Reducing False Positives In Anomaly Detectors Through Fuzzy Alert Aggregation

Federico Maggi [1] Matteo Matteucci [1] Stefano Zanero [1,*]

Abstract

In this paper we focus on the aggregation of IDS alerts, an important component of the alert fusion process. We exploit fuzzy measures and fuzzy sets to design simple and robust alert aggregation algorithms. Exploiting fuzzy sets, we are able to robustly state whether or not two alerts are "close in time", dealing with noisy and delayed detections. A performance metric for the evaluation of fusion systems is also proposed. Finally, we evaluate the fusion method with alert streams from anomaly-based IDS.

Key words: Intrusion Detection, Anomaly Detection, Fuzzy Measures, Fuzzy Sets, Aggregation, Multisensor Fusion

1 Introduction

In the modern, ever-changing scenario of computer security, the traditionally effective misuse-based approach to *Intrusion Detection* (ID) is dramatically showing its limits, being based on the exhaustive enumeration of each possible attack signature. As a matter of fact, misuse-based *Intrusion Detection Systems* (IDSes) turn out to be effective only against commonly known attack tools, for which a signature is already available. Their inability to detect unknown attacks (the so-called "zero-days" [1]) or new ways to exploit an old vulnerability is severely limiting. Evasion techniques [2, 3] are another well known example of weakness in misuse-based systems. A possible solution is to use an anomaly detection approach, modelling what is *normal* rather than what is *anomalous*. This is how an intrusion detection system was originally conceived to work [4, 5]. Such systems do not need an up to date database

* Corresponding author.
[1] All the authors are with the Dipartimento di Elettronica e Informazione of the Politecnico di Milano Technical University.

of *"known"* attacks, and therefore can detect unknown techniques and insider abuses as well.

The problem of intrusion detection becomes even more challenging in todays' complex networks. In fact, it is common to have more than one IDS deployed, monitoring different segments and different aspects of the whole infrastructure (e.g., hosts, applications, network, etc.). The amount of alerts fired by a network of IDSes running in a complex computer infrastructure is larger, by several orders of magnitude, than what was common in the smaller networks monitored years ago. In such a context, network administrators are loaded by several alerts and long security reports often containing a non-negligible amount of false positives. Thus, the creation of a clean, compact, and unified view of the security status of the network is needed. This process is commonly known as *alert fusion* (or alert correlation) [6] and it is currently one of the most difficult challenges of this research field.

Skipping the trivial alert format conversion phase, the core steps of an *alert fusion* process are called *aggregation* and *correlation*; the first one has the goal of grouping alerts sharing common features (for instance, those close in time); and the next phase, called *alert correlation*. It has to do with the recognition of logically linked alerts; note that the term "correlation" does not imply "statistical correlation" (even if statistical correlation based methods are sometimes used to reveal such relationships). Alert correlation usually works on already aggregated alert streams and, as shown in Fig. 1, the process is usually performed *after* the aggregation.

The focus of this paper is on alert aggregation, which is more complex when taking into account *anomaly detection* systems, because no information on the type or classification of the observed attack is available to any of the fusion algorithms. Beside the aforementioned drawbacks, misuse detectors are reliable, and give very precise and detailed information upon which a reaction can be planned; on the other hand, they cannot recognize new attacks. Anomaly detectors are symmetrically able to detect new attacks, but are generally prone to false positives and give less information to the user. Most of the algorithms proposed in the current literature on correlation make use of the information regarding the matching attack provided by misuse detectors; therefore, such methods are inapplicable to purely anomaly based intrusion detection systems. However, since failures and strengths of anomaly and misuse detection are symmetric, it is reasonable and noteworthy to try to integrate different approaches through an alert fusion process.

Toward such goal, we explore the use of fuzzy measures [7] and fuzzy sets [8] to design simple, but robust aggregation algorithms. In particular, we contribute to one of the key issues, that is how to state whether or not two alerts are "close in time". In addition, uncertainties on both timestamp measurements

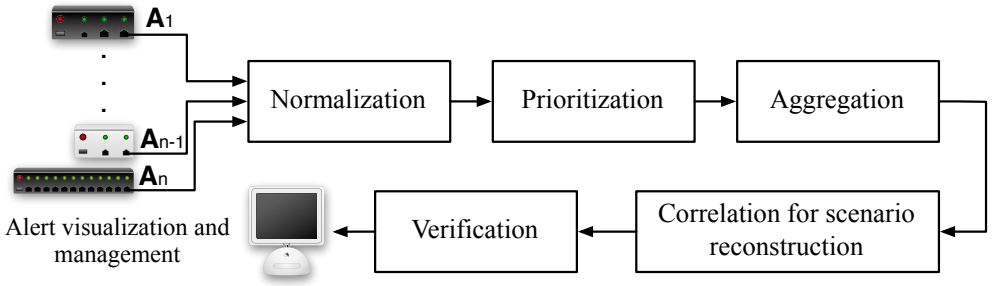


Fig. 1. The alert fusion model used in this work: a slightly modified version of the generic, comprehensive framework proposed in [6].

and threshold setting make this process even more difficult; the use of fuzzy sets allows us to precisely define a time-distance criterion which "embeds" unavoidable errors (e.g., delayed detections).

The remainder of the paper is organized as follows: in Section 2 the problem of alert aggregation and correlation is defined; we will also overview selected papers. Section 3 investigates our idea of exploiting fuzzy models for robust alert aggregation. Section 4 details our experimental setup, the anomaly detection prototype used and outlines the performances of our proposal; in this section, we also sketch the motivations, the main requirements and a framework for the evaluation of alert fusion systems. Finally, in Section 6 we will draw our conclusions and outline some future work.

2 Problem statement and state of the art

The desired output of an *alert fusion* process is a compact, high-level view of what is happening into the network as a whole. In the security field, such a process is also known as *security information monitoring* and has historically been a labor-intensive, error-prone, manual task. Effective algorithms for automatic alert fusion are needed to handle real world applications. In this work we will make use of a slightly modified version of the generic, comprehensive framework of alert fusion proposed in [6]. Fig. 1 summarizes the overall process flow: alerts streams are collected from distributed IDSes, normalized (i.e., translated into a common format), aggregated, and finally, logically linked events in the streams are correlated. In [6] the aggregation phase is called *fusion*; in order to avoid misunderstandings we call it "aggregation" reserving the word "fusion" to indicate the whole process. We will not investigate the prioritization phase and the verification process: the first deals with alert ranking and subsequent discarding, according to predefined priorities and heuristics; alert verification is instead focused on collecting required context information in order to verify whether the attack reported by an alert has succeeded or not.

In this work we will mainly focus on the *aggregation* phase. Aggregation should group together alerts which share common features, such as the same source or destination address port or the same attack name (if known). As a secondary effect, this phase should solve alert granularity issues due to how attacks are reported from an IDS. For instance, an IDS could fire an alert whenever a malicious network packet is detected, while another one might report the *same* alert at the *end* of the TCP connection regarding the *same* attack packet. Thus, the main issue of this phase is how to state that two alerts refer to the same attack, especially if they are reported with slightly different timestamps.

The correlation phase is even more challenging: it implements the *core algorithms* for recognizing related events. An effective correlation algorithm should be able to reconstruct unknown attack scenarios. To give an example, a simple attack scenario is the exploitation of a system and subsequently its use to penetrate another machine; obviously, the violation of both different hosts and the connecting network would cause different (types of) IDSes to fire alerts, all referring to the same scenario.

Alert fusion algorithms commonly need *a priori* knowledge; for instance, precise information about attacks names, division of attacks into classes, and alert priorities. Such a need is limiting for anomaly detectors, because the lack of attack names and classes do not allow grouping along these dimensions. In order to be useful also with anomaly detectors, a generic alert fusion system should not rely only on a priori knowledge, except for obviously required working hypothesis (i.e., alert format, network protocols, etc.) and system configuration settings (i.e., algorithm tuning, etc.).

2.1 Correlation and aggregation approaches

Due to space limitations, we do not attempt to review all the previous literature on intrusion detection, focusing instead on the latest alert fusion techniques. We refer the reader to [9] for a more comprehensive and taxonomic review of IDS literature.

According to an early definition, the alert fusion problem was formalized through state-transition analysis; in [10] this approach has been proven to be an effective technique for representing attack scenarios. In this type of matching engines, observed events are defined as transitions in finite state machine representing signatures of scenarios. When a machine reaches the acceptance state, the described scenario is detected. This approach allows for the modelling of complex intrusion scenarios (mixing network and host based detection) and it is capable of detecting slow or distributed attacks, but it relies on a priori generated scenarios signatures.

In a similar fashion, cause-effect analysis has been applied. JIGSAW [11] is a formal language to specify attacks (called "facts") by means of their pre- and post-conditions (called "requirements" and "providings", respectively). It enables the description of complex scenario and the definition of signatures through the specification of the context needed for an attack to occur. A similar approach that uses a different formalism is proposed in [12]. The same idea is also implemented in TIAA [13], an alert fusion tool that exploits both pre- and post-condition matching and time-distance criteria for aggregation and correlation. As with previous techniques, the major shortcoming of these approaches is the need for a priori knowledge (i.e., scenario pre- and post-conditions).

