Network Traffic Classification for Intrusion Detection

Tarek Abbes, Michaël Rusinowitch, Alakesh Haloi

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Network Traffic Classification for Intrusion Detection

Tarek Abbes — Michaël Rusinowitch — Alakesh Haloi

N° 5230

Juin 2004

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--5230--FR+ENG

INRIA
LORRAINE

Network Traffic Classification for Intrusion Detection

Tarek Abbes , Michaël Rusinowitch , Alakesh Haloi

Thème SYM — Systèmes symboliques
Projet Cassis

Rapport de recherche n° 5230 — Juin 2004 — 20 pages

Abstract: Nowadays enterprises are looking for efficient security devices, like Intrusion Detection Systems (IDS), to supplement the firewalls supervision. Nevertheless, IDS are plugged with several problems that slow down their development: the high speed traffic and the increasing number of attack detection rules. We discuss in this paper new propositions to solve the outlined problems. Our first contribution consists in defining a new classification algorithm that splits the traffic using security policies and IDS characteristics.

The proposed method can also be applied to quickly verify the detection rules. However, the memory consumption may grow up due to the increasing number of these rules. Therefore, we propose an efficient method to match the detection rules as our second contribution. The main idea is to properly organize the rules. This enables us to restrict the verification domain to only some ranges by taking advantage of the similarities and the differences between the different parts of the detection rules.

Key-words: Intrusion Detection, Traffic Classification, Detection Rules Organization.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 —Télécopie : +33 3 83 27 83 19

Classification du trafic réseau appliquée à la détection d'intrusion

Résumé : Les entreprises utilisent de plus en plus les systèmes de détection d'intrusions (IDS) afin de compléter le travail des pares feux. Les IDS doivent traiter deux problèmes: le haut débit et le nombre croissant des signatures d'attaques. Afin de résoudre ces deux problèmes, nous proposons dans la première partie de ce rapport une nouvelle approche de classification géométrique des règles. Nous appliquons ensuite la méthode pour diviser le trafic réseau. Nous définissons alors des règles de division du trafic déduites de la politique de sécurité de l'entreprise et des caractéristiques des IDS déployés sur le réseau.

La méthode de classification s'applique également pour vérifier rapidement les règles de détection d'attaques. Néanmoins et afin de réduire la complexité en espace de la recherche géométrique, nous proposons une méthode alternative pour vérifier une large liste de règles de détection d'attaques. La nouvelle technique se base principalement sur une organisation adéquate des règles de détection d'attaques en tirant profit des similarités et des différences entre les différents champs définis par les règles.

Mots-clés : Détection d'intrusion, Classification du trafic, Organisation de règles.

1 Introduction

Nowadays, firms and organizations suffer from the increasing number of illegal intrusions that they undergo every day. To preserve their security, they rely on Intrusion Detection Systems (IDS). An IDS monitors the system and the network resources and activities. It uses the information gathered from these resources to identify intrusions and notify the authorities. IDS's employ 2 families of detection methods: anomaly detection and misuse detection [2, 5]. Anomaly detection models the normal behaviour of the system and its users. It uses statistical tools and sophisticated learning machines. Then, a substantial deviation from normal behaviour can reveal some attacks. The main drawback of this method is the huge number of false alerts it generates. Nevertheless, it is able to discover new forms of attacks. The second approach, called "misuse detection", constructs databases for known signatures and scenarios of attacks. The intrusion detection system (IDS) analyses the logs and audit files to find the bad scenarios. They use several methods like pattern matching, expert systems, Petri nets, and model checking [17].

In this paper we are interested by the misuse detection approach. All the listed methods try to detect various types of intrusions. However, it would be interesting to pre-process the inspected traffic in order to quickly detect attacks. Packet classification is an attractive technique for many network applications such as IP routing, service differentiation and firewall filtering. In the same way, intrusion detection systems can benefit from the classification techniques in order to solve the challenging problem of supporting the high speed traffic. By classifying the network traffic, we can distribute the analysis among several IDS which ensure faster detection. Besides, each IDS maintains less detection rules and provides sharper analysis since it is aware about the traffic nature. We thus deploy more lightweight IDS presenting only the required functionalities in order to examine the forwarded traffic classes.

The work of Kruegel et al. [14] is pioneering in the domain as it is the first one to exploit the traffic classification. Charitakis et al. [11] propose another method to subdivise the network traffic (see Section 4). However, these works did not take into account the enterprise security policies and the IDS characteristics. Our first contribution in this paper is to define new rules that describe these criteria when splitting the traffic. We also propose a new algorithm that quickly processes the classification rules. The method combines a geometric search and a clever byte verification to reduce the number of overlaps between rules.

After traffic classification, we need to supervise the packet contents in order to filter possible attacks. The detection mechanism within a network based IDS mostly goes through an iterative verification of a list of detection rules. Some previous work optimise this process by clustering the rules [13] or factorizing common parts [1]. Our novel idea is to use the number of explicit fields in attack detection rules in order to classify them in different groups. Afterwards, we sort the rules inside each group and we compute relations of bad and good matching between the different classes. The method optimizes the search and therefore provides faster answer to detected attacks.

We start this paper by explaining in Section 2 the advantages of traffic classification for IDS. Then, Section 3 presents the format of the rules used to classify the traffic and outlines the overlap problem encountered when processing these rules. Section 4 reviews classification works, mostly in the context of firewall and routing applications. In Section 5, we present our approach for classifying network's traffic while in Section 6 we detail the organization of attack detection rules stored in IDS. We present in Section 7 some experimental results. Finally, we conclude our work in Section 8.

2 Traffic Classification Advantages

Network traffic classification offers several opportunities to improve intrusion detection. Indeed, the administrators find solutions for supervising both overloaded networks and switched environments. Others benefits such as fault tolerancy, false positives reduction and log files optimization will be progressively explored in this section.

2.1 High Traffic Analysis

The intrusion detection systems have to support high traffic in order to supervise the activities of high speed networks. This task becomes increasingly difficult with only one IDS. The traffic division constitutes an efficient solution to solve this problem. The division is based on classifying the traffic to be analyzed. Then every class will be forwarded to only one specialized IDS or a group of them to ensure load balancing. Therefore, every intrusion detection system contains less detection rules and performs a more detailed protocol analysis by limiting itself to a subset of protocols.

2.2 Switched Environment Support

Supervising switched network is considered as a complex task since the traffic will be divided in different segments. The multiple deployment of the same IDS on every segment constitutes a possible solution but it is very expensive and non feasible in real situations. An interesting alternative is to classify the traffic on every switch and to forward the different classes to the right IDS responsible for the corresponding traffic type. We avoid thus the installation of several identical IDS and we replace it by a lightweight traffic splitter.

2.3 Fault Tolerance

IDS sniff the network traffic and behave consequently as fail open devices. Therefore, skilled hackers first try to disable IDS before attacking others hosts in the network. They frequently rely on Denial of Services attacks to achieve their goals and facilitate further attacks on the remaining hosts of the network.

