A Feature Classification Scheme for Network Intrusion Detection

Iosif-Viorel Onut and Ali A. Ghorbani
(Corresponding author: Iosif-Viorel Onut)

Faculty of Computer Science, University of New Brunswick
540 Windsor Street, Fredericton, New Brunswick, PoBox 4400, Postal Code E3B 5A3, Canada

(Received Oct. 18, 2005; revised and accepted Dec. 3, 2005 & Feb. 21, 2006)

Abstract

One of the most important phases of the IDS/IPS implementation identifies the set of features that the system is going to use. We present a feature classification schema for network intrusion detection intended to provide a better understanding regarding the features that can be extracted from network packets. Furthermore, we present the design of a feature extractor that extracts and statistically analyze features with respect to attacks. The experimental results, conducted on DARPA dataset, are intended to statistically highlight the importance of each proposed feature category, as well as to identify some of the most sensitive features to attacks.

Keywords: Feature classification, feature extractor, intrusion detection, network security

1 Introduction

Network security is one of the top priorities of our modern society. The Internet growth, information sharing, and technology improvement are some of the factors that humans become dependent on. Illegal access to private and confidential data has become a new way of crime. Tools for malicious purposes are freely available for download from the Internet. Hackers all over the world no longer need to have a strong background in order to perform attacks. The biggest challenge that network administrators face is to find good intrusion detection solutions that can work on-line and are able to detect intrusions in real-time.

Whether the focus is on Intrusion Detection Systems (IDS) or on Intrusion Prevention Systems (IPS), the challenges remain the same: feature selection, intrusion identification method, intrusion identification latency, and false positive rates to name a few.

One of the most important phases of an IDS/IPS implementation is the decision upon the set of features that the system is going to use for detection/prevention purposes. This decision will directly influence the types of attacks that are going to be detected/prevented by the IDS/IPS (e.g., a network based IDS will not be able to detect an application-based intrusion initiated within the same host, similarly an antivirus will not be able to detect a Distributed Denial of Service (DDoS) attack).

Despite its importance, to the best of our knowledge, there is no comprehensive classification of features that an IDS/IPS might use for detecting network based attacks. Moreover, different researchers use different names for the same subsets of features, while others use the same name for completely different types. Network security related articles, journals, and white-papers usually concentrate on the detection techniques that they use, briefly mentioning the chosen features or the reasons behind that selection.

We propose a feature classification scheme for network intrusion detection that is intended to provide a better understanding of the large number of attributes that can be extracted from network packets, their relationships, as well as their usefulness in detecting different types of attacks.

The rest of this paper is organized as follows, Section 2 presents the main existing feature classifications schemes that are used in the literature. Section 3 identifies the main abstractions of the network security domain. The naming convention of the proposed feature categories is explained in Section 4. The domain abstractions are used in Section 5 for defining the proposed feature classification schema. Section 6 presents two available methods for reporting the features to the detection engines. Subsequently, Section 7 describes the architectural design of our feature extractor engine. Section 8 presents our experimental results while using DARPA 1999 [4], a well known intrusion detection and evaluation dataset. Finally, Section 9 presents the conclusion and possible future extensions of the work.

2 Background Review

The diversity of features that can be extracted from raw data packets and the time complexity that takes for some

of the features to be computed are two major factors that lead to compromises that researchers make when designing and deciding upon the features that they are going to use.

Even though there is no unanimous accepted classification regarding the features that can be extracted from raw packet data, most of the papers do make a distinction (even if not directly) between features that are computed with respect to a single TCP connection, versus those that are computed considering multiple TCP connections. Thus, two types of features can be identified as follows.

1) Basic TCP Features: features that characterize a single TCP/IP connection. The name of this category differs from author to author, but the semantic tends to be consistent. Accordingly, Dokas et al. [5], and Ertoz et al. [6] use the name *Basic Features*; Lee et al. [20] use *Essential Attributes*; KDD-1999 uses *Basic Features of individual TCP connection* [7], while Lichodzijewski et al.[9] use *Basic TCP Features* as a suitable name for this category. Finally, Powers [11] uses the name *Flow Statistics* for a superset of this category, which also includes connection-less protocols (e.g., UDP, ICMP).

2) Derived Features: features that can characterize multiple TCP/IP connections at the same time [5, 6]. Also, known as *Traffic Features* [7, 20].

The goal of using *Derived Features* is to find similarities that exist between different TCP connections in the network. In order to compute those features, two types of sliding window intervals are used. The first approach uses a time-window interval of few seconds (e.g., 5 sec), while the second one uses a connection-window interval of several connections (e.g., last 100 connections). The use of the two types of sliding windows further divides the *Derived Features* category into Time based and Connection based features as follows:

1) Time Based Features: are all the derived features computed with respect to the past x seconds, where x is the size of the time window interval [5, 6, 7, 20]. This type of features are useful for detecting bursty attacks (i.e., attacks that happen within a short period of interval) such as worm and DDoS.

2) Connection Based Features: are all the derived features computed with respect to the past k TCP connections that were encountered in the network [6, 5, 20]. These features are used when the detection of stealthy attacks is targeted (i.e., attacks that happen within a long period of time, usually several minutes or even hours).

Although theoretically it is possible to design a system that can extract and analyze a wide range of features, due to constraints such as, large computational time, diversity of protocols and applications that exist, and amount of memory that the IDS/IPS needs, most of the implementations make tradeoffs concentrating only on a particular set of intrusions (e.g., R2U, U2R, and DoS in [5, 6, 20]; Horizontal, Vertical, and Block PortScanning for TCP and UDP in [16]; Denial of Quality of Service in [19]; Worms in [2, 18, 22]; DDoS in [10, 13]; TCP-SYN Flooding Attacks in [21]; TCP-SYN Flood, UDP Flood, and ICMP Flood Attacks in [14]). Finally, due to possible false correlations between features that an IDS/IPS might detect, selecting larger number of features may not necessarily lead to a better detection [12, 17], but may have adverse effect on the performance of the system. For instance, by using only 17 features out of the 41 features provided by the International Knowledge Discovery and Data Mining Competition (KDD-1999), Chebrolu et al. [12] obtained almost the same detection rate, while the performance of the system was improved by almost 50 percent.

3 The Network Security Domain

Feature classification is a domain dependent problem that cannot be accomplished without identifying the main entities that reside in a given domain and the means that they use to communicate.

Let us consider a network as any arrangement of entities that are interconnected. In the case of a wired network those interconnections are realized through cables, routers, switches, to name a few. In the case of a wireless network, the interconnections are also achieved by using towers and antennas. The entities in a network are represented by hosts. A host can refer to almost any kind of computer that resides within the network (e.g., server, mainframe, desktop PC, or terminal).

The data exchanged between different hosts inside a network is wrapped in packets. A packet is the fundamental unit of information carriage in the network (also referred to as datagram or frame in the literature). Furthermore, let us consider a connection as the act of bringing two hosts into contact. Consequently, if the information unit is represented by a packet, a connection is represented by a collection of packets that are bidirectional, exchanged between the two hosts for fulfilling a goal. As an alternative to connections, some of the authors and IDS/IPS vendors adopt the flow definition as reported by CISCO[1] [3].

In brief, we have identified four main abstractions of the domain, namely network, host, connection, and packet. Accordingly, we further extract and classify their prominent characteristics, which we call features.

The proposed feature classification schema is defined for the Transport, Network, and Network Access layers of the TCP/IP Architecture Model. This corresponds to

[1] A flow is a unidirectional sequence of packets between a given source and a given destination. It is uniquely identified as a combination of seven factors as follows: source IP, destination IP, source port, destination port, layer 3 protocol type, the type of service byte, and the Input Logical Interface (ifIndex) value.

Data Link (MAC), Network, Transport and Session layers of the OSI (Open Systems Interconnection) standard. In particular, a host is uniquely identified by its IP address, while a connection (i.e., TCP, UDP, and ICMP[2]) is uniquely identified by the combination of 6 fields as follows source IP, destination IP, source port, destination port, protocol, and type of service. However, the schema can be easily extended to the other remaining protocols and layers of the TCP/IP architecture.

4 The Naming Convention

Due to the large number of proposed feature-categories, the current work uses acronyms for defining their names. As depicted in Figure 1 the naming convention can be followed as a path in the tree, where each edge represents a letter that adds to the end of the acronym. For instance, the DFSTU acronym is obtained by following the path of the right, left, left, and right vertices of the sub-children nodes of the root; and stands for the subset of **D**erived **F**eature category that are dependent on a **S**ingle connection, are computed using a **T**ime-window interval, and are **U**nidirectional.

Similarly, DFMCG acronym is obtained by following the path defined by the right, right, right, and center vertices of the sub-children nodes of the root; and stands for the subset of **D**erived **F**eatures that are computed using data extracted from **M**ultiple connections, in a **C**onnection-window Interval, involving only **O**ne host of the current connection.

