1. Introduction

With the heavy spread use of the Internet for all kinds of purposes during the last 15 years, there was an increase in the number of attempts by individuals who managed to break into these systems, either as a learning experience in the field of computer security (hackers) or as a way of having fun by destroying these systems (crackers).

The primer build their own networks and test them to the breaking point, the latter use the ones publicly available, which of course is the problem as the publicly available ones are owned by businesses and government.

The problem with standard computer security mechanisms, though, is their approach to dealing with security, by way of using standard methods to lock down and secure the systems they are implemented on.

Take firewalls, for instance. They are built to keep people out of the system but they do nothing for people who bypass them (because they cannot detect them) and they use standard ways of restricting access, ones which an imaginative technocratic person can circumvent with mostly very little trouble, given time. Nor do they protect the network from the people inside it who may want to do things the company/organization's policy does not allow them to do.

As this was realized, solutions had to be found to deal with the matter, or at the very least minimize the damage after the attack is performed. Intrusion Detection Systems are one of the ways the Computer Industry has devised to help towards that result.
2. History of Intrusion Detection Systems

The first work on Intrusion Detection was published in 1972 by J.P. Anderson, for the US Air Force. In his first paper, Anderson publicly stated the fact that computer systems vulnerabilities and intrusions have started spreading and becoming more dangerous.

In 1980, Anderson published a study detailing some mechanisms and ideas on how to improve security and detection of intrusions. One of those ideas was a method for the detection of anomalous events by studying the audit data the Operating Systems were producing.

It was his belief that intruders could be detected by monitoring log files for failed login attempts, failed attempts to access programs and documents and other resources, as well as recording the normal behavior of users in the systems and monitoring for any deviation in them.

In the years between 1984 and 1986, Dorothy Denning and Peter Neumann created the Intrusion Detection Expert System (IDES) as a first model for a real-time intrusion detection system. IDES started out as an expert system, monitoring known intruder activities, but was further refined to provide a system independent audit record format, metric and statistical models creating profiles of users and resources, an engine to correlate the data based on the user’s profile, anomaly records and activity rules. Thus the IDS known as Next-generation Intrusion Detection Expert System (NIDES) was formed.

Later work in the field of Intrusion Detection, in the years ranging from 1985 to the late 1990s involved work that both verified Anderson’s thoughts on anomaly detection as well as Denning’s and Neumann’s work, such as the IDS created by Sytek that proved Anderson’s theorems by creating a simple IDS based on anomaly detection and proving that it works. Those years also saw the creation
of the very well known Multics Intrusion Detection and Alerting System (MIDAS).

In the late 1990s (1995 to 1998), two of the most popular IDS were created, Wheelgroup's NetRanger and ISS's RealSecure,

NetRanger was the first complete IDS that used signature-matching algorithms (known also as Misuse Detection algorithms) to detect intrusions in real time.

Internet Security System’s RealSecure IDS was first announced as a tool to augment network security with attack recognition in real-time mode and later was ported to Windows NT 4.0 as version 1.0, making it a commercial success and a breakthrough.
3. Defining Intrusion Detection

First of all, we have to define Intrusion, in relation to Computer Systems.

An intrusion, therefore, is an attempt to gain unauthorized access to a system with the purpose of simply to test the security of the network, use the facility as a launching pad for further attacks on other systems, to modify information or to steal information, etc.

Intrusion is committed by gaining initial access to a particular host by discovering a password for a user account on the system. The intruders will then attempt to gain access to the compromised system. Intruders are actually committing the following activities:

1. Sniffer Attacks- capturing data as it traverses the net.
2. E-mail attacks- gaining system access through vulnerabilities in network service software.
3. Network File System attacks- gaining data access through vulnerabilities in operating system software.
4. Network Infrastructure attacks- denial of service through attacks on routers and name servers, i.e. for purpose of impersonating the server.
5. IP spoofing attacks- gaining system access by tunneling through firewalls.
6. WWW threats- gaining users or system information through the web or cgi programs.

So, following in the same path, Intrusion Detection can be defined as the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions.

Up until some time ago, the best defense of a company that actually had any security was firewalling its primary servers using either a hardware firewall or a
packet filter, thus controlling access to its intranet and preventing intruders from being able to access the intranet.