A traffic splitter forwards each traffic class to the corresponding IDS. It simultaneously controls the IDS activities in order to quickly detect any decrease in the performance. It also balances the traffic when noticing an abnormal increase of the load. Moreover, it discharges a stressed IDS, and sends its traffic classes to another IDS. In a nutshell, the splitter defeats the attackers intentions to disable IDS.

2.4 Detection Efficiency

Traffic classification solves some intricate problems in intrusion detection such as asymmetrical routing. In fact, an IDS relying on protocol analysis can miss some attacks because it receives only a part of the traffic. In the same context, it may generate false positives since it expects to receive some packets that are not captured because they take another route.

The resolution of the asymmetrical routing contributes to detect more attacks and allows one to resist evasion techniques. In fact, the IDS pattern matching routines will be able to filter attack signatures present on several packet fragments sent on various routes. Obviously, such detection should be the task of a single IDS.

2.5 Honeypots Deployment

The traffic classification is very useful to forward suspect traffic classes to honeypots. Indeed the analysis of the intrusive activities and the stressing IDS cases reveals the attack origins. Consequently, the splitter directly sends the suspected traffic to honeypots in order to know more about attackers objectives and techniques.

2.6 Log Files Optimization

The elaboration of log files is an interesting step in the execution of an intrusion detection system. Besides, the content of these files give a clear idea about the detection quality. Unfortunately, administrators are overwhelmed by huge security files containing a lot of false positives. Moreover, hackers initiate many attacks from spoofed addresses in order to hide their true identities inside other entries in the log files. The improvement of the detection efficiency already mentionned in Subsection 2.4 and generally expressed by the ROC criteria (Receiver Operator Characteristic) contributes to a better log files design.

In addition, packet classification allows a separate analysis of each traffic class. Therefore we obtain shorter and independent log files. The number of false positives will be reduced. This leads to an easier exploration and a faster attack search inside security files.

3 Traffic Classification Rules

We present in this section different forms of traffic classification that precedes the intrusion detection. We express the classification operations by means of rules. The first rule to be satisfied will forward the traffic to the corresponding IDS, to a specific honeypot or will reject the traffic. A rule can be divided in 2 parts. The left hand side called "header" contains a set of parameters to be matched by the packet. The set of these parameters is variable but it includes in general the IP addresses and ports. In addition, we have a direction parameter. It indicates the source of the traffic. It can be either an incoming or an outgoing traffic. The direction of the traffic is very useful for the detection process. In fact, when we adopt a protocol anomaly strategy, we need to observe the 2 directions to supervise the protocol execution and the client/server exchanges. However, when we rely only on a pattern matching, we have to know the direction of the traffic in order to apply the appropriate signatures. In the remaining parts of this paper, we assume that by default a rule is bidirectional.

The right hand side of the rule specifies the action to be taken when the header part is matched. We can forward the traffic to a specific IDS, send it to a particular honeypot or just drop it. We give below the schema of a traffic classification rule.

$$R : (field_1, field_2, ..., field_n)_{direction} \rightarrow action \; where \; direction \in \{in, out\}$$

3.1 Types of Classification Rules

We present in this section the different models for a classification rule. These templates can be composed to derive more complex rules.

3.1.1 Type 1: (address, port)$_{direction}$ $\rightarrow$ action

We frequently employ this model to supervise the services provided by a server. We give in the following some examples about the use of the filter.

$R_1 : (adr\_serv\_http, port) \rightarrow IDS_1$ where
adr_serv_http : 32 bits IP addresses of the Web servers installed on the home network.
port : a set or a range of 16 bits integers.
IDS_1 : An IDS devoted to analyse the HTTP protocol.

$R_2 : (adr\_serv\_rpc, port) \rightarrow IDS_2$ where
adr_serv_rpc : 32 bits IP addresses of machines running some RPC services (NFS, NIS, mount, etc.).
port : a set of 16 bits integers.
IDS_2 : An IDS devoted to analyse the RPC protocol and some others services based on it.

$R_3 : (adr\_machine\_ssh, port) \rightarrow NULL$ where
adr_machine_ssh : a 32 bits IP machine address using ssh.
port : the port used for tunnelling the traffic.
NULL : It is a void action. In fact, the data are encrypted and thus it will be impossible for a NIDS to filter the traffic.

3.1.2 Type 2: (address, *)$_{direction}$ → action

The second category of rules is generally used to tune the level of verification and protocol analysis exploration depending on the remote networks.

R_4 : $(adr\_network\_1, *) \rightarrow IDS_3$ where
adr_network_1 : It is a prefix of a 32 bits IP address. It represents thus a network address.
IDS_3 : the network adr_network_1 is a trusted network. The intrusion detection system IDS_3 performs a light analysis. Moreover, in overload situation, this type of traffic will be first neglected since it has low priority for analysis.

R_5 : $(adr\_network\_2, *) \rightarrow Honeypot$ where
adr_network_2 : It is a prefix of a 32 bits IP address. It represents thus a network address.
Honeypot : the network adr_network_2 is too suspect and was often been a source of attacks. The analysis of its traffic will be done in a honeypot.

3.1.3 Type 3: (*, port)$_{direction}$ → action

It defines rules used to supervise particular services. Generally the analysis is only based on a pattern matching strategy in order to filter a set of attack signatures. We notice the absence of addresses and thus the supervised connection involves an ordinary host in our home network that plays a client role. We choose to avoid the protocol analysis just because our servers are not included. Generally these rules are indexed by a direction in order to restrict the supervision to only incoming flow.

R_6 : $(*, port\_http)_{in} \rightarrow IDS_4$ where
port_http : a set or a range of ports often used by web servers.
IDS_4 : an IDS that supervises the incoming HTTP flow for a set of attack signatures.

3.1.4 Generalization

By composing the 3 previous schemas, we may obtain new filters. For example to supervise the IPv6 traffic between 2 gateways, we can design the following rule (R_7).

R_7 : $(adr\_1, port\_1)(adr\_2, port\_2) \rightarrow IDS_5$ where
Adr_1 : a 32 bits IP machine address of the first gateway.
Port_1 : the port used by adr_1 for tunnelling the traffic.
Adr_2 : a 32 bits IP machine address of the second gateway.
Port_2 : the port used by adr_2 for tunnelling the traffic.
IDS_5 : it is an intrusion detection system that supports the IPv6 analysis. Besides extracting the new IP header, the IDS checks the integrity of the traffic by verifying the AS field. In addition, it may supervise the ESP field if it knows the encryption key. It should be noted that since the IDS contains critical information (encryption keys), access to this machine should be restricted and all communications have to be encrypted.

3.2 Discussion

IP addresses can be defined as a 32 bits string refering to particular machines to be supervised. In addition, they can be represented on fewer bits to express network addresses. The address schema does not follow the classical partition of network classes : A (the network size is 2^{24}), B (the network size is 2^{16}) and C (the network size is 2^8). However, it uses different prefixe sizes applying the CIDR concept (Classless Inter Domain Routing). Moreover, a prefix can be considered as a range of addresses and conversely, a w bits range represents up to $2w-2$ prefixes. As a consequence, the address parameter in our classification rules leads to several overlapping forms, which complicate the classification execution (Figure 1).

