PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Scientific Research

University of Ghardaïa

Faculty of Sciences and Technology

Department of Mathematics and Computer Science

Laboratory of Mathematics and Applied Sciences

THESIS

Presented for the **MASTER diploma In**: Computer Science

Speciality: Intelligent Systems for Knowledge Extraction

A Frequent Pattern Based Extension of Snort for Intrusion Detection

Presented by Youcef Chettiba and Abdennour Ben Atallah

Jury members

D. Ziadi	PROF	Univ. Rouen France	President
S. Bellaouar	Doctor	Univ. Ghardaia	Examiner
C.A. Kerrache	Doctor	Univ. Ghardaia	Examiner
S. Oulad-Naoui	Doctor	Univ. Ghardaia	Supervisor

College year : 2019/2018

Dedication

Although I will not be grateful enough in these words, I would like to express my sincere appreciation to those who supported me and helped me through my studies.

To my beloved great **parents**, who have been the source of inspiration and gave me strength when I thought of giving up, who continually provide their moral, emotional, and financial support.

To my **brothers**, **sisters**, **relatives** who shared by their words of advice and encouragement to finish this study.

To all my **friends** and my **classmates**, I wish them success in their lives.

Finally, I would like to extend my heartfelt gratitude to those dedicated **professors** in the Mathematics and Informatics department at the University of Ghardaia for their diligence in our studies.

Abdennour Ben atallah

Dedication

Although I will not be grateful enough
in these words, to thank who sup-
ported me through my studies.

To my great **parents**, who never stop giving of
themselves in countless ways, who leads me
through the valley of darkness with light of
hope and support, the symbol of love and giving.

To my **brother**, **sister** : Ali and Fa-
tima who were faithfully and con-
stantly supportive when times were hard.

To all my dearest **friends**(C.s-
'paka',Y.b,L.s,G.k,D.m,B.a,G.a,S.n,N.a,B.l,..)
and my **cousin,cousins** and my classmates
also (O.m) for providing me with encourage-
ment and motivation throughout the years.

To our Professor: OULAD-NAOUI
Slimane who has been a constant
source of Knowledge and inspiration.

Youcef Chettiba

Acknowledgements

"All the praises and thanks to Allah."

We would still like to give our many thanks to our dears beloved parents for their endless guideness and love.

We are grateful to some people, who worked hard with us from the beginning till the completion of the present research particularly our supervisor Dr. **OULAD-NAOUI Slimane** whose steadfast support of this project was greatly needed and deeply appreciated.

We wish to express our sincere appreciation to say thanks to all people who took part of this thesis to make it real.

We are very appreciative to all members of the jury for agreeing to read the manuscript and to participate in the evaluation of this thesis.

ملخص

الصنورت نظام خفيف و مفتوح المصدر للكشف عن اختراق الشبكات يعتمد على قواعد. بالاساس، يتم التعرف على الاختراقات بفضل مجموعة من القواعد اعدت يدويا من طرف خبراء امن الشبكات. في هذه المذكرة، نطور طريقة مختلفة تتمثل في الانشاء الالي لقواعد الصنورت. الفكرة الأساسية هي استخدام خوارزميات الأنماط المتكررة لاستخراج مجموعة من قواعد توصيف حزم الهجوم باستخدام تحليل بيانات التدفق. نقوم بتصميم إطار عمل يتضمن مرحلة المعالجة الأولية ومرحلة التنقيب عن الأنماط المتكررة. نستخدم مجموعة بيانات *LBLN* وفئتين من خوارزميات التنقيب : جميع الأنماط المتكررة (*Apriori* و *FPGrowth* و *FIN*) والأنماط المتكررة القصوى (*FPMax*) كما طورت بالمنصة *SPMF* . تبين مجموعة التجارب في كل من نظامي التشغيل لينكس و ويندوز أن جودة النظام حساسة لقيمة الداعم الأدنى. نتحصل على أفضل نتيجة باستخدام خوارزمية *FIN* بدقة 0.75 عندما يكون الداعم الأدنى يساوي 0.4 .

كلمات مفتاحية: التنقيب عن الأنماط المتكررة، كشف التسلل، الصنورت، تحليل تدفق الشبكات

Abstract

Snort is a lightweight, open source, rule-based intrusion detection system. In principle, malicious traffic is recognized thanks to a manually elaborated set of rules by an expert. In this thesis, we develop a different approach, which consists of automatic generation of snort rules. The basic idea is to use frequent pattern algorithms to extract a set of characterization rules of attack packets using traffic data analysis. We design a framework which includes a preprocessing phase and frequent pattern mining phase. We use the LBLN dataset and two class of mining algorithms: all frequent patterns (Apriori, FPGrowth, FIN), and maximal frequent patterns (FPMax) as implemented in the SPMF library. The set of experiments in both linux and windows shows that the quality of the system is sensitive to the minimum support value. We reach the best result using the FIN algorithm with an accuracy of 0.75 when the minimum support is equal to 0.4.

Keywords: Frequent patterns mining, Intrusion detection, Snort, Network Traffic Analysis

Résumé

Snort est un système de détection d'intrusion léger et open source basé sur des règles. En principe, la reconnaissance du trafic malveillant est fait grâce à une élaboration manuelle des règles de classification par un expert. Dans ce mémoire, nous développons une approche différente, qui consiste en la construction automatique des règles de snort. L'idée de base est d'utiliser les algorithmes de fouille de motifs pour l'extraction des règles de caractérisation intéressantes des paquets d'attaques en analysant des données de trafic. Nous concevons un framework qui inclut une phase de prétraitement et une autre de fouille de motifs fréquents. Nous exploitons le dataset LBLN et utilisons deux classes d'algorithmes de fouille : i) de tous les motifs et ii) de motifs maximaux en exploitant les algorithmes : Apriori, FPgrowth et FIN pour la première classe et FPMax pour la deuxième comme implémentés dans la bibliothèque SPMF. Les test effectués sur différents scénarios d'attaques sur les systèmes linux et windows montrent que la qualité du système est sensible à la valeur du support minimal. Le meilleur résultat est obtenu avec l'algorithme FIN avec une exactitude égale à 0.75 lorsque le support minimal est fixé à 0.4.

Mots clés : Fouille de motifs fréquents, Détection d'intrusion, Snort, Analyse de trafic réseau

Contents

List of Figures

List of Tables

Introduction

Due to the tremendous development that has taken place on the internet in the last decade, the use of the Internet network has become indispensable from anyone, e-commerce, bank accounts are the most sensitive things, the tremendous growth of users and data in the Internet makes a challenge to protect this content from malicious users and illegal dealing, in order to achieve the notion of information security. In the past there has been an evolution in the number and types of tools to breach the security of information. In parallel, there has been an evolution in procedures and tools to prevent or detect tools occurrence. We distinguish what is known as intrusion detection systems (IDS). Much research has been done in this area and several techniques have been proposed. The most famous are statistics and data mining.

Data mining is an essential part of the process of knowledge discovery in databases. It is one of the most attractive and famous to scientists in many fields such as: economics, industry, etc. As it competes with other related disciplines such as algorithm, machine learning, mathematics, databases, etc. Data mining consists of extracting from a large amount of data a valid and understandable information [1]. In our field intrusion detection systems, data mining was used firstly by Wenke Lee and Salvatore J. Stolfo. Since then data mining still widely used by developers in this field. It turns out that the IDS based on data mining techniques is more adaptive and more effective [9].

Snort is a network-based IDS with use the so-called misuse rules for in on-line detection; but the problem that is these rules are defined manually by a network security expert, by its role: it processes and analyzes network traffic of a given attack and put its specific rules or signatures, the expert defines these rules relative to a pre-knowledge of the attack attributes discriminatory. The goal of our work consists in the creation of a system able to generate automatically snort rules relative to a given attack dataset. Our system will use frequent patterns mining techniques to extract single intrusion patterns and convert them to snort rules. This idea, in reality, may be an advantage because the computer can handle a huge data size than a human expert. This does not mean that we exclude the role of the expert. The expert role may enter to further refine the results of our system because it will not be more accurate or sometimes representative enough for each of two categories, attack or normal, because in our research we consider that frequency is the criterion of discrimination. There are examples of attack types that occur in a small number of packets and therefore the pattern of this attack will not appear as a result of our systems, which depends on frequent patterns.

Let us take the paper [10] as a start point for our work; it treats the same problem, but by using an algorithm called "apriori" which is primitive and has a bad performance in this area. In contrast, the current version of our system includes two class of algorithms for frequent patterns mining: total (FIN [11], FPgrowth [12], Apriori [13]) and maximal (FPmax [12]), these algorithms existing currently in our system they available in the library SPMF [14], already we have created our implementation, but for reasons to avoid errors, and the peace of mind on results, we used the existing implementations.

In a real experiment using a small network, we put our system together with Snort, to prove detection ability of Denial of service (DOS) attack by Snort using the rules produced by our system. Since most of the related work in intrusion detection systems, accuracy is used as a standard measure for evaluation in section 2.8, we followed a specific protocol to evaluate our system, each time we used an algorithm belonging to a particular class of frequent patterns mining algorithms. We found that the preferred option is FIN algorithm of the total algorithm class, with a minimum support value of approximately 0.45, but often there may be limits, so that the current version supports the TCP header and does not include payload part, and there is no maintenance of the extracted knowledge.

This work is organized around three chapters.

- Chapter 1: gives a short introduction to data mining as a step in knowledge discovery in databases, its tasks, association discovery, frequent patterns mining techniques which are explained in details.
- Chapter 2: describes Intrusion Detection System (IDS), architecture, the setup of an IDS with the network technology, their techniques, and types; also, we give a taxonomy of IDS for different criteria. Finally, we present state of the art in IDS domain.
- Chapter 3: contains a description of the proposed architecture of single intrusion pattern mining that was used to build our system for extension of snort, its usage, and evaluation with a discussion of the obtained results.

Chapter 1

Data Mining

1.1 Introduction

Data nowadays are on increase day after day; for this reason, researchers figure out some methods to get value from data, which can be helpful information.

Data mining (DM) is the main step in knowledge discovery in databases (KDD). Its performance is great, thanks to the intersection of research domains, like database, algorithms, artificial intelligence, statistics, etc. Data mining seeks to get patterns that include valuable informations in this chapter, we will use DM application, process and tasks, we also review the basic concept of pattern mining.

1.2 Definition

There were many definitions sets for data mining, we chose the most popular: "Data mining is a step in the KDD process that consists of applying data analysis and discovery algorithms that produce a particular enumeration of patterns (or models) over the data" [1].

Data mining uses some special algorithms on data to get patterns, for good perception that is makes it simple to analyze these data. Others steps in fig 1.1, such as how to prepare data, then selection, after that make it clean, etc. Those steps are needed before step of DM to make sure that we can apply our algorithms of DM for good results. we mention some essential tasks in section 1.5 that applied in DM, for more detail please reference to [1].

1.3 Domains of applications

Prediction of performance for any companies or domains is important, especially sensitive fields as health care or domain of insurance which analysis of “high risk” clients. DM as a leader for great performance, it makes big companies economic services as a bank or fraud detection uses in their technology search. Others applications of DM include: space science, health science, geography, and others [1].

1.4 Knowledge Discovery from Databases

In this section we, discover how to get helpful information from data. knowledge discovery from DB (KDD) includ many interactive and iterative steps. We have to pass through these steps shown in fig 1.1 [1].

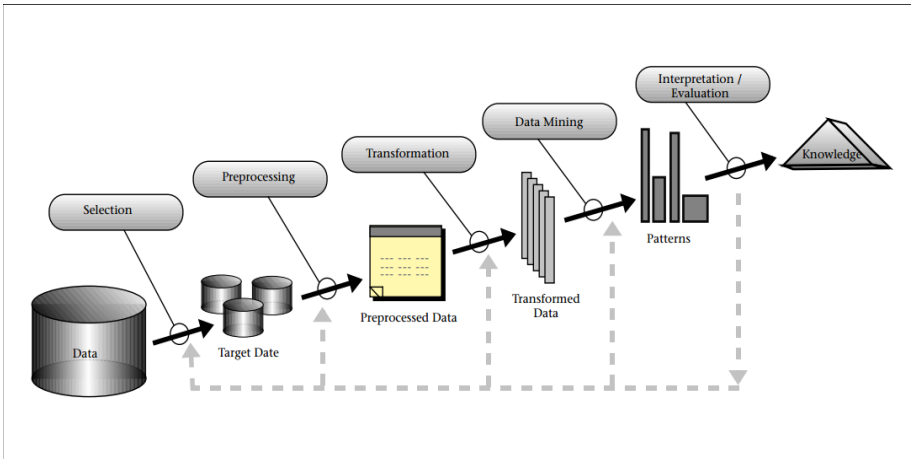


FIGURE 1.1: basic Steps of KDD Process [1].

- **Understanding of the application domain** building up a comprehension of the application field and previous relevant knowledge, as we can learn and know the objective of the KDD process from the client's perspective.
- **Creating a target data set** choosing a data collection, or concentrating on a subset of factors or information tests, on which revelation is to be performed.
- **Data cleaning and preprocessing** in this step, we have to prepare our data and dealing with any missing information fields, and gathering the fundamental data to the model. It is essential task before applying any algorithm.
- **Data reduction and projection** use helpful features for a good representation of our data, and that contingent upon the objective of the assignment. Reduce dimension, to make sure that we are ready for applying the tasks on the data.
- **Choosing the datamining algorithm(s)** Select specific methods of DM to looking for patterns in data, this step can incorporate choosing which models or parameters may be proper on our data because every algorithm has its own parameters and specific tactics of learning.
- **Applying datamining algorithm(s)**

  this is the most important step in KDD, which apply DM tasks on data that have been prepared by previous steps, for looking for patterns that we need, through regression, and clustering, classification, etc.
- **Evaluation** this step allows us to go back to any step in KDD if we have bad results; we can repeat algorithms of DM until we have good results. This progression can help our representation of the patterns and models that are given by DM tasks.
- **Using the discovered knowledge** finally, we can add this knowledge to our system for resolving problems or other activity, or just write it as a report for the practitioners.

1.5 Data mining tasks

There are many tasks used in DM. In general, essential objectives of DM are the prediction and description, both are useful for understanding data. The tasks of DM depend on the type of problems and some factors as dataset. We chose most tasks in DM [1].

1.5.1 Classification and regression

The main task of this feature it estimating a function that classifies input classes which defined early as training set, then apply function on test set, which has as results discrete output classes for classification, while regression as continuous(real numbers). For example, figure1.2, as demonstrates of partitioning the data of loan on tow category (class). It is difficult to have full separate of classes, but we can estimate output, in this case, the company can predict next time about her software if they can offer a loan or not

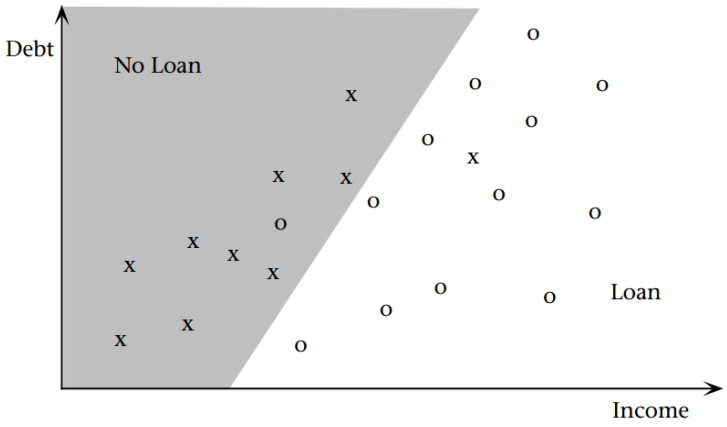


FIGURE 1.2: Linear Classification of Loan Data Set [1].

There are many techniques applying in classification. The principal ones are the following:

- **Decision Trees:** In this technique, we use tree structures for the representation form to indicate all possible decisions of classes. Many algorithms are mentioned in machine learning for decision tree as ID3 (Iterative Dichotomiser) which applying Entropy function and Information gain [1].

  $$\text{Entropy} : \sum -p_i \, log_2 \, p_i$$

  Where p_i is the probability of the class i.
  Information gain determines which attribute get more importance; it can be calculated by:
  Information Gain = Entropy (parent) – [weighted average entropy (childrens)]

- **Random Forest** RF is a supervised learning classifier based on a group of tree. Each tree produces a random selection and in training, the set is made from the examples of the classification tree. Every tree provide classification called

"votes" for that class. Forest chooses the overall classification trees in the forest, by the elementary value that a group of "weak learner"' can compose from a "strong learner" [15].

- **Support Vector Machines** (SVM) were originally developed by Vladimir vapnik, for classification problems. SVM can be applied for those problems when data cannot be nearly separated, they are based on maximum margin linear discriminants. The main task of SVM to find the optimal hyperplane that maximizes the gap or margin between the classes. as we can utilize the kernel trick for locating the ideal nonlinear decision boundary between classes, which corresponds to a hyperplane in some high-dimensional "nonlinear" space [16].

1.5.2 Clustering

The objective of clustering is grouping data into homogeneous groups called clusters are not defined before. So the main task of clustering is to maximize similarity between cluster [1]. The similarity is based on distance of (Euclidean, Manhattan, Minkowski), between instances, where many algorithms can be used such as using K-mean algorithm [17]. For example :

$$\text{Euclidean distance} : d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

where x,y are represented Euclidean vectors.

1.5.3 Description

Data scientists looking for the way to describe patterns when applying some task of DM on data. Descriptions of patterns might give a clear vision of hiding patterns on data or possible explanations [18].

1.5.4 Outlier detection

Known also as anomaly or deviation detection, can be defined as the inverse of grouping. The main task is to look for data which don't belong to any established model [19].

1.5.5 Association

Association in DM is looking for relations between items or objects. Which object follow another in the same transactions, with some metrics such as support and confidence. As a result, we get association rule for these objects. Because this task is important in the our research, therefore, we devoted a whole section 1.6 for it [18].

1.6 Association discovery

The association discovery or the association search is a well-known domain because of its effect, the interest and the application in certain varied field as telecommunication (detection of fraud), web (analysis of the behavior of the users of a site), and biology (DNA sequences), etc. [2].

In 1993, Agrawal and Srikant introdused the problem of extracting association rules between set of items in a collection of data. Through a paper titled *Fast Algorithms for Mining Association Rules* [13], for the purpose of discovering correlations or causal structures between data.

In this section, we will explain some concepts in the association discovery and how to extract the association rules based on a frequent itemsets.

1.6.1 Basic concepts

- **Itemset** I = $\{x_1, x_2, .., x_m\}$ let be a set of m elements called items. X is a set of items in which $X \subseteq I$ is called an itemset.
- **k-itemset** is an itemset of size equal to k.
- **Transaction database** A transaction database is a set of transactions in the form D = $\{t_1, ...t_n\}$, in which a transaction t_i is a set of items identified by a unique id i under the form $\langle i, X \rangle$, X is an itemset, where $i \in T$ and T = $\{i_1, i_2, .., i_n\}$ is set of transaction identifiers or tids.
- **Database representation** The transaction database representations can be in the form of a binary table of $n \times m$ dimension named binary database, n is the number of transactions and m is the number of items, an item attribute take a boolean value. Table 1.1 and the table 1.2 are an example of this type of database, with nine transactions and five items $\{I_1, I_2, I_3, I_4, I_5\}$

TID	Items
1	{I1,I2,I5}
2	{I2,I4}
3	{I2,I3}
4	{I1,I2,I4}
5	{I1,I3}
6	{I2,I3}
7	{I1,I3}
8	{I1,I2,I3,I5}
9	{I1,I2,I3}

TABLE 1.1: Transaction database [6]

N transaction	T1	T2	T3	T4	T5
1	1	1	0	0	1
2	0	1	0	1	0
3	0	1	1	0	0
4	1	1	0	1	0
5	1	1	0	1	0
6	0	1	1	0	0
7	1	0	1	0	0
8	1	1	1	0	1
9	1	1	1	0	0

TABLE 1.2: Binary database [6].

- **Patterns mining problem**

  The problem consists to find all frequent itemsets in a database D. An itemset X is frequent if sup$(x) \geq$ minsup (minsup: a threshold set by the user) [6]. Let be the database $D \subseteq T \times I$, over the tids T and items I, F is the set of all frequent itemsets, in which:

$$F = \{X \subseteq I \mid \text{sup(X)} \geq minsup\}$$

  A frequent itemset X that has no frequent supersets is called **maximal** [6]. M is the set of all maximal frequent itemsets, given with following formula:

$$M = \{X \mid X \in F \text{ and } \nexists Y \supset X, \text{ such that } Y \in F\}$$

- **Association rule**

  Can be seen as a logical application under expression $X \Rightarrow Y$; where X and Y are itemsets. The association rules express the associations, or correlations among the sets of elements in the transaction databases, it has some characteristics like **Support** and **Confidence** [13].

