

ANOMALY DETECTION IN COMMUNICATION NETWORKS

Prof. D.J.Parish and Francisco Aparicio-Navarro

Loughborough University (School of Electronic, Electrical and Systems Engineering).

Learning Outcomes

1. To understand where and how anomalies can occur in communication network traffic.
2. To understand how Network Intrusions and Traffic Performance changes can be detected.
3. To investigate how a selection of Anomaly based Detection techniques function in relation to communication networks.

ANOMALY DETECTION IN COMMUNICATION NETWORKS

INTRODUCTION

This lecture covers the following topics:

- Network Traffic Characteristics
- Anomalies in Communication Network Traffic
- Phases and Examples of Network Attacks.
- Rule Based Detection Approaches.
- Anomaly Based Intrusion Detection.
- Detecting Anomalies in Network Traffic.
- Performance Evaluation

Network Traffic Characteristics

Many traffic related parameters for communication networks are derived from, or related to three fundamental characteristics:

- Bandwidth – Average, Peak etc.
- Delay – Absolute and Variance.
- Loss.

More complex metrics can be related to the above such as:

- Communication medium.

Transfer Delay

Packet Transfer Delay is the time from when the first bit of a cell leaves one observation point to when the last bit of the cell passes the second observation point. It is also associated with the:

- Packetization delay, which is the time needed to accumulate the bits needed to form the cell.

- Propagation delay, which is due to the physical signal propagation over the communication medium.

Packet Delay Variation

DV indicates the variation in transfer delay suffered by a stream of packets.

The end-to-end delay of a packet can be given by:

$$D + W.$$

Where D is a fixed element of delay and W is a variable element of delay.

The inter-arrival time of two packets at a receiver can be given by:

$$\text{Delta} = (D + W[1]) - (D + W[2])$$

However, DV can be defined in a number of ways.

Anomalies in Communication Network Traffic

Changes in network traffic characteristics have been termed “Data Exceptions” by the author and others. A frequently observed Data Exception, known as a ‘Step Change’ (referred to as a ‘plateau’ by McGregor and Braun), occurs when the average delay either increases or decreases in such a way that all test packets are affected. This means that the delay is altered by a constant amount for each packet. Step changes can occur when the network is reconfigured. An example of a Step Change type Data Exception is shown in Figure 1. In this example, the delay is shown as a selection of delay percentiles, measured over a 3 hour window.