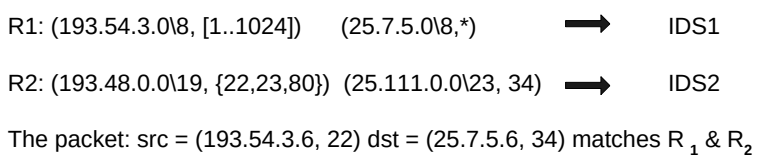

Figure 1: Rules overlap

The same problem appears with the port parameter. In fact, the port field in a classification rule can take a value of 16 bits integer. It may also be defined by a set or a range of values (Figure 1). The overlapping becomes more complicated when considering the two port parameters: the source and destination ports. As a result, many rules could be satisfied at the same time and we have to find a method which quickly extracts all candidate rules and choose the suitable one. Our strategy supposes that the classification rules are well sorted by a priority criteria. So the first matched rule will be considered to forward the traffic. This differs from the routing strategy that always considers the longest prefix match.

4 Previous Work on Packet Classification

In the last decade, the classification problem was well investigated in many application domains. For instance, routing algorithms try to find the longest IP address prefix in order to route the traffic. In the same context, firewall filtering and services differentiation protocols rely on similar rules to classify the traffic. Gupta and McKeown have given a survey on classification algorithms [10]. They classify the existing methods in 4 categories: basic, geometric, heuristic and hardware algorithms. We summarize in the following these methods and we add some new entries in order to update their work.

- basic data structures: the simplest data structure is the linked list of rules stored in order of decreasing priority. The linear search algorithm uses this representation to match the suitable rule. Other basic algorithms employ binary tries such as Hierarchical Trie and Set pruning Trie algorithms [27].
- geometric methods: classifying packets can be viewed as a geometric search. In fact, a rule in d-dimensions represents a d-dimensional hyper-rectangle. A classifier is therefore a collection of prioritized hyper-rectangles, and a packet header represents a point in d-dimensions [10]. Among the existing algorithms, we cite the Geometric Efficient Matching [20], the grid of trie [25], the cross producting [25], the 2-dimensional classifier [4], the Area Based QuadTree [4] and the Fat Inverted Segment Tree [7].
- heuristics methods: Most geometric methods present high complexity bounds. To support this complexity, practical solutions use heuristics. Among the proposed algorithms, we cite RFC [8], Hierarchical Intelligent Cutting [9], Tuple Space [23] and Woo Approach [29].
- hardware methods: Packet classification can be directly implemented on fast hardware devices. We give for instance the Ternary CAM, the Bitmap Intersection [15] and the Aggregated Bit Vector methods [3].

The listed algorithms are intended to find the suitable rule with the highest priority, and thus they are more adapted to firewalls filtering. However, unlike in firewalls, classification in routers tries to find the longest prefix match. There are two categories of classification algorithms: algorithms based on value and algorithms based on prefix length [21].

The first category uses a simple sequential or binary search. Binary and multi-bit trees are the mostly used data structures. Other optimization techniques can be employed such as path compression [18, 19, 22] or prefixes expansion to have the same length [24]. Additional transformations to generate disjoint prefixes are useful to avoid the use of the longest matching rule. Moreover, the choice of the multi-bit tree is very interesting since it affects the memory access. The stride size can be fixed or variable and it depends essentially upon the prefixes distribution.

The second category of algorithms carries out the search basing on prefix size. In fact, all equal prefixes are stored in the same hash table [28]. Thus, to classify incoming packets, an iterative query is performed in each table until success. Finally, we note that the search problem of the longest prefix match can be transformed to a search of prefixes range [16]. Obviously, the longest prefix corresponds to the smallest range of possible values.

We conclude that the classification problem has been thoroughly analyzed for the purpose of many network applications: routing, packet filtering, load balancing, billing, services differentiation, etc. In the remainder of this paper, we are simultaneously interested to packet filtering and traffic classification in order to improve the intrusion detection. Previous works perform the traffic division in order to balance the network load. However they didn't take into account the enterprise security policies or the IDS characteristics. Kruegel et al. [14] split the traffic by active scenarios on each IDS or destination addresses ranges. The method allows a further stateful analysis. It also reduces the number of rules stored in each IDS by filtering them according to the chosen frame routing strategy.

Charitakis et al. [11] propose a two steps subdivision of the network traffic. The first one performs a pre-filtering operation by detecting general attacks that match generic rules. In addition, empty packets are eliminated since they do not contain data to be analyzed. The second step uses some heuristics to hash some fields of the IP and TCP (UDP) headers and forwards the traffic depending on the obtained hash value. Nevertheless, the method can only be applied to pattern matching intrusion detection approach since the pre-filtering affects further stateful protocol analysis.

Finally, we note that Toplayer [6, 26] offers some commercial solutions in order to divide the traffic. They insist on the fact that the distribution must preserve the flow integrity, i.e. the packets of one flow must be analyzed by the same IDS. This is very important when performing stateful detection or reassembling traffic in order to prevent hackers evasion tactics.

5 Traffic Classification

We explain in this section our approach to classify the network traffic. We consider that the classification rules have 4 dimensions: source and destination addresses and source and destination ports. The number of these parameters can be extended without any change in the method. The classification proceeds in 2 steps. We first perform a geometric search in a 2 dimensional space in order to cluster rules by the port criteria. We construct thus several sets of candidate rules that fall in the same ranges of source and destination ports. The second step processes each group of rules in order to represent them in the form of graph (Figure 2). At runtime, classifying the traffic is an easy task since it suffices to look for the suitable directed graph to traverse. We detail in the following our algorithms and the datastructures used for each step.

5.1 Classification with the Port Criteria

The first step is to construct the sets of candidate rules with the same ranges of source and destination ports. The port definition in a classification rule can have simple integer values of 16 bits (from 0 to $2^{16} - 1$), or a more complex format given by sets and ranges. This leads to the overlap problem between classification rules. To solve