  Support: The support of an item-set X in a dataset D is the total number of transactions T in D that contain X; $sup(X) = card|T_x|$ where T_x is all transactions containing X.

  $$sup(X) = card\{t \in T | x \subseteq t\}$$

  Confidence of a rule $x \Rightarrow y$: Is the conditional probability that a transaction having X also contains Y, more precisely, it is the rate of transactions where the itemsets XY occur together, compared to the transactions where the items X occurs:

  $$conf(X \rightarrow Y) = \frac{card(t \in T | X \cup Y \subseteq t))}{card(t \in T | X \subseteq t)}$$

  This is equivalent to:

  $$conf(X \rightarrow Y) = \frac{sup(XY)}{sup(X)}$$

1.6.2 Association rules construction

To build the associations rules we must generate all frequent itemsets in our dataset. This task is difficult in terms of frequent itemsets extraction complexity, the search space is exponential with the number of the items in the database that can be very high [2].

- **Generating association rules from frequent itemsets**

  For each subset of a frequent itemsets we build the rules those satisfy both minimum support and minimum confidence [13]. We can generate association rules by the following steps:

  - For each frequent itemset l, s take each time a not null subset of l.
    for exemple: we consider the frequent itemset $X = \{I1, I2, I5\}$. The subsets taken by s are $\{I1, I2\}, \{I1, I5\}, \{I2, I5\}, \{I1\}, \{I2\}$, and$\{I5\}$.
  - Consider s is subset of l and $s \neq \phi$, the rule to be generated is $s \rightarrow (l - s)$ if $\frac{sup(l)}{sup(s)} \geq min\_conf$, where min_conf is the minimum confidence
    Let's take the exemple of $X = \{I1, I2, I5\}$
    1. $\{I1, I2\} \rightarrow I5$, confidence $= 2/4 = 50\%$
    2. $\{I1, I5\} \rightarrow I2$, confidence $= 2/2 = 100\%$
    3. $\{I2, I5\} \rightarrow I1$, confidence $= 2/2 = 100\%$
    4. $I1 \rightarrow \{I2, I5\}$, confidence $= 2/6 = 33\%$
    5. $I2 \rightarrow \{I1, I5\}$, confidence $= 2/7 = 29\%$
    6. $I5 \rightarrow \{I1, I2\}$, confidence $= 2/2 = 100\%$

    If we assume that the minimum confidence is 70%, then only the second, third, and last rules should be generated [2].

1.7 Frequent pattern mining techniques

Several solutions have been proposed to extract frequent patterns. In general, there is more than one main approach so far, each approach has its principle of data representation and its own method of extracting frequent patterns in the database. In the following, we'll see the three forms for finding frequent pattern, first the horizontal is the basic form of frequent pattern mining, the vertical and by projection.

1.7.1 Horizontal layout based

One of the most important algorithms in this category is the apriori algorithm that is a basic algorithm in this domain. It was introduced by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets [13].

- Apriori: generation candidate and test [13]
  - Apriori use an approach known as a level-wise search, in iterative way where (k+l)-itemsets is found by extending k-itemsets;
  - First, the set of frequent 1-itemsets is found by scanning the database to calculate the support for each item, and add each item into 1-itemsets that satisfy minimum support (the prune step). The resulting set is denoted Ll;
  - Next, Ll is used to find L2 by using the **join** and **prune** step respectively. An important property also used for the search space reduction is **Apriori property**: *All nonempty subsets of a frequent itemset must also be frequent.* In another expression, a superset of any infrequent itemset should not be generated or tested.
  - The exploration of L3 is based on the use of previous level L2, with the same way continue to discover other levels until no new frequent k- itemsets can be found.
  - The join step: To find L_k, we must construct the set of candidates C_k, which is generated by joining L_{k-l} itemsets with themselves [13].
  - The prune step: C_k is a set of candidates k-itemset, it contains frequent and infrequent itemsets; we must full scan the database and count for each candidate in C_k its support, remove from C_k each itemset that do not satisfy the minimum support. The set result is noted by L_k [13].
- Exemple:
  We use the transaction database presented in the table 1.1 to apply apriori algorithm as illustrate the figure 1.3.

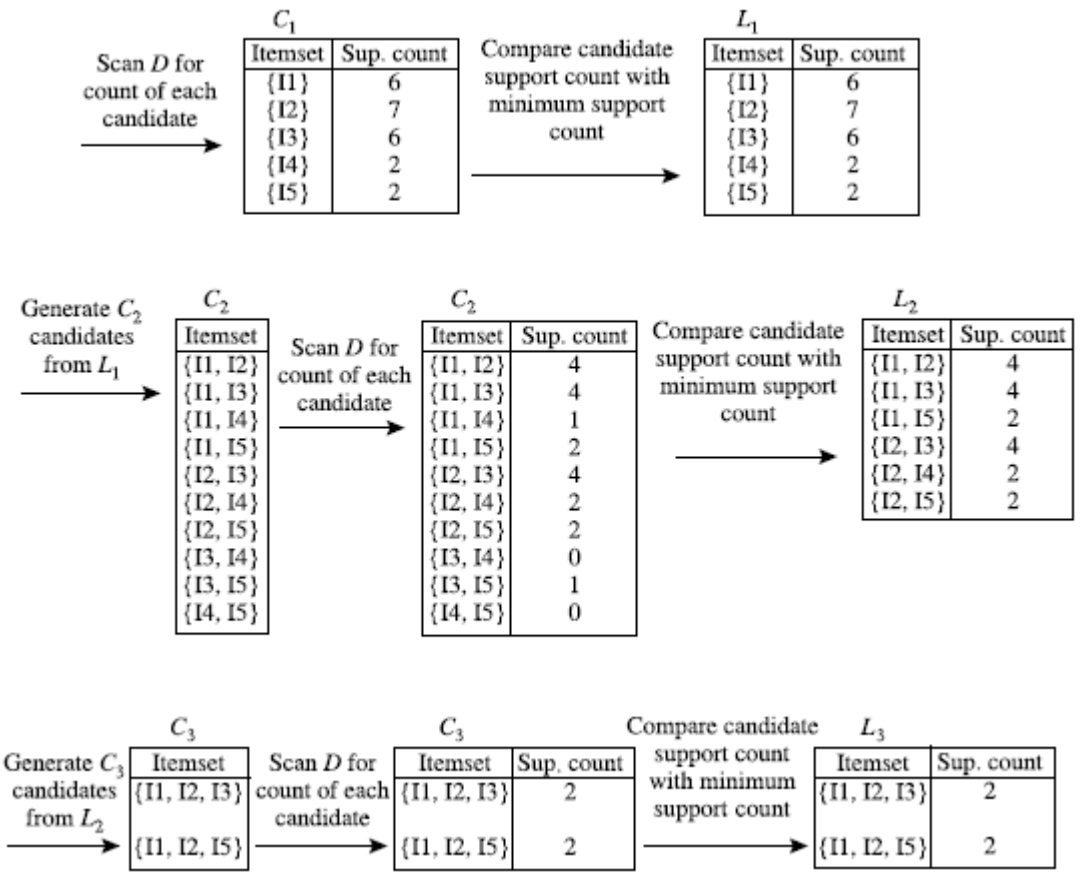


FIGURE 1.3: Generation of the candidate itemsets and frequent itemsets, assuming the minimum support count is 2 [2].

Algorithm 1 Apriori Algorithm [13].

C_k **:** the k-itemset candidates
L_K **:** the set k-itemset which satisfies minimum support,
each member of this set has two fields: the itemset and it support count.

Input **D :** database
Minsup : the minimum support

Output $\cup_K L_K$ the frequent itemsets of the database D

```
1   L1={1-frequent itemset};
2   for (k = 2; L_{k-1} ≠ φ; k++) do begin
3     C_K = candidate_gen(L_{k-1}) ;          /* New candidates */
4     forall transactions t ∈ D do begin
5       C_t = subset(C_k, t);                 /* Candidates contained in t */
6       forall candidates c ∈ C_t do
7         c.count++ ;
8     end
9     L_K = {c ∈ C_K | c.count ≥ minsup}
10  end
11  Answer = ∪_K L_K;

12  function candidate_gen(L_{k-1})
13  insert into C_k
14    select p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}
15    from L_{k-1} p, L_{k-1} q
16    where p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2},
          p.item_{k-1} < q.item_{k-1};
17  forall itemset c ∈ C_k do
18  forall (k-1)-subset s of c do
19    if(s ∉ L_{k-1}) then
        delete c from C_k;                    /* apriori property */
```

Discussion The complexity of the apriori algorithm in the worst cases is $O(|D|2^{|I|})$. If we chose a very small minimum support, in this case, the Apriori algorithm performs several data base scan until the longest frequent patterns in the database are determined, this behavior of the Apriori algorithm would result in a poor performance [6].

1.7.2 Vertical Layout based

The basic idea in this class of algorithms is the vertical representation of the data base, as an inverted list. For each item, we can have a transaction list where it appears.

An article was published in 1997 entitled *New Algorithms for Fast Discovery of Association Rules* by Zaki et al. presents the algorithm ECLAT (Equivalence Class Transformation). This algorithm uses a vertical database, there is no need to scan the database again and again. Eclat algorithm scans the database only once, it is

based on the logic of the Depth-First search and the equivalence class technique. The confidence is not calculated in this algorithm [6].

Example The table 1.3 shows an example of the database D in two representations.

TID	Items
1	{I1,I2,I5}
2	{I2,I4}
3	{I2,I3}
4	{I1,I2,I4}
5	{I1,I3}
6	{I2,I3}
7	{I1,I3}
8	{I1,I2,I3,I5}
9	{I1,I2,I3}

Tidsets

Vertical transformation of the database

Item	Tidlist	Tidlist_sup
I1	1,4,5,7,8,9	6
I2	1,2,3,4,6,8,9	7
I3	3,5,6,7,8,9	6
I4	2,4	2
I5	1,8	2

Tidlist

TABLE 1.3: Database D in two representations.

Items	**tidlist**
I1	1,4,5,7,8,9
I2	1,2,3,4,6,8,9
I3	3,5,6,7,8,9
I4	2,4
I5	1,8
frequent 1-itemset	

Items	**tidlist**
{I1,I2}	1,4,8,9
{I1,I3}	5,7,8,9
{I1,I4}	4
{I1,I5}	1,8
{I2,I3}	3,6,8,9
{I2, I4}	2,4
{I2, I5}	1,8
{I3, I5}	8
frequent 2-itemset	

Items	**tidlist**
{I1, I2, I3}	8,9
{I1, I2, I5}	1,8
frequent 3-itemset	

TABLE 1.4: Eclat algorithm with the minimum support count is 2 [6].

In table 1.4, an example of the application of eclat algorithm with minimum support equal to two.
$\sup(I1) = 6 \geq minSup$, then this pattern is common (it is retained in P)
$\sup(I1, I4) = 1 \leq minSup$, then this pattern is infrequent (it is removed from the equivalence class P).

Algorithm 2 Eclat Algorithm [6]

Input
D : the database
Minsup : the minimum support
P: Equivalence class contains only the frequent items

Output
F: the frequents itemsets of the database D

Call: $F \leftarrow \varnothing, P \leftarrow \{< i, t(i) > | i \in I, |t(i)| \geqslant minsup\}$
ECLAT (P, minsup, F):
1 **foreach** $< X_a, t(X_a) > \in$ P **do**
2 F $\leftarrow F \cup (X_a, \sup(X_a))$
3 $P_a \leftarrow \varnothing$
4 **foreach** $< X_b, t(X_b) > \in P, with \quad X_b > X_a \quad$ **do**
5 $X_{ab} = X_a \cup X_b$
6 $t(X_{ab}) = t(X_a) \cap t(X_b)$
7 **if** $\sup(X_{ab}) \geqslant minsup$ **then**
8 $P_a \leftarrow P_a \cup \{< X_{ab}, t(X_{ab}) >\}$
9 **if** $P_a \neq \varnothing \quad$ **then** ECLAT $(P_a, minsup, F)$

Discussion
The complexity of the algorithm ECLAT in the worst case is $O(|D|2^{|I|})$. when tid-list is large this makes the performance of the algorithm worse so that it takes more space and time to store candidate in the memory, also the time for intersection of Tid list will be longer [6].

1.7.3 Projected layout based

Frequent pattern growth or FP-growth was published in an article entitled *Mining Frequent Patterns without Candidate Generation* in order to solve the problem of extracting frequent patterns from a transaction base and overcome the disadvantages of its predecessors [20].

FP-growth is one of the algorithms which use the projection for mining frequent pattern. This algorithm adopts divide-and-conquer strategy and compresses the transactions database into a frequent pattern tree (FP-tree), then it divides the compressed tree to a set of conditional tree, which we can be used them for the project of frequents patterns, and for mining all frequent patterns by processing each conditional tree separately [2].

FP-tree construction

- At first, we are going to do a full scan on the transaction database to calculate the frequency of each item and sort them in descending order of their frequency and delete infrequent items. This resulting list or table is denoted by T.
- **Example** we reexamine the transaction database D of table 1.1 consider the minimum support count is 2.

Item	Tidlist_supt
I2	7
I1	6
I3	6
I4	2
I5	2

TABLE 1.5: The set of frequent items sorted in the order of descending support count.

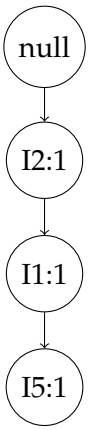
- Sort the items of each transaction in the order of T.

TID	Items	Items sorted
1	{I1, I2, I5}	{I2, I1, I5}
2	{I2, I4}	{I2, I4}
3	{I2, I3}	{I2, I3}
4	{I1, I2, I4}	{I2, I1, I4}
5	{I1, I3}	{I1, I3}
6	{I2, I3}	{I2, I3}
7	{I1, I3}	{I1, I3}
8	{I1, I2, I3, I5}	{I2, I1, I3, I5}
9	{I1, I2, I3}	{I2, I1, I3}

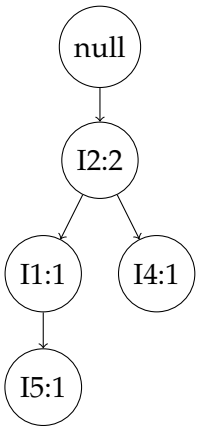
TABLE 1.6: The transactions itemset sorted in the order of T.

- Create the root of fp-tree labeled with "null."

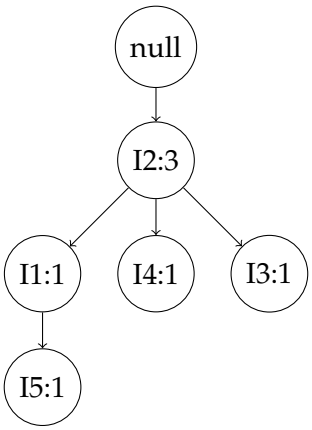
- For each transaction a branch is created in the tree, each item corresponds to a node in this latter. If a node already exists increment his count as we can see in the figure 1.4.



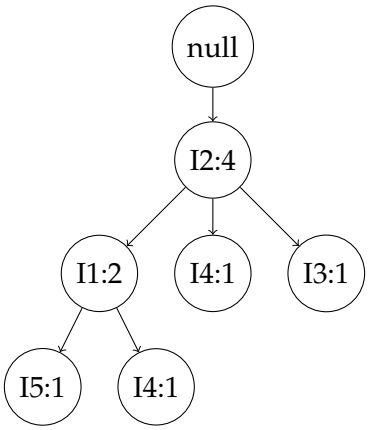
add transaction T1



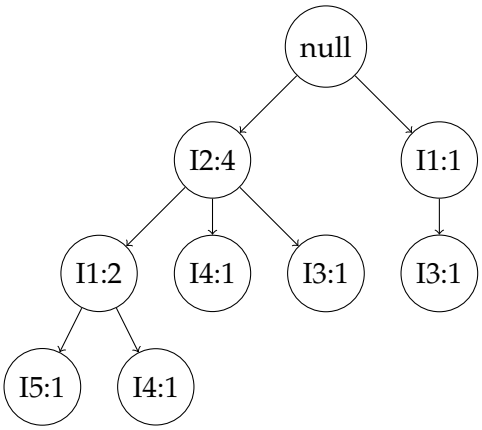
add transaction T2



add transaction T3



add transaction T4



add transaction T5

Repetitively, insert the other transactions T8 and T9.

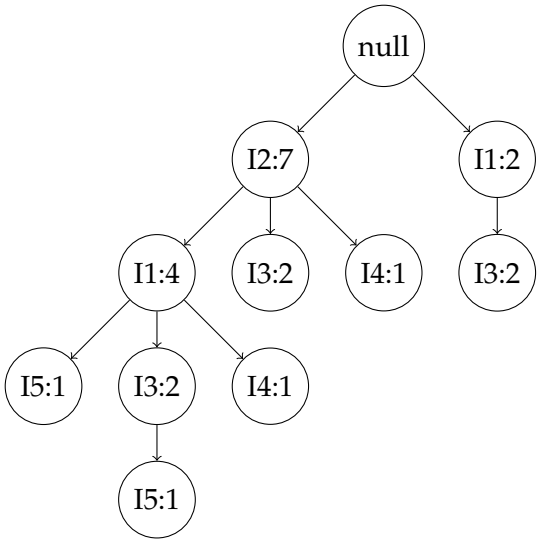


FIGURE 1.4: FP-Tree

Frequent patterns extraction

In the use of fp-tree 1.4 constructed in previous step will be able to generate all frequent patterns. For each suffix pattern (a frequent pattern with the length of equal to one) in fp-tree paths construct its conditional fp-tree, to perform this, browse paths of fp-tree and look for each path which appears suffix pattern with prefix path in it together, then, the conditional fp-tree will grow by each prefix path them, and will label related to this suffix pattern, for example the figure 1.5 illustrate the conditional fp-tree for suffix pattern I3 . By the concatenation of this pattern with each path in its conditional fp-tree, thus we go out all frequent patterns. The table 1.7 shows: the frequent patterns generated, suffix patterns and their conditional FP-trees.

Suffix pattern	Conditional FP-tree	Frequent patterns generated
I5	$\langle I2:2, I1:2\rangle$	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	$\langle I2:2\rangle$	{I2, I4: 2}
I3	$\langle I2:4, I1:2\rangle, \langle I1:2\rangle$	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	$\langle I2:4\rangle$	{I2, I1: 4}

TABLE 1.7: Generate the frequent patterns by creating the conditional FP-tree form FP-tree [2].

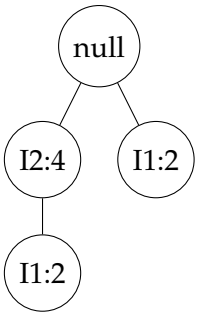


FIGURE 1.5: The conditional FP-tree associated with the conditional node I3 [2].

Algorithm 3 Algorithm FP-Growth [6]

Input **D :** the database
Minsup : the minimum support
P: Equivalence class contains only the frequent items

Output **F:** the frequents itemsets of the database D

Call:$R \leftarrow FP-tree(D), P \leftarrow \varnothing, F \leftarrow \varnothing$
FPGrowth (R, P, F, minsup):
1 Remove infrequent itemset from R
2 **if** ISPATH(R) **then**
3 **foreach** Y $\subseteq$ R **do**
4 X $\leftarrow P \cup Y$
5 sup$(X) \leftarrow min_{x \in Y}\{cnt(x)\}$
6 $F \leftarrow F \cup \{(X, \text{sup}(X))\}$
7 **else** /* FP-tree projected process for each frequent pattern i */
8 **foreach** i $\in R$ *in increasing order of* sup(i) **do**
9 $X \leftarrow P \cup \{i\}$
10 sup$(X) \leftarrow$ sup(Y) /* sum of cnt(i), for each node labeled i */
11 F$\leftarrow F \cup \{(X, \text{sup}(X))\}$
13 $R_X \leftarrow \varnothing$ /* projected FP-tree for X */
14 **foreach** path $\in PATHFROMROOT(i)$ **do**
15 cnt(i)$\leftarrow$ count of i in path
16 Insert path, excluding i, into FP-tree R_X with count cnt(i)
17 **if** $R_X \neq \varnothing$ **then** FPGrowth (R_X, X, F, minsup)

Discussion

The complexity of FP-Growth algorithm in the worst case is $O(|D|2^{|I|})$. The advantage of FP-Tree that it scans the database only twice and does not need a generation of candidates and it works poorly with a base having very long transactions (the depth of FP-Tree is very important) [6].

1.8 Conclusion

At the end of this chapter, we talked about the importance of data mining in the areas of informatics and scientific research, we discussed various concepts related to data mining: definition, tasks, and application domains. We also talked about KDD process and its various steps and we said that data mining is the most important step in the KDD process. Finally, we talked about the problem of extracting the association rules from frequent patterns and different approaches to search for these frequent patterns. In the next chapter, we will talk about the notion of intrusion detection system, architecture, techniques, tasks, taxonomy and about some related works.