Step Change Examples

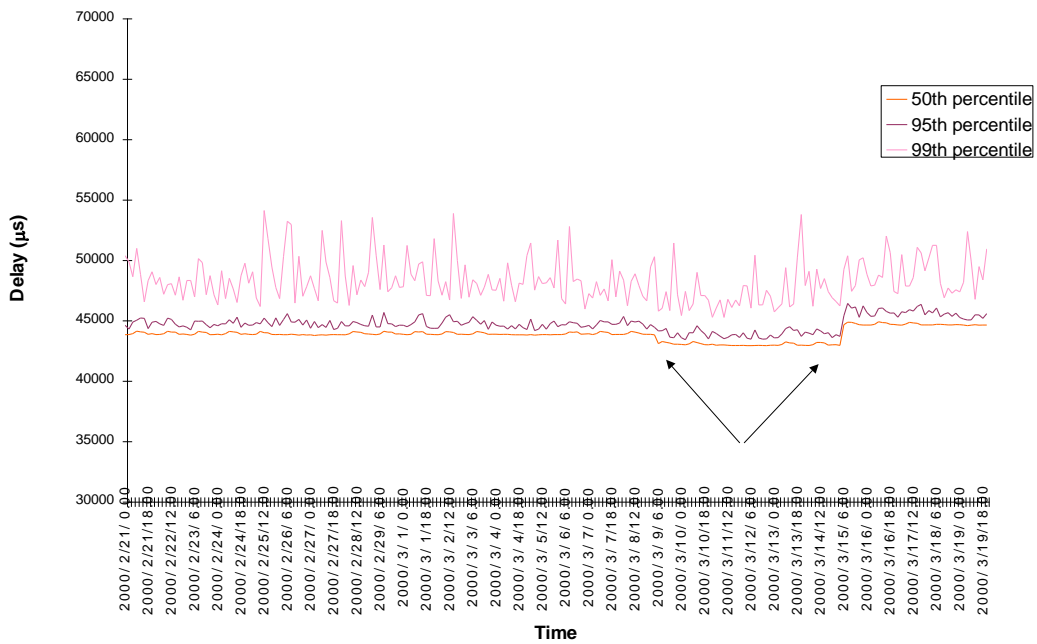


Figure 1 - Step Change Exception

Spike Data Exceptions are sharp increases in delay that last for a relatively short period of time (McGregor and Braun also use the term 'spike'). The increase need not affect all the test packets although the more test packets it affects the more significant it is likely to be. An example of such an Exception is shown in Figure 2. Spikes are often a characteristic of network equipment behaviour or short term malfunctions.

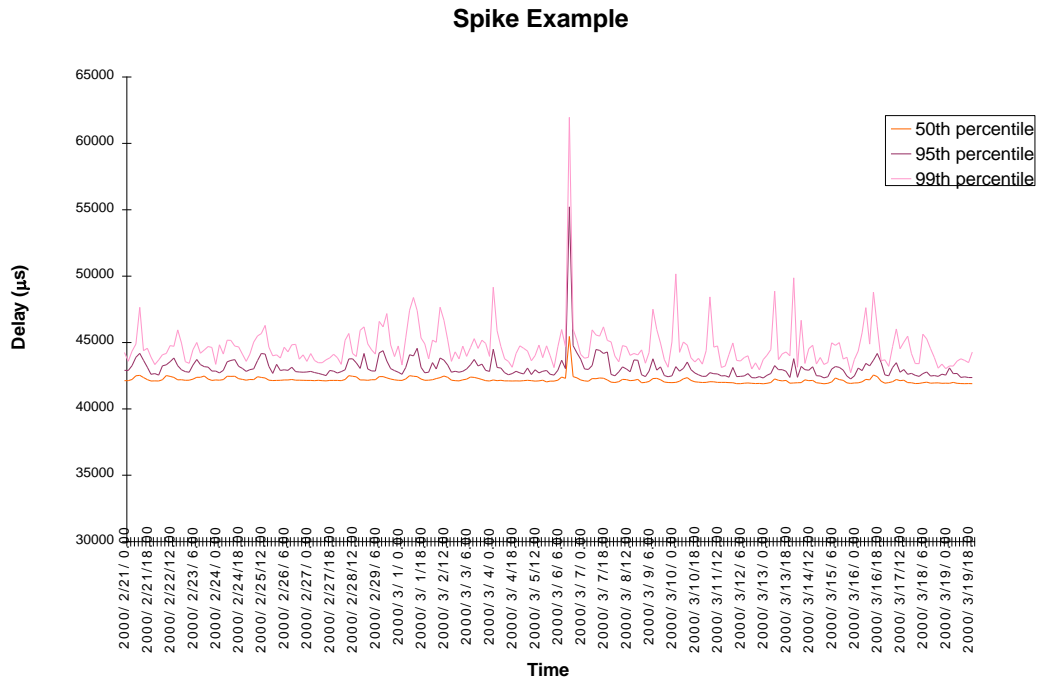


Figure 2 - Spike Exception

A Time of Day Delay Variation Exception is shown in Figure 3. The peaks seen on the 99th and 95th percentile of delay (and to a lesser extent on the 50th percentile plot) are the effects of variable network loading during a day. In the left-hand half of the graph, this delay is visibly lower than that in the right hand section. Such exceptions are caused by changes in user traffic load on a given path.

