Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

An approach to the correlation of security events based on machine learning techniques

Kleber Stroeh[1*], Edmundo Roberto Mauro Madeira[2] and Siome Klein Goldenstein[2]

Abstract

Organizations face the ever growing challenge of providing security within their IT infrastructures. Static approaches to security, such as perimetral defense, have proven less than effective — and, therefore, more vulnerable — in a new scenario characterized by increasingly complex systems and by the evolution and automation of cyber attacks. Moreover, dynamic detection of attacks through *IDSs* (*Instrusion Detection Systems*) presents too many false positives to be effective. This work presents an approach on how to collect and normalize, as well as how to fuse and classify, security alerts. This approach involves collecting alerts from different sources and normalizes them according to standardized structures — IDMEF (*Intrusion Detection Message Exchange Format*). The normalized alerts are grouped into meta-alerts (fusion, or clustering), which are later classified using machine learning techniques into attacks or false alarms. We validate and report an implementation of this approach against the *DARPA Challenge* and the *Scan of the Month*, using three different classifications — SVMs, Bayesian Networks and Decision Trees — having achieved high levels of attack detection with little false positives. Our results also indicate that our approach outperforms other works when it comes to detecting new kinds of attacks, making it more suitable to a world of evolving attacks.

Keywords: IDS, Security, Correlation, Machine learning

1 Introduction

Protecting IT infrastructures against the attack of crackers is an increasing challenge in the present time, and promises to continue so in, at least, the near future. According to ISO/IEC 17799, there are a number of factors that augment this challenge: (1) IT infrastructures have become increasingly complex with the advent of new technologies (wireless and P2P networks, ever shrinking devices such as memory keys, cameras, etc); (2) Complex attacks have been productized and are available for download in the Internet; (3) Business challenges demand new services offered faster and on the net.

This trend is corroborated by Joosen at al [1] who state that the increasingly complexity and dynamicity of systems and applications is linked to the *Internet of Things*. Hale and Brusil [2] also raise the impact of virtualization and service-orientation on security management.

In this context, classic defense techniques, such as firewall-based architectures and stand-alone IDSs, are no longer effective to protect the IT infrastructure, as noted by Ganame et al [3]. A more modern approach requires the cooperation of many IT security devices, with an emphasis on IDSs. Unfortunately, IDSs generate a high number of events (Perdisci et al [4]) and many false positives, making it difficult to determine which real attacks caused some subset of those events. Therefore, managing all the alerts generated by these devices overwhelms the security staff of most organizations, as observed by Ning et al [5], Boyer et al [6], Julisch [7] and Liu and Zang [8].

The difficulty imposed by IDSs generating a high number of alarms has been raised by many authors, such as Sabata and Orneds [9], Chyssler et al [10], among others. In this same line of research, Ohta et al [11] defend the necessity of decreasing false positives in order to reduce the cost of operation and increase the reliability of the system.

To tackle the excess of false positives, we propose a machine learning solution based upon two major concepts: (1) event fusion into meta-events: collecting, normalizing and fusing together events that are likely to be part of the same attack; (2) classification of meta-events:

*Correspondence: kleber.stroeh@icarotech.com
[1] Icaro Technologies, Campinas, SP, Brazil
Full list of author information is available at the end of the article

based upon the attributes of a meta-event, decide whether it represents an attack or a false alarm, using different machine learning techniques.

An implementation of this approach was tested against the DARPA Intrustion Detection Evaluation 1999 [12] and the Scan of the Month 34 from the Honeynet Project [13]. The results indicated that this approach can provide high levels of detection with lower levels of false positives. It also provides the ability to detect some new kinds of attacks (for which there was no previous information in the testing data), outperforming other works in this aspect. This is an important feature, as it makes our solution more flexible and suitable to an ever changing scenario of cyber attacks.

The contribution of the paper is proposing an approach to correlating security alerts based on machine learning techniques, a security event taxonomy and the fusion of different alerts into meta-alerts. We did experiments that verify how effective this approach is to detect real attacks while generating low levels of false positives, and how good it is in detecting new and stealthy attacks when compared to other works.

This work sets apart from most of the applied machine learning techniques on IDSs, once we do not analyze network traffic or series of system calls. Instead, we process alerts generated by sensors (IDSs such as Snort, applications, etc). By doing so, we work on a different level of abstraction, that carries more meaning to our machine learning approach, helping bridge the "semantic gap" (Paxson and Sommer [14]). We also apply specific security concepts to our fusing technique, which renders meta-alerts that are more relevant. These meta-alerts, appropriately filtered by machine learning algorithms, help reduce the "high cost of errors" [14] that overwhelm security groups nowadays.

