Alert correlation: Severe attack prediction and controlling false alarm rate tradeoffs

Karim Tabia, Philippe Leray

Alert correlation: Severe attack prediction and controlling false alarm rate tradeoffs

Karim Tabia
Philippe Leray
LINA/KOD UMR CNRS 6241
Ecole Polytechnique de l'université de Nantes
email: {Karim.Tabia,philippe.Leray}@univ-nantes.fr

July 26, 2010

Abstract

Alert correlation plays an increasingly crucial role in nowadays computer security infrastructures. It is particularly needed for coping with the huge amounts of alerts which are daily triggered by intrusion detection systems (IDSs), fire-walls, etc. While the use of multiple IDSs, security tools and complementary approaches is fundamental and highly recommended in order to improve the overall detection rates, this however inevitably causes huge amounts of alerts most of which are redundant and false alarms making the manual analysis of these triggered alerts time-consuming and inefficient. This paper addresses three important issues related to predicting severe attacks (attacks with high dangerousness levels) by analyzing inoffensive and preparatory attacks. i) Firstly, we address the issue of preprocessing alerts reported by the multiple detection tools in order to eliminate the redundant and irrelevant alerts and format them so that they can be analyzed by a severe attack prediction model. ii)Then, we propose a novel prediction model based on a Bayesian network multi-net allowing on one hand to better model the severe attacks and on the other hand handle the reliability of IDSs when predicting severe attacks. iii) Finally, we provide a flexible and efficient approach especially designed to limit the false alarm rates by controlling the confidence of the prediction model. The main benefits of our approach is an integrated model guaranteeing very promising prediction/false alarm rate tradeoffs with minimum expert intervention. Our experimental studies are carried out on a real and representative alert corpus generated by the *de facto* network-based IDS Snort, and show very interesting performances regarding the tradeoffs between the prediction rates and the corresponding false alarm ones.

Keywords: Alert correlation, IDSs' reliability, severe attack prediction, Bayesian multi-nets, reasoning with uncertain evidence, reject option.

1 Introduction

Computer security is always facing new challenges as information systems are more and more networked and technologies are changing and increasingly complex. Two kinds of solutions are currently deployed in order to ensure the integrity, confidentiality or availability of computer and network resources/services: *preventive solutions* (such as fire-walls and access control systems) aiming at preventing malicious users from performing unauthorized actions and *detective solutions* such as intrusion detection systems (IDSs) whose objective is detecting any malicious action targeting the information system resources and services [2]. IDSs act as burglar alarms and they are either misuse-based [29] or anomaly-based [26] or a combination of both the approaches in order to exploit their mutual complementarities [33].

Computer security practitioners often deploy multiple security products and solutions in order to increase the detection rates by exploiting their mutual complementarities. For instance, misuse-based IDSs are often combined with anomaly-based ones in order to detect both old and novel attacks and anomalies. It is important to note that all exiting anomaly-based approaches have a major drawback consisting in very high false alarm rates. These systems build profiles and models of legitimate activities and detect attacks by computing the deviations of the analyzed activities from normal activity profiles. In the literature, most anomaly-based IDSs are novelty or outlier approaches [26][30] adapted for the intrusion detection problem. Moreover, all modern IDSs (even the *de facto* network IDS Snort[1]) are well-known to trigger large amounts of alerts most of which are redundant and false ones. This problem is due to several reasons such as bad parameter settings and inappropriate IDS tuning, etc. [32]. As a consequence, huge amounts of alerts are daily reported making the task of the security administrators time-consuming and inefficient. In order to cope with such quantities of alerts, alert correlation approaches are used [14][12].

Alert correlation is the task of analyzing the alerts triggered by one or multiple IDSs in order to provide a *synthetic* and *high-level* view of the *interesting* malicious events targeting the information system. Alert correlation approaches aim either at reducing the number of triggered alerts by eliminating redundant and irrelevant ones [14] or detecting multi-step attacks [25] where the different alerts may correspond to the execution of an attack plan consisting in several steps. More recently, the authors in [5] proposed a method for prioritizing the triggered alerts according to the knowledge and preferences of the security administrators.

In this paper, we propose a complete approach which first reduces and preprocesses the alerts reported by several IDSs then efficiently predicts severe attacks. The main objectives of our approach are

[1] `www.snort.org`

i) *Severe attack anticipation:* The detection of dangerous attacks is anticipated allowing the security administrators to prevent the malicious actions by taking the appropriate countermeasures. For example, if a remote host is predicted as the source of a future dangerous attack, then the firewall can be reconfigured in order to deny access for this malicious host. Obviously, the countermeasures can be automated helping to secure an information system from potential dangerous attacks.

ii) *Efficient handling of IDSs' reliability:* Handling the IDSs' reliability is a crucial issue in alert correlation since most modern IDSs triggers huge amounts of false alerts. Our approach allows an easy and efficient handling of IDSs' reliability. Moreover, it can handle both false alert and false negative[2] rate problems. Note that most existing alert correlation approaches use alerts produced by IDSs without handling the reliability of these IDSs.

iii) *Flexible control of severe attack prediction/false alarm rate tradeoffs:* We propose a flexible mechanism allowing security administrators to control the prediction/false alarm rate tradeoffs according to the contexts, environments and the security level they want to ensure.

In our approach, the severe attack prediction model is viewed as a classification problem based on a Bayesian multi-net and we rely on Pearl's virtual evidence method [27] for handling the IDSs' reliability. In order to control the severe attack prediction/false alarm rate tradeoffs, we propose an approach allowing to reject the alert sequences where the alert correlation model's confidence is not sufficient to make a good prediction. As we will see in our experimental studies, our approach allows to significantly reduce the false alarm rates while ensuring very interesting severe attack prediction rates. Note that the other prediction and classification models such decision trees [28] cannot handle uncertain inputs (inputs provided by unreliable sources such as IDSs) and cannot offer means for assessing and controlling the model's confidence when making predictions. As we will see in the following sections, our approach requires minimum expert intervention. In fact, given a set of severe attacks to predict and historical alert logs, one can easily build and deploy an efficient model for predicting severe attacks. Moreover, our approach is easily deployed in real-time since one has just to preprocess the alert logs in real-time to extract the predictor vectors and perform real-time severe attack prediction. We can add (resp. discard) a new (resp. an existing) severe attack with minimum alterations on the existing prediction model as it is based on a multi-net.

Our approach is integrated and centered on Bayesian networks and intended to effectively address most of the challenging issues in alert correlation. As Bayesian networks are good prediction models, they are very suitable for the

[2]In the context of intrusion detection, a false negative denotes an attack that was not detected by the IDS.

severe attack prediction problem and provide simple and efficient means for dealing with the two major issues in alert correlation: dealing with false alarms and controlling prediction/false alarm rate tradeoffs. More precisely, Bayesian networks offer an elegant way for dealing with false alarms (using the *virtual evidence method* [27]) and controlling the prediction/false alarm tradeoffs with the reject option [10]. These methods are specific to Bayesian networks and perfectly meet the requirements of the severe attack prediction problem. Our experimental studies, carried out on real data collected on a university campus, clearly show the effectiveness of our approach in predicting severe attacks with low false alarm rates.

The rest of this paper is organized as follows: Section 2 provides the basic backgrounds and related works on alert correlation. In Section 3, we present our method for redundant and irrelevant alert elimination. Section 4 briefly presents Bayesian multi-nets and our prediction model taking into account the IDSs' reliability. Section 5 presents our approach for handling IDSs' reliability. In Section 6, we present our model for controlling the prediction/false alarm rate tradeoffs. Section 7 provides our experimental studies and finally, Section 8 concludes this paper.

2 Alert correlation: Approaches and challenges

In this section, we briefly review existing approaches in the alert correlation field then present the two main challenges related to this problem and for which we propose a solution.

2.1 Alert correlation

Alert correlation [14][12] consists in analyzing the alerts triggered by one or multiple IDSs and other security tools in order to provide a *synthetic* and *high-level* view of the *interesting* malicious events targeting the information system. The input data for alert correlation tools is gathered from various sources such as IDSs, fire-walls, web server logs, etc. Correlating alerts reported by multiple analyzers and sources has several advantages such as exploiting the complementarities of multiple analyzers. The main objectives of alert correlation are:

1. *Alert reduction and Redundant alerts elimination:* The objective of alert correlation here is to eliminate redundant alerts by aggregating or fusing similar alerts [14]. In fact, IDSs often trigger large amounts of redundant alerts due to the multiplicity of IDSs and the repetitiveness of some malicious events such scans, floodings, etc.

2. *Multi-step attack detection:* Most IDSs report only elementary malicious events while several attacks perform through multiple steps where each step can be reported by an alert. Detecting multi-step attacks requires analyzing the relationships and connections between several alerts [4][7][25].

3. *Alert filtering and prioritization:* Among the huge amount of triggered alerts, security administrators must select a subset of alerts according to their dangerousness and the contexts. Alerts filtering/prioritization aims at presenting to the administrators only the alerts they want to analyze[5].

