# Naive Bayes vs Decision Trees in Intrusion Detection Systems

Nahla Ben Amor
Institute Supérieur de Gestion
41 Avenue de la liberté
2000 Le Bardo, Tunisie
nahla.benamor@gmx.fr

Salem Benferhat
CRIL - CNRS
Université d'Artois
Rue Jean Souvraz
Lens, Cedex, France
benferhat@cril.univ-artois.fr

Zied Elouedi
Institute Supérieur de Gestion
41 Avenue de la liberté
2000 Le Bardo, Tunisie
zied.elouedi@gmx.fr

## ABSTRACT

Bayes networks are powerful tools for decision and reasoning under uncertainty. A very simple form of Bayes networks is called naive Bayes, which are particularly efficient for inference tasks. However, naive Bayes are based on a very strong independence assumption. This paper offers an experimental study of the use of naive Bayes in intrusion detection. We show that even if having a simple structure, naive Bayes provide very competitive results. The experimental study is done on KDD'99 intrusion data sets. We consider three levels of attack granularities depending on whether dealing with whole attacks, or grouping them in four main categories or just focusing on normal and abnormal behaviours. In the whole experimentations, we compare the performance of naive Bayes networks with one of well known machine learning techniques which is decision tree. Moreover, we compare the good performance of Bayes nets with respect to existing best results performed on KDD'99.

## 1. INTRODUCTION

Intrusion detection in the context of information systems is regarded as a set of attempts to compromise a computer network resource security. There are two general approaches to intrusion detection [1]:

***Anomaly detection:*** The idea is that each user has a certain profile within the system that will not be changed a lot in time. Then, any significant deviation from it will be considered as an anomaly. Examples of anomaly approaches are NIDES [9], EMERALD [11].

***Misuse detection:*** The idea is that any intrusion can be described by its signature characterized by the values of its features. Systems based on this approach use different models like state transition analysis e.g. STAT [8], or a more formal pattern classification e.g. IDIOT [7] and SNORT[16].


*SAC'04*, March 14-17, 2004, Nicosia, Cyprus.


Recently, Valdes [14] has performed a hybrid approach to intrusion detection based on Bayes networks. Bayes networks [5, 10] are tools to reason with uncertain information in the probability theory framework. They use directed acyclic graphs to represent causal relations, and conditional probabilities (of each node given its parents) to express uncertainty of causal relations. Valdes [14] uses a simple form of Bayes networks, called naive Bayes, composed of two levels: one root node which represents a session class (normal and different kinds of attacks), and several leaf nodes, each of them contain a feature of a connection.

Naive Bayes have several advantages due to their simple structure. In particular, the construction of naive Bayes is very simple. Moreover, the inference (classification) is achieved in a linear time (while the inference in Bayes networks with a general structure is known to be NP-complete [3]). Finally, the construction of naive Bayes is incremental, in the sense that it can be easily updated (namely, it is always easy to consider and take into account new cases in hand). However, naive Bayes make a strong independence relation assumption: features are independent in the context of a session class. Such assumption is not always true, and may have a negative influence on the inferred results.

The aim of this paper is to provide experimental results showing that naive Bayes, with their simple structure and despite their strong assumptions, can be very competitive. Experimental results obtained in this paper use KDD'99 [15] intrusion data sets. Indeed, different experimentations are performed according to three levels of attack granularities depending on whether dealing with whole attacks, or grouping them in four main categories or just focusing on normal and abnormal behaviours.

In order to evaluate the performance of naive Bayes, we compare, based on same data, results given by naive Bayes, to those of decision trees [12], considered as one of the well known learning methods. We show that naive Bayes are really very competitive, and the performance difference with respect to decision trees is not significant. However, from computation point of view, naive Bayes are more efficient both in the learning and in the classification tasks. Indeed, the construction of naive Bayes is linear while the construction of optimal decision trees is in general an NP-complete problem [4]. We also show that naive Bayes networks give very good results with respect to winning strategy performed on KDD'99 data sets [15].

Section 2 provides respectively a description of the basics of decision trees and naive Bayes. Section 3 presents KDD'99 data set. Then, Section 4 presents different study cases that will be handled in this paper. Section 5 focuses on all attacks within decision trees and naive Bayes approaches. Then, Section 6 emphasizes on the four categories of attacks (i.e. DOS, R2L, U2R, PROBING). Section 7 handles the case of normal and abnormal connection. Finally, Section 8 summarizes main results and concludes the paper.