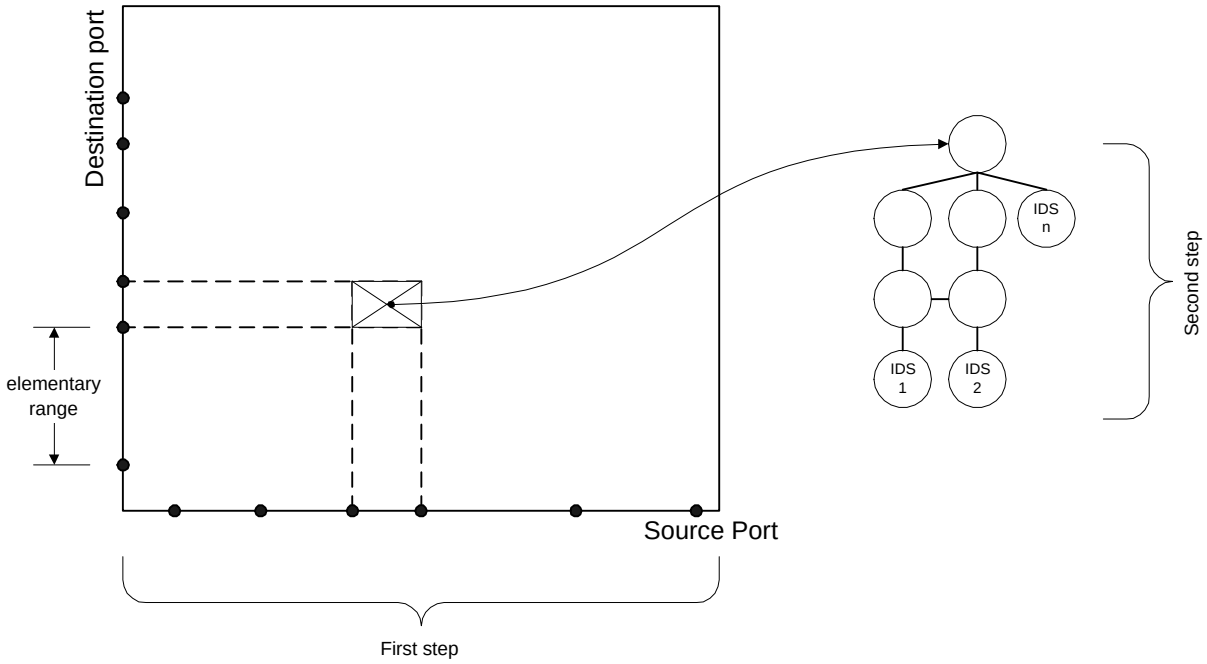

Figure 2: Trafic Classification

it, we divide the possible values and ranges in elementary ranges. Note that a range can be represented by its 2 endpoints. Hence to find a range, it suffices to consider one of the 2 boundaries. For instance if we choose the lower endpoint, we can look for the greatest endpoint smaller than the port value (Figure 3). We accomplish this search in sublinear time. After finding the elementary intervals of the source and the destination ports, we point to the graph used to classify the packet by address criteria.

5.2 Classification with the Address Criteria

The first step examines only the source and the destination ports to initiate the classification process. At the end of this phase, we obtain several groups of rules. The second phase builds a directed graph for each group. Then traversal of the graph defines the suitable IDS (or action) to forward the traffic. While the classification graph can be applied to any multidimensional rules, we only use it with two parameters: the source and the destination addresses. This is due to the nature of our classification rules. Therefore, our unique data in this step are the IP addresses.

The IP addresses used in the classification rules present some peculiarities that make the search operation a complex task. In fact, they can be defined on a number of bits lower than 32 bits. In this case, they represent a range of addresses which leads to rules overlap. The algorithm that we propose postpones the processing of these overlaps. In fact, during the graph construction, we divide every dimension to a set of bytes. The constructed graph looks like a multi-bit graph with a stride size equal to 8. An IPv4 address contains 4 bytes so that a host is represented by 4 complete bytes while a network is assigned less complete bytes and only one partial byte. The partial byte can be viewed as a couple of integer, the first one being the byte value and the second one is the number of masked bits (Figure 4). We concatenate all the dimensions and we modify the bytes order. We first

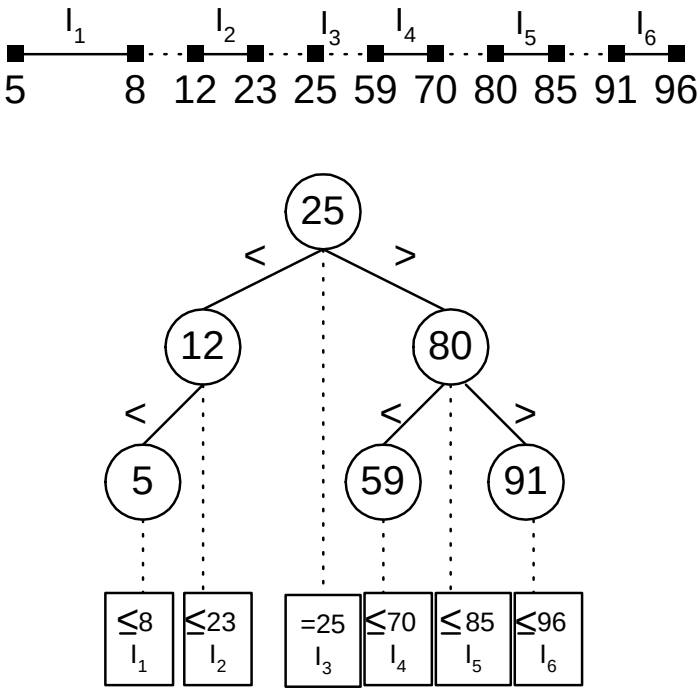

Figure 3: Port Ranges Look-up

put together all complete bytes and we complete them with the partial bytes. The number of rules that match the same complete bytes sequence will be low and therefore we minimize the number of overlapping partial bytes.

152.81.51.30 : IPv4 Address with 4 complete bytes

152.81.51/11 : IPv4 Address with 2 complete bytes and 1 partial byte equal to 51/3

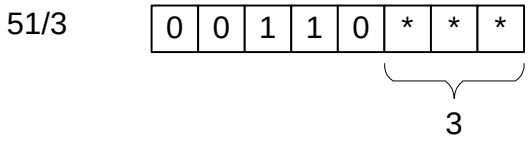

Figure 4: Byte Presentation

The graph construction consists in transforming complete bytes into labeled links between nodes. When we find at the same position partial and complete bytes, we only process complete bytes and we represent the partial bytes by a void link. At the end of the complete bytes sequence, we analyse the partial bytes. We start with the long prefixes since they represent small intervals. Then, we link them to overlapped smaller prefixes. Finally, we label the final nodes by the classification rules having the highest priority. This priority follows the apparition

order in the rule list. While our algorithm processes traffic classification rules, we note that we can adapt it to classify detection rules stored in every IDS. We discuss this point in the next section.

To each constructed link we associate a new node (see Figure 5). A node in the graph contains 3 parameters. The first one defines the type of the node and can take the value "C" to indicate that the next edges describe complete bytes. If the type of the node is "M" then we inspect a masked byte. Finally the type "F" denotes a final node that stores the action of the successful classification rule. The second node parameter is the search level, i.e. the dimension that we are verifying. The last parameter is a record of pointers that expresses the search progress on each dimension. Initially, all pointers have a null position except the first one which points to the first byte of the first dimension. As long as we progress in the search, the pointers are incremented (see Figure 5).

In summary, a "C" node contains several labeled links to the following nodes and possibly a void link if we encounter a partial byte. The "M" nodes have also many labeled links and a void link if there is an overlap between prefixes. The last node whith a type "F" does not offer any further links but it contains the winning classification rule.