In the literature, alert correlation approaches are often grouped into similarity-based approaches [14], predefined attack scenarios [25], pre and post-conditions of individual attacks [12] and statistical approaches [35][21].

In most similarity-based approaches, the objective consists in reducing the large volumes of alerts generated by the analyzers by aggregating *similar* ones on the basis of their features (victiom/attacker IP addresses, etc.). Examples of such approaches can be found in [14][35][13].

Approaches based on pre/post-conditions aim at detecting whether an attack plan (also called *attack scenario*) is in progress. An attack plan designates a complex multi-step attack consisting in several malicious actions. It often consists in a set of actions executed in a predefined sequence. Hence, in order to detect attack plans, there is a need to first detect the individual actions and correlate them in order to find which attack plan is ongoing. Pre/post-condition approaches encode attack plans by specifying for each action its pre-conditions (the actions/conditions that must executed/fulfilled before executing the current one) and post-conditions (corresponding generally to the consequences of an action). In [1], the authors propose a grammar-based approach to encode attack plans. In [25], the authors propose a logic-based approach for attack scenario detection while it is a graph representation-based approach that is used in [23]. It is important to note that most works on multi-step attack detection heavily rely on expert's knowledge. For instance, the model proposed in [12] requires identifying for each elementary attack, the preceding attacks and its consequences.

Several works propose statistical and data mining techniques for the alert correlation problem. The advantage of these techniques is their ability to exploit large data volumes and the fact that they do not require lot of expert knowledge. For instance, in [13] the authors apply clustering and data mining approaches to discover attack clusters which are then aggregated. The authors in [4] use naive causal Bayesian networks in order to detect whether an intruder is attempting to reach a given objective.

In this paper, we are interested in severe attack detection which can be viewed as a variant of multi-step attack recognition.

2.2 Alert correlation data

Alert correlation engines basically use as input data the alerts triggered by the detection and prevention tools monitoring the information system. In addition, they exploit other data such as the message notifying events, the log records of some applications such Web servers, database management systems, etc.

An alert is a message generated by un IDS when an attack or an abnormal activity is detected. It often contains an identification/name of the detected activity, its category, a severity level, the IP address of the attacker, the IP address of the victim, etc. Most of the modern IDSs can report alerts in IDMEF[3] format which is the XML intrusion detection message exchange format enabling inter-operability among IDSs and other security tools. In the following, we provide an example of an IDMEF alert generated by Snort IDS:

```
-----------------------------------------------------------------------------------------------
<IDMEF-Message>
  <Alert messageid="645478938321099">
    <Analyzer analyzerid="557835818140906" name="prelude-manager" ....
      ...
      <Analyzer analyzerid="2365682042807011" name="snort-public" manufacturer="http://www.snort.org"...
       ...
      </Analyzer>
    </Analyzer>
    <CreateTime ntpstamp="0xcae43e0c.0x4e29f000">2007-11-13T16:15:24.305327+01:00</CreateTime>
    <DetectTime ntpstamp="0xcae43e0c.0x4e13c000">2007-11-13T16:15:24.304989+01:00</DetectTime>
    <AnalyzerTime ntpstamp="0xcae43e0c.0x4e2d9000">2007-11-13T16:15:24.305383+01:00</AnalyzerTime>
    <Source spoofed="unknown" interface="eth1">
      <Node category="unknown">
        <Address category="ipv4-addr">
          <address>172.16.10.3</address>
        </Address>
      </Node>
      <Service ip_version="4" iana_protocol_number="6" iana_protocol_name="tcp">
        <port>53014</port>
      </Service>
    </Source>
    <Target decoy="unknown" interface="eth1">
      <Node category="unknown">
        <Address category="ipv4-addr">
          <address>XXX.XXX.XXX.XXX</address>
        </Address>
      </Node>
      ...
</IDMEF-Message>
-----------------------------------------------------------------------------------------------
```

It is important to note that alerts can be generated by different categories of detection tools (such as network-based IDSs, host-based IDSs, application-based IDSs, etc.) and prevention tools (such fire-walls, access control systems, etc.).

According to the IDSs' configurations, the alerts generated by IDSs are sent directly to the security administrators, stored in a log file, etc. Generally, alerts are generated in IDMEF format and sent to a server which collects these alerts for alert correlation purposes. For instance, Prelude-manager[4] is a collection

[3]IDMEF stands for the Intrusion Detection Message Exchange Format. `http://www.ietf.org/rfc/rfc4765.txt`

[4]`http://www.prelude-ids.org/`

of open source tools allowing to collect (via secure network connections) and store the alerts and events triggered by various IDSs and other security tools such that alert correlation engines can use them. Note that in our experimental studies, our data is collected on a university campus monitored by the Snort IDS and the triggered alerts are collected by the Prelude-manager tool.

2.3 IDSs' Reliability: A crucial issue

The most important problem users of IDSs and security practitioners face is the one of false alerts which correspond to legitimate activities that have been mistakenly reported as malicious actions by the IDSs. Indeed, nowadays IDSs are well-known to trigger high false alarm rates. For instance, the well-known Snort IDS indicates for each attack, whether false alerts could be triggered. In an experimental evaluation of Snort IDS [32], the authors concluded that 96% of the triggered alerts are false positives. Hence, taking into account the reliability of IDSs is an interesting issue for alert correlation tasks such as the prediction of severe attacks which is the focus of this work. For instance, if it is known that 90% of alerts reporting a malicious event triggered by a given IDS are false, then this information should be taken into account if such alerts should be exploited as inputs by the alert correlation tool. Note that there is to the best of our knowledge no work addressing the handling of IDSs' reliability for severe attack prediction. As we will see in Section 5.1, Bayesian networks offer a natural and efficient way to handle the reliability of IDSs as a problem of probabilistic reasoning in presence of uncertain data.

2.4 Controlling prediction/false alarm rate tradeoffs

Most of detection tools are equipped with means allowing their users to configure them according to theirs needs, required analysis overload, etc. For instance, some Snort detection modules offer thresholds to be set by the security administrators in order to limit the volume of alerts. If one wants to ensure a high security level then he must deal with large volumes of alerts since any suspected event will be detected and an alert will be triggered. If, on the contrary, one wants to ensure an acceptable rate of false alarms, then this can be done only at a lower level of security. Note that the security level may change from day to day depending on the security recommendations and bulletins. Hence, it is essential for our system to provide a way for controlling the prediction/false alarm rate tradeoffs. In our approach, we propose to rely on the confidence of the prediction model to control the number of false alarms.

3 Alert preprocessing and redundancy/irrelevance elimination

In this section, we present the method we use for eliminating redundant and irrelevant alerts and preprocess raw IDMEF alerts in order to be analyzed by

our severe attack prediction model. This method is inspired by the works of [3][4].

3.1 From IDMEF to preprocessed alert windows

In this paper, severe attack prediction is viewed as a classification problem which a special kind of prediction. Hence, the analyzed data must be preprocessed and summarized into *formatted* input data composed of features with *high information gain* in order to ensure high prediction rates. Then in order to analyze a sequence of alerts $Alert_1$, $Alert_2$,.., $Alert_k$ reported by one or multiple IDSs to determine if this alert sequence *plausibly* will lead or be followed by a severe attack $Attack_i$, we need to preprocess the raw IDMEF alerts into attribute variables. In IDMEF standard, three dangerousness levels are defined: *low*, *medium* and *high*. In our case, we analyze sequences of low and medium severity level alerts in order to predict high severity level attacks. Note that only the alerts with low/medium severity levels (often due to inoffensive attacks such as *scans*) are concerned with the reduction and preprocessing. The alerts with high severity levels are those we want to predict. The alert preprocessing provides two reduction levels:

1. **Redundant alert elimination step** In order to eliminate the redundant alerts (for instance, alerts with same identifier (sid), targeting the same victim by the same attacker), raw alerts $Alert_1$, $Alert_2$,.., $Alert_k$ generated by IDSs are summarized in alert windows where the occurrences of each alert $Alert_i$ is associated with a variable A_i whose domain is $\{0, 1\}$ where the value 0 means that the alert $Alert_i$ was not observed in the analyzed sequence while value 1 denotes the fact that the alerts $Alert_i$ have been reported.