Time of Day Delay Variation Example

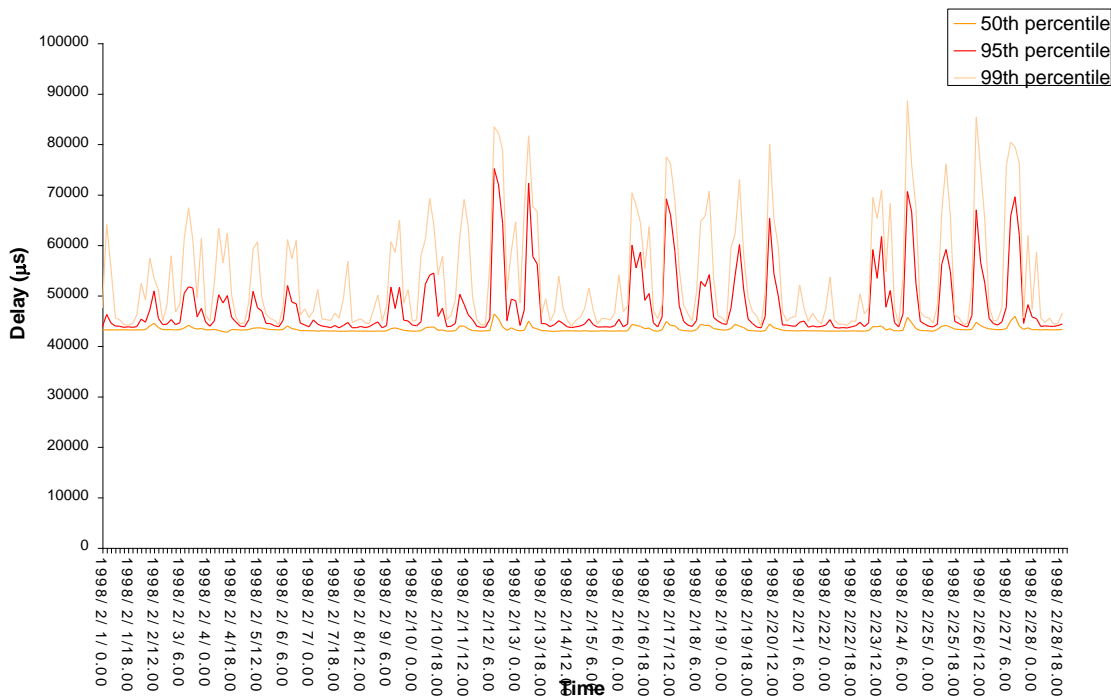


Figure 3 - Time of Day Delay Variation exception

Network Attacks

Intrusions into a communication network are a second form of anomaly which needs to be identified. This process is often termed Intrusion Detection and may be implemented via different processes, some of which are based on statistical anomaly detection. The starting point for a network intrusion is an attack. This can be carried out in a number of ways.

Phases of an Attack

Different types of attack have different life-cycles with different numbers of phases. A number of basic phases can be recognised however (some of these in practice may be combined).

Phase 1 Penetration - Getting access to the Target Machine. This is achieved in different ways. A Trojan often enters as an email; a virus via a file. Worms enter covertly by finding an “open” Port (i.e. a Transport Layer Port which is either not used but still active, or associated with an application which has a specific weakness, referred to as a vulnerability). When an automated attack is initiated from another machine, a Port Scan is conducted in this phase to identify potential vulnerabilities on the Target machine. A TCP Port Scan would send a TCP SYN segment to each Port Number in turn on the Target Machine. If a specific Port

number is active, a SYN/ACK reply will be sent back. The scanning software would then record the “open” ports for Phase 2. Port scanning can also be performed for other Transport Layer protocols; for example, UDP will often return a (ICMP) “Port Unavailable” message if a message is sent to an unopened Port.

NMAP is an example of Port Scanning tool. This is NOT malware as it was intended for network administration but can be used for Black activities. (NMAP was shown in the film, “The Matrix Reloaded”, to assist in gaining access to a computer system).

Phase 2 Exploitation – Compromising the Target Machine. A Worm may enter a Target machine via an open Port. To be effective however, the Worm then needs to compromise (i.e. establish an unofficial activity) in the Target machine. This process must occur without the knowledge or support of the legitimate user and will carry out actions which allow the attacker to gain some degree of control over the Target machine. Exploits (malware which actually compromises a Target machine) make use of some bug or design vulnerability that is some part of the Target system. Often this was in the Operating System, but can be in any software sub-system such as a Web-Browser or Database.

An example is a Buffer Overflow which is an attempt to make use of a software feature in certain programmes. This can be used if the target software does not check the size of the data which is being written into a buffer location. The malware writes a data string which is longer than the buffer size into the buffer. The target software does not check the size of this data (this is because the original programmer never considered that larger data values would ever occur) and hence some of the data spills over into adjoining memory. If this adjoining memory contains important variables used by the target software, these will be corrupted by the malware. In the (extreme!) example in Figure 4 below, the malware overwrites a legitimate destination IP address with the attacker’s IP address!. This would cause the Target programme to connect to the Attacker’s machine and further infection will then follow.