Statistical techniques have been also proposed. The current version of EMERALD [14] implements a "probabilistic" alert fusion engine. Described in [15], the approach relies on the definition of some similarity metrics between alerts; the correlation phase calculates a weighted similarity value and finds "near" alerts to be grouped together. The features used include source IDS identifiers, timestamps, source and destination addresses and ports. However, the alerts must be already grouped in thread (a characteristic of EMERALD sensors, which is not commonly present on other sensors). The authors recognize the problem of robustly evaluating the distance in time, but use a large, crisp time-window, whereas in this paper we advocate the use of fuzzy intervals. Thus, the two approaches could integrate each other. Unluckily, EMERALD was tested on data which is not available for reproducing the results, and the software itself is not available for download and testing: this makes it difficult to compare the results with our approach.

Classic time-series modelling and analysis have been also applied. The approach detailed in [16] constructs alert time-series counting the number of events occurring into fixed-size sampling intervals; authors then exploits trend and periodicity removal techniques in order to filter out predictable components and leave *real* alerts only as the output. The main shortcoming of this approach is the need for *long* alert streams in order to be effective.

The technique reported in [17] is particularly interesting because it does not require a priori knowledge, except for an *optional* prioritization phase that needs a basic probability assignment for each attack class. Alerts are aggregated into so-called "hyper-alerts" which may be seen as collections of time-ordered items belonging to the same "type" (i.e., with the same features) [16]; each hyper-alert (i.e., time series) is then transformed into frequency time series. The prototype implements a Granger statistical causality test [18]. Without going into details, the test is based on a *causality statistic* which quantifies how much of the history of a given hyper-alert is needed to explain the evolution of another hyper-alert. Repeating the procedure for each couple of hyper-alerts allows to identify "causally related" events and to reconstruct scenarios in an

unsupervised fashion. We compare our technique to [17] in Section 5, and we also analyzed it in depth in [19].

Also, association rule mining techniques have been used [20] in order to learn recurrent alert sequences for unsupervised alert scenario identification. Similar methods have been applied in [21], but the suggested approach requires a manually labeled set of attacks patterns. A similar proposal is described in [22].

In [23], a complementary approach is proposed; the described methodology takes into account the *alert priority* in order to achieve effective contextualization, better understanding of occurring phenomena, and minor attack verification w.r.t. business impacts. Alerts are prioritized and ranked exploiting a priori defined priorities and ranking trees; a subsequent aggregation phase computes similarities values to fuse related alerts together.

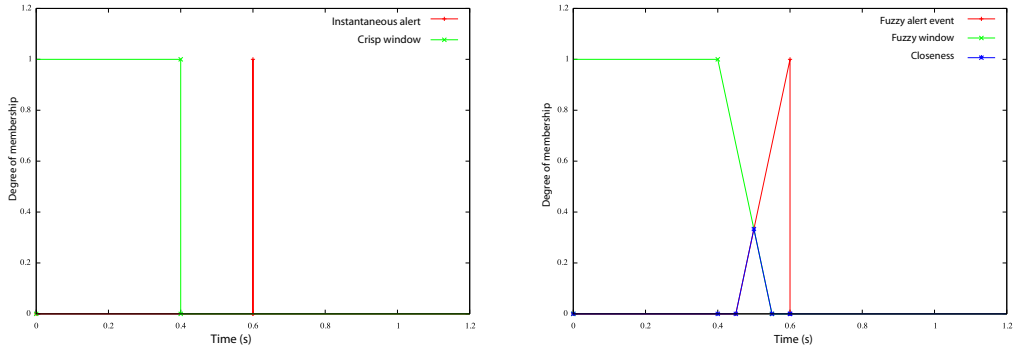
3 Time based alert aggregation: a fuzzy approach

As proposed in most of the reviewed literature, a first, naïve approach consists in exploiting the *time distance* between alerts for aggregation; the idea is to aggregate "near" alerts. In this paper we focus on this point, starting by the definition of "near".

Time-distance aggregation might be able to catch simple scenarios like remote attacks against remote applications vulnerabilities (e.g., web servers). For instance, consider the scenario where, at time t_0, an attacker violates a host by exploiting a vulnerability of a server application. An IDS monitoring the system recognizes anomalous activity at time $t_n = t_0 + \tau_n$. Meanwhile, the attacker might escalate privileges and break through another application; the IDS would detects another anomaly at $t_h = t_0 + \tau_h$. In the most general case t_n is "close" to t_h (with $t_n < t_h$), so if $t_h - t_n \leq T_{near}$ the two alerts belong to the same attack.

The idea is to *fuse* alerts if they are both close in time, raised from the same IDS, and refer to the same source and destination. This intuition obviously needs to be detailed. First, the concept of "near" is not precisely defined; secondly, errors in timestamping are not taken into account; and, a crisp time distance measure is not robust. For instance, if $|t_h - t_n| = 2.451$ and $T_{near} = 2.450$ the two alerts are obviously near, but not aggregated, because the above condition is not satisfied. To overcome such limitations, we propose a *fuzzy* approach to time-based aggregation.

In the following, we will use the well-known dot notation, as in object-oriented programming languages, to access a specific alert attribute: e.g., $a.start\_ts$



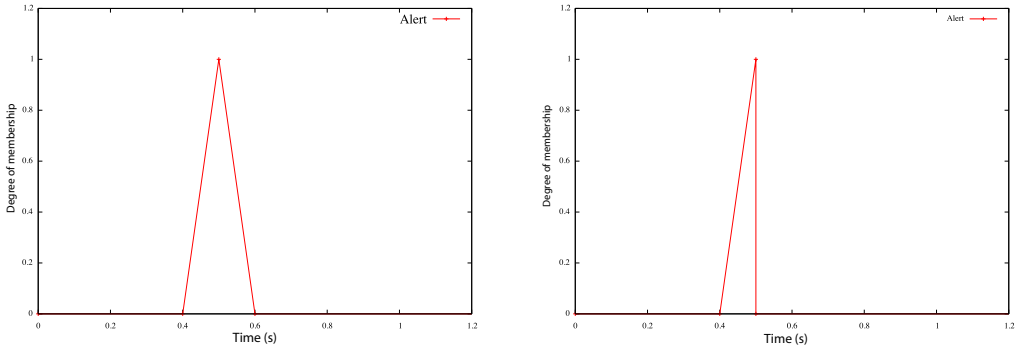
(a) (b)

Fig. 2. Comparison of crisp (a) and fuzzy (b) time-windows. In both graphs, one alert is fired at $t = 0$ and another alert occurs at $t = 0.6$. Using a crisp time-window and instantaneous alerts (a), the distance measurement is not robust to neither delays nor erroneous settings of the time-window size. Using fuzzy-shaped functions (b) provides more robustness and allows to capture the concept of "closeness", as implemented with the T-norm depicted in (b). Distances in time are normalized in [0,1] (w.r.t. the origin).

indicates the value of the attribute *start_ts* of the alert a. Moreover, we will use $T_{(\cdot)}$ to indicate a threshold variable: for instance, T_{near} is the threshold variable called (or regarding to) "near".

Our proposal relies on fuzzy sets for modelling both the uncertainty on the timestamps of alerts and the time distance in a robust manner. Regarding the uncertainty on measurements, we focus on delayed detections by using triangle-shaped fuzzy sets to model the occurrence of an alert. Since the measured timestamp may be affected by errors or delays, we extend the singleton shown in Fig. 2 (a) with a triangle, as depicted in Fig. 2 (b). We also take into account uncertainty on the dimension of the aggregation window: instead of using a crisp window (as in Fig. 2 (a)), we extend it to a trapezoidal fuzzy set, resulting in a more robust distance measurement. In both graphs, one alert is fired at $t = 0$ and another alert occurs at $t = 0.6$. Using a crisp time-window and instantaneous alerts (Fig. 2 (a)), the distance measurement is not robust to neither delays nor erroneous settings of the time-window size. Using fuzzy-shaped functions (Fig. 2 (b)) provides more robustness and allows to capture the concept of "closeness", as implemented with the T-norm depicted in Fig. 2 (b). In Fig. 2 distances in time are normalized in [0,1] (w.r.t. the origin).

Note that, Fig. 3 compares two possible manners to model uncertainty on alert timestamps: in Fig. 3 (a) the alert is recorded at 0.5 seconds but the measurement may have both positive (in the future) and negative (in the past) errors. Fig. 3 (b) is more realistic because positive errors are not likely to happen (i.e., we cannot "anticipate" detections), while events are often delayed, especially in network environments. In comparison to our proposal of using "fuzzy



(a) (b)

Fig. 3. Comparing two possible models of uncertainty on timestamps of single alerts.

timestamps", the *Intrusion Detection Message Exchange Format* (IDMEF) describes event occurrences using *three* timestamps (`Create-`, `Detect-`, and `AnalyzerTime`): this is obviously more generic and allows the reconstruction of the entire event path from the attack to the analyzer that reports the alert. However, all timestamps are not always known: for example, the IDS might not have such a feature, thus the IDMEF `Alert`s cannot be completely filled.

As stated above, the main goal is to measure the time distance between two alerts, a_1 and a_2 (note that, in the following, a_2 occurs after a_1). We first exemplify the case of instantaneous alerts, that is $a_i.start\_ts = a_i.end\_ts = a_i.ts$ for $i \in \{1, 2\}$. To state whether or not a_1 is close to a_2 we use a T_{near} sized time window: in other words, an interval spreading from $a_1.ts$ to $a_1.ts + T_{near}$ (Fig. 2 (a) shows the described situation for $a_1.ts = 0$, $T_{near} = 0.4$, and $a_2.ts = 0.6$: values have been normalized to place a_1 alert in the origin).