5 The Feature Classification Schema

One of the main dilemmas in network security is whether or not a packet belongs to a malicious event. If a definite answer can be deducted from the available pool of features, the detection problem is solved. Thus, our classification schema is defined with respect to the currently sniffed packet. Therefore, all the features are grouped in two main classes *Basic Features* and *Derived Features*.

- Basic Features (BF): contains all the features that can be extracted from a single packet without requiring any kind of extra information.
- Derived Features (DF): contains all the features that require relationship analysis of multiple packets over a period of time.

Obviously, without having the BF category, none of the DF features can be computed. Moreover, most researchers do not even mention the BF category, since they only use it as an intermediary layer which helps them to create other features.

[2]In the case of an ICMP connection, the source and destination ports are 0.

Figure 1 depicts the proposed feature classification schema. Each path from the root node to either a leaf or an intermediate node also denotes the naming convention.

5.1 Basic Features (BF)

Any field of a packet (datagram) is a possible candidate for this feature category including Timestamp, sourceIP, destinationIP, protocol, sourcePortNo (if applicable), destinationPortNo (if applicable), flags, ICMP Type (if applicable), to name a few.

Some of the basic features do not provide any kind of information when studied alone, but can contribute to the computation of other derived features. For instance, let us consider the case of a packet timestamp. This feature is linearly increasing in time and is not directly useful in mining the intrusions. Nevertheless, it can be successfully used to compute a derived feature such as the duration of a certain connection. Packet source IP (srcIP) and packet destination IP (dstIP) are two other examples of basic features that cannot directly participate in an anomaly detection process. On the other hand, it is impossible to extract any information about the connections in the network if these two features are not considered. Moreover, as our experimental results show (see Section 8) some of the features from this category are quite sensitive to intrusions.

Mahoney and Chan [8] use only BF for the purpose of detecting network attacks. Their algorithm splits the fields larger than 4 bytes (such as Ethernet address) into half, while concatenates fields smaller than 1 byte. However, we believe that, by examining the packets and their fields in isolation, a lot of information about packet interdependencies is lost. Basu et al. [1] also identifies this category of features under the name of *Packet Header Features*.

The BF are also used by any state-machine that needs to be implemented for a particular protocol. Usually, IDS/IPSs that implement a specification based engine closely monitor the observed states of the protocols and detect abnormal conditions that may happen. Here, BF are used as triggers to change state in the correspondent state machines (e.g., for TCP protocol, the packet Sequence number and packet Acknowledgement number are used in combination with other BF features).

5.2 Derived Features (DF)

This category contains all the features that require relationship analysis of multiple packets over a period of time. Since we define a connection as a collection of packets that are exchanged between two hosts for fulfilling a goal, the subgroup of packets that is used to compute a particular DF may belong to one or more connections. Two subcategories of related features can be identified with respect to the number of connections that they belong to as follows:

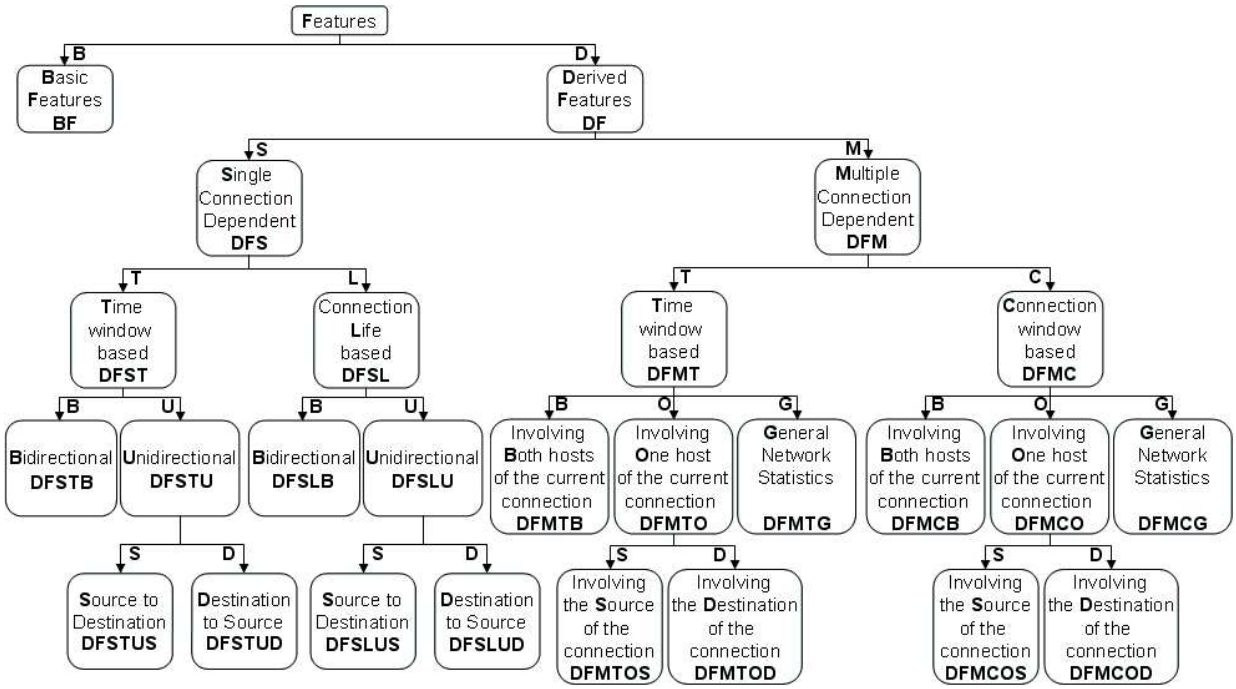

Figure 1: The feature classification schema and the naming conventions

- DF-Single connection dependent (DFS): contains features that are computed with respect to a single connection. The aim here is to detect if the current connection is a malicious one or not.

- DF-Multiple connection dependent (DFM): contains features that represent interdependencies between connections in time. These features are mostly used in the case of detecting worm attacks, DDoS attacks, or any kind of attack that requires more than one connection.

Note that our definition for connection includes connection oriented protocols (e.g. TCP) as well as connectionless protocols (e.g. UDP, ICMP). In the case of connectionless protocols we define a connection as the number of packets exchanged between two hosts in the last m seconds (using the same two ports when appropriate).

Most researchers in the area of Network Security agree with this classification, even though the names differ from author to author. For example Dokas et al. [5], Ertoz et al. [6], and Lichodzijewski et al. [9] use the name *Basic Features* for a subset of the DFS category, Lee et al. [20] use *Essential Attributes*, while Powers [11] uses *Flow Statistics* for the same subcategory. Finally, KDD-1999 distinguishes only the subset that refers to the TCP protocol as *Basic Features of individual TCP connection* [7].

The DFM category has also been referred to as *Traffic Features* [7, 20], and *Derived Features* [5, 6] by different researchers.

The DFS and DFM categories are further described in the following two subsections.

Table 1: Subset of the DFS features

No	Feature Description
1	no. of packets
2	no. of bytes
3	no. of fragmented packets
4	no. of overlapping fragments
5	no. of TCP packets that have flag X set
6	no. of TCP flags per packet
7	no. of ICMP echo request packets
8	no. of ICMP destination unreachable packets

5.2.1 DF-Single Connection Dependent Features (DFS)

This category contains all the features that can describe a single connection. Even though the DFS features can be computed for each existing connection in the network, from the intrusion detection point of view a special interest represents the connection that the current sniffed packet belongs to. Table 1 contains a small subset of the features that belong to this category.

The DFS features can be computed in two different ways depending on the type of intrusion targeted. For bursty attacks, the connection's activity for the past few seconds is a decisive factor, while stealthy attacks can be detected by studying the duration of the whole connection. These two methods lead to the following categories.

- DFS-Time-window based features (DFST): contains all the DFS features that are computed by using a time window interval.

- DFS-connection-Life based features (DFSL): includes all DFS features computed with respect to

connection's life time. Theoretically, if there is a connection that lasts for two days, a network security system should be able to compute all the features of that particular connection without any problem. In practice, due to the space and processing time restriction, if the connection is not closed after a reasonable period of time, it will be discarded.

There might be a concern that some of the DFSL features should be extracted only when a connection ends. The argument against this position is that an intrusion detection system does not have the luxury to wait until the connection is closed, but has to adopt a proactive approach. Thus, for example, even though the total duration of a connection cannot be exactly extracted before the connection ends, the current duration of connection is a valuable information.

Regardless of the method used to extract the DFS features (i.e., time-window based, connection-Life based), each of the previously discussed subcategories (i.e., DFST and DFSL) can be further analyzed with respect to the packet direction between the two hosts involved in the connection. Consequently, the next subsection describes them in parallel. For easier notation let DFSx stand for either DFST or DFSL categories.