Firewalls, though, do not detect if an intruder is trying to connect to a company's network via that company's dialup modems, or if they are trying to guess passwords. Furthermore, firewalls cannot see (or even care) what a person who has bypassed it is doing to the system, as it is not within their responsibility to do so. If they manage to circumvent it, it gives the company no protection and it presents the attacker with no more problems.

This is where Intrusion Detection and Intrusion Detection Systems come in, both aiding the firewalls and complementing them.

An Intrusion Detection System is a security system that monitors computer systems and network traffic and analyses that traffic for possible hostile attacks (intrusions) originating from outside the organization and also for system misuse or attacks (intrusions again) originating from inside the organization.

An IDS comprises of a Management Console, which is used to control the IDS and monitor all the alerts and responses, and one or more (depending on the type of IDS used and/or the number of hosts it monitors) Sensors, which are agents that monitor hosts or networks on a real-time basis. It also may or may not contain a database of attack signatures, the latter being patterns of different types of previously detected attacks.

When the sensor detects any malicious activity which also matches a signature in the database, it reports that activity to the management console and (depending on how it is configured) it can take different actions to protect the system from being compromised or enhance the logging so as to have better evidence of what has happened or do both and also inform (by e-mail, pager, sms, message on the console, etc.) the administrator of the events taking place.
To elaborate on the previous statement on collaboration between firewalls and IDS, the case is as follows: The firewall protects the organization from malicious attacks from the internet and the IDS detects if someone tries to break in through the firewall or manages to break in the firewall security and tries to have access on any system in the trusted side and alerts they stem administrator in case there is a breach in security.

Of course, this collaboration is but one of its uses, which will be further defined and explained at a later time.
4. Architecture of Intrusion Detection Systems

The architecture of IDS is basically the way they are placed in the overall structure of the network. There are currently 4 types of architecture IDS can exist in. Monolithic, which is the oldest one, Hierarchical, Agent-based and Distributed.

4.1 Monolithic Architecture

The oldest and most simple in logic architecture, where the IDS is a single application containing the sensor and the monitor, focusing on only one host and not on the entire network of hosts.

They are conceptually simple and their implementation is easy. However they are unable to cope with attacks that are acted against multiple hosts and are very subtle in the execution, as there is no intra-network communication of IDS sensors.

4.2 Hierarchical Architecture

In this architecture, there are some monitors and sensors in all the systems in the network that collect the data, which end up in a single controlling IDS monitoring and collecting system.

This results in the ability of the IDS to correlate data gathered from all the other IDS sensors in the network, resulting in the identification of distributed attacks.

4.3 Agent-based Architecture

Here, the sensors, the monitors and the controllers are spread all through the network, allowing for greater placement flexibility. Sensors still report to monitors
and monitors may or may not report to the controller of the system.

This allows for the survival of the IDS from overloads of attacks and makes the IDS harder to detect, as it is spread through many systems.

### 4.4 Distributed Architecture

The most recently developed architecture, viewing the entire system as a single unit, and attacks modeled as interconnection patterns between systems. Each link represents network activity, allowing the graphs of the interconnections to be viewed at different scales.

This architecture allows for high scalability and for better detection of distributed attacks and worms.
5. Types of Intrusion Detection Systems

Some of the IDS work by sitting on a network and analyzing all the network packets that go through it. Other IDS work by analyzing the logging information the computers in the network gather, basically by analyzing the log files of just about everything, from web-services to system logging (syslog) services.

According to the sources of the information they gather, Intrusion Detection systems can be grouped into two types, Host-Based and Network-Based.

5.1 Host-Based IDS (HIDS)

A Host-Based intrusion detection system is the oldest approach the security field took in the direction of Intrusion Detection.

It basically collects information from each of the computer systems. The information they gather reflects the activity that occurs on that particular system.

It gets that information from either operating system audit trails and/or system logs. The primer are usually created at the kernel level and have the advantage of being more detailed and better protected than the latter. System logs, though, are much more easily readable and much smaller than audit trails. Other sources of information for this type of IDS are other logs generated by processes within the operating system and also contents of other objects within the system that are not included in the standard logging mechanisms and audit trails. These objects can be history files, application-related logging files etc.

