

Data Mining and Detecting Complex Attacks

—
Délivrable n°15

Salem BENFERHAT
CRIL

Tayeb KENZA
CRIL et Ecole Militaire Polytechnique d'Alger

Philippe LERAY
LITIS / LINA

Data Mining and Detecting Complex Attacks

Salem Benferhat

CRIL (CNRS-UMR 8188)

Université d'Artois, rue Jean Souvraz

SP 18 F 62307 Lens Cedex

benferhat@cril.univ-artois.fr

Tayeb Kenaza

Ecole Militaire Polytechnique

BP 17 Bordj El-Bahri

16111 Alger

ken.tayeb@gmail.com

Philippe Leray

LINA (CNRS-UMR 6241)

Site de PolytechNantes, rue Christian Pauc

BP 50609 F-44306 Nantes Cedex

philippe.Leray@univ-nantes.fr

Abstract—Existing alert correlation either require a large amount of expert knowledge or use simple similarity measures which are not enough to detect complex attacks. In this paper, we propose a new modeling of alert correlation problems based on naive Bayes. Our modeling only needs a small part of expert knowledge. It takes advantage of available historical data to provide efficient algorithms to predicting coordinated attacks. In fact, our approach does not need to make explicit scenarios, nor the set of actions involved in scenarios. We have evaluated our approach using an experimentation, which is done on a real dataset. The experimentation shows how to detect severe attacks and how to exhibit alerts (with low severity) that contribute to perform them. Moreover, a comparison between our approach and CART is provided with the experimentation.

Keywords: intrusion detection, alert correlation, attack prediction, naive Bayes, CART.

I. INTRODUCTION

Intrusion Detection Systems (IDSs) are usually considered to be a second line of defense to protect against malicious activities. They have the function of analyzing events on monitored systems and of reporting alerts when suspicious events occur. Current IDS usually focus on low-level attacks or anomalies. They process alerts individually without considering logical connections that may exist between them. The output of IDS is then a set of alerts reporting elementary attacks.

In some situations, intruders may use complex attacks to achieve their objectives. Often, they perform series of actions (elementary attacks) in a predefined sequence, called “scenario” or “attack plan”. Most of these actions are reported by IDS but the logical relationships between these actions are not detected by standard tools. In fact, network administrators are usually constrained to analyze and to detect attack plans manually using their own knowledge. Accordingly, they are often overwhelmed by an important volume of alerts, with some uncertainty whether these alerts correspond or not to real attacks, and whether they correspond to isolated actions or belong to some complex scenarios. In such situations, the goal of alert correlation is to provide a correlation tool.

Many alert correlation approaches have been proposed in the literature but most of these approaches require a large amount of expert knowledge (such as a more or less explicit representation of scenarios) or fail to detect coordinated attacks. In [1], we have proposed a new modeling of alert correlation that does not require a large amount of knowledge. It is based on

a very simple form of Bayesian networks [2], [3] called naive Bayes [4]. Bayesian networks have been used for intrusion detection and for alert correlation. However, as it will be detailed in related works section, our approach has several important advantages with respect to existing works, especially regarding the amount of explicit expert knowledge needed for handling correlation problems.

We have illustrated our approach on the well know DARPA'2000 dataset, and we have easily detected the scenario. However, the dataset is simulated and it's far from the real word. In this paper, we present an in depth experimentation in order to evaluated our modeling in a real environment. The experimentation is done on a real dataset, and has as goal to shows how to detect severe attacks and how to exhibit alerts (with low severity) that contribute to perform them. A comparison between our approach and CART is also provided with this experimentation.

The rest of this paper is organized as follows. Section 2 presents a refresh on our approach. Section 3 presents the experimentation. Section 4 compares our approach with respect to existing works.

II. A REFRESH ON OUR APPROACH TO DETECTING COORDINATED ATTACKS

The purpose of our approach is to learn, from observation history, relationships between alerts that contribute to achieve intrusion objectives under the form of scenarios. We have proposed to use Bayesian networks to predict intrusion objectives. More precisely, we use naive Bayes to encode the influence of each action on intrusion objectives by computing conditional probability distributions from observation history. Once probability distributions on different nodes of naive Bayes are computed, this model can be used to predict whether an intrusion objective may occur or not, according to a partial observation of actions.

