

UNIVERSITY OF BELGRADE
FACULTY OF ELECTRICAL ENGINEERING

Yousef H. Abuadlla

**Flow-Based Anomaly Intrusion
Detection System Using Two Neural
Network Stages**

Doctoral Dissertation

Belgrade, 2014

UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Yousef H. Abuadlla

**Sistem za detekciju upada zasnovan na
tokovima sa dve neuralne mreže**

doktorska disertacija

Beograd, 2014

Graduation committee:

Chairman:

Professor Zoran Jovanović
School of Electrical Engineering, University of Belgrade

Members:

- 1) Professor Dušan Starčević
Faculty of Organizational Sciences, University of Belgrade
- 2) Professor Goran Kvaščev.
School of Electrical Engineering, University of Belgrade
- 3) Professor Slavko Gajin.
School of Electrical Engineering, University of Belgrade

Abstract

Internet users and computer networks are suffering from rapid increase in number of attacks. In order to keep them safe, there is a need for effective security monitoring systems, such as Intrusion Detection Systems. Many researchers concentrate their efforts on this area using different type of approaches to build reliable intrusion detection system. Today's commercially available intrusion detection systems are predominantly signature-based intrusion detection systems that are designed to detect known attacks by utilizing the signatures of those attacks. Such systems require frequent rule-base updates and signature updates, and are not capable of detecting unknown attacks. In contrast, anomaly detection systems, a subset of intrusion detection systems, model the normal system/network behavior which enables them to be extremely effective in finding and foiling both known as well as unknown attacks. While anomaly detection systems are attractive conceptually, a host of technological problems need to be overcome before they can be widely adopted. These problems include: high false alarm rate, failure to scale to gigabit speeds, etc. Flow-based Anomaly intrusion detection systems are one of these approaches that rely on aggregated traffic metrics. Their main advantages are host independence and usability on high speed networks.

The aim of this research is to propose and investigate a neural network based Intrusion Detection System that can promptly detect and classify attacks, either if they are known or never seen before. The proposed system makes use of neural network as analysis method and flow-based network data as data source. A Two Stages Neural Network intrusion detection system based on flow data is proposed for detecting and classifying attacks in network traffic. The first stage detects significant changes in the traffic that could be a potential attack, while the second stage defines if there is a known attack and in that case classifies the type of attack. The first stage is crucial for selecting windows where attacks, known or unknown, are more probable. Two different neural network structures were used, multilayer and radial basis function network, with the objective to compare performance, memory consumption and the time required for network training. The experimental results demonstrate that the designed models are promising in terms of accuracy and computational time, with low probability of false alarms.

резиме

Корисници интернета и рачунарских мрежа суочавају се са брзим пораст напада на мреже. Да бисмо их учинили безбедним, постоји потреба за ефикасним системима надзора безбедности, као што су системи за детекцију упада. Многи истраживачи концентришу своје напоре у овом подручју, користећи различите врсте приступа како би изградили поуздан систем за детекцију упада. Данашњи комерцијално расположиви системи за детекцију упада су углавном системи за детекцију упада на бази потписа, који су дизајнирани да открију познате нападе коришћењем потписа тих напада. Овакви системи захтевају честа ажурирања потписа и правила препознавања, а нису у стању да детектују непознате упаде. Насупрот томе, системи за детекцију засновани на аномалијама, који представљају подкуп система за детекцију упада, моделују нормално понашање система/мрежа, које им омогућава да буду изузетно ефикасни у проналажењу како познатих тако и непознатих напада. Док су системи за детекцију засновани на аномалијама атрактивни као концепт, многи технолошки проблеми морају да буду превазиђени пре него што постану широко прихваћени. Ови проблеми обухватају: високу стопу лажних аларма, неуспех да се скалирају на гигабитне брзине, итд. Системи за детекцију аномалија засновани на токовима су један од приступа који се ослањају на агрегираној саобраћајној метрици. Њихове главне предности су независност од рачунара домаћина и употребљивост на мрежама високих брзина.

Циљ овог истраживања је да предложи и истражи систем за откривање упада заснован на неуралним мрежама који може брзо да открије и класификује нападе, било да су у питању познати или никада раније откривени напади. Предложени систем користи неуралне мреже за анализе и токове скупљене из мрежног саобраћаја као извор података. Систем за детекцију упада помоћу неуралних мрежа у два нивоа заснован на протоку података је предложен за откривање и класификацију упада у мрежном саобраћају. Прва фаза детектује значајне промене у саобраћају које би могле бити потенцијални упад, док друга фаза дефинише да ли постоји познати упад и у том случају класификује врсту

упада. Прва фаза је кључна за избор прозора где су упади вероватнији, било да су познати или не. Две различите структуре неуралне мреже су коришћене, вишеслојна и мрежа радијалне функције, са циљем да се упореде перформансе, потрошња меморије и потребно време за тренирање мреже. Експериментални резултати показују да су дизајнирани модели обећавајући у погледу тачности и времена извршавања, уз малу вероватноћу појаве лажних аларма.

Content

Abstract	i
Table of Content	iv
List of Figures	vii
List of Tables	viii
Acknowledgements	ix
Chapter 1: Introduction	1
1.1 History.....	1
1.2 Motivations and Aims.....	2
1.3 Organization of the Dissertation.....	6
Chapter 2: Background	7
2.1 Network flows.....	7
2.1.1 Flow definition.....	7
2.1.2 The metering and collection process.....	8
2.1.3 Flow export protocols.....	9
2.1.4 Flow Sampling.....	11
2.2 Intrusion Detection Systems.....	11
2.2.1 Earlier Research on Intrusion Detection.....	13
2.2.2 Current Intrusion Detection Systems.....	14
2.2.3 Anomaly Intrusion Detection Systems.....	16
2.2.4 Misuse Intrusion Detection Systems.....	18
2.2.5 Hybrid of misuse and anomaly Intrusion Detection System.....	19
2.2.6 Misuse versus Anomaly Detection.....	19
2.3 Intrusion Prevention System.....	20
2.3.1 How Does Intrusion Prevention Work?.....	21
2.3.2 Intrusion Prevention System: Strengths & Weaknesses.....	21
2.4 Attack methods.....	22
2.4.1 Denial of Service (DoS).....	24
2.4.2 Trojan Horses.....	26

Content

2.4.3 Viruses and worms.....	26
2.5 What could be better for today’s system?	27
Chapter 3: Neural Networks	30
3.1 Introduction.....	30
3.2 Neural Network Operation.....	32
3.3 Neural Network Architectures.....	33
3.3.1 Multi Layer Perceptron.....	35
3.3.2 Radial Basis Function Network.....	37
3.3.3 RBF Networks vs. Multilayer Perceptrons.....	39
3.4 Neural Network Learning.....	40
3.4.1 Supervised Learning.....	43
3.4.2 Unsupervised learning.....	44
3.4.3 Reinforcement learning.....	44
3.5 Back-Propagation Network.....	45
3.6 Applications of Neural Network.....	47
3.7 Neural Network and Intrusion Detection System.....	47
Chapter 4: Proposed Intrusion Detection System	51
4.1 Introduction.....	51
4.2 Flow-Based Solutions.....	53
4.2.1 Denial of Service.....	53
4.2.2 Scans.....	55
4.2.3 Worms.....	57
4.3 Proposed Approach.....	59
4.3.1 Flow Collector Module.....	60
4.3.2 Feature Preparation Module.....	61
4.3.3 Anomaly Detection Module.....	61
4.3.4 Detection and Classification Module.....	62
4.3.5 Alert module.....	63
4.4 Training Dataset.....	64

Content

4.4.1 Existing Dataset.....	64
4.4.2 Flow-Based Dataset.....	65
Chapter 5: Experimental Results	67
5.1 Training and Testing Proposed System.....	67
5.2 Anomaly Detection module test and results.....	68
5.3 Detection and Classification Module test and results.....	70
5.4 Discussion of Results.....	72
5.5 Comparison of Results.....	73
Chapter 6: Conclusions and Future Work	76
6.1 Conclusion.....	76
6.2 Future work.....	77
References	78
Biography	86
Appendices	87

List of Figures

Figure 1.1	Trends in incidents and vulnerabilities.....	3
Figure 2.1	IP flow exporting and collecting architecture.....	8
Figure 2.2	The evolution of attack sophistication.....	23
Figure 2.3	Distributed Denial of Service attack.....	25
Figure 2.4	Firewall IDS and Honey Net protecting a LAN.....	28
Figure 3.1	Neural Network model.....	31
Figure 3.2.	Sigmoid Function.....	32
Figure 3.3	Neural network architecture.....	34
Figure 3.4	Neural network active nodes.....	35
Figure 3.5	Multi Layer Perceptron.....	36
Figure 3.6.	Radial basis function network structure.....	38
Figure 3.7	Neuron Weight Adjustments.....	40
Figure 3.8	Supervised Learning.....	43
Figure 3.9	Unsupervised Learning.....	44
Figure 4.1	Example of sketch.....	55
Figure 4.2	Categories of scans.....	56
Figure 4.3	Example of 2D sketches.....	57
Figure 4.4	Host classes and their intersections.....	58
Figure 4.5	Proposed Approach.....	59
Figure 4.6	IP Flow exporting and collecting architecture.....	60
Figure 5.1	Detection rate of stage one neural network.....	69
Figure 5.2	Performance of the anomaly detection module.....	69
Figure 5.3	Performance of the detection and classification module.....	71
Figure 5.4	Detection rate for stage two neural networks.....	72
Figure 6.1	Detection Rate on Different Datasets for IDSs.....	77

List of Tables

Table 2.1	Cisco NetFlow Flow Record Fields.....	10
Table 2.2	Misuse vs. Anomaly intrusion detection.....	20
Table 5.1	Used Data set.....	67
Table 5.2	Results of Anomaly Detection phase.....	68
Table 5.3	Neural Network Classified Categories.....	70
Table 5.4	Detection and Classification Procedure.....	70
Table 5.5	Results of detection and classification.....	71
Table 5.6	The results of classification stage.....	72
Table 5.7	Comparison of Intrusion Detection Systems Using NN.....	75

Acknowledgements

I would like to express my deep gratitude to my supervisors, Professor Zoran Jovanovic for his patient guidance, enthusiastic encouragement and constructive suggestions during the planning and development of this research work. I would also like to thank Dr. Slavko Gajin and Dr. Goran Kvascev for their advice, assistance, and support throughout this research work. My grateful thanks are also extended to all members of Electrical Faculty.

I would also like to thank my mother, sisters, and brothers. They were always supporting me and encouraging me with their best wishes.

Finally, I would like to thank my wife, my sons, and my daughter, who supported me through the good times and bad. Without their endless love and trust, I would not even dream of enjoying my little achievement today.

Chapter 1: Introduction

1.1 History

Internet has almost become a “new world”, and as in the real world the “new world” has criminals and vandals. The big threat of vandalism and theft has given users a need for security components to protect themselves.

In 1983 the ARPAnet, and every network attached to the ARPAnet, officially adopted the TCP/IP networking protocol. The TCP/IP networking protocol had been under development since 1973, and had been tested in an internet .in 1973 [1]. From 1983, all networks that used TCP/IP were collectively known as the Internet. The standardization of TCP/IP allows the number of Internet sites and users to grow exponentially [2, 3]. When Internet started to be widely used, the users were so excited about connecting systems that security was forgotten. Everyone just wanted to use the Internet, and did not think of the dangers it also brought. The first Internet worm was unleashed on November 2 1988 by Robert T. Morris Jr [3, 4]. Since then, the number of incidents is growing rapidly each year. In 2003, the number of incidents was 137529 [5]. In 1989, Kevin Mitnick was arrested for invading Digital Equipment Corporation’s computer system and allegedly stealing software. He had then been breaking into different computer systems for several years, and is now known to be the first high profile computer hacker [3]. All information systems and computer networks are threaten by electronic attacks. Computer systems today have a variety of threats, such as [6]:

- Integrity
- Confidentiality
- Denial of Service
- Authentication

A totally safe system is per today impossible to achieve when we have Internet access. Surveys show that the threat from computer crime and other information security breaches continues unabated, and that the financial toll is mounting [7]. Therefore we have to stay alert for attacks and misuse. Most likely they will happen sooner or later.

We can say that the silver bullet in network security would be to lock the computers in a bank vault, with no external access at all and armed guards to guard the vault. But still there would be the threat of inside attacks from for example the guards. It is of course not possible to have a system like this, because most systems need access to the outside world. This is why we have to get the security level up close to the same level as

Chapter 1: Introduction

locking the computers into a bank vault. Network security can be seen as a chain. It is said that a chain is not stronger than its weakest link. The same can be said about network security. Your security levels are not higher than the weakest part in your security components. And in this occasion a quotation from Babylonian Talmud, Tractate Baba Metzia [6] is illustrative; “It is not the mouse that is the thief, it is the hole that lets the mouse in”. A couple of years ago someone who wanted to break into a computer system had to have very good computer skills. He had to know the security holes, and how to exploit these. Today, the intrusion threats are bigger than ever.

This is because of the fact that there are applications available on the Internet that gives people with almost no computer experience the possibility to break into computer systems. Because of this, attacks against computer systems and networks have increased significantly in the last years [8]. Today, almost everyone can find tools to use for attacks. Someone who is interested in this can easily search for such tools at for example Google [9] and start using them from home. And for someone who for example just want to attack a small neighborhood firm, their IP address can easily be hidden by the use of public proxy servers found on the Internet.

As Kelly Schupp from Guarded-Net noted [10], “We don’t believe there’s one silver bullet product, nor will there ever be. However, hopefully with the implementation of newer solutions, life will become a little more manageable and (at least temporarily) more secure”. Most of the prevalent Internet attacks today can be stopped or mitigated proactively with little fear of false attacks. But what about the new and unknown attacks? Are these attacks not the worst? It is hard to protect yourself against something you do not know anything about.

1.2 Motivations and Aims

The Internet is a complex system in constant evolution. Nevertheless, it is possible to make some observations with respect to security. A first observation is that the number of attacks continues to grow. The Cert Coordination Center [5], one of the most well-known risks, security threats and incidents response centers, offers summaries of the yearly security situation of the Internet. The Cert/CC maintains a database of vulnerabilities, with the aim to categorize them according to their severity level and

Chapter 1: Introduction

damaging impact on the systems. Vendors, system administrators and users are encouraged to submit vulnerabilities.

In a similar way, in the past, Cert/CC asked the Internet community for collaboration in order to report the incidents the users were subject to. Cert/CC defines an incident as the act of violating an explicit or implied security policy. This definition, according to the Cert/CC, covers attempts to gain access to (information on) a system, Denial of Service, disruptions, unauthorized uses and changes to hardware and software. Since 1995, Cert/CC published each year the number of catalogued vulnerabilities. In fact, the reporting of incidents started already in 1988, but ended in 2003. The reason to stop can easily be understood from Figure 1: the growth of reported incidents is nearly exponential, while the number of catalogued vulnerabilities shows a slower growth factor. The Cert/CC gives the following explanation:

“Given the widespread use of automated attack tools, attacks against Internet-connected systems have become so commonplace that counts of the number of incidents reported provide little information with regard to assessing the scope and impact of attacks. Therefore, we stopped providing this statistic at the end of 2003.”

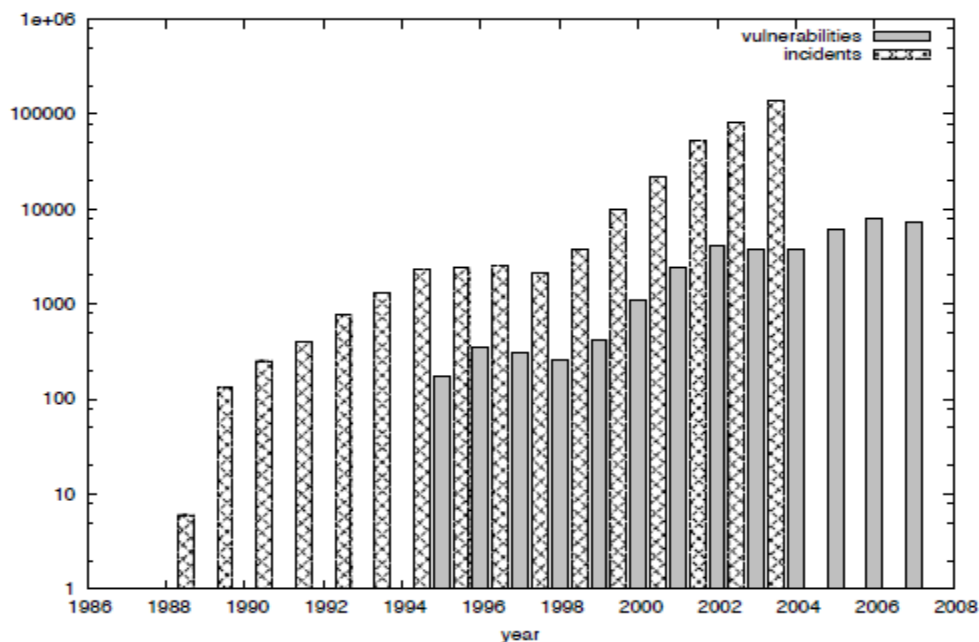


Fig. 1.1: Trends in incidents and vulnerabilities (logarithmic scale).

A second observation is that Internet traffic, as well as line speed, continues to grow. Nowadays an access speed of 1- 10Gbps is not unusual. A university network, for

Chapter 1: Introduction

example, reaches traffic averages in the order of hundreds of Mbps, with high activity peaks in the order of Gbps. On backbone networks, the throughput will even be higher. Internet2 [11], for example, publishes weekly reports of the Abilene traffic.

It is clear that Network Intrusion Detection Systems (NIDS) should be able to handle the growing number of attacks, the growth in Internet traffic as well as the increase in line speed. Researchers assess the current, payload-based, NIDS processing capability to lie between 100Mbps and 200Mbps [12], [13]. Well known systems like Snort [14] and Bro [15], exhibit high resource consumption when confronted with the overwhelming amount of data found in today's high-speed networks [16]. In addition, the spread of encrypted protocols poses a new challenge to payload-based systems. An example is the work of Taleb *et al.* [17], [18], where the authors propose an intrusion detection systems based on per-packet inspection that rely only on header information in order to identify misuses in encrypted protocols. Given these problems, flow based approaches seem to be a promising candidate for Intrusion Detection research. Flows are created by specialized accounting modules usually placed in network routers. The same modules are responsible of exporting the flows to external collectors. Flow-based Intrusion Detection Systems will analyze these flows and detect attacks. Compared to traditional NIDS, flow-based NIDS have to handle considerable lower amount of data. Flow based intrusion detection is therefore the logical choice for high-speed networks. However, there might exist situations in which the benefit of using flows is not so pronounced. The worst case scenario would be when a flow is created for each packet passing through the monitoring point, as a consequence of a distributed DoS attack (DDoS), for example. In this case, the number of flows would increase dramatically and extra load would be put on the monitoring and analysis systems. To mitigate this problem, or, in general, to improve the performance of routers and monitoring stations, sampling techniques or flow aggregations [19] can be applied. Sometimes it is argued that flows do not carry enough information, compared to payload inspection, for being useful for intrusion detection. The answer to this question highly depends on the user's goals. Flows, which represent by nature aggregated information, do not carry any payload. They, therefore, do not provide the detection precision of packet-based inspection, which allows for example pattern matching in payload content. Flows are limited to information regarding network interactions. With this information, it is still possible,

Chapter 1: Introduction

however, to identify communication patterns between hosts, when communication takes place and which amounts of packets and bytes have been moved. For many attacks, this information is sufficient. In any case, it is important to underline that flow-based intrusion detection is not supposed to substitute the packet-based one, but rather complements the approach by allowing early detection in environments in which payload-based inspection is not feasible. As described by Schaffrath et al. [20], in an ideal world payload-based solutions would always outperform flow-based ones in accuracy. In high-speed networks, however, the processing capabilities of the NIDS may be too limited to allow payload-based approaches.

The aim of this research is to find out what needs to be done to make a computer system safer, without having a system that sends out false alarms that takes up much of the time of an already busy system administrator. The job for security administrators is almost impossible today. No matter how many holes the security administrators finds in their network, and no matter how many bugs they fix to keep intruders out, the intruder just needs to find one hole to get in.

The combination of growing network load and attack frequency is challenging if we are aiming to effectively detect intruders. The network monitoring community reacted to the ever growing amount of data by focusing on network flows, rather than individual network packets. A flow is defined as a set of packets that have common properties, as, for example, having the same source and the same destination (see chapter 2, Section 2.1.1). Measuring flows offers an aggregated view of traffic information and drastically reduces the amount of data to be analyzed. Flows are therefore a possible solution to cope with scalability issues in IP monitoring. However, from a security perspective, we do not yet see a definite answer to the problem of intrusion detection in situations, as high-speed networks, in which the traditional packet-based solutions may no longer be feasible. Flows therefore appear as a promising approach that may lead to improved results in the field of intrusion detection in high-speed networks.

The goal for this research is to develop an Intrusion Detection System that is able to detect both known and unknown attacks without relying on signatures or other hard coded updates to stay protected against the latest attacks. For this we will examine neural networks ability to learn user behavior, and we will use this for intrusion detection.

Chapter 1: Introduction

1.3 Organization of the Dissertation

The content of this thesis is divided into six chapters. Chapter one introduces the history, motivation and aim for the research. Chapter two presents the overview of security components, and goes deeper into Intrusion Detection Systems. Chapter three describes the neural networks in details. Chapter four proposes and describes the methods that are used in this research. Chapter five lists and analyzes the results of the experiments. Chapter six draws out the conclusions and future work.

Chapter 2: Background

2.1 Network flows

In the last decade, flows have become quite popular in IP network monitoring, since they help to cope with the scalability issues introduced by the increasing network speeds. Nowadays all major vendors offer flow-enabled devices, such as, for example, Cisco routers with Netflow [21]. The Internet Engineering Task Force (IETF) is currently working on an IP flow standard, IP Flow Information eXport (IPFIX).

2.1.1 Flow Definition

In the literature, several flow definitions can be found [22, 30, 29, and 23]. We present the definition of IP flow as it is described by the IPFIX working group within the IETF [43, 24]:

“A flow is defined as a set of IP packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties.”

In the IPFIX terminology, the common properties are called flow keys. An example of flow keys commonly used for characterizing a flow is:

(Source IP; Destination IP; Source port; Destination port; IP protocol):

Aggregated views on the network traffic can be obtained by choosing coarser grained flow definitions, according to the need of the network administrator. An overview of this process is given by Fioreze et al. [30, 29]. It is analogously possible to have more detailed flow definitions. For example, additional fields can be introduced as extensions for diverse applications. These additional fields are described in IPFIX and they have been recently included in Flexible Netflow [21].

Note: There are important differences between flows and Transmission Control Protocol (TCP) connections. A TCP connection determines a pair of flows: one from the initiator of the connection to the destination, and one from the destination to the initiator. However, a flow should not necessarily be due to the TCP protocol. For example, stream of User Datagram Protocol (UDP) packets between source host A and a destination host B will result in a flow.