Since there are always two hosts involved in a connection, the features can be further categorized into unidirectional features (DFSxU) and bidirectional features (DFSxB), where x can be replaced with either T or L for time-window based and connection-life based features, respectively (see Figure 1).

1) DFSx-Unidirectional features (DFSxU): are features computed with respect to either one of the two hosts that belong to the connection. For instance, the "no. of packets" feature will become "no. of packets sent by srcIP to dstIP", and "no. of packets sent by dstIP to srcIP".

   This is important in the cases when the features computed for the current connection show an abnormality and the system has to decide which one of the two hosts is the attacker and which one is the victim. Note that the initiator of the connection is not always the attacker, while the target of the connection is not always the victim. For example, consider the case of a *Trojan Horse* installed on a computer that opens a backdoor with the real attacker. This is especially relevant when the application layer is also considered for features extraction. Thus, the DFSxU features are further divided into two types as follows:

   a. DFSxU-Source to destination features (DFSxUS): includes all DFSxU features computed with respect to the active behavior of the connection source IP.

   b. DFSxU-Destination to source features (DFSxUD): includes all DFSxU features computed with respect to the active behavior of the connection destination IP.

   The previous distinction is implicitly done by all the systems that consider flows[3] instead of connections. Even though computationally a system that uses flows might be superior, due to lack of correlation between the two flows that belong to a connection, some of the attacks will remain undetected (e.g., attacks that can be identified by the use of specification based detection techniques).

2) DFSx-Bidirectional features (DFSxB): this category covers all the features that can depict the traffic of a connection as a whole, considering the contribution of both hosts at the same time. This category of features can be used to build connection anomaly profiles, to define connection specification based rules, or to compare the current connection against Quality of Service factors.

   The bidirectional features can also be used for identifying Remote to Local or DoS attacks. As an example, consider the case of an *UDP Storm* attack, where by the use of a single spoofed packet, the attacker compromises two hosts. In this particular case, the two participants to the current connection are both victims, and thus, intuitively the features representing the connection will better highlight the anomaly than the previously presented unidirectional features.

5.2.2 DF-Multiple Connection Dependent Features (DFM)

All the feature-groups identified in the previous subsection were defined with respect to a single connection. The main issue that may arise so far is that by their use, an IDS/IPS would not be able to detect intrusions that use more than one connection at a time such as *worm*, *scanning*, and *DDoS* attacks. Thus, for this purpose, the DFM group includes features that characterize interdependencies between connections in time.

Since the targeted information has to be extracted from multiple connections, two methods can be identified for feature construction: features extracted based on a predefined time interval (e.g., *number of TCP packets* exchanged through the network in the last 5 seconds), and features extracted considering a fixed number of connections (e.g., *number of TCP packets* exchanged through the network by the last 100 TCP connections) [1, 5, 6, 20]. Consequently, the DFM category is further separated into the following subcategories:

- DFM-Time-window based features (DFMT): includes all the features that are computed with respect to the last time interval. This type of features is mostly used for detection of bursty attacks that happened in the last several seconds.

- DFM-Connection-window based features (DFMC): includes all the DFM features that are computed

[3]The flow definition is described by CISCO[3]

Table 2: A subset of the DFMxB features

No	Feature Description
1	no. of *AAA* connections between *SrcIP* and *DstIP*
2	no. of *BBB* connections created by *SrcIP* using any port to connect to any port on *DstIP*
3	no. of *BBB* connections created by *SrcIP* using the *SrcPort* to connect to any port on *DstIP*
4	no. of *BBB* connections created by *SrcIP* using any port to connect to the *DstPort* on *DstIP*
5	no. of *BBB* connections created by *DstIP* using any port to connect to any port on *SrcIP*
6	no. of *BBB* connections created by *DstIP* using the *SrcPort* to connect to any port on *SrcIP*
7	no. of *BBB* connections created by *DstIP* using any port to connect to the *DstPort* on *SrcIP*
8	no. of *AAA* packets received by *SrcIP* from *DstIP*
9	no. of *AAA* packets sent by *SrcIP* to *DstIP*
10	no. of *AAA* bytes received by *SrcIP* from *DstIP*
11	no. of *AAA* bytes sent by *SrcIP* to *DstIP*
12	average no. of *AAA* bytes per packet received by *SrcIP* from *DstIP*
13	average no. of *AAA* bytes per second received by *SrcIP* from *DstIP*
14	average no. of *AAA* bytes per packet sent by *SrcIP* to *DstIP*
15	average no. of *AAA* bytes per second sent by *SrcIP* to *DstIP*
16	no. of TCP packets sent by *SrcIP* to *DstIP* with *XXX* flag
17	no. of TCP header flags sent by *SrcIP* to *DstIP*
18	no. of TCP packets received by *SrcIP* from *DstIP* with *XXX* flag
19	no. of TCP header flags received by *SrcIP* from *DstIP*
20	no. of ICMP echo request packets sent by *SrcIP* to *DstIP*
21	no. of ICMP echo request packets received by *SrcIP* from *DstIP*
22	no. of ICMP destination unreachable packets sent by *SrcIP* to *DstIP*
23	no. of ICMP destination unreachable packets received by *SrcIP* from *DstIP*

with respect to the last several encountered connections. This type of feature is used to detect stealthy attacks (i.e., attacks that happen over a long period of time).

Stealthy attacks are famous due to their ability to pass the security layer undetected. This is mostly because their behavior slightly changes in time, and an IDS that uses a predefined time window interval cannot detect anything as the time window slides. This does not happen in the case of a connection-window interval in which the entire connection (not just the last several seconds) is considered for computing the features.

Depending on the type and number of protocols that an IDS/IPS is analyzing, a connection-interval may be assigned to each protocol or a single connection-interval may be assigned for all the protocols. Consequently, in the first case, the IDS/IPS would have to handle multiple connection-intervals while in the second case only one. The first method might be preferred due to the intuitive behavioral difference between different protocols. For example, factors such as traffic burst, number of connections, and load of connections, are fundamentally different from protocol to protocol (e.g., the load created in the case of a TCP connection cannot be compared with the load created by an ICMP connection[4]). The main disadvantage of this approach is that it is computationally intensive. The second approach, on the other hand, uses only a connection-interval and makes the computational power manageable. However, in the case of simultaneous attacks, the connection-window might be flooded by only some attacks, making the other ones undetectable.

Each of the previously discussed DFM subcategories can be further analyzed versus the connections that are taken into consideration when the features are computed. Consequently, the next subsection describes the DFMT and DFMC categories in parallel. For easier notation let DFMx stand for both DFMT and DFMC categories.

The features belonging to both of these two categories can be further grouped into three distinct sets (see Figure 1): features that describe the activity between two hosts, features that describe the activity of a single host, and features that describe the activity of the whole network.

Let CSrcIP, CDstIP, CSrcPort and CDstPort represent the connection source IP, destination IP, source port, and destination port, respectively. Similarly, SrcIP, DstIP, SrcPort, and DstPort denote the current packet's source IP, destination IP, source port, and destination port, respectively. Since for the ICMP protocol no port numbers are defined, we consider CSrcPort, CDstPort, SrcPort, DstPort only for TCP and UDP protocols. Furthermore, let AAA, BBB, and XXX denote the following sets {TCP, UDP, ICMP}, {TCP, UDP}, and {URG, ACK, PHS, RST, SYN, FIN}, respectively. These notations are used in Tables 2, 3, and 4.

The previously mentioned subcategories are defined as follows:

1) DFMx-involving Both hosts of the current connection (DFMxB): all the features in this category are computed considering the contribution of both srcIP and dstIP hosts over multiple connections between them (see Table 2). This set of features is especially useful in the case of vertical scanning attacks, DoS attacks, or whenever the attacker initiates multiple connections between his host and the victim.

2) DFMx-involving One host of the current connection (DFMxO): this category of features are constructed in order to monitor the interaction between one host and the whole network. This is especially useful in the case of DDoS attacks (i.e., when the attacker controls multiple daemons which aim to compromise a certain host), a horizontal scanning attacks (i.e., when the attacker scans multiple hosts in order to

[4]A connection is previously defined as the act of bringing two hosts into contact, thus, the definition can also be applied in the case of connectionless protocols.