Our approach does not need to determine preconditions and postconditions associated with actions. It allows directly predicting the most plausible intrusion objectives using available observation history. In fact, we are not interested in determining the exact order in which a set of actions has been executed in order to achieve a given intrusion objective. We are more interested to determining which actions that may be involved (whatever is the order) in intrusion objectives, and a

tool that predicts, on-line, whether an intrusion objective may be compromised or not.

Our approach includes tree main steps :

1) *Data preprocessing*: to prepare the training of naive Bayes representing intrusion scenarios, we need to apply some preprocessing on observed events. Data contain a set of alerts that reports executed actions and information about intrusion objectives (whether they have been compromised or not). These actions are candidates for being variables of Bayesian networks, if there are relevant to intrusion objectives.

We first gather all observed objectives into a single class called “Objectives-Intrusion” and we assign to each objective a number from 1 to N , where N represents the number of possible objectives. We add to this class the number zero to represent the normal traffic. Then, we sort chronologically observed alerts and we split them into subgroups, according to a given time window (W_1, W_2, W_3 , etc.)(figure 1), determined experimentally (from few minutes to two hours). These windows depend on intrusion objectives and represent the time required to achieve them. They are crucial to determine the set of actions involved in scenarios.

If an objective is observed inside a window, we move this window to the left until it ends on this objective (figure 3). We do this in order to ensure that all actions involved in each intrusion objective are present in a single window. Proceeding in this way produce an overlapping with the previous window. Hence, some actions may be considered in two windows simultaneously. For example, in figure 2, action 4 will be considered on both windows W_1 and W_2 . Window W_1 contains a normal data while W_2 contains an attack plan (since at the end of W_2 an objective is compromised). So according to the observation frequency of action 4 on normal or abnormal traffic, we can determine if action 4 is suspicious or not. Lastly, we label each subgroup by a number corresponding to the observed intrusion objective.

	$Action_1$	$Action_2$...	$Action_N$	Objectives
W_1	true	false	...	true	0
W_2	false	false	...	true	1
W_3	false	false	...	true	0
W_4	false	false	...	true	0
...

TABLE I
PREPROCESSED DATA

Table 1 shows a sample preprocessed data. The value “true” means that the action/objective has been observed on the corresponding window.

Note that observations concern all hosts on the monitored network. Hence we have to apply the data preprocessing procedure described below for each host individually and merge results in a single table. The data preprocessing procedure is summarized in Algorithm 1.

Algorithm 1: Data preprocessing

Data: Observation history

Result: Table of vectors;

begin

 Gather all intrusion objectives into a single class called “Objectives-Intrusion”;

 Assign to each objective a number from 1 to N (add 0 to represent no objective);

for each host do

 Sort observed actions chronologically;

 Split observed actions into groups, according to a given window of time;

if an objective is observed inside a window then

 Move this window to the left until it ends on this objective;

 Label each subgroup by the number corresponding to the observed objective;

 Merge vectors in a single table;

end

2) *Construction of naive Bayes*: we construct one naive Bayes for each objective. The reason is that intrusion objectives are not exclusive. It may happen that two different intrusion objectives O_1 and O_2 to be simultaneously compromised, namely $P(O_1) = P(O_2) = 1$. By defining one naive Bayes per intrusion objective, it is possible to represent such situation. However, if only one naive Bayes is used and if there are N objectives which are compromised, we cannot represent such situation because the probability will be divided over several objectives. Consequently, it will produce a wrong probability of each objective, namely $P(O_i) = \frac{1}{N}$.

Now, on the basis of this observation, we need to modify slightly Table 1, by splitting it in several tables. Each of them concerns only a same intrusion objective. More precisely, for each objective, we replace its number in the column “Objectives” by “true” and the other intrusion objectives by “false”. Thus, we obtain a table for each intrusion objective (Table 2).

	$Action_1$	$Action_2$...	$Action_N$	O_1
W_1	true	false	...	true	false
W_2	false	false	...	true	true
W_3	false	false	...	true	false
W_4	false	false	...	true	false
...