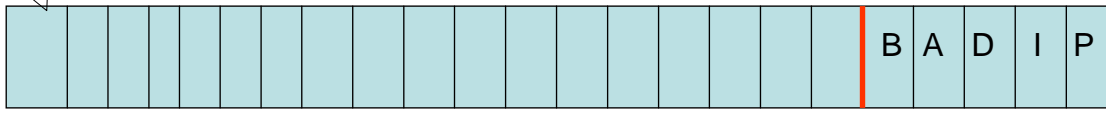
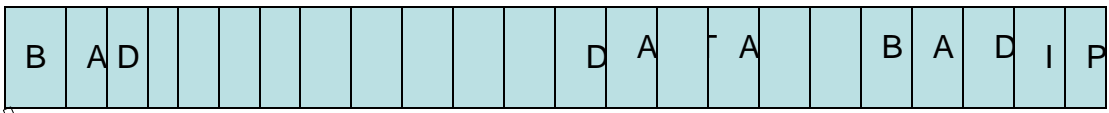
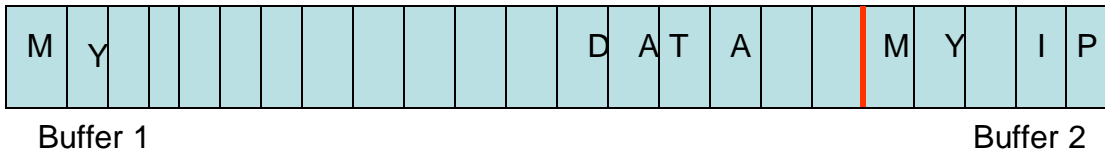


Figure 4 Example of Buffer Overflow Exploit

Phase 3 Operation – Carrying out whatever activity the controller wishes.

In many cases, Phase 3 may involve initiating attacks on other machines. In the case of a DDoS attack, Phase 3 would involve the compromised machine generating traffic at a new Target. This could be achieved by the attacker downloading further malicious software onto the first Target machine, or simply by sending instructions to this Target (now a “bot” or “zombie”) whenever the attacker wants to launch an attack.

Most Worms will start to attempt to propagate to other machines once they have compromised a given target machine.

The overall lifecycle can be seen in Figure 6. Note that the Phases do not need to follow each other immediately. In some cases, weeks or even months could separate the different phases.



Phase 1

Rule Based Detection Approaches: Snort

“Snort’s open source network-based intrusion detection system (NIDS) has the ability

to perform real-time traffic analysis and packet logging on Internet Protocol (IP) networks. Snort performs protocol analysis, content searching, and content matching. The program can also be used to detect probes or attacks, including, but not limited to, operating system fingerprinting attempts, common gateway interface, buffer overflows, server message block probes, and stealth port scans.” (Taken from Snort.org web site)

Snort can be configured in three main modes: sniffer, packet logger, and network intrusion detection. In sniffer mode, the program will read network packets and display them on the console. In packet logger mode, the program will log packets to the disk. In intrusion detection mode, the program will monitor network traffic and analyze it against a ruleset defined by the user. The program will then perform a specific action based on what has been identified.

Snort is a signature based system and is configured by the addition of Snort Rules. The following is an example of an Alert. This will generate a response if a specific action (as seen in the network packets) occurs. In this example, the action is an attempt by a non-approved user to access the root directory of the target machine as a SuperUser (su).

```
alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any (msg:"TELNET Attempted SU from wrong group"; flow: from_server,established; content:"to su root"; nocase; classtype:attempted-admin; sid:715; rev:6;)
```

Notes:

- The variable \$TELNET_SERVERS is defined in the snort.conf file and shows a list of Telnet servers.
- Port number 23 is used in the rule, which means that the rule will be applied to TCP traffic going from port 23. This rule checks only response from Telnet servers, not the requests.
- The variable \$EXTERNAL_NET is defined in the snort.conf file and shows all addresses which are outside the private network. The rule will apply to those telnet sessions which originate from outside of the private network. If someone from the internal network starts a Telnet session, the rule will not detect that traffic.
- The flow keyword is used to apply this rule only to an established connection and traffic flowing from the server.