Table 3: A subset of the DFMxOy features, where the letter **y** stands for **S** or **D** in DFMxOS and DFMxOD categories, respectively

No	Feature Description
1	no. of *AAA* connections created by HostY
2	no. of *AAA* connections that use HostY
3	no. of *BBB* connections created by HostY using any port to connect to the *CDstPort* on any other hosts
4	no. of *BBB* connections created by any host using any port to connect to the *CDstPort* on HostY
5	no. of *BBB* connections created by HostY using the *CSrcPort* to connect to any ports on any other hosts
6	no. of *BBB* connections created by any host using the *CSrcPort* to connect to any ports on HostY
7	no. of TCP connections created by HostY which have SYN Flag
8	no. of TCP connections that use HostY which have SYN Flag
9	no. of TCP connections created by HostY which have RST Flag
10	no. of TCP connections that use HostY which have RST Flag
11	no. of *AAA* packets received by HostY
12	no. of *AAA* packets sent by HostY
13	no. of *AAA* bytes received by HostY
14	no. of *AAA* bytes sent by HostY
15	average no. of *AAA* bytes per packet received by HostY
16	average no. of *AAA* bytes per second received by HostY
17	average no. of *AAA* bytes per packet sent by HostY
18	average no. of *AAA* bytes per second sent by HostY
19	no. of ICMP packets received by HostY with destination unreachable flag
20	no. of ICMP packets sent by HostY with echo reply flag
21	no. of TCP packets received by HostY with *XXX* flag
22	no. of TCP header flags received by HostY

find possible vulnerabilities), or a worm propagation attacks (i.e., when the worm scans the network prior to its migration).

The features in this category can be computed with respect to CSrcIP and CDstIP as follows:

a. DFMx-involving the Source of the connection (DFMxOS): contains features that depict the interaction of the CSrcIP host with the whole network (i.e., all the other hosts that CSrcIP has at least one connection with). Table 3 lists a subset of the features contained in this category when HostY is replaced with CSrcIP.

b. DFMx-involving the Destination of the connection (DFMxOD): similar to the previous subcategory but computed with respect to the CDstIP, the DFMxOD features describe the interaction of CDstIP with the whole network. Table 3 lists a possible subset of the features contained in this category when HostY is replaced with CDstIP.

The distinction made between the DFMxOS and DFMxOD categories is very important since in the first case (i.e., DFMxOS) the source of the connection is targeted for possible malicious activities while in the second case (i.e., DFMxOD) the destination of the connection is targeted for possible infection. Thus, some of the attacks can be identified by using this two categories in conjunction. For instance, in the case of a *worm* attack the connections that have as destination the CSrcIP can be compared against those connections that are initiated afterwards by CDstIP. If these two features exhibit a similar pattern shifted in time, then one conclusion might be that the hosts are infected.

3) DFMx-General network statistics (DFMxG): these features are used to provide statistical information about the state of the whole network. Any feature belonging to this category (see Table 4) is not directly related to any of the two hosts that participate in the current connection (i.e., CSrcIP and CDstIP). Thus, DFMxG provides valuable information about intrusions that are not related to the current packet, but represent a threat to the network itself (e.g., scanning, probing, and DDoS). This information can also be used as a trace to network quality of service related issues such as inbound/outbound traffic ratios, number of dropped packets, number of retransmitted packets, number of connections closed with a reset signal, number of packets that have small *time to live* value, to name a few.

A subset of the DFMTG category is also identified by KDD-99 under the name of *Traffic Features* [7], while Lee et al. defines other two main categories that are also contained into the DFMx category as follows: *The Same Host* features and *The Same Port* features [20]. The former category defined by Lee et al. is contained in DFMxO features, while the latter one is divided between DFMxO and DFMxG categories. We believe that *The Same Port* category cannot be defined as a single category since a *port* is neither an entity nor an activity of the previously defined knowledge domain. Therefore, the part of *The Same Port* category which is computed versus the whole network is included in the DFMxG category (note that it is not directly related with any of the two hosts participating to the current connection), while the part which is computed versus either of the CSrcIP or CDstIP is included in DFMxO category.

Table 4: A subset of the DFMxG features

No	Feature Description
1	no. of *AAA* connections
2	no. of *AAA* packets
3	no. of *BBB* connections that use the *CDstPort* as destination port
4	no. of *BBB* connections that use the *CSrcPort* as source port
5	no. of TCP packets with *XXX* flag
6	no. of received ICMP packets with destination unreachable flag
7	no. of sent ICMP packets with echo reply flag
8	no. of ICMP connections that have at least one destination unreachable message
9	no. of ICMP connections that have at least one echo reply message
10	no. of TCP connections that use *CSrcPort* as source port and have SYN Flag
11	no. of TCP connections that use *CDstPort* as destination port and have SYN Flag
12	no. of TCP connections that use *CSrcPort* as source port and have RST Flag
13	no. of TCP connections that use *CDstPort* as destination port and have RST Flag
14	no. of TCP connections that have SYN Flag
15	no. of TCP connections that have RST Flag
16	the percent of partially opened TCP connections vs. all connections
17	the percent of TCP connections that were closed with reset vs. all connections
18	the percent of *AAA* connections vs. all connections

6 Reporting the Features to a Detection Engine

Once the features are constructed, they need to be forwarded to the detection engine(s) for further processing. There are two main methods for reporting the features. For instance, if the detection engine implements a *Signature based* technique on packets, it is most likely that it will analyze each packet as it is encountered from the network. Furthermore, a detection engine that uses a *Specification based* technique based on protocol analysis, will use a state machine that also needs to be fed with data each time a packet is encountered. On the other hand, the *Signature* and *Specification based* detection techniques together with the *Anomaly based* technique may also be implemented accept data that is collected at a rate of several seconds. Thus, in this case, the data is collected and reported to the detection engine in a synchronized manner. There are two methods of reporting the data as follows:

- Event-based reporting: the data is reported to the detection engine whenever a packet is encountered. The principal candidates for such an approach are all BF features, but also part of the DFSL and DFMC features. The most common consumers of this reporting technique are packet signature and protocol state machine analyzers.

- Time-based reporting: the data is reported to the detection engine based on a predefined rate of several seconds. In this case, any feature that belong to DF category is a possible candidate for this method. The mostly common subgroups that are sent this way are DFMxG features, and part of the DFMxB and DFMxO features. This method is mostly suitable for traffic and quality of service analysis as well as anomaly detection.

7 Implementation of the Feature Extractor

To further study the features that have been presented in the previous sections, a feature extractor engine along with a postprocessing module have been implemented. This section presents the underlying high-level architecture of the system, along with some of the algorithms that are used for feature construction. The main objective is to design the feature extractor as an external entity that can afterwards be plugged in a real IDS/IPS system. Moreover, the design had to encompass features from all the previously mentioned categories, and to easily allow the addition of other network features later on. Finally, since the system is primarily concerned with feature behavior mining, it must also be highly customizable, allowing the user to change several feature construction variables such as granularity and dimensionality of the *time-window interval*, dimensionality of the *connection-window interval*, maximum time to live interval that triggers the deletion of inactive connections, the option to analyze connectionless protocols as flows or connections, to name a few.

7.1 Top Level View

Figure 2 depicts the main building blocks of the system. The first processing unit is the *Data Reader Module*, which allows the system to either directly sniff the network packets from the Network Interface Card (NIC), or to read them from a binary TCP dump file. At this stage, all the features from the BF category are extracted and passed along to the *Time Reconstruction Module* (TRM). The TRM is bypassed when the data is directly sniffed from the network, but plays a critical role when the data is read from a TCP dump file in replaying the packets at their normal speed. Thus, the implementation of the next module becomes independent of the input data. The *Connection Reconstruction and Features Extraction Module* represents the core of the system. This module is responsible for extracting all the other feature categories that exist in our classification schema. The module uses the *event-based reporting* technique for forwarding the fea-

tures to the next modules. The rational behind choosing this type of reporting against *time based reporting* is that for feature post-analysis the packet by packet reporting is more appropriate. This module is currently design to extract 613 features per packet[5]. As final step of the feature extraction process, the data is stored in a MySQL database, each day of the data being stored in a separate table. After the features are saved into the MySQL database, and if the labels are provided for the data, the *Data Labeler Module*, working as an offline process, applies those labels. Due to processing speed and memory consumption, the developing platform/language for all the previously presented modules are Linux and C++. Finally, the *Statistical Extractor Module* has been implemented for feature post analysis. The aim of this module is to provide statistical properties of the features while under normal as well as intrusion traffic, so that their relevancy over different attack scenarios can be extracted and analyzed. Since in this case the speed and memory consumption are not critical factors, the program was written as a combination of Java and Matlab languages under windows platform.

The following two subsections describe in more detail some of the key points in the design of both: The *Connection Reconstruction and Features Extraction Module*, as well as the *Statistics Extractor Module*.

7.2 The Feature Extraction Process

Several key issues must be addressed at the designing stage of the feature extraction, such as data dimensionality, data redundancy, memory usage, and speed. This is especially important in the case of network traffic due to the tremendous volume of data that must be analyzed, combined with the number of features (i.e., 613) that the feature extractor was designed to produce. While data dimensionality is accommodated by using connection and time-window intervals, the data redundancy can be solved through a careful analysis of both the network security domain abstractions (as defined in Section 3) and the proposed classification schema. Consequently, five main data structures can be identified that will allow the extraction of all the feature categories from the classification schema as follows: packet data group, host data group, connection data group, HostX-HostY data group, and network data group.