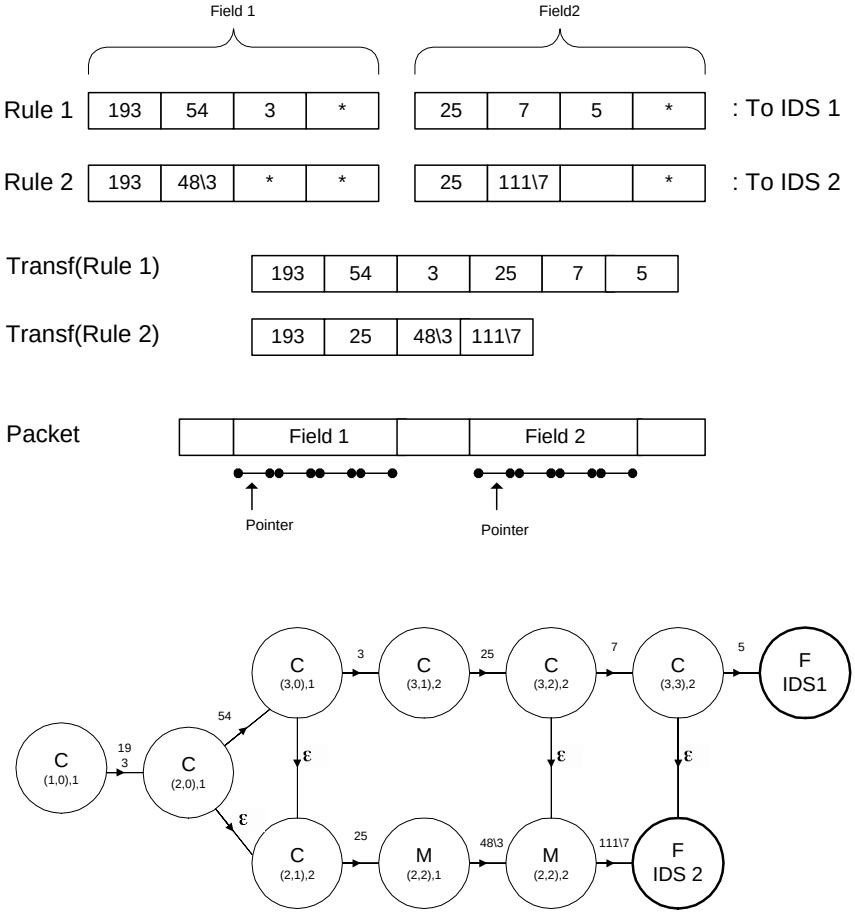

Figure 5: Classification Graph

5.3 Classification Algorithms

The construction of the graph is done recursively. We start from the initial node. We apply the construction function *Build*, which given a node, generates new links that correspond to possible complete or partial bytes. At each link, we associate a new node that will be also processed by the *Build* function. The nodes generation ends when it remains no complete and partial bytes. We thus mark the current node with "F" and we assign to it the rule having the highest priority. We present the whole algorithm for constructing the classification graph in Algorithm 1.

During the search time, we prefer labelled links over void ones. We also favor the long prefixes links to the short ones. We describe this strategy in Algorithm 2.

```
begin
    if node.type=C then
        foreach complete byte do
            Construct a new state S' and a labeled link
            Assign a type to S'
            Assign candidate rules to S'
            Build (S')
        if it exists a partial byte then
            Construct a new state S' and a void link
            Assign a type to S'
            Assign candidate rules to S'
            Build (S')
    else if node.type=M then
        for size(mask) from 1 to 7 do
            foreach prefix value do
                Construct a new state S' and a labeled link
                Assign a type to S'
                Add void links between states that overlap with S'
                Assign candidate rules to S'
                Build (S')
    else if node.type=F then
        Label final nodes by the winning rule
end
```

Algorithm 1: Classification Graph Construction : Build function

Our classification method presents a good time complexity that is in the worse case equal to the size of the longest rule in term of complete and partial bytes. We denote by d the number of available dimensions, f_i the i^{th} field and N the total number of rules. We also denote by r a classification rule and by R the set of all rules. Then the amortized cost is $\sum_{i=1}^{d} \mid r.f_i \mid / \mid R \mid$ The good lookup time is penalized by a large space complexity that is in the worse case equal to $min(N^{4d}, 256^{4d})$. The latter limit is never reached since network addresses are defined on less than 4 bytes and we have common complete and partial bytes. Beside this, the number of rules will never be very high due the small number of possible IDS that will analyze the forwarded traffic.

```
begin
    Node = initial state of the Graph
    if the type of the node is C then
        if it exists a labeled link then
            Fire a transition
        else if it exits a void link then
            Fire a transition
        else
            Drop the packet
    else if the type of the node is M then
        for prefix size from 1 to 7 do
            if it exists a labeled link then
                Fire a transition
                exit
        if it exists a void link then
            Fire a transition
        else
            Drop the packet
    else if the type of the node is F then
        Forward the packet
end
```

Algorithm 2: Lookup with Classification Graph

5.4 Fragmentation Problem

The traffic splitter must ensure the flow integrity especially in the case of fragmented packets. In fact, the IP fragment does not usually include the TCP header but only the first one contains this information. Therefore the traffic splitter cannot apply the classification rules that use the ports parameters. To solve this problem, we keep in the splitter the IP packet identifier of the first fragment as well as the winning rule to classify this packet. All fragments can now be routed to the suitable IDS. These entries are stored for a certain time period and are recorded with the IP source address to avoid collision.

6 Detection Rules Classification

We have presented in the previous section our algorithm to classify the traffic and forward it to the suitable IDS. The same method allows one to quickly choose the candidate detection rules stored in each IDS. In fact, network based intrusion detection systems frequently apply a misuse approach to detect intrusions. They rely on a set of detection rules which define some tests to perform before triggering the alarms. Snort is such an IDS that presents a huge list of detection rules. For fixing ideas, we base our discussion on its rules but we can adapt the proposed solutions to any other IDS rules.

A detection rule in Snort has 2 parts. The first one, called header, defines the source and the destination addresses and ports. It is used to build a particular structure called RTN (Rule Tree Node). The second part is the body of the rule and it contains all options to be verified in order to detect an attack. All these options will be stored in a special structure called OTN (Option Tree Node). The two described structures (RTN and OTN) are arranged in a 3 dimensional linked list which form the detection rule list. At running time, the IDS

analyzes every packet by traversing the rule list of Snort. It starts by the RTN structure and if it is verified then it inspects the OTN structures attached to it. If not, or when failing to match the OTN list, the IDS checks the next RTN. It continues its traversal until finishing the rule list or detecting one attack. The detection goes through the verification of all options stored in an OTN structure.

The iterative examination of the detection rules list is very expensive since many useless tests are done. In fact, the IDS does not preprocess the similarities and differences between RTN structures and does not use these relations to cleverly performs next verifications. Concretely, if an RTN with a source port equal to 80 is checked, then we can skip all other RTN with a source port different from 80 (good match property). In the same context, if we fail to match a destination port equal to 111 then we skip all similar RTN (bad match property).