# 2. BRIEF REFRESHER ON DECISION TREES AND NAIVE BAYES

## 2.1 Decision trees

Decision trees are among the well known machine learning techniques. A decision tree is composed of three basic elements:
- *A decision node* specifying a test attribute.
- *An edge* or *a branch* corresponding to the one of the possible attribute values which means one of the test attribute outcomes.
- *A leaf* which is also named *an answer node*, contains the class to which the object belongs.
In decision trees, two major phases should be ensured:

1. *Building the tree.* Based on a given training set, a decision tree is built. It consists of selecting for each decision node the 'appropriate' test attribute and also to define the class labeling each leaf.

2. *Classification.* In order to classify a new instance, we start by the root of the decision tree, then we test the attribute specified by this node. The result of this test allows to move down the tree branch relative to the attribute value of the given instance. This process will be repeated until a leaf is encountered. The instance is then being classified in the same class as the one characterizing the reached leaf.

Several algorithms have been developed in order to ensure the construction of decision trees and its use for the classification task. The *ID3* and *C4.5* algorithms developed by Quinlan [12] are probably the most popular ones. We can also mention the *CART* algorithm of Breiman and al. [2]. The majority of these algorithms use a descendent strategy, i.e. from the root to the leaves. To ensure this procedure, the following generic parameters are required:
- *The attribute selection measure* taking into account the discriminative power of each attribute over classes in order to choose the 'best' one as the root of the (sub) decision tree. In other words, this measure should consider the ability of each attribute $A_k$ to determine training objects' classes. In the literature many attribute selection measures are proposed [2, 12]. We mention the gain ratio, used within the C4.5 algorithm [12] and based on the Shannon entropy, where for an attribute $A_k$ and a set of objects $T$, it is defined as follows:

$$Gain(T, A_k) = Info(T) - Info_{A_k}(T) \text{ where} \quad (1)$$

$$Info(T) = -\sum_{i=1}^{n} \frac{freq(c_i, T)}{|T|} log_2 \frac{freq(c_i, T)}{|T|} \quad (2)$$

$$Info_{A_k}(T) = \sum_{a_k \in D(A_k)} \frac{|T_{a_k}^{A_k}|}{|T|} Info(T_{a_k}^{A_k}) \quad (3)$$

and $freq(c_i, T)$ denotes the number of objects in the set $T$ belonging to the class $c_i$ and $T_{a_k}^{A_k}$ is the subset of objects for which the attribute $A_k$ has the value $a_k$ (belonging to the domain of $A_k$ denoted $D(A_k)$).

Then, $Split\ Info(A_k)$ is defined as the information content of the attribute $A_k$ itself [12]:

$$Split\ Info(T, A_k) = -\sum_{a_k \in D(A_k)} \frac{|T_{a_k}^{A_k}|}{|T|} log_2 \frac{|T_{a_k}^{A_k}|}{|T|} \quad (4)$$

So, the gain ratio is the information gain calibrated by Split Info:

$$Gain \text{ ratio}(T, A_k) = \frac{Gain(T, A_k)}{Split\ Info(A_k)} \quad (5)$$

- *The partitioning strategy* having as objective to divide the current training set by taking into account the selected test attribute.
- *The stopping criteria* dealing with the condition(s) of stopping the growth of a part of the decision tree (or even all the decision tree). In other words, they determine whether or not a training subset will be further divided.

## 2.2 Naive Bayes

Bayes networks are one of the most widely used graphical models to represent and handle uncertain information [5, 10]. Bayes networks are specified by two components:
- A *graphical component* composed of a directed acyclic graph (DAG) where vertices represent events and edges are relations between events.
- A *numerical component* consisting in a quantification of different links in the DAG by a *conditional* probability distribution of each node in the context of its parents.

*Naive Bayes* are very simple Bayes networks which are composed of DAGs with only one root node (called parent), representing the *unobserved* node, and several children, corresponding to *observed* nodes, with the strong assumption of independence among child nodes in the context of their parent.

The classification is ensured by considering the parent node to be a hidden variable stating to which class each object in the testing set should belong and child nodes represent different attributes specifying this object.