Chapter 2: Background

Moreover, a flow does not have size restrictions: each communication between source and destination hosts will generate a flow, even if a single packet has been exchanged. Traditionally, flows are also unidirectional, whereas TCP connections are by definition bidirectional. However, IETF has recently introduced a definition of bidirectional flows [44], since bidirectional data can further improve export and collection efficiency.

2.1.2 The Metering and Collection Process

Monitoring flows entails a two-step process: flow exporting and flow collection. These tasks are respectively performed by two components: the exporter and the collector. Figure 2.1 presents an overview of the metering and collection process. The flow exporter, or monitoring point, is usually a router or a different flow enabled device. It is responsible for the metering process, i.e., the creation of flow records from observed traffic.

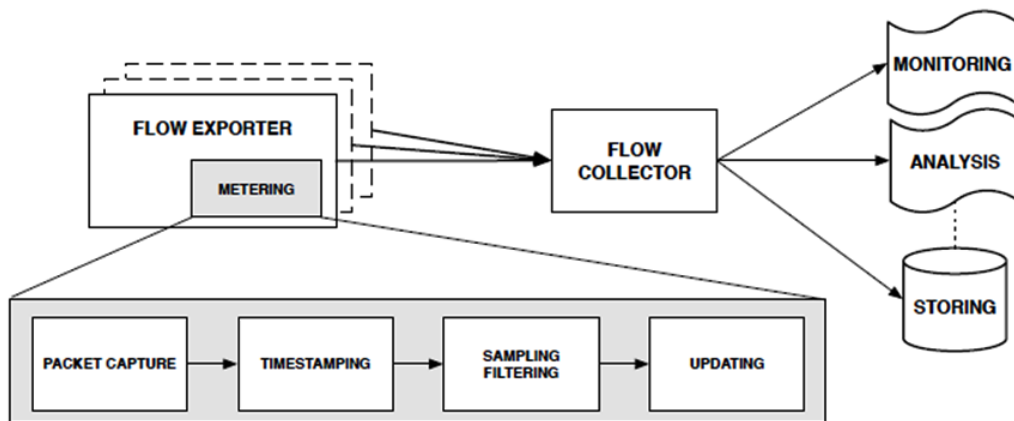


Figure 2.1: IP flow exporting and collecting architecture

The flow exporter extracts the packet header from each packet passing through the monitoring interface. Each packet header is marked with the timestamp when the header was captured. The header is then processed by a sampling-filtering module, where it can be sampled (see Section 2.1.4) and filtered according to specific administrative requirements (e.g., a specific protocol or IP range). The final step is the update module. Each incoming packet header triggers an update to a flow entry in the flow cache. If there is no flow matching the packet header, a new flow entry is created.

Chapter 2: Background

A flow record is exported to the flow collector when it is considered expired. In the case of Cisco NetFlow [22] and similarly in IPFIX [43, 24], a flow expires when:

- The flow was idle (no packets belonging to the flows have been observed) for a time interval longer than a given threshold. This threshold is known as inactive timeout. The default value for the inactive timeout for Cisco Netflow [22] is 15 seconds. However, it can be tuned according to the operators' requirements. GEANT [32] uses an inactive timeout of 60 seconds;
- The flow reaches a maximum allowed lifetime, known as active timeout. For Cisco Netflow [22], the active timeout is 30 minutes, but our experience showed that shorter timeouts are also common. SURFnet [45] and GEANT [32] both use an active timeout of 5 minutes;
- The FIN or RST flags have been seen in a TCP flow, indicating the end of a TCP connection;
- The flow-cache memory is exhausted. In this case, a subset of the flows in the cache is marked as expired and exported to the collector. Least Recently Used algorithms may be used to free the flow-cache memory, as well as heuristic algorithms.

The aim of the flow collector is to receive the flow records from the flow exporter and to store them in a form suitable for further monitoring or analysis. Examples of flow collector and analysis tools are flow-tool [46], nfdump [47], sFlowTrend [34], IsarFlow [35] and DiCAP [39, 40].

2.1.3 Flow Export Protocols

A flow export protocol defines how expired flows are transferred by the exporter to the collector. The information exported to the collector is usually referred as flow record.

Note: The terminology flow and flow record usually raises the question about the actual difference between the two. A flow is the complete unidirectional stream of packets between a source and destination in a network, while a flow record is the information stored in the flow exporter cache. A flow can coincide with a flow record, if the flow duration is shorter than the exporter active timeout. Flows longer than active timeout,

Chapter 2: Background

will be split into several flow records. In other words, a flow record is the information describing (part of) a flow as we obtain it directly from a flow exporter.

Cisco Netflow version 5 [22] is a simple protocol that exports flow records of fixed size (45 bytes in total). Each export datagram will contain up to 30 flow records. The fields of a Netflow Version 5 flow record are summarized in Table 2.1.

Table 2.1: Cisco NetFlow Flow Record Fields.

Field Description
Source IP address
Destination IP address
Next hop router IP address
SNMP input and output interfaces indexes
Total number of packets in the flow
Total number of Layer 3 bytes in the flow packets
Start of flow timestamp
End of flow timestamp
Source and destination port number
Cumulative OR of TCP flags
IP protocol (for example, 6 = TCP, 17 = UDP)
IP Type of Service
Source and Destination Autonomous system
Source and destination address prefix mask bits

Cisco Netflow version 9 and IPFIX [43, 24] propose flexible protocols in which flow record formats can be defined by using templates. These protocols allow also a larger set of parameters to be exported, such as, for example, sampling rate and algorithm, source and destination VLAN identifiers, MAC addresses and autonomous system numbers [23, 42]. An IPFIX packet is logically divided into sections known as sets. A message can normally consist of three kinds of sets, namely template sets (format template exchange), data sets (flow records) and options template sets (necessary for the correct interpretation of a template set). For a more detailed treatment of the IPFIX message format, we refer to [24]. In Netflow v9 terminology, template sets are referred as template FlowSet and data sets as data records.

Chapter 2: Background

2.1.4 Flow Sampling

In certain situations the network load might be too high to export every flow.

Sampling is a useful technique in such situations since it can significantly reduce CPU load as well as the amount of exported flows. Sampling is a methodology for selecting only a predefined subset of all available network flows. Indeed, sampling implies that many flows are lost and are not exported to the collector. But if network traffic is too high or the hardware not efficient enough, then sampling might be the only possibility to counter the high network load.

Cisco devices usually distinguish between deterministic, time based and random sampling. Deterministic sampling selects every n th network packet. Time based sampling selects a network packet every n mille-seconds and random sampling randomly selects one network packet out of n packets. In each case the variable n is specified by the network operator. Most of the time random sampling is advised.

2.2 Intrusion Detection Systems

The goal of intrusion detection is seemingly simple; to detect intrusions. An Intrusion Detection System is a program that can detect and inform the Network Administrator about an attack or misuse. The use of Intrusion Detection Systems is getting more and more common nowadays. It is important to know that an Intrusion Detection System alone is not the silver bullet in network security. Using an Intrusion Detection System is more an addition to other security components, as for example firewalls, to make the protected system more secure.

Intrusion detection is the process of monitoring computer networks and systems for violations of security policy. The assumptions of Intrusion Detection Systems are that the intruder has to behave differently from the normal users.

The components of an Intrusion Detection System are [48]:

- Information Source: data utilized by the Intrusion Detection System.
- Analysis engine: process by which the intrusion detection is made.
- Response: action taken when an intrusion is detected.

Chapter 2: Background

Intrusion Detection Systems works by gathering information from the protected system and network and search for information or patterns that can be an attack or misuse. They can detect intruders by examining parameters as network traffic, CPU and I/O utilization, user location and file activity for signs of an attack [25]. Intrusion Detection System can be used to detect misuse from within the organization and to detect attacks from the outside world. The major functions for an Intrusion Detection System are [26]:

- Monitoring and analyzing user and system activity.
- Assessing the integrity of critical system and data files.
- Recognizing activity patterns reflecting known attacks.
- Responding automatically to detected activity.
- Reporting the outcome of the detection process.

The main goal of an effective Intrusion Detection System is to provide high rates of attack detection with very small rates of false alarms [27]. The Intrusion Detection Systems that are used today are a long way from achieving this goal. There are two types of errors that are important to know in intrusion detection [26]:

- False positives: Also known as false alarms. These errors occur because the Intrusion Detection System misinterprets normal traffic or activities as an attack.
- False negatives: These errors occur because an attacker is misclassified as a normal user by the Intrusion Detection System.

False positives are those error messages that take much of the system administrator's time. A high rate of these errors will degrade the productivity of the system by invoking unnecessary countermeasures.

False negatives are those errors that are hard to detect because the system sees the attacker as an ordinary user. These attacks are also the most dangerous and these errors can cause big losses for a company.

Intrusion Detection Systems differs from on-line to off-line systems [28]. Offline systems are run periodically and they detect intrusions after-the-fact based on system logs. On-line systems are designed to detect intrusions while they are happening, thereby allowing for quicker intervention. Intrusion Detection Systems can also be classified according to the kind of audit source location they analyze [26]:

- Network-based detection: The Intrusion Detection System analyzes network packets captured in the network.

Chapter 2: Background

- Host-based detection: The Intrusion Detection System analyzes different logs for traces of an attack.

The host based Intrusion Detection System looks at communication in and out of the computer, checks the integrity of the system files and suspicious processes. The network based Intrusion Detection System looks at packets on the network as they pass the intrusion detection sensor. The best solution will be to have a system that combines these two systems.

2.2.1 Earlier Research on Intrusion Detection

The last 20 years there has been conducted much research on intrusion detection, starting with James P. Anderson's whitepaper "Computer Security Threat Monitoring and Surveillance" in 1980 [50]. Anderson introduced the concept of computer threats and detection of misuse. This is the same concept that is applied to host based Intrusion Detection Systems. Dorothy Denning wrote a report in 1987 [51]. This report has almost become a fundamental stone and has inspired many researchers in the intrusion detection research field. Almost every research paper on intrusion detection uses this paper as a reference. Denning introduced the first model for intrusion detection, and most of her work are still of current interest today.

Most of the newer research on intrusion detection focuses on anomaly detection [52]. This is because the strength in intrusion detection lies in anomaly detection, where the system does not need to depend on a signature before it can detect an attack. The use of neural networks in intrusion detection has been used several times by researchers the last decade. This will be explained further later in the thesis. There has also been research on other using soft computing techniques in intrusion detection.

In 2002 S. B. Cho showed in his report [53] that the use of hidden Markov models and attempts to detect intrusions by noting significant deviations from the model can be used with success in anomaly Intrusion Detection Systems. In this experiment he used systems call, process and file access as parameters for the intrusion detection. Experiments with the use of Self- Organizing Maps in intrusion detection have also been done [41]. The parameters that were used in this experiment were username, host, type of connection and time session started.

Chapter 2: Background

A report from late 2000 [37] concluded that all the evaluation performed to that date indicated that Intrusion Detection Systems were only moderately successful at identifying known intrusions, and quite a bit worse at identifying those that had not been seen before.

2.2.2 Current Intrusion Detection Systems

Today's IDS is a combination of signature analysis, network traffic monitoring, and network behavior analysis (also referred to as anomaly detection) technologies. The heart of most solutions today is signature analysis (i.e. monitoring traffic for known attack patterns - everything from minor attacks through the latest, high-profile worms). Typically, a signature-based IDS is configured with thousands of rules that detect potentially malicious attacks and codes. Positive proof of the effectiveness of IDS solutions and why they are an important component to a layered network security strategy is the sheer volume of attacks they are able to detect. Two drawbacks to signature-based IDS solutions are false-positives and the time lapse to create signatures for new exploits. False-positives are pattern matches inaccurately identified as attacks. Historically false-positives have inundated administrators thus causing an even greater problem - desensitization.

Today's solutions are actively and aggressively trying to solve the issue of false-positives, including providing elaborate "tuning" mechanisms which effectively disable signatures that cause false-positives.

Similar to anti-virus software, an attack must be analyzed before a signature is developed to recognize it. This time lapse can be critical. Recently, the time between a new vulnerability and its associated exploit has been decreasing, placing more pressure on IDS manufacturers to rush signatures to the market. The recent attacks benefited from this time interval, allowing it to become the fastest spreading worm in history. Timely delivery of signatures is integral to overall IDS effectiveness.

An alternative to signature-based IDS is called behavioral or anomaly-based IDS.

The basic premise of this sub-category of IDS is that normal network traffic generally behaves within certain patterns. For example, opening network ports in rapid succession is typically not seen in normal traffic, so a behavioral or anomaly-based IDS may flag

Chapter 2: Background

that traffic as abnormal and identify it as a port scan (generally a precursor to an attack). Large, sustained amounts of fragmented packets are also abnormal patterns and will also be flagged. These systems have not broken into the mainstream, but seek to provide an alternative to the drawbacks of signature-based systems.

Monitoring for intrusions is a critical component of any network security policy. The greatest challenge when working with IDS systems has been sifting through and utilizing the large volume of data generated. Due to the nature of its design, the roles of IDS systems have largely been one of postmortem or historical reporting. The critical question facing IDS solutions is this: Is detecting attacks enough?

Also several other questions unsolved, and most of them are still not answered completely today:

- Soundness of approach: Does the approach actually detect intrusions? Is it possible to distinguish anomalies related to intrusions from those related to other factors?
- Completeness of approach: Does the approach detects most, if not all, intrusions, or are a significant proportion of intrusions detectable by this method?
- Timeliness of approach: Can we detect most intrusions before significant damage is done?
- Choice of metrics, statistical models and profiles: Which metrics, models, and profiles provide the best discriminating power? Which are most cost-effective? What are the relationships between certain types of anomalies and different methods of intrusion?
- System design: How should a system based on the model be designed and implemented?
- Feedback: What effect should detection of an intrusion have on the target system? Should Intrusion Detection Expert System automatically direct the system to take certain actions?
- Social implications: How will an Intrusion Detection System affect the user community it monitors? Will it deter intrusions? Will the users feel their data is better protected? Will it be regarded as a step towards “big brother”? Will its capabilities be misused to that end?

Chapter 2: Background

2.2.3 Anomaly Intrusion Detection Systems

Anomaly detection uses models of the intended behavior of users and applications, interpreting deviations from the “normal” behavior as a problem [49]. Maxion and Tan [36] have expanded this definition: “An anomaly is an event (or object) that differs from some standard or reference event, in excess of some threshold, in accordance with some similarity or distance metric on the event”.

The task of anomaly intrusion detection is to determine if an activity is unusual enough to suspect an intrusion. A basic assumption of anomaly detection is that attacks differ from normal behavior [32]. If an organization implements an anomaly based Intrusion Detection System, they must first build profiles of normal user and system behavior to serve as the statistical base for intrusion detection, and then use deviations from this baseline to detect possible intrusions [25]. Any activity sufficiently deviant from the baseline will be reported as anomalous and considered as a possible attack. Anomaly intrusion detection was the originally type of Intrusion Detection Systems. It was an anomaly Intrusion Detection System Denning proposed in her report [51] from 1987. Her Intrusion Detection Expert System model is based on the assumption that it is possible to establish profiles to characterize the normal interactions of subjects (typically users) with objects (typically files or programs). This type of intrusion detection can detect a variety of abnormal patterns of system usage. Here are some examples from D. Dennings report [35]:

- Attempted break-in: Someone attempting to break into a system might generate an abnormally high rate of password failures with respect to a single account or the system as a whole.
- Masquerading or successful break-in: Someone logging into a system through an unauthorized account and password might have a different login time, location, connection type from that of the account’s legitimate user. In addition, the penetrator’s behavior may differ considerably from that of the legitimate user. In particular, he might spend most of his time browsing through directories, and executing system status commands, whereas the legitimate user might concentrate on editing or compiling and linking programs. Many break-ins have been discovered by security officers or other users on the system who have noticed the alleged user behaving strangely.

Chapter 2: Background

- Misuse from legitimate users:

1. A user attempting to penetrate the security mechanisms in the operating system might execute different programs or trigger more protection violations from attempts to access unauthorized files or programs. If his attempt succeeds, he will have access to commands and files not normally permitted to him.

2. A user trying to leak sensitive documents might log into the system at unusual times or route data to remote printers not normally used. A user attempting to obtain unauthorized data from a database through aggregation and inference might retrieve more records than usual.

- Denial-of-Service attacks: An intruder able to monopolize a resource might have abnormally high activity with respect to the resource, while activity for all other users is abnormally low.

The main advantage of anomaly Intrusion Detection Systems is that they can detect previously unknown attacks. By defining what is normal, they can identify any violation, whether it is part of the threat model or not. In today's system the advantages of detecting previously unknown attacks is paid for in terms of high false-positive rates [49, 25]. Disgruntled employees, bribery and coercions make networks vulnerable to attacks from the inside [31].

Anomaly intrusion detection can detect if any employees differs from their normal routines to make any attempts to an attack. Disadvantages with anomaly Intrusion Detection Systems are that they are less effective in dynamic environments, where employees have erratic hours or switch project resources frequently. Also, inaccurate or incomplete user and system profiling can lead to false-positives [25]. This type of intrusion detection also has difficulty with classifying or naming the attacks, since they just depend on deviations from normal behavior [37]. When new users are introduced into the target system, two potential problems occur [51].

- Lack of profile information about the user's behavior.
- The user is inexperienced with the system.

Both these problems will give a high rate of false positives to the system so it is hard to know how to deal with these. One way to "solve" those problems is to ignore anomalies during a short period, or raise the deviation value. Both of these two solutions will give

Chapter 2: Background

even more dangerous problems. What if the new users make an intrusion? And what happens if the system is attacked during this period?

2.2.4 Misuse Intrusion Detection Systems

Misuse detection contains attack descriptions (or “signatures”) and matches them against the audit data stream, looking for evidence of known attacks [49]. These signatures are detailed descriptions of the sequence of actions performed by a hacker. This is a good method to stop known attacks, because known attacks can be characterized by a sequence of events.

Originally, and still, anomaly Intrusion Detection Systems has limitations because of the problems with dynamic environment and high rates of false positives. Because of this, misuse Intrusion Detection Systems was introduced [38]. Misuse Intrusion Detection System typically monitors parameters such as network traffic; CPU and I/O use, and file activity for activities that match known patterns or attack profiles [33].

The main advantage of misuse Intrusion Detection Systems is that they focus analysis on the audit data and typically produce few false-positives [25]. Since they rely on signatures, the system knows what kind of attack it is when it occurs. This way the system can easily assign names to the attacks when they occur, and the system administrator can see what kind of attack the system is under. The problem with these systems is that it is script based and only recognize known scripts (“signatures”), but are unable to detect truly novel attacks [10, 37]. Since misuse Intrusion Detection Systems have no capability of autonomous learning they require frequent updates. As new attacks are discovered, developers must model and add them to the signature database. A report from 1999 [31] showed that misuse Intrusion Detection Systems can be very effective in reducing false alarms if they are implemented properly. The problem is that there can also be small changes in the attack methods and to detect the changes new signatures has to be written. There are often written many variations of one signature and over time this will slow down the system because the signature database grows so big.

Chapter 2: Background

Today, nearly all Intrusion Detection Systems are signature based. The performance of these systems is limited by the signature database they work from. Many known attacks can be easily modified to present many different signatures. If the database does not contain all the different variations, even known attacks may be missed [38]. Attackers can also bypass the signatures by encrypting the code so that the packets do not match any known attack signatures [31].

2.2.5 Hybrid of Misuse and Anomaly Intrusion Detection System

There are systems out now that combines the two types of Intrusion Detection Systems. Hybrid systems can use a rules base to check for known attacks against a system, and an anomaly algorithm to protect against new types of attacks [25]. This type of Intrusion Detection System takes the advantages from both systems, but unfortunately it also takes some of the disadvantages. Misuse detection could be used in combination with anomaly detection to name the attacks. This will shorten the response time the system administrator needs as he can see what type of attack the system are under.

2.2.6 Misuse versus Anomaly Detection

Despite the fact that there has been done a lot of research on intrusion detection it is pretty clear that anomaly intrusion detection has more potential because of its ability to catch novel attacks. If for example there is an anonymous FTP connection attempts from an outside IP address this may not cause the system to be suspicious at all. But if the FTP connection attempt is within a set period of time after a scan from the same IP, it should become more suspicious. This can be done with the use of anomaly systems. An anomaly intrusion detection system will not grow “big and slow” over time, because it learns the pattern of the users over time. Table 2-2 contains the advantages and disadvantages of misuse and anomaly intrusion detection as they are today.

Chapter 2: Background

Table 2.2 Misuse vs. Anomaly intrusion detection

	Advantages	Disadvantages
Misuse IDS	<ul style="list-style-type: none">- Can name attacks- System administrators can write their own signatures- Easy to implement- Properly implemented, it does not give many false alarms.	<ul style="list-style-type: none">- The signature database tends to get big and clustered after a while. This can slow down the system- Cannot completely detect novel attacks- Needs to be updated with new signatures to catch newly discovered attacks- Unprotected against new attacks during the time it takes to write new signatures
Anomaly IDS	<ul style="list-style-type: none">- Can easily detect attacks from the inside- Hard for an intruder to know how he should behave to not raise an alarm since profiles can be on individual users- Can detect previously unknown attacks- Can use more sophisticated rules	<ul style="list-style-type: none">- Complex to implement- High rate of false alarms- Still not satisfying enough in a dynamic environment- Cannot name attacks.

2.3 Intrusion Prevention System

While Intrusion Detection Systems automatically handle intrusion detection, the system administrator usually manages intrusion recovery. Intrusion Prevention Systems was introduced because it was not accomplishing enough with just passive monitoring of system as today's Intrusion Detection System do. Intrusion Prevention Systems work by offering active threat handling capabilities that stop intruders and attackers before they can enter a computer system. The difference between Intrusion Detection Systems and Intrusion Prevention Systems is that when an Intrusion Detection Systems detects a problem, Intrusion Prevention Systems blocks it. Just like Intrusion Detection Systems, some of the Intrusion Prevention Systems are host based, and some are network based.

Chapter 2: Background

There are split meanings on whether Intrusion Prevention System is a new technology, or if it just is a “new way of thinking” where several security components are combined to collaborate with each other [33].

Newer Intrusion Prevention Systems are beginning to rely on software based heuristic approaches. But here, as in anomaly Intrusion Detection Systems, there are problems with dynamic environment and with defining accurate user profiles.

2.3.1 How Does Intrusion Prevention Work?

Intrusion Prevention is an advanced intelligent way of scanning the different layers for vulnerabilities. It consists of many techniques to ensure the optimal and most advanced security level. This includes:

- Database updated multiple times daily for the latest signature definitions.
- Traffic abnormalities are being identified and if consisting of dangerous content will be blocked.
- When a port scan is being performed, an attack will most likely follow in a matter of minutes afterwards.
- Denial of Service (DOS) attacks protection, because a successful DOS attack can cause your system to crash or be permanently damaged.
- Protection for known buffer overflow attacks and or other exploits being launched.
- Zero Day Protection, which is a module that protects for known and unknown Zero Day Vulnerabilities
- Wide protection for webmail, ftp, Windows, Linux, BSD, UNIX, Routers, Firewalls, Databases such as DB2, Oracle, MySQL, MS SQL, PostgreSQL.