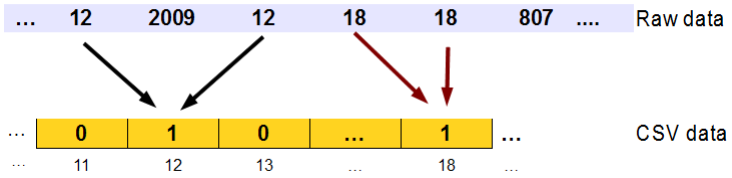


Figure 1: Example of formatting raw alert sequences into presence/absence data

As it is shown in Figure 1, raw alert sequences are formatted into CSV[5] data where several occurrences of a given alert is denoted by the value 1 while the value 0 denotes the fact that this alert has not been triggered during the analyzed alert sequence. For instance, the alert with identifier 12 has been triggered twice in this example but in the formatted data, we have only once the value 1 corresponding to this alert and denoting that

[5]Comma separated values

the alert 12 has been observed. As for the alert whose identifier is 11, its value is 0 since it has not been reported in this example. Hence, even if during an alert sequence, an alert $Alert_i$ has been reported several times, this information will be represented with one variable A_i. Clearly, this way of preprocessing raw redundant data into presence/absence data is efficient for eliminating the redundant occurrences of the same alerts especially since there are several alerts which are reported hundreds of times during short time durations. Note that one can also count the occurrences of the alerts $Alert_i$ instead of just reporting whether this alert has been reported or not. One can also take into account time constraints between alerts. For instance, if we want to eliminate the alerts $Alert_i$ occurring after another alert $Alert_j$, then the variable A_i is set to 1 only if an $Alert_i$ is reported before an alert $Alert_j$ within the alert sequence. As for the duration of alert windows, it can be fixed experimentally or manually set by the expert according to the analysis periodicity, the processing overload, etc. For instance, in our experimentations, a two hours raw alert sequence involving thousands of alerts is summarized by a single alert window.

2. **Irrelevant alert elimination step** In the redundant alert elimination step, we eliminate the redundant alerts by substituting several occurrences of the same alert by a single variable value. However, for predicting severe attacks, there is not need to use all the low/medium level alerts. This problem is well-known in the machine learning field as the feature selection problem: given an initial input set of features, we want to select a subset of features guaranteeing *good* performances for the considered task (classification, regression, etc.). Indeed, not only feature selection methods [36] significantly reduce the dimensionality of the feature set but often improves the model's performances. Moreover, the feature selection process does not require expert knowledge as there are several statistical methods for achieving this task [36]. In our application, feature selection helps us to eliminate the irrelevant alerts, namely low/medium level alerts which are irrelevant for predicting severe attacks. As we will see in our experimental studies, only some dozens of low/medium severity level alerts are needed to predict most of severe Web attacks. Note that Bayesian networks (which are used to build our prediction model) can also be used as a feature selection method since the Markov-blanket property [24] allows to determine the subset of input variables which are sufficient to predict a given target variable.

Regarding the criteria used for grouping the raw alerts, the alerts reported by IDSs can be grouped into several levels of granularity according to the alerts semantics, attack strategies and characteristics of the attacks to predict, etc. For instance, raw alerts can be summarized into alert windows by grouping alerts per IDSs, victims, attackers, etc. In our experimental studies, Snort IDS alerts are grouped according to their *sid* (alert signature identifier) and victim's IP address and attacker's IP address. Namely, an alert window ω summarizes the alerts targeting the same victim and launched by the same attacker. Our

choice is motivated by the nature of the severe attacks we want to predict which are not distributed and target only single Web applications.

It is important to note that the preprocessing of raw alerts data into presence/absence data ignores the sequential or temporal relationships between the alerts. Very often, the attackers execute on purpose some of their actions each time in a different sequence to escape detection. Our main objective in this paper is to show on real data that even if we ignore these sequential/temporal dependencies between alerts, we can still achieve good prediction/false alarm rate. Note also that our model exploits some correlations between attributes since it is based on non naive structures (in our experimental studies, we used MWST [11] to build our multi-net where each network has a tree structure involving most of the strong correlations between alert features). These correlation relationships are automatically extracted from the training data.

4 Bayesian network-based model for predicting severe attacks

This section briefly presents Bayesian networks and their use as severe attack prediction models taking into account the IDSs' reliability.

4.1 Bayesian network classifiers

Bayesian networks are powerful graphical models for modeling and reasoning with uncertain and complex information [20]. They are specified by:

i) *A graphical component* consisting in a DAG (Directed Acyclic Graph) allowing an easy representation of the domain knowledge in the form of an influence network (vertices represent events while edges represent *dependance* relations between these events), and

ii) *A probabilistic component* allowing to quantify the uncertainty relative to relationships between domain variables using conditional probability tables (CPTs).

Bayesian networks are used for different types of inference such as the maximum a posteriori (MAP), most plausible explanation (MPE), etc. As for applications, they are used as expert systems for diagnosis, simulation, classification, etc.

Supervised classification consists in predicting the value of a target variable given the values of observed variables. Namely, given observed variables A_1,..,A_n describing the objects to classify, it is required to predict the *right* value of the class variable C among a predefined set of class instances. Bayesian network-based classification is a particular kind of probabilistic inference ensured by computing the greatest a posteriori probability of the class variable

given the instance to classify. Namely, having an instance of the attribute vector $a_1a_2..a_n$ (observed variables A_0=a_0, A_1=a_1,.., A_n=a_n), it is required to find the most plausible class value c_k ($c_k \in$ C=$\{c_1, c_2,..,c_m\}$) for this observation. The maximum a posteriori classification rule can be written as follows:

$$Class = argmax_{c_k \in C}(p(c_i/a_1a_2..a_n)), \quad (1)$$

where the term $p(c_i/a_1a_2..a_n)$ denotes the posterior probability of having the class instance c_i given the evidence $a_1a_2..a_n$. This probability is computed using Bayes rule as follows:

$$p(c_i/a_1a_2..a_n) = \frac{p(a_1a_2..a_n/c_i) * p(c_i)}{p(a_1a_2..a_n)} \quad (2)$$

The denominator of Equation 2 can be ignored because it does not depend on the different classes. Equation 2 means that posterior probabilities are proportional to the likelihood of the evidence and class prior probabilities while the evidence probability is just a normalizing constant. Note that most works use naive or semi-naive Bayesian network classifiers such as TAN (Tree Augmented Naive Bayes) and BAN (Bayesian Network Augmented Naive Bayes) [9] which make strong assumptions in order to simplify the classifier's structure learning from the data. The other Bayesian network classifiers require more general structure learning and parameter estimation (building the CPT tables).

Bayesian network-based approaches are widely used in many areas of computer security. More particularly, Bayesian classifiers are used in intrusion detection in several works such as [37][34][31]. In alert correlation, a Bayesian approach is used in [35] for alert fusion. Bayesian classifiers are also used in [3][4][15] where the authors use naive, TAN and other Bayesian network models for detecting attack plans and severe alerts. Note that all the works on detecting multi-step and severe attacks use naive or semi-naive prediction models. Note also that to the best of our knowledge, there is no work addressing the IDSs' reliability handling. In the following, we propose a Bayesian network-based approach for severe attack prediction.

4.2 Severe attack prediction as a classification problem

Severe attack prediction consists in analyzing sequences of alerts or audit events in order to predict future severe attacks. In this paper, severe attack prediction is modeled as a classification problem where the variables are defined as follows:

1. **Predictors (attribute variables):** The set of predictors (observed variables) is composed of the set of relevant alerts for predicting the severe attacks. Namely, with each relevant alert variable A_i reports the presence/absence of alert $Alert_i$ in the analyzed sequence.

2. **Class variable:** The class variable C represents the severe attacks variable whose domain involves all the severe attacks $Attack_1$,.., $Attack_n$

to predict and another class instance *NoSevereAttack* representing alert sequences that are not followed by severe attacks.

Note that in order to automatically build the Bayesian network classifiers, the training data is labeled. Namely, for each alert vector (a presence/absence data representing an alert sequence) in the training data set, we associate a label denoting either the severe attack following this alert sequence or a special label *NoSevereAttack* in case where the alert sequence will not be followed by a severe attack. In our experimental studies, the labeling task is done automatically: we have only to list the severe attacks we want to predict. Then the preprocessing tool we developed for this purpose (see Section 7.1), checks for each preprocessed alert sequence whether a severe attack is detected. In the positive case, the current alert sequence terminates exactly at the detected severe attack and the alert window is shifted backwards to ensure that all the actions that led to this severe attack are present within the current alert sequence.

The main advantages of this approach for predicting severe attacks are:

1. *Minimum expert intervention*: the expert has just to identify the severe attacks he wants to predict and use feature selection methods to select the *relevant* features for predicting these attacks on historical alert logs. Moreover, the prediction models are automatically learnt from validated empirical data.

2. *Easy deployment:* one has just to preprocess the alerts logs in real-time to extract the predictors and perform real-time severe attack prediction.

3. *Easy update:* the prediction model can be easily updated and tuned by automatically relearning the models on more appropriate data.

4.3 Bayesian multi-net classifiers

In standard Bayesian classifiers, there is a unique network which encodes the influence relationships relative to all the training data. However, the independence relationships are not the same over the different classes. More particularly, in our severe attack prediction application, each severe attack is correlated (in the statistical/causal sense) only with a small and specific set of other alerts most of time because several attacks are undertaken by worms and scripts executing the same malicious events. For instance, the attack *WEB-IIS CodeRed v2 root.exe access* whose Snort signature identifier (*sid*) is 1256 is correlated in our data set with alerts having *sid*=2 (*(http_inspect) DOUBLE DECODING ATTACK*), 18(*(http_inspect) WEBROOT DIRECTORY TRAVERSAL*) and 1002 (*WEB-IIS cmd.exe access*) because this worm uses a directory traversal attack (causing alert with *sid* 18 and 2) in order to access the *cmd.exe* executable on MS Windows systems. As we will see in our experimental studies, local correlation modeling leads to better likelihood estimation for the classification task. Note that when most structure learning algorithms identify correlations

using statistical tests, consequently the obtained network encodes mainly the correlations relative to the *majority* class. In a multi-net classifier, each class instance c_i is associated with a network N_{c_i} encoding only local independence relationships learnt from training instances belonging exclusively to the class c_i. The a priori frequencies relative to the different classes can be encoded by a root node C representing the class variable [8] or just by a local probability distribution as in [9]. The generic procedure for learning a multi-net from empirical data is provided in Algorithm 1.