TABLE II
PREPROCESSED DATA FOR INTRUSION OBJECTIVE O_1

Data allow us to estimate parameters. This can be done by a simple frequency calculation. However, when an attribute value does not occur together with a given class value, it produces a zero estimate for $P(A|C)$. This is problematic, since it will produce a wrong probabilities when they are multiplied. To overcome this problem, we use the Laplace estimator. Given a predefined factor f , if there are N matches of n instance for a K value problem, Laplace estimates the probability as $(N + f)/(n + kf)$. For two valued problem

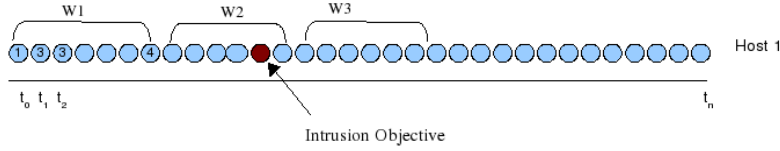


Fig. 1. Data preprocessing (a)

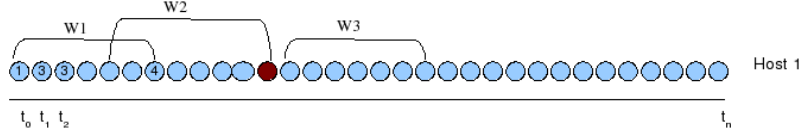


Fig. 2. Data preprocessing (b)

with $f = 1$, we obtain the well-known Laplace of succession $(N + 1)/(n + 2)$ [5].

Namely, once observations (alerts) are obtained and formatted as in Table 2, we have to compute the probability distribution for each variable using Laplace estimator. The naive Bayes construction procedure is summarized in Algorithm 2.

Algorithm 2: Naive Bayes construction

```
Data: Table of vectors
Result: Naive Bayes;
begin
  for each intrusion objective do
    Replace its number in table of vectors by "true" and others
    by "false";
    Compute parameters for the corresponding naive Bayes ;
  end
end
```

3) *Intrusion objective prediction*: The goal of prediction (inference) is to estimate the values of hidden nodes, given the values of observed ones. In our context, the prediction consist of computing the new probability of intrusion objectives given some observed actions.

During the detection phase, we first define some timeout and we wait for reported alerts. Then, each new alert will induce evidence setting on each naive Bayes. According to the influence of this action on intrusion objectives, the probability of each objective will increase or decrease. After evidence setting, we verify the new probability. If it exceeds the threshold, an alarm is generated or we proceed to an appropriate countermeasure. Otherwise, we wait for the next alert. After timeout, if no probability exceeds the threshold, we can confirm that no attack plan is running. Then, we clear all evidences and we restart the detection phase.

The timeout is equal to the time window used in the preprocessing step. Note that in the detection phase, the prediction is done using several overlapped timeout, which are separated with some time offset. We do this because we cannot determine when an attack scenario begins. Using a single timeout, an intrusion objective can be spread on both

sides of the timeout. The prediction procedure is summarized in Algorithm 3.

Algorithm 3: Objective prediction

```
Data: Observed actions
Result: Prediction of intrusion objectives ;
Define timeout;
Define timeoffset;
procedure [prediction] ;
begin
  while (true) do
    t = timeout;
    while (t) do
      if an action A is observed then
        for (Objective O = O1 to On) do
          if Influence(A,O) = Positive then
            Focus in this intrusion objective;
          if Influence(A,O) = Critical then
            The objective O is highly expected;
        end
      end
    end
    N = Integer(timeout/timeoffset);
    for (th = 1 to N) do
      wait(timeoffset);
      create a thread and call the prediction procedure;
    end
  end
end
```

III. EXPERIMENTATION

In this section we evaluate our approach with a real dataset¹, which contains an important volume of alerts reported during three months of network monitoring. In the first subsection, we explain how our approach can be used to monitor and predict severe attacks. In the second subsection, we describe the dataset. In the last subsection we present the results of the experimentation, and we compare results to to a commercial data mining tool.

¹Thanks to French national project PLACID for providing us such data.

A. Attacks severity

IDS can assign a severity to an action representing its impact on systems. Some actions may just collect information about systems such as Probe actions. Other actions may alter systems with different severity levels (usually low, medium and high level). Actions with low and medium severity alter systems without really compromising them. However, in presence of severe actions there is a high probability that systems can be compromised. In the following, we define a severity class as:

$$Severity = \{low, medium, high\} \quad (1)$$

In some applications, severe attacks are not isolated and may be prepared by attacks with low severity. These attacks (with low severity) can be viewed as actions that are needed to be executed before achieving severe attacks.