2.3.2 Intrusion Prevention System: Strengths & Weaknesses

- ✓ Intrusion Prevention System was a leap forward from their predecessors, intrusion detection systems. At first, these new systems were spotty, and network security professionals were wary of using them. They were slow and drained valuable bandwidth, and often times blocked the wrong traffic.

Chapter 2: Background

Nowadays, the intrusion prevention industry has matured and the top competitors have been able to lower or eliminate the bandwidth utilized and ensure customized settings for blocking threats while still letting in the 'good guys.' Major strengths of intrusion prevention systems are:

- ✓ Reduces Time Spent Reviewing Log Files to Identify Threats.
- ✓ Reduces Need for Manpower to Monitor Threats.
- ✓ Enhances Network Security Architecture.
- ✓ Automatically Identifies and Blocks Threats.

While Intrusion Prevention System provide a baseline for network security, it is no longer enough. While improved, these systems were created for the static networks of yesteryear. Wireless devices, virtualization, cloud environments, and PDA devices (personal digital assistant) have all made today's networks more dynamic. The threats to these networks have adapted to take advantage of these changes, but the majority of intrusion prevention systems have not.

Weaknesses of many current intrusion prevention systems are:

- Lack of Network Visibility
- Lack of User Visibility
- Inability to Adapt to Network Changes in Real-Time

2.4 Attack Methods

There are numerous attack methods to use against a computer system, and several different types of each method. A good security administrator should keep himself updated with attack methods by visiting security websites where new attack methods are shown. There are several different attack types, and these will be explained further in this chapter. The attacks can mainly be sorted into three categories [61]:

- Attacks that deny someone else access to some services or resources a system provides.
- Attacks that allow an intruder to operate on a system with unauthorized privileges.
- Attempts to probe a system to find potential weaknesses.

Chapter 2: Background

All these and other attacks have been increasing in sophistication and power to harm. Attack tool developers are using more advanced techniques. It is more difficult to write signatures for signature-based systems such as antivirus software and misuse based Intrusion Detection Systems. We have seen tools like Code Red and Nimda propagate themselves to a point of global saturation in less than 18 hours [62].

As Figure 2.2 [65] shows, the sophistication of the attacks and attack tools has grown very much in complexity. And these attack tools has also been automated, so the skill needed to use these attack tools and to launch attacks has been reduced.

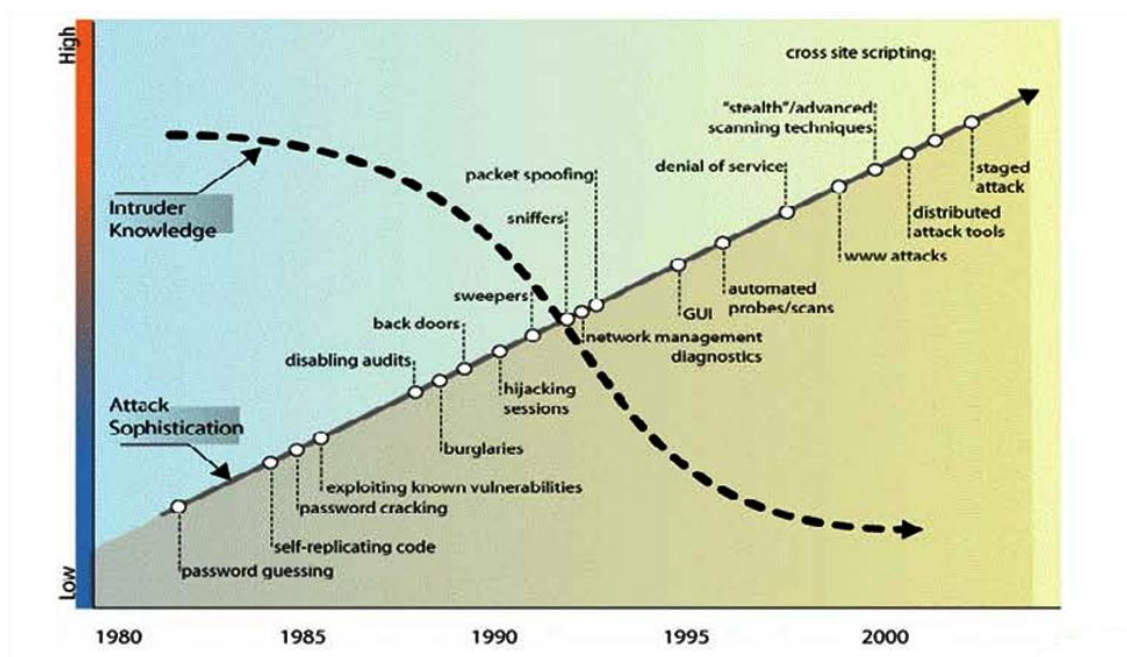


Figure 2.2 the evolution of attack sophistication

As an example of the difficulties posed by sophisticated attack tools, many common tools use protocols like IRC or HTTP to send data or commands from the intruder to compromised hosts [62]. As a result, it has become increasingly difficult to distinguish attack signatures from normal, legitimate network traffic.

The level of sophistication and knowledge required to carry out an attack has been decreasing. This is because there are very many know-how's available on Web sites all over the world. Hackers constantly invent new attacks and disseminate them over the Internet [61, 31]. Young and inexperienced hackers can use these tools with almost the same power as experienced hackers can. Some of the newer attack methods

Chapter 2: Background

also use encrypted signals. This keeps the signals from being recognized by Intrusion Detection Systems that scans for bit strings from known commands. The malicious code writers also works with an open source model in which they freely share successive code improvements, thereby making their attacks more sophisticated.

2.4.1 Denial of Service (DoS)

Denial of Service attacks is attacks where the attacker is not interested in any information from the network. He just wants to crash the system so that other users can't reach the targeted system [54]. In general, denial of service attacks does little harm besides wasting people's time and bandwidth [63].

The attacker just wants to deny the legitimate users to use the services provided by the attacked server. In the first versions of Denial of Service attacks [64], hackers usually tried to block access to a Web site by using a single computer to send millions of phony requests, thereby overloading the site so it could not respond to legitimate queries, or even causing the host to crash altogether. But it was pretty easy to stop these attacks. All requests from the attacking computer were simply blocked, and the attack was stopped.

A newer version of the Denial of Service attack, also called Distributed Denial of Service attack or DDoS, has evolved. These types of attacks are done by using other computers on the Internet to attack a system. In most attacks, the source address is faked [63]. This means that the attacker uses other people's computers to run the attack. The users who are used in such attack normally do not know that they have been used in an attack. The development of automation in attack tools enables a single attacker to install their tools and control tens of thousands of compromised systems for use in attacks [62]. Figure 2-3 shows how Distributed Denial of Service attacks are done against a single victim. The attacker uses remotely controlled computers to generate more request than the victims server can handle. Before the attack is launched, the attacker has installed a program on each of the remotely controlled computers, often called zombies. These zombies can be normal Internet users with ADSL or broadband connection, but often University networks are used because of their high speed networks.

Chapter 2: Background

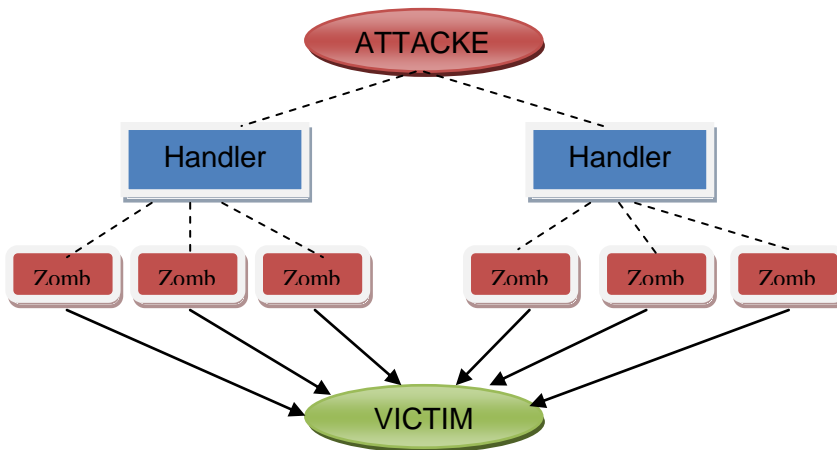


Figure 2.3 Distributed Denial of Service attack

The first known large scale Distributed Denial of Service attack was seen in August 1999 [64, 58]. This attack used 227 hosts to bring down the network of University of Minnesota in USA for three days. In February 2000 some of the major Internet players as Yahoo!, Amazon, eBay and other dot-coms were attacked with denial of service attacks that lasted for three days [63, 64, and 58]. These attacks slowed down the servers to make them unusable for normal users. And the attacks did actually affect the whole Internet. The attacks pumped out so much traffic, and so many people browsed the Web for information about the incidents that the entire Internet slowed down. On the last day of the attacks, the Internet's performance was 26, 8% worse than the week before [58].

In October 2002 nine of the 13 root-servers around the world were attacked by a Denial of Service attack. The attacks used commandeered computers to flood the root servers with Internet control message protocol requests [54].

Distributed Denial of Service attacks are seen as one of the biggest threats for businesses on the Internet. "Distributed Denial of Service attacks constitute one of the single greatest threats facing businesses involved in electronic commerce because an attack can completely shut down a Web site", said Morgan Wright from REACT [58]. Others are even more pessimistic about these attacks. Charles Palmer from IBM [52] had this to say about Distributed Denial of Service attacks: "You're not going to be able to stop denial of service. The best thing you can do is reduce its impact".

Chapter 2: Background

In today's e-commerce environment, users have a low tolerance for web site delay or failure. They will simply click their way to another site if the first is unavailable. There has been conducted a research where they tried to develop a new and efficient technique for the detection and alleviation of Denial of Service attacks [59]. Their technique is similar to an Intrusion Detection System, using anomaly based methods with data mining to detect attacks.

2.4.2 Trojan Horses

A Trojan horse is an illegal computer program disguised as legal, or hidden as part of a legal program. It can be described as a secret defect (or trap) that is intentionally inserted into legal software [60]. The Trojan horse can attack almost all programs, from basic systems software to users' application software. When the Trojan horse is installed on the victim's computer, it is often used to [6]:

- Propagate a virus or a worm
- Install a backdoor
- Destroy data

When it is installed, the Trojan horse gives the intruder access to the data stored on the victim's computer. It can also give the attacker access to other computers if the victim's computer is in a local network.

2.4.3 Viruses and Worms

Even though a virus is not actually an attack method, it causes much damage and is expensive and time consuming so it should be mentioned. Viruses and worms are malicious codes made to do some damage on the infected system. 85% of the respondents in the FBI/CSI survey [7] reported virus and worm outbreaks. Computer Economics estimated that the worldwide impact of Code Red was \$2.62 billion and the worldwide impact of Nimda was \$635million in 2002 [7].

Viruses and worms exploit vulnerabilities in the system, and large numbers of systems can be infected within a matter of hours. The Code Red worm infected more than 250.000 systems in just 9 hours on 19 July 2001 [65].

Computer Viruses started to spread through floppy disks on Apple computers as early as in 1981 [65]. They started to appear in large number in 1987, apparently starting in

Chapter 2: Background

Pakistan, Israel and Germany, and later appearing through the whole world. This caused thousands of computers to become unusable for short periods of time, hundreds of thousands computers to display spurious messages, tens of thousands of users to experience denial of services and several international networks to experience denial of service for a short period of time [57].

A decade ago, viruses were relatively easy to find and fix, and they spread slowly, generally by floppy disks or LANs. Now, however, increasingly creative authors are exploiting the Internet, open-source software, peer-to-peer technology, and other developments to write viruses and worms that invade computer systems in new ways, propagate around the world quickly, and wreak havoc to victims [55].

During a virus' lifetime, it normally goes through 4 stages [6]. These stages are:

- Dormant phase: The virus is idle, waiting to be activated.
- Propagation phase: Replicating itself to programs or disk.
- Triggering phase: The virus is activated to do its tasks by some event such as time, date, number of replications.
- Execution phase: The function in the virus is performed.

The detection of new viruses has become very difficult. Virus writing has gone to a new level where the viruses are polymorphic, uses changing encryption and decryption, and can infect both Windows and Linux platforms [56]. They infect machines not only by using their own code, but also by linking to and accessing malicious codes from newsgroups and Web sites. New software from different vendors is out now that requires users to define which actions they will and will not allow on a computer or network. Joe Hartman from Trend Micro [57] said: "If a machine suddenly starts to send hundreds of e-mails, the software will know that something is wrong and notify the user or system administrator".

2.5 What could be better for today's system?

The use of several security components can make a network more secure because misconfigurations or weaknesses in one component can be equalized by another component. Both firewalls and Intrusion Detection Systems deliver functionality that the other component cannot deliver. An Intrusion Detection System complements a

Chapter 2: Background

firewall by detecting what is going on in the network. A firewall is only a kind of fence, so it will not detect what's happening on the inside. Also, the Intrusion Detection System can catch attempts against the network that fails. This is important because it shows how big the threats from the outside are. Even more important, an Intrusion Detection System can catch attacks that pass the firewall, like for example Denial of Service attacks.

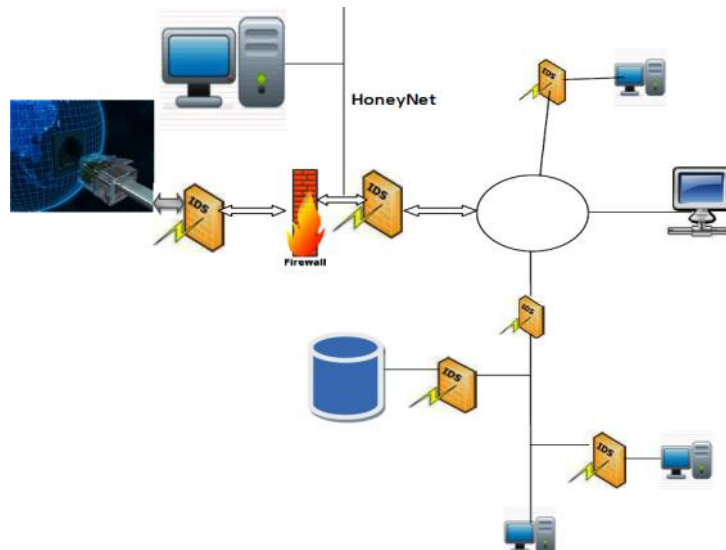


Figure 2.4 Firewall, IDS and Honey Net protecting a LAN

The idea with several security components is to establish a network perimeter and to identify all possible points of entry to the network. It is also recommended to protect sensitive servers with intrusion detection sensors on every server. The square boxes with magnifying glasses in them illustrate intrusion detection sensors. The Intrusion Detection Systems' sensors should be both host based and network based. Host based sensors are more useful for protecting critical servers, and network sensors are more useful for detecting abnormal traffic on the local network. The Central Manager receives reports from both the host based sensors and network based sensors, and process and correlates these reports to detect intrusions. The firewall protects the internal network from unwanted and unauthorized traffic from the outside. Sensors for the Intrusion Detection System should be placed on strategic places around the network. The first sensor is there to identify attack on servers in the demilitarized zone and attacks that are directed on the company's network. The second sensor is placed right after the firewall. This sensor serves to confirm secure configuration and operation of

Chapter 2: Background

the firewall, and it can also identify attacks that pass the firewall. The third sensor identifies any attacks from the inside against the local servers. The fourth and fifth sensors are sensors that protect single servers. These sensors can protect the servers against attacks from outside and has passed the firewall and the other sensors and against attacks from inside. All the sensors should be configured to report to one central Intrusion Detection System console. In addition to these security components, the use of Honeynets can also be very useful for a larger system. Here the system administrator could analyze the Honeynet and adjust their security components after gaining knowledge how the attackers behave during an intrusion.

Chapter 3: Neural Network

3.1 Introduction

The work on neural networks was inspired by the human brain. The human brain consists of neural networks. As a person learns new things, paths between different parts of the brain are created. If a person does not refresh his mind from time to time, these paths will eventually vanish.

The earliest work in neural computing goes back to the 1940's when McCulloch and Pitts introduced the first neural network computing model. In the 1950's, Rosenblatt's work resulted in a two-layer network, the perceptron, which was capable of learning certain classifications by adjusting connection weights. Although the perceptron was successful in classifying certain patterns, it had a number of limitations. The perceptron was not able to solve the classic XOR (exclusive or) problem. Such limitations led to the decline of the field of neural networks. However, the perceptron had laid foundations for later work in neural computing. In the early 1980's, researchers showed renewed interest in neural networks.

A neural network is a powerful data modeling tool that is able to capture and represent complex input/output relationships. This tool can acquire knowledge through learning of input data. Neural networks are essentially a network of computational units that jointly implement complex mapping functions [66]. It consists of a collection of processing elements that are highly interconnected and transforms a set of inputs to a set of desired outputs. Here are some of the characteristics of a neural network:

- The handling of data is done by many simple connected elements, called neurons.
- There is an interconnection between the connected neurons.
- A weight factor is associated to each connection in the network. This factor weights the signal that is sent from one neuron to another.
- Each neuron has its own task, and does some calculations.

The neural network consists of interconnected neurons. By modifying the connections between these nodes the network is able to adapt to the desired outputs [67]. Each neuron can be looked at as being a separate computer running its own program. The neuron computes the weighted sum of the inputs it gets from other neurons and gives an output as a single number to another neuron that performs the same task. The result of

Chapter 3: Neural Network

the transformation is determined by the characteristics of the neurons and the weights associated with the interconnections among them.

The neurons in a neural network are organized into layers. This is showed in Figure 3-1. The layers is divided into an input layer, hidden layer (there can be several hidden layers) and output layer. The inputs to the input layer are set by the environment. This layer does not play any significant role to the computing of the result. It only feeds information into the neural network. The hidden layers have no external connections; they only have connections with other layers in the network. The interaction between the hidden layers continues until some condition is satisfied. The outputs from the output layer are returned to the environment.

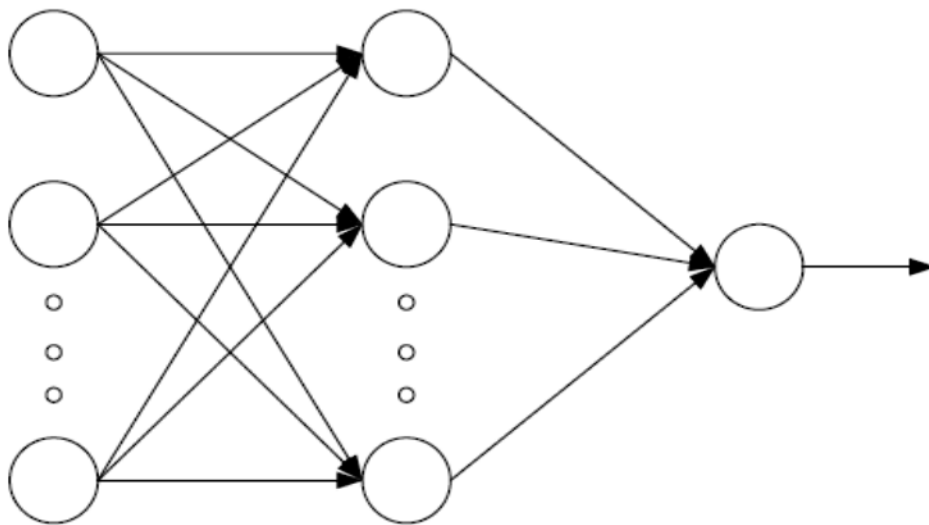


Figure 3.1: Neural Network model

Neural networks can be used to learn an anomaly Intrusion Detection System normal behavior. Initially, the neural network is trained with normal system behavior traces. Observed event streams are then fed into the network, and the neural network conducts an analysis of the information and provides a probability estimate that the data matches with the characteristics that is has been trained to recognize.

Traditional neural networks are unable to improve its analysis of new data until it is taken off-line and retrained using representative data that includes the new information. Today, neural networks are widely used in both software and hardware products around the world.

3.2 Neural Network Operation

Chapter 3: Neural Network

The output of each neuron is a function of its inputs. In particular, the output of the j th neuron in any layer is described by two sets of equations:

$$U_j = \sum(X_i \cdot w_{ij}) \quad [\text{Eqn 1}]$$

$$Y_j = F_{th}(U_j + t_j) \quad [\text{Eqn 2}]$$

For every neuron, j , in a layer, each of the i inputs, X_i , to that layer is multiplied by a previously established weight, w_{ij} . These are all summed together, resulting in the internal value of this operation, U_j . This value is then biased by a previously established threshold value, t_j , and sent through an activation function, F_{th} . This activation function has an input to output mapping as shown in Figure 3.2. The resulting output, Y_j , is an input to the next layer or it is a response of the neural network if it is the last layer. Neuralyst allows other threshold functions to be used in place of the sigmoid described here.

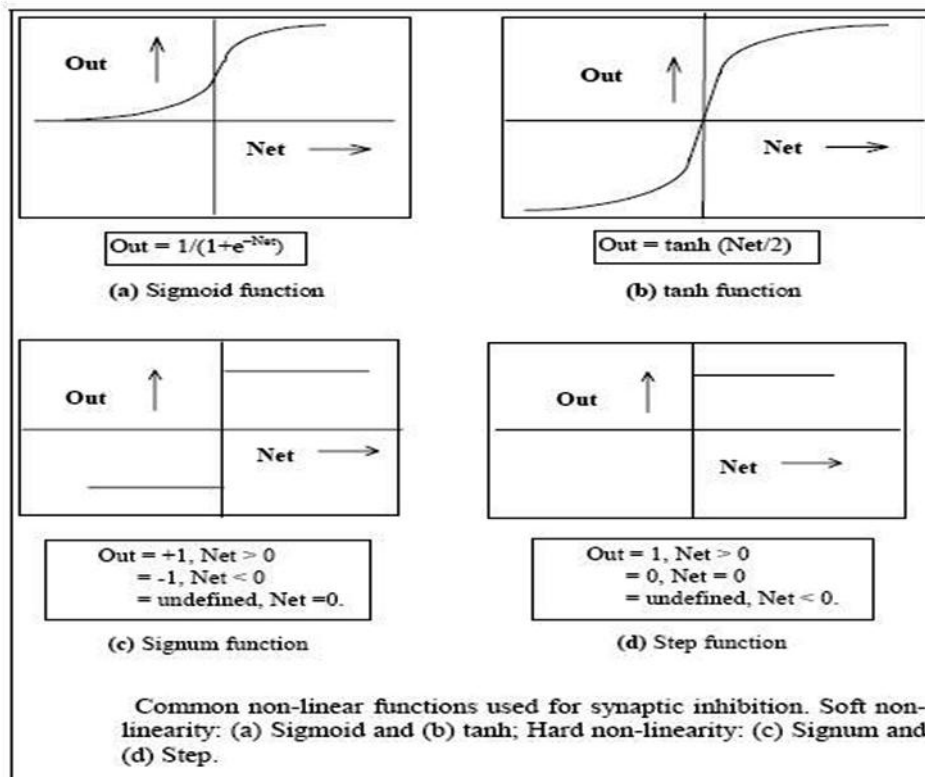


Figure 3.2: Sigmoid Function

In essence, Equation 1 implements the combination operation of the neuron and Equation 2 implements the firing of the neuron. From these equations, a predetermined set of weights, a predetermined set of threshold values and a description of the network

Chapter 3: Neural Network

structure (that is the number of layers and the number of neurons in each layer), it is possible to compute the response of the neural network to any set of inputs. And this is just how Neuralyst goes about producing the response. But how does it learn?