Input: Data set D of labeled training examples, Structure learning algorithm and its parameters
Output: Multi-net
Begin

1. Partition the training set D into subsets S_i where each S_i contains the data belonging only to class c_i.
2. For each training subset S_i,
   (a) Learn a Bayesian network N_{c_i} on S_i
      i. Learn the structure of N_{c_i} on S_i,
      ii. Learn the parameters of N_{c_i} on S_i,
   (b) Compute the frequency of class instance c_i.

ALGORITHM 1: Bayesian multi-net classifier learning.

Algorithm 1 shows that learning a Bayesian multi-net is performed by partitioning the training data set D into several partitions where each partition S_i contains only the training instances sharing the same class c_i. Note that learning a network on each S_i can be done using a different algorithm and feature space since not all the feature are relevant for all classes. The multi-net based classification's procedure is provided by Algorithm 2.

Input: Bayesian multi-net classifier, the instance to classify $a_1..a_n$
Output: Most probable class instance $c_i \in D_C$
Begin

1. For each possible class instance c_i,
   (a) Compute the likelihood of the evidence, namely $p_{N_{c_i}}(a_1..a_n)$.
   (b) Combine the likelihood with a priori frequency of class c_i, namely compute $p_{N_{c_i}}(a_1..a_n)$*$p(c_i)$.
2. Return the class instance having the utmost a posteriori probability degree, namely $argmax_{c_i \in D_c}(p_{N_{c_i}}(a_1..a_n)$*$p(c_i))$.

ALGORITHM 2: Classification based on a multi-net classifier.

In [18][19], Bayesian multi-nets are considered as variants of unrestricted Bayesian classifiers. Authors in [9] argue that multi-net classifiers are as effective as the best state of the art classifiers while they are more expressive and less complex than BANs. Moreover, multi-nets allow to update the model with minimum modifications. For instance, in order to add a new class, one has just to learn an additional network for the added class. One can also update existent models by repeating the learning of local networks of the updated classes. In [8], author proposed approaches for selecting the branching variable A_i that will be linked with the class variable C. Finally, multi-net classifiers offer the opportunity to choose a specific learning algorithm and select the most appropriate features for each class.

5 Handling IDSs' reliability

This section deals with the handling of IDSs reliability using Pearl's virtual evidence method.

5.1 Reasoning with uncertain data: Pearl's virtual evidence method

Pearl's virtual evidence method [27] offers a natural way for handling and reasoning with uncertain evidence in the framework of probabilistic networks. In this method, the uncertainty indicates the confidence on the evidence: to what extent the evidence is believed to be true. In our context, if an IDS triggers an alert and we know (from past experience for example) that this event is a false alarm in 95% of the cases then we are in presence of uncertain evidence. The main idea of Pearl's virtual evidence method is to recast the uncertainty relative to the uncertain evidence E on some *virtual* sure event η: the uncertainty regarding E is specified as the likelihood of η in the context of E.

Example

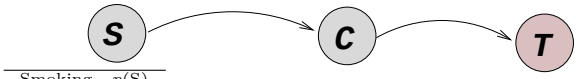
Let us illustrate Pearl's virtual evidence method on the simplified lung cancer problem presented by the network of Figure 2.

Smoking	p(S)
True	0.52
False	0.48

Cancer	Smoking	p(C|S)
True	True	0.73
True	False	0.4
False	True	0.27
False	False	0.6

Figure 2: A Bayesian network modeling the lung cancer/smoking problem with variables S (Smoking) and C (Cancer)

According to the network of Figure 2, if a patient has lung cancer (assume that the result of an infallible test is positive), then the probability that this patient is smoker is $p(S{=}True/C{=}True){=}0.66$. Assume now that the used medical tests revealing whether a patient has lung cancer or not are reliable at 95%. According to Pearl's virtual evidence method, this uncertainty will be recasted on a virtual event (say T for $Test$) as illustrated in network of Figure 3.



Smoking	p(S)
True	0.52
False	0.48

Cancer	Smoking	p(C|S)
True	True	0.73
True	False	0.4
False	True	0.27
False	False	0.6

Test	Cancer	p(T|C)
True	True	0.95
True	False	0.05

Figure 3: The Bayesian network of Figure 2 extended with the virtual evidence node T ($Test$)

Now, given a positive test saying that the considered individual has lung cancer, than the probability that this person smokes is $p(S{=}True/T{=}True){=}0.65$. In the following we provide our method for handling IDSs' reliability for predicting severe attacks.

5.2 Handling IDSs' reliability using Pearl's virtual evidence method

In order to apply Pearl's virtual evidence method for efficiently handling IDSs' reliability, we must first assess the IDSs' reliability by means of empirical evaluations (an expert can examine for each alert type triggered by an IDS, the proportion of true/false alerts). An expert can also subjectively (by experience) fix the reliability of the IDSs composing his intrusion detection infrastructure. Now, after assessing the reliability of the IDSs in triggering the alerts A_1,..,A_n, the handling of the uncertainty regarding an alert sequence, proceeds as follows:

1. For each alert variable A_i, add a child variable R_i as a virtual evidence recasting the uncertainty bearing on A_i. The domain of R_i is $D_{R_i}{=}\{0,1\}$ where the value 0 is used to recast the uncertainty regarding the case $A_i{=}0$ (alert A_i was not triggered) while the value 1 is used to take into account the uncertainty in the case $A_i{=}1$ (alert A_i was triggered).

2. Each conditional probability distribution $p(R_i/A_i)$ encodes the reliability that the observed values (triggered alerts) are actually true attacks. For example, the probability $p(R_i{=}1/A_i{=}1)$ denotes the probability that the observation $R_i{=}1$ is actually due to a real attack.

The example of Figure 4 gives a tree-augmented naive Bayes network augmented with five nodes R_1,R_2,R_3,R_4,R_5 for handling the uncertainty relative to

variables A_1,A_2,A_3,A_4,A_5 respectively. Henceforth, the observed variables are R_1,R_2,R_3,R_4,R_5 while variables A_1,A_2,A_3,A_4,A_5 are associated with the actual malicious/normal activities and they cannot be directly observed.

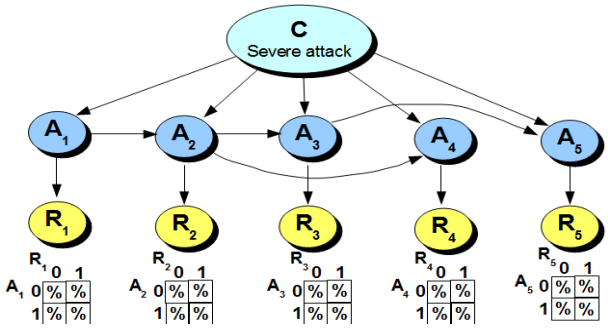


Figure 4: Example of a Bayesian network classifier handling the reliability of inputs

When analyzing an alert sequence $r_1 r_2..r_n$ (an instance of observation variables R_1,..,R_n), we compute $argmax_{c_i}(p(c_k/r_1..r_n))$ in order to predict severe attacks. Note that in practice it is less complicated to assess the false/true positive rates than assessing the false/true negative rates which requires analyzing the whole activities (for example, all the network traffic) in order to evaluate the proportion of attacks that were not detected by the IDSs. In the following, we provide an efficient mechanism for controlling the severe attack prediction/false alarm rate tradeoffs.

6 Controlling severe attack prediction/false alarm rate tradeoffs

This section presents our approach based on classification with reject option for controlling the severe attack prediction/false alarm rate tradeoffs.

6.1 Classification with reject option

Classification with reject option [10] is an efficient solution allowing to identify and reject the data objects that will be *probably* misclassified. Indeed, in many application areas such medical diagnosis, military target identification, etc. it is better to reject (not classify) an object than misclassifying it. The reject option is crucial in our application especially for limiting the false alarm rates because the reliability of inputs directly impacts the predictive and discrimination power of our prediction models (the greater the uncertainty in the input data, more difficult will be the prediction of the nature of the analyzed alert sequence). Moreover, this approach allows the user to control the tradeoffs between the

severe attach prediction and the underlying false alarm rates.
A classification model can be seen as a means of discriminating the frontiers defined by the objects sharing the same class (as shown on illustration (a) of Figure 5 where the classifier is represented by the line separating classes c_1, c_2 and c_3). As shown on Figure 5, in the literature there are two kinds of classification with reject option:

1. **Ambiguity reject:** As shown in illustration (b) of Figure 5, in ambiguity reject the object to classify *belongs* to several classes simultaneously, which makes the classifier confused. This may be caused by the fact that the modeled classes are not completely disjoint. This type of rejection is implemented by detecting data objects that are close to several classes simultaneously. Several techniques are used to implement this approach. In [10], the author proposed using the a posteriori probability of the instance to be classified in different classes. In our approach for controlling the prediction/false alarm rate tradeoffs, we will act on the width of the grey-colored region of illustration (b) of Figure 5 separating the different classes to specify the confidence with which our prediction model is expected to make the good predictions (see Equation 4).