Extending the example shown in Fig. 2 (a) to uncertainty in measurements is straightforward; let us suppose to have an average uncertainty of 0.15 seconds on the measured (normalized w.r.t. the origin) value of $a_2.ts$: we model is as a triangle-shaped fuzzy set as the one drawn in Fig. 2 (b).

In the second place, our method takes into account uncertainty regarding the thresholding of the distance between two events, modeled in Fig. 2 (b) by the trapezoidal fuzzy window: the smoothing factor, 0.15 seconds, represents potential errors (i.e., the time values for which the membership function is neither 1 nor 0). Given these premises, the fuzzy variable "near" is defined by a T-norm [8] as shown in Fig. 2 (b): the resulting triangle represents the alerts overlapping in time. In the example we used $\min(\cdot)$ as T-norm.

In the above examples, simple triangles and trapezoids have been used but more accurate/complex membership functions could be used as well. However, we remark here that trapezoid-like sets are conceptually different from triangles as the former have a membership value of 100%; this means cer-

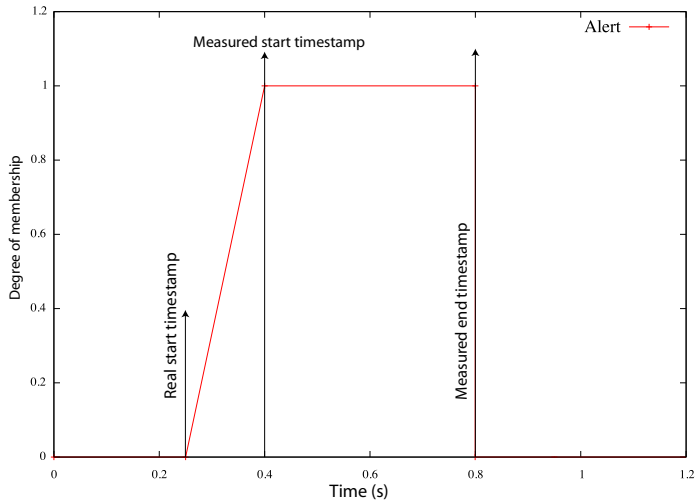


Fig. 4. Non-zero long alert: uncertainty on measured timestamps are modeled.

tainty on the observed/measured phenomenon (i.e., closeness), while lower values mean uncertainty. Trapezoids-like functions should be used whenever a 100%-certainty interval is known: for instance, in Fig. 2 (b) if two alerts occur within 0.4 seconds they are near at 100%; between 0.4 and 0.55 seconds, such certainty decreases accordingly.

The approach can be easily generalized to take into account non-instantaneous events, i.e. $a_i.start\_ts < a_i.end\_ts$. In this case, alerts delays have to be represented by a trapezoidal fuzzy set. Note that, smoothing parameters are configurable values as well as fuzzy set shapes, in order to be as generic as possible w.r.t. the particular network environment; in the most general case, such parameters should be estimated before the system deployment. In Fig. 4 the measured value of $a_i.start\_ts$ is 0.4, while $a_i.end\_ts = 0.8$; moreover, a smoothing factor of 0.15 seconds is added as a negative error, allowing the real $a_i.start\_ts$ to be 0.25.

3.1 Alert pruning

As suggested by the IDMEF `Confidence` attribute, our IDSes provide an `attack_belief` feature. For a generic alert a the `attack_belief` represents the deviation of the anomaly score, $a.score$, from the threshold, T. Intuitively, it gives the idea of the "power" of the detected anomaly. It should be noted that the concept of anomaly score is typical of anomaly detectors and, even if there is no agreement about its definition, it can intuitively interpreted as an internal, absolute indicator of abnormality. To complete our approach we consider the `attack_belief` attribute for alert pruning, after the aggregation phase itself.

As detailed in Section 4, our IDSes rely on probabilistic scores and thresholds in order to isolate anomalies from normal activity; thus, we implemented a first naïve belief measure:

$$\mathrm{Bel}(a) = |a.score - T_{anomaly}| \in [0,1] \tag{1}$$

We remark that $a.score \in [0,1] \wedge T_{anomaly} \in [0,1] \Rightarrow \mathrm{Bel}(a) \in [0,1]$. Also note that in the belief theory [7, 8] $\mathrm{Bel}(B)$ indicates the belief associated to the hypothesis (or proposition) B; with an abbreviate notation, we indicate $\mathrm{Bel}(a)$ meaning the belief of the proposition "the alert a is associated to a real anomaly". In this case, the domain of discourse is the set of all the alerts, which contains both the alerts that are real anomalies and the alerts that are not.

The belief theory has been used to implement complex decision support systems, such as [24], in which a more comprehensive belief model has been formalized taking into account both belief and misbelief. The event (mis)belief basically represents the amount of evidence is available to support the (non-)occurrence of that event. In a similar vein, we exploit both the belief and the *a-priori* misbelief, the *False Positive Rate* (FPR). The FPR tells how much the anomaly detector is likely to report false alerts (i.e., the *a-priori* probability of reporting false alerts, or the so called *type I error*), thus it is a good candidate for modelling the misbelief of the alert. The more the FPR increases, the more the belief associated to an alert should decrease. In the first place, such intuition can be formalized using a linear-scaling function:

$$\mathrm{Bel}_{lin}(a) = (1 - \underbrace{FPR}_{\text{misbelief}})\mathrm{Bel}(a) \tag{2}$$

However, such a scaling function (see, dashed-line plot in Fig. 5) makes the belief to decrease too fast w.r.t. the misbelief. As Fig. 5 shows, a smoother rescaling function is the following:

$$\mathrm{Bel}_{exp}(a) = (1 - FPR)\mathrm{Bel}(a)e^{FPR} \tag{3}$$

The above defined `attack_belief` is exploited to further reduce the amount of alerts. Alerts are reported only if $a_i.attack\_belief > T_{bel}$, where T_{bel} must be set to reasonable values according to the particular environment (in our experiments we used 0.66). The attack belief variable may be modeled using fuzzy variables with, for example, three membership functions labeled as "Low", "Medium", and "High" (IDMEF uses the same three-values assignment for each alert `Confidence` attribute and we used crisp numbers for it), but this has not been implemented yet in our working prototype.

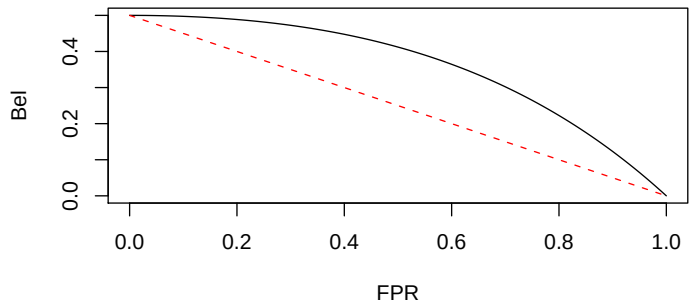


Fig. 5. A sample plot of $\text{Bel}_{exp}(a)$ (with $|a.score - T_{anomaly}|, FPR \in [0,1]$) compared to a linear scaling function, i.e. $\text{Bel}_{lin}(a)$ (dashed line)

Let us now to summarize the incremental approaches; the first "*naïve time-distance*" uses a crisp window (as in Fig. 2 (a)) to groups alerts w.r.t. their timestamps. The second approach, called "*fuzzy time-distance*", extends the first method: it relies on fuzzy sets (as shown in Fig. 2 (b)) to take into account uncertainty on timestamping and window sizing. The last version, "*fuzzy time-distance with belief-based alert pruning*", includes the last explained criterion. In the following we will show the results of the proposed approaches on a realistic test bench we have developed using alerts generated by our IDS on the complete IDEVAL testing dataset (both host and network data).

4 The problem of evaluating alert fusion systems

This section investigates the problem of testing an alert fusion engine. In order to better understand the data we used, the experimental setup is here detailed and, before reporting the results obtained with the fuzzy approach introduced, we also briefly sketch the theoretical foundations, the accuracy, and the peculiarities of the overall implementation of the IDSes used for gathering the alerts. At the end of the section, we compare the results of the aggregation phase in comparison with an analysis of the accuracy of an alternative fusion approach [17].

4.1 A common alert representation

The *Internet Engineering Task Force* (IETF) proposed IDMEF [25] as a common format for reporting alert streams generated by different IDS. Even if this imposes a parsing overhead being XML based, the use of IDMEF is strongly recommended for distributed environments, i.e. for alert correlation systems.

The output format of our prototypes has been designed with IDMEF in mind, in order to make the normalization process (see Fig. 1) as straightforward as possible. Being more precise, alerts are reported by our tools as a modified version of the original SNORT [26] alert format, in order to take into account both the peculiarities of the anomaly detection approach, namely the lack of attacks name, and suggestions from the IDMEF data model. In this way, our testing tools can easily implement conversion procedures from and to IDMEF.