Hence, in presence of a training set we should only compute the conditional probabilities since the structure is unique.

Once the network is quantified, it is possible to classify any new object giving its attributes' values using the Baye's rule. expressed by:

$$P(c_i \mid A) = \frac{P(A \mid c_i) \cdot P(c_i)}{P(A)} \quad (6)$$

where $c_i$ is a possible value in the session class and A is the total evidence on attributes nodes. The evidence A can be dispatched into pieces of evidence, say $a_1, a_2, ..., a_n$ relative to the attributes $A_1, A_2, ..., A_n$, respectively. Since naive Bayes work under the assumption that these attributes are independent (giving the parent node $C$), their combined

probability is obtained as follows:

$$P(c_i \mid A) = \frac{P(a_1 \mid c_i) \cdot P(a_2 \mid c_i) \cdot ... \cdot P(a_n \mid c_i) \cdot P(c_i)}{P(A)} \quad (7)$$

Note that there is no need to explicitly compute the denominator $P(A)$ since it is determined by the normalization condition.

## 3. DESCRIPTION OF KDD'99 DATA SET

The data used in this paper are those proposed in the KDD'99 for intrusion detection [15] which are generally used for benchmarking intrusion detection problems. They set up an environment to collect TCP/IP dump raws from a host located on a simulated military network. Each TCP/IP connection is described by 41 discrete and continuous features (e.g. duration, protocol_type, flag, etc.) and labeled as either normal, or as an attack, with exactly one specific attack type (e.g. Smurf, Perl, etc.). Attacks fall into four main categories:
- *Denial of Service Attacks (DOS)* in which an attacker overwhelms the victim host with a huge number of requests.
- *User to Root Attacks (U2R)* in which an attacker or a hacker tries to get the access rights from a normal host in order, for instance, to gain the root access to the system.
- *Remote to User Attacks (R2L)* in which the intruder tries to exploit the system vulnerabilities in order to control the remote machine through the network as a local user.
- *Probing* in which an attacker attempts to gather useful information about machines and services available on the network in order to look for exploits.

## 4. DIFFERENT EXPERIMENTAL STUDY CASES

We handle 10% of the whole KDD'99 dataset corresponding to 494019 training connections and 311029 testing connections. Different experimentations performed in this paper suppose that the connections to classify are certainly known which is not always the case in the real TCP/IP traffic. Namely, the used testing set corresponds to an Off line traffic. The strategy behind different experimentations presented in this paper is based on the following points:
- Three levels of attack granularities: we can focalize on three cases relative to different attacks in order to handle :

- ***Whole-attacks***: all attack classes presented by KDD dataset (see [15]) in addition to the normal situation.

- ***Five-classes***: the four attack categories (i.e. DOS, R2L, U2R, Probing). Note that there are 19.65% (resp. 79.07%, 0.23%, 0.22%, 0.83%) of normal (resp. DOS, R2L, U2R, Probing) training connections and 19.48% (resp. 73.90%, 5.21%, 0.07%, 1.34%) of normal (resp. DOS, R2L, U2R, Probing) testing connections.

- ***Two-classes***: i.e. Normal and Abnormal by grouping all attacks in the same class (i.e. Abnormal).

- Gathering attacks: in the five-class and two-class cases, there are two strategies to gather results either before or after classification:

- *Gathering before classification*: the idea is to slightly modify the dataset by grouping attacks belonging to the same attack category (i.e. DOS, R2L, U2R or Probing) or by grouping them in a unique class i.e. abnormal (if we are interested with only normal and abnormal connections).

- *Gathering after classification*:
  - For the five classes, the training set remains unchanged. However, each connection classified into one of the 38 attacks is assigned to the one of the four categories it belongs to.
  - For the two classes, there are two strategies: either we do not modify the training set and each connection classified into one of the 38 attacks is simply labeled as abnormal, or we first modify the training set by gathering attacks into four categories, then each connection classified in one of these categories will be labeled as abnormal.

In each of the studied cases, the evaluation of classification efficiency is based on the *Percent of Correct Classification* (PCC) of the instances belonging to the testing set.