Therefore, when the intruder attempts to access a monitored host, what they make and touch are watched, made a list of and if they fit into any of the rules the IDS has for unusual events or events that are not fitting of the user who is performing them, then IDS sounds the alarm.
This approach offers some advantages against Network IDS. They can detect attacks that a NIDS is unable to see, as they have the ability to monitor events local to a system. They can work in systems where network traffic is encrypted, as long as that encryption takes place after the system information sources are created and/or after the information is decrypted at the destination system. In environments where a switched network is present they are not affected, and they can help detect attacks involving some sort of software integrity breach, such as in the case of a Trojan Horse.

Using a HIDS gives the system the ability to monitor who accessed what; therefore the system is able to map erroneous activities to a specific user id. They also give the system the chance to track behavioral changes that serve as a pre-amble to a misuse and can distribute the load of information of the monitoring process across many hosts in the network and thus cut down on performance costs.

Finally, they are more easily made to protect the system without jeopardizing the privacy of the individuals who use the systems in the network, as no personal information concerning those individuals is collected. Also, A HIDS can verify the success or failure of an attack with greater accuracy and fewer false positives.

However, they do possess some disadvantages over NIDS. For one thing they are harder to manage, as every host to be monitored needs to be configured properly. They are unable to detect network scans across the network, as they only detect scans against the specific host. If a breach does occur, the system log files are going to be one of the intruder's primary objectives for alteration or inactivation, so there is high integrity risk, and of course, some Denial of Service attacks can disable the IDS altogether.

The amount of information a HIDS receives can be quite big under normal
circumstances, not to mention if an attack does occur on the network, therefore a lot of space needs to be available for that information, and there is also the matter of performance straining the process of gathering and sorting through that information will inflict on the system.

5.2 Network-Based Intrusion Detection Systems (NIDS)

NIDS detect attacks by capturing and analyzing network packets. They can passively monitor the network traffic affecting multiple hosts that are connected to the network, listening on a network segment or switch. Once analyzed, the sessions are reconstructed, thus recognising patterns of attack.

They often consist of a set of single-purpose sensors or hosts placed at different points of the network, with their network interfaces set to a promiscuous mode and using some sort of a packet sniffer. Many of these sensors are designed with stealth in mind and without interfering with the normal network activity, so as to make finding their presence and location more difficult by the intruder.

NIDS usage in a network offers some advantages over the use of HIDS. First of all, a few well-placed NIDS sensors can monitor very broad networks at a stretch, without interfering with the operation of the network, which makes them easy to install.

They are good at sniffing out attacks that involve low-level manipulation of the network and can easily correlate attacks against multiple machines on a network, therefore they can monitor network attacks such as SYN floods and Denial of Service attacks.

NIDS can also be made very secure and even invisible to the intruder and they do not have any requirements for auditing or logging mechanisms. Also, their monitoring of the system can be in real time, thus affording for a quick response
as it detects attacks the moment they occur.

However, they do have certain disadvantages over Host-Based Intrusion Detection Systems. They may have difficulty processing all the packets in a large network and in periods of higher activity such as rush hours may fail to recognize an attack. They also have big problems operating behind switched networks, as the latter divide the network into many small and dedicated network segments, and cannot analyse encrypted information, which is a big problem in a Virtual Private Network scenario.

They cannot also tell if an attack has been successful, so it is left to the administrator to determine if the system has been compromised on each of the attacked hosts. They also have problems with attacks that involve fragmented packages, which cause the IDS to become unstable and sometimes crash. Finally, they cannot see what commands have been executed (making it impossible to reconstruct the attack). Also, their real-time monitoring functionality becomes a problem in high-speed networks, as in that environment the IDS has to process a very big amount of traffic at very short notice, leading to the IDS getting clogged up.
6. The Process of Analysis used in Intrusion Detection Systems

In order to detect attacks, and after it collects the information it requires, an IDS has to analyse that information. There are basically two types of analysis IDS use, Anomaly Detection and Misuse detection, and there are many approaches to each of those types to analyse data, which this section is going to describe and analyse.

6.1 Anomaly Detection in Intrusion Detection Systems

Anomaly detectors identify abnormal behavior on a host or network. They work under the premise that all attacks differ from the normal activity and can be traced if a system that identifies these differences is introduced. They also try do detect the complement of abnormal behavior.