Basically, we represent an alert as a tuple with the following 9 attributes:

```
(ids_id, src, dst, ids_type, start_ts, end_ts, attack_class,
                attack_bel, target_resource)
```

The `ids_id` identifies the IDS, of type `ids_type` (host or network), that has reported the alert; `src` and `dst` are the IP source and destination addresses, respectively; `start_ts` and `end_ts` are the detection NTP [27] timestamps: they are both mandatory, even if the alert duration is unknown (i.e., `start_ts` equals `end_ts`). The discrete attribute `attack_class` (or name) indicates the name of the attack (if known); anomaly detectors cannot provide such an information because recognized attacks names are not known, by definition. Instead, anomaly detectors are able to quantify "how anomalous an activity is" so we exploited this characteristic and defined the `attack_belief` (formally defined in Section 3.1). Finally, the `target_resource` attribute stores the TCP port (or service name), if known, or the program begin attacked. An example instance is the following one:

```
(127.0.0.1-z7012gf, 127.0.0.1, 127.0.0.1, H, 0xbc723b45.0xef449129,
        0xbc723b45.0xff449130, -, 0.141044, fdformat(2351))
```

The example reports an attack detected from an host IDS (`ids_type = H`), identified by `127.0.0.1-z7012gf`: the attack against `fdformat` is started at NTP-second `0xbc723b45` (NTP-picosecond `0xef449129`) and finished at NTP-second `0xbc723b45` (NTP-picosecond 0xff449130); the analyzer detected the activity as an attack with a belief of 14.1044%. An equivalent example for the network IDS (`ids_type = N`) anomaly detector would be something like:

```
(172.16.87.101/24-a032j11, 172.16.87.103/24, 172.16.87.100/24, N,
0xbc723b45.0xef449129, 0xbc723b45.0xff449130, -, 0.290937, smtp)
```

Here the attack is against the `smtp` resource (i.e., protocol) and the analyzer believes there is an attack at 29.0937%.

Alert fusion is a relatively new problem, thus evaluation techniques are limited to a few approaches [28]. The development of solid testing methodologies is needed from both the theoretical and the practical points of view (a problem shared with IDS testing).

The main goal of fusion systems is to reduce the amount of alerts the security analyst has to check. In doing this, the *global Detection Rate* (DR) should ideally not decrease while the *global* FPR should be reduced as much as possible. This suggests that the first sub-goal of fusion systems is the *reduction* of the *global* FPR (for instance, by reducing the total number of alerts fired by source IDS through a rejection mechanism). Moreover, the second sub-goal is to limit the decreasing of the *global* DR. Let us now formalize the concepts we just introduced.

We indicate with $\mathbb{A}_i$ the *alert set* fired by the i-th IDS. An *alert stream* is actually a structure $\langle \mathbb{A}_i, \prec \rangle$ over $\mathbb{A}_i$, where the binary relation $\prec$ means "occurs before". More formally, $\forall a, a' \in \mathbb{A}_i : a \prec a' \Rightarrow a.start\_ts < a'.start\_ts$ with $a.start\_ts, a'.start\_ts \in \mathbb{R}^+$ (NTP timestamps have picoseconds precision, this let us safely assume that it is practically impossible for two alerts to occur simultaneously). We also assume that common operations between sets such as the union $\cup$ preserves the order or, otherwise, that the order can be recomputed; hence, given the union between two alert sets $\mathbb{A}_k = \mathbb{A}_i \cup \mathbb{A}_j$, the alert stream (i.e., ordered set) can be always reconstructed.

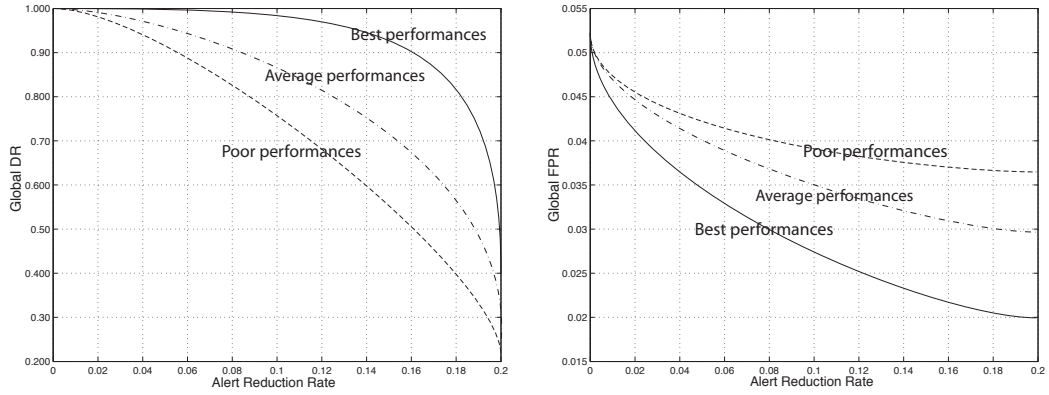
In the second place, we define the two functions $d : \mathbb{A} \times \mathbb{O} \mapsto [0, 1]$, $f : \mathbb{A} \times \mathbb{O} \mapsto [0, 1]$ representing the computation of the DR and the FPR, respectively, that is: $DR_i = \mathrm{d}(\mathbb{A}_i, \hat{\mathbb{O}})$, $FPR_i = \mathrm{f}(\mathbb{A}_i, \hat{\mathbb{O}})$. The set $\mathbb{O}$ contains each and every true alert; $\hat{\mathbb{O}}$ is a particular instance of $\mathbb{O}$ containing alerts occurring in the system under consideration (i.e., those listed in the truth file). We can now define the *global* DR, $DR_{\mathbb{A}}$, and the *global* FPR, $FPR_{\mathbb{A}}$, as:

$$\mathbb{A} = \bigcup_{i \in I} \mathbb{A}_i \tag{4}$$

$$DR_{\mathbb{A}} = \mathrm{d}\left(\mathbb{A}, \hat{\mathbb{O}}\right) \tag{5}$$

$$FPR_{\mathbb{A}} = \mathrm{f}\left(\mathbb{A}, \hat{\mathbb{O}}\right) \tag{6}$$

The output of a generic alert fusion system, is another alert stream $\mathbb{A}'$ with $|\mathbb{A}'| \leq |\mathbb{A}|$; of course, $|\mathbb{A}'| = |\mathbb{A}|$ is a degenerative case. To measure the "impact" of an alert fusion system we define the *Alert Reduction Rate* (ARR) which quantifies:



(a) (b)

Fig. 6. Hypothetical plots of (a) the global DR, $DR_{\mathbb{A}'}$, vs. ARR and (b) the global FPR, $FPR_{\mathbb{A}'}$, vs. ARR.

$$ARR = \frac{|\mathbb{A}'|}{|\mathbb{A}|}. \tag{7}$$

The aforementioned sub-goals can now be formalized into two performance metrics. The ideal correlation system would maximize both ARR and $\frac{DR_{\mathbb{A}'}}{DR_{\mathbb{A}}}$; meanwhile, it would minimize $\frac{FPR_{\mathbb{A}'}}{FPR_{\mathbb{A}}}$. Of course, $ARR = 1$ and $ARR = 0$ are degenerative cases. Low (i.e., close to 0.0) $\frac{FPR_{\mathbb{A}'}}{FPR_{\mathbb{A}}}$ means that the fusion system has significantly decreased the original FPR; in a similar vein, high (i.e., close to 1.0) $\frac{DR_{\mathbb{A}'}}{DR_{\mathbb{A}}}$ means that, while reducing false positive alerts, the loss of detection capabilities is minimal.

Therefore, a fusion system is better than another if it has *both* a greater $\frac{DR_{\mathbb{A}'}}{DR_{\mathbb{A}}}$ and a lower $\frac{FPR_{\mathbb{A}'}}{FPR_{\mathbb{A}}}$ rate than the latter. This criterion by no means has to be considered as complete or exhaustive. In addition, it is useful to compare $DR_{\mathbb{A}'}$ and $FPR_{\mathbb{A}'}$ plots, vs. ARR, of different correlation systems obtaining diagrams like the one exemplified in Fig. 6; this gives a graphical idea of which correlation algorithm is better. For instance, Fig. 6 show that the algorithm labeled as Best performances is better than the others, because it shows higher $FPR_{\mathbb{A}'}$ reduction while $DR_{\mathbb{A}'}$ does not significantly decrease.

4.3 Our experimental setup

The experimental setup we used involves two novel prototypes [2] for network and host anomaly detection. These tools were used to generate alert streams for testing the three variants of the fuzzy aggregation algorithm we proposed.

[2] The prototypes are available upon request to the authors.