An obvious solution is to use our algorithm for classifying the traffic and adapt it to verify the detection rules. If we try to classify for example the OTN structures based on the RTN structures then our classification criteria are again the source and the destination addresses and ports. The main difference between the two classification graphs is the content of the final nodes. In fact, the final nodes in a traffic classification graph store only one rule having the highest priority. However, with a detection rules classification graph, these nodes contain all the candidate rules (OTN structures). The solution ensures a quick examination of detection rules by avoiding the iterative traverse of an RTN linked list. However, the number of detection rules usually increases because of the new kind of attacks discovered every day. This fact can hinder the classification graph due to its relatively high space complexity. Thus, we propose another method to quickly verify the detection rules and decrease the memory consumption. This solution is based on a suitable organization of detection rules.

6.1 Detection Rules Organization

The RTN structures are indexed by source and destination addresses and ports. Every parameter can take an explicit value or a generic one (denoted by a wildcard). Our idea is to organize the RTN by their explicit fields. For example, rules that only define the source address parameter are grouped together and will appear before rules that assign values to all parameters. After that, we sort each RTN group by a lexicographic order. This allows us to perform sublinear searches inside the group.

In our method, we decide to prioritize the most generic RTN groups. This is because during a pre-processing phase, we mark for each value in these groups, its possible ranges in more specific RTN groups (see Figure 6). Obviously, the search inside these intervals would be sublinear since we have already sorted the RTN structures in lexicographic way. A second optimization consists in adding some *white rules* in first groups just to mark ranges in next RTN groups. This is very interesting when some values are frequently observed in higher specific RTN groups.

Finally, we subdivide the RTN groups in two parts. The first one contains all rules with explicit values already appeared in previous groups and the second one includes the rest of RTN structures. The division accelerates the search since a failure to match all upper RTN classes means that next good RTN match can be found only in the second part of the group.

We give in Algorithm 3 the method for sorting the RTN structures. The constructed RTN list replaces a rules classification graph known to be space greedy despite its efficiency. We then schematize the sorting/search mechanism in Figure 6. Note that we have only used three dimensional rules to represent all possibilities.

6.2 RTN Search

The new RTN schema avoids the iterative traversal of the detection rules list and therefore contributes to fast detection inside IDS. In fact, the search becomes sublinear in time in the first RTN group and sublinear in only some ranges of generic RTN groups. The search is slower than a rule classification graph as explained in the previous section but it consumes less memory. Indeed, the only extra space allocated for saving the RTN list is for pointers between the RTN groups and the white supplementary rules.

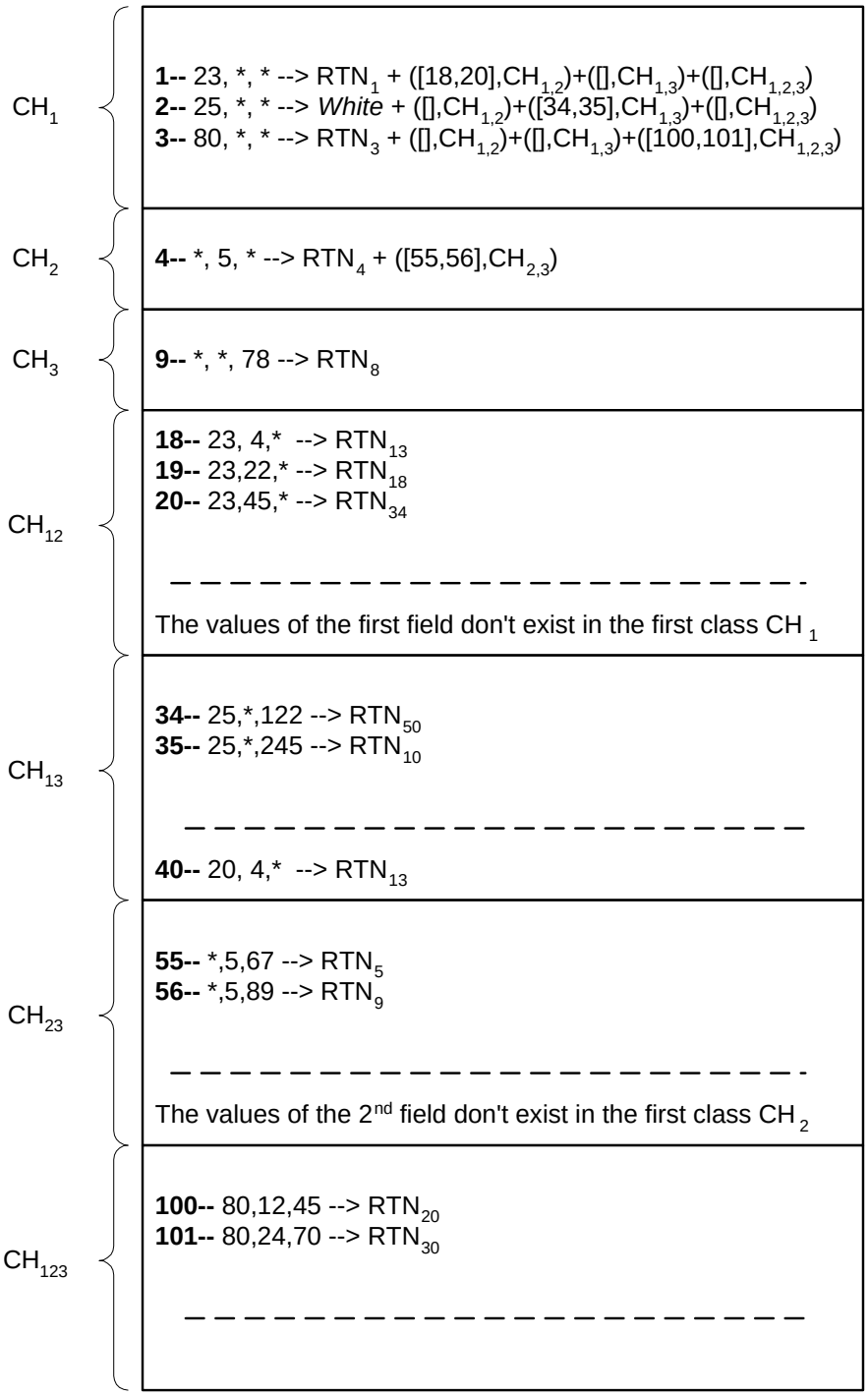

Figure 6: Detection rules search

begin
Group the RTN structures by the number of explicit fields
Extract the frequent values that appear only in specific RTN groups
Add white RTN structures in generic groups
Divide each group on 2 parts
1- the first subclass contains all rules with a first field value appearing in upper classes
2- the second subclass contains the other rules
Sort the rules in each group using a lexicographic order
Concatenate all groups and enumerate the rules
foreach *first field value in generic groups* **do**
compute in specific groups the ranges of rules with the same value
end

Algorithm 3: Detection Rules Organization

7 Experiments

The main goal of the experiments described in this section is to establish the performance of the traffic classifier in real time high speed traffic environment. The traffic classification algorithm is implemented and used for classifying and forwarding the traffic into a number of virtual IDSs. It is assumed that once the traffic is classified, it will follow a normal route via the assigned IDS. For our second contribution about the detection rules organization, the implementaion is not yet finished.