These tools create profiles normal behavior of users, hosts and networks from historical data over a period of normal operation of the system/network. Once the training period ends, those detectors collect all available data off the system and check them to find deviations from the norms. If they do find such evidence of deviations they inform the administrator.

Behavioral records include the number of files accessed by a user at a given period of time, the amount of CPU utilized by a process a user started at a given time, the number of failed logins a user has made in order to enter the system etc. The level of such observations can be either static or heuristic.

So, therefore, having described how the Anomaly Detection process works, we can now describe some of the approaches to AD research has yielded.
6.1.1 Statistical Approaches

These methods initially generate behavior profiles for users, hosts and networks, with variances of each profile being generated continuously as the system continues running. Detection of any anomalous behavior can be achieved by monitoring for behavior patterns that deviate from those defined in the user profile.

They can be either parametric, where the distribution of the profiled attributes is assumed to fit a particular pattern, and non-parametric, where the distribution of the profiled attributes is taught from a set of values having been observed over time.

The modeling of the behavior patterns in a statistical approach can generate the following counters to be used for detecting deviation patterns:

- **Threshold measures**, in which heuristic limits are applied to events and the number of times they occur on the system.
- **Mean and standard deviation**, where comparing the deviations to the events measures produces a confidence interval for abnormal behavior.
- **Multivariate model**, which is calculated by the correlation between multiple events and the profile expectations based on historical data on the user.
- **Markov process model**, where an event is considered to be anomalous if its probability is too low.
- **Clustering analysis**, a non-parametric method relying on representing streams of events in a vector representation, the examples of which will then be grouped into classes of behaviors.

The advantage of a statistical approach is that the system adaptively learns the behavior of users and is therefore potentially more sensitive.
However, the problem with this approach is an intruder, given a period of time, can teach them to consider their actions as within normal behavior, creating false positives and false negatives. There is also the problem of selecting which measures to follow exactly, as it is not known what the subset of all possible measures that predict intrusions is.

6.1.2 Predictive Pattern Generation

This method tries to predict future events based on the events that have already occurred.

It has certain advantages, which for one thing involve the ability to detect anomalous activities difficult to spot using traditional methods. Also, systems that were built using this method adapt to changes very easily and it is much easier to detect intruders who are trying to train the system during the training period. Finally, Predictive Pattern Generation systems can detect and report anomalous activities within seconds of receiving audit events.

However, there are some problems inherent in this method, such as that some intrusion scenarios that are not described by the rules are therefore not flagged as intrusive. Solutions to this problem will create either a precedent for too many false positives or too many false negatives.

6.1.3 Threshold Models

In these models, a set of static or heuristic limits is applied to instances of events (they being certain attributes of user and system behavior) or to the number of counted events over a predetermined interval.

Most of the commercial IDS use either this method or the statistical one in their anomaly detection techniques.
6.1.4 Rule-based measures

Similar to non-parametric statistical measures, they count on observed data to define acceptable usage patterns. However, they differ as to how those patterns are specified, with the Rule-based measures specifying patterns as rules instead of numeric values.

6.1.5 Other less frequently used methods

Other methods which can be used for Anomaly Detection, but which are less frequently used are Immune systems approach, Protocol Verification, File Checking, Taint Checking and Whitelisting.

Immune systems approach models applications in terms of sequence of system calls for a variety of different conditions, such as normal, error and attempted-exploitation conditions. Anomalous events are detected by the comparison of this model to observed event traces.

This approach has shown the ability to detect a number of typical attack techniques but cannot detect race condition-based attacks or policy violation or masquerading ones.

In Protocol Verification, detailed checks are conducted against protocol fields and behavior against established standards or heuristic expectations, and data that violates those standards/expectations is tagged as anomalous.

It can detect many standard attacks but the main problem is that there is no strict compliance between protocol standards and protocol implementations.

File checking uses cryptographic checksums in sensitive data to detect changes. The MD5 algorithm used in newer distributions of Linux and the MD4 algorithm used in old ones is the most typical example of such an approach.

However, the technique is useless if the system that does the checking is
compromised or bypassed.

Taint checking is an application-centered approach, creating risk-aware applications that consider data provided by the user as tainted and as such must and cannot be used in any sensitive context without an alarm sounding.