3.3 Neural Network Architectures

Humans and other animals process information with *neural networks*. These are formed from trillions of **neurons** (nerve cells) exchanging brief electrical pulses called action potentials. Computer algorithms that mimic these biological structures are formally called artificial neural networks to distinguish them from the squishy things inside of animals. However, most scientists and engineers are not using this formal and use the term *neural network* to include both biological and nonbiological systems.

Neural network research is motivated by two desires: to obtain a better understanding of the human brain and to develop computers that can deal with abstract and poorly defined problems. For example, conventional computers have trouble understanding speech and recognizing people's faces. In comparison, humans do extremely well at these tasks.

Many different neural network structures have been tried, some based on imitating what a biologist sees under the microscope, some based on a more mathematical analysis of the problem. The most commonly used structure is shown in Fig. 3.3. This neural network is formed in three layers, called the **input layer, hidden layer, and output layer**. Each layer consists of one or more **nodes**, represented in this diagram by the small circles. The lines between the nodes indicate the flow of information from one node to the next. In this particular type of neural network, the information flows only from the input to the output (that is, from left-to-right). Other types of neural networks have more intricate connections, such as feedback paths. The nodes of the input layer are passive, meaning they do not modify the data. They receive a single value on their input, and duplicate the value to their multiple outputs.

Chapter 3: Neural Network

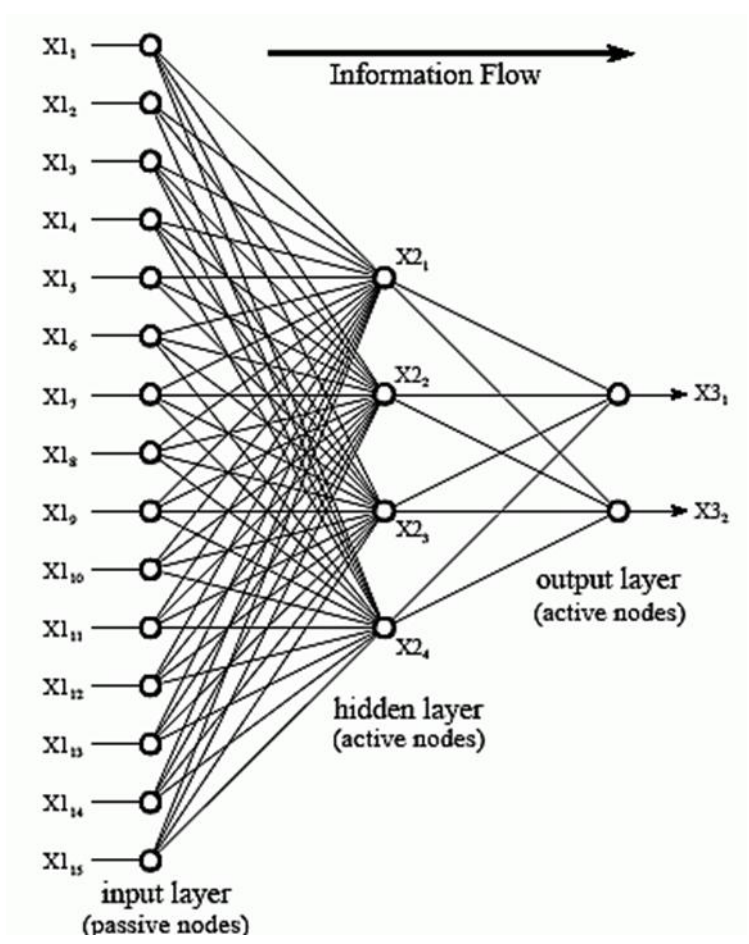


Figure 3.3: Neural network architecture

In comparison, the nodes of the hidden and output layer are **active**. This means they modify the data as shown in Fig. 3.4. The variables: $X1_1, X1_2 \dots X1_{15}$ hold the data to be evaluated (see Fig. 3.3). For example, they may be pixel values from an image, samples from an audio signal, stock market prices on successive days, etc. They may also be the output of some other algorithm, such as the classifiers in our cancer detection example: diameter, brightness, edge sharpness, etc.

Each value from the input layer is duplicated and sent to *all* of the hidden nodes. This is called a fully interconnected structure. As shown in Fig. 3.4, the values entering a hidden node are multiplied by weights, a set of predetermined numbers stored in the program. The weighted inputs are then added to produce a single number. This is shown in the diagram by the symbol, Σ . Before leaving the node, this number is passed through a nonlinear mathematical function called a *sigmoid*.

Chapter 3: Neural Network

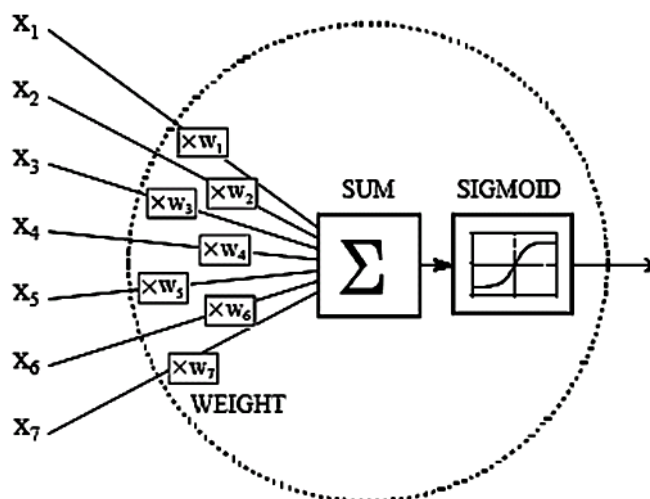


Figure 3.4 Neural network active nodes

This is an "s" shaped curve that limits the node's output. That is, the input to the sigmoid is a value between $-\infty$ and $+\infty$, while its output can only be between 0 and 1.

The outputs from the hidden layer are represented in the flow diagram (Fig 3.3) by the variables: $X2_1, X2_2, X2_3$ and $X2_4$. Just as before, each of these values is duplicated and applied to the next layer. The active nodes of the output layer combine and modify the data to produce the two output values of this network, $X3_1$ and $X3_2$.

Neural networks can have any number of layers, and any number of nodes per layer. Most applications use the three layer structure with a maximum of a few hundred input nodes. The hidden layer is usually about 10% the size of the input layer. In the case of target detection, the output layer only needs a single node. The output of this node is thresholded to provide a positive or negative indication of the target's presence or absence in the input data.

3.3.1 Multi Layer Perceptron

A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate output. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one as shown in figure 3.5. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network [68, 69]. MLP is a

Chapter 3: Neural Network

modification of the standard linear perceptron and can distinguish data that is not linearly separable [70].

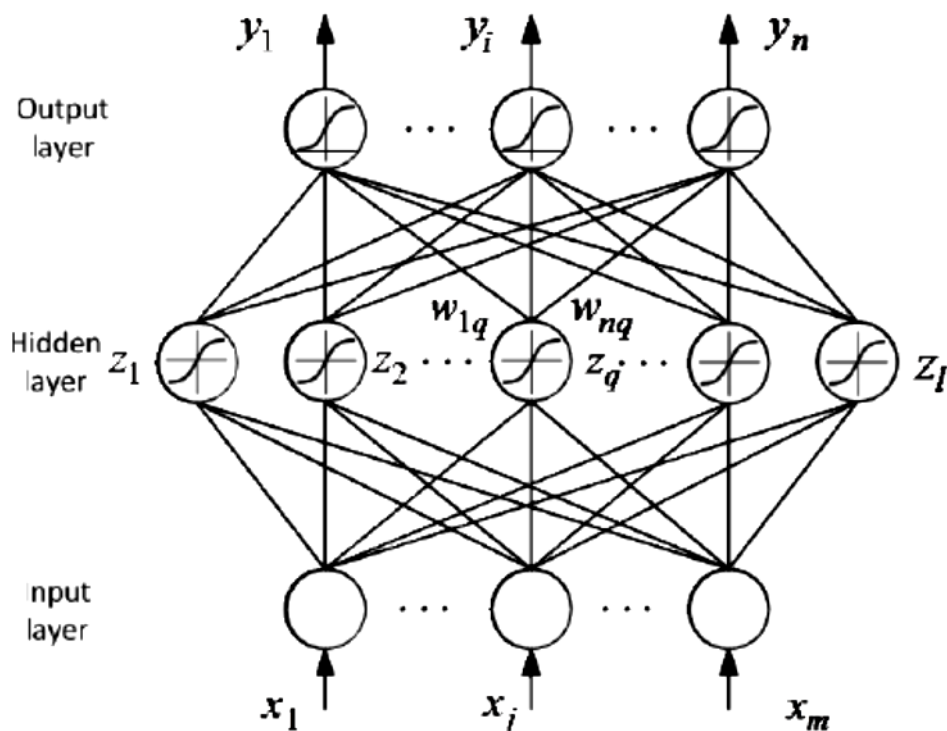


Figure 3.5: Multi Layer Perceptron, m – inputs (x_j), l – neurons in hidden layer (z_q), n – outputs (y_i), with different activation functions f for hidden and output layer

The training algorithm rule repetitively calculates an *error function* for each input and backpropagates the error from one layer to the previous one. The weights for a particular node (w_{ij}) are adjusted in direct proportion to the error in the units to which it is connected.

Let E_p - error function for pattern p

d_{pj} - target (desired) output for pattern p on node j

y_{pj} - actual output for pattern p on node j

w_{ij} - weight from node i to node j

The error function is defined to be proportional to the square of the difference of desired and actual output

$$E_p = \frac{1}{2} \sum_j (d_{pj} - y_{pj})^2$$

Chapter 3: Neural Network

The output from each unit j is determined by the non-linear transfer sigmoid function f_j

$$f(\text{net}) = \frac{1}{1 + e^{-k \cdot \text{net}}}$$

$$y_{pj} = f_j(\text{net}_{pj})$$

where net is activation of each unit j , for pattern p

$$\text{net}_{pj} = \sum_i W_{ji} Y_{pi}$$

and y_{pi} are outputs of previous layer.

The backpropagation algorithm implements weight changes that follow the path of steepest descent on a surface in weight space. The height of any point on this surface is equal to the error measure E_p . This can be shown by showing that the derivative of the error measure with respect to each weight is proportional to the weight change dictated by the delta rule, with a negative constant of proportionality, i.e.,

$$\Delta w_y = -\eta \frac{\partial E_p}{\partial w_{pj}}$$

The most used training algorithm is back propagation algorithm gradient descent (GDA) with disadvantage of slow training. In other hand Levenberg-Marquardt [71], [72] is one of the accurate algorithms and faster than GDA, but consumes more memory space.

3.3.2 Radial Basis Function Network

The radial basis function network (RBFN) [71] has the architecture of the instar-outstar model (Figure 3.6) and uses the hybrid unsupervised and supervised learning scheme, unsupervised learning in the input layer and supervised learning in the output layer.

Chapter 3: Neural Network

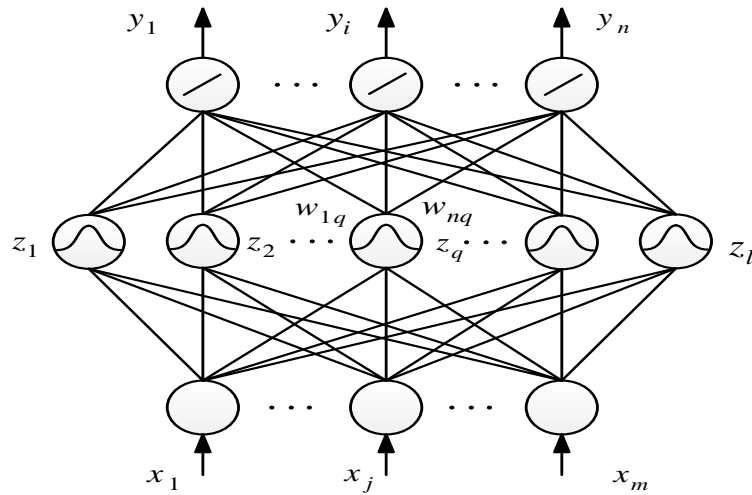


Figure 3.6: Radial basis function network structure with m inputs, L nodes in hidden layer, and n outputs

The purpose of the RBFN is to pave the input space with overlapping receptive fields. For an input vector \mathbf{x} lying somewhere in the input space, the receptive fields with centres close to it will be appreciably activated. The output of the RBFN is then the weighted sum of the activations of these receptive fields.

The RBFN is designed to perform input-output mapping trained by examples, pairs of inputs and outputs (\mathbf{x}, \mathbf{y}) . The hidden nodes in the RBFN have normalized Gaussian activation function

$$z_q = g_q(\mathbf{x}) = \frac{R_q(\mathbf{x})}{\sum_k R_k(\mathbf{x})} = \frac{\exp\left(-\frac{|\mathbf{x} - \mathbf{m}_q|^2}{2\sigma_q^2}\right)}{\sum_k \exp\left(-\frac{|\mathbf{x} - \mathbf{m}_k|^2}{2\sigma_k^2}\right)}$$

where \mathbf{x} is the input vector and z_q output of hidden layer. \mathbf{m}_q and σ_q are the mean (an m -dimensional vector) and variance of the q -th Gaussian function in hidden layer.

The output of the RBFN is simply the weighted sum of the hidden node output:

$$y_i = a_i\left(\sum_{q=1}^L w_{iq} z_q + \theta_i\right)$$

where $a_i(\cdot)$ is the output activation function, generally linear function, and θ_i is the threshold value.

Chapter 3: Neural Network

The weights in the output layer can be updated simply by using the delta learning rule (supervised learning). The unsupervised part of the learning involves the determination of the receptive field centres m_q and widths σ_q , $q = 1, 2, \dots, l$. The proper centres m_q can be found by unsupervised learning rules such as the vector quantization approach, competitive learning rules, or simply the Kohonen learning rule.

Another learning rule for the RBFN with node-growing capability is based on the orthogonal least squares learning algorithm [72]. This procedure chooses the centres of radial basis functions one by one in a rational way until an adequate network has been constructed, or maximal number of nodes is reached.

The RBFN offers a viable alternative to the two-layer neural network in many applications of signal processing, decision making algorithms, pattern recognition, control, and function approximation. It has been shown that the RBFN can fit an arbitrary function with just one hidden layer [73], but they cannot quite achieve the accuracy of the back-propagation network. Although, RBFN can be trained several orders of magnitude faster than the back-propagation network, and this is very important advantage in real or semi real time applications

3.3.3 RBF Networks vs. Multilayer Perceptrons

- **Similarities**
 - ✓ The RBF Networks as well as the Multilayer Perceptrons are *layered feedforward networks* that produce *nonlinear function mappings*;
 - ✓ They are both proven to be *universal approximators*;
- **Differences**
 - ✓ An RBF network has *only one hidden layer*, while MLP networks have one or more hidden layers depending on the application task;
 - ✓ The nodes in the hidden and output layers of MLP use the *same activation function*, while RBF uses *different activation functions* at each node (Gaussians parameterized by different centers and variances);
 - ✓ The *hidden and output layers* of MLP are *both nonlinear*, while only the *hidden layer* of RBF is *nonlinear* (*the output layer is linear*);
 - ✓ The activation functions in the RBF nodes compute the *Euclidean distance* between the input examples and the centers, while the activation

Chapter 3: Neural Network

functions of MLP compute *inner products* from the input examples and the incoming weights;

- ✓ MLP constructs *global approximations* while RBF construct *local approximations*.

3.4 Neural Network Learning

Learning in a neural network is called *training*. Like training in athletics, training in a neural network requires a coach, someone that describes to the neural network what it should have produced as a response. From the difference between the desired response and the actual response, the *error* is determined and a portion of it is propagated backward through the network. At each neuron in the network the error is used to adjust the weights and threshold values of the neuron, so that the next time, the error in the network response will be less for the same inputs.

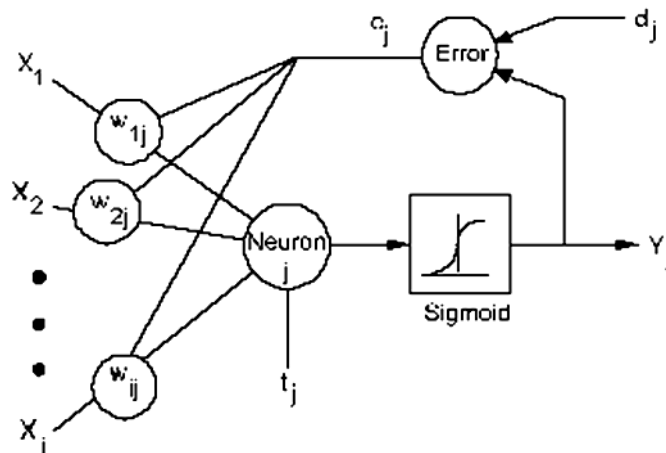


Figure 3.7: Neuron Weight Adjustments

This corrective procedure is called *backpropagation* (hence the name of the neural network) and it is applied continuously and repetitively for each set of inputs and corresponding set of outputs produced in response to the inputs. This procedure continues so long as the individual or total errors in the responses exceed a specified level or until there are no measurable errors. At this point, the neural network has learned the training material and you can stop the training process. Backpropagation starts at the output layer with the following equations:

Chapter 3: Neural Network

$$W_{ij} = \dot{W}_{ij} + LR \cdot e_j \cdot X_i \quad [\text{Eqn 3}] \text{ and}$$

$$e_j = Y_j \cdot (1 - Y_j) \cdot (d_j - Y_j) \quad [\text{Eqn 4}]$$

For the i th input of the j th neuron in the output layer, the weight w_{ij} is adjusted by adding to the previous weight value, w'_{ij} , a term determined by the product of a *learning rate*, LR , an error term, e_j , and the value of the i th input, X_i . The error term, e_j , for the j th neuron is determined by the product of the actual output, Y_j , its complement, $1 - Y_j$, and the difference between the desired output, d_j , and the actual output.

Once the error terms are computed and weights are adjusted for the output layer, the values are recorded and the next layer back is adjusted. The same weight adjustment process, determined by Equation 3, is followed, but the error term is generated by a slightly modified version of Equation 4. This modification is:

$$e_j = Y_j \cdot (1 - Y_j) \cdot \sum(e_k \cdot \dot{w}_{jk}) \quad [\text{Eqn 5}]$$

In this version, the difference between the desired output and the actual output is replaced by the sum of the error terms for each neuron, k , in the layer immediately succeeding the layer being processed (remember, we are going backwards through the layers so these terms have already been computed) times the respective pre-adjustment weights.

The learning rate, LR , applies a greater or lesser portion of the respective adjustment to the old weight. If the factor is set to a large value, then the neural network may learn more quickly, but if there is a large variability in the input set then the network may not learn very well or at all. In real terms, setting the learning rate to a large value is analogous to giving a child a spanking, but that is inappropriate and counter-productive to learning if the offense is so simple as forgetting to tie their shoelaces. Usually, it is better to set the factor to a small value and edge it upward if the learning rate seems slow. In many cases, it is useful to use a revised weight adjustment process. This is described by the equation:

$$w_{ij} = \dot{w}_{ij} + (1 - M) \cdot LR \cdot e_j \cdot X_j + M \cdot (\dot{w}_{ij} - \ddot{w}_{ij}) \quad [\text{Eqn 6}]$$

Chapter 3: Neural Network

This is similar to Equation 3, with a *momentum* factor, M , the previous weight, w'_{ij} , and the next to previous weight, w''_{ij} , included in the last term. This extra term allows for momentum in weight adjustment. Momentum basically allows a change to the weights to persist for a number of adjustment cycles. The magnitude of the persistence is controlled by the momentum factor. If the momentum factor is set to 0, then the equation reduces to that of Equation 3. If the momentum factor is increased from 0, then increasingly greater persistence of previous adjustments is allowed in modifying the current adjustment. This can improve the learning rate in some situations, by helping to smooth out unusual conditions in the training set.

As you train the network, the total error, that is the sum of the errors over all the training sets, will become smaller and smaller. Once the network reduces the total error to the limit set, training may stop. You may then apply the network, using the weights and thresholds as trained.

It is a good idea to set aside some subset of all the inputs available and reserve them for *testing* the trained network. By comparing the output of a trained network on these test sets to the outputs you know to be correct, you can gain greater confidence in the validity of the training. If you are satisfied at this point, then the neural network is ready for *running*. Usually, no backpropagation takes place in this running mode as was done in the training mode. This is because there is often no way to be immediately certain of the desired response. If there were, there would be no need for the processing capabilities of the neural network! Instead, as the validity of the neural network outputs or predictions are verified or contradicted over time, you will either be satisfied with the existing performance or determine a need for new training. In this case, the additional input sets collected since the last training session may be used to extend and improve the training data.

The learning methods in neural networks are classified into three basic types:

- Supervised Learning,
- Unsupervised Learning and
- Reinforced Learning

Chapter 3: Neural Network

3.4.1 Supervised Learning

Supervised learning is a machine learning technique that sets parameters of an artificial neural network from training data. The task of the learning artificial neural network is to set the value of its parameters for any valid input value after having seen output value. The training data consist of pairs of input and desired output values that are traditionally represented in data vectors. Supervised learning can also be referred as classification, where we have a wide range of classifiers, each with its strengths and weaknesses. Choosing a suitable classifier (Multilayer perceptron, Support Vector Machines, k-nearest neighbor algorithm, Gaussian mixture model, Gaussian, naive Bayes, decision tree, radial basis function classifiers,...) for a given problem is however still more an art than a science.

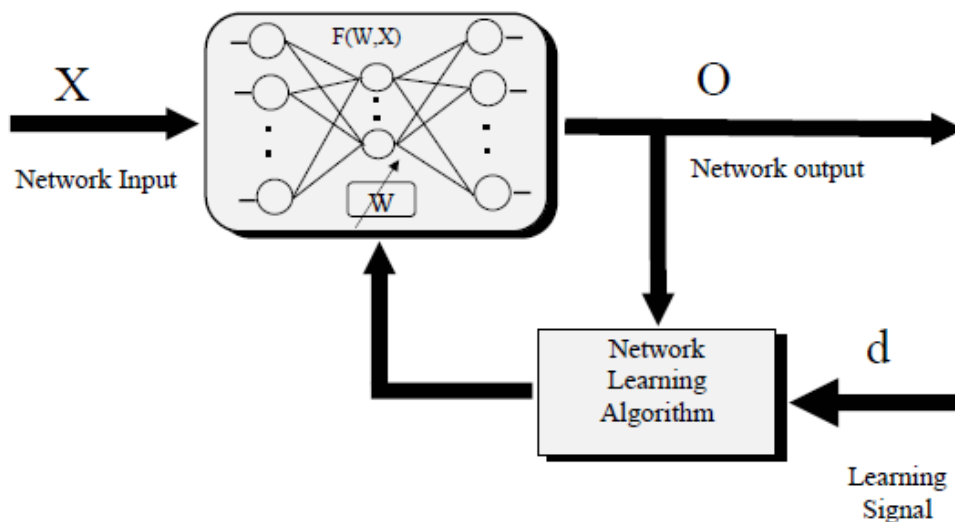


Figure 3.8: Supervised Learning

In order to solve a given problem of supervised learning various steps has to be considered. In the first step we have to determine the type of training examples. In the second step we need to gather a training data set that satisfactory describe a given problem. In the third step we need to describe gathered training data set in form understandable to a chosen artificial neural network. In the fourth step we do the learning and after the learning we can test the performance of learned artificial neural

Chapter 3: Neural Network

network with the test (validation) data set. Test data set consist of data that has not been introduced to artificial neural network while learning.