This document is organized as follows: Section 2 presents related works in security alert correlation; Section 3 describes our approach to correlating alerts, including the layers of processing, a security alert taxonomy and an algorithm to fuse alerts into meta-alerts; Section 4 presents the experiments used to test our approach and the corresponding results; finally, Section 5 summarizes the conclusions derived from this work and indicates possible future works.

2 Related work

Ever since the Lincoln Laboratory at the Massachusetts Institute of Technology launched the DARPA Intrusion Detection Evaluation 1998 and 1999 [12], many teams around the globe have put efforts into the development of IDS technology. As a matter of fact, this very work (by the Lincoln Laboratory) became the main data set reference for intrusion detection systems testing ever since.

Some research groups focused on misuse detection systems. Bowen et al, for example, developed a domain-specific language, called BSML (behavioral monitoring specification language), to specify relevant properties for intrusion detection [15]. A similar approach is presented by the STAT Tool Suite, where a machine state-oriented language - STATL - is used to describe attack scenarios [16]. Systems based on misuse detection tend to be more precise in pinpointing specific attacks, generating less false-positives, while presenting more difficulty to detect previously unknown attack patterns.

On the other hand, other teams, like Lee et al, have used anomaly detection techniques for intrusion detection [17]. This particular team proposed the use of data-mining techniques and its concepts (accuracy, efficiency, usability, ROC, ensembles, among others) to the IDS challenge. Solutions based on anomaly detection present the ability to potentially detect new attack patterns at the cost of a higher false-positive detection rate.

Hybrid approaches, encompassing anomaly and misuse detection, have also been proposed, as in EMERALD [18], where a building-block architectural strategy hosts potentially different correlation, inference and reasoning systems, varying from signature engines to Bayesian analysis.

Interest in intrusion detection systems can also be perceived in Europe, where Safeguard (European project for information security in telecommunications and electricity networks) has fueled research in IDS technology. Chyssler et al propose a framework for SIEM (Security Information and Event Management), comparing the possible use of Neural Networks, K-Neighbours and Naive Bayes in detecting attacks [10]. Ganame et al [3] propose a distributed architecture named DSOC (Distributed Security Operation Center) to improve the detection of more complex attacks such as coordinated ones.

In a military and defense context, Grimaila et al [19] also propose a distributed approach to security event correlation in order to "identify potential threats in a timely manner". Rieke and Stoynova [20] present a blueprint of an archtitecture for predictive security analysis that uses process models in extension to security policies and models.

One of the most inspiring works in this area has been the probabilistic alert correlation presented by Valdes and Skinner [21]. This work, developed at SRI Internation, proposes a hierarchy of correlations. Security alerts - detected by sensors spread over a network - are fused into meta-alerts at three different processing levels: (1) intra-sensor or synthetic threads - alerts from a single sensor are fused together according to a high minimum expectation similarity on the sensor itself, the attack class, and source and target IPs; (2) security incidents - alerts that belong to the same class and target the same IP are fused together

despite the expectation of similarity on the sensor identifier; (3) correlated attack reports: multistage attacks are possibly detected by relaxing the minimum expectation in similarity on the attack class.

Detecting multistage attacks is also handled by Ning et al, where prerequisites and consequences of attacks are represented as predicates of first order logic [5]. Alerts that represent different stages of a single attack are fused together into hyperalerts. This concept diminishes the number of alerts to be analysed by the security team.

Julisch proposes reducing the number of alerts by using clustering techniques [7]. He observes that a few dozens of rather persistent root causes generally account for over 90% of the alarms that an IDS triggers. He suggests the use of generalized attributes for an offline process of alarm clustering. This process indicates the root causes of alarm storms, enabling a security officer to tackle these causes and reduce the amount of alarms.

Asif-Iqbal et al [22] address the clustering of security logs by using data mining techniques, where the clustered logs are further filtered to remove unneeded entries. Corona et al [23] present a broad review on information fusion (clustering) for computer security.