Lastly, note for continuous variables there are different ways to construct naive Bayes. We have analyzed two cases: normal distributions and kernel density estimation. We have noticed that the PCC when using kernel gives general better results that normal distributions. This result is not surprising, since it has been shown in [6] that kernel distributions are usually better than gaussian distributions for naive Bayes classifiers. In the following, we only give results of the best strategy, namely when using kernel density estimation.

## 5. FOCUSING ON ALL ATTACKS

This section presents experimental results, where we are interested to evaluate the ability of naive Bayes and decision trees to detect each elementary attack. Results of these experimentations are summarized in Table 1. It shows that both decision trees and naive Bayes are completely in accordance with the training set which means that this latter is coherent, i.e., almost all training instances characterized by the same attributes' values belong to the same class. This behaviour is also kept with the classification phase with a little bit advantage for decision trees.

**Table 1: PCC's in the whole-attack case**

| TRAINING SET | TESTING SET |
|---|---|
| *Decision tree* | |
| 99.99% | 91.41% |
| *Naive Bayes* | |
| 99.23% | 91.20% |

## 6. FOCUSING ON THE FOUR CATEGORIES OF ATTACKS

This section presents experimental results where we are only interested to know to which category (normal, DOS, R2L, U2R, Probing) a given connection belongs. In order to achieve these experimentations, we have grouped attacks belonging to the same category together. This is done before and after classification. For the latter, we use results

of the whole-attack case by summing the number of occurrences relative to each attack category (i.e. DOS, R2L, U2R, Probing). Table 2 gives PCC's relative to these two experimentations[1].

**Table 2: PCC's relative to five classes**

| TRAINING SET | TESTING SET |
|---|---|
| *Decision tree* | |
| 99.99% (99.99%) | 92.28% (91.81%) |
| *Naive Bayes* | |
| 99.16% (99.28%) | 91.47% (92.10%) |

Similarly to the whole-attack case, decision trees' PCC's slightly exceed those of naive Bayes in both learning and classifications phases.

Note that gathering attacks before or after classification has no influence on decision trees. However, with naive Bayes it is slightly better to gather results after classification rather than before it.

Table 3 presents confusion matrices (a two-dimensionel table with a row and column for each class. Each element of the matrix shows the number of test examples for which the actual class is the row and the predicted class is the column). It shows that Normal, DOS and Probing connections are well classified with the two techniques. This is not the case for R2L and U2R connections which are always misclassified.

**Table 3: Confusion matrices relative to five classes**

| *Decision tree* | | | | | |
|---|---|---|---|---|---|
| → | Normal | DOS | R2L | U2R | Probing |
| Normal (60593) | **99.50%** (**99.43%**) | 0.13% (0.14%) | 0.01% (0.02%) | 0.01% (0.02%) | 0.36% (0.39%) |
| DOS (229853) | 2.76% (2.94%) | **97.24%** (**96.57%**) | 0.00% (0.10%) | 0.00% (0.00%) | 0.00% (0.39%) |
| R2L (16189) | 96.55% (75.77%) | 0.02% (2.79%) | **0.52%** (**0.45%**) | 0.15% (4.27%) | 2.76% (16.71%) |
| U2R (228) | 79.82% (23.25%) | 2.63% (0.00%) | 1.75% (5.26%) | **7.89%** (**13.60%**) | 7.89% (57.89%) |
| Probing (4166) | 19.54% (15.22%) | 5.16% (6.67%) | 0.34% (0.19%) | 0.00% (0.00%) | **74.96**% (**77.92%**) |
| PCC | 92.06% (92.80%) | | | | |
| *Naive Bayes* | | | | | |
| Normal (60593) | **96.64%** (**97.68%**) | 2.78% (1.29%) | 0.23% (0.30%) | 0.11% (0.15%) | 0.25% (0.58%) |
| DOS (229853) | 3.09% (2.75%) | **96.38%** (**96.65%**) | 0.10% (0.01%) | 0.00% (0.00%) | 0.43% (0.58%) |
| R2L (16189) | 85.09% (88.80%) | 4.84% (0.01%) | **7.11%** (**8.66%**) | 2.92% (1.54%) | 0.04% (0.99%) |
| U2R (228) | 54.39% (76.32%) | 0.00% (0.00%) | 8.33% (3.95%) | **11.84%** (**10.96%**) | 25.44% (8.77%) |
| Probing (4166) | 14.19% (10.80%) | 3.36% (0.38%) | 3.79% (0.38%) | 0.48% (0.10%) | **78.18%** (**88.33%**) |
| PCC | 91.47% (92.10%) | | | | |

Regarding decision trees, this behaviour does not reflect the optimistic results obtained on the training data where 82.69% (resp. 98.93%) of U2R (resp. R2L) connections are well-classified. This is due to the fact that the proportions, in the training set, of U2R and R2L attacks are very low (0.22% for U2R and 0.23% for R2L).