Chapter 2

Intrusion Detection Systems

2.1 Introduction

There are valuable things must be seen or manipulated only by their owners and must protect them from destruction. Some houses have an alarm device in case of theft or intrusion, that's not encourages targeting them. The same measures can be applied in the field of information technology. As we know, the internet and networks in general are full by personal contents like passwords, business accounts, etc. So it is very important to protect them from violator of security and illegal use [21].

Users, data and transaction grow exponentially in our networks. Therefore, security is becoming more important and more challenging. Malicious users are looking for systems with vulnerability or poorly designed, or networks running insecure services. Alarms are needed to inform administrators and members of the security team to prevent this type of task. That is exactly what the intrusion detection systems (IDS) were designed for.

In this chapter, we start by definite the IDS, their task and explain the notion of intrusion detection as well as its two branches: anomaly detection and misuse detection. The general architecture of an IDS and the most popular taxonomies based on different criteria also are discussed. We show where and how we should place an IDS in a network, we also give brief description the of categories of network attack. We talk about some commercial or free IDS software. This chapter includes some representative works of the state-of-the art on IDS.

2.2 Definition of an IDS

An Intrusion Detection System is a software, hardware or combination of both used for the purpose of monitoring events of a system (network or a computer) and scanning it for signs of intruder activity, defined as attempts to steal or corrupt the security policy of the computer or network. These intruders activity are caused by attacks accessing the system from the external, authorized system users trying to gain additional privileges that they were not allowed to do. An IDS may have other functionalities depending upon how are complex and sophisticated its components [21].

So intrusion detection is a set of techniques based on traffic analysis that are used to reveal any intruder activity in computer system or network. Intrusion detection fall into two basic categories **misuse detection** and **anomaly detection** [10]: Misuse detection uses pattern (a rule or a signature) of well-known attacks to identify intrusions. But the major problem in this case is the inability to detect new attacks or attacks without known pattern early. Anomaly detection first establishes normal use models using some techniques like statistical measures or data mining on system features, and then detects any deviation from these established models.

When we speak about IDS tasks, we mention that an IDS can identify anomalies in network traffic as detecting who is trying to discover the network, detecting if the attack was successful or not, denial of service, even infection level of the computer system and network zones affected and make Alert centrally for all attacks.

Among the weaknesses of IDS, we have a problem of false positives, as can be complex and long configuration, also practical attacks are difficult to detect like SQL injection, many packets can be lost because of slowing scans, which make a real attack may go unnoticed [21].

2.3 Architecture of an IDS

The first prototype of intrusion detection was developed by Dorothy Denning [22]. Since that time, several schemes have been proposed to describe the components of an IDS in both research and business world. Among them, we chose one of the resulting works of Intrusion Detection exchange format Working Group (IDWG) of the Internet Engineering Task Force (IETF). The work goal of this group is the definition of a communication standard between certain components of an IDS [23]. No matter how different these systems in terms of the techniques they used it to collect and analyze data, most of them are based on a relatively general planner, which consists of components shown in fig 2.1 [3].

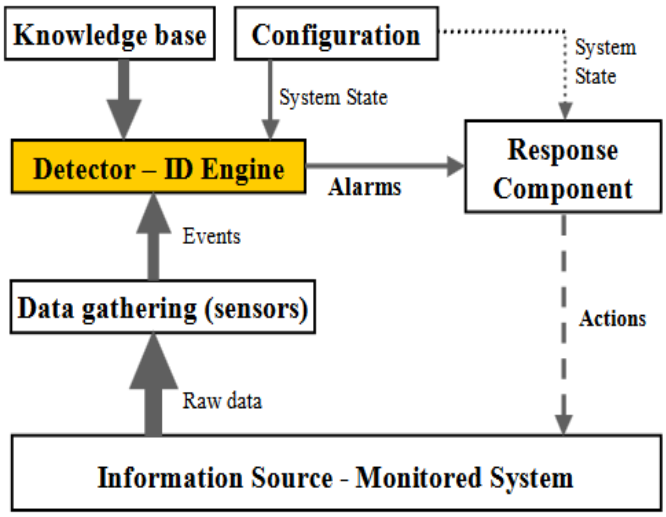


FIGURE 2.1: General architecture of an intrusion detection system [3].

In what follow we describe the different components of this architecture.

– **Data gathering device (sensor):** is responsible for collecting and filtering data from the monitored system and send it to the detection engine.

- **Intrusion Detection Engine:** its role is the matching to collected data from sensors with the corresponding in knowledge base to identify intrusive activities.
- **Knowledge base:** it usually contains set of rule or signature for different attacks or information about them. The knowledge base is usually provided by security experts or some techniques like statistical measures.
- **Configuration:** this device provides information about the current use state of the IDS in which we can applied security policy.
- **Response component:** passive or active measures (can either be automated or involve human interaction) taken in response to the detection of an attack, to stop it or to correct its effects.

2.4 IDS network setup

Where and how IDS should be connected to the network is very important in how effective it is. In this section will talk about how to connect an IDS in network and where should be placed it dependent to some strategies.

Network position of an IDS

The place of IDS depends on the network topology and the security policy, and also what the kinds of intrusion activities we want to detect (internal, external or both). We can have one or more places to intrusion detection systems. If we want to detect the intrusion coming from the outside of a local network may be the best place for an intrusion detection system just behind the router or a firewall. if we have multiple router connecting to the internet it is recommended to place one IDS box (machine execute an IDS program) at every entry point. However, if we want to detect internal possible threats as well, we may want to place an IDS box in every network paths. We note that more intrusion detection systems used to mean more work and more costs, in this case we can limit them only to sensitive network areas (like servers or databases) [4]. Figure 2.2 shows typical locations of an intrusion detection system in network technology.

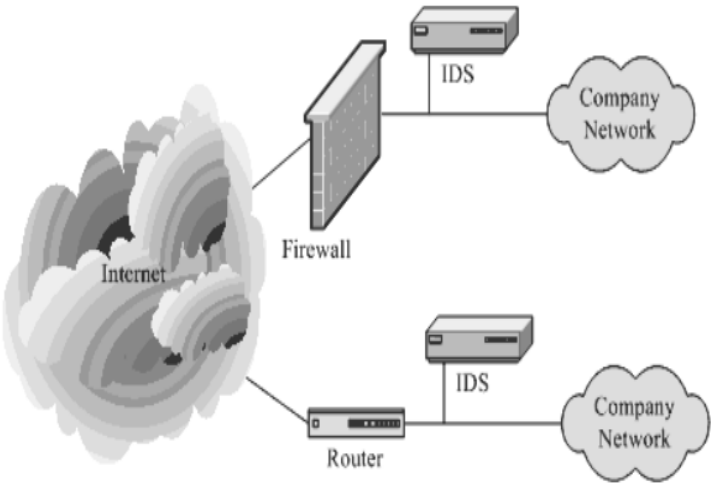


FIGURE 2.2: Typical locations for an intrusion detection system [4].

Dealing with switch

Some switches like CISCO can replicate all ports traffic into one port where we can attach the machine executing an IDS . These ports are usually called spanning ports. If you install IDS machines behind the firewall or router, this case guarantees that all the Internet traffic is visible for it [4]. As shown in figure 2.3.

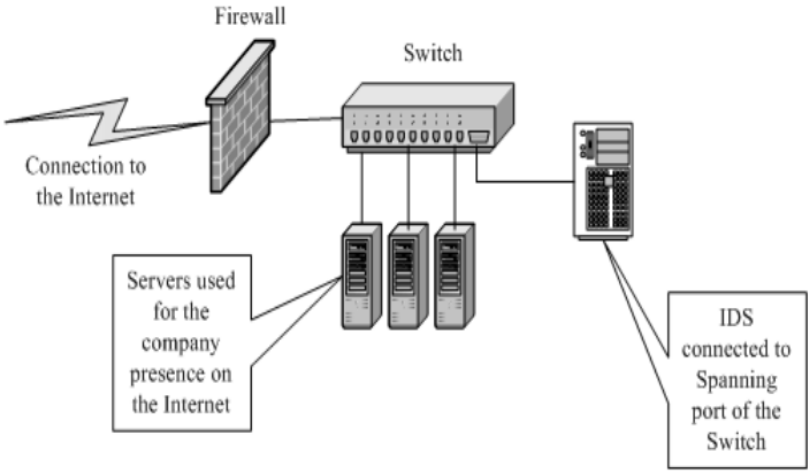


FIGURE 2.3: IDS connected in using switch [4].

Dealing with hub

A HUB allow to concentrate the transmissions for several devices on a same medium in a network. In this case, we can connect the IDS to a small HUB behind the firewall; more precisely, between firewall and the switch to guarantee that all incoming traffic is visible to the IDS [4]. Figure 2.4 illustrates this case.

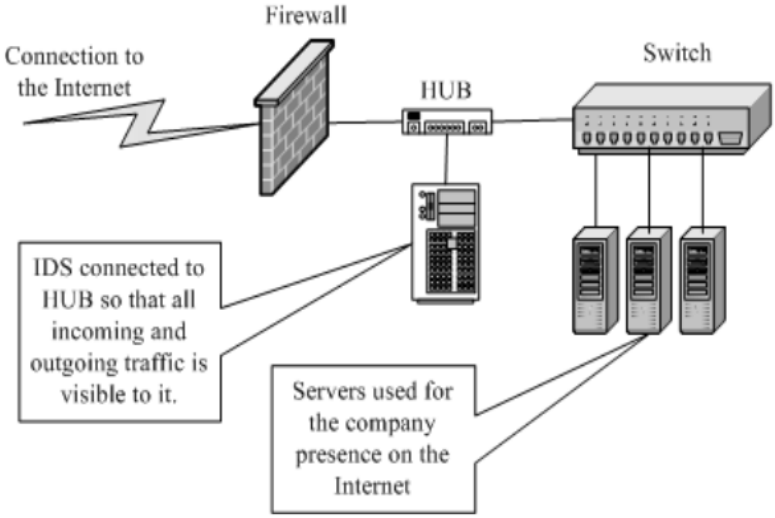


FIGURE 2.4: IDS connected in using hub [4].

2.5 Network intrusion types

In general, there are four traditional network intrusions and in reality even the web services concerned by them. They are in the following order according to the security risk degrees: User to Root Attack (U2R), Remote to User Attack (R2U), Denial of Service Attack (DoS), Probes Attack (Probes) [24].

User to root attack

In this case, attackers have legal accounts, but exploit some vulnerability in the victim system to gain more unauthorized privileges, e.g., sending buffers that are longer than the maximum buffer length or hiding some malicious executable code [24].

Remote to user attack

In this case, the attacker has no account in the victim system. He exploits some vulnerability for normal user access, after that can may be used U2R attack for more privileges in the system and cause more sever damage, like dictionary and Ftp-write attacks. The Dictionary attack is based on the ignorance of the user who does not choose the password carefully, where attackers use an assistant to generate common passwords and try in more time until they reach the target [24].

Denial of service attack

DoS attacks is a somewhat subversive activity. By making floods of requests to a service provider, until it becomes too busy and too full to handle legitimate requests. For example denial of service attack is SYN flood and Ping of Death attacks [24].

Probe attack

It consists of scanning the network by means of probes to find a weak point or a door in the software or hardware to infuse the attack in victim system. Ipsweep and Mscan are some tools used them to scan for look at these vulnerabilities [24].

2.6 Taxonomy of IDS

There are several classifications that exist for IDS according to different criteria. Known classifications have been proposed according to five criteria: the detection method used, the behavior on detection, detection paradigm, the frequency of use [5]. As depicted by the figure 2.5. Others have classified IDS relative to the monitored domain, as mentioned by [25] and [26].

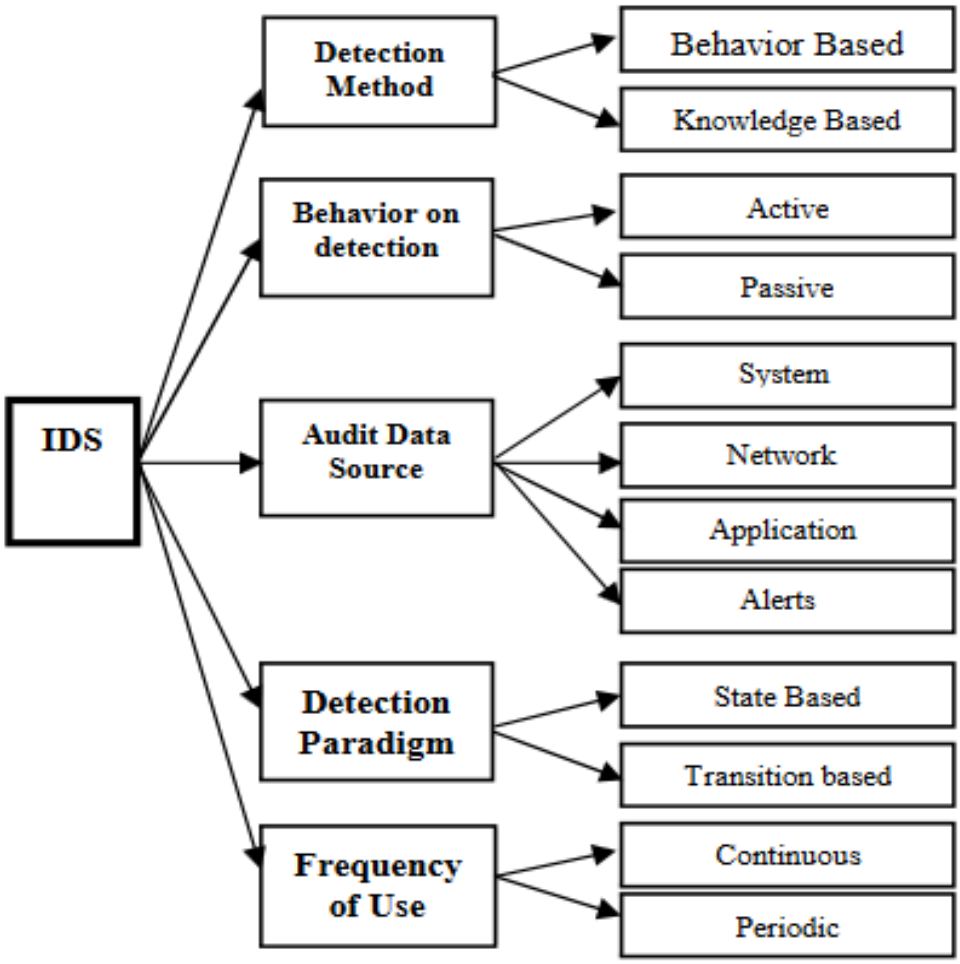


FIGURE 2.5: Taxonomy of Intrusion Detection Systems [5].

2.6.1 Detection method

Two detection approaches have been proposed in this category [21]:

- **Behavior based:** Also called anomaly-based, this approach is based on the assumption that we can define normal behavior models of the user and flag any deviation from them as suspicious occurrence and therefore a sign of a possible attack.
- **Knowledge based:** Also called misuse detection or signature-based, it relies on a model made up of forbidden sections in the computer system, this model is based on specific vulnerabilities and the attackers exploiting them. Any matching with these latter with the audit monitored data alarms are raised.

2.6.2 Behavior on detection

Another way of IDS classification by their reaction type once an attack has been detected. We can distinguish tow classes [27]:

- **Passive:** Once an attack has been detected, the system raises an alarm with the system administrator's notification, with no procedures applied to prevent, it is the role of the system administrator.

- **Active:** Active systems in addition to the notification of the system administrator, can automatically take measures to stop the current attack. For example, they can cut suspicious connections or even, disconnecting TCP connections; re-configuring routers, switches, and firewalls; stopping vulnerable services; modifying configuration files are some examples of active responses. Tools such as RealSecure or NetProwler offer this type of reaction. Most IDSs are passive, because no one wants run a risk of taking a wrong active response based on the wrong alarm, the danger is that the attackers use this mechanism to prevent the service or other innocent party [27].

2.6.3 Audit source of data

Audit source of data to be analyzed are an essential feature of an IDS. This data are the raw material of the detection process ; it may come either from logs generated by the operating system, from application logs, from the network, or from alerts generated by other IDS [27].

a. **System logs** Most modern operating systems generally offer several sources of information like the following:

   - System commands to have what is happening instantly .
   - provides accounting information on the use of resources shared by users (processor time, memory, disk space, network throughput, launched applications, ...).
   - Security audit to provide information about everything that users do (or have done) in an audit file.

b. **Network logs** Several hardware or software devices (snifer) can capture network traffic as Wireshark software. This type of information source is particularly suitable when it comes to searching for network oriented attacks or remote penetrations attempte.

c. **Application logs** Applications can also be a source of information for IDSs. There are two types of sensor application:

   - **Internal sensor:** the filtering on the application activities is executed by the code of the application.
   - **External sensor:** he filtering is done outside the application. To do this, we can filter the logs produced by the application or the execution of the application can be intercepted at the level of its calls of libraries or an application proxy.

d. **Alerts logs** An intrusion detector also can use another type of high-level data, as we previously mentioned. These high-level are alerts sent by scanners from an another IDS. They can be used by an IDS to trigger a finer analysis following a potential attack indication. In addition, by combining several alerts one can sometimes detect a complex intrusion of higher level, which one qualifies by meta-alert.

2.6.4 Detection paradigm

There are two known paradigm in IDS area. The first category of intrusion detector is **State based** systems detect that the monitored system becomes in failure state; and second **Transition based** systems recognize successive elements that cause the transition from normal to failure. It is easy to imagine that any attack would change the situation of the system from normal state into a failure state [27].

2.6.5 Frequency of use

Also we can classify IDSs in their frequency of use, we can distinguish two modes:
Periodic: The analysis is periodically done to audit files of search a possible sign of intrusion, this method can be sufficient in the less sensitive case.
Continuous: Most recent intrusion detection systems perform their ongoing audit analysis in a continuous manner to effect detection in nearly real-time detection. This is necessary in sensitive contexts (e.g. commerce). Remember that this trend will be costly in terms of computation time and the reply, because you have to analyze everything that is happening on the system now [27].

2.6.6 Monitored domain

As we mentioned early, we can classify IDS by their monitoring domain, this one can be at the level of an enterprise network, a host machine or an application. Hybrid IDSs combine between characteristics both of Network-based intrusion detection (NIDS) and Host-Based Intrusion Detection (HIDS) [25].

- **Host-Based Intrusion Detection** Host-based Intrusion Detection Systems or HIDs analyze exclusively the information concerning this host. Since they do not have to control the traffic of the network but "only" the activities of a host they are usually more accurate on the types of attacks.

  Next, the impact on the machine concerned is immediately sensitive, for example in the case of a successful attack by a user. These IDS use two types of sources for providing information on the activity of the machine: logs and audit traces of the operating system. Each has its advantages: the audit traces are more precise and detailed and provide better information while logs that only provide essential information. These can be better controlled and analyzed because of their size, but some attacks may go unnoticed, while they are detectable by an audit trace analysis [25].

  This type of IDS has a number of advantages: it is possible to see immediately the impact of an attack and that mean react better. that return to the quantity of information studied, it is possible to observe the activities taking place on the host with accuracy and optimize the system according to the observed activities. for example attack of "Trojan horse", is difficult to detect by NIDS while HIDS can. But also has weaknesses, by his qualities, the large of data generated. This type of IDS is very sensitive to DoS attacks, which can explode the size of log files. Another disadvantage is size of the alert report files to examine, which is very restrictive for the security administrator.

  HIDS are usually placed on sensitive machines that are susceptible to attacks and possessing sensitive data for the company. Servers, web and applications, can in particular be protected by a HIDS. Finally, here are some known HIDS:Tripwire, WATCH, DragonSquire, Tiger, Security Manager, Etc [21].

- **Application-Based Intrusion Detection** Application-based IDSs are a subset of host IDSs. They control the interaction between a user and a program by adding log files in order to provide more information about the activities of a particular application. Since they operate between a user and the monitored program, it is easy to filter any notable behavior [25].

  The advantage of this IDS is that it can detect and prevent particulars commands that the user could use with the program and monitor each transaction between the user and the application. Also, the data is decoded into a known context. On the other hand, since this IDS does not act at the kernel level, the security provided is more weak, especially with regard to "Trojan horse" attacks. The log files generated by this type of IDS are easy targets for attackers, not as the audit Traces of the system. This type of IDS is useful for monitoring the activity of a very sensitive application, but its use is generally executed in association with a HIDS. It will be necessary in this case to control the CPU utilization rate of the IDS so as not to compromise the performance of the machine.