2. **Distance reject:** This situation occurs when the instance to classify does not belong to any of the classes represented by the classification model. This may be due to the existence of a class which is not represented or that the item to classify is an outlier. As shown in illustration (c) of Figure 5, the distance reject is used to define the classes modeled by the classifier and can thus reject what is beyond its frontiers. In the illustration (c) of Figure 5, the grey-colored region represents the modeled classes and all the data instances that fall outside its frontier will be rejected and considered as outliers. In practice, this solution is implemented by measuring the degree of belonging or distance from the object to classify with the different classes. A threshold is often set below which the objects to classify are rejected. In Bayesian network-based classifier, this reject option can easily be implemented by fixing a threshold on the likelihood or the a posteriori probability of the object to classify with respect to the existing classes.

The distance reject is often used for anomaly and outlier detection. For instance, we used a Bayesian network reject in [6] in order to detect novel attacks in network traffic. However, this reject is not relevant for controlling the severe attack prediction/false alarm rate tradeoffs since alert sequences are either followed by severe attacks or not (there is no other alternative to be captured by the distance rejection). The reader can refer to [22] for discussions on classifiers' confidence evaluation and reject option rules interpretation. In the following, we present our approach for controlling the attack prediction/false alarm rate tradeoffs based on the ambiguity reject option.

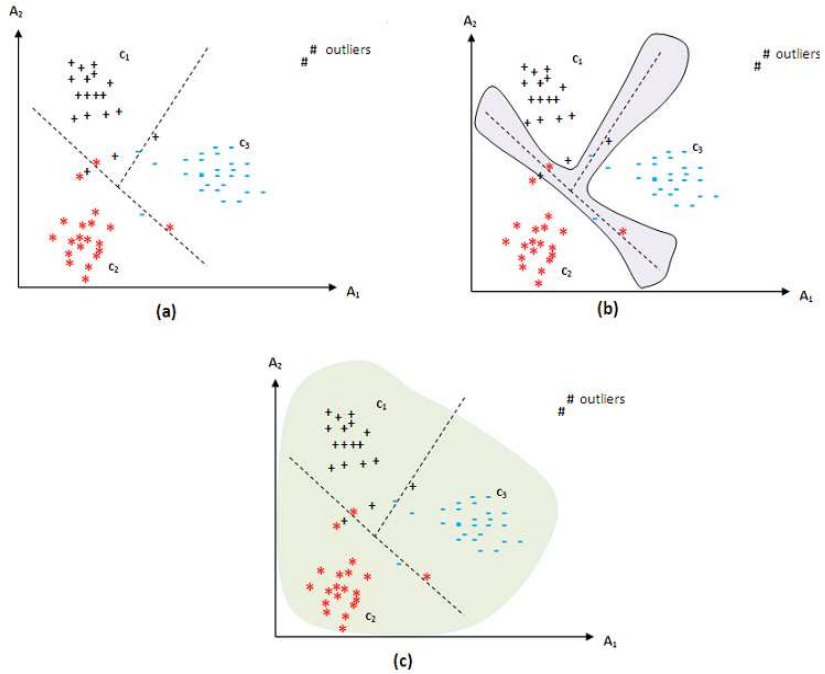


Figure 5: Geometric representation of classification (a) ambiguity reject (b) and distance reject (c)

6.2 Controlling severe attack prediction/false alarm rate tradeoffs

Bayesian network-based classifiers are naturally suitable for implementing the classification with reject option as classification is ensured by computing a posteriori probabilities of class instances given the data to be classified. Each probability $P(c_i/a_1..a_n)$ can be interpreted as the confidence of the classifier that the instance to classify belongs to class instance c_i. In our application, we are interested in controlling the attack prediction/false alarm rate tradeoffs according to the contexts and needs of each final user. For example, a user may want an alert correlation tool with high confidence (with minimum false alerts). This objective requires rejecting the alert sequences where the tool is not very confident. Let us define the *confidence* concept in our application as the unsigned value of the difference of the probability that the instance to classify $a_1a_2..a_n$ is not a severe attack and the probability that $a_1a_2..a_n$ is actually a severe attack. This measure is adapted from the works of Leray *et al* [22] on classifiers confidence evaluation. It is done by measuring the gap between the probability that the alert sequence will not be followed by a severe attack (namely $p(c_i = 0/a_1..a_n)$) and the greatest probability that the event will be

followed by a severe attack. Namely,

$$\varphi(a_1..a_n) = |p(c_i = 0/a_1..a_n) - \max_{c_i \neq 0} p(c_i/a_1..a_n)|, \quad (3)$$

where c_i=0 denotes the class instance representing alert sequences that are not followed by severe attacks while class instance $c_i \neq 0$ denote class instances associated with the severe attacks to predict. The value of $\varphi(a_1 a_2..a_n)$ gives an estimate of the classifier's confidence that the analyzed alert sequence will/will not be followed by a severe attack. Hence, a user wanting to reject all alert sequences where the probability of not being an attack is not twice the probability that they are severe attacks is done by setting the reject threshold L to the value 1/3. Note that the a posteriori probabilities of the classes given the alert sequence to analyze must be normalized in order to be used for implementing the reject option. Then, the Bayesian decision rule of Equation 1 will be reformulated as follows:

$$Class = \begin{cases} argmax_{c_k \in D_C}(p(c_i/a_1..a_n)) & if \varphi(a_1..a_n) > L \\ \emptyset & otherwise \end{cases} \quad (4)$$

The value $\emptyset$ denotes the reject decision, namely the instance to be classified is rejected because the condition $\varphi(a_1..a_n) > L$ is not satisfied. In the following, we provide our experimental studies on real IDMEF alert corpus.

7 Experimental studies

Our experimental studies are carried on real and recent alert log files produced by Snort IDS monitoring a university campus network. These alert logs represent three months activity collected during summer 2007 within the framework of PLACID project[6] dealing with probabilistic and logic approaches for alarm correlation in intrusion detection. The input to our system consists in alerts generated by Snort IDS gathered in IDMEF format. In the following we briefly present our alert preprocessing tool needed both in the training phase (for preparing the labeled training data) and analysis phase for predicting severe attacks.

7.1 Data preparation: IDMEF alert preprocessing

In order to preprocess IDMEF alerts into CSV data that can be used for training our models, we developed an alert preprocessing tool taking as inputs IDMEF alert log files and preprocessing options and outputs alert sequences in CSV format.

Among the preprocessing options provided by the user in the preprocessing option file, we find:

[6] `http:\\placid.insa-rouen.fr`

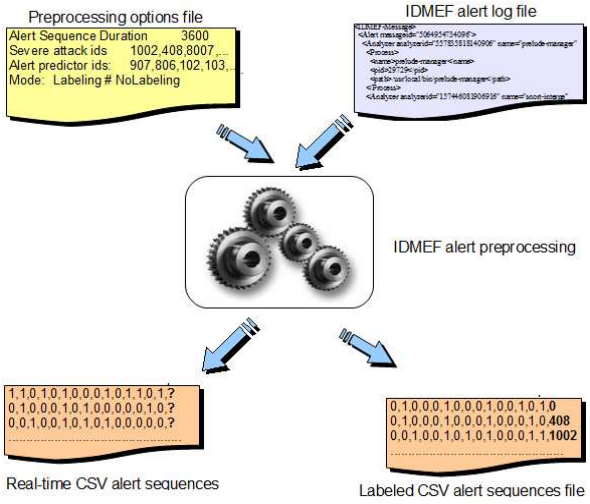


Figure 6: IDMEF alert preprocessing process

- *Window duration (in secs):* the duration of the alert windows can be defined by the user according the traffic flow rates, the processing overload, analysis periodicity, etc. Our prediction models analyze alerts summarized in alert sequence vectors. For instance, if the alert sequence duration is set to 1 hour, than our preprocessing tool will represent all the alerts involving the same attacker and victim and reported during the last hour in one alert sequence vector.

- *Predictors set:* This set provides the alert identifiers (*sid*) that will be used as predictor variables. If the provided set is empty, then the preprocessing tool will perform a first pass to identify all possible predictors (every low/medium alert will be considered as a predictor) and a second pass to preprocess the alerts.

- *Severe attacks set:* This set lists the set of severe attack identifiers (*sid*) the user wants to predict. This set is need only when building training sets which need to be labeled.

Note that our preprocessing tool is used in off-line mode to provide the labeled data for training the prediction models. More precisely, our preprocessing tool proceeds for each alert log file as follows

1. It selects from each IDMEF alert only the needed information for our formatting task (Source and Destination IP addresses, Sid (Snort alert Identifier), Severity level). Recall that IDMEF alerts are XML data and our preprocessor uses an XSLT script to achieve this selection.