Each one of these structures stores and maintains a particular set of features. For instance, the packet data group is used to store the BF category. A container is created for each encountered packet, and populated as the BF features are extracted.

Next, the host data group is used to store any feature that is related to a particular host. This includes DFMTO and DFMCO categories. One instance of this container is needed for each existing host in the network. Due to memory constraints, once a particular host is not involved anymore in any connection, its container is automatically deleted. Since the host IP address is unique for each active host, this key is used as a unique identification ID.

The connection data group is used to store those features that target a single connection (i.e., DFSTB, DFSTU, DFSLB, DFSLU). Like in the previous cases, a container is created for each new encountered connection. Furthermore, each time when a connection is closed (or becomes inactive for a long period of time), its container is disregarded. Each connection is identified by a unique ID composed of 5 features as follows: <IP1, IP2, Port1, Port2, Protocol>, where host IP1 uses Port1, host IP2 uses Port2, the connection is using Protocol, and P1<[6]IP2. In this way, regardless of the packet direction, the connection ID will remain the same.

Next, HostX-HostY data group is used to store any features that characterize the exchanged information between HostX and HostY (i.e., DFMTB and DFMCB features). Therefore, a such container is created for each pair of hosts that exchange at least one packet. Once is no more activity between a pair of hosts, the corespondent container is deleted. As additional information, each container also carries a list of all the connections that the two hosts create between them, and is uniquely identified by the <P1,P2> tuple, where P1 and P2 are the IP addresses for the two hosts, and P1<P2.

Finally, since the network is uniquely identified by the sniffer's position, a single data structure called network data group is sufficient to contain both the DFMTG and DFMCG features.

7.2.1 Implementing the Connection-Window Interval

The connection-window interval is used for computing the connection-window based features (i.e., DFTCOS, DFTCOD, DFTCB, DFTCG), and it is internally stored as a doubly-linked list.

Using a linked list confers several advantages that directly influence the speed of the program, since the connection-window interval can be pictured as a FIFO[7]. Consequently, each time when a new connection is encountered the oldest one must be disregarded. Thus, by using a doubly-linked list the program does not have to shift all the elements inside the list in order to attach the new connection. Note that the size of the connection-window interval does not influence the performance of the algorithm since updating the head and tail required $\mathbf{O}(1)$.

Storing the connection-window interval as a simple FIFO is not enough. This is mostly because active connections may and up thrown outside the window interval, instead of closed connections. Consider a scenario where a active connection x opened several minutes ago is at the end of the window. This connection will be the first one to be disregarded when a new connection is encoun-

[5]Section 5 mentions most of the features. Please note that Tables 1,2,3, and 4 incorporate features from multiple categories.

[6]The '<' operator can be overloaded, since the internal representation of each IP address is a double long number.

[7]First In First Out vector also known as a queue

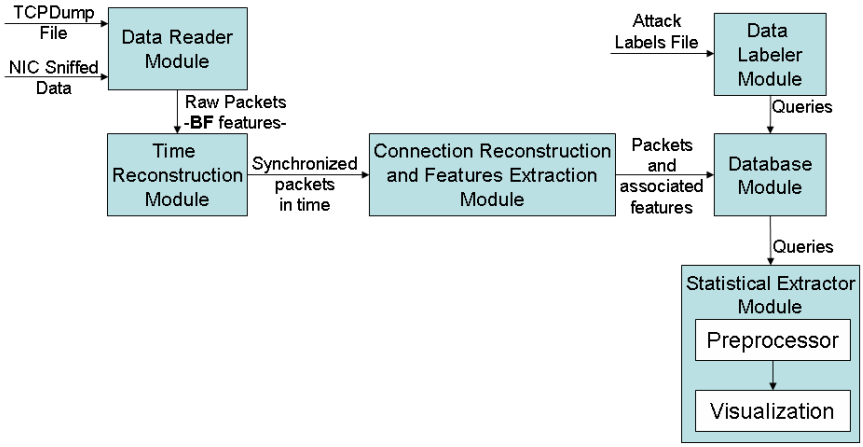

Figure 2: The underlying architecture of the feature extractor and feature post-processing system

tered. This does not exclude the possibility of several already closed connection being inside the interval. Consequently, in such situation disregarding a closed connection will be more beneficial. To accommodate this issue, each time when a packet of an old connection is encountered the connection is moved at the beginning of the connection interval. In this way, the connections which recently exchanged packets will always be near the head of the doubly-linked list, while the other connections will end up near the tail of the window.

This modification of the connection-window interval does not influence too much the performance of the algorithm since the correspondent *connection data group* is already identified, and it stores a pointer to the correspondent cell in the FIFO. Thus, moving the cell in front of the connection-window will take at most four pointer assignments (i.e. two to remove the *connectiondata group* from the current position in the FIFO, and two to insert it back into the FIFO), making the time complexity of the task $\mathbf{O}(1)$.

7.2.2 Implementing the Time-Window Interval

The primary challenges that the construction of a time-window Interval faces are the tradeoff between window-step granularity and computational time, as well as the tradeoff between window dimension and memory consumption.

Let us define window-step granularity, χ, as the time latency before the window slides again. The smaller this value is chosen, the smoother the time-window moves, and thus, the closer its value becomes to the real value. On the other hand, this decrease of χ is in the detriment of the computational time. Thus, a tradeoff must be made between the computational time required to compute the features and the smoothness of the time-window interval movement.

Regardless of granularity size, the window dimension can be expressed as $\tau = n \cdot \chi$, where $n \in \mathbb{N}^*$ represents the number of steps that the time-window encompasses. Thus, by choosing $n = 1$ the size of the window becomes equal with the granularity χ. This practice is widely accepted especially in the case when the size of the time-window is not big (e.g., a one second time-window interval updated each second). In this particular case, each time when the time-window moves, its value will be replaced with a new one. This practice becomes inefficient as the time-window increases. For instance, if $\tau = 1$ minute, it is quite ineffective to also have $\chi = 1$ minute. Here, one would rather prefer to increase the step-granularity to (let us say) $\chi = 20$ seconds (i.e., a time-window of one minute updated every 20 seconds). In this case, the new value of the time-window will be dependent on the previous one (since there will be a 40 seconds overlap between the two).

Let $\chi(a_i)$ represent the real value of feature a in i^{th} step, and $\tau(a_j)$ denote the value of a computed with respect to τ interval at time j. The current $\tau(a_t)$ value can be computed as follows:

$$\tau(a_t) = \sum_{i=t-n}^{t} \chi(a_i), \tag{1}$$

where t represents the current time. Note that this computation has a linear time complexity (i.e., $\mathbf{O}(n)$). A better solution would be to incrementally compute $\tau(a_t)$ as a function of $\tau(a_{t-1})$. Thus, Equation (1) can be further extended to:

$$\begin{aligned} \tau(a_t) &= \chi(a_t) + \sum_{i=t-n}^{t-1} \chi(a_i) \\ &= \left(\chi(a_t) + \sum_{i=t-n}^{t-1} \chi(a_i)\right) + \chi(a_{t-n-1}) - \chi(a_{t-n-1}) \\ &= \chi(a_t) + \left(\sum_{i=(t-1)-n}^{t-1} \chi(a_i)\right) - \chi(a_{t-n-1}). \end{aligned} \tag{2}$$

Thus the incremental formula is:

$$\tau(a_t) = \chi(a_t) + \tau(a_{t-1}) - \chi(a_{t-n-1}). \tag{3}$$

In Equation (3), $\tau(a_t)$ is no longer dependent on the size of the window (i.e., n), and can be computed in $\mathbf{O}(1)$, as long as the program keeps track of each individual $\chi(a_i)$ that belongs to the current interval (i.e., $i \in [t-n, t]$).

The time-window interval is implemented as a circular linked list of size n. Each cell of the list keeps a particular $\chi(a_i)$. Thus, each time when the time-window slides, the oldest χ value is subtracted from τ while the newest $\chi(a_t)$ one is added, and the position of the current pointer in the circular linked list is updated.

7.3 The Statistical Extraction Process

Figure 3 depicts the *Statistical Extractor Module* which comprises two processing stages (i.e., *preprocessing stage* and the *visualization stage*). These two stages have been implemented as a mixture of Java and Matlab languages. While the second stage is more concerned with data presentation and GUI, the first stage mines and extracts statistics from the *Database Module*. The presence of labels is critical at this stage, since without them no deviation between normal and abnormal can be defined.

The module, analysis one feature at the time reading all its stored values for a particular day. In order to study the behavior of a feature versus different types of attacks, the intrusive and normal portion of the data must be filtered out and statistical profiles must be created for each one of the behaviors. The filtering process is done by the Intrusion Filter, while the latter process is done by the Profiler.