3.4.2 Unsupervised learning

Unsupervised learning is a machine learning technique that sets parameters of an artificial neural network based on given data and a cost function which is to be minimized. Cost function can be any function and it is determined by the task formulation. Unsupervised learning is mostly used in applications that fall within the domain of estimation problems such as statistical modeling, compression, filtering, blind source separation and clustering. In unsupervised learning we seek

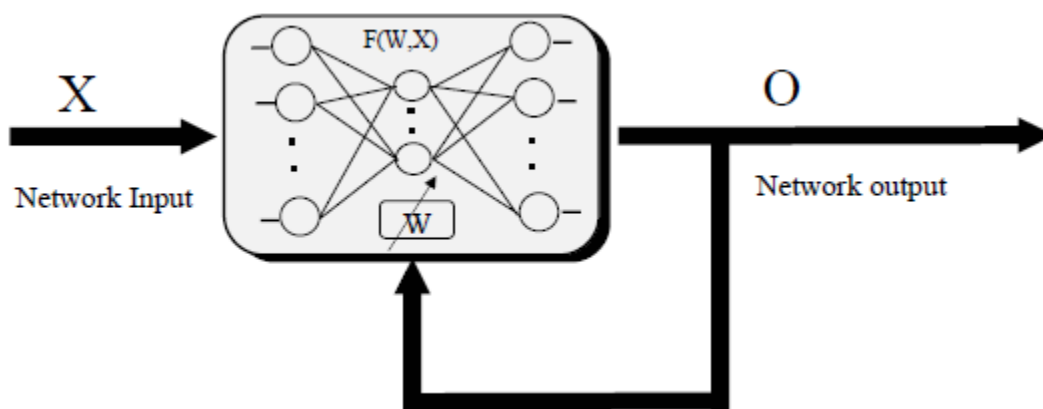


Figure 3.9: Unsupervised Learning

to determine how the data is organized. It differs from supervised learning and reinforcement learning in that the artificial neural network is given only unlabeled examples. One common form of unsupervised learning is clustering where we try to categorize data in different clusters by their similarity. Among above described artificial neural network models, the Self-organizing maps are the ones that the most commonly use unsupervised learning algorithms.

3.4.3 Reinforcement learning

Reinforcement learning is a machine learning technique that sets parameters of an artificial neural network, where data is usually not given, but generated by interactions with the environment. Reinforcement learning is concerned with how an artificial neural

Chapter 3: Neural Network

network ought to take actions in an environment so as to maximize some notion of long-term reward. Reinforcement learning is frequently used as a part of artificial neural network's overall learning algorithm. After return function that needs to be maximized is defined, reinforcement learning uses several algorithms to find the policy which produces the maximum return. Naive brute force algorithm in first step calculates return function for each possible policy and chooses the policy with the largest return. Obvious weakness of this algorithm is in case of extremely large or even infinite number of possible policies. This weakness can be overcome by *value function approaches* or *direct policy estimation*. Value function approaches attempt to find a policy that maximizes the return by maintaining a set of estimates of expected returns for one policy; usually either the current or the optimal estimates. These methods converge to the correct estimates for a fixed policy and can also be used to find the optimal policy.

Similar as value function approaches the direct policy estimation can also find the optimal policy. It can find it by searching it directly in policy space what greatly increases the computational cost. Reinforcement learning is particularly suited to problems which include a long-term versus short-term reward trade-off. It has been applied successfully to various problems, including robot control, telecommunications, and games such as chess and other sequential decision making tasks.

3.5 Back-Propagation Network

The most used learning rule algorithm in Neural Network, so what's BPN?

- A single-layer neural network has many restrictions. This network can accomplish very limited classes of tasks.

Minsky and Papert (1969) showed that a two layer feed-forward network can overcome many restrictions, but they did not present a solution to the problem as "how to adjust the weights from input to hidden layer"?

- An answer to this question was presented by Rumelhart, Hinton and Williams in 1986. The central idea behind this solution is that the errors for the units of the hidden layer are determined by back-propagating the errors of the units of the output layer. This method is often called the Back-propagation learning rule.

Chapter 3: Neural Network

Back-propagation can also be considered as a generalization of the delta rule for non-linear activation functions and multi-layer networks.

- Back-propagation is a systematic method of training multi-layer artificial neural networks.

Real world is faced with situations where data is incomplete or noisy. To make reasonable predictions about what is missing from the information available is a difficult task when there is no a good theory available that may to help reconstruct the missing data. It is in such situations the Back-propagation (Back-Prop) networks may provide some answers.

- A Back Propagation network consists of at least three layers of units:
 - An input layer,
 - At least one intermediate hidden layer, and
 - An output layer.
- Typically, units are connected in a feed-forward fashion with input units fully connected to units in the hidden layer and hidden units fully connected to units in the output layer.
- When a BackPropagation network is cycled, an input pattern is propagated forward to the output units through the intervening input-to-hidden and hidden-to-output weights.
- The output of a BackPropagation network is interpreted as a classification decision.
- With BackPropagation networks, learning occurs during a training phase.

The steps followed during learning are:

- ✓ Each input pattern in a training set is applied to the input units and then propagated forward.
- ✓ The pattern of activation arriving at the output layer is compared with the correct (associated) output pattern to calculate an error signal.
- ✓ The error signal for each such target output pattern is then back-propagated from the outputs to the inputs in order to appropriately adjust the weights in each layer of the network.
- ✓ After a BackPropagation network has learned the correct classification for a set of inputs, it can be tested on a second set of inputs to see how well it classifies untrained patterns.

Chapter 3: Neural Network

- An important consideration in applying BackPropagation learning is how well the network generalizes.

3.6 Applications of Neural Network

Neural Network Applications can be grouped in following categories:

■ Clustering:

A clustering algorithm explores the similarity between patterns and places similar patterns in a cluster. Best known applications include data compression and data mining.

■ Classification/Pattern recognition:

The task of pattern recognition is to assign an input pattern (like handwritten symbol) to one of many classes. This category includes algorithmic implementations such as associative memory.

■ Function approximation:

The tasks of function approximation are to find an estimate of the unknown function subject to noise. Various engineering and scientific disciplines require function approximation.

■ Prediction Systems:

The task is to forecast some future values of a time-sequenced data. Prediction has a significant impact on decision support systems.

Prediction differs from function approximation by considering time factor. System may be dynamic and may produce different results for the same input data based on system state (time).

3.8 Neural Network and Intrusion Detection System

The computational changes in the last several decades have brought growth to new technologies. One of these technologies is artificial neural networks (ANNs). Over the years, ANNs have given various solutions to the industry. Designing and implementing intelligent systems have become an important activity for the innovation and development of better products for human life. Examples might include the case of the

Chapter 3: Neural Network

Implementation of artificial life and giving solution to interrogatives, that linear systems are not able to resolve [74].

A neural network (NN) is an information processing system that is inspired by the way biological nervous systems, such as the brain, process information. It is composed of a large number of highly interconnected processing elements (PEs) working with each other to solve specific problems. Each processing element (or neuron) is basically a summing element followed by an activation function. The output of each PE (after applying the weight parameter associated with the connection) is fed as the input to all of the PEs in the next layer. The learning process is essentially an optimization process in which the parameters of the best set of connection coefficients (weights) for solving a problem are found and includes the following basic steps (Theodorios and Koutroumbas, 1999):- Present the neural network with a number of inputs (vectors each representing a pattern).

- Check how closely the actual output generated for a specific input matches the desired output.
- Change the neural network parameters (weights) to better approximate the outputs.

One of the projects dealing with the approach results in the system is called Hyperview Debar et al., [75]. It is a system that is built on two components. An ordinary expert system component has a task to monitor logs and, according to the defined policy, search the intrusions. It is a signature based IDS. Second component is a neural network that can observe the behavior of a user and send the alarm if the observed behavior is violated. This work shows how neural network can be used in combination with expert systems and improves intrusion detection qualities. In Ghosh and Schwartzbard [76], it is shown how neural networks can be employed for the anomaly and misuse detection. The works present an application of neural network to learn previous behavior since it can be utilized to detection of the future intrusions against systems. Experimental results indicate that neural networks are “suited to perform intrusion state of art detection and can generalize from previously observed behavior” according to the authors.

Horeis [77] describes and concludes that the combination of RBF and SOM is convenient to use as an intrusion detection model. They conclude that the “evaluation of human integration” is necessary to reduce the classification error. Experimental

Chapter 3: Neural Network

results are promising and show that RBF–SOM achieves, compared to RBF, similar or even better results. Lin et al. [78] design a new intrusion detection system based on the neural network NNID (neural network intrusion detector) and back-propagation algorithm. The experimental results show that NNID can be used and can identify users by what commands they use and how often. In Charron et al. [79], the work in using neural networks for detecting misuse of programs is described. The authors conclude that their work gives two distributions to the community. First one is a demonstration of how misuse of programs can be detected with help of neural networks and the second is that the result of their work shows “the benefit of applying anomaly detection to the process level such that an abnormal process behavior can be detected irrespective of individual users’ behavior”. Heywood et al. [80], describe an approach to dynamic intrusion detection using SOM. The authors estimate that “hierarchically built unsupervised neural network approach is able to produce encouraging results”. Binh Viet [81] presents a machine learning approach that can be used for the anomaly detection problem. SOM is, according to the authors, a powerful mechanism for modeling the network traffic. Ryan et al. [82] described an offline anomaly detection system (NNID) which utilized a back-propagation MLP neural network. The MLP was trained to identify users’ profile and at the end of each log session, the MLP evaluated the users’ commands for possible intrusions (offline). The authors described their research in a small computer network with 10 users. Each feature vector described the connections of a single user during a whole day. One hundred most important commands are used to describe a user’s behavior. They used a three layer MLP (two hidden layers). The MLP identified the user correctly in 22 cases out of 24. Cannady [83] used a three layer neural network for offline classification of connection records in normal and misuse classes. The system designed in this study was intended to work as a standalone system (not as a preliminary classifier whose result may be used in a rule-based system). The feature vector used was composed of nine features all describing the current connection and the commands used in it. A data set of 10,000 connection records including 1000 simulated attacks was used. The training set included 30% of the data. The final result is a two class classifier that succeeded in classification of normal and attack records in 89–91% of the cases. In yet another study Mukkamala, [84] the authors used three and four layer neural networks and reported results of about 99.25%

Chapter 3: Neural Network

correct classification for their two class (normal and attack) problem. Cunningham and Lippmann [85] used NNs in misuse detection. They used an MLP to detect Unix-host attacks by searching for attack specific keywords in the network traffic. Mehdi Moradi and Mohammad Zulkernine [86] present a NN approach to intrusion detection. A multi-layer perceptron is used for intrusion detection based on an offline analysis approach and applying the early stopping validation method on the proposed NN. Rachid Beghdad [87] aimed to determine which of the NN classifies well the attacks and leads to a higher detection rate of each attack. The paper focused on two classification types of records: a single class (normal, or attack), and a multiclass, where the category of attack is also detected by the NN. Five different types of NNs were tested: Multi-Layer Perceptron (MLP), Generalized Feed Forward (GFF), Radial Basis Function (RBF), Self- Organizing Feature Map (SOFM), and Principal Component Analysis (PCA) NN. Yuehui Chen et al., [88] proposed an IDS model based on a general and enhanced Flexible Neural Tree (FNT). Based on the predefined instruction/operator sets, the framework allows input variables selection. Over layer connections and different activation functions for the various nodes involved. Different groups used self-organizing maps (SOM) for intrusion detection, such as [89], [90], [91] and [92]. These works use SOM (Self Organizing Maps) and some variations to store data from the neural network training. The main idea of the artificial neural network approach for intrusion detection is the provision of an unsupervised classification method, which is fast and efficient for a large amount of data with many variables (source IP, destination IP, source port, destination port, size of packets, protocol, etc). One problem present in the artificial neural network approach is the time for training these networks, which is usually performed off-line. However, once trained, the time for analysis is considerably low. Works involving neural networks to detect intrusions show promising results, such as decrease in the false positive rates and improve in the detection rate compared to other anomaly based IDSs. However, IDSs that use neural networks face the difficulty of training with real traffic and real attacks. Samples of real traffic, may have some kind of malicious traffic (noise) not identified. The application of malicious traffic in the neural network training period (for normal traffic) can affect the value of weights of the neurons, causing errors in the process of detection (depending on the learning rate).

Chapter 4: Proposed Intrusion Detection System

4.1 Introduction

Attacks on our networks and server infrastructures are a growing source of concerns for network operators and users. They may be generated by both inexperienced and professional hackers, but in any case, attacks create unwanted traffic that can affect the performance and dependability of existing services. Therefore operators employ intrusion detection systems to identify and possibly filter suspicious traffic.

The constant increase in network traffic and the fast introduction of high speed (tens of Gbps) network equipment [11] make it hard to still employ traditional packet based intrusion detection systems. Such systems rely on deep packet payload inspection, which does not scale well. In high speed environments, approaches that rely on aggregated traffic metrics, such as *flow-based* approaches, show a better scalability and therefore seem more promising. The advantage of *flow-based* approaches is that only a fraction of the total amount of data needs to be analyzed.

A flow is defined as a unidirectional stream of packets that share common characteristics, such as source and destination addresses, ports and protocol type. In addition, a flow includes aggregated information about the number of packets and bytes belonging to the stream, as well as its duration. Flows are often used for network monitoring, permitting to obtain a real time overview of the network status; common tools for this purpose are Nfsen [93] and Flow scan [94], while the *de facto* standard technology in this field is Cisco Netflow, particularly its versions 5 and 9 [95,96]. The IETF IPFIX working group [97] is currently working on a standard for IP flow exporting, based on Netflow version 9.

Large networks, when creating flows, often apply *packet sampling* in order to make the approach even more scalable. In this case, only a percentage of the total number of packets passing through the monitoring point is considered in the flows. Statistical studies have been performed about correctness and precision of sampling strategies for Internet traffic [34] and high speed environments [98], as well as estimation of traffic flow characteristics from real sampled data [99]. These studies show that, despite the reduced amount of information, it is still possible to offer a correct statistical overview of the network status [34]. Packet sampling in flow creation is vastly deployed [96, 100]. In particular, NetFlow relies on *systematic* sampling, where only 1 out of every n packet is considered for the accounting (1: n).

Chapter 4: Proposed Intrusion Detection System

In the last years there has been an increasing interest in the application of flow-based techniques for anomaly and intrusion detection. The works of [101,102,103], which applies principal component analysis to traffic time series, and [104], which proposes a framework for network anemography, are examples of contributions in this field. Another example is provided by [105], which aims to detect worm spread in high speed network on a connection basis. In a similar environment, [106] addresses the problem of detecting DoS attacks and scans. In this case, the authors particularly focus on aggregated header information, as they can be exported by NetFlow (TCP flags). In addition, the presented approach is interesting because it explicitly addresses the problem of measure variation over time (with the use of value forecasting). In [107], the role of timely analysis of flow data is central. The author proposes a general purpose platform for parallel time-based analysis of flow information for attack detection, focusing in particular on DoS attacks (SYN-flood and web server overloading). From a network monitoring point of view, time series on flows, packets, and bytes are considered to be a useful tool: they permit to have a *dynamic* and *real time* overview of the network on the basis of the stream of information coming from the exporter [91, 90].

It is not surprising that intrusion detection (ID) has become an important research area in the last decade. A large number of ID techniques have been proposed and many of them have been implemented as prototypes or in commercial products. Moreover, the research community has recently focused on flow-based approaches. When proposing a new intrusion detection system (IDS), researchers usually evaluate it by testing it on labeled (or annotated) traffic traces, i.e., traffic traces with known and marked anomalies and incidents [108]. Labeled traces are important to compare the performance of diverse detection methods, to measure parameter effectiveness and to fine-tune the systems. Ideally, a labeled traffic trace should have the following properties:

It should be realistic (opposed to “artificial”), completely labeled, containing the attack types of interest and, not less importantly, publicly available. Despite the importance of labeled traces, research on IDS generally suffers of a lack of shared data sets for benchmarking and evaluation. Moreover, we have no knowledge of any publicly available *flow-based* traffic trace that satisfies all these criteria.

Chapter 4: Proposed Intrusion Detection System

Several difficulties prevent the research community to create and publish such traces, in first place the problem of balancing between privacy and realism. It is natural that the most realistic traces are those collected “in the wild”, for example at Internet service providers or in corporate networks. Unfortunately, these traces would reveal privacy sensitive information about the involved entities and hence are rarely published. On the other hand, artificial traces, i.e., traces that have not been collected but artificially generated, can avoid the problem of privacy but they usually require higher effort and deeper domain knowledge to achieve a realistic result. Moreover, labeling is a time consuming process: it could easily be achieved on short traces, but these traces could present only a limited amount of security events. Therefore, most publications use non-public traffic traces for evaluation purposes. The only notable exception is the well-known DARPA traces [109,110], which still are, despite their age, the only publicly available labeled datasets specifically created for intrusion detection systems evaluation.

4.2 Flow-Based Solutions

This section presents the state of art solution for each category of attack that can be detected using flows. In particular, our survey of Flow-Based Solutions shows that most of the researches focus on detecting Denial of Service attacks, Scans, and Worms.

4.2.1 Denial of Service

In a denial-of-service (DoS) attack, an attacker attempts to prevent legitimate users from accessing information or services. By targeting your computer and its network connection, or the computers and network of the sites you are trying to use, an attacker may be able to prevent you from accessing email, websites, online accounts (banking, etc.), or other services that rely on the affected computer.

The most common and obvious type of DoS attack occurs when an attacker "floods" a network with information. When you type a URL for a particular website into your browser, you are sending a request to that site's computer server to view the page. The server can only process a certain number of requests at once, so if an attacker overloads

Chapter 4: Proposed Intrusion Detection System

the server with requests, it can't process your request. This is a "denial of service" because you can't access that site.

An attacker can use spam email messages to launch a similar attack on your email account. Whether you have an email account supplied by your employer or one available through a free service such as Yahoo or Hotmail, you are assigned a specific quota, which limits the amount of data you can have in your account at any given time. By sending many, or large, email messages to the account, an attacker can consume your quota, preventing you from receiving legitimate messages.

In a distributed denial-of-service (DDoS) attack, an attacker may use your computer to attack another computer. By taking advantage of security vulnerabilities or weaknesses, an attacker could take control of your computer. He or she could then force your computer to send huge amounts of data to a website or send spam to particular email addresses. The attack is "distributed" because the attacker is using multiple computers, including yours, to launch the denial-of-service attack.

The work of Gao et al. [106] approach the problem of Denial of Service detection by means of aggregate flow measures accounted in appropriate data structures; named sketches (figure 4.1). A sketch is originally a one-dimensional hash table suitable for fast storage of information [111]: it counts occurrences of an event. Sketches permit to statistically characterize how the traffic varies over time. An anomaly-based engine triggers alarms based on a statistical forecast of the values the sketches are storing: a sharp variation from the expected forecast values is flagged as an anomaly. Gao et al. developed a prototype that receives exported flows from a netflow enabled router in real time. A similar approach is proposed by Zhao et al.[112], in this approach a data streaming algorithm is used to filter part of the traffic and identify IP addresses that show an abnormal number of connections. The authors consider both the case in which a host is the source of an abnormal number of outgoing connections, as well as the case in which a host is the destination of an unusual number of connection attempts. The first case it is to match a scanning host, while the second is used for detecting DoS victims. The method is based on 2D hash tables, clearly resembling the contributions of Gao et al. [106] and Li et al. [113]. Zhao et al. apply a flow sampling algorithm. Sampling reduces the amount of data to be processed and significantly raises the processing speed.

Chapter 4: Proposed Intrusion Detection System

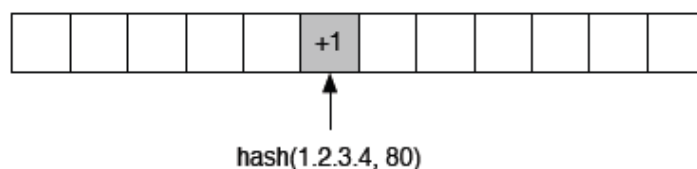


Figure 4.1: Example of sketch.

Kim et al. [114] describe several different types of DoS attacks in terms of traffic patterns. A traffic pattern is an attack signature expressed in terms of the number of flows and packets, the flow and packet sizes, as well as the total bandwidth used during the attack. The authors present as example the pattern differences between instances in the class of “flooding attacks”: SYN Flooding (exploiting the resource exhaustion in old TCP stack implementation in presence of half open TCP connections), ICMP flooding (provoking ICMP replies from an unaware network towards the victim) and UDP flooding (a stream of UDP packets aiming to exhaust the resource on the victim and possibly also the connection bandwidth towards the victim). The attack pattern produced by a SYN Flooding attack is characterized by a large flow count; yet small packet counts, as well as small flow and packet sizes and no constraints on the bandwidth and the total amount of packets. The pattern is significantly different from the one generated by an ICMP or UDP flooding attack, in which we observe large bandwidth consumption and intensive packet transfer. Kim et al. clearly identify the metrics they are interested in and formalize them into detection functions, which give the likelihood of an observed traffic sequence to be malicious. In the context of DoS monitoring and detection, it is important to cite also the work of Munz et al. [107], who propose a general platform for DoS detection. Attention must also be given to the work of Lakhina et al. [115, 116, 103, and 102].

4.2.2 Scans

Port Scanning is one of the most popular reconnaissance techniques attackers use to discover services they can break into. All machines connected to a Local Area Network (LAN) or Internet run many services that listen at well-known and not so well known ports. A port scan helps the attacker find which ports are available (i.e., what service might be listening to a port). Essentially, a port scan consists of sending a message to each

Chapter 4: Proposed Intrusion Detection System

port, one at a time. The kind of response received indicates whether the port is used and can therefore be probed further for weakness.

Due to their nature, scans can easily create a large number of flows, since the attacker may contact several different destination hosts using many source or destination ports.

There are three categories of scans:

- Horizontal scan: a host scanning a specific port on many destination hosts.
- Vertical scan: a host scanning several ports on a single destination host.
- Block scan: a combination of both.

Figure 4.2 shows the possible scan categories, displaying on the x-axis the IP addresses and on the y-axis the victim destination ports. Scans have generally been investigated by considering their most evident characteristic; the scanning source shows an unnaturally high number of outgoing connections, Zhao et al. [112]. Looking at host behavior from an incoming/outgoing connections perspective allows addressing DoS and scan attacks as different faces of the same problem: hosts with a suspicious and unusual fan-in/out.