2. The alerts are sorted according to their source/destination IP addresses.

3. The sorted alert sequences are formatted into presence/absence CSV data as explained in Section 3.

The labeling task is done automatically following the attack identifiers listed in the severe attacks set. In prediction mode, the tool preprocesses in real-time sequences of raw IDMEF alerts generated by IDSs and submits the preprocessed alerts for analysis. Note that in order to determine the relevant alerts for our task, we first preprocessed the raw IDMEF alerts into CSV data (where each possible alert is associated with an alert variable in the presence/absence data sequences). Then using the information gain measure selection, we selected only relevant alert variables for our prediction task. Note also that the labeling is done automatically as explained in Section 4.2. However, if the detected severe attacks are false alarms, then the labeling will be done incorrectly. This issue can be dealt with by excluding the alerts revealing mistakenly severe attacks.

7.2 Training and testing data sets

Our data sets are obtained from real IDMEF alerts reported during three months by the Snort IDS. We first preprocessed the first month of collected alerts in order to build the training data set and preprocessed the second month to build the testing set. Table 1 provides details on the severe attacks we used in our experimentations.

		Training set		Testing set	
Sid	Snort alert name	#	%	#	%
1091	WEB-MISC ICQ Webfront HTTP DOS	87	0,18%	6	0,01%
2002	WEB-PHP remote include path	50	0,10%	231	0,47%
2229	WEB-PHP viewtopic.php access	5169	10,42%	1580	3,20%
1012	WEB-IIS fpcount attempt	3	0,01%	10	0,02%
1256	WEB-IIS CodeRed v2 root.exe access	2	0,004%	3	0,01%
1497	WEB-MISC cross site scripting attempt	5602	11,30%	7347	14,90%
2436	WEB-CLIENT Microsoft wmf metafile access	145	0,29%	53	0,11%
1831	WEB-MISC jigsaw dos attempt	659	1,33%	153	0,31%
1054	WEB-MISC weblogic/tomcat .jsp view source...	3412	6,88 %	3885	7,88%

Table 1: The distributions of the training and testing data sets

Among the severe attacks detected by Snort, we selected 9 Web-based severe attacks to predict on the basis of the alerts that often precede/prepare these severe attacks. All these attacks are associated with a high severity level and are targeting either Web servers or related web-based applications. Such attacks may result in arbitrary code execution and full control of the targeted system. Interested readers can refer to Snort signature database for additional information and references on these attacks.
The first month (used to build the training data set) contains 333566 IDMEF alerts while the second month (used to build the testing set) is composed of 288425 alerts. In the training set, there are 23454 alerts with high severity level, 86679 with medium severity level and 223433 with low severity level alerts (similar proportions of low, medium and high level alerts are found in the testing set).

Let us now focus on the redundant/irrelevant elimination task performed by the preprocessing tool. Recall that the redundant alerts are eliminated by representing all the occurrences of an alert $Alert_i$ within the same time window by a unique variable telling whether this alert has been reported or not.
In order to eliminate the irrelevant alerts for our prediction task, we first extracted all the existing alerts involving the same attackers and victims as the severe attacks then using the information gain [36] selection feature procedure, we selected a subset of relevant features. Then among more than two hundreds of low/medium candidate alerts composing our training and testing sets, we selected only 27 Snort alerts whose *sid* are 2, 3, 4, 7, 15, 16, 18, 839, 853, 882, 895, 1013, 1112, 1141, 1142, 1147, 1214, 1288, 1301, 1478, 1767, 1852, 2142, 2280, 2286, 2565 and 2566. The remaining input alerts are not relevant for predicting the selected 9 Web-based attacks. Note that the feature selection task does not require any expert knowledge while it allows to train a model on a large data set by reducing the feature space. Hence, each sequence of raw IDMEF alerts will be preprocessed into an alert window of 27 variables. Hence, using a 2 hours alert window, the 333566 IDMEF (resp. 223433) alerts composing the training (resp. testing) set are preprocessed into 48409 (resp. 49301) alert window vectors where each vector is composed of 27 variables. Note that the feature extraction/labeling process is similar to the works of [3][7].

In the following, we report our experimental results where *Experimentation 1* compares the severe attack prediction model using a Bayesian multi-net with a C4.5 decision tree [28], a naive Bayes classifier [9] and a Bayesian network built with the MWST [11] algorithm. In *Experimentation 2* we provide our results on handling the IDS' reliability while *Experimentation 3* provides our experimental results on controlling the prediction/false alarm rate tradeoffs. In all these experimentations, the models are built (resp. evaluated) on the same training data set of Table 1 (resp. testing data set).

7.3 *Experimentation 1:* Severe attack prediction using Bayesian multi-nets

In order to evaluate the effectiveness of our multi-net classifier, we compare it with a C4.5 decision tree, a naive Bayes classifier and a network classifier built using MWST algorithm [11] which is a structure learning algorithm using the mutual information measure that rapidly builds simple and efficient tree structures [17]. As for the multi-net, we also used the MWST algorithm to build the networks representing each class of alert sequences. Note that the C4.5 is among the most efficient classifiers in the literature and it is capable to handle both categorical and numerical features. As for the naive Bayes classifier, it is the simplest form of Bayesian network classifiers based on the strong assumption that the attribute features are independent in the context of the class node. In spite of this simplifying assumption, the performances of this model are very competitive on problems where there is enough training data. We trained the four classifiers on the same training set and evaluated them on the same testing

set of Table 1. The results of this experimentation are given in Table 2.

Sid	Snort alert name	C4.5	Naive Bayes	MWST	Multi-net
1091	WEB-MISC ICQ Webfront HTTP DOS	0%	0%	0%	0%
2002	WEB-PHP remote include path	26,41%	26,41%	26,84%	26,84%
2229	WEB-PHP viewtopic.php access	72,97%	74,81%	74,94%	72,15%
1012	WEB-IIS fpcount attempt	0%	10%	0%	0%
1256	WEB-IIS CodeRed v2 root.exe access	0%	0%	0%	0%
1497	WEB-MISC cross site scripting attempt	92,02%	95,62%	95,62%	95,92%
2436	WEB-CLIENT Microsoft wmf metafile..	56,60%	60,38%	60,38%	56,60%
1831	WEB-MISC jigsaw dos attempt	50,3%	54,90%	54,25%	46,41%
1054	WEB-MISC weblogic/tomcat .jsp view..	38,74%	41,36%	38,69%	47,77%
	Prediction rate	72,26%	75,32%	74,53%	**76,92%**
	False alarm rate	1,66%	3,21%	3,10%	**1,58%**

Table 2: Experimental results of C4.5, naive Bayes, MWST and multi-net classifiers on data sets of Table 1

Table 2 compares the results of the C4.5 decision tree, naive Bayes classifier, MWST classifier and our multi-net classifier with respect to their prediction rate and the false alarm rate. The results of Table 2 show that the multi-net outperforms both the decision tree, naive Bayes and MWST classifiers regarding the overall prediction and the false alert rates. In particular, the multi-net classifier predicted 76,92% of the severe attacks at a false alarm rate of 1,58% (29 false alarms/day) while the naive Bayes (resp. MWST) classifier triggered 3.21% (58 false alarms/day) (resp. 3,10% (56 false alarms/day)). This performance is due to the better modeling of each class leading to better estimation of the likelihoods of the alert sequences to analyze. It is important to note that the three classifiers failed to predict some severe attacks mainly because of the imbalance of class frequencies in the training set. This is for instance the reason why the severe attack with *Sid*=1256 (*WEB-IIS CodeRed v2 root.exe access*) which is represented in the training set only by 2 instances was not predicted. The C4.5 decision tree achieved the second best false alarm rate (1.66%) but its prediction rate is the worst (only $72,26\%$). Finally, note that the naive and MWST classifiers achieved comparable prediction rates but suffer from high false alarm rates which is a crucial issue when using IDSs.

7.4 *Experimentation 2:* Handling IDSs' reliability in Bayesian multi-nets

In this experimentation, we are interested in the effect of taking into account the IDSs' reliability of the prediction/false alarm rate tradeoffs. We implemented the virtual evidence method as follows:

- For each alert A_i used as a predictor, we first checked in Snort's database whether the rule associated with this attack is known to produce false positives. In the positive case, we computed on a representative corpus of the training data set the proportion of alerts A_i which actually correspond to true attacks. Namely, we computed two parameters $p(A_i{=}1/Attack{=}True)$ and $p(A_i{=}1/Attack{=}False)$. Note that taking account false negatives is

in our case impossible because we have not the original network traffic in order to check whether there are attacks which were not detected by Snort.

- When an alert sequence is submitted for analysis, the prediction is performed on the Bayesian network where the alert variables A_i are augmented by virtual evidence nodes (observed variables) R_i to handle the reliability of inputs.

In order to evaluate the effectiveness of handling IDSs' reliability in Bayesian multi-nets, we compare it with a standard Bayesian network-based classifier built using MWST as in *Experimentation 1*. Table 3 gives the results of handling the reliability of Snort IDS producing the alert sequences we analyze.