- The content keyword shows that an alert will be generated when a packet contains “to su root”.
- The nocase keyword allows the rule to ignore case of letters while matching the content.
- The classtype keyword is used to assign a class to the rule. The attempted-admin class is defined with a default priority in classification.config file.
- The rule ID is 715.
- The rev keyword is used to show version of the rule.

Rules and Signatures will not be able to respond to new activity. To detect this, alternative approaches are needed which include:

Looking for changes (anomalies)

Considering groups of packets rather than individual or a small number of packets

Using Machine Learning Algorithms and Artificial Intelligence Algorithms.

Anomaly Based Intrusion Detection

Researchers in the field of intrusion detection are mainly focused on anomaly IDS. The anomaly IDSs create statistical references of normal behaviour of the protected systems by monitoring attributes of previous events, for a given period of time. Any deviation from the normal behaviour of the monitored systems is interpreted by the anomaly IDSs as the presence of attacks or intrusion attempt. This approach works based on the assumption that the normal and abnormal behaviours of a system are differentiable from each other. The difference between the normal behaviour of the protected systems and the anomalous behaviour must be statistically differentiable, and the difference must be quantifiable. This condition should be assumed, otherwise, if the characteristics of both behaviours were completely similar, any effort to identify the attacks would be impractical. Also, for the correct implementation of the anomaly IDSs, the number of legal instances in the analysed dataset must be larger than the malicious instances. These two conditions are similar to the conditions that need to be assumed for the unsupervised IDSs.

The process of creating the statistical reference of normal behaviour is commonly considered the training process or training phase. The training process starts with the monitoring sensors gathering information from the protected system. Then, the anomaly IDS analyses this information to create a statistical reference of normality. Numerous techniques can be employed to generate this reference. Once the training process has finished and the reference of normal behaviour has been generated, the anomaly IDS calculates the level of deviation of the currently analysed information from the reference of normality. High level of deviation between both values is likely to correspond to evidence of intrusion. But the difficulty here is to define the boundary between normal and malicious information. As part of the training process, anomaly detection systems also have to define an alarm threshold. This alarm threshold defines the boundary between normal and malicious actions. If the deviation exceeds the defined alarm threshold, the system concludes that an attack is taking place and raises an alarm. In fact, the generation of a precise alarm threshold is one of the most complicated, and the most crucial tasks for this type of IDSs.

Detecting Anomalies in Network Traffic

The Kolmogorov-Smirnov (KS) test statistic can be used initially to establish that there has been an anomaly in the data; and a neural network can then be used to classify any detected change.

The KS statistic is the best known of several distribution-free procedures which compare two sample CDFs (Cumulative Distribution Functions) in order to test for general differences between two distribution. The KS test compares two sample CDFs using the maximum vertical distance between them as a test statistic. A normalised example of this is shown in Figure 6. Any kind of substantial difference between the two distributions will be indicated by a significantly large difference between the sample CDFs. Such differences may be in location, spread or may be more general differences in the shape of the distributions.