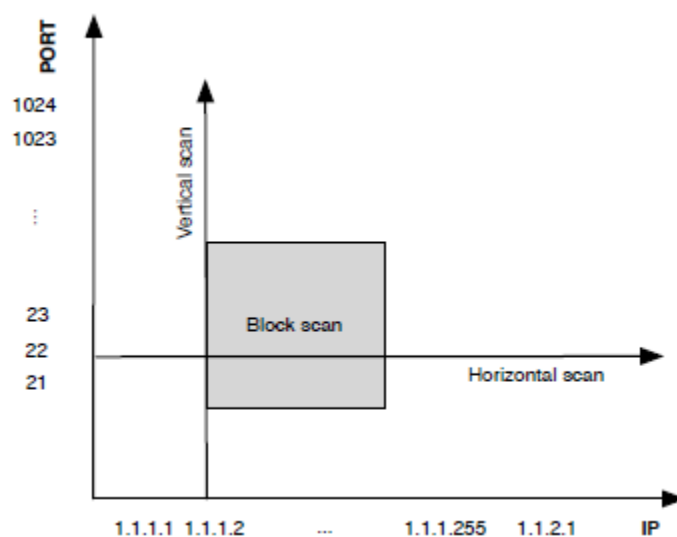


Figure 4.2: Categories of scans [117].

In the same manner, Kim et al. [114] describe a scan in terms of traffic patterns, the authors differentiate between network (horizontal) scans and host (vertical) scans. Li et al. [113] extend the approach of Gao [106], he is introducing 2D sketches (figure 4.3), a more powerful extension of the original ones.

Chapter 4: Proposed Intrusion Detection System

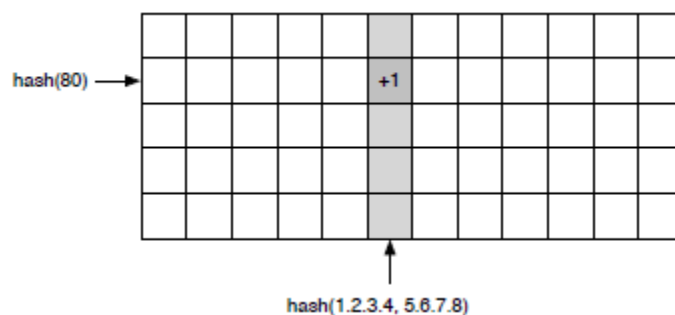


Figure 4.3: Example of 2D sketch, as in [113].

2D sketches are suitable not only for DoS detection, but also for scan detection. The authors hash a different key for each dimension of the sketch, improving in this way the overall detection capabilities of the system. Wagner et al. [118] propose to use the probabilistic measure of entropy to disclose regularity in connection-based traffic (flows). Entropy has been introduced in Information Theory in 1948 [119].

Entropy is related to loss-less data compression: the theoretical limit of the compression rate of a sequence of bits is the entropy of the sequence. Starting from this well-known result, Wagner et al. created an efficient analysis procedure based on compression of sequences of network measurements. They observe that, in the case of a scanning host, the overall entropy in a specific time window will change. In particular, the presence of many flows with the same source IPs (the scanning host) will lead to an abrupt decrease of the entropy in the distribution of the source IP addresses. At the same time, the scanning host will attempt to contact many different destination IPs on different ports, generating an increase in these entropy measurements. The combined observation of multiple entropy variations helps in validating the presence of an attack.

4.2.3 Worms

A worm is a computer program that has the ability to copy itself from machine to machine. Worms use up computer processing time and network bandwidth when they replicate, and often carry payloads that do considerable damage. Worm behavior is usually divided into a target discovery phase (the worm explores the network in order to find vulnerable systems) and a transfer phase (the actual code transfer takes place) [120, 121]. Code Red [122] and Sapphire/Slammer [123] are examples of this mechanism.

Chapter 4: Proposed Intrusion Detection System

Flow-based detection systems usually focus on the target discovery phase, since the transfer of malicious code cannot easily be detected without analyzing the payload. In many cases, worm detection can be similar to scan detection, and many researchers use the same approach for both threats. The approach adopted by Wagner et al. [118], for example, can naturally be extended to worms, as well as the ones of Gao et al. [106] and Zhao et al. [112]. Dubendorfer et al. [105] and Wagner et al. [124] attempt to characterize the host behavior on the basis of incoming and outgoing connections. The proposed algorithm assigns the hosts of a network to a set of predefined classes: the traffic class, the connector class and the responder class. The traffic class includes hosts that send more traffic than what they receive. Hosts that show an unusual high number of outgoing connections are part of the connector class. Finally, hosts involved in many bidirectional connections belong to the responder class. The definition of these classes is such that only suspicious hosts will belong to them. In the proposed model, a host can also belong to one or more classes. Figure 4.4 describes the three classes (sets) and their possible intersections. The method periodically checks the status of the hosts of an entire network. Massive changes in the cardinality of one or more classes are an indication of a worm outbreak. The authors validate their approach on fast spreading worms such as Witty and Blaster.

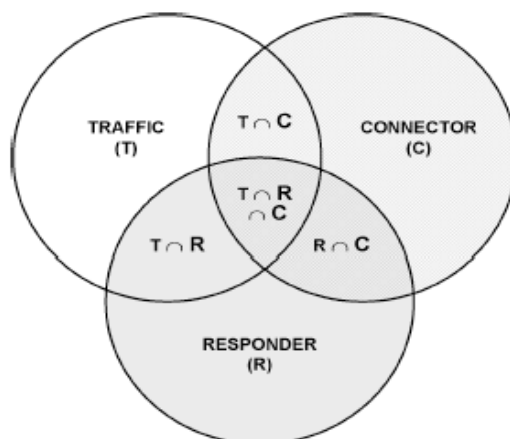


Figure 4.4: Host classes and their intersections.

A different approach is taken by Dressler et al. [125]. The authors exploit the correlation between flows and honeypots logs. In this case, the need for a ground truth, i.e., a

Chapter 4: Proposed Intrusion Detection System

trusted source of information for the system validation, made the authors rely on a honeypot. In this way, deploying at the same time a honeypot, a flow monitor and a data collection database, it is possible to carefully identify worm flow-signatures, i.e., a sequence of connections and flow related information about the scanning and transmitting behavior of a worm.

4.3 Proposed Approach

We approach the problem of traffic characterization by mean of flow based traffic. Since flows carry no payload, a single flow will in general not provide enough information to prove that an attack is ongoing. We believe, however, that attacks, or more generically, anomalies can be characterized looking at the evolution of flow traffic over time, as presented in flow-based traffic. Flows offer diverse metrics for building IDS. Some are directly derived from the definition of flow, such as the number of different accessed ports in a time bin. We concentrate on the number of flows, packets and bytes per time bin. A time bin can have duration from a millisecond to several minutes. The main key point for all researchers is to develop an effective intrusion detection system that has ability to detect known and unknown attacks with few false alarms.

For our work on intrusion detection we have used a two stages backpropagation (BP) neural network. And we have used the NetFlow data set for training and testing the neural network. Our research approach is shown in Figure 4.5:

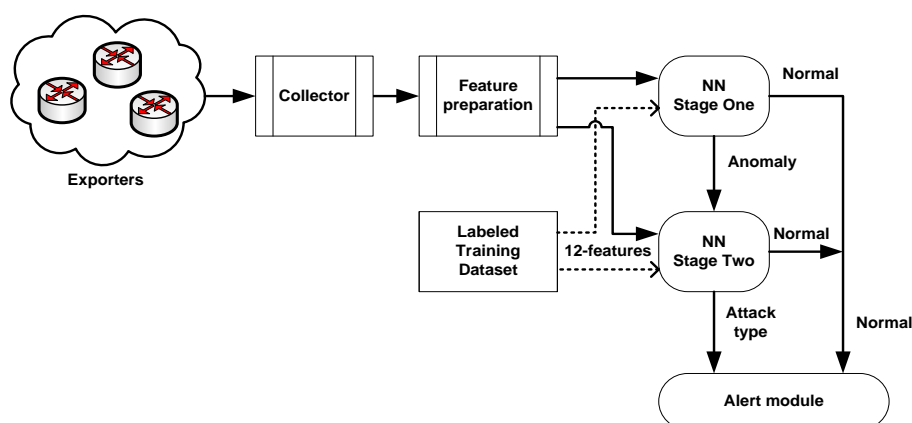


Figure 4.5 Proposed approaches

Chapter 4: Proposed Intrusion Detection System

Our proposed approach for intrusion detection and classification (Figure 4.6) consists of the following five main modules:

1. Flow collector module,
2. Feature preparation module,
3. Anomaly detection module (NN stage one),
4. Detection and classification module (NN stage two),
5. Alert module.

NetFlow enabled routers are considered as external devices which permanently monitor network traffic, account statistics, and export flow-data to our system according to Cisco NetFlow [19] or similar protocols.

4.3.1 Flow Collector Module

The main operation of this module is to collect flow-data exported from one or several exporters. The received data need to be recognized by protocol and version (for instance NetFlow version 5 or 9, J-flow or IPFIX) and transform into an internal format. The flows are periodically extracted and collected to NetFlow collector server. The collecting period can be configured according to the flow time-out in order to aggregate flow information during a predetermined time. A typical value is in the range of minutes. Figure 4.6 shows this exporting/collecting process. These data are constantly being sent to the Feature preparation module.

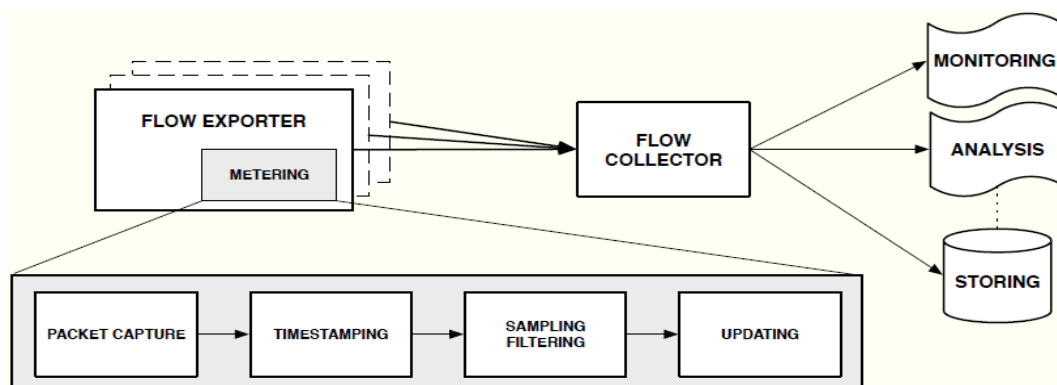


Fig. 4.6: IP Flow exporting and collecting architecture [21, 24]

Chapter 4: Proposed Intrusion Detection System

4.3.2 Feature Preparation Module

In order to train the neural network, we must first try to find the most features that can help for detecting intrusions attacks networks. Our proposed methodology is to have two stages neural network which give more reliability on detecting and classifying attacks, therefore the training features are different from one stage to another. The Feature preparation module receives and processes flow-data sent from the Flow collection module. The main function is to prepare the features that are important for anomaly intrusion detection and classification modules. The features are combined in two groups: *7-tuples* and *12-tuples* that are passed to stage one and stage two detection modules respectively. Section 4.3.3 gives a more detailed explanation of the stage one features and section 4.3.4 demonstrate the added features for stage two modules. The selected features for both stages are suitable only for the selected attack in our study, and many other attacks have deviations of these features. Preprocessing must be done on all selected features before passing them to the detection modules; this phase involves normalizing all features by mapping all the different values for each feature to [0, 1] range.

4.3.3 Anomaly Detection Module

The reason for having two stage neural networks is to have fast and accurate detection system, the purpose of stage one is to provide information about the existence or not of an anomaly at low cost anomaly detection mechanism. The selected features for stage one is:

1. **Average Flow Size:** it provides a useful hint for anomalous events, such as port scan, and it is typically very small in order to increase the efficiency of attacks.
2. **Average Packet Size:** another factor is the size of each packet in the flow; low average size can be a sign of anomaly. For example, in TCP flooding attacks, packets of 120 bytes are typically sent.
3. **Average Packet Number:** one of the main features of DoS attacks is the source IP spoofing, which makes the task of tracing the attacker's true source very difficult. A side effect is the generation of flows with a small number of packets,

Chapter 4: Proposed Intrusion Detection System

i.e. about 3 packets per flow. This differs from normal traffic that usually involves a higher number of packets per flow.

4. Number of different flows to the same Destination IP: This feature counts the number of flows to the same destination IP address. A high number of flows could mean a flood attack or a port scan attack.
5. Number of flows to different Destination Ports: also it has influence on detecting attacks. An abnormally large number of different destination ports means that the system is probably under attack (port scan attack).
6. Land: this feature is responsible for checking whether there is a land attack in the network or not.(i.e.SrcIP=DestIP,SrcPort=DestPort)
7. SYN - SYN/ACK: this feature was used by many researchers to detect DoS Attack, by comparing the numbers of SYN and SYN/ACK packets that a host receives and returns respectively. Under normal conditions, the two numbers should be balanced since every SYN packet is answered by a SYN/ACK packet. Consequently, a high number of unanswered SYN packets is an indication of ongoing SYN flood.

Multilayer perceptron with resilient backpropagation was used for training the stage one neural network which is the most used and reliable neural network algorithm. The number of hidden layers, and the number of nodes in the hidden layers, was also determined based on the process of trial and error. The Neural Network was trained with the labeled training data which contains attack records, and nonattack records. Once the training was over, the weight value is stored to be used in recall stage.

4.3.4 Detection and Classification Module

The presence of second stage neural network is to ensure whether the captured anomaly traffic is a real attack or not and also trying to classify the type of attack hitting the network. More features were used in stage two in order to allow correct and accurate detection, and classification of anomalies. There are five new features used in stage two in addition to the seven features that have been used in stage one (Anomaly Detection Phase), that makes them 12 features in total. The added features are listed below:

Chapter 4: Proposed Intrusion Detection System

- 1- Number of flows from the same source IP: attacker can send for example ICMP ping packets to every possible address within a subset and wait to see which machine respond.
- 2- Number of flows from different source IP: IP spoofing is widely used by attackers to attack the networks. A high number of different IP addresses to the same destination address within a short period of time are a strong sign for attack (DoS/ DDoS attack).
- 3- Number of flows to the same Destination Port: in some cases the attacker sends GET request to some ports only (ex. Port 80) to crash the server.
- 4- Number of flows from different source Port: As IP spoofing is generated by DDoS attack; ports can also be changed during an attack at random.
- 5- Protocol type (TCP, UDP, and ICMP): knowing the protocol type in combination to the all previous features can help to determine the type of attack.

All added features are playing important role in detection and classification of the attack. Multi-layer feed forward networks (MLP), and RBF neural networks are used in this stage. We chose these two methods based on prior research and relevance to our problem context. MLP neural networks have been widely used for data mining and have also been found to be effective in intrusion detection systems. Two training algorithms were used (Resilient back propagation, and Levenberg-Marquardt) for training our neural network. The Neural Network was trained with the labeled training dataset that contains attack records, and nonattack records. The output from this neural network stage is the attack type or nonattack traffic.

4.3.5 Alert Module

This is the final stage of the proposed system. This stage involves identifying the events that occurred whether abnormal or not, then sending the required signals to the administrator in order to take appropriate action, and quick decision is taking to stop the intruder to penetrate to the computer network.

Chapter 4: Proposed Intrusion Detection System

4.4 Training Dataset

When researchers propose new IDS, they usually evaluate it by testing it on labeled traffic traces (dataset), i.e., traffic traces with known and marked anomalies and incidents [126]. Several questions remain open such as the way training data should be organized to achieve optimal classification results or, more abstract, what characterizes good training data. Furthermore, researchers are often confronted with typical problems when creating or evaluating training sets. There are two types of training dataset, Realistic and Artificial Training Data.

In terms of network intrusion detection, realistic sets are based on real world network traffic which is captured and later labeled by a human expert or a machine. So, realistic sets contain training samples which originally were productive network traffic. In contrast, artificial sets are based on artificially generated network traffic. So the simulated data was never part of real-world traffic. The labeling is straightforward since the researcher created the traffic himself and thus has full control over the generation of malicious and benign traffic.

Usually researchers favor realistic over artificial data sets, provided that both data sets are comparable in their content. The reason is that artificial data sets might be flawed in some disguisedly way and hence do not allow proper generalization. An example of artificial data set is DARPA data set [127]. On the other hand, the publication of realistic data sets is often difficult or even impossible since privacy laws complicate publication of sensitive data such as IP addresses. Thus, realistic data sets often have to stay in the hands of the researchers who created them although the research community lacks comprehensive data sets. Section 4.4.1, gives an overview of the existing labeled data sets for intrusion detection and section 4.4.2 describes our created NetFlow Datasets.

4.4.1 Existing Dataset

A flow based intrusion detection system requires high-quality training and testing datasets. Unfortunately, there are few labeled datasets for evaluation of IDSs exists and

Chapter 4: Proposed Intrusion Detection System

are publicly available and all of them are not flow-based dataset except the work of Sperotto [132]:

- The DARPA 1998 and DARPA 1999 data sets developed by the MIT Lincoln Labs and sponsored by the US Defense Advanced Research Projects Agency. The DARPA data sets [128, 129] consist of artificial background traffic, which simulates the normal network usage of an air force base, combined with malicious attack traffic.
- The KDD99 data set [130] and the NSL-KDD data set [131]. The KDD99 data set is build upon the traffic in the DARPA 1998 data set, but uses an extended labeling. The NSL-KDD data set, on the other hand, is a reduced version of KDD99 that aims to avoid record redundancy in the data set.
- A recent attempt to propose a database of labeled traffic for IDSs comparison and evaluation is the work of Sperotto et al. [132]. This work is the first contribution on flow-based labeled dataset intended for evaluating and training network intrusion detection system.

4.4.2 Flow-Based DataSet

Our approach is NetFlow based intrusion detection system, and in order to train the neural network labeled dataset must be used that, and considering the current situation, all research on IDS generally suffers from a lack of shared data sets for benchmarking and evaluation. Several difficulties prevent the research community to create and publish such traces, in the first place the problem of balancing between privacy and realism. It is natural that the most realistic traces are those collected at Internet service providers or in corporate networks. Unfortunately, these traces would reveal privacy sensitive information about the involved entities; hence, they are rarely published. On the other hand, artificial traces, i.e., traces that have not been collected but artificially generated, can avoid the problem of privacy but they usually require deeper domain knowledge to achieve a realistic result. Therefore, most publications use non-public traffic traces for evaluation purposes. Moreover, we have no knowledge of any publicly available labeled flow-based traffic trace.

Chapter 4: Proposed Intrusion Detection System

Considering the fact that the previous works commonly use DARPA dataset as a trusted labeled dataset for intrusion detection research, we built our NetFlow dataset as a subset of DARPA dataset. Since DARPA dataset is in form of TCP dump data, therefore we created flows from the raw DARPA dataset using a modified version of softflowd [133]. In our dataset, a flow closely follows the NetFlow v5 definition and has the following form:

$$F = \{IPsrc, IPdst, Psrc, Pdst, Pckts, Octs, Flags, Protcl, Tstart, Tend\}$$

It represents the unidirectional communication from the source IP addresses IPsrc and port number Psrc to the destination IP address IPdst and port number Pdst, using protocol type Protcl. The Pckts and Octs give the total number of packets and octets transferred during this communication. The field Flags is related to the TCP header flags which are computed as a binary OR of TCP flags in all packets of the flow. The start and end time of the flow are given by Tstart and Tend respectively, in millisecond resolution. The extracted flows are labeled according to the log file of DARPA dataset and used to prepare all selected features, which used to train NN1 and NN2.

Chapter 5: Experimental Results

This chapter presents the experimental results obtained by using two neural network stages based research methodology proposed in the previous chapter. The experiments were conducted in three parts. The first part is to train both stages of neural networks and to find out the optimum number of nodes in hidden layers. The second part of experiment was conducted to test detection module (neural network stage one). The third part of experiments was conducted to test the detection and classification module (neural network stage two) and to see how many percent of the detection rate for normal traffic and the attacks that were detected and classified correctly.

5.1 Training and Testing Proposed System

The experiments were performed in MATLAB, using neural network toolbox and our created NetFlow data set as shown in table 5.1, which implements several training algorithms including Resilient Backpropagation, Radial Basis Function net, and Levenberg-Marquardt.

Table 5.1. Used Data set

Total Data set	Normal Records	Attack Records
145438	48586	96852

In the experimental stages we have used different number of iterations and hidden layers to determine the level of training. This test has been done to find out when the neural network was trained properly to detect attacks. This test has also provided the background for choosing the number of hidden layers and iterations for the training of the neural network for the last experiments.

The experiments show that Levenberg-Marquardt is the best training algorithm because it takes less time, low number of epochs and has good performance and high accuracy. The Detection Rate (DR) and False Positive rate (FP) have been calculated for different scenarios according to the following formulas:

$$DR = \frac{\text{Number of detected patterns}}{\text{Total number of patterns}} * 100[\%]$$

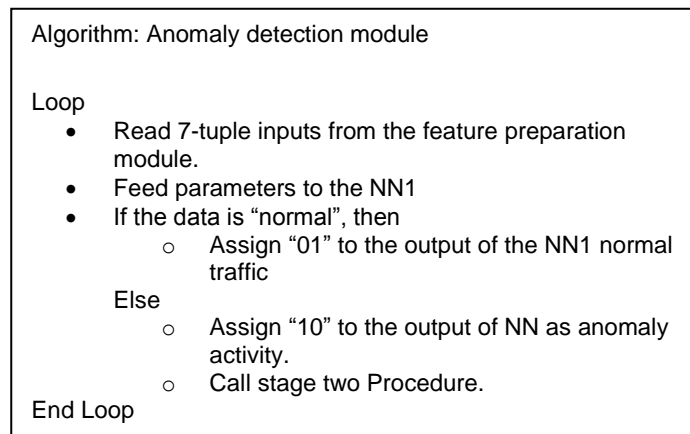
False positive means if it is normal and the system detected as attack and false positive rate can be calculated by the following equation:

Chapter 5: Experimental Results

$$FP = \frac{\text{Number of normal classified as attack}}{\text{Total number of normal records}} * 100[\%]$$

5.2 Anomaly Detection module Test and Results

Anomalies in our system are defined as unusual activities in the network. The purpose of this module is to find out such activities using a small number of features extracted from NetFlow raw data. For the neural network that was used in stage one the algorithm below is a simplified general description of the detection process.



The number of input nodes of the NN1 corresponds to the number of the selected features of the NetFlow dataset for the first stage (7 Features). The implemented NN1 includes one input layer, one hidden layer and an output layer of 2 nodes (01 as normal traffic, and 10 as anomaly traffic). The number of nodes in the hidden layers has been determined based on the back propagation (BP) computation process and the process of trial and error. Table 5.2 shows the training, validation, and testing results of anomaly detection module (Stage One).