- **Network-based intrusion detection** The essential role of a NIDS is the analysis and interpretation of the packets circulating on the network. The implementation of a NIDS is done in the following way: sensors are placed at strategic network locations and generate alerts if they detect attacks. These alerts are sent to a secure console, which analyzes and processes them eventually. This console is usually located on an isolated network, which connects only the sensors and the console [25]. The advantages of NIDS are: sensors can be well secured since they are content to observe the traffic and therefore allow discreet monitoring of the network, scans-type attacks are easily detected, and it is possible to filter the traffic. NIDS are widely used, but they have many weaknesses, like the probability of false negatives is high, and it is difficult to control all the network. They function in an encrypted way, which make a complied the analysis of packets. Finally, unlike, host-based IDSs, they don't see the impacts of attack. Some examples of NIDS are: Snort, NetRanger, Dragon, NFR, ISSRealSecure, Etc [21].

- **Hybrid IDS** Hybrid IDSs combine between characteristics of NIDS and HIDS. They allow in a single tool to monitor the network and terminals. The sensors are placed in points strategically, and act as NIDS or HIDS depending on their locations. We understand that hybrid IDS are based on a distributed architecture, where each component unifies its sending format communicate and extract more pertinent alerts.

  The advantages of hybrids IDS are less false positives, better correlation (the correlation allows to generate new alerts from old one), and the possibility of reaction on the analyzer [25].

2.7 Related Work

Intrusion detection is an active domain a lot of techniques have used by many works in this area. Based on our study, generally, we can distinguish two approaches. The most popular those based on statistical techniques and those based on data mining techniques and machine learning [9]. In this section, without claim of exhaustiveness, we will describe some representative works in this domain.

2.7.1 Statistical techniques for intrusion detection

Denning is one of the greatest authors in this field and his works are described in this book [28]. There are many different kinds of work done in the using statistics to detect intrusion, somewhat complicated are those techniques. Most of those works are available in paid books, so we mention only some of them. All that we mention in this part is taken from this paper [29]. In general, statistical techniques are used for anomaly detection.

Markov Process Model or Marker model

This type of intrusion detection examines the system in specific intervals and tracks its state. This model is widely used we list some works:

- **A Markov Chain Model Of Temporal Behavior For Anomaly Detection [30]:** The author Yong in this work use Markov chain to detect anomalies where this model is trained on the historical data as normal behavior of the system. A lower probability that the Markov chain is not compatible with observed behavior, explains that an intrusion has happened. The experiment was done on the audit data of a UNIX system.
- **A Hybrid High-Order Markov Chain Model For Computer Intrusion Detection [31]:** This research study is a hybrid model mostly based on the Markov chain, to profiling the Unix command sequence of the computer user in order to determine the "signature behavior" of that user. Next command depends on the recent history, can say the last three commands. The probability of transition from that order is calculated during the training phase. These probabilities are used to predict next command during the prediction phase. The results of this experiment showed that this model proved to be successful.

Statistical moments or mean and standard deviation model

The moment the statistics may be some measures such as mean, standard deviation or any other correlation. The analyst who uses this model knows these moments and any event that falls outside of moments area is said to be abnormal.

- **NIDES [32]:** To make a full profile according to several properties, the NIDES focuses on statistics such as frequencies, means, variances, and co-variances of the profile. If a profile exists with multiple metrics, NIDES distinguishes a dotted domain, to be considered as anomalous at any point it sufficiently far from the expected or specified value in the defined bitmap.
- **Host anomalies from network data [33]:** They have created a simple model based on statistics to detect intrusions on a network; but these approaches are applied to every host in a network so that the administrator can know which host has intruder activity.

Operational model or threshold metric

Depending on events passed on the interval of time, an alert shows up when there is less than "a" or more than "b" event happened. As an example, the Win2k take a user down if have "b" unsuccessful login, here less limit is 0 and more limit is "b". They proposed work as bellow for solving such problems in order to determine "a" and "b".

- **Application Of anomaly detection algorithms for detecting SYN Flooding attacks [34]:** Vasilios A. Siris and Fotini Papagalou tested out *Adaptive threshold algorithm* for detection of SYN Flood attacks. It is a simple algorithm based on examining the traffic size and the number of SYN packets that have been studied, by giving an interval which skips a specific threshold. In order to account for seasonal (daily and weekly) variations and trends, the value of the threshold is set adaptively makes by an estimate of the mean number of SYN packets, which is calculated from those traffic size. the Adaptive threshold algorithm demonstrates that it is good to high-intensity attacks.

Operational model or threshold metric, multivariate model and time series model are statistical approaches used as well, we have limited our description, all we mention in this section is available in this paper [29] with detail. In the use of statistical, we will face a more complex intrusion to detect if our information systems become complex and large, generally, the system intrusion detection based on the statistical method always rest complex. In the literature, intrusion detection systems based on data mining techniques and machine learning are more adaptive and more effective [9].

2.7.2 Data mining for intrusion detection

There are many tasks of data mining had been applied in IDS. Examples are projects as MADAM ID and ADAM [35], the following tasks that are most widely known in this area, for more details read [36].

- **Feature selection** Feature selection, also known as subset selection or attributes selection, is the choice of attributes used during data mining processes it has an impact on the quality of the obtained results. Data mining methods are more efficient if there is a prior knowledge of domain attributes, the priority of these attributes, the less important attributes and relationships or the correlation between the attributes. The attribute selection method my be based on "heuristic-try-error" principle. This principle gives approximate results, we make attempts to improve the results further. Also we can use certain techniques as principal component analysis (PCA) for features spaces reduction. In the literature, a good choice of attributes consists of intrinsic attributes, and calculated attribute, to determine the relevance of each choice we can use certain measures like information gain [9].

- **Classification technique for intrusion detection** The purpose of using classification on intrusion detection is trying to arrange all traffic as normal or abnormal; at the same time limit the number of false positives and false negatives. General techniques have been used on IDS; Examples are the following:

  - **Decision Tree :** Kumar and Jain were worked on ids based decision tree technique by constructing intrusions rules. These rules determine which network traffic behavior is normal or abnormal. They use ID3 algorithm to show that evaluation gives less false alarm and high accuracy rate [37].
  - **Random Forest :** Jiong Zhang and al, proposed a new systematic framework that apply a random forests, for misuse detection, by training data, which created automatically patterns of intrusions. For anomaly detection, they use outlier detection [38] mechanism of the random forests algorithm, which they detected a novel intrusions [39].

- **SVM :** are used as a supervised learning machine that analyses data and recognize patterns, in order to decrease the false alarm rate in IDS and increasing the true positive rate simultaneously and also minimizing number of features to enhance low learning and computation time. So researchers proposed Genetic Algorithm (GA) which has high potential of finding the best solution in a search space [40] and for optimization algorithm to maximize the performance of the SVM, they catch a detection accuracy rate equals to 80.14% [41], and use kernel principal component analysis (KPCA) as a feature selection technique to decrease the dimension of feature vectors and reducing the training time [42].

  Least squares support vector machine is a kind of modified support vector machine for classification, to compare the performance of some classification techniques and their proposed method in network intrusion detection. It is shown that LS-SVM detection method has higher detection accuracy than support vector machine and neural network [43].
- **ANN :** The main thing of Artificial Neural Network (ANN) in IDS it is learning and development, which makes them more accurate and efficient in facing the increasing number of unpredictable attacks [44]. Recently, an improvement alternative of ANN is proposed called Multi-Layer Perception (MLP) ANN [45]. The MLP made IDS more valid and demonstrates that detection result much better than traditional methods.
- **KNN :** In order to solve the problem of high data dimension in network intrusion detection, KNN classifier and two kinds of effective feature selection algorithms (Auto encoder) a branch of Artificial Neural Network, and Principal Component Analysis (PCA). Experimental results show that the combination of (KNN-Autoencoder) makes the accuracy of ID reach 93%, and the combination of (KNN-PCA) makes the accuracy of ID reach 91% [46].

- **Clustering** In this area the most used clustering technique is the k-means, but the difficulty is to choice number k of cluster. Using the k-means algorithm requires priori knowledge of the cluster number. In [47] the authors have developed a method called ADMIT is an IDS based on dynamic clustering where the number of clusters is determined during clustering. The principle: if we can not put an element in a cluster (because of the constraint similarity), we create a new cluster [9].

- **Association search** Lih-Chyau Wuu and al. in a paper entitled: *Building intrusion pattern miner for Snort network intrusion detection system* [10], they apply association rules in the network based intrusion detection systems. This work, is divided into tow parts both using the data mining tasks the association and sequential mining. In the first part, they use association search task to automatically generate single detection rules used by snort in on-line detecting mode snort doesn't have this ability before, an experts must first analyze and categorize attacking packets and hand-coding the corresponding snort rules with a specific syntax. By using those rules, snort can judges if an incoming packet is an attack package or not. They see an association rule as a frequent pattern or frequent episode, after generating set of association rules by applying the famous algorithm "apriori". Then, they convert them to the corresponding snort rules, each rule is the union of the intrusion patterns of packet header (like source port and destination port, flags, etc.) and the intrusion patterns of

packet payload (the data part). The second part in this work implements an intrusion behavior detection engine which create an alert when a series of incoming packets match the signatures representing sequential intrusion scenarios. This engine uses sequential rules when detecting the behavior. At the beginning, they must explore all sequential intrusion rules by applying sequence search algorithm, then it constructs a finite automation for each rule to automatically compare them with the sequence of incoming packets. The automation is passive initially. It becomes active when some incoming packet matches the rule of its first node, and it will making an alert if other packets matched the rules of the second node, the third node, etc. and the final node during a specific active time of the current sequential rule, the state machine becomes passive again after the end of time duration. In the training phase. They did not follow the same way as other developers in this domain which use both the dataset DARPA or KDD-Cup 99 instead, they collect a set of packets from attacking hosts execute the same attacking program (like Netbus Backdoor and Winnuk DOS, Land Dos, etc.) and there is no normal traffic seen in this moment just to obtain pure attacking packets to training their miner. They reach good results, but their system supports TCP protocol only.

As long as both the frequent patterns mining and the association search they are really close, the Ministry of Information and Communication of Indonesia have rising threat (malware) coming from internet, they were looking how it is growing and how are these characteristics, they apply frequent itemset mining (both Apriori and FP-Max) on data collected from intrusion detection systems sensor of their ministry. From 620 rule that have discovered on 2013 they found that 90% of attack occurred by only eight rules of IDS [48].

- **Hybrid techniques** In the literature the more performed data mining techniques in order to detect intrusions is by using hybrid techniques. This means use of mixed techniques together as classification and clustering. Hybrid model usually use one of clustering technique as the first component for "pre-classification" and one classification technique as the second component for the final classification. In particular, the first clustering technique performed as the data reduction (or outlet detection) task. The representative data without the noisy data used in the training are good to improve the classification result [7].

  An article published in 2010 entitled: *"A triangle area based nearest neighbors approach to intrusion detection"* by CF Tsai and CY Lin [7] presents a hybrid learning model for intrusion detection using the k-clustering firstly to identifier center class attack and the k-NN classifier to classify with similar attacks.

  Knowledge extracting and real time analysis in network intrusion detection systems are difficult and more expensive in terms of time because of the huge data and the number of big features number. In [49], author describes the advantages of Evolutionary Algorithms (EA) for feature selection combined with Particle Swarm Optimizations (PSO). This experience showed that they have very good performance in reducing the number of features significantly and sometimes improving the classification accuracy.

The table 2.1 compares the recent related works using hybrid techniques in order to detect intrusions in terms of their detection techniques developed, datasets used and evaluation.

Work	Technique	Dataset	Problem domain	Evaluation
Abadeh and al.	GA+FL	DARPA 1998	Anomaly detection	DR, FA
Chen and al.	GA+ANN	DARPA 1998	Anomaly detection	FP, FN
Kayacik and al.	SOM	KDD-Cup 99	Anomaly detection	FP, DR
Khan and al.	SOM+SVM	DARPA 1998	Anomalydetection	FP, FN
Li and Guo	TCM k-NN	KDD-Cup 99	Anomaly detection	TP, FP
Liu and al.	SOM+ANN	DARPA 1998	Anomaly and misuse detection	DR,FA,FP
Ozyer and al.	Genetic fuzzy classifier	KDD-Cup 99	Anomaly and misuse detection	DR
Peddabachigari and al.	DT+SVM	KDD-Cup 99	Anomaly and misuse detection	Accuracy
Shon and Moon	GA+SVM	DARPA 1999	Anomaly detection	DR, FP, FN
Shon and al.	GA+ANN/k-NN/SVM	DARPA 1998	Anomaly detection	DR, FP, FN
Wang and al.	Bayesian latent class	KDD-Cup 99	Anomaly detection	DR, FP
Chen and al.	SVM, ANN	DARPA 1998	Anomaly detection	DR, FP
Mukkamala and al.	Ensemble of SVM/ANN	KDD-Cup 99	Anomaly detection	Accuracy
Zhang and Shen	Robust SVM,one-class SVM	DARPA 1998	Anomaly detection	DR, FA
Zhang and al.	C-means clustering+ANN	KDD-Cup 99	Anomaly and misuse detection	DR, FP ANN
Liu and al.	Nearest neighbor clustering+GA	KDD-Cup 99	Anomaly detection	DR, FP
Peddabachigari and al.	SVM, DT	KDD-Cup 99	Anomaly detection	Accuracy

TABLE 2.1: Comparisons of IDS related works in this area [7].

GA: genetic algorithm.
FL: fuzzy logic.
DR: detection rate.
FA: false alarm.
ANN: artificial neural networks.
FP: false positive.
FN: false negative.
SVM: support vector machines.
SOM: self-organizing maps.
TP: true positive.
DT: decision trees.
LR: logistic regression.

2.8 Evaluation of IDS

Most searchers use accuracy for evaluation of IDS's 2.1, Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of detection our model got right. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{number Total of predictions}}$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Where those terms are defined below 2.8 in this confusion matrix 2.2

	Predicted YES	Predicted NO
Actual YES	True Positive	False Negatives
Actual NO	false Positive	True Negatives

TABLE 2.2: Confusion matrix for binary classification [8].

Definitions of terms

– True positive (TP): is the number of positive examples correctly classified.

- True negative (TN): is the number of negative examples correctly classified.
- False positive (FP): is the number of negative examples incorrectly classified.
- False negative (FN): is the number of positive examples incorrectly classified.

2.9 Tools

There are several free and commercial IDS available, we present below some notables tools:

1. **ISS RealSecure:** Internet Security Systems (ISS) provides ISS RealSecure IDS, a platform of integrated intrusion detection. ISS RealSecure IDS uses an approach based on norms for comparing network traffic entries and probable methods of attackers. ISS RealSecure IDS integrates with many applications of network and system management applications [21]. RealSecure combines in one agent has three essential features:
   - An intrusion detection engine.
   - A personal firewall.
   - An application and communication control module.
2. **Enterasys DRAGON:** Released by Enterasys Networks, it is an intrusion detection system considered as market leaders because of its performance, its faculties adaptation to any type of environment and its ability to analyze. Dragon detects intrusions on some of it infrastructure that they producing and allows to have global visibility on the information system. This makes it possible to optimize human resources necessary to analyze logs from different firewalls or Web servers in federating all these logs at a single Dragon console that will analyze automatically the related data [21].
3. **SNORT:** is a particularly responsive IDS because provided in open source. It free and easy to obtain. It has the advantage of a very large database of signatures made by a community of others users. It has the guarantee of getting updates from the base as soon as a new threat is reported. It was originally designed for the linux system but it was also ported to Windows users. There are several books dedicated to the installation and use of SNORT. SNORT is usually used in conjunction with other open source software named BASE which is the management and analysis console. This constitutes as negative point of snort; but we can consider that SNORT is less powerful in terms of analysis engine like commercial solutions such IDS of ISS [21].

2.10 Conclusion

In the past. Intrusion detection systems have seen significant evolution from different technologies they become able to conduct real-time analysis of data traffic in monitored systems. It can handle complex and fast traffic and store information about suspicious ones and make alarms. In addition to providing a detailed vision that was not previously available for what is currently happening in the system. It seems that its future will be very promising, because it can not be dispensed in any comprehensive model of enterprise security [3].

We provided an overview of computer attacks taxonomy, concept of intrusion detection. We lookd around detection systems and highlights about some of its properties, component, architectures, and classification. Although the development is ongoing. At the commercial and research level, research is still ongoing in this field. There are still a number of research issues related to performance of prediction, efficiency and error tolerance.

We have also discussed some recent related works. We have seen some approaches to detect intrusion like those based on statistical or data mining. In the use of statistics we have seen that we must have prior knowledge of certain characteristics, and detection of intrusion it is usually rest complex whenever the information system is complicated. Also we have seen that the choice of attributes affects in the quality of the results and most authors use a combination of intrinsic attributes and calculated attributes, and we talked about the works which use data mining techniques like classification and clustering. Finally, we saw that the combining of both the clustering and classification technique has been more effective.

In the next chapter, we represent the practical side of our work the extension of Snort. We will talk about Snort: installation, configuration, modes, usage. And will put the proposed architecture of our extension.

Chapter 3

Snort extension with frequent patterns

3.1 Introduction

Given the importance of information security, several tools has been developed. Snort is one of the known and used intrusion detection systems for network security; it is currently available for Linux and Windows. Snort uses the so-called rules base to operate; but the problem is these rules should be made by an expert, who analyzes the traffic and looks for the features of a specific attack and define its rules. This task may be difficult with huge traffic.

The main focus of our work is creating a system capable of automatically extracting these rules based on network traffic attack. In this chapter, we will talk about the proposed architecture of our system and explain its components, also we will see how to use our system for knowledge extraction and how to use this latter with Snort in a real case. We will evaluate the results of our system in a particular protocol and discuss the results.

3.2 The Snort Tool

Snort can be seen as an open-source project; it was created in 1998 by Martin Roesch. Today, the modern commercial IDS cost thousands of dollars at minimum, than, Snort can be used as alternative because it is free. However, it is can be used by both commercially and privately. In the begin of Snort was created for linux and now is available also for Windows, until now, has been downloaded more than 3 million times from its official site [50].

Snort nowadays is developed by Sourcefire (Martin Roesch's own company) which in turn recently was bought by security giant Checkpoint.

Snort is a tool based on libpcap, it can be used with different modes as a packet sniffer and logger, as network intrusion detection system (NIDS). it is based on rule features to detect a variety of attacks and probe and can perform on real, with alerts being sent to log file, it uses a simple language to put commands and define new detection rules [51].

In the following sections, we will see how to install Snort NIDS and how to use it in different modes, and also will see how to work with Snort rules and their different components.

3.2.1 Installation

In this section, we will show how to install and configure Snort for both the operating systems Linux and Windows.

On linux

In this section we will learn how to install Snort in linux OS (ubuntu 18.10). We follow these steps:

- **Download and install** Before actually installing snort, their are some of its per-requisites, you can run following commands to install all the required depending:
  - sudo apt-get update
  - sudo apt-get install build-essential
  - sudo apt-get install -y libpcap-dev libpcre3-dev 3libdumbnet-dev
  - sudo apt-get install -y zlib1g-dev liblzma-dev openssl libssl-dev
  - sudo apt-get bison flex

  install daq (snort require daq to run), use commands lines:
  - wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz
  - tar -xvzf daq-2.0.6.tar.gz
  - cd daq-2.0.6
  - ./configure
  - make
  - sudo make install

  install snort with commands lines :
  - wget https://www.snort.org/downloads/snort/snort-2.9.12.tar.gz
  - tar -xvzf snort-2.9.12.tar.gz
  - cd snort-2.9.12
  - ./configure
  - make
  - sudo make install
- **Configuration of snort files**
  1. First, Snort needs some folders and files to place its logs, and rules files.
     - sudo mkdir /etc/snort
     - sudo mkdir /etc/snort/rules
     - sudo mkdir /etc/snort/preproc_rules
     - sudo mkdir /usr/local/lib/snort_dynamicrules
     - sudo mkdir /etc/snort/so_rules
     - sudo mkdir /var/log/snort
  2. Creat empty files needed by snort :
     - sudo touch /etc/snort/rules/black_list.rules

- sudo touch /etc/snort/rules/white_list.rules
- sudo touch /etc/snort/rules/local.rules
- sudo touch /etc/snort/rules/sid-msg.map

3. Copy files from snort to new path :
   - cd /snort/snort-2.9.12/etc/
   - sudo cp *.conf* /etc/snort
   - sudo cp *.map /etc/snort
   - sudo cp *.dtd /etc/snort
   - cd /snort/snort-2.9.12/src/dynamic-preprocessors/
   - sudo cp * /usr/local/lib/snort_dynamicpreprocessor/
4. We want to set permissions to (Owner,Group,Others)=>(7,7,5), recursively (-R):
   - sudo chmod -R 775 /etc/snort
   - sudo chmod -R 775 /var/log/snort
   - sudo chmod -R 775 /etc/snort/so_rules
   - sudo chmod -R 775 /usr/local/lib/snort_dynamicrules
5. chown = CHange OWNer (Making paths belong to the user snort and the group snort, recursively (-R)):
   - sudo chown -R snort:snort /etc/snort
   - sudo chown -R snort:snort /var/log/snort
   - sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules
6. Next, open the snort.conf file in /etc/snort/.
   - ipvar HOME_NET any

   make change to "any" in this line to your internet ip address with "/24" for example : 192.168.1.0/24.
7. at line 104, make sure your paths look like this.
   - var RULE_PATH /etc/snort/rules
   - var SO_RULE_PATH /etc/snort/so_rules
   - var PREPROC_RULE_PATH /etc/snort/preproc_rules
   - var WHITE_LIST_PATH /etc/snort/rules
   - var BLACK_LIST_PATH /etc/snort/rules
8. UN-comment the 545th line and make it look like this
   - include $RULE_PATH/.rules

- **Test configuration**
  - As a final step we need to make sure that snort is running in our system. Test the configuration using the parameter -T to enable test mode.
    * sudo snort -T -c /etc/snort/snort.conf

After running the Snort configuration test, we get a message like this example below 3.1.