Finally, Whitelisting is basically a data-reduction technique that involves passing a raw event stream through a number of filters each of which corresponds to some benign pattern. Anything that remains out of this filtering process is deemed as anomalous.

### 6.2 Misuse Detection Systems

In Misuse Detection, the system analyses system activity, looking for events or sets of events that match a predefined pattern of events that describe a known attack. Those patterns are also called signatures, so the entire process is also called signature-based detection. Its most common form specifies each pattern of events corresponding to an attack as a separate signature.

There are many approaches to Misuse Detection, such as the use of Expert Systems, Pattern Matching, State transition analysis, Expression Matching, Dedicated languages and Keystroke Monitoring.

#### 6.2.1 Expert Systems

Expert systems are modeled in such a way as to separate the rule-matching phase from the action phase. The matching is done according to audit trail events.

The system encodes known intrusion scenarios and attack patterns.
However, the expert system has to be formulated by a security professional, therefore making the system only as strong as the security personnel who programs it. Furthermore, additions and deletions of rules from the rule-base must take into account the dependencies between different rules in the rule-base. Also, there is no recognition of the sequential ordering of data, because the various conditions that make up a rule are not recognized to be ordered.

### 6.2.2 Pattern Matching

This model encodes known intrusion signatures as patterns that are then matched against the audit data. Like the state transition analysis model, this model attempts to match incoming events to the patterns representing intrusion scenarios.

An advantage of this model is that only the patterns that need to be matched need to be specified, not the way to match them. The streams can be processed independently and their results can be analyzed together to give evidence of intruder activity. The signatures can be ported to other operating systems, as they are universal. The system has excellent capabilities for real-time monitoring and detection.

However, it can only detect attacks based on known vulnerabilities, which makes the method vulnerable to newly discovered attack scenarios. It is also not very useful for representing patterns that are not well defined, as it is not easy for attack scenarios to be translated into patterns to be used by this model. Finally, it cannot detect passive wire-tapping intrusions nor can it detect IP spoofing attacks.

### 6.2.3 State Transition Analysis

In this method, the monitored system is represented as a state transition
diagram. When data is analyzed the system makes transitions from one state to
another, with each transition taking place according to the truth or falsity of a
Boolean condition.

For the final compromised state to be reached, some conditions have to be
fulfilled, those conditions being called guard conditions. If they are true, then
there is almost certainly an intrusion being going on. If any of these conditions do
not hold, the possibility of intrusion is considerably less certain. Therefore, this
can serve as a data pruning mechanism.

One of the advantages of this model is that it can detect co-operative attacks and
ones that span across multiple user sessions. It can also foresee impending
compromise of the system, based on the present system state, enabling it to take
measures to deal with the compromise.

The problem, though, is that attack patterns cannot specify complex sequences
of events and there are no general-purpose methods to prune the search. Finally,
they are unable to detect Denial of Service attacks or failed logins or variations
from normal usage or passive listening attacks, as these attacks are not recorded
by the auditing mechanisms and cannot be represented by state transition
diagrams.

6.2.4 Expression Matching

It is the simplest form of intrusion detection, as it searches the audit trail using a
list of known signatures to find instances of known patterns.

The resulting expression, though, can very easily become hard to read and
representing context is somewhat difficult. However, signatures can be quite
easy to construct.
6.2.5 Dedicated languages

Intrusion detection signatures can also be represented by the use of specialized languages. The generality of such languages can vary, but they all offer great flexibility in matching attack scenarios.

Thus, a signature becomes a specialized program, with raw data as input. Any of the input triggering a filtering program or matching internal alert conditions is detected as an attack.

Those signatures, though, may require a very good understanding of the protocols, packets and attacks involved as well as some programming knowledge. Also, attacks that use variations of the signature attacks can probably bypass this type of filtering.

6.2.6 Keystroke Monitoring

A very simple technique that basically logs all keystrokes, monitoring those logs for any patterns of commands that would indicate a pending attack. However, this is most difficult in operating systems with many command shells, which accept user-defined aliases for commands. A malicious program cannot be flagged to be of any danger to the system as its running is not monitored. Also, operating systems do not by themselves have a lot of facilities for keystroke capturing, although in the last two years an attempt has been performed (with various types of success) to introduce such a facility as a Linux Kernel Module.