KS Statistic

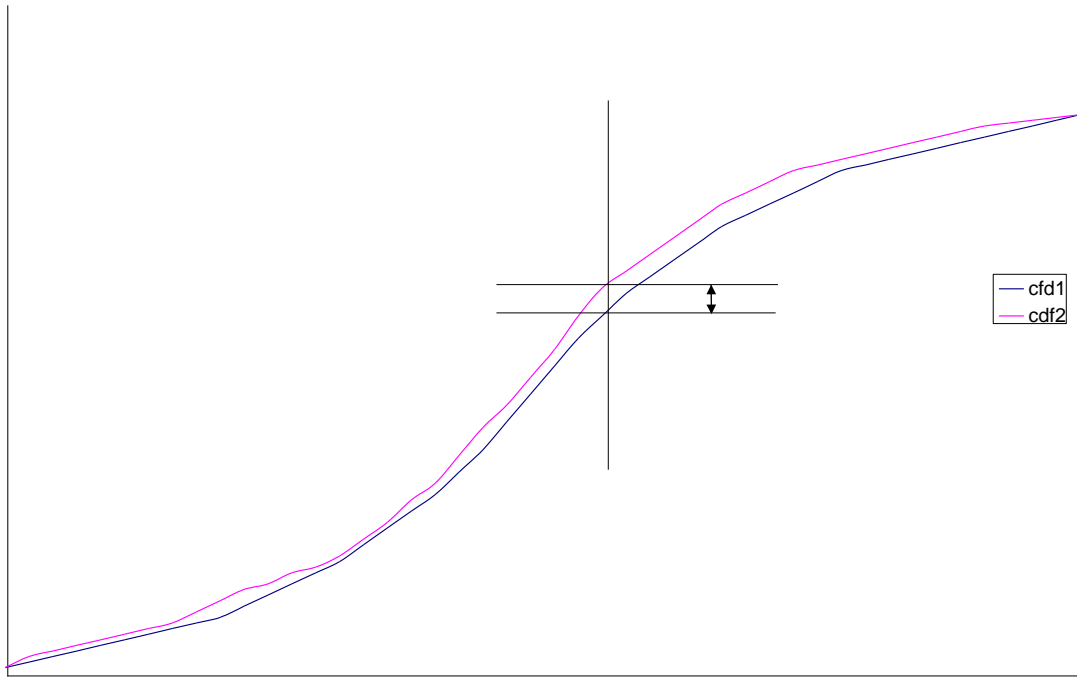


Figure 6 - The KS Statistic

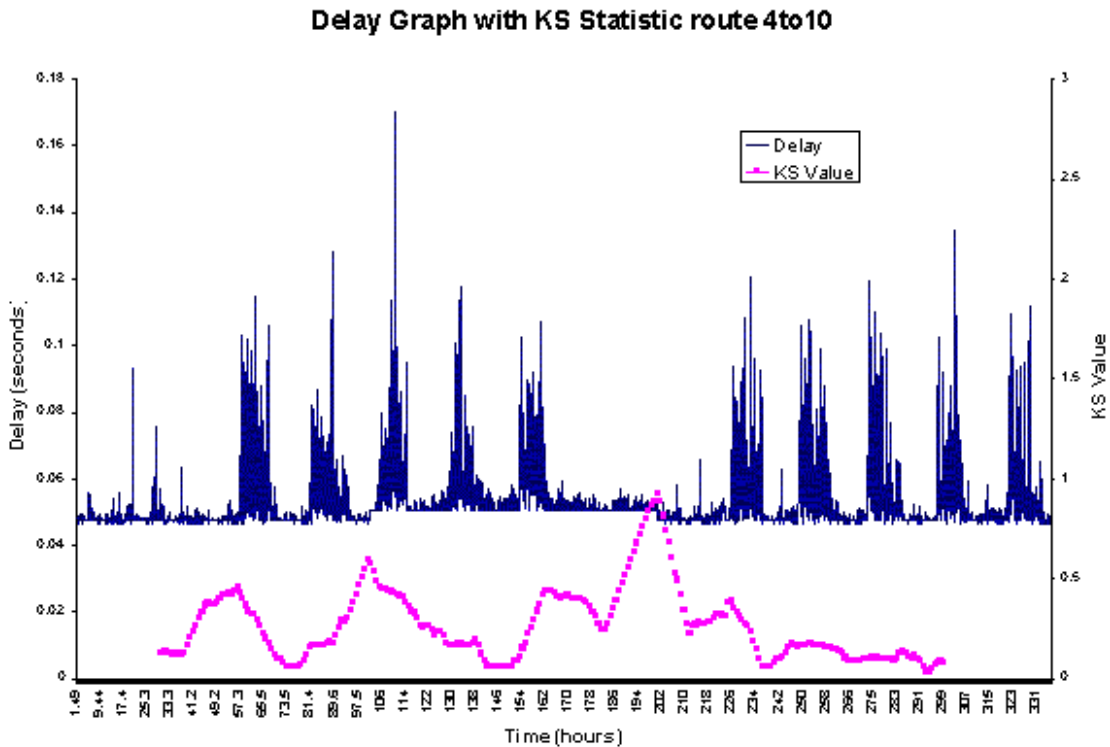


Figure 7 -The KS Test applied to delay data (1).

The scale on the primary y-axis in Figure 7 above refers to delay while the scale on the secondary y-axis (right hand side) refers to the KS statistic (which will always return a value between 0 and 1). The x-axis represents the number of hours from the beginning of the data.