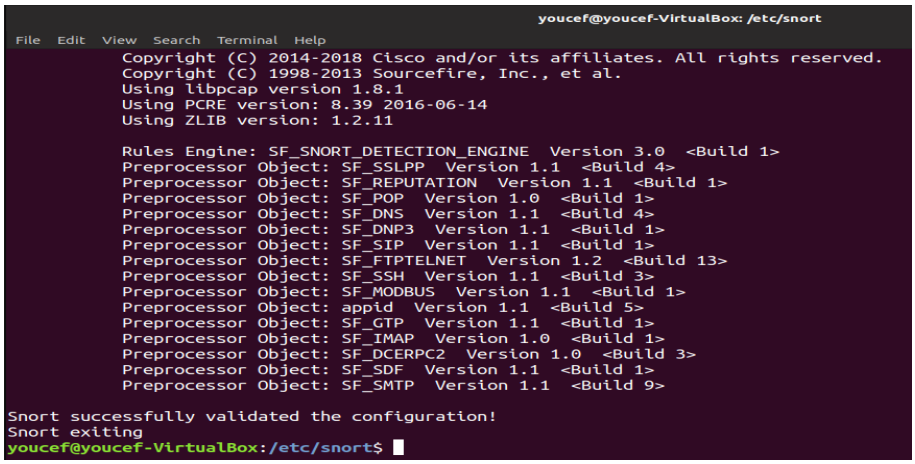


FIGURE 3.1: Test configuration.

On windows

- **Download and install** The installation on Windows operating system is easy, just download the program and open it, then, select the destination folder and click next until the install is complete. Snort windows version is available form: https://www.snort.org.

- **Configuration of snort file** Open the file $\backslash Snort \backslash etc \backslash snort.conf$ with a text editor and make the following steps:

  1. Set up of *HOME_NET* addresses, in this case we put the addresses to be protected: *var HOME_NET any* will be var *HOME_NET* 192.168.1.0/24, note that all addresses in Snort are written using the CIDR notation,
  2. Set up of EXTERNAL_NET addresses, EXTERNAL_NET mean the external network addresses:
     var EXTERNAL_NET any will be *var EXTERNAL_NET* !$HOME_NET.
  3. Set the path of dynamic preprocessor libraries:
     dynamicpreprocessor directory: $\backslash Snort \backslash lib \backslash snort\_dynamicpreprocessor$
  4. Set the path of base preprocessor engine:
     dynamicengine: $\backslash Snort \backslash lib \backslash snort\_dynamicengine \backslash sf\_engine.dll$
  5. make these lines as comments by add "#" in the beginning of each line:
     # dynamicdetection directory /usr/local/lib/snort_dynamicrules
     And:
     # Does nothing in IDS mode
     # preprocessor normalize_ip4
     # preprocessor normalize_tcp: ...
     # preprocessor normalize_icmp4
     # preprocessor normalize_ip6
     # preprocessor normalize_icmp6
  6. The creation of white.list, black.list files and place their paths:
     var WHITE_LIST_PATH $\backslash snort \backslash rules$
     var BLACK_LIST_PATH $\backslash snort \backslash rules$

7. Finally, download the set of rules from https://www.snort.org, and put them in their own folder *snort**rules*.

– **Test of the configuration** To make sure that the configuration of Snort file previously set are correct, type in the command interpreter:
snort -i 4 -c \Snort\etc\snort.conf -T , as show the following figure 3.2.

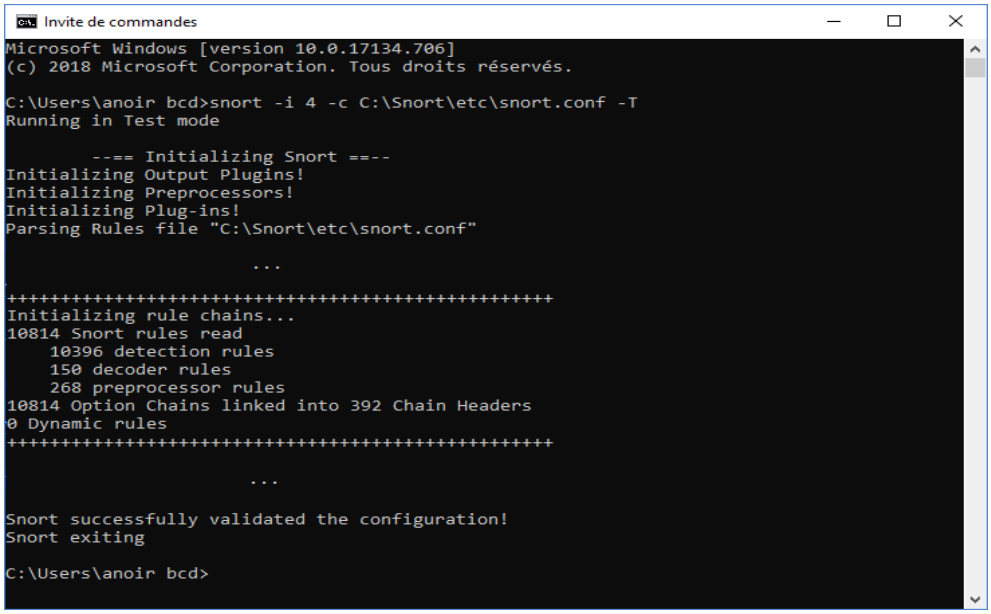


FIGURE 3.2: Test configuration file of Snort.

3.2.2 Modes

We can distinguish two essential snort running modes : network intrusion detection mode and packet sniffer mode. In sniffer mode snort work like tcpdump program, also we can log these packets into log file on tcpdump format and view it later by snort itself or other programs. There are better programs having this function like Wireshark and tcpdump, tcpdump is a program it comes with all Linux distributions. In the other mode snort can detect any intruder network traffic using its rules [4].

3.2.2.1 Network Sniffer

In network sniffer mode, snort reads the packets circulating on the network and displays them continuously with different levels on the screen. The command: snort -dev -i 4,tells snort to sniffing for IP of third layer, TCP/UDP/ICMP of fourth layer and packet payload in seventh layer on the specific network interface number 4, as shown in the figure 3.3. To show the available network interfaces type: snort –W.

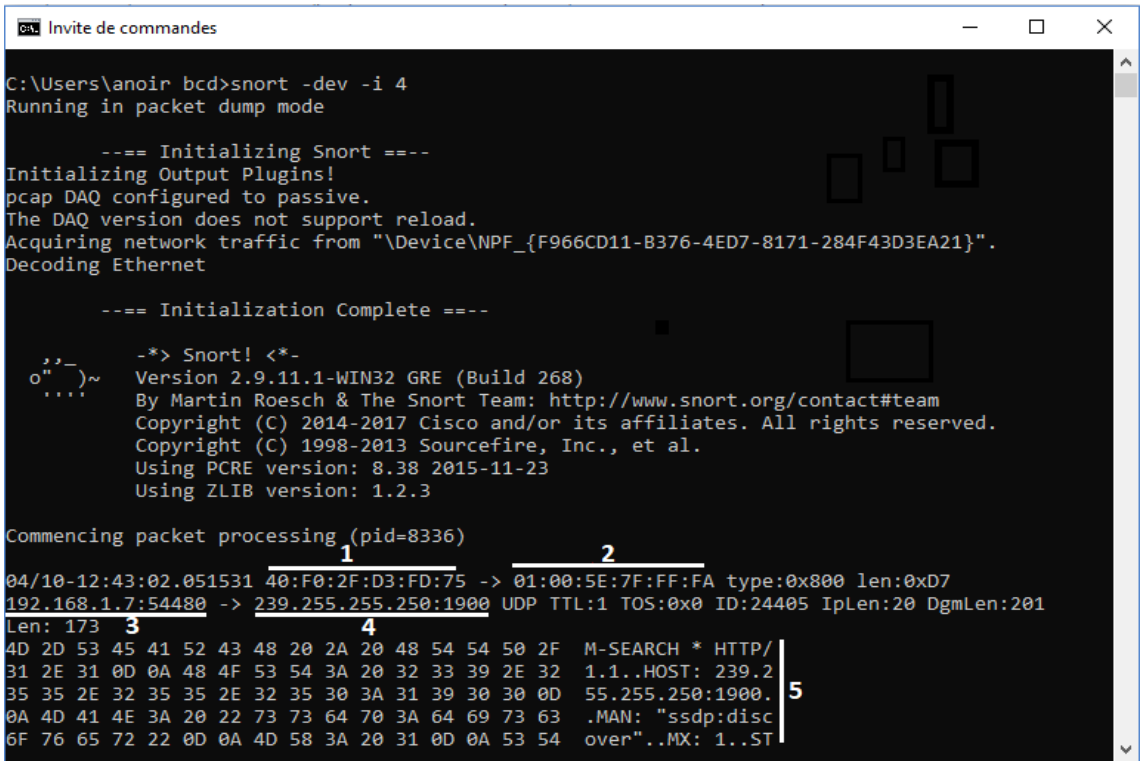


FIGURE 3.3: Snort on network sniffer mode.

Description of the elements in the figure

1 Source MAC address
2 Destination MAC address
3 Source IP and port
4 Destination IP and port
5 Packet payload (or data)

3.2.2.2 Network Intrusion Detection

In this mode, SNORT analyzes network traffic, compares that traffic with rules already defined, and establishes actions to execute. In this case, we need to provide configuration file snort.config, this file typically contains a references to rules file.

Example:
consider the following:

- alert tcp any any -> any any (msg:"test TCP"; sid:1000003;), in the use of this rule snort will make an alert for each tcp packet coming. We put it in the Snort rules file with the extension "local.rule".
- snort -i 4 -c $C : \backslash Snort \backslash etc \backslash snort.conf$ -A console, this command tell snort to run in NIDS mode sniffing on network interface number 4 and sending alerts to console, we can show the list of available interfaces by using the command: snort -W.

When we use command and rule precedents, the result will be as shown in figure 3.4.

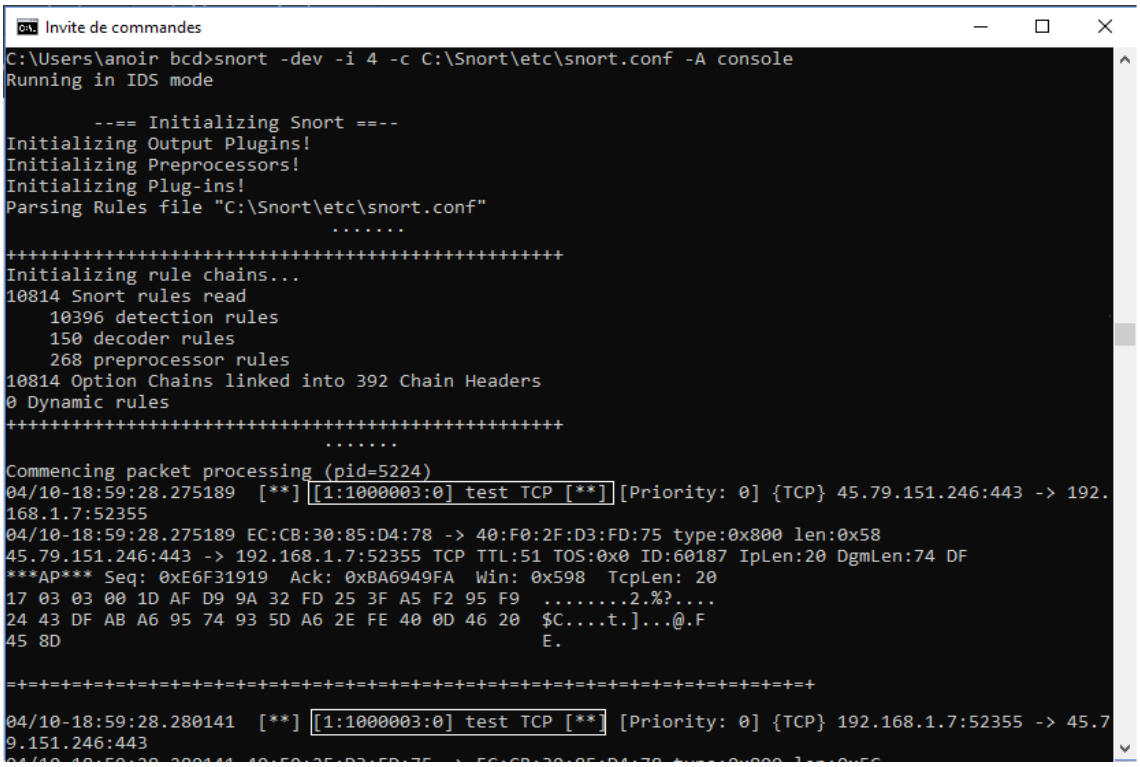


FIGURE 3.4: Snort on network intrusion detection mode.

3.2.3 Dealing with Rules

In general, there are two basic parts in the structure of Snort rule: rule header and rule options as shown in the figure 3.5:

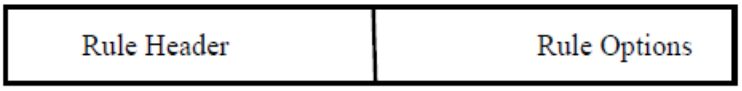


FIGURE 3.5: Basic structure of Snort rules [4]

Consider the following simple rule shown in Figure 3.6, let's talk a little about some of its elements:

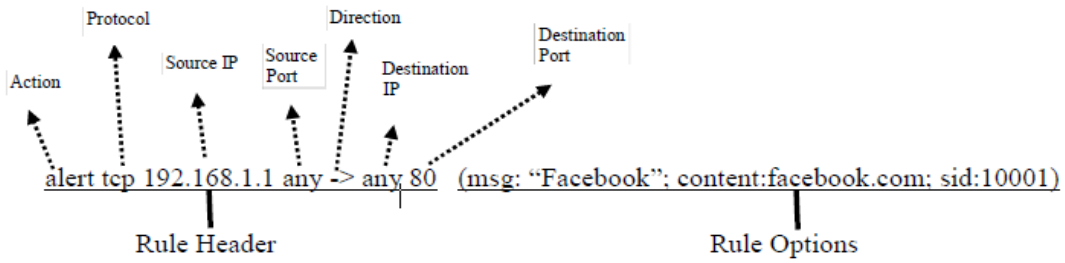


FIGURE 3.6: A simple snort rule.

By the rule above, we will create an alert if a packet matches both the header part and the rule option part to gather, more precisely, it will create an alert if an incoming

packet using TCP protocol is sent by the address 192.168.1.1 form any source port towards the destination having any IP address and the destination port number 80 and contains "facebook.com" [52].

What mentioned before, just an example of a simple rule. In the following, we will see the elements of a Snort rule with more detail, what these elements are, and the values they take, and the syntax writing in addition to see some examples.

Rule headers

The rule header is present in the section before the parentheses, it contains information about the action taken if a packet matching a rule criteria and includes some parts as illustrate in the figure 3.7.

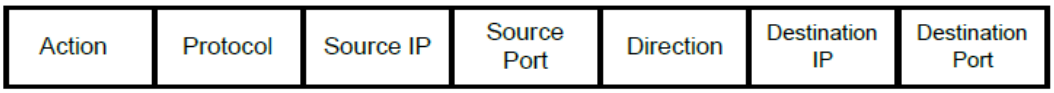


FIGURE 3.7: Snort rule header [4].

- **Actions** The action is the first element of a Snort rule. The rule action tells Snort what to do when it finds a packet that matches the criteria of the rule. There are five actions accessible by default in Snort, alert, log, pass, activate, and dynamic. This does not mean that it is not possible for the user to define its new actions. As a precaution, keep in mind that there is a difference in the application of the rules for each of two versions Snort 1.x and 2.x. If multiple rules matched with given packet, in Snort 1.x, only the action of the first rule is taken, but in 2.x version all matched rules are applied [4].

  1. **Pass** This action tells Snort to ignore this packet.This case plays an important role if we try to find some vulnerabilities in the network, we may allow some intrusion packets to pass through the network.
  2. **Log** This action used to log a packet, may be logged with a different ways, for example can be logged to files or in a database.
  3. **Alerts** The alert action is used to send an alert if an incoming packet matching with a rule conditions.
  4. **Activate** This type of actions generates an alert and activates another rule for further verification; dynamic rules are invoked in purpose.
  5. **Dynamic** This rule remains passive until activated by an another rule using the “activate” action.

- **Protocols** Protocol is the second part of a Snort rule. It defines the type of packet to which the rule will applied. There are four protocols that Snort currently can scan for suspicious behavior: ip, tcp, udp, and icmp. In the future there will be more, such as ARP, IGRP, GRE, OSPF, RIP, IPX, etc. [21]. We note that the option part of a rule can check attributes of other protocol as well. This means, it may have some criteria unrelated to a specified protocol in the header rule. In the following rule TTL (Time To Live) value is not part of the ICMP header, put Snort checks it as a part of IP header [4]:
  alert icmp any any -> any any (msg: "Ping with TTL=100"; ttl: 100; sid:100001)

- **Address** There are two places for an address in Snort rule from which to know the source of the packet and the destination. The address field may contains a

single address or a list of network addresses.The keyword "any" can be used to express all addresses. We mentioned earlier that the Snort uses CIDR notation, in which we find the address followed by a slash character and the number of the bits with the value is one on the net-mask. A CIDR block mask of /24 indicates a class C network, /16 a class B network, /8 a class A network, and /32 indicates the specific address of a machine. For example, an address 172.16.1.23/16 represents B class network 172.16.1.23 with 16 bits in the network mask. A network mask with 16 bits is 255.255.0.0 [4].See the rule below: *alert tcp any any -> 192.168.1.10/32 80 (msg: "TTL=100"; ttl: 100;)*

– **Port Number** Port numbers can be specified with a different ways. We can use the keyword "any" to express generic ports values, meaning literally all ports. Also can use intervals and negations of ports, the port ranges are indicated with the interval operator ":", see the rules below [21]. Let note that port number is meaningful only if a rule header include TCP or UDP protocol, for other protocols used like IP or ICMP port number does not play any role in the rule [4].

  – **Detect the Telnet traffic:**
  alert tcp 192.168.2.0/24 23 -> any any (msg: "Telnet traffic Detected"; sid : 1000001)
  – **A range of ports:**
  alert udp any 1024:2048 -> any any (msg: "UDP ports"; sid : 1000002)
  – **The negation symbol:**
  log udp any !53 -> any any log (msg: "UDP traffic with port different of 53"; sid : 1000003)

Table 3.1 shows the well-known port numbers for commonly used applications:

Port	Number Description
20	FTP data
21	FTP
22	SSH or Secure shell
23	Telnet
25	SMTP, used for e-mail server like Sendmail
37	NTP (Network Time Protocol) used for synchronizing time on network hosts
53	DNS server
67	BootP/DHCP client
68	BootP/DHCP server
69	TFTP
80	HTTP, used for all web servers
110	POP3, used for e-mail clients like Microsoft Outlook
161	SNMP
162	SNMP traps
443	HTTPS or Secure HTTP
514	Syslog
3306	MySQL

TABLE 3.1: Well-Known Port Numbers [4].

– **Direction** The direction part of the rule actually determines whether the address or port number is source or destination [4].

1. **A ->** Direction symbol means, the IP address and port information on the left hand side of the direction are considered as traffic coming from the source system, and the address and port information on the right side are a party of the destination system.
2. **A <-** Direction symbol means, the source system will have the address and port information on the left hand side of the direction, and the IP address and port information on the left side are considered for destination system.
3. **A <>** Bidirectional operator. This tells Snort to consider the address/port pairs either as a source or as a destination. This symbol is useful when we want to scan all traffic coming from and going to a server. An example of the bidirectional operator being used in a rule below to register both sides of a telnet session:

   log !192.168.1.0/24 any <> 192.168.1.0/24 23 (msg:"telnet session"; sid:1000001)

   – !192.168.1.0/24: this means the external addresses.
   – 192.168.1.0/24: this means the internal network addresses.