Using our approach, we aim to provide a mechanism that help security operators filter alerts by differentiating between low and high-severity alerts. The proposed mechanism will allow security operators to focus on severe attacks and on the subset of low-severity alerts that contribute to perform them. In fact, what plays a role of intrusion objective is severe attacks and what plays a role of actions are attacks with low and medium severity.

B. Dataset description

the dataset contains more than one million alerts that report observation of more than two hundreds types of attacks. This important volume of alerts confirms the problem of alerts flooding, which make security operators incapable to inspect each alert. Most of reported alerts does not represent a real threat for monitored systems (attack with low severity), but security operators can not simply analysis severe attacks and ignore the rest. Our aim is to show that some severe attacks may be prepared by attacks with low severity. Then, we will use this property to reduce the volume of alerts and present to security operators a reasonable number of alert that reports the real relevant attack set. Namely, we present to the security operator not only severe attacks but also attacks with low or medium severities that contribute to the realization of severe attacks.

The dataset contains 62.95%, 31.07% and 5.98% of action with low, medium and high severity, respectively. It will be splitted into two parts. The first part contains two months of network traffic and will be used for training phase. The second part contains one month of network traffic and will be used for test phase. Training dataset contains 23 severe attacks and more than 400 monitored hosts. Many hosts are only concerned by a few severe attacks and some hosts are not concerned by any severe attack. After preprocessing training dataset, with two hours as a window time (section 4.1), we have obtained 23 naive Bayes (a naive Bayes per severe attack).

C. Evaluation

in this section, we present the results of detecting severe attacks contained in the test dataset using different learned naive

Bayes. This consists on applying the classification mechanism of naive Bayes. Like training dataset, the test dataset is first preprocessed and evaluated using naive Bayes to construct a set of windows. In fact, we format test dataset for each naive Bayes separately. For a given naive Bayes (naive Bayes constructed for some severe attack), each window may contain the severe attack, as it may contain a normal traffic. Then, we classify each window by setting all observed actions inside this window in the naive Bayes and we infer the new probability of the severe attack. According to the new observed probability we decide if the window should contain the severe attack or not ($P(class) > threshold$). And according to the real content of window, we distinguish 4 situations represented under a matrix of confusion. This matrix contains 4 cells, where cells (1,1), (2,2), (1,2) and (2,1) correspond to true positive (TP), true negative (TN), false positive (FP) and false negative (FN), respectively. The evaluation of the classification efficiency is based on the *Percent of Correct Classification* (PCC) which can be computed as following:

$$PCC = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

Table 3 (left table) shows the classification efficiency of our approach. It shows that most of naive Bayes have classified correctly the severe attack according to observed actions. This confirms the assumption that severe attacks may be influenced by low and medium observed actions. Another way to show the classification efficiency of our approach is to group the whole classified windows into two classes corresponding to normal and abnormal traffic, where abnormal traffic corresponds to windows classified containing a severe attack and normal traffic correspond to windows classified containing a normal traffic. Tables 4 (left table) present the confusion matrix and the classification efficiency of our approach.

Table 4 (left table) shows 96.86% of PCC which represents good classification efficiency for our approach. Moreover, our approach presents a reasonable rate of false negative and false positive (13.19% and 2.92% respectively).

Let's now compare results obtained with our approach to an other classification approach. We have opted to compare our approach to the commercial tool CART². We have tested CART with the same dataset and the PCC of detecting each severe attack is presented in table 3 (right table). Results show that our approach presents a better PCC for the majority of severe attacks. In general, there are a significant difference between our approach and CART, more than 5% (e.g attacks 4,5,8,9,11,13, etc). However, CART is some time slightly better (e.g attacks 1,2 and 6) with a difference less than 5%.

Table 4 present a global comparison between our approach and CART. The global PCC confirm that our approach is better than CART with 7% of difference. However, in term of false negative and false positive, our approach miss much more attacks than CART (13.19% against 7.06%) but it generate a few false positive than CART (2.92% against 10.76%). Note that our approach is parametrized with a threshold equal to 50%, which is not necessarily the better compromise between false positive and false negative. It's possible to decrease the