Given that features tend to have different values for different protocols (e.g., the size of the ICMP packets is expected to be less than the size of the TCP packets), creating a common profile is not recommended. A better practice would be to filter the data by protocol, and after that to analyze it. Consequently, once the Feature Normal Behavior is extracted, the Protocol Filter is used to split the data into three streams, each stream corresponding to one of the three protocols that we analyze (i.e., TCP, UDP and ICMP).

Our proposed system uses the event-based reporting technique in order to save the data into the database. As previously explained, this technique produces one value for each encountered packet. Consequently, the set of all values for a particular feature will not resemble any information regarding the inter-arrival time between packets, but only regarding the arrival sequence of packets on the network. Furthermore, if at this stage any statistics (e.g., mean, variance) is computed the bursts in the data will bias the result towards their values[8]. Therefore, before extracting any statistics from a feature, the data has to be divided into non-overlapping groups with respect to the packet timestamp and a window size. Our system is design to accept any granularity-size for the groups. Let Δt represent this granularity in seconds. Consequently, a day of data will be divided into k non-overlapping sets of Δt seconds.

Our assumption is that a feature is influenced by a certain intrusion if its value significantly changes during the attack period. This will obviously lead to a change in the feature's mean. Hence, based on Δt, the *Profiler Module* computes the mean $\Delta t_\mu(i)$ of the feature, where i represents the i^{th} group of data, $i \in [1, k]$. This process is done separately for each one of the normal data streams (i.e., TCP, UDP, and ICMP) as well as for each individual intrusion that is to be analyzed.

Next, the *Comparator Module* reports any deviation between the expectation of the normal behavior (i.e., mean of all normal $\Delta t_\mu(i)$) and the value of each intrusion using Chebyshev's inequality (Equation (4)).

$$P\left(|x - E(x)| \geq k\sigma\right) \leq \frac{1}{k^2}, \quad \forall k > 0, \tag{4}$$

where $P(x)$, $E(x)$ and σ denote the probability, expectation, and standard deviation of x respectively, and k is any real number greater than 0. This inequality defines the probability of a random variable x to be part of a certain distribution, and holds for any kind of distribution, as long as its $E(x)$ and variance are finite [15].

8 Experimental Results

The experimental results presented in the current section aim to highlight the importance of each already presented feature categories from the proposed classification schema. The presented system extracts and analyzes a total of 613 different features. For the purpose of this work we have chosen to report the results using a connection-window interval of 100 connections, a time-window interval of $\tau = 1$min., with a granularity of $\chi = 10$sec., and a *time to live* for the inactive connections of 1 minute. Furthermore, the *Statistical Extractor Module* uses a granularity of $\Delta t = 20$min. in order to split the data into non-overlapping sets, as explained in the previous section. Next, we have used the 4^{th} and 5^{th} week of the DARPA dataset as input for our feature extractor engine, in particular, days 1 to 5 from week 4 as well as days 1 and 3 from the 5^{th} week.

Chebyshev's theorem states that for any population or sample, at least $1 - (\frac{1}{k^2})$ of the observations are within k standard deviations of the mean, where $k > 1$. For example, 96% of a population lies within five standard deviations from the mean, whereas 99% lies within 10 standard deviations. Therefore, in order to identify the features that are highly sensitive to intrusions, we take into considerations only those features that have their values within five standard deviations from the mean during normal operation, and deviate a minimum of 10 standard deviations while in attack.

Table 5 displays the top 4 features for each one of the leafs categories in the proposed classification schema. Each row of the table constitutes a feature. Since usually one feature is sensitive to multiple attacks, the table

[8] Consider the case where 20% out of all the packets encountered within last 6 hours were collected in 10 minutes.

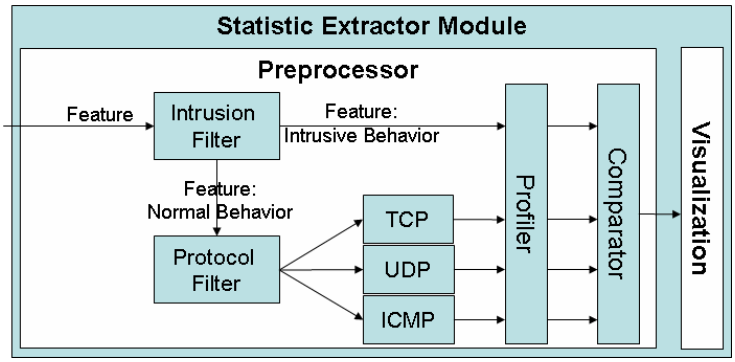

Figure 3: The statistical extractor module

also states the total number of intrusions as well as the minimum and maximum probability for the feature to belong to the normal population while in attack (see Equation (4)). Furthermore, the table also reports the k factor from Chebyshev's inequality which states the number of standard deviations that the value of the points deviates from the mean.

Furthermore, Table 6 depicts all the previously proposed feature-categories along with the total number of intrusions that they detected, as well as the total number of hits that the features belonging to the category had. As an example, for the BF category, the number of intrusions that the features in this category are sensitive to is 24, while the total number of times that the features from this category identify intrusions is 53. Additionally, the table also provides the percentage of the five main types of attacks as defined by DARPA, namely denial of service (DoS), probing, remote to local (R2L), user to remote (U2R), and Data [4].

As expected, the BF and DFS categories identify only intrusions that are not distributed. Moreover, the DFST category does not detect as many intrusions as the DFSL category does. The reason behind this difference is the size of the time-window interval (i.e., 1 min), which is relatively small comparing with the duration of the intrusions that the DFSL category detects, making the time-interval unable to grasp the change. The DFSxUS detects more attacks than the DFSxUD category since it mostly models the attacker behavior. Next, the DFM category identifies more intrusions versus DFS category since the features that belong to this category mine similarities between multiple connections in time. Please note that the number of detected intrusions is also boosted by the variety and number of features belonging to this class. A quite surprisingly observation is that the DFMxB category is probably one of the most sensitive in this dataset. This results from the fact that it picks vertical scanning attacks or dos attacks like apache2, which involve multiple connections between the two hosts. Finally, the DFMxOy and DFMcG categories highlight intrusions that use multiple connections for scanning, or Distributed DoS attacks. Note that even though the DFMcG category defines the general behavior of the network, it proves to be less efficient than the other DFM subcategories.

9 Conclusions and Future Work

In this paper we presented a feature classification schema for network intrusion detection. The aim is to provide a better understanding regarding the huge amount of features that can be extracted from network packets. We have identified 26 categories of features that can be extracted in real time by sniffing the packets from the network. The proposed categories are defined against the main abstractions of the network domain such as network, host, connection, and packet.

For each category in our classification schema, we have enumerated a subset of possible features that might be used for intrusion detection purposes. Part of the features are collected from different papers, while part of them are proposed by us. Next, we have discussed two methods of reporting any feature to the detection engine (i.e., by time and by event).

In order to study the network features, we designed and implemented a framework for real-time feature construction. Chapter 7 describes the architectural design of this framework intended to study each subset of features from the proposed classification schema. Finally, last chapter is intended to highlight the importance of each feature category in the detection process. Furthermore, using the DARPA dataset, we have statistically identified the most sensitive features to attacks and reported them.

Our future work will target to further statistically analyze and identify the most important features that need to be considered in each subgroup for detecting different types of attacks. In order to accomplish this goal we propose to study the statistical behavior of each feature against several standard intrusion evaluation databases (e.g., DARPA 99, DARPA 2000), as well as against real data. For real traffic labeling we are planning to use the alerts generated by an IDS. We will target the main types of attacks such as DoS, DDoS, scaning, and worms to name a few.

Table 5: Top 4 features for each leaf category, ranked by the number of intrusions that they are influenced by