Sid	Snort alert name	Multinet	VE-Multinet
1091	WEB-MISC ICQ Webfront HTTP DOS	0%	0%
2002	WEB-PHP remote include path	26,84%	25,97%
2229	WEB-PHP viewtopic.php access	72,15%	74,30%
1012	WEB-IIS fpcount attempt	0%	0%
1256	WEB-IIS CodeRed v2 root.exe access	0%	0%
1497	WEB-MISC cross site scripting attempt	95,62%	93,32%
2436	WEB-CLIENT Microsoft wmf metafile access	56,60%	56,60%
1831	WEB-MISC jigsaw dos attempt	56,41%	37,25%
1054	WEB-MISC weblogic/tomcat .jsp view source...	47,77%	41,83%
	Prediction rate	**76,92%**	73,88%
	False alarm rate	1,58%	**0,74%**

Table 3: Experimental results of Multinet and VE-Multinet prediction models

The results of Table 3 show that the *VE-Multinet* classifier (the multi-net model implementing the virtual evidence method for handling the reliability of Snort IDS) achieves comparable prediction rates with respect to the standard multi-net classifier *Multinet* but significantly reduces the false alarm rate down to **0,74%** (the false alarm rate was decreased from 29 down to only 13 false alarms/day). Note that this result is achieved by handling the true/false positive reliability relative to only three alerts (those having sid=882, sid=1288 and sid=1852) constituting the majority of false alerts triggered by Snort in our data sets (see [32] for an analysis of these false alarms triggered by Snort). Such results are very promising but require a rigorous reliability assessment and handling false negatives which are very time consuming tasks. Indeed, in order to efficiently use our approach, one has to rigorously assess both the true/false positive and negative rates which is a very time consuming tasks. More specifically, in order to assess the true/false positive rates, one has to check for each alert A_i the proportion of A_i instances which actually correspond to real attacks. In order to assess the true/false negative rates, all the network traffic should be analyzed in order to evaluate the proportion of attacks that were not detected by the IDSs. Clearly, assessing the true/false positive and negative rates are very complicated and time consuming tasks. Moreover, there is a need to reevaluate them more frequently in order to take into account new menaces and attacks, etc.

7.5 *Experimentation 3:* Controlling prediction/false alarm rate tradeoffs

In *Experimentation 1* and *Experimentation 2*, we showed that the Bayesian multi-net prediction model offers the best prediction/false alarm rate and naturally allows to handle the IDSs' reliability improving the model's performances. However, in real situations it is important to have means of configuring the prediction model so as not to exceed a given false alarm rate. In *Experimentation 3*, we provide our results on using the ambiguity reject for controlling the attack prediction/false alarm rate tradeoffs. Note that we defined different confidence levels L and we used the same Bayesian multi-net classifier as in *Experimentation 1*.

Sid	Snort alert name	Multi-net	L=1/5	L=1/3
1091	WEB-MISC ICQ Webfront HTTP DOS	0%	0%	0%
2002	WEB-PHP remote include path	25,97%	15,02%	15,02%
2229	WEB-PHP viewtopic.php access	74,30%	74,19%	73,89%
1012	WEB-IIS fpcount attempt	0%	0%	0%
1256	WEB-IIS CodeRed v2 root.exe access	0%	0%	0%
1497	WEB-MISC cross site scripting attempt	93,32%	93,32%	93,32%
2436	WEB-CLIENT Microsoft wmf metafile access	56,60%	37,50%	37,50%
1831	WEB-MISC jigsaw dos attempt	37,25%	25,97%	25,45%
1054	WEB-MISC weblogic/tomcat .jsp view source...	41,83%	39,60%	39,60%
	Prediction rate	73,88%	65,39%	65,21%
	False alarm rate	0,74%	0,68%	0,65%

Table 4: Experimental evaluation of controlling the attack prediction/false alarm rate tradeoffs

Table 4 provides detailed results on the effect of using the reject option in order to control the attack prediction/false alarm rate tradeoffs. As expected, the false alarm rate decreases proportionally to the value of the confidence level L. However, the prediction rates of some severe attacks also decrease. Figure 7 gives the variation of the prediction rate ($FPRate$) and the rejection rate ($RRate$) at different confidence levels L. The rejection rate gives the proportion of alert sequences that were rejected by our severe attack predictor.

The results of Figure 7 show that the $TPRate$ decreases slightly as we augment the value of the confidence level L while the rejection rate $RRate$ increases significantly. Indeed, Figure 7 shows that it is possible to achieve a very high severe attack prediction rate while rejecting only a small amount of the analyzed alert sequences (see $TPRate$ and $RRate$ when L=.33). However, when L is set to .9 (to force the model to take decisions only when it is very confident) the proportion of alert sequences which are rejected attains 31,92%. As for the evolution of the false alarm rate ($FPRate$) at the different reject rates, Figure 8 gives the $FPRate$ of the same experimentation of Figure 7.

Figure 8 shows that the false alarm rate can be controlled by setting the appropriate value for the reject rate. **The results of this figure can guide the user to fix the threshold (for the reject option) according to the acceptable false alarm rate he wants not to exceed.** Clearly, our approach offers an efficient, flexible and configurable model for predicting severe

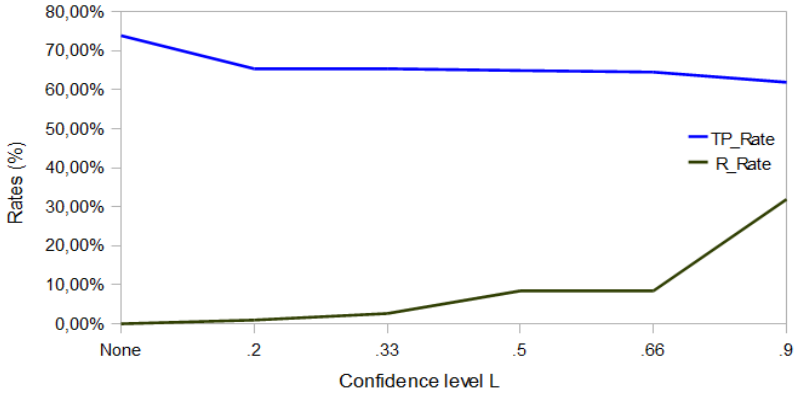


Figure 7: $TPRate$ and $RRate$ variation at different levels of L

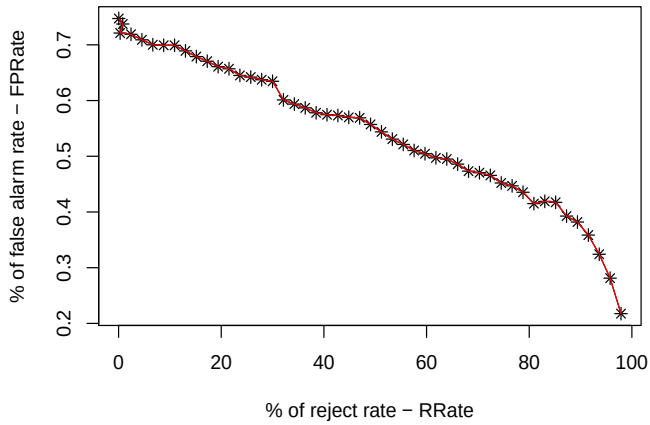


Figure 8: $FPRate$ variation at different reject rates $RRate$

attacks. Moreover, our approach requires minimum expert knowledge and the computational complexity of handling IDSs' reliability and implementing the reject option is nearly the same as the standard classification based on Bayesian networks.

Because of the class imbalance in our testing data set and the difference in misclassification costs, the evaluation of our prediction model based only on the prediction rate ($TPRate$) is not sufficient. Indeed, our testing data set is domi-

nated by alerts sequences which mostly are not followed by severe attacks while the misclassification cost of a false alarm and the cost of a missed attack (false negative) are clearly not equivalent. Therefore, additional experimentations are carried out in order to draw the ROC curve [7] of our prediction model. A ROC curve [16] allows to visualize the fluctuations existing between the True Positive Rate (in our case, the true prediction rate $TPRate$) and the corresponding false positive rate (denoted in this paper $FPRate$) which are the two most important measurements of IDSs performance. More precisely, a ROC curve is a two-dimensional graph where the $TPRate$ is plotted on the Y-axis while the $FPRate$ is plotted on the X-axis. Each couple $TPRate$ and its corresponding $FPRate$ is represented by a point in the ROC curve. Note that in order to draw the ROC curves evaluating our severe attack prediction model, we used the method proposed in [16] and sorted testing data instances after computing for each testing alert sequence a score measuring how much the instance in hand is not likely a severe attack. This score is the a posteriori probability of not being a severe attack.

Figure 9 gives the ROC curves of our prediction model evaluated on the testing data of Table 1.