²CART is a commercial Data mining tool based on decision tree

	Attack ID (snort)	PCC
1	1002	98.95%
2	1091	98.35%
3	1243	97.5%
4	1250	98.49%
5	1256	97.38%
6	1391	97.96%
7	1497	89.22%
8	1807	96.26%
9	2002	93.4%
10	2123	100%
11	2183	94.15%
12	2229	98.03%
13	2329	100%
14	2436	85.17%
15	2518	99.47%
16	2589	100%
17	2590	86.4%
18	3461	99.87%
19	3665	99.74%
20	3656	98.16%
21	3714	97.5%
22	7070	99.8%
23	7978	99.21%

	Attack ID (snort)	PCC
1	1002	99.8%
2	1091	99.28%
3	1243	100%
4	1250	89.86%
5	1256	70.49%
6	1391	99.93%
7	1497	93.09%
8	1807	84.26%
9	2002	80.54%
10	2123	88.08%
11	2183	86.22%
12	2229	90.52%
13	2329	100%
14	2436	84.26%
15	2518	90.53%
16	2589	100%
17	2590	81.96%
18	3461	77.59%
19	3665	90.59%
20	3656	90.59%
21	3714	87.58%
22	7070	94.67%
23	7978	81.67%

TABLE III
PCC OF OUR APPROACH VS CART

PCC	False negative rate	False positive rate
96.86%	13.19%	2.92%

PCC	False negative rate	False positive rate
89.32%	7.06%	10.76%

TABLE IV
CLASSIFICATION EFFICIENCY OF OUR APPROACH VS CART

false negative by decreasing the threshold, but false positive will certainly increase.

D. Discussion

To evaluate our approach on a real environment, we have used a real dataset in this experiment. We have adapted our approach to detect severe attacks, where some severe attacks may be prepared by low-severity attacks. Experimentation results confirm that most of severe attacks can be expected according to reported low-severity attacks, as it is shown in table 3. Clearly, our approach was able to predict and detect most of severe attacks.

We have also compare our approach to the commercial data mining tool CART. Results show that the classification accuracy of our approach is better than CART. However, we need to modify the default threshold and find a good compromise between false negative and false positive, while preserving a good PCC (more than 90%).

Our approach have several advantages. Firstly, it does not require a large amount of expert knowledge. It does not require specifying preconditions and postconditions of actions comparing to approaches given in [6], [7], [8], nor an explicit representation of attack scenarios such as in [9]. Secondly, it takes advantage of available data by training attack scenarios from historical observations. Lastly, the prediction is done in real time, since the inference in naive Bayes can be achieved efficiently in a linear time.

IV. RELATED WORKS

Bayesian networks have been used in many applications including intrusion detection [10], [11], [12], [13], [14]. However, few works have applied Bayesian network to alert correlation [15], [16]. In fact, the few existing works that apply Bayesian methods to detect coordinated attacks require expert knowledge that the scenario (under the form of attack trees) to be previously defined. In our approach such explicit representation of scenario is not required, and even we do not require to explicitly determining the set of actions involved in a scenario. Everything is obtained from observations history. This section compares our approach with respect to existing works, according to different criteria:

- 1) **Approaches using Bayesian network in intrusion detection** : Bayesian networks have been introduced in intrusion detection area by several researchers. For instance they have been used as Classifier for anomaly and misuse detection [11], [17], [18], [13], [14]. They have also been used for cybercrime detection [10], plan recognition [15], [16], distributed and multi-agent intrusion detection system [19], [12], [20], etc. Abouzakhar et al [10] have proposed a Bayesian learning networks approach to cybercrime detection, in order to detect distributed network attacks as early as possible. Ben Amor et al [17] have proposed a comparative study of using decision trees and naive Bayes as classifier to differentiate between normal and abnormal connections.

Axelsson [11] has proposed an interactive detection system based on simple Bayesian statistics combined with visualization component, in order to counteract the low detection rate and high rate of false alarms. His approach is based on the principle of Bayesian filtering like Spam filtering in Email. It allows system to differentiate between normal and malicious access.

In [20] Scott has described a paradigm for designing network intrusion detection systems based on stochastic models. The principle is to base intrusion detection systems on stochastic models of user and intruder behavior combined using Bayes theorem.

Most recently, Gowdia et al [12] have developed a probabilistic agent-based intrusion detection system. This system is a cooperative agent architecture in which autonomous agents can perform specific intrusion detection tasks and also collaborate with other agents by sharing its beliefs on the same shared Bayesian network.