To run our experiments we used the traffic produced by MIT Lincoln Labs as part of the DARPA IDS evaluation. To be more precise we used the outside tcpdump data of Friday, second week 1999. The traffic log was injected to the Ethernet interface eth0 at different rate in Mbps by means of tcpreplay. Then we create different log files corresponding to the amounts of traffic to be forwarded to each IDS.

The first experiment aims at calculating the time taken to process a fixed amount of network traffic at varying network traffic speed. Ideally we should obtain a plot for inverse of the network traffic speed. In real time experiment we obtain a similar plot in Figure 7.

The second experiment aims at measuring the performance of the classifier against different traffic speed. We define a performance index for the classifier as follows.

$$Performance = \frac{Rate_{classifier}}{Rate_{network}}$$

$Rate_{network}$ is defined as the network traffic speed. Similarly, $Rate_{classifier}$ is defined as the amount of traffic handled by the classifier at a given network traffic speed. It is calculated by measuring the total system time taken to classify a fixed amount of network traffic generated at $Rate_{network}$ by tcpreplay from DARPA evaluation log file. Performance is measured as the ratio of these two because it has been observed from real time network behaviour that as network traffic speed increases the rate of handling traffic by classifier tends to decrease. Ideally we should obtain a plot parallel to x-axis. However in real time traffic when network speed approaches the hardware speed limit, packet loss tends to happen thereby reducing the number of packet received by the classifier and hence the plot tends to have a negative slope. In our experiment, we observe that performance index starts falling when network traffic exceeds 170 Mbps (Figure 8).

Our third experiment was aimed at measuring amount of memory consumed by the classifier against various number of configuration rules. This was accomplished by calculating the number of active nodes in the classification DAG in runtime. In this experiment we observe that as expected the plot of nodes created in runtime against the number of rules in rule file is almost linear (Figure 9). So the memory used by nodes created as a part of the DAG also tend to be linear (Figure 10).

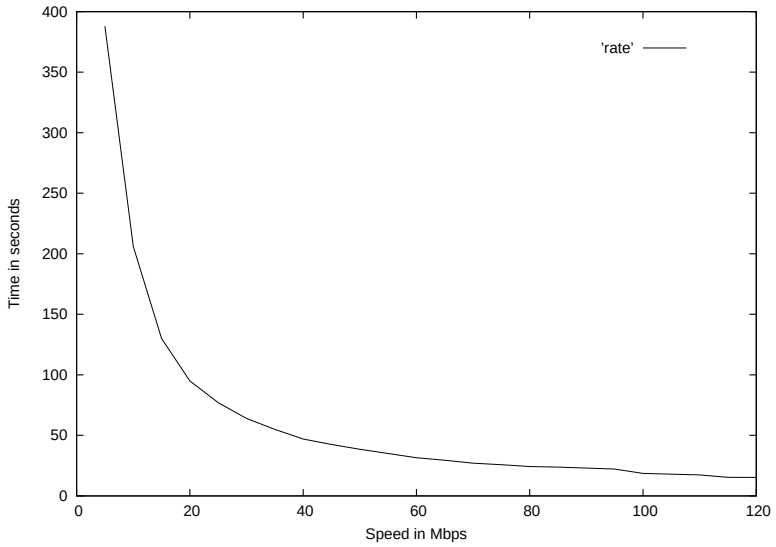

Figure 7: Classification time against network speed

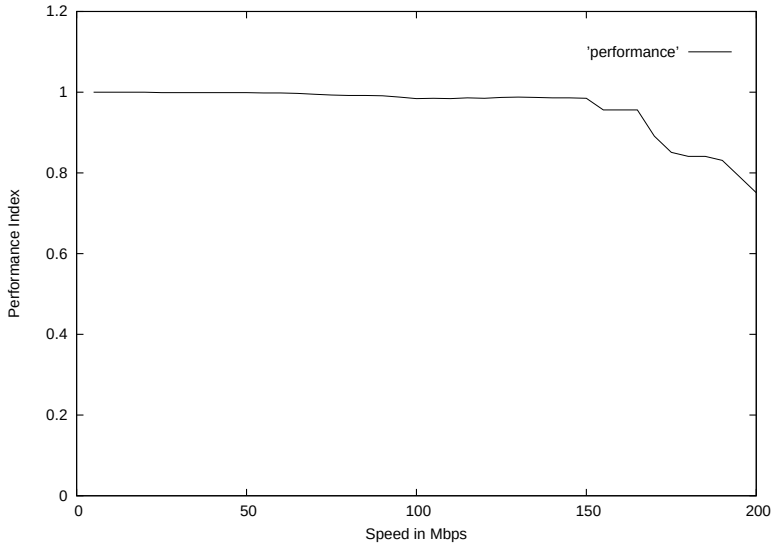

Figure 8: Performance index against network speed

The results obtained from above experiments makes it clear that the classifier is capable of handling packets in high speed network. The degrading performance of the classifier beyond 170 Mbps speed can be attributed to packet loss happening because of decrease in performance of pcap library.

8 Conclusion

Intrusion detection systems have to face the challenge of high speed traffic. In order to tackle it, we propose a new approach for classifying the network traffic. The method constructs classification graphs that postpone the