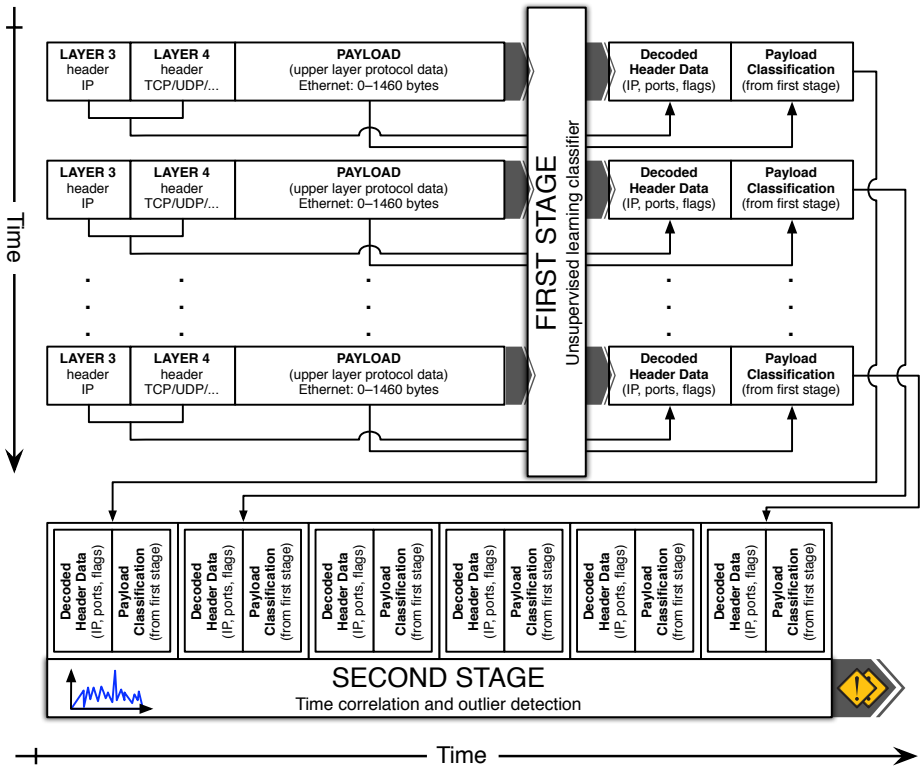


Fig. 7. The overall architecture of the two-stage network-based IDS prototype.

The same tools were also used to generate alerts data for testing our correlation algorithm detailed in [19].

4.3.1 Network Anomaly Detector

In previous works [29–32] we proposed a novel network based anomaly detection system which uses a two-tier architecture to overcome dimensionality problems and apply unsupervised learning techniques to the payload of packets, as well as to the headers. Most existing researches avoid this problem altogether by discarding the payload and retaining only the information in the header [33–37].

Without going into details, the overall architecture is shown in Fig. 7. The first tier exploits the clustering capabilities of a *Self Organizing Map* (SOM) [38] to analyze network packet payloads, characterizing their recurring patterns in an unsupervised manner [29,30]. On most network segments, the traffic belongs to a relatively small number of services and protocols that can be mapped onto a relatively small number of clusters. In [31] we analyzed the performance of these algorithms, and suggested improvements in order to decrease the computational cost of the SOM algorithm, and to increase the throughput of the system. The second tier is based on a modified version of SmartSifter [39], a discounting learning algorithm for outlier detection. We showed that our

Threshold	Detection Rate	False Positive Rate
0.03%	66.7%	0.031%
0.05%	72.2%	0.055%
0.08%	77.8%	0.086%
0.09%	88.9%	0.095%

Table 1
Detection rates and false positive rates for our prototype

two-tier architecture greatly enhances efficiency of the prototype.

In [29–32] we also analyzed the accuracy and the performance of the prototype. The average results are reported in Table 1 and Fig. 8. The first column contains the sensitivity threshold of the algorithm, generated as a quantile of the approximated distribution of observed data [40]; as it could be expected, it is a proper statistical predictor of the percentage of data that will be flagged as outliers by the algorithm (i.e., the empirical distribution is close to the real one). As we can see, it is also a good predictor of the FPR, if the attack rate is not too high. The prototype is able to reach a 66.7% DR with as few as 0.03% false positives. In comparison, in the only prototype dealing with payloads, available in literature [41], the best overall result leads to the detection of 58.7% of the attacks, with a FPR that is between 0.1% and 1%. Our prototype shows thus a better DR, with a number of false positives which is between one and two order of magnitudes lower than comparable systems (e.g., [41]), as shown in the *Receiver Operating Characteristic* (ROC) curve Fig. 8.

4.3.2 System Call Anomaly Detector

The second prototype, proposed and detailed in [42], is an *Host-based IDS* (HIDS) which can detect anomalies by analyzing system call arguments and sequences. The system is an almost complete re-engineering of SyscallAnomaly [43], that was able to detect only anomalies in arguments in its original release. In particular, our prototype implements some of the ideas of SyscallAnomaly along with Markovian based modelling, clustering and behavior identification outperforming the original application with both an increased DR and a reduced FPR.

The tool works by analyzing syscall traces logged in OpenBSM [44] or other custom ASCII formats. The overall architecture is drawn in Fig. 9. During training, a hierarchical clustering algorithm, based on a distance measure between probabilistic models, is used to identify groups of similar syscalls (for details see [32, 42]); the resulting clusters become the nodes a of Markov chain

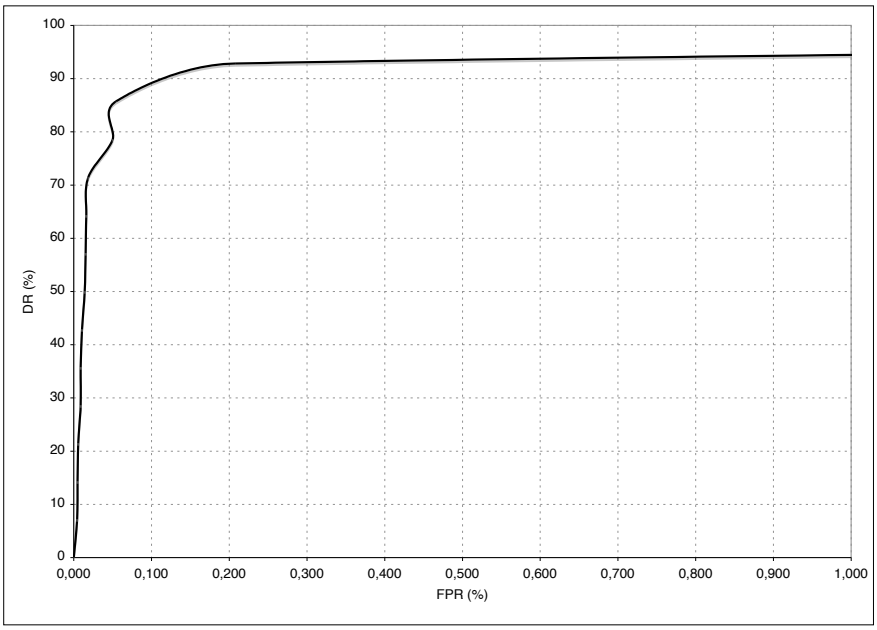


Fig. 8. The ROC curve of the network anomaly detector: the tool has been trained with about 10^6 TCP/IP dumped packets and tested with about 10^4 packets.

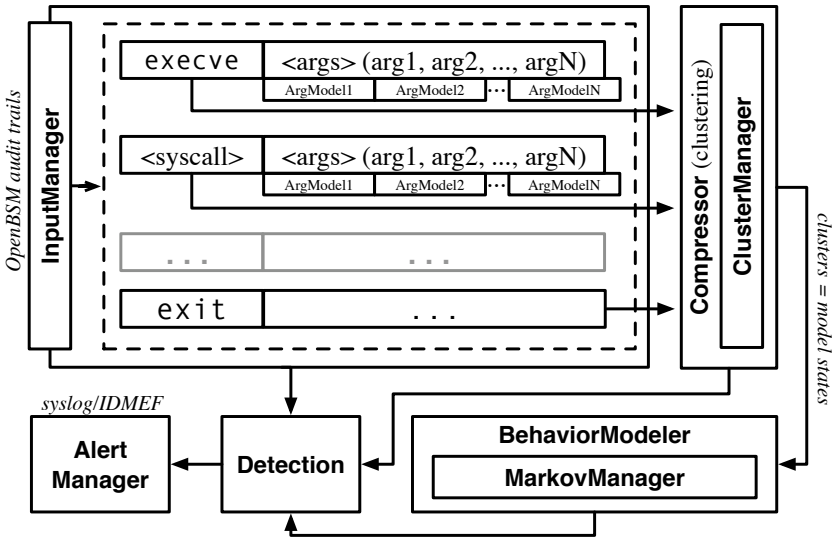


Fig. 9. The overall architecture of the host-based IDS prototype.

built in the second stage to characterize and learn the process behavior of each application in terms of syscall sequences behavior. At the end of this training step we store both clusters and transition models, along with two thresholds (syscall threshold and sequence threshold) required at runtime for deciding whether or not a new call or a new syscall sequence is anomalous. For further details on cluster models, implementation of the transition models, and threshold computation please refer to [32, 42].

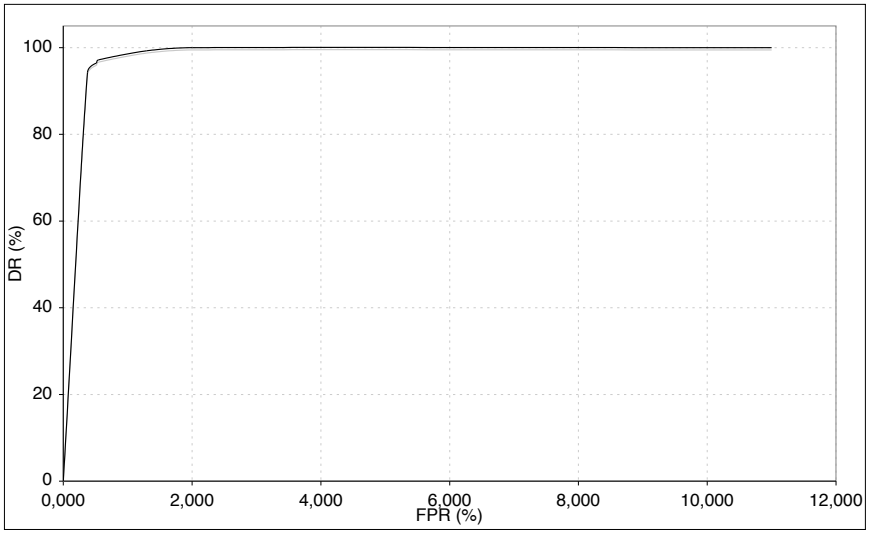


Fig. 10. The ROC curve of the host anomaly detector.