Clearly all the above approaches have been defined for intrusion detection purposes and not for alert correlation. In particular, the input of all of these systems is not a set of alerts. The following subsection positions our work with some approaches that use Bayesian Network in alert correlation.

2) **Approaches using Bayesian network in alert correlation** : All above works apply Bayesian networks to intrusion detection, but none of the cited works use Bayesian networks to detect coordinated attacks. Now, among existing works using Bayesian network, the one of Qin and Lee [16] is close to our approach.

Qin and Lee [16] have proposed an approach for attack plans recognition and prediction using causal networks. In this approach, authors use attack trees to define attack plan libraries for correlating isolated alerts. They then convert attack trees into causal networks on which they can assign probability distribution by incorporating domain knowledge to evaluate the likelihood of attack goals and predict future attacks.

Most recently [21], Frigault et al have proposed Bayesian network-based graph approach to measure network security. They first interpret a given attack graph as a Bayesian network (this attack graph is supposed obtained by an automatic tool), then they combine individual CVSS (Common Vulnerability Scoring System) [22] scores using their causal relationship. Lastly they integrate the effect of temporal scores of CVSS to derive a final measurement of security.

Clearly, the main difference with our approach is that attack graph should be explicitly defined by an expert in [16] or provided by an automatic external tool [21], which in our approach it is obtained automatically (we even do not determine a priori the set of actions involved in attack scenarios). This is an important advantage of our approach. Our approach is easier to implement and does not need a large amount of expert knowledge. Analyst has just to determine intrusion objectives to

protect and labels when these objectives have been compromised in the observation history. Moreover, our approach implicitly filters false alarms by focusing on relevant actions. Namely, every alert that do not contribute to compromise an intrusion objective will be considered as irrelevant.

3) **Amount of expert knowledge and approaches based on preconditions and postconditions** : Several researchers have proposed several mechanisms based on preconditions and postconditions [6], [8], [7]. Templeton [8] et al have proposed the language JIGSAW for describing attack components in terms of capabilities and concepts (in this work, *precondition* correspond to *requires* and *postcondition* correspond to *provides*).

Cuppens et al [6] have used preconditions and postconditions of actions to implicitly construct attack scenarios. However, this mechanism requires a large amount of expert knowledge to define the preconditions and postconditions associated with each action. Moreover, in [6] when some actions are not observed, some virtual alerts are produced. This increases the number of possible scenarios, and the weighted alert correlation proposed in [23] only limits consequences of this explosion of high number of scenarios.

In parallel to Cuppens works, a similar approach has been proposed by Ning and Cui [7].

The major weakness of alert correlation methods based on precondition and postcondition mechanisms is that these mechanisms involve a large amount of expert knowledge for defining preconditions and postconditions associated with attacks. For instance, in [6], [7], it is required to provide for each action, that may be executed by systems and users, the preconditions and postconditions of this action. This is not always realistic and clearly needs a large amount of expert knowledge. It is clearly difficult to ask an expert to give preconditions and postconditions of all actions of systems, and it is simply impossible to models users own actions. In our two case studies, it is possible to model the DDoS scenarios with preconditions and postconditions approach, however it is hard to apply precondition and postcondition method to the second case study.

In addition, the detection of coordinated actions is very sensitive on the way actions are modeled. A non-necessary condition added to preconditions or postconditions of an action often change the result of correlation and generally produce additional scenarios. For example, suppose that in a given scenario, the next step expected to be executed is to get a root access on system. It is clear that many actions may attempt to get a root access on systems, which give many new variants of first scenario. Similarly, forgetting conditions may leads to a missing of some plausible scenarios. Indeed, a scenario, due to missing conditions, may be viewed as two separate and independent scenarios.

Clearly our approach allows detecting coordinated at-

tacks without requiring a large amount of expert knowledge.

V. CONCLUSION

In this paper, we have shown with an experimentation, how our approach can be easily used to detect severe attacks on the base of observed low-severity alerts. In some applications, alerts of high severity are not isolated, and may be prepared by alerts of low severity. These alerts (of low severity) can be viewed as actions that are needed to be executed before achieving high severe attacks. This problem is clearly related to alert correlation, where intrusion objectives correspond to high severe attacks. This implicitly reduce the number of alerts by focusing on high severe attacks. More precisely, our approach determines the most plausible high severe attacks and actions that contribute to execute them. Actions that do not contribute to the presence of high severe attacks can be considered as irrelevant alerts.