6.3 New methods developed in the last few years

As new ways of data analysis and mining were developed during the last few years, a use of their functionality was added in Intrusion Detection Systems research. Methods for both Anomaly Detection and Misuse Detection were found
by using research made in other fields of computer science, such as Artificial Intelligence.

Some of the techniques to be discussed in this section are going to be the use of Neural Networks, Genetic Algorithms, Probabilistic Networks (Bayesian Belief Networks) and Protocol Anomaly Detection.

### 6.3.1 Neural Networks

The ability of neural networks to cope with lots of imprecise and noisy data and their ease of modification for new users entering the system was a decisive factor for a research into using them to perform anomaly detection.

So, the neural network is trained to predict a user's next action or command, given his history of previous actions and commands. The network is trained on a set of representative user commands. After the training period, the network tried to match actual commands with the actual user profile already present in the net. All incorrectly predicted events actually measure the deviation of the user from the established profile.

The problems, however, with neural networks in intrusion detection are first of all the margins, which if big will result in irrelevant data and increase the rate of false positives, and if small will result in false positives. There is also going to be a big period of trial and error so as for the neural network to determine the network topology of the system. Finally, the intruder can train the net when still in learning phase.

### 6.3.2 Genetic Programming and Intrusion Detection

Genetic Programming is used in IDS for its learning capability. It allows programs to be evolved, which could be then executed in a production environment.
The final results of an evolution run is a set of programs, coded in a simple meta-language tailored to solving a specific problem, which are placed in a real system and are run continuously to detect intrusions.

In a single-system solution those programs are interpreted by an evaluator, which supplies them with an agent in the final system. The evaluator contains audit information from the System Abstraction Layer (which computes various statistics from the audit records and supplies them to the agents).

The agents monitor and detect very simple intrusions. Their code is composed of a set of operators and a set of primitives, the latter being things like average CPU utilization, average number of login attempts by a user, etc. This set of agents is assigned a fitness score during the test scenario phase. If an agent fails to report a high suspicion of a known intrusion, the agent will be given a low fitness score. In the end of the training, a penalty is assigned to the sensors, based on how this scenario is ranked.

Once agents are evolved in the training system, they must be stripped down so that they can be placed in a production system and impose a minimum overhead.

However, the training scenarios have to be well-developed and well-thought out, as well as to cover a big range of intrusions.

6.3.3 Bayesian Probabilistic Networks

Traditionally, Bayesian networks are made up of a set of variables and a set of directed links connecting pairs of nodes. Each node has a number of states and a conditional probabilistic table that describes the probabilistic distribution of the states for the corresponding variable given the states of its parent nodes.
In the attack detection, what is of interest is how likely various audit events correlate in a moving window during normal activities.

During testing, the support probability of Bayesian network is computed on the evidence of audit events in each moving window of the testing data, using an inference algorithm. The smaller the support probability, the more likely that audit events in the moving window come from intrusion.

### 6.3.4 Protocol Anomaly Detection

Instead of training models on normal behavior, protocol anomaly detectors build models of TCP/IP protocols according to their related RFCs (Request For Comments). As all protocols have a state, some PAD tools are built as state machines. The transitions between the states describe the legal and expected changes between states. Thus, by building a model based on the normal state of the protocols it improves on signature detection and eliminates the need for frequent signature updates, providing enhanced performance in the process.

In plain terms, what a Genetic Algorithm does is searches for the combination of known attacks that best matches the observed event stream. The hypothesis vector is evaluated based on the risk associated with the attacks involved. In each cycle the current set of best hypotheses are mutated and retested so that the probability of false positives and negatives approaches or even becomes zero.

This approach offers good performance, but does not help discover the reason the attack was detected. Furthermore, it does not detect attacks that fall in the accepted norms of the protocol states, such as viruses.
6.4 Advantages and Disadvantages of Intrusion Detection Methods

This section is going to analyse the advantages and disadvantages of Anomaly Detection and Misuse Detection.

6.4.1 Advantages of Anomaly Detection

One advantage of Anomaly Detection in an Intrusion Detection System is that the system has the ability to detect symptoms of attacks without specific knowledge of the details. Also, anomaly detectors can produce information that can then be used to define the signatures for misuse detectors. Finally, anomaly detection can lead to the detection of unknown attacks and offers low overhead to the process of ID.