As detailed in [42], the tool is implemented in ANSI C and it is capable of processing around 10^3 system calls per second in the training phase; the runtime throughput is as high as 22266 $syscall/s$. We recall that the prototype is also very accurate: on novel attacks (e.g., custom and stealthy exploits against `bsdtar` and `eject`), it is capable of DR between $88 - 100\%$ with no more than 1.6% as FPR. For comprehensive accuracy and performance tests, please refer to [42]. We also recall the accuracy of the prototype: the DR vs. FPR is plot in Fig. 10.

4.4 Test data generation

A well-known problem in IDS research is the lack of reliable sources of test data. Gathering full traces of real world networks is generally impossible for privacy reasons; also, IDS researchers need clearly labeled data where attacks are described in full details, something which is usually impossible to achieve with real-world dumps. The dataset created by the Lincoln Laboratory at M.I.T., also known as "DARPA IDS Evaluation dataset" [45], is basically the only dataset of this kind which is freely available along with complete truth files. Other datasets exist (e.g., the DEFCON CTF packet capture [46]), but they are not labeled and do not contain "background traffic" (i.e., attack-free activity for IDS training). Thus, most existing researches on network based IDS use mainly the DARPA datasets for evaluation.

These data have been artificially generated and collected in order to evaluate detection rates and false positives rates of IDS. There are two datasets: 1998 and 1999. The 1999 dataset [47], which we extensively used in this work, spans over 5 weeks, and contains the packet dumps in `tcpdump` format of 5 weeks, over 2 sniffers, one placed between the gateway and 5 "target" machines (thus emulating an "internal" sniffer), and the other placed beyond the gateway, recording packets flowing between the simulated LAN and the simulated Internet. Both attack-free data and clearly labeled attack traces are present. Besides the `tcpdump` traces, BSM auditing data for Solaris systems, NT auditing data for Windows systems, directory tree snapshots of each system, the content of sensitive directories, and inode data are available.

A common question is how realistic these data are, having been artificially generated specifically for IDS evaluation. Many authors already analyzed the network data of the 1999 dataset, finding many shortcomings [48,49]: no detail is available on the generation methods, there is no evidence that the traffic is actually realistic, and no spurious packet, checksum errors or fragmented packets are present. In addition, the synthetic packets share strange regularities. Host auditing data is all but immune from problems as well, as we showed in [42]: the dataset is limited, full of artifacts, and prone to overfitting. However, it must be noted that these anomalies are mainly an issue when evaluating the performance of single intrusion detectors. In our case, they are not extremely relevant.

What is relevant, actually, is that the whole dataset is outdated, both in the background traffic and the in the attacks. The most used attack technique is buffer overflow, and intrusion scenarios are extremely simple. Additionally, even if IDEVAL contains both host and network auditing data, some attacks are not directly detectable in both systems, making them less relevant for correlation testing. The only such attacks are the ones in which an attacker exploits a vulnerability in a local or remote service to allow an intruder to obtain or escalate privileges. One of the best target hosts for finding such correlations is `pascal.eyrie.af.mil`, which runs Solaris 2.5.1.

For all the previous reasons, in our testing we will use the IDEVAL dataset with the following simplification: we will just try to fuse the stream of alerts coming from a HIDS sensor and a NIDS sensor, which is monitoring the whole network. To this end, we ran the two prototypes described above in Section 4.3 on the whole 1999 IDEVAL testing dataset, using the network dumps and the host-based logs from `pascal`. We ran the NIDS prototype on `tcpdump` data and collected 128 alerts for attacks against the host `pascal.eyrie-.af.mil`. The NIDS also generated 1009 alerts related to other hosts. Using the HIDS prototype we generated 1070 alerts from the dumps of the host `pascal.eyrie.af.mil`. With respect to these alerts, the NIDS was capable of detecting almost 66% of the attacks with less than 0.03% of false positives;

the HIDS performs even better with a detection rate of 98% and 1.7% of false positives.

In the following, we use this shorthand notation: Net is the substream of all the alerts generated by the NIDS. $HostP$ is the substream of all the alerts generated by the HIDS installed on `pascal.eyrie.af.mil`, while $NetP$ regards all the alerts (with `pascal` as a target) generated by the NIDS; finally, $NetO = Net \backslash NetP$ indicates all the alerts (with all but `pascal` as a target) generated by the NIDS.

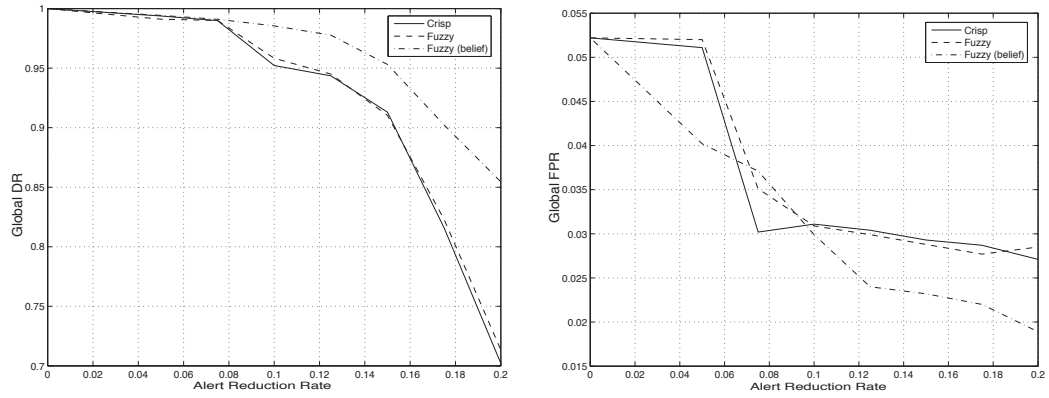
5 Experimental Results

Using the data generated as described above in Section 4.4, and the metrics proposed in Section 4.2, we compared three different versions of the alert aggregation algorithms proposed in Section 3. In particular we compare the use of crisp time-distance aggregation, the use use of a simple fuzzy time-distance aggregation; and finally, the use of `attack_belief` for alert pruning.

Numerical results are plotted in in Fig. 11 for different values of ARR. As we discussed in Section 4.2, Fig. 11 (a) refers to the reduction of DR while Fig. 11 (b) focuses on FPR. $DR_{\mathbb{A}'}$ and $FPR_{\mathbb{A}'}$ were calculated using the complete alert stream, network and host, at different values of ARR. The values of ARR are obtained by changing the parameters values: in particular, we set $T_{bel} = 0.66$, the alpha cut of T_{near} to 0.5, the window size to 1.0 seconds, and varied the smoothing of the trapezoid between 1.0 and 1.5 seconds, and the alert delay between 0 and 1.5 seconds. It is not useful to plot the increase in false negatives, as it can be easily deduced from the decrease in DR.

The last aggregation algorithm, denoted as "Fuzzy (belief)", shows better performances since the $DR_{\mathbb{A}'}$ is always higher w.r.t. the other two aggregation strategies; this algorithm also causes a significant reduction of the $FPR_{\mathbb{A}'}$. Note that, taking into account the `attack_belief` attribute makes the difference because it avoids true positives to be discarded; on the other hand, *real* false positive are not reported in the output alert stream because of their low belief.

It is difficult to properly compare our approach with other other fusion approaches proposed in the reviewed literature, because the latter were not specifically tested from the aggregation point of view, separately from the correlation one. Since the prototypes used to produce results are not released, we are only able to compare our approach with the *overall* fusion systems presented by others.



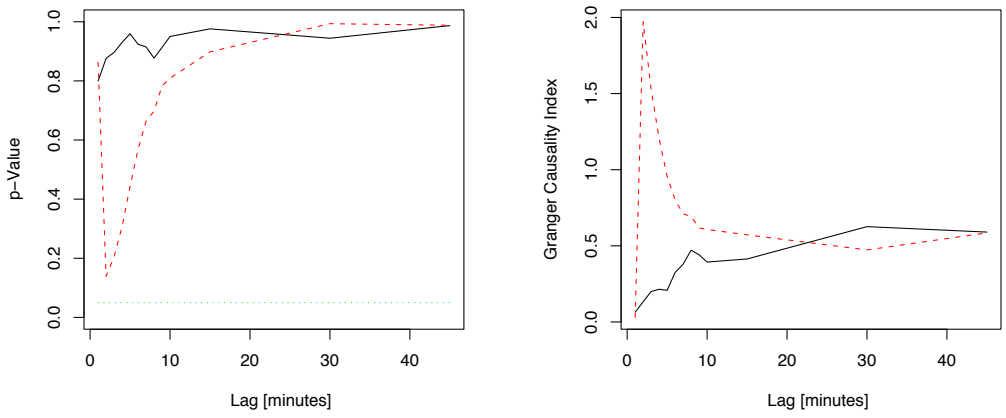
(a) (b)

Fig. 11. Plot of the $DR_{\mathbb{A}'}$ (a) and $FPR_{\mathbb{A}'}$ (b) vs. ARR. "Crisp" refers to the use of the crisp time-distance aggregation; "Fuzzy" and "Fuzzy (belief)" indicates the simple fuzzy time-distance aggregation and the use of the `attack_belief` for alert discarding, respectively.