Rule options

The rule option part comes directly after the rule header part; it is included between two parentheses. A Snort rule may contains one or more options separated from each other by a semicolon character. The action defined in rule header will execute only if all specific options are occurred. Keywords in the rule options are separated from their arguments with a colon character. There are many keywords for options, in the following we will discuss the most important of them, more details are available in [4].

– **The ack Keyword** The acknowledgement field is party in tcp header, it shows the next number of 32 bit expecting to receive but is meaningful only when a tcp packet header tagged as the ACK flag. Nmap is a tool use to ping between network hosts, verify that the connection exists. try to send a packet with tagged as ACK flag, send acknowledgement number equal 0 to a desired host and waiting. If there is a response with a packet RST flag (see the table 3.2) that means this host is alive [4]. The following rule ables to detect this type of TCP ping:

alert tcp any any -> 192.168.1.0/24 any (flags: A; ack: 0; msg: "Ping detected"; sid: 1000001)

– **The content Keyword** The attackers may use specific keywords that can be seen as a special signature for that attacker. Snort has the ability to find a given on ASCII or hexadecimal format included in a packet data, with the keyword "content" and its argument maybe an ASCII string enclosed by two quotes, in the case of hexadecimal characters format will be inside a pair of tow bar as usual enclosed by two quote like this "| |" [4]. See the following tow rule to have more idea about how to use the keyword content:

1. *alert tcp any any -> any any (content: "login"; msg: "login matched";)*
2. *alert tcp any any -> any any (content: "|6c 6f 67 69 6e 0a|"; msg: "login matched";)*

We mention that there are some problems when using the keyword content. So, be careful from using it frequently, car this kind of search operations is very expensive with the huge number of processing packets in real time.

The first rule detect any packet tcp contain the string "login", where the second does the same thing, because the word "login" is just "6c 6f 67 69 6e 0a" Hexadecimal format. We can use other keywords like "depth" and offset wid "containt" to specify size of the search field. Note that content matching as default is sensitive in case of using "nocase" keyword to override this effect. [4].

– **The dsize Keyword** there are some attacks who send big size of packets, like buffer overflow. To know the size, we can use the dsize keyword which allow us to know the length of the data in packets [4]. The next rule shows an alert when the size is bigger than 6000 bytes:

alert ip any any -> 192.168.1.0/24 any (dsize: > 6000; msg: "alert of big size";)

– **The flags Keyword** This feature define in TCP header which flag bits are belong. every flag in Snort rules is argument of flags keywords [4]. Snort flags in this Table 3.2.

TCP flag bits	
Flag	**Argument character used in Snort rules**
FIN or Finish Flag	F
SYN or Sync Flag	S
RST or Reset Flag	R
PSH or Push Flag	P
ACK or Acknowledge Flag	A
URG or Urgent Flag	U
Reserved Bit 1	1
Reserved Bit 2	2
No Flag set	0

TABLE 3.2: flags table [4].

This rule show any scan trying to use RST-PSH TCP packets:

alert tcp any any -> 192.168.2.0/24 any (flags: RP; msg: "alert of RST-PSH ";)

– **The ip_proto Keyword** to detect the protocol number we need this feature, which use keyword of ip proto. in this case the protocol number is an argument. The rule detect if packets use IPIP protocol:

alert ip any any -> any any (ip_proto: ipip; msg: "alert of IP-IP ";)

Here we use the previous rule, with the number of the protocol [4]:

alert ip any any -> any any (ip_proto: 94; msg: "alert of IP-IP ";)

– **The msg Keyword** this feature is used to create text to alerts and logs, its syntax simple:

msg: "write your msg ";

If we have some symbol you want to add with message just write them with backslash [4].

– **The seq Keyword** if functionality is to check out the sequence number of a TCP packet [4]. its argument as follows:

seq: "write your sequence number here";

– **The sid Keyword** every rule has "Snort ID" created by this feature. which help us to identify rules to output modules or log scanners. Fields between 0-99 are reserved for future work. Those from 100 to 1,000,000 for rules already exist in snort, rest of numbers that more than 1,000,000 are for local rules [4]. as following rule wich add 1000001 to SID:

alert ip any any -> any any (msg: "make sid 1000001"; sid: 1000001;)

– **The tos Keyword** this keyword allow us to know about value of the Type of Service in IP header [4]. To use this feature just write as following:

tos: 3;

– **The ttl Keyword** in IP header there is a feature which allow us to know the live time of a packet That ttl have to be the same as keyword value, and we can find it with many types of IP protocol like ICMP, UDP and TCP [4]. The syntax of ttl as follow:

ttl: 132;

3.3 System Architecture

This section describes the practical side of our work, we will see the proposed architecture of our system and a detailed explanation of its elements and how to use it in an experimental case with Snort.

3.3.1 System description

The goal of creating our system is to automatically generate snort rules from a dataset using frequent patterns mining. Its architecture consist a set of successive processes as shown in figure 3.8. In the following, we will explain each phase.

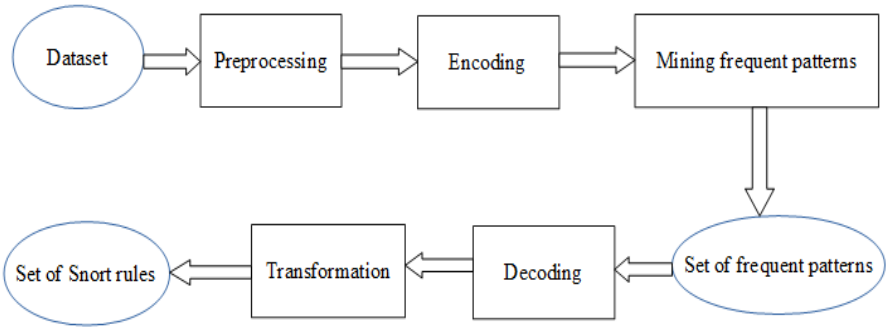


FIGURE 3.8: General architecture of our system.

– **Preprocessing** The data set LBLN used is a set of files in the format tcpdump/pcap. It is necessary to Preprocessing with a tool capable of reading the file type tcpdump/pcap, we use Wireshark software [53] for this purpose see the section 3.4.2. At this step, we can specify the number of packets to be used, also we extract only the packets of the protocol TCP because: it is the mostly used one among other protocols such as UDP, ICMP, IP which Snort can recognize in the actual version, as we described early in section 3.2.3. We know that it is difficult to integrate packets with different attributes in the same dataset, because each protocol has its own attributes. To do all this, we apply a filter by using Wireshark as we can see in figure 3.9, and save these packets result as JSON format file for performing the next step, why in particular JSON, just to keep all packets attributes and easy to track them in program implementation.

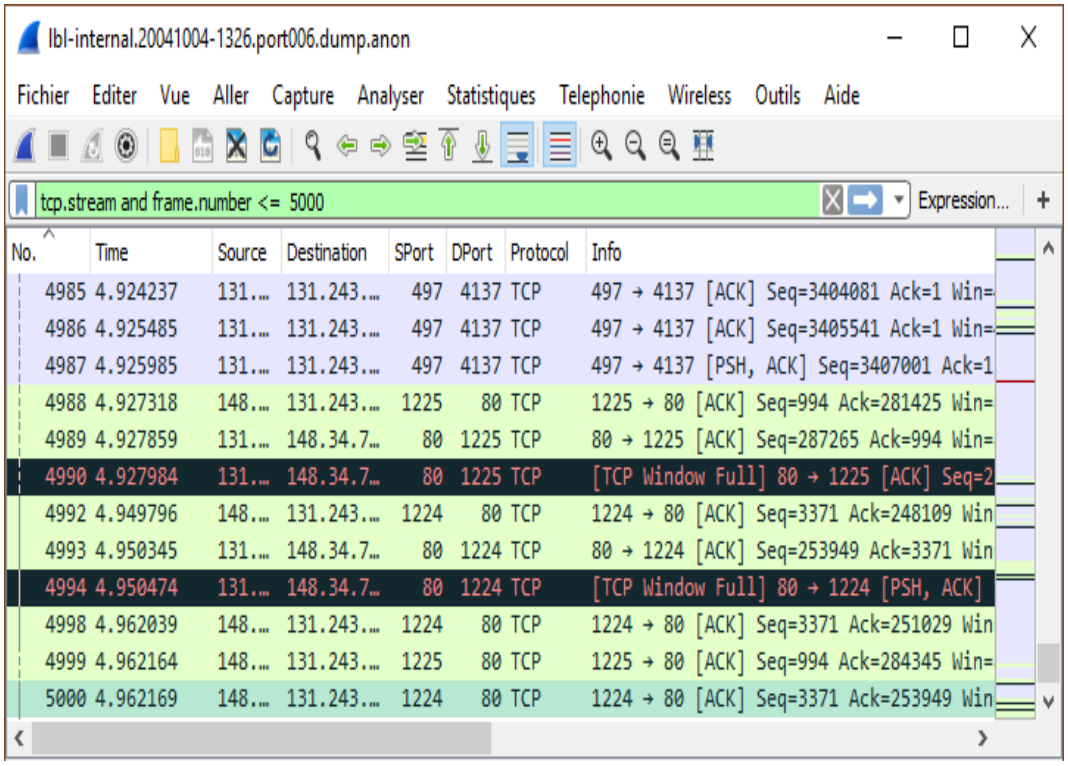


FIGURE 3.9: Dataset preprocessing using a filter in Wireshark software.

The next preprocessing is to remove all attributes that have no match with a defined Snort keyword. It seems difficult to do this manually and the huge packets number. So we created a program for this purpose, the input is a file formatted as JSON and the output file is in CSV format, it also decodes the TCP flags to the format which snort can understand as mentioned early in section 3.2.3, because these flags by default come in binary format. What should be preserved is eight attributes as we can see in the figure 3.10 is result data set header capture.

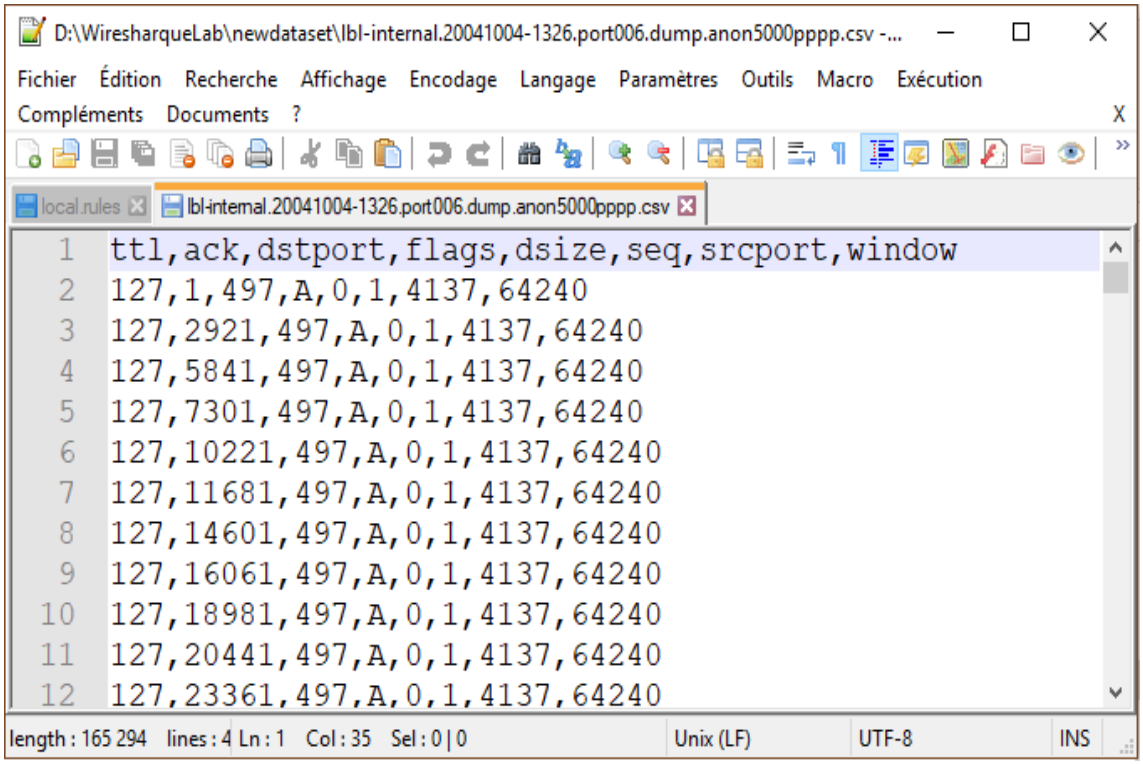


FIGURE 3.10: Dataset header after preprocessing.

All these attributes extracted: ttl (time to live), ack (acknowledgment number), dstport (destination port number), dzise (data size), seq (sequence number), srcport (source port number), window (window size value) we can find their corresponding keyword and snort accepts their value as an argument.

– **Encoding** For each item in the dataset, we will replace it with a unique integer code must be an integer number, for example: flags AB take 0, flags A takes 1, dstport 80 takes 2, etc. In the same time, we make a **dictionary** of item find its code. Why we do that, because to perform mining we use a library with algorithms ready to work and they require some conditions such as the input file must in txt format and the data inside it is integer numbers space not real characters string. This data must be separated one the other by character space. We have created a program to do this the input is a CSV file and the output is a file with txt format. But there was a problem. It can replicate the same value in different attributes, for example, ttl, scrport, dstport, seq these can take the same value as 64 and our program coder considers it as a unique. We suggested as a solution reformatting the data before coding by concatenating each value with its attribute, for example, the value 64: in the column tll will

replace by "tll:64", in the column scrport will be replaced by "scrport:64" and the others. Thus it will be coded distinctly.

- **Mining frequent patterns** At this step, a given algorithm for pattern mining is applied to the output of the previous step, we explained this in detail because of its importance in the section 3.3.2.

- **Decoding** In this step, we recuperate frequent patterns output from the previous step, but they come as a list of itemset each item is an integer number as a code, it should be decoded using the dictionary constructed early in the coding step 3.3.1.

- **Transformation** After decoding the list of frequent patterns we will take only **the maximal**, the challenge is to translate them into list of rules because Snort has a specific syntax for rules expression, as an example:

  1. [window:8760]
  2. [dsize:0, dstport:80]
  3. [flags:A, dsize:1460, srcport:80, ttl:64]
  4. [flags:AP]

  Will be:

  1. alert tcp any any -> any any (msg: "scanner attack"; window: 8760; sid: 10000;)
  2. alert tcp any any -> any any (msg: "scanner attack"; dsize: 0; flags: A; sid: 10001;)
  3. alert tcp any 80 -> any any (msg: "scanner attack"; dsize: 1460; flags: A; ttl: 64; sid: 10002;)
  4. alert tcp any any -> any any (msg: "scanner attack"; flags: AP; sid: 10003;)

3.3.2 Mining Phase

As our system is based on frequent pattern mining, two class of algorithms in the actual version of our system are used for this purpose: maximal and total. There is one algorithm available to use for the class of maxim named FPmax [12] and for the class of total, there are three algorithms: FIN [11], FPgrowth [12], Apriori [13]. Only to give the user the freedom to choose an algorithm as needed, but in our case, we prefer to use the maximal algorithm, just to eliminate the partial frequent patterns and thereby reduce false alarms resulting from non-representative rules. All the algorithms available currently on our system are using a library of pattern mining called SPMF. Knowing that we have created our own implementation, but for reasons of avoiding errors, we have used ready implementations.

We will see by using our system the mining phase in two scenarios, one by using a maxim algorithm and the other using a total algorithm for the mining. The set of data used in this phase called LBNL (see the section 3.4.2). We will use only 5000 TCP packet of them. It is important to note that a minsup value should be no bigger or smaller so much, because if we use a fairly large minsup there will be no output patterns or few if there are. For the first, we used FIN is form the total class algorithm for mining patterns proposed by Deng et al [11], the input of FIN is a set of transactions, each transaction is a set of items. And a threshold called minsup (a value between 0 and 100% or 0 and 1). For example, in figure 3.11, it contains minsup equals 0.5 which mean 50%, the first id of rules is 10000 because the snort allow us to add external rules that should be started from 10000 or above for id

rule, the minimum length of the pattern is 4, in this field we give the user to specify the number of the elements in each output pattern because this class of algorithms produces a large number of patterns and parts patterns that contain few elements that can be non-representative. The output of FIN algorithm after converting from Frequent patterns into Snort rules is a file **local.rules** contains 128 rule.

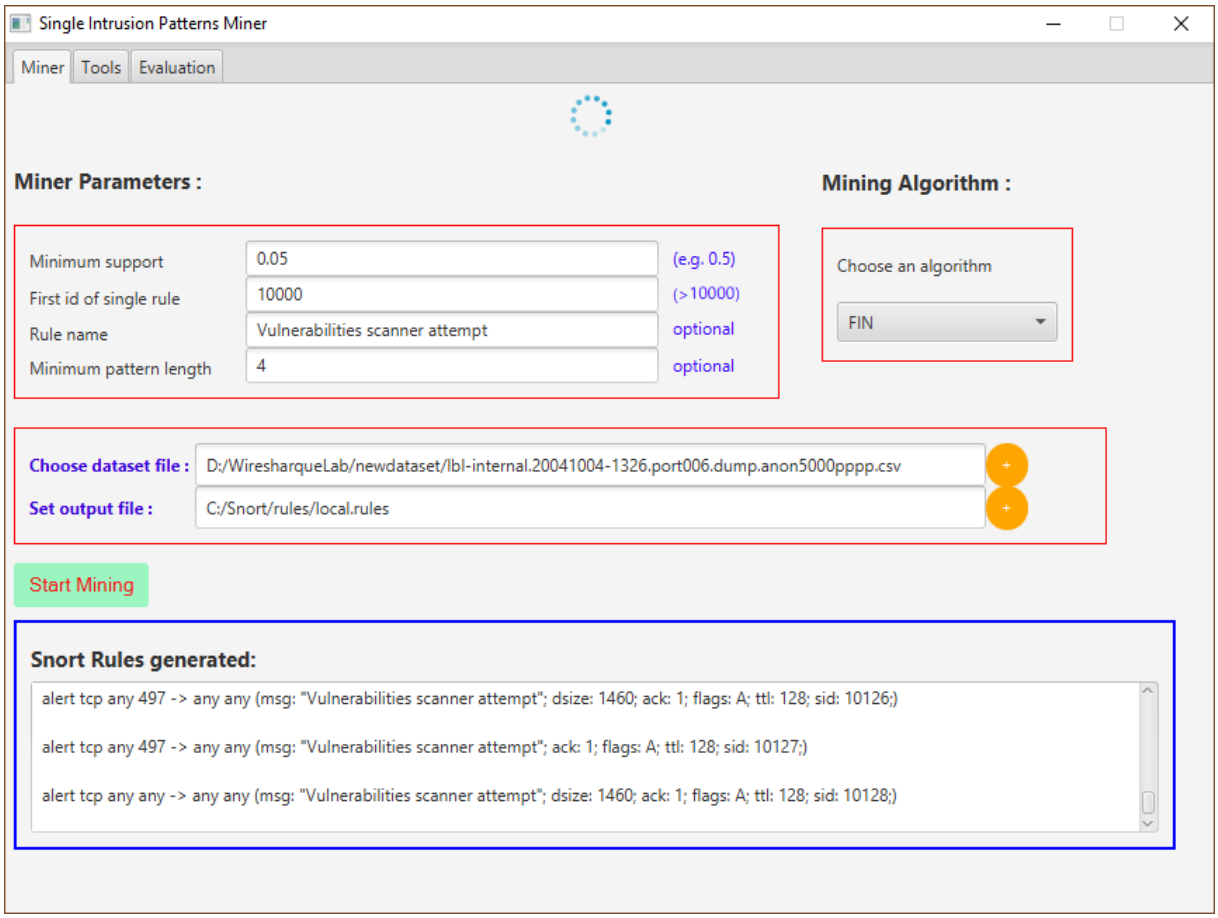


FIGURE 3.11: knowledge discovery using FIN algorithm.

In the second, we use FPMax which is an algorithm for discovering frequent maximal itemsets. It is based on the famous FPGrowth algorithm[12] and includes several strategies for mining maximal. The input of this algorithm is the same as FIN except we don't fix the length of the pattern, FPMax has the ability to avoid exponential results by eliminating Frequent Itemsets that are already included in other Frequent Itemsets, for our system, FPMax may be the best choice, because we are interested only for significant and non-recurring rules to be a good performance, those needed by Snort. In figure 3.12, the algorithm FPMax generated only 6 rules.