Data Mining

Data Mining is a technique which allows the natural relationships between data to be determined and explored. It can be supervised, whereby some prior knowledge of the data characteristics is used to guide the process, or unsupervised in which the tool itself will attempt to discover relationships in the data which may or may not be relevant. As with all these techniques, care must be exercised to ensure that the relationships are sensible. For example, if such a tool was given a set of data where all the attacks just happened to occur at night, it would probably find a relationship between time of day and attack which would be erroneous when more generally applied. Research conducted at Loughborough has indicated that it is better to present packet data in a summary or metadata form for DataMining to reduce the data volume; to increase the information in each data element and to reduce the amount of erroneous information.

In addition to ANNs, Data Mining tools may include:

Genetic algorithms: Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of natural evolution.

- Decision trees: Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic Interaction Detection (CHAID) . CART and CHAID are decision tree techniques used for classification of a dataset. They provide a set of rules that you can apply to a new (unclassified) dataset to predict which records will have a given outcome. CART segments a dataset by creating 2-way splits while CHAID segments using chi square tests to create multi-way splits. CART typically requires less data preparation than CHAID.
- Nearest neighbour method: A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset (where $k \geq 1$). Sometimes called the k-nearest neighbor technique.
- Rule induction: The extraction of useful if-then rules from data based on statistical significance.
- Data visualization: The visual interpretation of complex relationships in multidimensional data. Graphics tools are used to illustrate data relationships.

Performance Evaluation

The efficiency of IDSs could be evaluated using multiple parameters, such as the amount of resources (CPU, Memory, etc.) the system consumes, or the required time to conduct the detection. Nonetheless, the most important aspect to evaluate the effectiveness of the proposed intrusion detection methodology is its ability to make correct predictions [58]. This is achieved using a series of evaluation functions over the generated system outcome. These evaluation functions have been widely used among the researching community in the field of IDSs. Before listing the functions, it is also necessary to define four evaluation parameters utilised in the functions.

These are:

- True Positive (*TP*) refers to one attack frame that has been correctly classified as malicious.
- True Negative (*TN*) refers to one non-attack frame that has been correctly classified as legal frames.
- False Positive (*FP*) refers to one non-attack frame that has been misclassified as malicious.
- False Negative (*FN*) refers to one attack frame that has been misclassified as legal frames.

These parameters provide quantifiable evidence of how effective are the IDSs at making correct detections. All these parameters are utilised to calculate the following evaluation functions:

- Detection Rate (*DR*) is the proportion of attack frames correctly classified as malicious, among all the attack frames.

$$DR (\%) = \frac{TP}{FN + TP}$$

- False Positive Rate (*FP_{Rates}*) is the proportion of non-attack frames misclassified as malicious, among all the evaluated frames.

$$FP_{Rates} (\%) = \frac{FP}{TP + FP + TN + FN}$$

- False Negative Rate (FN_{Rate}) is the proportion of attack frames misclassified as legal, among all the attack frames.

$$FN_{Rate} (\%) = \frac{FN}{TP + FN}$$

- Overall Success Rate (OSR) or *Accuracy* is the proportion of the total number of frames correctly classified, among all the evaluated frames.

$$OSR (\%) = \frac{TN + TP}{TP + FP + TN + FN}$$

- *Precision* or *Recall* is the proportion of attack frames correctly classified as malicious, among all the alarms generated.

$$Precision (\%) = \frac{TP}{TP + FP}$$

- *F-Score* or *F-Measure* is a tradeoff between Precision and DR. The *F-Score* produces a high result when *Precision* and DR are both balanced [33] [97]. The higher the *F-Score*, the better the *Precision* and the DR.

$$F-Score (\%) = \frac{2 * Precision * DR}{Precision + DR}$$

The most important action for the IDSs is to generate the maximum number of *DR* and not generating any false alarm. Generating either *FN* or *FP* is not desirable. Both situations show decrease of the effectiveness of an IDS detecting intrusions. However, cost of generating *FNs* is often higher than the cost of *FPs* [80]. An IDS that generates too many *FPs* works against legitimate communications. However, the protected system is not actually compromised by any threat. The administrator of the IDS might ignore the raised attacks alerts [78]. On the other hand, every single *FN* is an attack that has gone undetected and reached the protected system.