In particular, we analyzed the use of the Granger causality test [50] as proposed in [17]. These results were preliminarily presented in [19]. We used the same dataset used for evaluating our system, with the following approach: we tested if $NetP$ has any causal relationship[3] with $HostP$; we also tested $HostP \not\rightsquigarrow NetP$. Since only a reduced number of alerts was available and since it was impossible to aggregate alerts along the "attack name" attribute (unavailable on anomaly detectors), we preferred to construct only two, large time series: one from $NetP$ (denoted as $NetP(k)$) and one from $HostP$ (denoted as $HostP(k)$). In the experiment reported in [17] the sampling time, w, has been fixed at 60 seconds, although we tested different values from 60 to 3600 seconds.

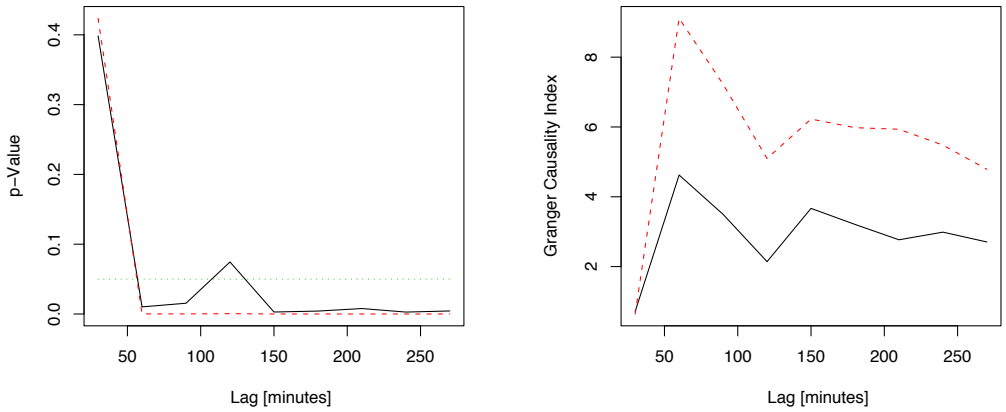
Since the first experiment ($w = 60$) led us to unexpected results (i.e., using a lag, l, of 3 minutes, both $NetP(k) \not\rightsquigarrow HostP(k)$, and vice versa) we decided to plot the test results (i.e., p-value and Granger Causality Index (GCI)) vs. both l and w. In Fig. 12 (a) l is reported in minutes while the experiment has been performed with $w = 60s$; the dashed line is the p-value of the test "$NetP(k) \rightsquigarrow HostP(k)$", the solid line is opposite one. As one may notice, neither the first nor the second test passed, thus nothing can be concluded: fixing the test significance at $\alpha = 0.20$ is the only way for refusing the null hypothesis (around $l \simeq 2.5$ minutes) to conclude both that $NetP \rightsquigarrow HostP$" and that $HostP \not\rightsquigarrow NetP$; the GCI plotted in Fig. 12 (b) confirms the previous result. However, for other values of l the situation changes leading to opposite results.

[3] In the following we will use the symbol "$\rightsquigarrow$" to denote "Granger-causes" so, for instance, "A $\rightsquigarrow$ B" has to be read as "A causes B"; the symbol "$\not\rightsquigarrow$" indicates the negated causal relationship, i.e., "does not Granger-cause".



(a) (b)

Fig. 12. p-value (a) and GCI (b) vs. l (in minutes) for the first Granger causality test experiment ($w = 60.0$ seconds): "$NetP(k) \rightsquigarrow HostP(k)$" (dashed line), "$HostP(k) \rightsquigarrow NetP(k)$" (solid line).

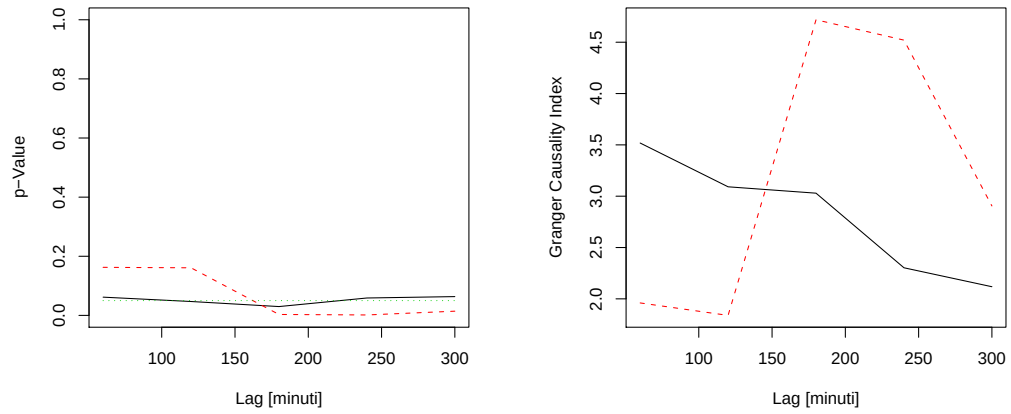


(a) (b)

Fig. 13. p-value (a) and GCI (b) vs. l (in minutes) for the first Granger causality test experiment ($w = 1800.0$ seconds): "$NetP(k) \rightsquigarrow HostP(k)$" (dashed line), "$HostP(k) \rightsquigarrow NetP(k)$" (solid line).

Fig. 13 shows the results of the test for $w = 1800$ seconds and $l \in [50, 300]$ minutes. If we suppose a test significance of $\alpha = 0.05$ (dotted line), for $l \simeq 120$ the result is that $HostP \rightsquigarrow NetP$ while $NetP \not\rightsquigarrow HostP$, the opposite of the previous one. Moreover, for other values of l the p-value leads to different results.

The last experiment results are shown in Fig. 14: for $l > 230$ minutes, one may



(a) (b)

Fig. 14. p-value (a) and GCI (b) vs. p (in minutes) for the first Granger causality test experiment ($w = 3600.0$ seconds): "$NetP(k) \rightsquigarrow HostP(k)$" (dashed line), "$HostP(k) \rightsquigarrow NetP(k)$" (solid line).

conclude that $NetP \rightsquigarrow HostP$ and $HostP \not\rightsquigarrow NetP$; i.e., the expected result, even if the p-value for the second test is close to $\alpha = 0.05$.

The previous experiments show only that the Granger causality test failed on the dataset we generated, not that it doesn't work in general. It may be the case that the Granger causality test is not suitable for "blind" alerts (i.e., without any information about the attack) reported by anomaly detectors: in fact, [17] used time series built from hyper-alerts resulting from an aggregation along *all* attributes: instead, in the anomaly-based case, the lack of attack names does not allow such hyper-alerts to be correctly constructed. In any case, the test result significantly depends on l (i.e., the test is asymptotic); this means that an appropriate choice of l will be required, depending on specific environmental conditions. The optimal l is also likely to change over time in a given environment.

6 Conclusions

In this paper we described and evaluated a technique which uses fuzzy sets and measures to fuse alerts reported by the anomaly detectors. After a brief framework description and precise problem statement, we analyzed previous literature about alert fusion (i.e., aggregation and correlation), and found that effective techniques have been proposed, but they are not really suitable for anomaly detection, because they require a priori knowledge (e.g., attack names or division into classes) to perform well.

Our proposal defines simple, but robust criteria for computing the time distance between alerts in order to take into account uncertainty on both measurements and the threshold-distance sizing. In addition, we considered the implementation of a post-aggregation phase to remove unaggregated alerts according to their belief, a value indicating how much the IDS believes the detected attack to be real. Moreover, we defined and used some simple metrics for the evaluation of alert fusion systems. In particular, we propose to plot both the DR and the FPR vs. the degree of output alert reduction vs. the size of the input alert stream.

We performed experiments for validating our proposal. To this end, we used two prototypes we previously developed: a host anomaly detector, that exploits the analysis of system calls arguments and behavior, and a network anomaly detector, based on unsupervised payload clustering and classification techniques that enables an effective outlier detection algorithm to flag anomalies. During our experiments, we were able to outline many shortcomings of the IDEVAL dataset (the only available IDS benchmark) when used for evaluating alert fusion systems. In addition to known anomalies in network and host data, IDEVAL is outdated both in terms of background traffic and attack scenarios.

Our experiments showed that the proposed fuzzy aggregation approach is able to decrease the FPR at the price of a small reduction of the DR (a necessary consequence). The approach defines the notion of "closeness" in time as the natural extension of the naïve, crisp way; to this end, we rely both on fuzzy set theory and fuzzy measures to semantically ground the concept of "closeness". By definition, our method is robust because it takes into account major uncertainties on timestamps; this means the choice of window size is less sensitive to fluctuations in the network delays because of the smoothing allowed by the fuzziness of the window itself. Of course, if the delays are varying too much, a dynamic resizing is still necessary. The biggest challenge with our approach would be its extension to the correlation of distributed alerts: in the current state, our modelling is not complete, but can potentially be extended in such a way; being the lack of alert features the main difficult.