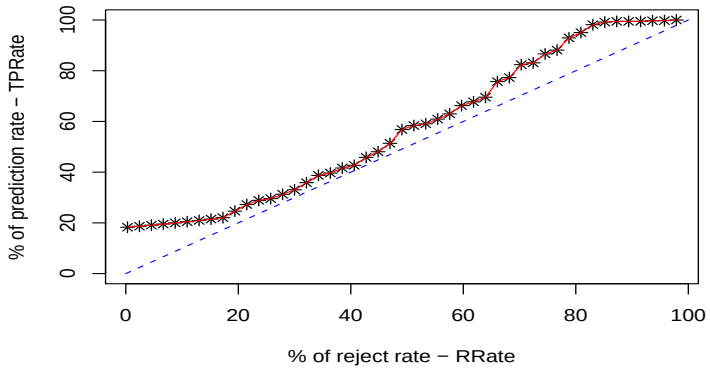


Figure 9: ROC curve of the prediction model ($TPRate$) with different reject rates ($RRate$)

The graph of Figure 9 shows the variation of our model prediction rate with respect to different reject rates. This ROC curve shows that our reject option improves the prediction model and allows an expert to know which rejection

[7] A Receiver Operating Characteristics (ROC) curve is a technique originally used in the signal detection theory in order to describe the relationships between the capacity of detecting a signal and the underlying noise. For a detailed tutorial on ROC curves for machine learning and data mining techniques, see [16].

rate will correspond to each prediction rate he may want to guarantee. Clearly, Figure 9 shows that our prediction model exploiting the reject option is more effective than the same model without using the prediction/false alarm tradeoffs mechanism. The experimental results provided in this section clearly show the effectiveness of our prediction model for predicting severe attacks and controlling the prediction/false alarm rate tradeoffs.

8 Conclusions

This paper addressed crucial issues in the field of alert correlation consisting in efficient Bayesian network modeling/reasoning for handling IDSs' reliability and controlling the prediction/false alarm rate tradeoffs. More specifically, we proposed a method allowing to cope with the huge amount of alerts daily triggered by IDSs. In addition to redundant/irrelevant alert elimination, this method also allows to cope with the class imbalance problem by encoding the local correlations leading to better likelihood estimation and prediction performance. Then we proposed to take into account the reliability of IDSs' as it is a relevant information on the inputs used by the alert correlation engines. Finally, in order to better control the prediction/false alarm rate tradeoffs, we proposed an approach based on classification with reject option allowing to reject alert sequences where the prediction model has not enough confidence to make good predictions. Handling IDSs' reliability and implementing the reject option are naturally and easily implemented using Bayesian network-based classifiers. Our experimental results are very promising especially when one appropriately assesses the reliability of the IDSs and the confidence levels.

As future directions, it is very interesting to take into account and exploit the IDSs' reliability not only during the prediction phase, but also when building the prediction models from empirical data. Indeed, while reasoning with unreliable information (data provided by unreliable sources) has received much interest, all the approaches for learning probabilistic graphical models (and other prediction models) implicitly assume that training data is cleaned and ignore the reliability of sources even if such information is available. Our objective is to propose heuristics for learning probabilistic graphical models talking into account the available information on the IDSs' reliability. We also plan to carry out supplementary experimental studies in order to detect some recent attack scenarios such as the *Conficker* worm propagation which infected thousands of computers since November 2008.

Acknowledgements

This work is supported by the (ANR) SETIN 2006 PLACID project (Probabilistic graphical models and Logics for Alarm Correlation in Intrusion Detection `http://placid.insa-rouen.fr/`)

References

[1] S. O. Al-Mamory and H. Zhang. Ids alerts correlation using grammar-based approach. *Journal in Computer Virology*, 5(4):271–282, 2009.

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

[3] S. Benferhat, T. Kenaza, and A. Mokhtari. False alert filtering and detection of high severe alerts using naive bayes. In *Computer Security Conference(CSC'08)*, South Carolina, apr 2008.

[4] S. Benferhat, T. Kenaza, and A. Mokhtari. Tree-augmented naive bayes for alert correlation. In *3rd conference on Advances in Computer Security and Forensics(ACSF'08)*, pages 45–52, jul 2008.

[5] S. Benferhat and K. Sedki. Alert correlation based on a logical handling of administrator preferences and knowledge. In *International Conference on Security and Cryptography(SECRYPT'08)*, pages 50–56, Porto, Portugal, jul 2008.

[6] S. Benferhat and K. Tabia. Novel and anomalous behavior detection using bayesian network classifiers. In *Proceedings of the International Conference on Security and Cryptography*, pages 13–20, Porto, Portugal, 2008. INSTICC Press.

[7] Z. Bin and A. Ghorbani. Alert correlation for extracting attack strategies. *I. J. Network Security*, 3(3):244–258, 2006.

[8] A. Cano, J. G. Castellano, A. R. Masegosa, and S. Moral. Methods to determine the branching attribute in bayesian multinets classifiers. In *8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'05*, pages 932–943, 2005.

[9] J. Cheng and R. Greiner. Learning bayesian belief network classifiers: Algorithms and system. In *14th Conference of the Canadian Society on Computational Studies of Intelligence*, pages 141–151, London, UK, 2001. Springer-Verlag.

[10] C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, Jan 1970.

[11] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968.

[12] F. Cuppens and A. Miège. Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Security and Privacy*, pages 187–200, USA, 2002.

[13] O. Dain and R. K. Cunningham. Fusing a heterogeneous alert stream into scenarios. In *In Proceedings of the 2001 ACM workshop on Data Mining for Security Applications*, pages 1–13, 2001.

[14] H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *Recent Advances in Intrusion Detection*, pages 85–103, London, UK, 2001. Springer.

[15] A. Faour and P. Leray. A som and bayesian network architecture for alert filtering in network intrusion detection systems. In *RTS - Conference on Real-Time and Embedded Systems*, pages 1161–1166, 2006.

[16] T. Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Laboratories, Palo Alto, CA, USA, 2003.

[17] O. Francois and P. Leray. Evaluation d'algorithmes d'apprentissage de structure pour les réseaux bayésiens. In *Proceedings of 14eme Congrès Francophone Reconnaissance des Formes et Intelligence Artificielle*, pages 1453–1460, Toulouse, France, 2004.

[18] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth. Bayesian network classifiers. *Machine Learning*, pages 131–163, 1997.

[19] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artif. Intell.*, 82(1-2):45–74, 1996.

[20] F. V. Jensen and Thomas D. Nielsen. *Bayesian Networks and Decision Graphs (Information Science and Statistics)*. Springer, June 2007.

[21] K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *Eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 366–375, New York, NY, USA, 2002. ACM.

[22] P. Leray, H. Zaragoza, and F. d'Alch-Buc. Pertinence des mesures de confiance en classification. In *12eme Congres Francophone AFRIF-AFIA Reconnaissance des Formes et Intelligence Articifielle (RFIA 2000)*, pages 267–276, Paris, France, 2000.

[23] W. Lingyu, L. Anyi, and J. Sushil. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Comput. Commun.*, 29(15):2917–2933, 2006.

[24] Michael G. Madden. Evaluation of the performance of the markov blanket bayesian classifier algorithm. *CoRR*, cs.LG/0211003, 2002. informal publication.

[25] P. Ning, Y. Cui, and D. S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *9th ACM conference on Computer and communications security*, pages 245–254, NY, USA, 2002. ACM.

[26] A. Patcha and J. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470, 2007.

[27] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[28] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[29] M. Roesch. Snort - lightweight intrusion detection for networks. pages 229–238, 1999.

[30] R. Smith, N. Japkowicz, M. Dondo, and P. Mason. Using unsupervised learning for network alert correlation. In *21st conference on Advances in artificial intelligence*, pages 308–319, Berlin, Heidelberg, 2008. Springer-Verlag.

[31] S. Staniford, J. A. Hoagland, and J. M. McAlerney. Practical automated detection of stealthy portscans. *J. Comput. Secur.*, 10(1-2):105–136, 2002.

[32] G. C. Tjhai, M. Papadaki, S. Furnell, and N. L. Clarke. Investigating the problem of ids false alarms: An experimental study using snort. In *23rd International Information Security Conference SEC 2008*, pages 253–267, 2008.

[33] E. Tombini, H. Debar, L. Mé, and M. Ducassé. A Serial Combination of Anomaly and Misuse IDSes Applied to HTTP Traffic. In *Annual Computer Security Applications Conference 2004*, page 10 pages, Beijing Chine, 12 2004.

[34] A. Valdes and K. Skinner. Adaptive, model-based monitoring for cyber attack detection. In *Recent Advances in Intrusion Detection*, pages 80–92, 2000.

[35] A. Valdes and K. Skinner. Probabilistic alert correlation. In *Recent Advances in Intrusion Detection*, pages 54–68, London, UK, 2001. Springer-Verlag.

[36] M Verleysen, F. Rossi, and D. François. Advances in Feature Selection with Mutual Information. In M.; Hammer B.; Verleysen M. Villmann, Th.; Biehl, editor, *Similarity-Based Clustering*, Lecture Notes in Computer Science, pages 52–69. Springer Berlin / Heidelberg, 05 2009.

[37] T. Wojciech. Anomaly-based intrusion detection using bayesian networks. *depcos-relcomex*, 0:211–218, 2008.