[1]values between parentheses are relative to gathering whole-attacks results into five classes after classification

In fact, within decision trees, when a class is represented by a low number of training instances, then it leads to a weak learning regarding this class and consequently to a misclassification of testing connections really belonging to it. Hence, we can have new testing instances really belonging to U2R and R2L attacks, but characterized by attributes' values which deviate from those characterizing these two classes in the training set. These instances are not already learned in the construction phase and their resulting class when applying the induced tree are generally wrong.

A thorough analysis of sub-attacks pertaining to the U2R class confirms this remark since, for instance, the *"Http-tunnel"* attack which is massively presented in the testing test (158 connections over 228 U2R attacks) is characterized by $REJ$ value for the attribute *flag* but it never appears in the training set. The same explanation holds for naive Bayes since in the learning phase the conditional probability of $REJ$ in the context of U2R will be equal to zero (i.e. $P(REJ \mid U2R) = 0$). Thus, testing connections pertaining, effectively, to U2R but presenting the value $REJ$ in the attribute *flag* will be missclassified.

# 7. FOCUSING ON NORMAL AND ABNORMAL CONNECTIONS

In this section, we emphasize on normal behaviour namely we are only interested to know if a given connection is normal or not. For this purpose, we have studied confusion matrices and PCC's values by focusing on normal connections over the abnormal ones. We first consider the case where we gather all attacks before classification, then the case where gathering is made after classification using results on the whole-class case and those relative to the five-class case. Induced results are summarized in Table 4[2].

**Table 4: PCC's relative to the normal and abnormal connections**

| TRAINING SET | TESTING SET |
|---|---|
| *Decision tree* | |
| 99.99% (99.99%, 99.99%) | 93.02% (93.55%, 92.52%) |
| *Naive Bayes* | |
| 98.71% (99.31%, 99.24%) | 91.45% (92.69%, 92.40%) |

As with previous experimentations, Table 4 shows that the gap between decision trees and naive Bayes is insignificant since the PCC is almost the same within the two approaches and presents a very good rate. More precisely, the confusion matrix presented by Table 5 shows that in all kinds of experimentations, decisions trees are slightly better than naive Bayes except in the case where gathering is made before classification.

# 8. RELATED WORK AND CONCLUDING DISCUSSIONS

The major deduced remarks are the followings:

According to Table 6, we can see that dealing with all attacks, five classes or only two classes do not considerably affect the classification quality using either the decision tree

[2]values between parentheses are relative to gathering whole-attacks and five classes results into two classes after classification

**Table 5: Confusion matrix relative to the normal and abnormal classes**

| *Decision tree* | | |
|---|---|---|
| → | Normal | Abnormal |
| Normal (60593) | **99.39%** **(99.43%, 98.50%)** | 0.61% (0.57%, 0.50%) |
| Abnormal (250436) | 8.53% (7.87%, 9.17%) | **91.47%** **(92.13%, 90.83%)** |
| PCC | 93.02% (93.55%, 92.52%) | |

| *Naive Bayes* | | |
|---|---|---|
| → | Normal | Abnormal |
| Normal (60593) | **98.59%** **(97.68%, 96.64%)** | 1.41% (2.32%, 3.36%) |
| Abnormal (250436) | 10.28% (8.52%, 8.62%) | **89.72%** **(91.47%, 91.38%)** |
| PCC | 91.45% (92.69%, 92.40%) | |

**Table 6: Summary Table: PCC's on the testing set**

| WHOLE ATTACKS | FIVE CLASSES | NORMAL AND ABNORMAL |
|---|---|---|
| *Decision tree* | | |
| 91.69% | 92.06% (92.8% ) | 93.02% (93.55%, 92.52%) |
| *Naive Bayes* | | |
| **91.20%** | 91.47% (**92.10%**) | 91.45% (**92.69%**, 92.40% ) |

or naive Bayes. Namely, this never exceeds 1% difference between different strategies.