Our modeling has the advantage to make alerts correlation more easily thanks to simplicity and efficiency of naive Bayes. It takes advantage of available data and only need a small part of expert knowledge. Contrary to existing approaches, relationships between attacks are not explicitly provided by experts but are learned automatically from observed data.

REFERENCES

- [1] Benferhat S., Kenaza T., and Mokhtari A. Naive bayes approach for predicting attack scenarios. In *3rd IEEE International Workshop on Security, Trust, and Privacy for Software Application, 28 July - 1 August Turku, Finland, 2008*.
- [2] Finn Verner Jensen. *Introduction to Bayesian networks*. UCL Press, London, 1996.
- [3] Judea Pearl. Probabilistic reasoning in intelligent systems: Networks of plausible inference. *Artif. Intell.*, 48(1):117–124, 1991.
- [4] Nir Friedman and Moises Goldszmidt. Building classifiers using bayesian networks. In *AAAI*, 1996.
- [5] Ron Kohavi, Barry Becker, and Dan Sommerfield. Improving simple bayes. In *European Conference on Machine Learning*, 1997.
- [6] Frédéric Cuppens and Alexandre Miège. Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Security and Privacy*, pages 202–215, 2002.
- [7] Peng Ning, Yun Cui, and Douglas S. Reeves. Analyzing intensive intrusion alerts via correlation. In *RAID*, pages 74–94, 2002.
- [8] J. T. Steven and L. Karm. A requires/provides model for computer attacks. In *New Security Paradigms Workshop*, pages 31–38, 2000.
- [9] Oliver Dain and Robert K. Cunningham. Fusing a heterogeneous alert stream into scenario. In *ACM Workshop on Data Mining for Security Application*, pages 1–13, 2001.
- [10] Naser S. Abouzakhar, A. Gani, G. Manson, M. Abuitbel, and D. King. Bayesian learning networks approach to cybercrime detection. In *the 2003 PostGraduate Networking Conference*, 2003.
- [11] S. Axelsson. Combining a bayesian classifier with visualisation: Understanding the ids. In *VizSEC/DMSEC-04 ACM*, pages 99–108, 2004.
- [12] Vaibhav Gowadia, Csilla Farkas, and Marco Valtorta. Paid: A probabilistic agent-based intrusion detection system. *Computers & Security*, 24(7):529–545, 2005.
- [13] Christopher Krügel, Darren Mutz, William K. Robertson, and Fredrik Valeur. Bayesian event classification for intrusion detection. In *ACSAC*, pages 14–23, 2003.
- [14] Ricardo Puttini, Zakia Marrakchi, and Ludovic M. A bayesian classification model for real-time intrusion detection. In *22nd International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, volume 659, pages 150–162, 2003.
- [15] Christopher W. Geib and Robert P. Goldman. Plan recognition in intrusion detection systems. In *DISCEX*, volume 1, pages 46–55, 2001.
- [16] Xinzhou Qin and Wenke Lee. Attack plan recognition and prediction using causal networks. In *ACSAC*, pages 370–379, 2004.
- [17] Nahla Ben Amor, Salem Benferhat, and Zied Elouedi. Naive bayes vs decision trees in intrusion detection systems. In *SAC*, pages 420–424, 2004.
- [18] Dae-Ki Kang, Doug Fuller, and Vasant Honavar. Learning classifiers for misuse and anomaly detection using a bag of system calls representation. In *IEEE Workshop on Information Assurance and Security*, pages 118–125, 2005.
- [19] Daniel J. Burroughs, Linda F. Wilson, and George V. Cybenko. Analysis of distributed intrusion detection systems using bayesian methods. In *21th IEEE International Conference on Performance, Computing, and Communications*, pages 329–334, 2002.
- [20] L. S. Scott. A bayesian paradigm for designing intrusion detection systems. In *Computational Statistics & Data Analysis*, pages 69–83. Elsevier, 2004.
- [21] M. Frigault and L. Wang. Measuring network security using bayesian network-based attack graph. In *3rd IEEE International Workshop on Security*, 2008.
- [22] Common vulnerability scoring system, <http://www.first.org/cvss/>.
- [23] Salem Benferhat, Fabien Autrel, and Frédéric Cuppens. Enhanced correlation in an intrusion detection process. In *MMM-ACNS*, pages 157–170, 2003.