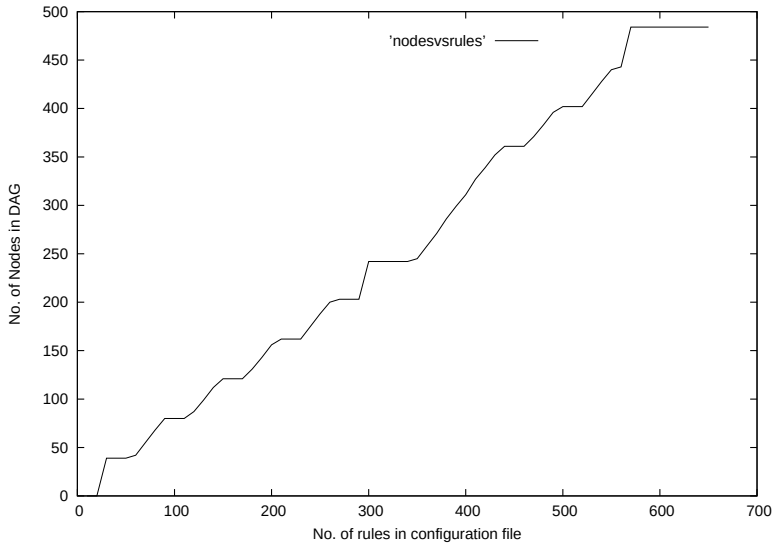

Figure 9: No. of nodes in DAG against no. of rules

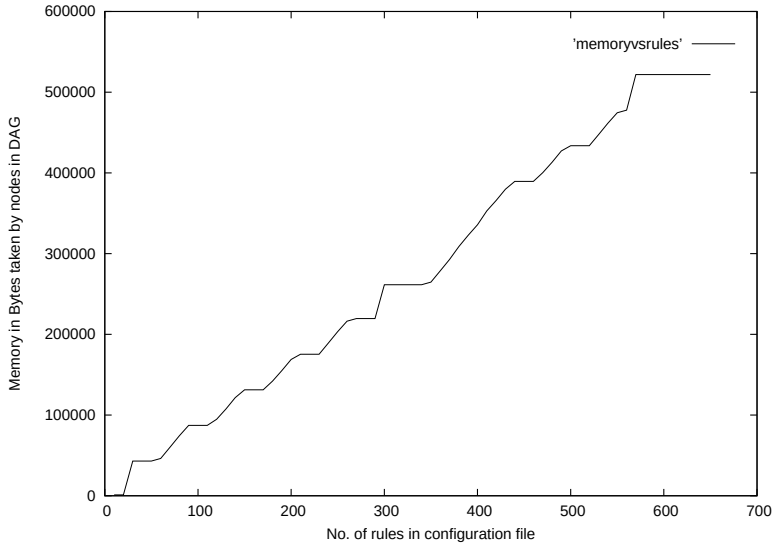

Figure 10: Memory used by nodes in DAG against no. of rules

analysis of overlapping rules to last stages. Beside the load balancing benefit, our classification contributes to satisfy the enterprise security policies from one side, and to take into account the IDS capabilities on the other side.

The classification graph can also be used to verify the detection rules frequently used by network intrusion detection systems. Nevertheless, because of the large number of these rules, we propose an alternative way to inspect them. The main idea is to properly organize the detection rules and to take advantage of the differences and the common parts between rules, in order to perform faster and clever analysis.

We plan in a near future work to first complete the experiments about the detection rules organization. Secondly, we will supervise the IDS activities in order to detect overloaded one and to dynamically balance the

forwarded traffic to equivalent intrusion detection systems. If there are no such IDS then the splitter acts as a traffic differentiator for the stressed IDS and rejects all traffic that match lower priority rules. We note that these solutions may require a dynamic updating of the classification graph.

References

[1] T. Abbes, A. Bouhoula, and M. Rusinowitch. Efficient pattern matching in intrusion detection. *to apper in Annales de Telecommunications.*

[2] S. Axelsson. Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Chalmers Univ., March 2000.

[3] F. Baboescu and G. Varghese. Aggregated bit vector search algorithms for packet filter lookups. Technical Report cs2001-0673, University of California, San Diego, June 2001.

[4] M. M. Buddhikot, S. Suri, and M. Waldvogel. Space decomposition techniques for fast layer-4 switching. In *Proceedings of the IFIP TC6 WG6.1 & WG6.4 / IEEE ComSoc TC on Gigabit Networking Sixth International Workshop on Protocols for High Speed Networks VI*, pages 25–42. Kluwer, B.V., 2000.

[5] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. *Comput. Networks*, 31(9):805–822, 1999.

[6] S. Edwards. Network intrusion detection systems: Important ids network security vulnerabilities. Technical report, Top Layer Networks, Inc., September 2002.

[7] A. Feldmann and S. Muthukrishnan. Tradeoffs for packet classification. In *INFOCOM (3)*, pages 1193–1202, march 2000.

[8] P. Gupta and N. McKeown. Packet classification on multiple fields. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 147–160. ACM Press, 1999.

[9] P. Gupta and N. McKeown. Packet classification using hierarchical intelligent cuttings. In *Proceedings Hot Interconnects VII, Stanford*, pages 147–160, August 1999.

[10] P. Gupta and N. McKeown. Algorithms for packet classification. *IEEE Network*, vol. 15, no. 2, pp. 24-32, 2001.

[11] I.Charitakis, K.Anagnostakis, and E.Markatos. An active traffic splitter architecture for intrusion detection. In *Proceedings of 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2003), Orlando*, pages 238–241, October 2003.

[12] E. Kohler, J. Li, V. Paxson, and S. Shenker. Observed structure of addresses in ip traffic. In *Proceedings of the second ACM SIGCOMM Workshop on Internet measurment*, pages 253–266. ACM Press, 2002.

[13] C. Kruegel and T. Toth. Using decision trees to improves signature-based intrusion detection. In *Proceedings of the 6th International Workshop on the Recent Advances in Intrusion Detection (RAID'2003)*, LNCS v. 2820, pages 173–191, November 2003.

[14] C. Kruegel, F. Valeur, G. Vigna, and R.A. Kemmerer. Stateful Intrusion Detection for High-Speed Networks. In *Proceedings of the IEEE Symposium on Research on Security and Privacy*, Oakland, CA, May 2002. IEEE Press.

[15] T. V. Lakshman and D. Stiliadis. High-speed policy-based packet forwarding using efficient multi-dimensional range matching. In *Proceedings of ACM SIGCOMM '98*, pages 203–214, 1998.

[16] B. Lampson, V. Srinivasan, and G. Varghese. IP lookups using multiway and multicolumn search. *IEEE/ ACM Transactions on Networking*, 7(3):324–334, 1999.

[17] L. Mé and C. Michel. Intrusion detection: A bibliography. Technical Report SSIR-2001-01, Supélec, Rennes, France, September 2001.

[18] Donald R. Morrison. Patricia : Practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM*, 15(4):514–534, 1968.

[19] S. Nilsson and G. Karlsson. IP-address lookup using LC-tries. *IEEE Journal on Selected Areas in Communications*, 17(6):1083–1092, June 1999.

[20] D. Rovniagin and A. Wool. The geometric efficient matching algorithm for firewalls. Technical Report EES2003-6, Dept. Electrical Engineering Systems, Tel Aviv University, 2003.

[21] M. Ruiz-Sanchez, E. Biersack, and W. Dabbous. Survey and taxonomy of ip address lookup algorithms. *IEEE Network Magazine*, pages pp. 8–23, march/April 2001.

[22] K. Sklower. A tree-based packet routing table for berkeley unix. In *USENIX Winter Conference, Dallas, Texas*, January 1991.

[23] V. Srinivasan, S. Suri, and G. Varghese. Packet classification using tuple space search. In *SIGCOMM*, pages 135–146, 1999.

[24] V. Srinivasan and G. Varghese. Fast address lookups using controlled prefix expansion. *ACM Transactions on Computer Systems*, 17(1):1–40, 1999.

[25] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and scalable layer four switching. In *Proceedings of ACM SIGCOMM '98*, pages 191–202, September 1998.

[26] Inc. Top Layer Networks. IDS balancer with etrust intrusion detection. http://www.toplayer.com/pdf/IDSBdsCAfipdf.pdf.

[27] P. F. Tsuchiya. A search algorithm for table entries with non-contiguous wildcarding. unpublished paper, 1991.

[28] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner. Scalable high speed ip routing lookups. In *Proceedings of SIGCOMM '97*, pages 25–36, September 1997.

[29] Thomas Y. C. Woo. A modular approach to packet classification: Algorithms and results. In *Proceedings of IEEE Infocom2000*, pages 1213–1222, 2000.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399