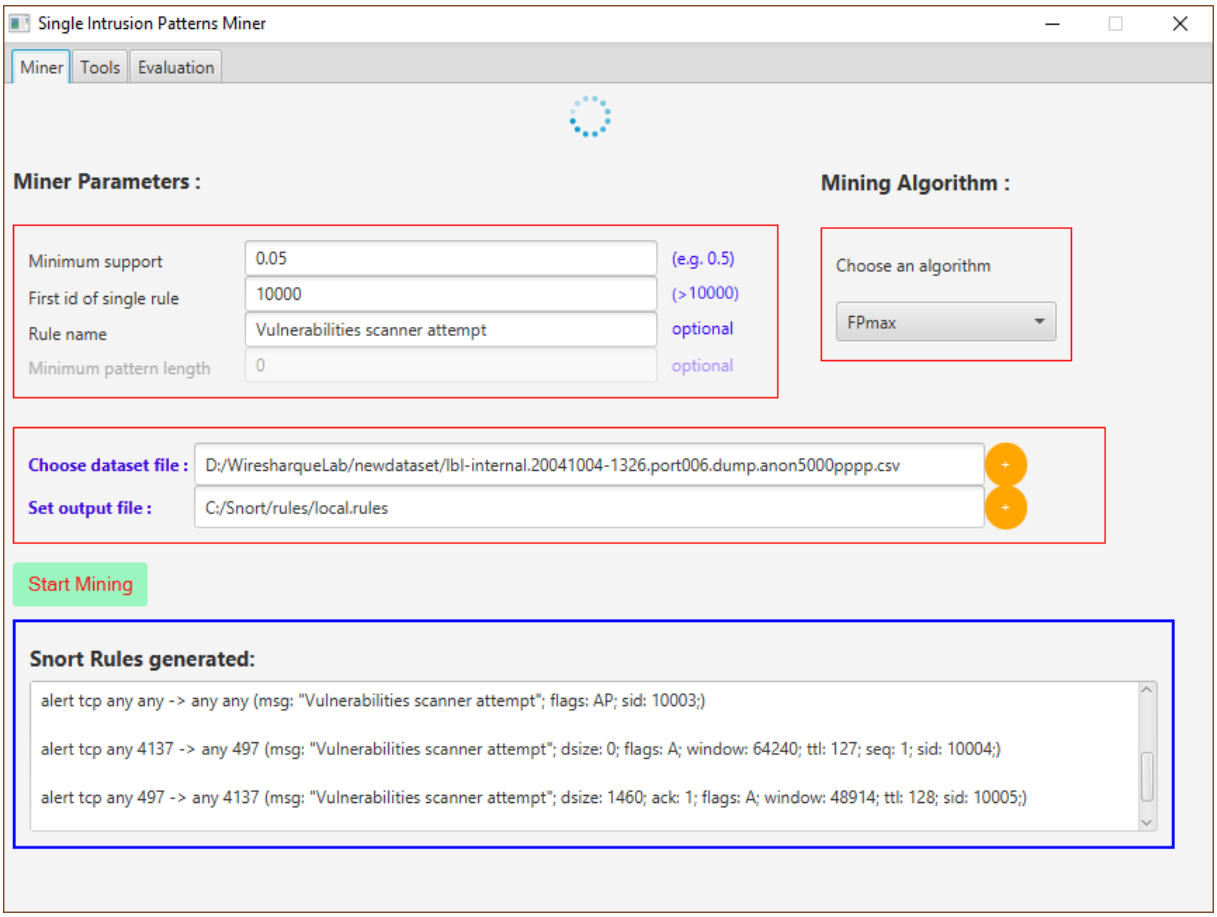


FIGURE 3.12: knowledge discovery using FPmax algorithm.

3.3.3 Production Phase

In this section, we will discuss how to use our system or our Snort extension in the real case or quasi-real case, we will do an experiment to explain that, consider the following:

- **Firstly**, with the use of Wireshark software [53] installed in the victim host, then we sniffing packets coming from a host that executes an attack software for use them later as a dataset for our extension. Our typical framework as in the figure 3.13. **Xerxes** [54] is an attack software that can block some network services (DOS attack) towards a specific target, we use it as an attacking program installed in the attack host, See the table 3.3 for more understanding.

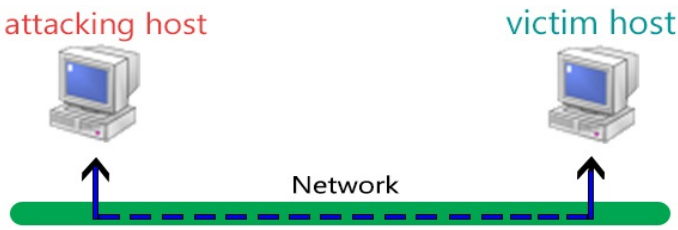


FIGURE 3.13: Framework architecture of on-line detecting .

	RAM	CPU	OS + Software
Victim Host	2048 MB	intel i5 4P	Ubuntu18.10(Linux) + snort v-2.9.13
Attackin Host	2048 MB	intel i5 4P	Kali2019.1(Linux) + DOS(xerxes)

TABLE 3.3: The description of on-line detecting environment.

- **Secondly**, now the dataset generated by us in the previous step will be used by our extension to extract knowledge which is a set of snort rule as shown in the figure 3.14. There is more than one method to add produced knowledge into Snort base, the easiest, add it in the file **local.rules** of Snort, our program does it automatically, then try to test the validity of the rules added to Snort using the command **sudo snort -T -c /etc/snort/snort.conf** If any wrong written rule will display a failure message in the console.

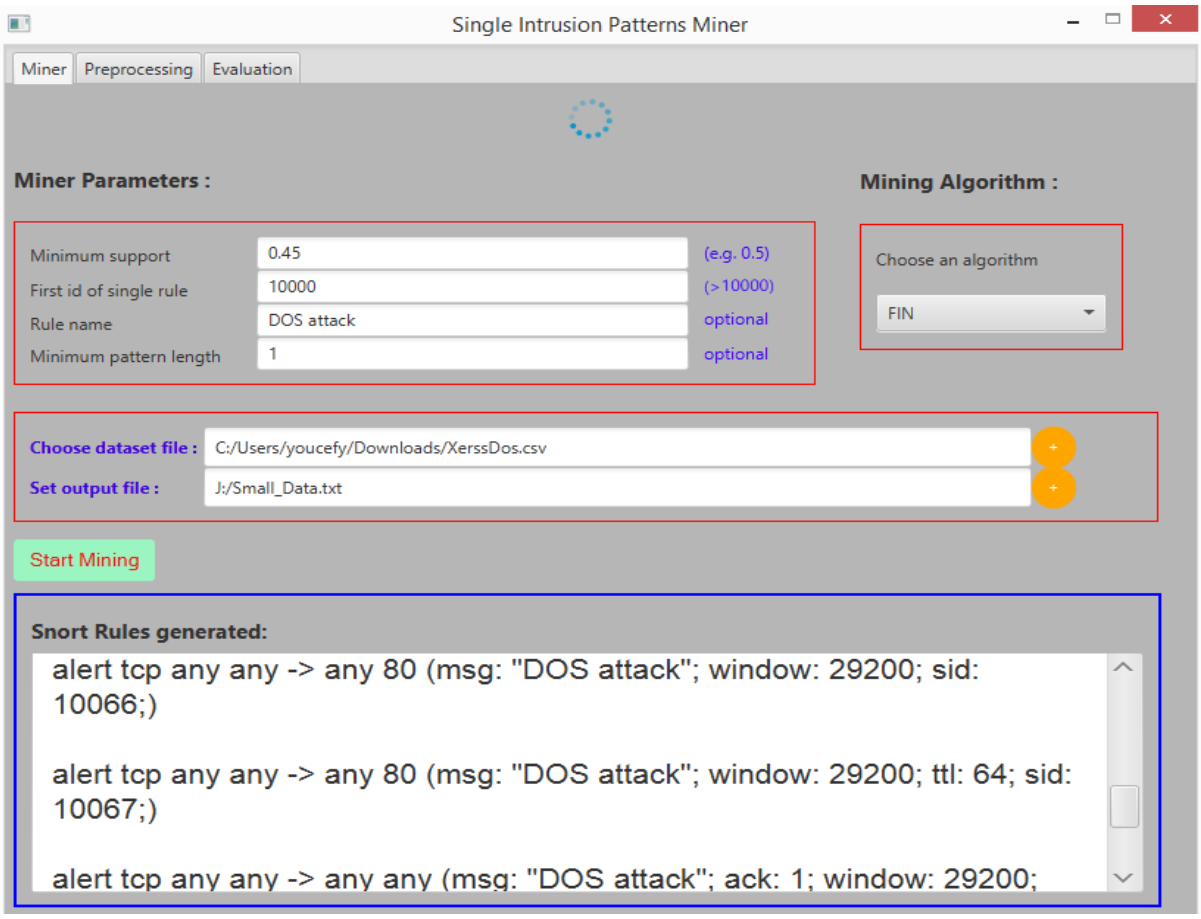


FIGURE 3.14: Knowledge extraction with our system.

- **Thirdly**, in this step, we investigate whether Snort really can detect this attack by using the previously produced knowledge. For this purpose, we will recruit the previous framework 3.13. We execute the attack on the victim of the IP address **192.168.1.6** and port 80, using the command in terminal **sudo Xerxes 192.168.1.6 80**. As shown in figure 3.15.

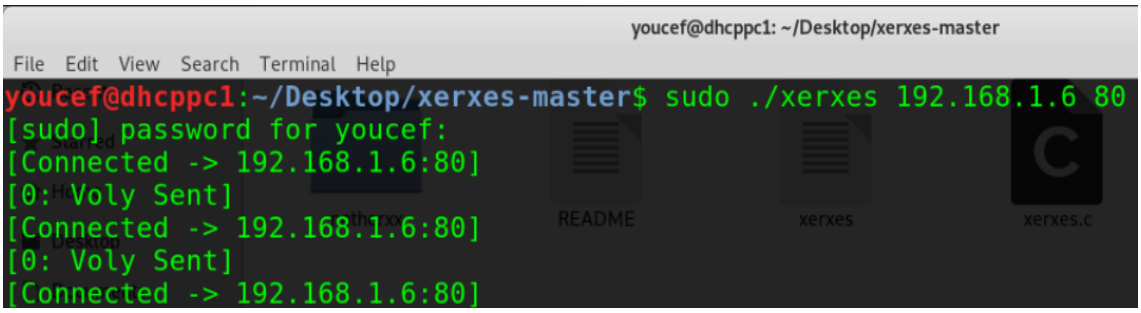


FIGURE 3.15: Executing Xerxes tools on the Attacking host.

This command **sudo snort -A console -i enp0s3 -u snort -g snort -c /etc/snort/snort.conf** tells Snort to run on IDS mode from a specific network interface "enp0s3" with the configure file path "/etc/snort/snort.conf" and send the alerts to the console, as we can see in 3.16 Snort detect this attack. We note that this scenario has done in a virtual network, knowing that we have made the same experiment in a real network.

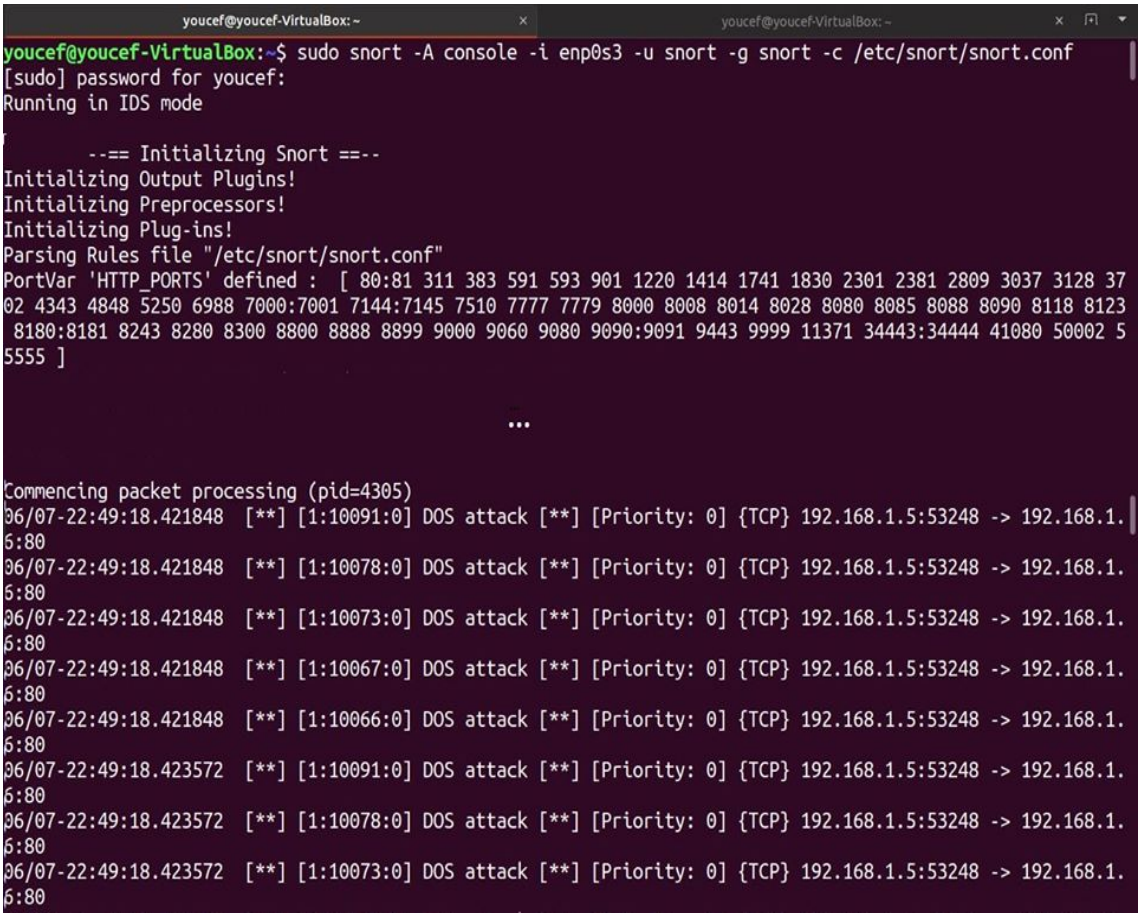


FIGURE 3.16: Attack packets detected by Snort.

3.4 Experiment and Evaluation

In this section, we will discuss a description of all experiments experiments performed on our system, which gets some result that depending on the machine power

and the choose algorithm, and corresponding test results and analysis are given.

3.4.1 Environment and setup

All programs of our system are written in Java language version 8. NetBeans 8.1 as an IDE and we use tools javafx for the interface of our system, and all algorithms for mining used are implemented in SPMF library [14].

For the evaluation our experiments are performed on ACER aspire machine, based on an Intel(R) i3-3 with 1.80 GHz processor, equipped with 4 GB of RAM, running Windows 10, 64 bit OS.

To evaluate the results of our system we adopt the following protocol. Using LBLN data set, we pick up a part of it containing only 5000 packets so as not to waste too much time and apply the following splitting: 0.7 for the mining, 0.3 for the test, of course with pre-mix the data set. As long as the data set LBLN (see 3.4.2) is pure attack packets, therefore, we will create a data set for the normal packets by sniffing normal network traffic using Snort on mode sniffer or Wireshark. Just to be able to make the confusion matrix, then calculate the value of some standard measures such as accuracy.

Using mining algorithm, each time we change the value of minimum support on the range of values, we compute the accuracy and record it to make the accuracy histogram in terms of minimum support.

To make all this we created a program that can evaluate the results of our system, it can do the matching of the knowledge base result with set of attacking and normal packets, because Snort have been created to perform in real time we can not calculate standard measures and audit control, so that Snort it can drop some packets and therefore the calculations of these measures will not be accurate. As shown in figure 3.17, begin min_sup is the starting point of minimum support which is 0.01, in each iteration, it increments by 0.05 as in the Gap field. We will see the section 3.4.3 the results and discussion of the evaluation in the use of a total and maximal algorithm for mining patterns by changing parameters in each case.

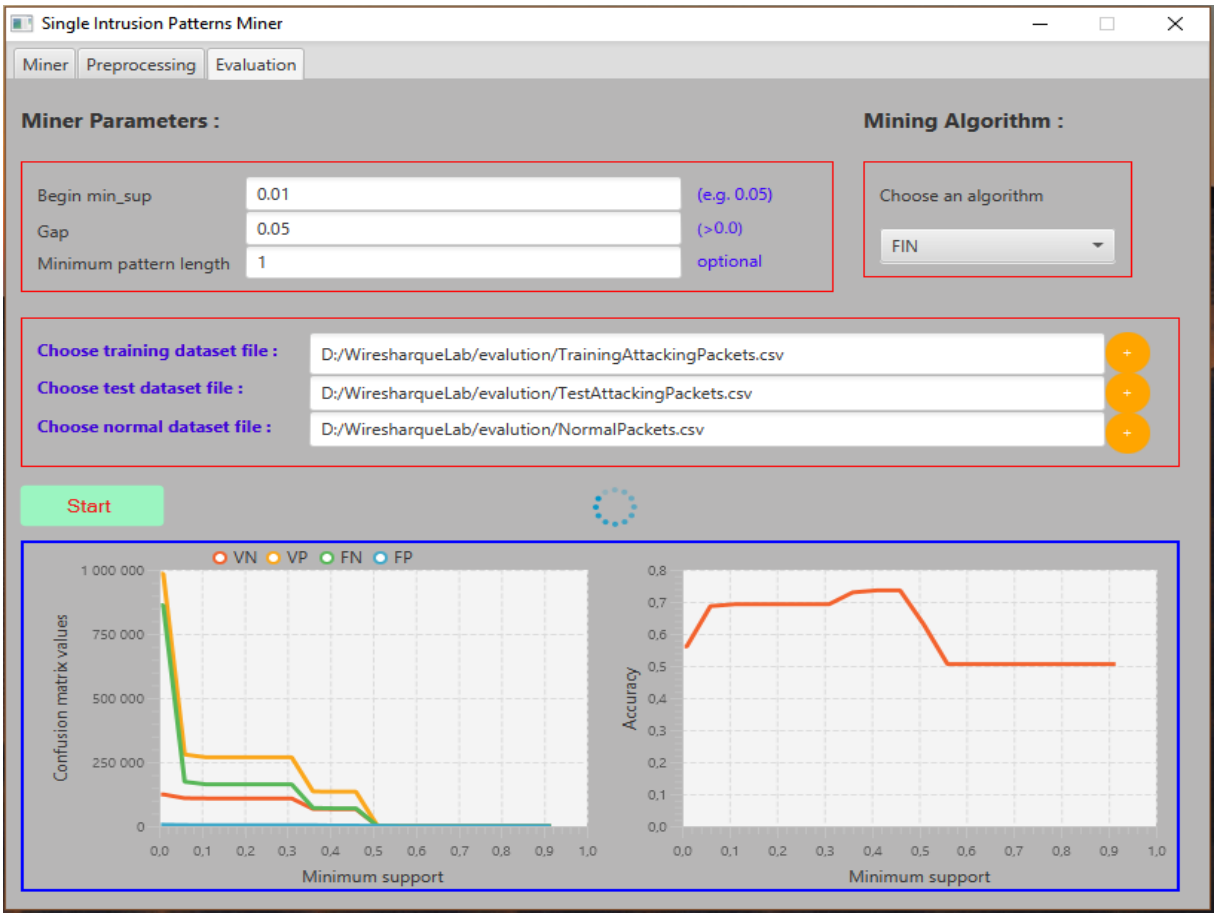


FIGURE 3.17: Evaluation results in the use of FIN algorithm.

3.4.2 Datasets

Based on our study, there are many data sets that can be used in the field of intrusion detection network-based like DARPA98, KDD99, ISC2012, and ADFA13. Many of these data sets are outdated and unreliable. Some of these data sets suffer from a lack of traffic diversity and sizes, some of them do not cover a variety of attacks, while others specify packet information and their payload that cannot reflect current trends or lack a set of features and metadata, these problems are discussed in this paper [55].

At first, we chose both the dataset DARPA and KDD CUP 99 for our study because are mostly used. But in addition to the problems mentioned earlier, there is another in them, the problem they are not suitable to the extension of Snort, these two datasets contains a small number of the attributes that we can find their correspondence with keywords defined in Snort, to be expressed in a Snort rule, then, we found out another dataset which is NSL-KDD, is a refined version of its predecessor KDD CUP 99, with it, appeared to us the same problem as in the previous. NSL-KDD dataset and its attributes described in this paper [56].

In the context of searching for alternatives, we found this paper dealing with the creation of a new dataset and discussing existing datasets, we found in it, description about the dataset named LBNL (Lawrence Berkeley National Laboratory and ICSI 2004-2005), which seemed might solve our problem, this last is full header network traffic represent the scanner attack pure and not labeled, recorded at a medium-sized site. It is a set of packets for each protocol contains all attributes put does not

have payload and suffers from heavy anonymization to remove any private information which could identify an individual IP, contains various protocols of transport layer such as TCP, UDP, ICMP and of application layer the protocols HTTP and SSH. This dataset is set of files of the total size of 11 GB, under the format of *.anon is in tcpdump/pcap save format discussed in this article [57]. We can open it with any software able to read tcpdump/pcap like Wireshark, as shown in figure 3.18.