We also showed preliminary results on the use of the Granger causality test to recognize logically linked alerts, also giving a statistical quantification of the degree of "causality". However, a full exploitation of this technique is the subject for future extensions of this work. Even if the method does not require a priori knowledge, we identified two significant issues: first, the statistical test relies on non-obvious configuration parameters which values significantly affect the final result; second, in order to extensively test such a methods a better dataset than IDEVAL would be needed. We believe that the use of the Granger causality test might be applied to alerts reported by anomaly detectors as well. Another possible extension of this work is the investigation of algorithms and

criteria to correlate anomaly and misuse-based alerts together, in order to bridge the gap between the existing paradigms of intrusion detection.

Acknowledgments

Most of this work was supported by the Italian Ministry of Education and Research under the FIRB Project "Performance evaluation for complex systems", in the research unit led by Prof. Giuseppe Serazzi, whose support we gratefully acknowledge. We need also to thank Prof. Andrea Bonarini for his helpful comments.

References

[1] S. Zanero, Detecting 0-day attacks with learning intrusion detection systems, in: Blackhat USA 2004 Briefings, 2004.

[2] T. H. Ptacek, T. N. Newsham, Insertion, evasion, and denial of service: Eluding network intrusion detection, Tech. Rep. T2R-0Y6, Secure Networks, Calgary, Canada (1998).

[3] G. Vigna, W. Robertson, D. Balzarotti, Testing Network-based Intrusion Detection Signatures Using Mutant Exploits, in: Proceedings of the ACM Conference on Computer and Communication Security (ACM CCS), Washington, DC, 2004, pp. 21–30.

[4] D. E. Denning, An intrusion-detection model, IEEE Transactions on Software Engineering SE-13 (2) (1987) 222–232.

[5] J. P. Anderson, Computer security threat monitoring and surveillance, Tech. rep., J. P. Anderson Co., Ft. Washington, Pennsylvania (April 1980).

[6] F. Valeur, A comprehensive approach to intrusion detection alert correlation, IEEE Trans. Dependable Secur. Comput. 1 (3) (2004) 146–169, member-Giovanni Vigna and Member-Christopher Kruegel and Fellow-Richard A. Kemmerer.

[7] Z. Wang, G. J. Klir, Fuzzy Measure Theory, Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[8] G. J. Klir, T. A. Folger, Fuzzy sets, uncertainty, and information, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

[9] R. G. Bace, Intrusion detection, Macmillan Publishing Co., Inc., Indianapolis, IN, USA, 2000.

[10] S. Eckmann, G. Vigna, R. Kemmerer, STATL: An attack language for state-based intrusion detection, in: Proceedings of the ACM Workshop on Intrusion Detection, Athens, 2000.

[11] S. J. Templeton, K. Levitt, A requires/provides model for computer attacks, in: NSPW '00: Proceedings of the 2000 workshop on New security paradigms, ACM Press, New York, NY, USA, 2000, pp. 31–38.

[12] F. Cuppens, A. Miége, Alert correlation in a cooperative intrusion detection framework, in: SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy, IEEE Computer Society, Washington, DC, USA, 2002, p. 202.

[13] P. Ning, Y. Cui, D. S. Reeves, D. Xu, Techniques and tools for analyzing intrusion alerts, ACM Trans. Inf. Syst. Secur. 7 (2) (2004) 274–318.

[14] P. A. Porras, P. G. Neumann, EMERALD: Event monitoring enabling responses to anomalous live disturbances, in: Proc. 20th NIST-NCSC Nat'l Information Systems Security Conf., 1997, pp. 353–365.

[15] A. Valdes, K. Skinner, Probabilistic alert correlation, in: RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, Springer-Verlag, London, UK, 2001, pp. 54–68.

[16] J. Viinikka, H. Debar, L. Mé;, R. Séguier, Time series modeling for IDS alert management, in: ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, ACM Press, New York, NY, USA, 2006, pp. 102–113.

[17] X. Qin, W. Lee, Statistical causality analysis of infosec alert data., in: RAID, 2003, pp. 73–93.

[18] W. N. Thurman, M. E. Fisher, Chickens, eggs, and causality, or which came first?, American Journal of Agricultural Economics.

[19] F. Maggi, S. Zanero, On the use of different statistical tests for alert correlation - short paper, in: C. Krügel, R. Lippmann, A. Clark (Eds.), RAID, Vol. 4637 of Lecture Notes in Computer Science, Springer, 2007, pp. 167–177.

[20] K. Julisch, M. Dacier, Mining intrusion detection alarms for actionable knowledge, in: KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM Press, New York, NY, USA, 2002, pp. 366–375.

[21] O. Dain, R. Cunningham, Fusing heterogeneous alert streams into scenarios, in: Proc. of the ACM Workshop on Data Mining for Security Applications, 2001, pp. 1–13.

[22] H. Debar, A. Wespi, Aggregation and correlation of intrusion-detection alerts, in: RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, Springer-Verlag, London, UK, 2001, pp. 85–103.

[23] P. A. Porras, M. W. Fong, A. Valdes, A mission-impact-based approach to infosec alarm correlation, Lecture Notes in Computer Science 2516/2002 (2002) 35.

[24] E. H. Shortliffe, Computer-based medical consultations: MYCIN, Elsevier, 1976.

[25] H. Debar, D. Curry, B. Feinstein, The intrusion detection message exchange format, Tech. rep., France Telecom and Guardian and TNT (March 2006).

[26] Snort, http://www.snort.org (2006).

[27] D. L. Mills, Rfc1305: Network time protocol (version 3), Available online at: `http://www.ietf.org/rfc/rfc1305.txt` (1992).

[28] J. Haines, D. K. Ryder, L. Tinnel, S. Taylor, Validation of sensor alert correlators, IEEE Security and Privacy 01 (1) (2003) 46–56.

[29] S. Zanero, S. M. Savaresi, Unsupervised learning techniques for an intrusion detection system, in: Proc. of the 2004 ACM Symposium on Applied Computing, ACM Press, 2004, pp. 412–419.

[30] S. Zanero, Analyzing tcp traffic patterns using self organizing maps, in: F. Roli, S. Vitulano (Eds.), 13th International Conference on Image Analysis and Processing - ICIAP 2005, Vol. 3617 of Lecture Notes in Computer Science, Springer, Cagliari, Italy, 2005, pp. 83–90.

[31] S. Zanero, Improving self organizing map performance for network intrusion detection, in: SDM 2005 Workshop on "Clustering High Dimensional Data and its Applications", 2005.

[32] S. Zanero, Unsupervised learning algorithms for intrusion detection, Ph.D. thesis, Politecnico di Milano T.U., Milano, Italy (May 2006).

[33] M. Mahoney, P. Chan, Detecting novel attacks by identifying anomalous network packet headers, Tech. Rep. CS-2001-2, Florida Institute of Technology (2001).

[34] C. Chow, Parzen-Window network intrusion detectors, in: ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 4, IEEE Computer Society, Washington, DC, USA, 2002, pp. 385–388.

[35] K. Labib, R. Vemuri, NSOM: A real-time network-based intrusion detection system using self-organizing maps, Tech. rep., Dept. of Applied Science, University of California, Davis (2002).

[36] M. V. Mahoney, P. K. Chan, A machine learning approach to detecting attacks by identifying anomalies in network traffic, Tech. Rep. CS-2002-08, Florida Institute of Technology (2002).

[37] M. V. Mahoney, Network traffic anomaly detection based on packet bytes, in: Proceedings of the 19th Annual ACM Symposium on Applied Computing, 2003.

[38] T. Kohonen, Self-Organizing Maps, 3rd Edition, Springer-Verlag, Berlin, 2001.

[39] K. Yamanishi, J.-I. Takeuchi, G. Williams, P. Milne, On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms, Data Min. Knowl. Discov. 8 (3) (2004) 275–300.

[40] K. Yamanishi, J.-I. Takeuchi, G. J. Williams, P. Milne, Online unsupervised outlier detection using finite mixtures with discounting learning algorithms, Knowledge Discovery and Data Mining 8 (3) (2004) 275–300.

[41] K. Wang, S. J. Stolfo, Anomalous payload-based network intrusion detection, in: RAID Symposium, 2004.

[42] F. Maggi, M. Matteucci, S. Zanero, Detecting intrusions through system call sequence and argument analysis, Submitted for publication (2006).

[43] C. Kruegel, D. Mutz, F. Valeur, G. Vigna, On the Detection of Anomalous System Call Arguments, in: Proceedings of the 2003 European Symposium on Research in Computer Security, Gjøvik, Norway, 2003.

[44] R. N. M. Watson, W. Salamon, The FreeBSD audit system, in: UKUUG LISA Conf., Durham, UK, 2006.

[45] M. Zissman, Darpa intrusion detection evaluation, `http://www.ll.mit.edu/IST/ideval/data/dataindex.html` (1999).

[46] B. Potter, The Shmoo Group Capture the CTF project, `http://cctf.shmoo.com` (2006).

[47] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, K. Das, Analysis and results of the 1999 DARPA off-line intrusion detection evaluation, in: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection, Springer-Verlag, London, UK, 2000, pp. 162–182.

[48] J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory, ACM Trans. on Information and System Security 3 (4) (2000) 262–294.

[49] M. V. Mahoney, P. K. Chan, An analysis of the 1999 DARPA / Lincoln laboratory evaluation data for network anomaly detection, in: Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003), Pittsburgh, PA, USA, 2003, pp. 220–237.

[50] C. Granger, Investigating causal relations by econometric methods and crossspectral methods, Econometrica 34 (1969) 424–428.