Category	#Intr	max (P(X))	min (P(X))	max (σ(X))	min (σ(X))	Description
BF	10	0.00346	0.00033	55.1857	17.0029	packet TCP acknowledgement flag (true/false)
	7	0.00867	0.00006	124.107	10.7414	packet TCP reset flag (true/false)
	5	0.00457	0.00134	27.3075	14.7914	packet TCP synchronize flag (true/false)
	5	0.00788	0.00488	14.3206	11.2681	packet TCP finalize flag (true/false)
DFSTB	3	0.00609	0.00556	13.4067	12.8176	number of TCP packets that have finalize flag set
	3	0.00375	0.00000	792.84	16.3234	number of TCP packets that have urgent flag set
	2	0.00866	0.00044	47.4352	10.744	number of TCP packets that have reset flag set
	2	0.00935	0.00102	31.3437	10.342	number of packets exchanged
DFSTUS	6	0.00754	0.00064	39.4253	11.5174	number of TCP packets sent by host1 to host2 with finalize flag
	2	0.00669	0.00060	40.7499	12.2274	number of packets sent by host1 to host2
	2	0.00669	0.00060	40.789	12.2267	number of TCP packets sent by host1 to host2 with acknowledgement flag
	2	0.00630	0.00322	17.6259	12.6019	number of TCP packets sent by host1 to host2 with push flag
DFSTUD	2	0.00000	0.00000	1.60E+07	1.00E+07	number of ICMP destination unreachable packets received by host1 from host2
	1	0.00450	0.00450	14.9142	14.9142	number of packets sent by host1 to host2
	1	0.00375	0.00375	16.322	16.322	number of TCP packets sent by host1 to host2 with urgent flag
	1	0.00450	0.00450	14.9142	14.9142	number of TCP packets sent by host1 to host2 with acknowledgement flag
DFSLB	4	0.00806	0.00737	11.6508	11.1389	number of packets/second exchanged
	4	0.00636	0.00288	18.6398	12.5373	connection end time
	3	0.00391	0.00352	16.8667	16.0013	number of TCP packets that have finalize flag set
	2	0.00636	0.00025	63.1981	12.5361	number of bytes exchanged
DFSLUS	6	0.00754	0.00042	49.0656	11.5168	number of TCP packets sent by host1 to host2 with finalize flag
	1	0.00000	0.00000	2.80E+07	2.80E+07	number of TCP packets sent by host1 to host2 with urgent flag
	1	0.00969	0.00969	10.1594	10.1594	number of TCP packets sent by host1 to host2 with push flag
	1	0.00990	0.00990	10.052	10.052	number of TCP flags sent by host1 to host2
DFSLUD	2	0.00000	0.00000	2.00E+07	2.00E+07	number of ICMP destination unreachable packets received by host1 from host2
	1	0.00771	0.00771	11.3882	11.3882	number of bytes sent by host1 to host2
DFMTB	19	0.00998	0.00015	81.2067	10.0081	average number of TCP bytes per packet sent by *SrcIP* to *DstIP*
	12	0.00651	0.00000	14310.73	12.3923	number of TCP connections created by *SrcIP* using any port to connect to any port but *DstPort* on *DstIP*
	12	0.00793	0.00000	961.3655	11.2303	number of TCP connections created by *DstIP* using any port to connect to any port but *DstPort* on *SrcIP*
	12	0.00796	0.00000	958.266	11.2106	number of TCP connections between *SrcIP* and *DstIP*
DFMTOS	12	0.00843	0.00008	113.92	10.8886	average number of ICMP bytes per packet sent by HostY
	12	0.00190	0.00000	1826.66	22.9457	number of TCP packets received by HostY with reset flag
	10	0.00838	0.00036	52.9447	10.9224	average number of TCP bytes per packet sent by HostY
	8	0.00512	0.00037	52.1261	13.9687	number of ICMP connections created by HostY
DFMTOD	10	0.00595	0.00001	318.550	12.9679	number of TCP connections created by HostY
	10	0.00595	0.00001	318.836	12.9688	number of TCP connections created by HostY using any port to connect to any port but *CSrcPort* on any other hosts
	10	0.00978	0.00002	229.040	10.111	number of TCP connections that use HostY
	9	0.00978	0.00002	229.236	10.1111	number of TCP connections created by any host using any port to connect to any port but *CSrcPort* on HostY
DFMTG	5	0.00537	0.00042	48.6867	13.6436	number of TCP packets with reset flag
	4	0.00641	0.00008	111.7244	12.4941	number of TCP packets with urgent flag
	4	0.00944	0.00012	91.2141	10.2929	number of TCP connections that use *CDstPort* as source port and have RST Flag
	3	0.00880	0.00293	18.4779	10.658	number of TCP connections that use different ports than *CDstPort*
DFMCB	18	0.00859	0.00016	79.8677	10.7896	average number of TCP bytes per packet sent by *SrcIP* to *DstIP*
	13	0.00778	0.00000	554.94	11.3351	number of TCP connections created by *SrcIP* using any port to connect to any port but *DstPort* on *DstIP*
	12	0.00866	0.00000	554.94	10.7448	number of TCP connections between *SrcIP* and *DstIP*
	10	0.00727	0.00000	554.94	11.7284	number of TCP connections created by *SrcIP* using any port to connect to any port but *SrcPort* on *DstIP*
DFMCOS	12	0.00279	0.00000	2258.42	18.9319	number of TCP connections that use HostY which have RST Flag
	11	0.00861	0.00032	56.1709	10.7747	average number of TCP bytes per packet sent by HostY
	8	0.00917	0.00007	118.804	10.4439	number of TCP connections created by any host using any port to connect to any port but *CDstPort* on HostY
	8	0.00048	0.00004	161.142	45.7665	(OUT) number of TCP connections that do not start with SYN
DFMCOD	12	0.00730	0.00005	138.876	11.7063	average number of ICMP bytes per packet received by HostY
	10	0.00918	0.00005	146.568	10.4357	number of TCP connections created by any host using any port to connect to any port but *CDstPort* on HostY
	8	0.00818	0.00005	146.183	11.0555	number of TCP connections that use HostY
	8	0.00818	0.00005	146.509	11.0555	number of TCP connections created by any host using any port to connect to any port but *CSrcPort* on HostY
DFMCG	3	0.00762	0.00028	59.9254	11.4521	number of TCP connections that use *CDstPort* as destination port and have SYN Flag
	3	0.00327	0.00098	31.8772	17.4927	number of TCP connections that use *CDstPort* as source port and have RST Flag
	2	0.00430	0.00000	4252.54	15.2464	number of UDP connections that use *CSrcPort* as destination port
	2	0.00430	0.00000	4252.54	15.2464	number of UDP connections that use *CSrcPort* as source port

Table 6: All the identified categories versus the types of distinct attacks that they detect, and number of hits that their features have

	Number of Intrusions Detected						Total Number of Hits					
Category	**total**	**DoS%**	**Probe%**	**R2L%**	**U2R%**	**Data%**	**total**	**DoS%**	**Probe%**	**R2L%**	**U2R%**	**Data%**
F	*75*	*32.00%*	*22.67%*	*34.67%*	*8.00%*	*2.67%*	*1062*	*69.11%*	*11.21%*	*16.76%*	*2.73%*	*0.19%*
BF	**24**	37.50%	37.50%	20.83%	0.00%	4.17%	**53**	39.62%	47.17%	11.32%	0.00%	1.89%
DF	*72*	*33.33%*	*20.83%*	*36.11%*	*8.33%*	*1.39%*	*1009*	*70.66%*	*9.32%*	*17.05%*	*2.87%*	*0.10%*
DFS	*22*	*31.82%*	*45.45%*	*13.64%*	*9.09%*	*0.00%*	*70*	*32.86%*	*30.00%*	*22.86%*	*14.29%*	*0.00%*
DFST	*14*	*28.57%*	*50.00%*	*14.29%*	*7.14%*	*0.00%*	*39*	*25.64%*	*20.51%*	*30.77%*	*23.08%*	*0.00%*
DFSTB	**8**	37.50%	25.00%	25.00%	12.50%	0.00%	**16**	31.25%	12.50%	31.25%	25.00%	0.00%
DFSTU	*12*	*33.33%*	*50.00%*	*8.33%*	*8.33%*	*0.00%*	*23*	*21.74%*	*26.09%*	*30.43%*	*21.74%*	*0.00%*
DFSTUS	**9**	44.44%	33.33%	11.11%	11.11%	0.00%	**16**	31.25%	18.75%	25.00%	25.00%	0.00%
DFSTUD	**5**	0.00%	60.00%	20.00%	20.00%	0.00%	**7**	0.00%	42.86%	42.86%	14.29%	0.00%
DFSL	*20*	*35.00%*	*45.00%*	*15.00%*	*5.00%*	*0.00%*	*31*	*41.94%*	*41.94%*	*12.90%*	*3.23%*	*0.00%*
DFSLB	**16**	37.50%	37.50%	18.75%	6.25%	0.00%	**19**	42.11%	31.58%	21.05%	5.26%	0.00%
DFSLU	*9*	*44.44%*	*55.56%*	*0.00%*	*0.00%*	*0.00%*	*12*	*41.67%*	*58.33%*	*0.00%*	*0.00%*	*0.00%*
DFSLUS	**7**	57.14%	42.86%	0.00%	0.00%	0.00%	**9**	55.56%	44.44%	0.00%	0.00%	0.00%
DFSLUD	**2**	0.00%	100.00%	0.00%	0.00%	0.00%	**3**	0.00%	100.00%	0.00%	0.00%	0.00%
DFM	*72*	*33.33%*	*20.83%*	*36.11%*	*8.33%*	*1.39%*	*939*	*73.48%*	*7.77%*	*16.61%*	*2.02%*	*0.11%*
DFMT	*65*	*35.38%*	*15.38%*	*38.46%*	*9.23%*	*1.54%*	*583*	*72.90%*	*6.69%*	*17.50%*	*2.74%*	*0.17%*
DFMTB	**40**	45.00%	15.00%	37.50%	2.50%	0.00%	**199**	74.87%	5.53%	17.59%	2.01%	0.00%
DFMTO	*57*	*38.60%*	*17.54%*	*35.09%*	*7.02%*	*1.75%*	*336*	*74.40%*	*7.14%*	*16.96%*	*1.19%*	*0.30%*
DFMTOS	**43**	39.53%	20.93%	37.21%	2.33%	0.00%	**170**	74.71%	10.00%	14.71%	0.59%	0.00%
DFMTOD	**43**	46.51%	6.98%	37.21%	6.98%	2.33%	**166**	74.10%	4.22%	19.28%	1.81%	0.60%
DFMTG	**24**	45.83%	12.50%	29.17%	12.50%	0.00%	**48**	54.17%	8.33%	20.83%	16.67%	0.00%
DFMC	*57*	*36.84%*	*22.81%*	*36.84%*	*3.51%*	*0.00%*	*356*	*74.44%*	*9.55%*	*15.17%*	*0.84%*	*0.00%*
DFMCB	**38**	44.74%	15.79%	39.47%	0.00%	0.00%	**116**	74.14%	6.90%	18.97%	0.00%	0.00%
DFMCO	*46*	*41.30%*	*26.09%*	*28.26%*	*4.35%*	*0.00%*	*211*	*73.46%*	*11.85%*	*13.27%*	*1.42%*	*0.00%*
DFMCOS	**36**	44.44%	25.00%	27.78%	2.78%	0.00%	**101**	71.29%	14.85%	12.87%	0.99%	0.00%
DFMCOD	**37**	48.65%	18.92%	27.03%	5.41%	0.00%	**110**	75.45%	9.09%	13.64%	1.82%	0.00%
DFMCG	**15**	73.33%	6.67%	20.00%	0.00%	0.00%	**29**	82.76%	3.45%	13.79%	0.00%	0.00%

