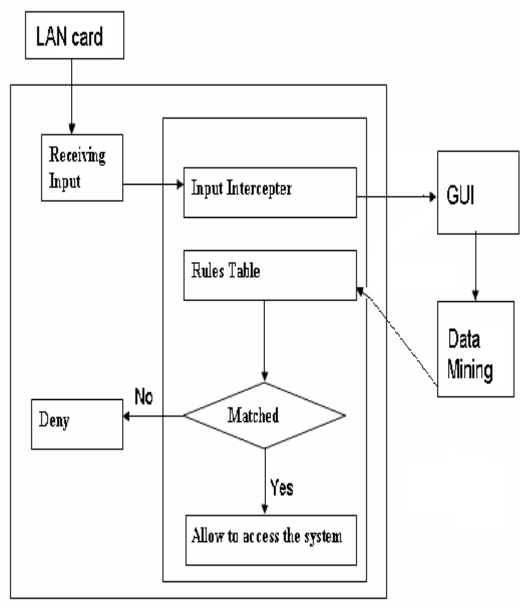

**Figure 3: Block Diagram of INS**

## JPCAP

In field of computer network administration pcap (packet capture) [9] consists of an application programming interface (API) for capturing network traffic. Jpcap [8] is a Java library for capturing and sending network packets. Using Jpcap, we develop applications to capture packets from a network interface and visualize/analyse them in Java. We can also develop Java applications to send arbitrary packets through a network interface. Jpcap has been tested on Microsoft Windows (98/2000/XP/Vista), Linux (Fedora, Mandriva, Ubuntu), Mac OS X (Darwin), FreeBSD, and Solaris. Jpcap can capture Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, and ICMPv4 packets. Jpcap is open source, and is licensed under GNU LGPL.

## CONCLUSIONS

Using an intelligent network sniffer [11] we can automatically detect or receive alerts about the malicious attacks on out network and also about the intrusion of a third party in our network. We can easily detect patterns using java program [7] of the intrusion using data mining algorithms thereby prevent the attack. Apart from prevention we can also monitor our network by analysing the traffic with the help of INS [10]. Thus with time new patterns can be added to the database and we can therefore detect the threats. It also aims to support the daily work for networking engineers, developers, administrators or Linux users by providing support with or in network monitoring, protocol analysis, reverse engineering, network debugging and penetration testing

## FUTURE SCOPE

This bare bone implementation of a network sniffer is the foundation towards development of better and more complicated security tools. The immediate future of the project is to realize the implementation of the advanced features of Port Scanning, Network Mapping and Client Configuration Monitoring capabilities. By using the search-promiscuous plug-in of ettercap we can now get the IP addresses of sniffers. If we can find out the corresponding MAC addresses, it is possible to send mails to such sniffing machines warning them about their suspicious activities. Finally we can develop a script running in the server to send mails to sniffing machines if a list of MAC addresses is given as input.

## REFERENCES

1. Fang-YieLeu and Kai–Wei Hu, “A Real –time Intrusion Detection System using Data Mining Technique”, IEEE Electrical and Computer Engineering , May/mai 2003, pp.789-792
2. Meera Gandhi, S. K. Srivatsa, “Detecting and preventing attacks using network intrusion” Detection systems
3. Biswanath Mukherjee, L. Todd Heberlein, Karl N. Levitt, “Network Intrusion Detection:” IEEE, June 1994.
4. Lee, W., S. J. Stolfo, and K. W. Mok, In Proc. of the 1999IEEE Symp., “Data mining framework for building intrusion detection models”, On Security and Privacy, Oakland, CA, pp. 120132 IEEE Computer Society Press, 9- 12 May 1999.
5. Victor H. Garcia, Raul Monroy, and Maricela Quintana, “Web Attack Detection Using IDS*,” Computer Science DepartmentTecnologico de Monterrey, Campus Estado deMexicoCarretera al lago de Guadalupe, Km 3.5, Atizapan, 52926, Mexico
6. Klaus Julisch, IBM Research Zurich Research Laboratory McHugh, J. (2000), “DATA MINING FOR INTRUSION DETECTION: A Critical Review”, The 1998 Lincoln Laboratory IDS Evaluation.
7. http://gforgeek.blogspot.com/2005/04/simple-packet-sniffer-using-java.html
8. http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/
9. http://en.wikipedia.org/wiki/Pcap
10. http://en.wikipedia.org/wiki/Packet_analyzer
11. http://www.packet-sniffer.net/
12. http://en.wikipedia.org/wiki/Firewall