Table 7 shows that both techniques presented in this paper are competitive with the winning strategy in KDD'99 [15](and also share their failure to correctly classify R2L and U2R connections) which is based on a mixture of bagging and boosting decision tree technique [13]. It shows that even we have not used these two options there are some cases where results with decision trees and naive Bayes are equal or slightly better than those of the winning strategy. Indeed, there are strategies when within decision trees we get better PCCs with Normal, DOS and R2L categories while within naive Bayes, we perform better with U2R and Probing. These interesting results are obtained by applying different strategies such as gathering attacks before or after classification and also discretization of some continuous attributes. The complementarity between decision trees and naive Bayes can be exploited to develop a meta classifier to provide good results in each category of attacks. This is left for future works. Of course, it is clear that globally decision trees give slightly better results. However, from computation point of view, the construction of naive Bayes is largely faster than decision trees. For instance, with a Pentium III 700 Mhz, it is impossible to construct the tree (with a training set having 494019 instances) while with naive Bayes it only took few minutes. When the construction of decision trees is possible, then learning and classifying with naive bayes is generally 7 times faster than learning and classifying with decision trees.

**Table 7: Comparisons between the winning strategy, decision trees and naive Bayes**

| | WINNING STRATEGY | DECISION TREES | NAIVE BAYES |
|---|---|---|---|
| Normal | **99.50%** | **99.50%** | 97.68% |
| Dos | 97.10% | **97.24%** | 96.65% |
| R2L | 8.40% | 0.52% | **8.66%** |
| U2R | 13.2% | **13.60%** | 11.84% |
| Probing | 83.3% | 77.92% | **88.33%** |

## 9. ACKNOWLEDGMENTS

This work is supported by a french national project RNTL (Réseau National des Technologies Logicielles), DICO (Détection d'Intrusions COopérative). The authors would like to thank Frédéric Cuppens for his useful comments.

## 10. REFERENCES

[1] Axelsson, S.: Intrusion detection systems: a survey and taxonomy. Technical report 99-15, March 2000.

[2] Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J.: Classification and regression trees. Monterey, CA Wadsworth & Brooks, 1984.

[3] Cooper, G. F.: Computational complexity of probabilistic inference using Bayes belief networks. Artificial Intelligence, Vol. 42, 393-405, 1990.

[4] Hyafil, L., Rivest, R. L: Constructing optimal binary decision trees is NP-complete. Information Processing Letters, 5(1):15-17, 1976.

[5] Jensen, F. V.: Introduction to Bayesien networks. UCL Press, 1996.

[6] John, G.: Enhancements to the Data Mining Process. PhD thesis, Stanford University, 1997.

[7] Kumar, S., Spafford., E. H.: A software architecture to support misuse intrusion detection. In proceedings of the 18th National Information Security Conference, 194-204, 1995.

[8] Ilgun, K., Kemmerer., R. A., Porras, P. A.: State transition: A rule-based intrusion detection approach. IEEE Transactions on Software Engineering, 21(3), 181-199, 1995.

[9] Lunt, T.: Detecting intruders in computer systems. In proceedings of the Sixth Annual Symposium and Technical Displays on Physical and Electronic Security, 1993.

[10] Pearl J.: Probabilistic Reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmman , Los Altos, CA, 1988.

[11] Porras, P. A., Neumann., P. G., EMERALD: Event monitoring enabling responses to anomalous live disturbances. In proceedings of the 20th National Information Systems Security Conference, Baltimore, Maryland, USA, NIST, 353-365, 1997.

[12] Quinlan, J. R.: C4.5, Programs for machine learning. Morgan Kaufmann San Mateo Ca, 1993.

[13] Quinlan, J. R.: Bagging, boosting, and C4.5. Proceedings of the thirteenth national conference on AI, Vol. 1, 725-730, 1997.

[14] Valdes, A., Skinner K.: Adaptive Model-based Monitoring for Cyber Attack Detection. In proceedings of Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, 80-92, 2000.

[15] http://kdd.ccs.uci.edu/databases/kddcup99/task.html

[16] R. Marty: Snort the open source network IDS, http://www.snort.org/, 2001.