Acknowledgement

The authors graciously acknowledge the funding from the Atlantic Canada Opportunity Agency (ACOA) through the Atlantic Innovation Fund (AIF) and through grant RGPN 227441 from the National Science and Engineering Research Council of Canada (NSERC) to Dr. Ghorbani.

References

[1] R. Basu, R. K. Cunningham, S. E. Webster, and R. P. Lippmann, "Detecting low-profile probes and novel denial-of-service attacks", in *Proceedings of the workshop on Information Assurance and Security, United States Military Academy (IEEE, ed.)*, pp. 5-10, June 2001.

[2] V. Berk, G. Bakos, and R. Morris, "Designing a framework for active worm detection on global networks", in *Proceedings of the IEEE International Workshop on Information Assur-ance* (Darmstadt, Germany), pp. 13-23, Mar. 2003.

[3] CISCO Co., Netflow services solutions guide website, http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/products_ implementation design guide09186a00800d6a11.html, Last accessed Oct. 10, 2005.

[4] DARPA, Darpa intrusion detection and evaluation dataset, 1999, Website, last access Feb. 2006.

[5] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, and P. Tan, "Data mining for network intrusion detection", in *Proceedings of NSF Workshop on Next Generation Data Mining* (Baltimore, MD), pp. 21-30, Nov. 2002.

[6] L. Ertoz, E. Eilertson, A. Lazarevic, P. N. Tan, P. Dokas, V. Kumar, and J. Srivastava, "Detection of novel network attacks using data mining", in *ICDM Workshop on Data Mining for Computer Security* (DMSEC) (Melbourne, FL), pp. 30-39, Nov. 2003.

[7] KDD, Kdd-cup-99 task description, *The Fifth International Conference on Knowledge Discovery and Data Mining*, http://kdd.ics.uci.edu/databases/kddcup99/task.html, last access Oct, 2005.

[8] M. V. Mahoney and P. K. Chan, "PHAD: Packet header anomaly detection for identifying hostile network traffic", *Tech. Report CS-2001-4*, Florida Tech, 2001.

[9] P. Lichodzijewski, A. N. Zincir-Heywood and M. I. Heywood, "Dynamic intrusion detection using self-organizing maps", in *Proceedings of the 14th Annual Canadian Information Tech- nology Security Symposium*, May. 2002,

[10] T. Peng, C. Leckie, and R. Kotagiri, "Proactively detecting ddos attack using source IP address monitoring", in *Proceedings of the Networking 2004* (Athens, Greece), pp. 771-782, May. 2004.

[11] A. Powers, "Behavior-based IDS: Overview and deployment methodology", White Paper Lan-cope, Inc., 2003.

[12] J. P. Thomas, S. Chebrolu, and A. Abraham, "Hybrid feature selection for modeling intrusion detection systems", LNCS 3316, pp. 1020-1025, Springer-Verlag, Oct. 2004.

[13] K. C. S. Noh, C. Lee and G. Jung, "Detecting distributed denial of service (ddos) attacks through inductive learning", in *Proceedings of the Intelligent Data Engineering and Automated Learning, 4th International Conference, IDEAL 2003, (Hong Kong,*

China), LNCS 2960, pp. 287-295, Springer-Verlag, 2003.
[14] C. Siaterlis and B. Maglaris, "Towards multisensor data fusion for dos detection", in *Proceedings of the 2004 ACM symposium on Applied computing*, ACM Press, pp. 439-446, 2004.
[15] M. R. Spiegel, *Theory and problems of probability and statistics*, Mathematical Expectation, McGRAW-Hill, pp. 83-84, 1975.
[16] S. Staniford, J. Hoagland, and J. McAlerney, "Practical automated detection of stealthy portscans", in *Journal of Computer Security* vol. 10, no. 1 & 2, pp. 105-126, 2002
[17] A. H. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks", in *Proceedings of the IEEE Symposium on Applications and the Internet*, pp. 209-216, 2003.
[18] T. Toth and C. Kruegel, "Connection-history based anomaly detection", in *Proceedings of IEEE Workshop on Information Assurance and Security*, pp. 1-6, June. 2002.
[19] V. A. Mahadik, X. Wu and D. S. Reeves, "A summary of detection of denial-of-QoS attacks on Diff-Serv networks", *Proceedings of the DARPA Information Survivability Conference and Exposition*, pp. 277-282, April 2003.
[20] S. J. Stolfo, W. Lee and K. W. Mok, "Mining in a data-flow environment: Experience in network intrusion detection", in *Proceedings of the 5 International Conference on Knowledge Discovery and Data Mining*, pp. 114-124, 1999.
[21] D. H. Wang and K. Shin, "Detecting syn flooding attacks", in *Proceedings of the IEEE Infocom*, pp. 1530-1539, June 2002.
[22] C. C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and early warning for internet worms", in *Proceedings of the 10th ACM conference on Computer and communication security*, ACM Press, pp. 190-199, Oct. 2003.

Iosif-Viorel Onut is a Ph.D. candidate at the University of New Brunswick, Faculty of Computer Science, Fredericton, Canada. He received his M.Sc. and B.Eng. in computer science from Technical University of Cluj-Napoca, Romania. The work for his B.Sc. thesis was done throughout a 6 months scholarship at the Daimler-Chrysler AG Research and Technology center in Berlin, Germany (2002). During 2004-2005 he worked in the field of Privacy Security and Thrust at National Research Council of Canada, Institute for Information Technology, Fredericton, Canada. He also worked as a researcher for the City of Fredericton, Network Infrastructure, during 2005. Since 2003 he is an active member of the Network Security Laboratory (http://nsl.cs.unb.ca) at University of New Brunswick, Fredericton, Canada. His main research focus is network security. Eng. Iosif-Viorel Onut a member of IEEE.

Ali A. Ghorbani received his PhD and Master in Computer Science from the University of New Brunswick, and the George Washington University, Washington D.C., USA, respectively. He was on the faculty of Computer Engineering, Iran University of Science and Technology, Tehran, Iran, from 1987 to 1998. Since 1999 he has been at the faculty of Computer Science, University of New Brunswick (UNB), Fredericton, Canada, where he is currently a Professor of Computer Science. He is also a member of the Privacy, Security and Trust (PST) network, University of New Brunswick. He has held a variety of positions in academia for the past 25 years. His research originated in software development, where he designed and developed a number of large-scale systems. His current research focus is Neural Networks, Web intelligence, agent systems, complex adaptive systems, and trust and security. He established the Intelligent and Adaptive Systems (IAS) and Network Security research groups in 2002 and 2004, respectively, at the faculty of Computer Science, UNB. The IAS group (http://ias.cs.unb.ca) pursues research on machine and statistical learning, data mining, intelligent agents and multiagent systems and Web intelligence. The NSL group (http://nsl.cs.unb.ca) is home to R&D in computer and network security. He authored more than 100 research papers in journals and conference proceedings and has edited 6 volumes. He is on the editorial board of the Computational Intelligence (CI), an international journal. He is also associate editor of International Journal of Information Technology and Web Engineering. Dr. Ghorbani a member of ACM, IEEE, IEEE Computer Society, and ANNS.