Table 5.2 Results of Anomaly Detection phase

Training Algorithm Parameters	Resilient Backpropagation Test 1	Levenberg-Marquardt Test2	Radial Basis Function Net Test 3
Training dataset	101806		
Validation Data	21816		
Testing set	21816		
Hidden Layer	50	50	20
Number of detected attacks(14527)	13468	13684	13234
Number of detected traffic as normal(7289)	6757	6866	6640
Detection Rate	92.7%	94.2%	91.1%
False positive Rate	3.6%	3.4%	5.1%

Chapter 5: Experimental Results

As shown in table, 101806 records were used for training NN1, 21816 records were used for testing the stage one neural network. It contains 14527 records as attack records and 7289 as normal records, Figure 5.1, and figure 5.2 shows respectively the detection rate and performance of detection module (stage one).

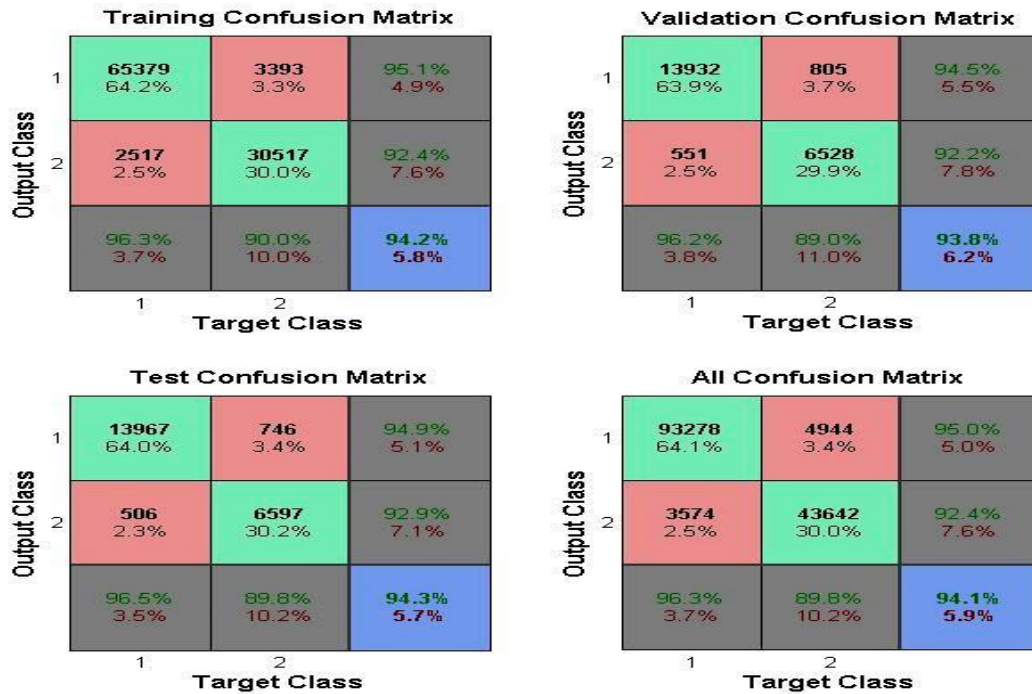


Figure 5.1 Detection rate of stage one neural network

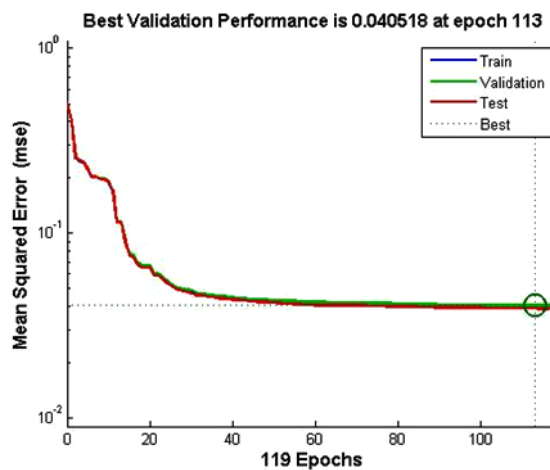


Figure 5.2 performance of the anomaly detection module

Chapter 5: Experimental Results

5.3 Detection and Classification Module Test and Results

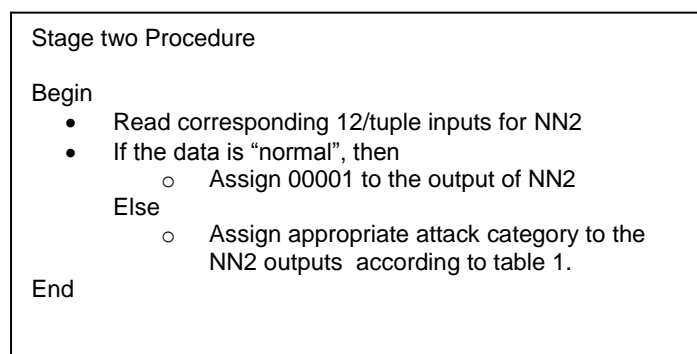
Several techniques that can be used in the process of classifying data ,such as Neural Networks, statistical methods, and others. In our work, NNs have been used in classification of data. The results can only be obtained after completing both of training and testing phases. The intrusion data have been classified into five categories. Table 5.3 describes these categories and the actual outputs from Neural Network stage two module.

Table 5.3 Neural Network Classified Categories

No	Category	NN outputs
1	Dos/DDos Attack	10000
2	Port Scan Attack	01000
3	Land Attack	00100
4	Other/unknown Attack	00010
5	Normal	00001

Table 5.4 below shows a general description of detection and classification procedure in stage two neural networks.

Table 5.4. Detection and Classification Procedure



The number of input nodes to the NN2 corresponds to the number of the selected features from NetFlow dataset for the second stage NN2 (12 Features). The implemented neural network includes one input layer, one hidden layer and an output layer of 5 nodes (Table 5.3 contains the descriptions of the outputs). The numbers of nodes in the hidden layers has been determined based on the back propagation (BP)

Chapter 5: Experimental Results

computation process and the process of trial and error. Table 5.4 describes the detection and classification procedure. By applying number of test experiments to evaluate our approach (NN2), the results are shown in table 5.5.

Table 5.5 Results of detection and classification module

Training Algorithm Parameters	Resilient Backpropagation Test 1	Levenberg-Marquardt Test2	Radial Basis Function Net Test 3
Training dataset	101806		
Validation Data	21816		
Testing set	21816		
Hidden Layer	50	50	20
Number of detected attacks(14527)	13468	13684	13234
Number of detected traffic as normal(7289)	6757	6866	6640
Detection Rate	92.7%	94.2%	91.1%
False positive Rate	3.6%	3.4%	5.1%

As shown in table the total input data is 145438 records, 96852 records as attacker and 48558 records as normal, 21816 records were used for testing the neural network stage two and it contains 14527 records as attack and 7289 records as normal. Figure 5.3 shows the best performance of neural network stage two, while figure 5.4 shows the detection rate of neural network stage two.

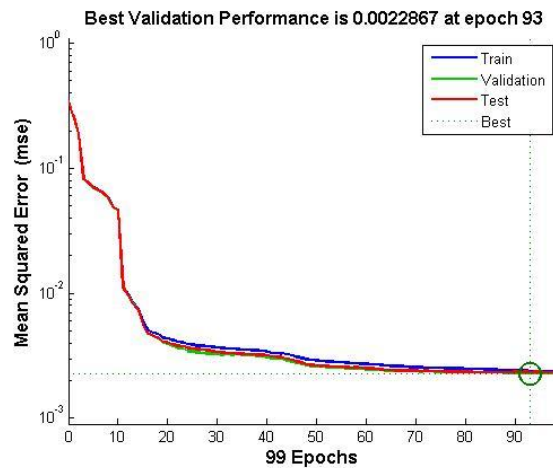


Figure 5.3. Performance of the detection and classification module

Chapter 5: Experimental Results

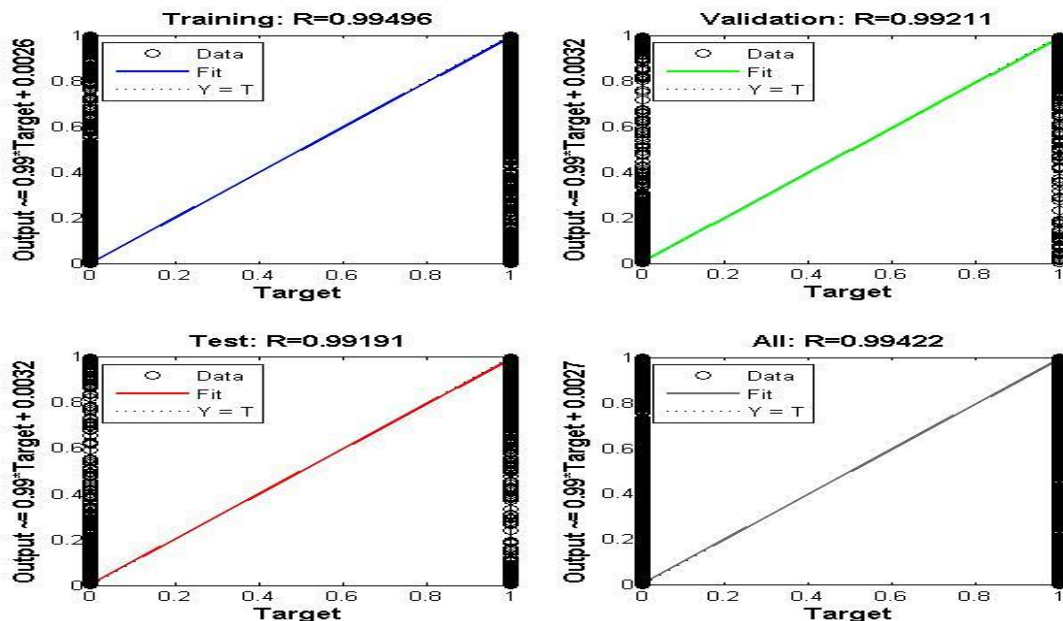


Figure 5.4 Detection rate for stage two neural networks

The classification rate of each attack types was calculated according to the following formula:

$$\text{Classification rate} = \frac{\text{number of classified attack}}{\text{total number of attack type}} \times 100[\%]$$

The best result of the classification module during the test phase is shown in table 5.6.

Table 5.6. The results of classification stage

Attack name	Total number of attacks	Number of classified attacks	Classification rate
DoS	4490	4490	100%
Port scan	9929	9919	99.9%
Land	85	85	100%
Unknown	23	18	78%

From the table 5.6, the accuracy classification is calculated for each category of attack and it's almost 100% for all categories except unknown attacks which is about 78%.

5.4 Discussion of Results

Two stages of Anomaly detection systems using neural networks and based on NetFlow dataset have been proposed and tested. Three different training algorithms (Resilient

Chapter 5: Experimental Results

Backpropagation, Radial Basis Function net, and Levenberg-Marquardt) were used for training of both neural network stages. Anomaly detection stage (NN1) was trained until the best validation performance 0.0405 was met at epoch 113 as shown in Figure 5.2. The results in Table 5.2 show that the detection rate is 94.2% with false positive of 5.8%. Results from detection and classification stage (NN2), show significantly larger improvement of prediction accuracy than the Anomaly detection phase. Figure 5.3 shows that, the best validation performance 0.0022 was met at epoch 93. Table 5.5 shows that the detection rate is relatively high at 99.42% for MLP, and 95.4% for RBF detection algorithm. The false alarms were as low as 0.58% in MLP neural network and 4.6% in RBF neural network. Table 5.6 shows that, 100% of DoS attack, 99.9% of port scan attack, 100% of land attack, and 78% of unknown attack were detected and classified correctly by using stage two neural networks. The analysis of both stages results shows that, MLP with Levenberg-Marquardt is found to be fast compared to Resilient Backpropagation, has low memory consumption compared to Radial Basis Function, and has a lower false alarm rates.

5.5 Comparison of Results

In this section, we compare our results with the other researcher's results available in the literature. Vallipuram and Robert [134] used backpropagation neural network based on KDD'99 dataset, the detection rate was 86% with high false alarm rate at 14%. Mukkamalaa [135] used backpropagation neural networks with the use of DARPA dataset, and the detection rate was 97.04% and false alarm rate of 2.06%. Dima, Roman, and Leon [136] used both MLP and RBF neural network with KDD'99 as a dataset, their results was 93.2% for RBF, and 92.2% for MLP with 7.2% as false alarm. Muna Mohammad [138] used MLP AND Fuzzy-clustering algorithm with the use of DARPA dataset, the detection rate was 99.9% and low false alarm rate 0.1%. Rodrigo Braga [139] used unsupervised neural network with flow dataset and the results for detection rate was 99.11% with false alarm rate of 0.99%. Govindarajan and Chandrasekaran [140] used neural based hybrid classification methods and they used flow dataset, their results were 96.67% for abnormal traffic, and 96.54% for normal traffic. Prasanta, Bhattacharyya, Borah and Jugal [141] used both supervised and unsupervised neural

Chapter 5: Experimental Results

network with the use of flow dataset and KDD'99 dataset, the detection rate were 99.1% for flow dataset and 92.26% for KDD'99 dataset with false rate of 0.9%.

In our research with two neural network stages based on extracted NetFlow dataset, we have achieved the detection rate at 99.4% for MLP, and 94.6% for RBF neural network with low false alarm rate at 0.6%. The results show that our proposed system is greatly competitive and performs significantly better Detection Rate (DR). From Table 5.7, we observe and conclude that our system with two neural network stages based on flow dataset and the use of a small number of extracted features can effectively and efficiently detect and classify both known and unknown attacks. The obtained false alarm rate is low compared to other methods that use different techniques and different datasets.

Chapter 5: Experimental Results

Table 5.7. Comparison of Intrusion Detection Systems Using NN.

Research	NN type	Dataset used	Detection Rate (%)	False Alarm Rate
Vallipuram and Robert,2004	Backpropagation	KDD-99	86% for normal traffic	14%
Mukkamalaa S., 2005	Backpropagation	DARPA	97.04%	2.06%
Dima, Roman and Leon,2006	MLP and RBF	KDD-99	93.2% using RBF and 92.2% using MLP	8.8%
[137] Sammany M,2007	2 hidden layers MLP	DARPA	96.65%	3.35%
Muna Mhammad T. Jawhar,2009	MLP and Fuzzy C-Mean (FCM) clustering algorithms	DARPA	99.9%	0.1%
Rodrigo Braga,2010	SOM	Open flow dataset	99.11%	0.99%
LAHEEB MOHAMMAD IBRAHIM,2010	DISTRIBUTED TIME-DELAY NEURAL NETWORK	KDD-99	97.24%	2.76%
Govindarajan , Chandrasekaran,2011	hybrid classification methods	Flow data set	96.67% for abnormal traffic, and 96.54% for normal traffic	3.33%
Prasanta Gogoi, Bhattacharyya,Borah and Jugal Kalita,2013	Supervised and unsupervised neural network	Packet Level and Flow Level dataset, KDD-99	99.1% for packet/flow level data, and 92.26% for KDD	0.9%
Our proposed IDSs	Two stage neural network	NetFlow dataset	94.2% for stage one NN, and 99.4% for stage two NN	0.6%

Chapter 6: Conclusions and Future Work

6.1 Conclusion

We have proposed and developed a flow based intrusion detection and classification method using two neural networks stages for separate tasks. One neural network detects traffic anomalies that can be attacks and the other one classifies attacks if they exist. This system can easily be extended, configured, and/or modified by replacing some features or adding new features for new types of attacks.

The training of the NNs modules requires a very large amount of NetFlow data with known types of attacks and considerable time to ensure that the results from the NNs are accurate. The changes in patterns of usage of the network should not be undetected, but at the same time, these changes are isolated to NN1. Appearance of new patterns of attack affects only classification in NN2, which is the main reason to have two stage neural networks instead of one. Consequently, the events that require retraining for the two networks are completely independent. Experiments with different NNs were crucial to define the NN which yields the best classification and training speed results for both NN stages.

The experimental results of the proposed method prove that the use of NetFlow dataset and extracting only features that significantly contribute to intrusion detection gives promising results. The obtained detection rate (94.2% for anomaly detection at stage one, and 99.4% for classification at stage two) is remarkably good compared to other approaches, which use larger training sets [142]. These results are comparable to the best researches that are based on a similar approach using the different type of training dataset Figure 6.1 illustrates the previous researches results compared to our approach.

The multilayer Feedforward neural network has a better classification ability compared to RBFN, but memory and time consumption is 3-5 times greater. Otherwise, RBFN has a simple architecture and hybrid learning algorithm which leads to less time/memory consumption and it is better for working in real-time and for retraining with new data.

Chapter 6: Conclusions and Future Work

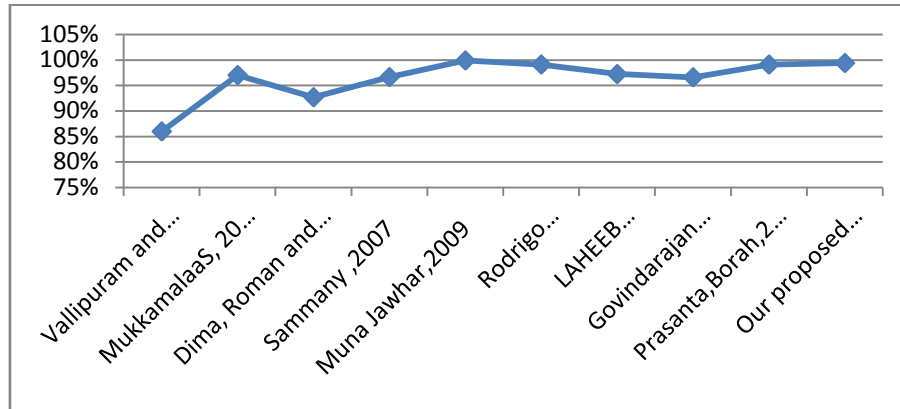


Figure. 6.1. Detection Rate on Different Datasets for IDSs.

6.2 Future Work

While our work has produced some promising results, it is necessary to improve our system further to detect more known and unknown attacks. Our proposed system also needs further testing on a wider data set with more variety of attacks.

Future work will concentrate on minimizing the number of selected features and to find out features that only have influence on detection attacks. Our future work will also be directed towards developing a more accurate model that can be used in real-time for detecting and classifying anomaly with minimum false alarms and less time.

References

- [1] B. A. Forouzan, “TCP/IP Protocol Suite”, 1st Edition, McGraw-Hill Companies, 2000.
- [2] E. G. Britten, J. Tavs, R. Bournas, “TCP/IP: The Next Generation”, *IBM Systems Journal*, Vol. 34, No. 3, pp. 452-472, 1995.
- [3] C. J. P. Moschovitis, H. Poole, T. Schuyler and T. M. Senft, “History of the Internet: A Chronology, 1843 to the Present”, *Santa Barbara, CA*, 1999.
- [4] B. Carlson, A. Burgess and C. Miller, “Timeline of Computing History”, <http://www.computer.org/computer/timeline/timeline.pdf>, 2003.
- [5] CERT Coordination Center, <http://www.cert.org/certcc.html>, Jul. 2008.
- [6] W. Stallings, “Cryptography and Network Security: Principles and Practices”, 3rd Edition, *Prentice Hall*, 2003.
- [7] Richard Power, “2002 CSI/FBI Computer Crime and Security Survey”, Vol. VIII, No.1, spring 2002.
- [8] E. Strassberg, R. J. Gondek and G. Rollie, “Firewalls: The Complete Reference”, *McGraw-Hill/Osborne*, 2002.
- [9] Google, <http://www.google.com>, accessed 16.June, 2003.
- [10] T. Titsworth, “Silver Bullets and Cybersecurity”, *IEEE Multimedia*, Vol. 10, No. 1, pp. 64- 65, 2003.
- [11] Internet2 NetFlow Weekly Reports, <http://netflow.internet2.edu/weekly>, Jul. 2008.
- [12] H. Lai, S. Cai, H. Huang, J. Xie, and H. Li, “A parallel intrusion detection system for high-speed networks”, in *Proc. of the Second International Conference Applied Cryptography and Network Security (ACNS'04)*, pp. 439–451, May 2004.
- [13] M. Gao, K. Zhang, and J. Lu, “Efficient packet matching for gigabit network intrusion detection using TCAMs”, in *Proc. of 20th International Conference on Advanced Information Networking and Applications* , pp. 249–254, 2006.
- [14] M. Roesch, “Snort, intrusion detection system”, Available at: <http://www.snort.org>, Jul. 2008.
- [15] V. Paxson, “Bro: a system for detecting network intruders in real-time”, *Computer Networks*, vol. 31, no. 23–24, pp. 2435–2463, 1999.
- [16] H. Dreger, A. Feldmann, V. Paxson, and R. Sommer, “Operational experiences with high- volume network intrusion detection,” in *Proc. SIGSAC: 11th ACM Conference on Computer and Communications Security (CSS'04)*, pp. 2–11,2004.
- [17] Z. Fadlullah, T. Taleb, N. Ansari, K. Hashimoto, Y. Y. Miyake, Y. Nemoto, and N.Kato, “Combating against attacks on encrypted protocols”, in *IEEE International Conference on Communications (ICC'07)*, pp. 1211–1216, 2007.
- [18] T. Taleb, Z. M. Fadlullah, K. Hashimoto, Y. Nemoto, and N. Kato, “Tracing Back attacks against encrypted protocols”, in *Proc. of the 2007 international Conference on Wireless communications and mobile computing*, pp. 121–126, 2007.
- [19] S. Song and Z. Chen, “Adaptive network flow clustering,” in: *proc. IEEE International Conference on Networking, Sensing and Control*, pp.596–601, 2007.
- [20] G. Schaffrath and B. Stiller, “Conceptual integration of flow-based and packet-based network intrusion detection,” in *Proc. of 2nd International Conference on Autonomous Infrastructure, Management and Security* , pp. 190–194, 2008.
- [21] Cisco.com, “Cisco IOS Flexible NetFlow White Paper”, <http://www.cisco.com>.
- [22] Cisco.com, “Cisco IOS NetFlow Configuration Guide”, *Release 12.4 available at* <http://www.cisco.com>, Sept. 2010.