6.4.2 Advantages of Misuse Detection

Misuse Detection IDS do not create huge amounts of false alarms when detecting attacks and can very quickly find out what tool/attack/technique was used in the intrusion. They are also very easy for even security illiterate people to use to diagnose security problems on their system.

6.4.3 Disadvantages of Anomaly Detection

The results produced by an Anomaly Detection IDS usually yield large numbers of false alarms, as the prediction of users' actions is difficult, and they also require extensive training sets of audit trails so as to understand what the normal patterns of system activity are.

That leads us to understanding that the better the definition of “normal” the better
the Anomaly Detection method will work.

Also, they are not going to be able to give out data with any certainty. They will tell that something happened but not exactly what did happen.

6.4.4 Disadvantages of Misuse Detection

Misuse Detection-based IDS will recognize only the attacks it knows about (has in its signature database) therefore it has to be updated regularly with new signatures so as to be able to identify the intrusions.

That means it will not also be able to identify new attacks when faced with them.

As the signatures are strictly defined in their description, the system will not have the flexibility required to identify variations of the attacks whose signatures it has in its database.
7. Response Methods of an IDS

IDS can also have methods by which it responds to detected attacks aimed against the network. This section will examine some of those methods of response.

7.1 Attack tracing

The system tries to gather information (either passively or actively) so as to identify the source of the attack.

7.2 Shunning

The system configures some other module or system such as the firewall to add rules to block the IP of the attacker and close the connection they have to the system.

7.3 Extended information gathering

The system increases the level of information the system gathers concerning the attack.
8. Strengths and Weaknesses of Intrusion Detection Systems

IDS are neither the panacea of all security attack, nor a totally wasted field of research. They do work, and they do bring results and they do their job. Granted, the results are mixed and the implementation requires both consideration and a great deal of thought in the designing and planning stage before deployment.

This section details their strengths and weaknesses so as to gain a better perspective as to who needs IDS and what for.

8.1 Strengths of Intrusion Detection Systems

Intrusion Detection Systems can help the administrators establish a better policy for the network, and can detect errors in the system configuration that would otherwise be hard to see, thus adding a greater degree of integrity to the rest of the network's infrastructure.

They can also trace user activity from the entry to the impact on the system and can give information about the data the intruder changed/attempted to change. Also, they can automate the task of staying current with attack methods and can detect when your system is under attack, allowing at the same time the security management of the network by non-security-expert staff.

8.2 Weaknesses of Intrusion Detection Systems

There are also tasks and areas where the use of IDS is not suitable.

For instance, they cannot compensate for weak network security infrastructure,
weak identification and authentication of the users. They require human presence for the investigation of attacks and cannot analyse all the traffic on the network, especially if it is busy and during rush hours, when it is busy.

It cannot also do anything about the inherent weaknesses protocols and quality of system information sources, and cannot deal with very detailed packet level attacks or modern network hardware.
9. Conclusion

The creation and development of Intrusion Detection Systems over the years has helped system and network administrators to have one more line of early warning and defense against malicious individuals who posed a threat for their networks.

Their degree of help is still argued in the computer security circle with one side speaking of advantages they give and the other speaking about disadvantages they present.

Whatever the arguments, one must not treat IDS either as one more item the management has found to make their lives difficult nor as one that will stop the attacks and allow the administrators to relax.

They certainly do provide help, and they do serve as early warnings, but only if combined with a well-maintained network, well-defended computer, good security policy concerning user identification and authentication and well informed security personnel.

10. References and Bibliography

(Barber; Base; Crosbie; ICSA; Sundaram; Lunt 1993; Frank 1994; Ptacek 1998; Nong 2000; Bruneu 2001; DelVeccio 2001; Planquart 2001; Sarmah 2001; Verwoerd 2001; Das 2002)


Crosbie, M., Dr. Spafford, G. Applying Genetic Programming to Intrusion Detection. West Lafayette, COAST Laboratory.


ICSA An Introduction to Intrusion Detection Assesment for Systems and Network Security Management, ICSA.


Sundaram, A. "An Introduction to Intrusion Detection."