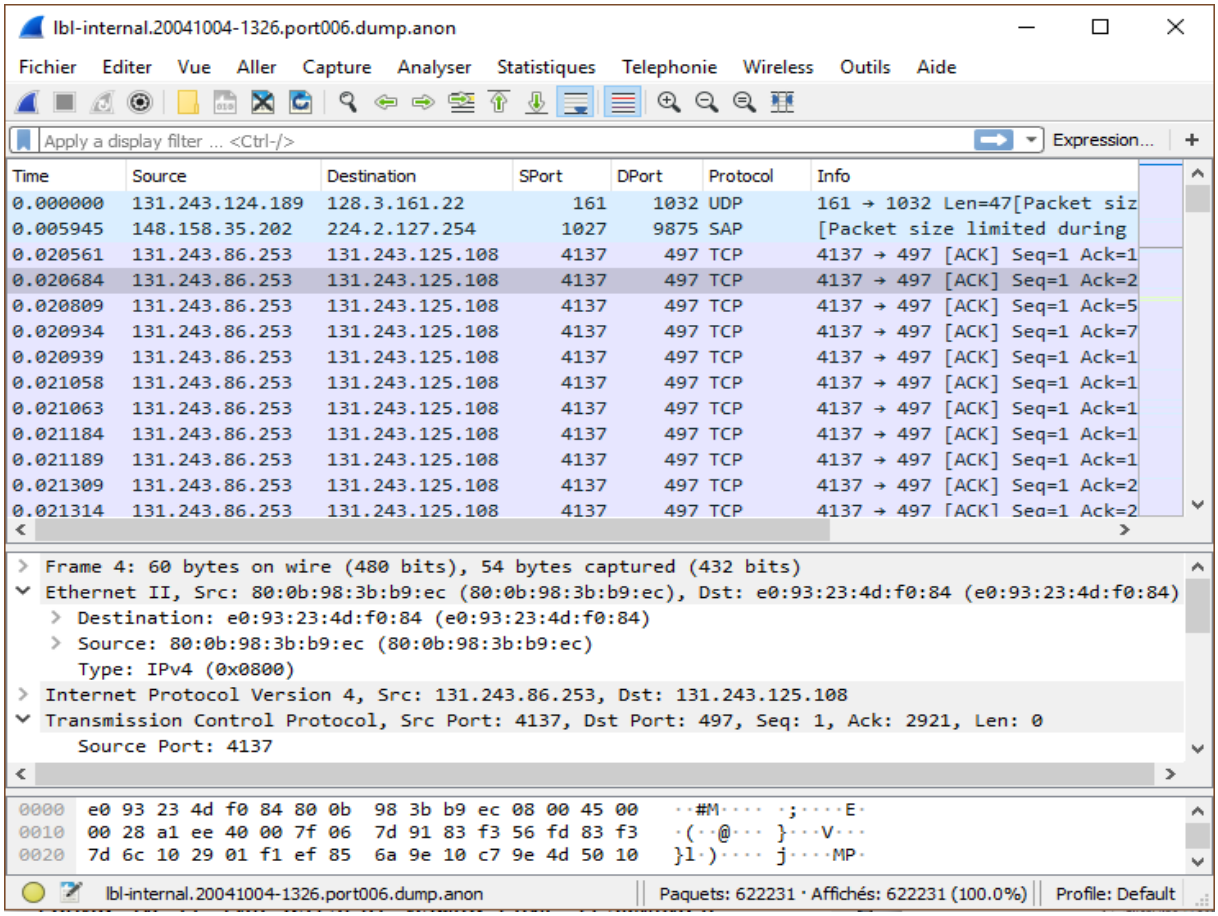


FIGURE 3.18: LBLN dataset file decoded in using Wireshark.

As we can see, Wireshark can display all attributes of a given packet with their values, below, it shows the data of any packet in decimal hexadecimal format.

3.4.3 Result and discussion

In the section 3.4.1, we talked about the evaluation protocol of our system that was created, in this section, we will discuss the results obtained. The figures 3.19 and 3.20 represent the evaluation results of our system in using a maximal (FPmax) and total (FIN) algorithm for patterns mining, which the minimum patterns length in each case for FIN algorithm equal 1 and 3 respectively, we note for these two cases by FIN1 and FIN3.

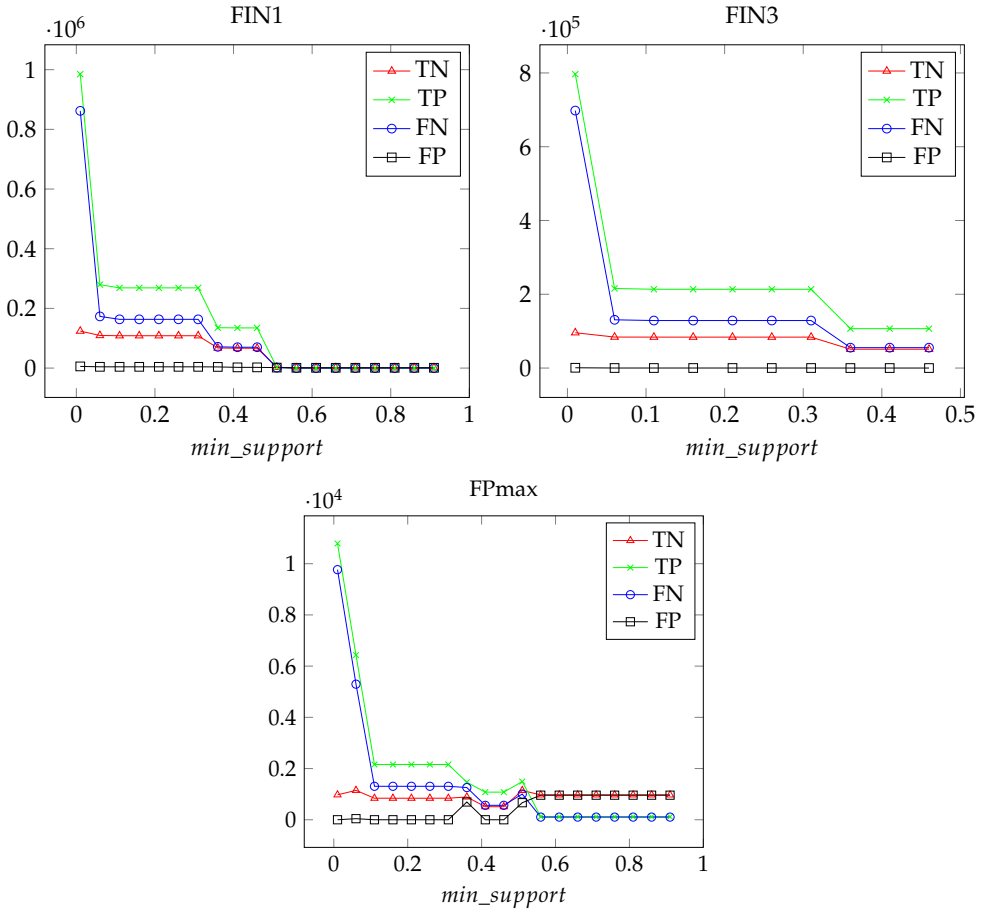


FIGURE 3.19: Confusion matrix values for the results three algorithms

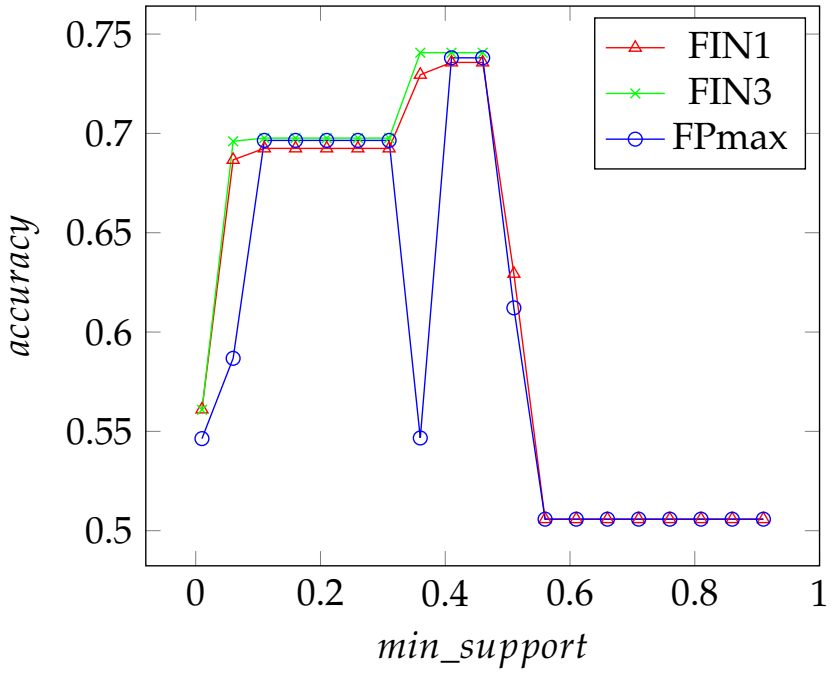


FIGURE 3.20: Accuracy values for the results three algorithms

In the term of accuracy, both algorithms FIN1 and FPmax have similar behavior but troubled at FPmax so that its highest value is 0.75 at minsup equal 0.4, this last it increases by increasing or decreasing the minsup values, that's because, at lower or higher minsup values, the most of the extracted frequent patterns will be non-representative for the tow classes, and therefore the values of accuracy be low.

In the use of FIN3, the accuracy is shown high with the height of the minsup value, this is because we only take patterns larger than or equal to 3 and therefore the number of non-representative extracted patterns will be small. The highest accuracy value is approximate is 0.75 at the minsup value of 0.4.

In term number of VN, VP, FN, FP FIN1 generate a big number then FIN3 and FPmax in order, this depends on the number of patterns extracted if they are large, the number of tests will be large.

After all this, we conclude that using a total algorithm for mining FIN with parameters: minimum patterns length is 3 and minsup is 0.4 may be the best choice based on our experiment unlike what we have mentioned in 3.3.2 and 3.3.2 the use of a maximal algorithm for mining patterns.

3.5 Conclusion

This chapter described the practical aspect of our research about "Snort extension with frequent patterns" in which we talked about Snort as a tool for detecting intrusion in computer networks, how to install and configure it on both operating systems Linux and Windows, its different running modes, and talked about details of rules expression. In addition, we talked about the architecture proposed of our system, algorithms and dataset and how to use it. We concluded the chapter with an experiment to evaluate the results of the system work and it appeared that we got some satisfactory results.

Conclusion and future works

The intrusion detection system is important for computer network security. Snort, is a misuse detection-based. It use a knowledge base to perform, in which we defined attack signatures, as we mentioned this intrusion detection approach is unable to detect new cases. Snort needs a security expert to make for each attack its definition relative to Snort syntax according to its representative features, Snort does not any positive modification for this knowledge after used.

The main problem represents in the difficult to define the attack features by a human expert in the case of huge data. In this work, we dealt with the creation of a program to extract the previously mentioned knowledge in an automatic manner based on a particular attack dataset. Assuming that the attack pattern is characterized by the frequency. Our proposed system uses frequent patterns mining techniques to extract single intrusion patterns and convert them to snort rules, the current version of the system includes two class of algorithms for frequent patterns mining: total (FIN, FPgrowth, Apriori) and maximal (FPmax), these algorithms used available in the library SPMF. The results of experiment and evaluation were acceptable to some extent, based on our experiments. We reach the best result using the FIN algorithm with an accuracy of 0.75 when the minimum support is equal to 0.4. Our work did not treat periodic maintenance issue of the extracted knowledge, in order to maintain its validity. Perhaps by integrating the role of a human expert with our system the performance would be better. Given some problems with the dataset this expansion including one protocol, it is rather complicated to configure our system with Snort as the case of all intrusion detection systems.

Our work in the future represents in further improvements to the Snort extension system, making it capable of handling other protocols network known for Snort: UDP and ICMP, and make the maintenance ability of the knowledge extracted after a period of use. In our experiments, we have included a small number of data because of the time and lack of equipment. In the future, we will explore using huge data volumes, also we think of creating a special data set only to the Snort extension for avoiding the problems we encountered. The fact is that the attack can be seen as successive events rather than one frequent event. We were thinking about creating intrusion behavior detection, which uses a technique of sequential patterns mining, to defined sequential intrusion behaviors and used them in detecting mode, we were unable to do because of time, which may be the focus of our future work.

Bibliography

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996.

[2] J. Han, M. Kamber, and J. Pei, "Data mining: concepts and techniques (the morgan kaufmann series in data management systems)," *Morgan Kaufmann*, 2000.

[3] A. Lazarevic, V. Kumar, and J. Srivastava, "Intrusion detection: A survey," in *Managing Cyber Threats*. Springer, 2005, pp. 19–78.

[4] R. U. Rehman, *Intrusion detection systems with Snort: advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID*. Prentice Hall Professional, 2003.

[5] N. SREENIVAS and P. S. KUMAR, "Detection and segregation of misbehavior node (s) for manets olsr protocol," *networks*, vol. 3, no. 11, 2014.

[6] M. J. Zaki, W. Meira Jr, and W. Meira, *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

[7] C.-F. Tsai and C.-Y. Lin, "A triangle area based nearest neighbors approach to intrusion detection," *Pattern recognition*, vol. 43, no. 1, pp. 222–229, 2010.

[8] X. Qiu, H. Li, W. Luo, and J. Huang, "A universal image forensic strategy based on steganalytic model," in *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*. ACM, 2014, pp. 165–170.

[9] H. Saneifar, "Clustering de motifs séquentiels application la détection d'intrusions," *Master's thesis, montpellier*, 2008.

[10] L.-C. Wuu, C.-H. Hung, and S.-F. Chen, "Building intrusion pattern miner for snort network intrusion detection system," *Journal of Systems and Software*, vol. 80, no. 10, pp. 1699–1715, 2007.

[11] Z.-H. Deng and S.-L. Lv, "Fast mining frequent itemsets using nodesets," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4505–4512, 2014.

[12] G. Grahne and J. Zhu, "High performance mining of maximal frequent itemsets," in *6th International Workshop on High Performance Data Mining*, vol. 16, 2003, p. 34.

[13] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[14] P. Fournier-Viger, J. C.-W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and H. T. Lam, "The spmf open-source data mining library version 2," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2016, pp. 36–40.

[15] N. Ashraf, W. Ahmad, and R. Ashraf, "A comparative study of data mining algorithms for high detection rate in intrusion detection system," *Annals of Emerging Technologies in Computing (AETiC)*, vol. 2, no. 1, 2018.

[16] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[17] A. Shafeeq and K. Hareesha, "Dynamic clustering of data with modified k-means algorithm," in *Proceedings of the 2012 conference on information and computer networks*, 2012, pp. 221–225.

[18] D. T. Larose and C. D. Larose, *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

[19] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[20] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 1–12.

[21] K. A. Jackson *et al.*, "Intrusion detection system (ids) product survey," *Los Alamos National Laboratory*, 1999.

[22] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987.

[23] H. Debar, B. Morin, F. Cuppens, F. Autrel, L. Mé, B. Vivinis, S. Benferhat, M. Ducassé, and R. Ortalo, "Détection d'intrusions: corrélation d'alertes," *Revue des Sciences et Technologies de l'Information-Série TSI: Technique et Science Informatiques*, vol. 23, pp. 32–pages, 2004.

[24] C. I. Ezeife, J. Dong, and A. K. Aggarwal, "Sensorwebids: a web mining intrusion detection system," *International Journal of Web Information Systems*, vol. 4, no. 1, pp. 97–120, 2008.

[25] M. K. Kundu, D. P. Mohapatra, A. Konar, and A. Chakraborty, *Advanced Computing, Networking and Informatics-Volume 2: Wireless Networks and Security Proceedings of the Second International Conference on Advanced Computing, Networking and Informatics (ICACNI-2014)*. Springer, 2014, vol. 28.

[26] S. C. Satapathy, J. K. Mandal, S. K. Udgata, and V. Bhateja, "Information systems design and intelligent applications," *Advances in intelligent System and Computing*, vol. 2, pp. 219–223, 2016.

[27] A. Labbi, *Handbook of integrated risk management for e-business: Measuring, modeling and managing risk*. J. Ross Publishing, 2005.

[28] R. G. Bace, *Intrusion detection*. Sams Publishing, 2000.

[29] A. Qayyum, M. Islam, and M. Jamil, "Taxonomy of statistical based anomaly detection techniques for intrusion detection," in *Proceedings of the IEEE Symposium on Emerging Technologies, 2005.* IEEE, 2005, pp. 270–276.

[30] N. Ye *et al.*, "A markov chain model of temporal behavior for anomaly detection," in *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, vol. 166. West Point, NY, 2000, p. 169.

[31] W.-H. Ju and Y. Vardi, "A hybrid high-order markov chain model for computer intrusion detection," *Journal of Computational and Graphical Statistics*, vol. 10, no. 2, pp. 277–295, 2001.

[32] H. S. Javitz, A. Valdes, and C. NRaD, "The nides statistical component: Description and justification," *Contract*, vol. 39, no. 92-C, p. 0015, 1993.

[33] C. Gates and D. Becknel, "Host anomalies from network data," in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*. IEEE, 2005, pp. 325–332.

[34] V. A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting syn flooding attacks," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, vol. 4. IEEE, 2004, pp. 2050–2054.

[35] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM transactions on Information and system security (TiSSEC)*, vol. 3, no. 4, pp. 227–261, 2000.

[36] T. Lappas and K. Pelechrinis, "Data mining techniques for (network) intrusion detection systems," *Department of Computer Science and Engineering UC Riverside, Riverside CA*, vol. 92521, 2007.

[37] S. Kumar and S. Jain, "Intrusion detection and classification using improved id3 algorithm of data mining," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, pp. 352–356, 2012.

[38] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," in *ACM Sigmod Record*, vol. 30, no. 2. ACM, 2001, pp. 37–46.

[39] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008.

[40] M. F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Computers & Security*, vol. 70, pp. 238–254, 2017.

[41] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, pp. 2227–2235, 2011.

[42] L. Zhuo, J. Zheng, X. Li, F. Wang, B. Ai, and J. Qian, *A genetic algorithm based wrapper feature selection method for classification of hyperspectral images using support vector machine*, 2008, vol. 7147.

[43] L. L. Zhong, Z. Y. Ming, and Z. Y. Bin, *Network intrusion detection method by least squares support vector machine classifier*, 2010, vol. 2.

[44] V. Bapuji, R. Kumar, A. Goverdan, and S. Sharma, "Soft computing and artificial intelligence techniques for intrusion detection system," *Networks and Complex Systems*, vol. 2, no. 4, 2012.

[45] M. Alkasassbeh and M. Almseidin, "Machine learning methods for network intrusion detection," *arXiv preprint arXiv:1809.02610*, 2018.

[46] S. Dong and X. Wang, *Research on Network Intrusion Data Based on KNN and Feature Extraction Algorithm*, 2018.

[47] K. Sequeira and M. Zaki, "Admit: anomaly-based data mining for intrusions," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 386–395.

[48] B. C. Hidayanto, R. F. Muhammad, R. P. Kusumawardani, and A. Syafaat, "Network intrusion detection systems analysis using frequent item set mining algorithm fp-max and apriori," *Procedia Computer Science*, vol. 124, pp. 751–758, 2017.

[49] I. Syarif, "Feature selection of network intrusion data using genetic algorithm and particle swarm optimization," *EMITTER International Journal of Engineering Technology*, vol. 4, no. 2, pp. 277–290, 2016.

[50] A. Mahajan, A. Gupta, and L. S. Sharma, "Performance evaluation of different pattern matching algorithms of snort," *International Journal of Advanced Networking and Applications*, vol. 10, no. 2, pp. 3776–3781, 2018.

[51] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.

[52] E. Azimi, M. Ghaznavi-Ghoushchi, and A. M. Rahmani, "Implementation of simple snort processor for efficient intrusion detection systems," in *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 3. IEEE, 2009, pp. 533–537.

[53] L. Chappell and G. Combs, *Wireshark network analysis: the official Wireshark certified network analyst study guide*. Protocol Analysis Institute, Chappell University, 2010.

[54] S. Heron, "Denial of service: motivations and trends," *Network Security*, vol. 2010, no. 5, pp. 10–12, 2010.

[55] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

[56] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[57] B. Nechaev, M. Allman, V. Paxson, and A. V. Gurtov, "A preliminary analysis of tcp performance in an enterprise network." *INM/WREN*, vol. 10, 2010.