References

- [23] B. Claise, “Cisco Systems NetFlow Services Export Version 9”, *RFC 3954 (Informational)*, July 2008.
- [24] B. Claise, “Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information”, *RFC 5101*, 2008.
- [25] C. Herringshaw, “Detecting Attacks on Networks”, *IEEE Computer*, Vol 30, No 12, pp.16- 17, 1997.
- [26] D. Joo, T. Hong and I. Han, “The Neural Network Models for IDS based on the Asymmetric costs of false negative errors and false positive errors”, *Expert Systems with Applications* Vol. 25, pp. 69-75, 2003.
- [27] G. Giacinto, F. Roli and L. Didaci, “Fusion of Multiple Classifiers for Intrusion Detection in Computer Networks”, *Pattern Recognition Letters*, Vol. 24, pp. 1795-1803, 2003.
- [28] Lin M, Miikkulainen R, Ryan J., “Intrusion detection with neural networks”, *Advances in Neural Information Processing Systems*, pp.943–957, 1998.
- [29] T. Fioreze, “Self-management of hybrid optical and packet switching networks”, *PhD thesis, University of Twente*, Feb. 2010.
- [30] T. Fioreze, M. O. Wolbers, R. van de Meent, and A. Pras., “Finding elephant flows for optical networks”. In *Proc. of 10th IFIP/IEEE Int. Symposium on Integrated Network Management (IM '07)*, pages 627–640, 2007.
- [31] R. Durst, T. Champion, B. Witten, E. Miller and L. Spagnuolo, “Testing and Evaluating Computer Intrusion Detection Systems”, *Communications of the ACM*, Vol. 42, No. 7, pp. 53-61, July 1999.
- [32] GEANT, available at: <http://www.geant.net>, Sept. 2010.
- [33] G. He and J. C. Hou., “An in-depth, analytical study of sampling techniques for self-similar internet traffic”, In *Proc. of the 25th IEEE Int. Conf. on Distributed Computing Systems (ICDCS '05)*, pp. 404–413, 2005.
- [34] InMon Corporation, “sflowtrend”, available at: <http://www.inmon.com>, 2010.
- [35] IsarNet Software Solutions, “Isarflow”, available at: <http://isarflow.com/>, 2010.
- [36] C. A. Carver, J. M. D. Hill and U. W. Pooch, “Limiting Uncertainty in Intrusion response”, *2001 IEEE Man Systems and Cybernetics Information Assurance Workshop*, pp. 142-147, New York, June 2001.
- [37] R. A. Maxion and K. M. C. Tan, “Anomaly Detection in Embedded Systems”, *IEEE Trans. On Computers*, Vol. 51, No. 2, pp. 108-120, February 2002.
- [38] J. McHugh, A. Christie, and J. Allen, “Defending Yourself: The Role of Intrusion Detection Systems”, *IEEE Software*, Vol. 17, No. 5, pp. 42-51, 2000.
- [39] C. Morariu, P. Racz, and B. Stiller, “Design and Implementation of a Distributed Platform for Sharing IP Flow Records, In *Proc. of the 20th IFIP/IEEE Int. Workshop on Distributed Systems: Operations and Management* , 2009.
- [40] C. Morariu and B. Stiller, “DiCAP: Distributed Packet Capturing architecture for high-speed network links”, In *Proc. of the 33rd IEEE Conf. on Local Computer Networks (LCN '08)*, Oct. 2008.
- [41] S. B. Cho, “Incorporating Soft Computing Techniques Into a Probabilistic Intrusion Detection System”, *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Revise*, Vol. 32, No. 2, pp. 154-160, 2002.
- [42] J. Quittek, S. Bryant, B. Claise, P. Aitken, and J. Meyer, “Information Model for IP Flow Information Export”, *RFC 5102 (Proposed Standard)*, Jan. 2008.
- [43] J. Quittek, T. Zseby, B. Claise, and S. Zander, “Requirements for IP Flow Information Export (IPFIX)”, *RFC 3917 (Informational)*.

References

- [44] B. Trammell and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", *RFC 5103 (Proposed Standard)*, 2008.
- [45] SURFnet. www.surfnet.nl, Sept. 2010.
- [46] M. Fullmer, "Flow-tools", <http://www.splintered.net/sw/flow-tools/>, Sept. 2010.
- [47] P. Haag, "Nfdump", available at: <http://nfdump.sourceforge.net/>, Sept. 2010.
- [48] T. Draelos, D. Duggan, M. Collins and D. Wunsch, "Adaptive Critic Designs for Host- Based Intrusion Detection", *Proc. of the 2002 International Joint Conference on Neural Networks*, Vol. 2, pp. 1720- 1725, May 2002.
- [49] R. A. Kemmerer and G. Vigna, "Intrusion Detection: A Brief Introduction and History", *Security & Privacy Supplement IEEE Computer Magazine*, pp. 27-30, 2002.
- [50] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance", *Technical Report*, James P. Anderson Co., Fort Washington, Pa, 1980.
- [51] D. E. Denning, "An Intrusion-Detection Model", *IEEE Trans. Software Eng.*, Vol. 13, No. 2, pp. 222-232, February 1989.
- [52] S. C. Lee and D. V. Heinbuch, "Training a Neural-Network Based Intrusion Detector to Recognize Novel Attacks", *IEEE Transactions On Systems, Man, And Cybernetics - Part A: Systems And Humans*, Vol. 31, No. 4, pp. 294-299, 2001.
- [53] R. A. Maxion and K. M. C. Tan, "Benchmarking Anomaly-Based Detection Systems", *IEEE Computer Society Press – International Conference on Dependable Systems and Networks*, pp. 623-630, 25- 28 June 2000.
- [54] G. Goth, "Securing the Internet Against Attack", *IEEE Internet Computing*, Vol.7, No. 1, pp. 8-10, 2003.
- [55] Y. Lapid, N. Ahituv and A. Neumann, "Approaches to Handling Trojan Horse Threats", *Computers and Security*, Vol. 5, No. 3, pp. 251-256, September 1986.
- [56] F. Cohen, "Current Best Practice Against Computer Viruses", *25th IEEE International Carnahan Conference on Security Technology*, Oct.1-3, 1991.
- [57] S. Mohiuddin, S. Hershkop, R. Bhan and S. Stolfo, "Defending Against a Large Scale Denial-of-Service Attack", in *Proc. 2002 IEEE Workshop on Information Assurance and Security*, New York, pp. 17-19, June 2002.
- [58] S. Northcutt and J. Novak, "Network Intrusion Detection", *an Analyst's Handbook, 2nd Edition*, New Riders Publishing, 2001.
- [59] R. Comerford, "No Longer in Denial", *IEEE Spectrum*, Vol. 38, pp. 59-61, 2001.
- [60] L. Garber, "Denial-of-Service attacks Rip the Internet", *IEEE Computer*, Vol. 33, No. 4, pp. 12-17, April 2000.
- [61] DARPA Intrusion Detection Evaluation, *Lincoln Laboratory, Massachusetts Institute of Technology*, <http://www.ll.mit.edu/IST/ideval/index.html>, 2003.
- [62] S. Mukkamala, G. Janoski and A. Sung, "Intrusion Detection Using Neural Networks and Support Vector Machines", *Proc. of the 2002 International Joint Conference on Neural Networks*, Vol. 2, pp. 1702- 1707, May 2002.
- [63] A. Householder, K. Houle and C. Dougherty, "Computer Attack Trends Challenge Internet Security", *Security & Privacy – Supplement – IEEE Computer Magazine*, pp. 5-7, April 2002.
- [64] D. B. Chapman, S. Cooper and E. D. Zwicky, "Building Internet Firewalls", *2nd Edition*, O'Reilly & Associates, 2000.
- [65] J. McHugh, A. Christie, and J. Allen, "Defending Yourself: The Role of Intrusion Detection Systems", *IEEE Software*, Vol. 17, No. 5, pp. 42-51, 2000.

References

- [66] T. Verwoerd and R. Hunt, "Intrusion Detection Techniques and Approaches", *Computer Communications*, Vol. 25, No. 15, pp. 1356-1365, 2002.
- [67] J. Cannady, "Next Generation Intrusion Detection: Autonomous Reinforcement Learning Of Network Attacks", *Proc. 23rd National Information Systems Security Conference*, pp. 1- 12, Baltimore, 2000.
- [68] Rosenblatt, Frank. x.," Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms", *Spartan Books*, Washington DC, 1961.
- [69] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation", *David E. Rumelhart, James L.McClelland, and the PDP research group. (Editors), Parallel distributed processing:Explorations in the microstructure of cognition*, Vol.1, MIT, 1986.
- [70] Cybenko, G. "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals, and Systems*, 2(4), 303–314.1989
- [71] Hagan, M.T., H.B. Demuth, and M.H. Beale, "Neural Network Design", *Boston, MA: PWS Publishing*, 1996.
- [72] Hagan, M.T., and M. Menhaj, "Training feed-forward networks with the Marquardt algorithm", *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, pp. 989–9931999,.
- [73] Moody, J., and Darken C., "Fast learning in networks of locally-tuned processing units", *Neural Computer*, MIT Press Cambridge, MA, USA, Vol. 1, pp.281-294. 1989.
- [74] R. Ghosh, "A novel hybrid learning algorithm for artificial neural networks", *Ph.D.Thesis*, School of Information Technology, Griffith University, 2002.
- [75] Debar H, Becker M, Siboni D.,"A neural network component for an intrusion detection system", *In: The proceedings of the 1992 IEEE symposium on research in computer security and privacy*, Oakland, CA, p. 240-250, May 1992.
- [76] Ghosh AK, Schwartzbard A.,"A study in using neural networks for anomaly and misuse detection", *In: The proceeding on the 8th USENIX security symposium*, 1999.
- [77] Horeis T.,"Intrusion detection with neural networks-combination of self-organizing maps and radial basis function networks for human expert integration", *Computational Intelligence Society, Research report*, 2003.
- [78] Lin M, Miikkulainen R, Ryan J.,"Intrusion detection with neural networks", *Advances in Neural Information Processing Systems*, pp.943–957, 1998.
- [79] Charron F, Ghosh A, Wanken J.,"Detecting anomalous and unknown intrusions against programs", *In: 14th Annual Computer Security Applications Conference*, P.259–67,1998
- [80] Heywood M, Lichodzijewski P, Zincir-Heywood N.,"Dynamic intrusion detection using self-organizing maps", *In: The annual Canadian information technology security*, 2002.
- [81] Binh Viet N., "Self organizing map (SOM) for anomaly detection", 2002.
- [82] Lin M, Miikkulainen R, Ryan J.,"Intrusion detection with neural networks", *Advances in Neural Information Processing Systems*, pp.943–957, 1998.
- [83] Cannady J.,"Artificial neural networks for misuse detection", *In: The proceedings of the1998 national information systems security conference (NISSC'98)*, 1998.
- [84] S. Mukkamala, G. Janoski and A. Sung, "Intrusion Detection Using Neural Networks and Support Vector Machines", *Proc. of the 2002 International Joint Conference on Neural Networks*, Vol. 2, pp. 1702- 1707, May 2002.

References

- [85] Cunningham R, Lippmann R., "Improving intrusion detection performance using keyword selection and neural networks", *In: The proceedings of the international symposium on recent advances in intrusion detection*, Purdue, IN, 1999.
- [86] Mehdi Moradi, and Mohammad Zulkernine, "A Neural Network Based System for Intrusion Detection and Classification of Attacks", *International Conference on Advances in Intelligent Systems, Theory and Applications*, Luxembourg, Kirchberg, IEEE, November 2004.
- [87] Rachid Beghdad, "Critical Study of Neural Networks in Detection Intrusions", *Press, Computer and Security, Elsevier*, June 2008.
- [88] Yuehui Chen, Ajith Abraham, and Bo Yang, "Hybrid Flexible Neural-Tree-Bas Intrusion Detection Systems", *International Journal of Intelligent Systems*, Vol.22, 2007.
- [89] Lichodzijewski P, Zincir Heywood AN, Heywood MI. Host-based intrusion detection using self-organizing maps. In: The proceedings of the 2002 IEEE world congress on computational intelligence, Honolulu, HI; 2002. p. 1714–9.
- [90] S. Zanero and S. M. Savaresi, "Unsupervised learning techniques for an intrusion Detection system", *in Proc. of the ACM symposium on Applied computing*, pp.412– 419, 2004.
- [91] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "On the capability of an SOM based intrusion detection system", *in Proceedings of CNN*, pp.1808–1813, 2003.
- [92] J. Z. Lei and A. Ghorbani, "Network intrusion detection using an improved competitive Learning neural network", *in Proceedings of CNSR*, pp. 190–197, 2004.
- [93] P. Haag., "Nfsen: Netflow sensor", available at: nfsen.sourceforge.net, April 2008.
- [94] D. Plonka, "Flowscan", www.caida.org/tools/utilities/flowscan/, April 2008.
- [95] B. Claise, "Cisco Systems NetFlow Services Export Version 9", *RFC 3954 (Informational)*, July 2008.
- [96] Cisco.com, "Cisco IOS Flexible NetFlow White Paper", <http://www.cisco.com>, 2010.
- [97] IP Flow Information Export Working Group, available at: www.ietf.org/html.charters/ipfix-charter.html, April 2008.
- [98] E. Izkue and E. Magaña., "Sampling time-dependent parameters in high-speed network monitoring", *In : Proc. of the ACM international workshop on Performance monitoring measurement, and evaluation of heterogeneous wireless and wired networks*, ACM, pp. 13–17, 2006.
- [99] L. Yang and G. Michailidis, "Sampled based estimation of network traffic flow characteristics", *In INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 1775–1783, 2007.
- [100] sFlow. www.sflow.org, April 2008.
- [101] A. Lakhina, M. Crovella, and C. Diot., "Characterization of network-wide anomalies in traffic flows", *In IMC '04: Proc. of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 201–206, 2004.
- [102] A. Lakhina, M. Crovella, and C. Diot., "Diagnosing network-wide traffic anomalies", *In: Proc. of the Conference on Applications, technologies, architectures, and protocols for computer comm.*, pp. 219–230, 2004.
- [103] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft. "Structural analysis of network traffic flows", *SIGMETRICS Perform. Eval.*

References

- Rev., 32(1), pp.61– 72, 2004.
- [104] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. “Network anomography”. In *Proceedings of the Internet Measurement Conference 2005 on Internet Measurement Conference*, pages 317–330, 2005.
 - [105] T. Dubendorfer and B. Plattner, “Host behavior based early detection of worm outbreaks in internet backbones”. In: *Proc. of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, WETICE '05*, pages 166–171, 2005.
 - [106] Y. Gao, Z. Li, and Y. Chen, “A dos resilient flow-level intrusion detection approach for high- speed networks”, *ICDCS 2006: 26th IEEE International Conference on Distributed Computing Systems*, pp. 39, 2006
 - [107] G. Munz and G. Carle, “Real-time Analysis of Flow Data for Network Attack Detection”, In *Proc. of 10th IFIP/IEEE Int. Symposium on Integrated Network Management (IM'07)*, pp.100–108, 2007.
 - [108] Mell, P., Hu, V., Lippmann, R., Haines, J., Zissman, M., “An overview of issues in testing intrusion detection systems”, *Technical Report NIST IR 7007*, National Institute of Standards and Technology, June 2003.
 - [109] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S.E.Webster, D.Wyschogrod, R. Cunningham, and M. Zissman, “Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation”, In *Proc. of the DARPA Information Survivability Conf. and Exposition*, pp. 12–26, 2000.
 - [110] Haines, J., Lippmann, R., Fried, D., Zissman, M., Tran, E., Boswell, and S., “1999 DARPA Intrusion Detection Evaluation: Design and Procedures”, *Technical Report TR 1062*, MIT Lincoln Laboratory, February 2001.
 - [111] R. Schweller, L. Zhichun, Y. Chen, Y. Gao, A. Gupta, Y. Zhang, P. Dinda, M.-Y.Kao, and G. Memik, “Reverse hashing for high-speed network monitoring: Algorithms, evaluation, and applications”, In *Proc. of the 25th IEEE Int. Conf. on Computer Communications (INFOCOMM '06)*, pp. 1–12, 2006.
 - [112] Q. Zhao, J. Xu, and A. Kumar. Detection of super sources and destinations in high-speed networks: Algorithms, analysis and evaluation. *IEEE Journal on Selected Areas in Communications*, 24(10):1840–1852, 2006.
 - [113] Z. Li, Y. Gao, and Y. Chen., “Towards a high-speed router-based anomaly intrusion detection system”, <http://conferences.sigcomm.org/sigcomm/2005/poster-121>, 2005.
 - [114] M.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung and J. Hong, “A flow-based method for abnormal network traffic detection”, In *Proc. of IEEE/IFIP Network Operations and management Symposium (NOMS'04)*, pp. 599–612, 2004.
 - [115] A. Lakhina, M. Crovella, and C. Diot., “Characterization of network-wide anomalies in traffic flows”, In *IMC '04: Proc. of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 201–206, 2004.
 - [116] A. Lakhina, M. Crovella, and C. Diot., “Mining anomalies using traffic feature distributions”, *ACM SIGCOMM Computer Communication Review*, vol. 35 (4), pp. 217–228, 2005.
 - [117] J. Kinable, “Detection of network scan attacks using flow data”, In *Proc. of the 8th Twente Student Conference on IT*, 2008.
 - [118] A.Wagner and B. Plattner, “Entropy based worm and anomaly detection in fast IP networks”, In *Proceedings of the 14th IEEE International Workshops on*

References

- Enabling Technologies Infrastructure for Collaborative Enterprise*, 2005.
- [119] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.
 - [120] A. Wagner, T. Dubendorfer, B. Plattner, and R. Hiestand, "Experiences with Worm propagation simulations", In *Proc. of 2003 ACM workshop on Rapid Malcode (WORM '03)*, pp.34–41, 2003.
 - [121] M. Lee, T. Shon, K. Cho, M. Chung, J. Seo, and J. Moon, "An Approach for Classifying Internet Worms Based on Temporal Behaviors and Packet Flows", In *Proc. Of 3rd Int. Conf. on Intelligent Computing*, pp. 646–655, 2007.
 - [122] C. Zou, W. Gong, and D. Towsley. Code red worm propagation modeling and analysis. In *Proc. of 17th USENIX Security Symposium (USENIX Security '08)*, pages 138–147, 2002.
 - [123] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm", *IEEE Security & Privacy*, Vol.1, pp.33–39, 2003.
 - [124] T. Dubendorfer, A. Wagner, and B. Plattner, "A framework for real-time worm attack detection and backbone monitoring", In *Proc. of 1st IEEE Int. Workshop on Critical Infrastructure Protection (IWCIP)*, pp. 3–12, Nov. 2005.
 - [125] F. Dressler, W. Jaegers, and R. German, "Flow-based Worm Detection using Correlated Honeypot Logs", In *Proc. of 15th GI/ITG Fachtagung Kommunikation in Verteilten Systemen (KiVS '07)*, pages 181–186, 2007.
 - [126] Mell, P., Hu, V., Lippmann, R., Haines, J., Zissman, M., "An overview of issues in testing intrusion detection systems", *Technical Report NIST IR 7007*, National Institute of Standards and Technology, June 2003.
 - [127] S. M. Specht and R. B. Lee, "Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures", In *Proc. of the ISCA 17th Int. Conf. on Parallel and Distributed Computing Systems (ISCA PDCS)*, pp. 543–550, 2004.
 - [128] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, "Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation", In *Proc. of the DARPA Information Survivability Conf. and Exposition*, pp. 12–26, 2000.
 - [129] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das., "The 1999 DARPA off-line intrusion detection evaluation", *Computer Networks*, vol. 34(4), pp.597–595, 2000.
 - [130] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project", In *Proc. of the 2000 DARPA Information Survivability Conference and Exposition*, pp.130–144, 2000.
 - [131] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *Proc. of the 2nd IEEE Int. Conf. on Computational Intelligence for security and defense applications*, pages 53–58, 2009.
 - [132] Sperotto, A., Sadre, R., van Vliet, D.F. and Pras, A. A Labeled Data Set For Flow-based Intrusion Detection. In: *IP Operations and Management*, Proceedings of the 9th IEEE International Workshop IPOM, October 29-30, 2009.
 - [133] Softflowd, available at: <https://code.google.com/p/softflowd/>.
 - [134] M. Vallipuram and B. Robert, "An Intelligent Intrusion Detection System based on Neural Network", *IADIS International Conference Applied Computing*, 2004

References

- [135] Mukkamala, S.; and Sung, A.H.,” Feature selection for intrusion detection using neural networks and support vector machines”, *Transportation Research Record*, pp. 33-39. 2003.
- [136] D. Novikov, V. Roman Yampolskiy, and L. Reznik, "Anomaly Detection Based Intrusion Detection", *IEEE Third International Conference on Communication, Networking & Broadcasting*, pp 420-425, 2006.
- [137] Sammany, M.; Sharawi, M.; El-Beltagy, M.; and Saroit, I.,” Artificial neural networks architecture for intrusion detection systems and classification of attacks”, *Accepted for publication in the 5th international conference INFO2007*, Cairo University.2007
- [138] Muna Mhammad T. Jawhar,” Design Network Intrusion Detection System Using hybrid Fuzzy- Neural Network”, *International Journal of Computer Science and Security*, Vol. (4).2009.
- [139] Rodrigo Braga,” Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow”, *35th Annual IEEE Conference on Local Computer Networks LCN*, Denver, Colorado 6, 2010
- [140] M.Govindarajan, and RM.Chandrasekaran,” Intrusion detection using neural based hybrid classification methods”, *computer networks journal*. Vol. 55, issue 8, pp.1662-1671, 2011
- [141] Prasanta Gogoi, D.K. Bhattacharyya, B. Borah, and Jugal K. Kalita “MLH-IDS: A Multi-Level Hybrid Intrusion Detection Method”, *the Computer Journal Advance Access published May 12, 2013*
- [142] DARPA1998, available at: <http://www.ll.mit.edu/IST/ideval/docs/1998>.

Biography

Yousef Abuadlla was born on April, 10, 1969 in Zahra, Libya. In 1989, he began his Bachelor's degree at Faculty of Electrical Engineering, University of Al-Fatah-Tripoli, Libya. He graduated in 1993 with B.Sc. degree in Computer Engineering. He started to work in 1993 for Research and Development Center (R&D) in Computer Department. He continued his studies at the Faculty of Electrical Engineering University of Belgrade in 2001, where he enrolled in the master studies. The final work of M.Sc. (thesis) defended at the Faculty of Electrical Engineering in Belgrade. He graduated in 2003 with degree in Engineering Science in the field of onboard computers. Upon his return to Libya continued to work in Research and Development Center in Tripoli.

Between 2004- 2006, he worked as assistant lecturer as a part time for Computer network and Operating system courses at the Technical Institute in Zahra.

He started PhD degree at the Faculty of Electrical Engineering University of Belgrade in the fall 2008. He participated in one international conference. Upon completion of his doctoral studies, he will continue his work for Research and Development Center.

Прилог 1.

Изјава о ауторству

Потписани-а Yousef Abuadlla

број уписа 930

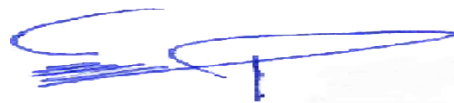
Изјављујем

да је докторска дисертација под насловом

Систем за детекцију упада заснован на токовима са две неуралне мреже

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

Потпис докторанда



У Београду, 22/09/2014

Appendices

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Yousef Abuadlla

Број уписа 930

Студијски програм Computer Engineering

Наслов рада Систем за детекцију упада заснован на токовима са две
неуралне мреже

Ментор Zoran Jovanovic

Потписани 

изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду**.

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда



У Београду, 22/09/2014

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Систем за детекцију упада заснован на токовима са две неуралне мреже

која је моје ауторско дело.

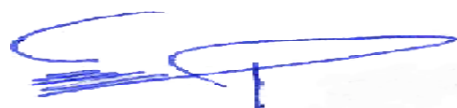
Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу

- ① Ауторство
2. Ауторство – некомерцијално
3. Ауторство – некомерцијално – без прераде
4. Ауторство – некомерцијално – делити под истим условима
5. Ауторство – без прераде
6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда



У Београду, 22/09/2014

Appendices

1. Ауторство - Дозвољавање умножавања, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство – некомерцијално. Дозвољавање умножавања, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавање умножавања, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство - некомерцијално – делити под истим условима. Дозвољавање умножавања, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство – без прераде. Дозвољавање умножавања, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство - делити под истим условима. Дозвољавање умножавања, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.