Burroughs et al propose applying BMHT (Bayesian Multiple Hypothesis Tracking) onto fusing sensor process output in order to achieve situational awareness and allow security teams to respond to attacks more quickly [24]. Sabata, on the other hand, handles Bayesian Network fragments to correlate events and reduce the number of alerts analysed by security officers [9,25].

This work extends the concepts introduced by Julish [7] by having the algorithm run in real time instead of batch. It also does event correlation and fusion, like Valdes and Skinner [21], but it introduces hierarchies of event taxonomies to support the fusing process. This provides for a flexible, yet powerful, way of describing security events and meta-alerts. Deciding whether a meta-alert is an attack or not is implemented using machine learning techniques, similarly to Chyssler et al [10]; we, however, introduce more modern techniques such as SVMs and Bayesian Networks. Finally, not only do we compare our results to the ones of the DARPA challenge [12], but we also use real data from a honeynet to test our approach.

3 Proposed approach

Processing and correlating security events is key to an effective security management solution. Here, one seeks to achieve situational awareness, that is, the ability to analyze alerts in a broader and more holistic context.

Actually, a more precise and widely accepted definition of situational awareness is "the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" [26].

Put in simple words, the more different the evidences that a security incident is taking place, the more likely it is true. Thus, we want to collect events from different sources and normalize them, so that they can be processed homogeneously.

The normalized events are fused (or clustered) into groups, or meta-events. Meta-events present a more complete description of a possible attack scenario than single events. They constitute a more refined expression of the underlying attack when compared to isolated alerts.

Grouping events into meta-events leads to better situational awareness, improving the classification between real attacks and false alarms; it also enhances the performance of the system, as the classification operates on meta-events rather than events (events tipically outnumber meta-events by orders of magnitude).

Meta-events enable the contrast between new scenarios to previously learned attack scenarios. Machine learning techniques can be used to automate this process.

3.1 Hierarchical layers

Our approach encompasses three layers of processing stages: Collection (and Normalization), Fusion and Classification. Figure 1 shows the three layers.

Each stage provides a level of abstraction to the following one. Raw data are processed and transformed into standardized IDMEF [27] -like records by the Collection stage. The fusion stage takes these IDMEF-extended records and groups them into standardized meta-alerts. Finally, it is up to the Classification stage to take these meta-alerts and sort them into attacks and false alarms.

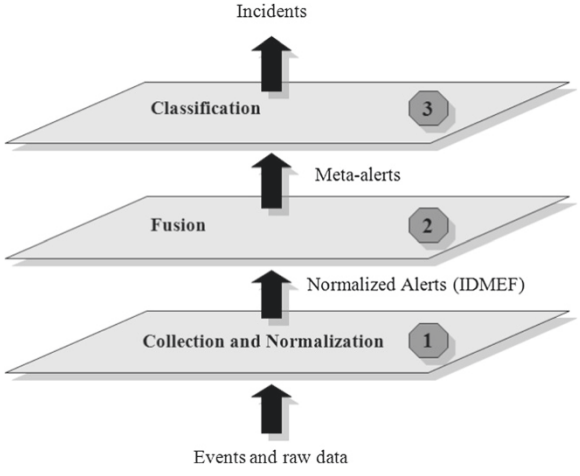


Figure 1 Processing in stages or hierarchical layers.

3.2 Collection and normalization

The Collection and Normalization layer is linked to the event sources: IDSs, firewalls, system logs, SNMP traps, and so on.

Collection elements are divided into two groups: passive and active. Passive elements gather alerts by the simple observation of evidences, events or element states. Examples of passive collection are the parsing of systems logs or the receival of SNMP traps. Active elements, on the other hand, interact with the managed objects in order to sinthetize alerts. Examples are ICMP probes or SNMP pollers.

Collection elements also differ on the cardinality of the generated alerts. In some scenarios, for each incoming event or element state (an SNMP trap, e.g.), there is an outgoing alert. In others, the collection element is responsible for filtering the incoming events, in order to generate fewer outgoing alerts. One typical example of the latter is the processing of operating system calls; only a subset of the logs of system calls correspond to potential threats.

Normalization guarantees that the outgoing alerts are represented in a standardized way. Our approach uses a linearized extension of the IDMEF standard. It corresponds to a string representation of the IDMEF record with minimum overhead. Overhead, here, meaning any character that does not relate to content data. Linearization provides for a better perfomance of the system when compared to the originally XML bus prescribed by the IDMEF standard, since the latter would be too heavy and resource consuming, as also noted by Ganame et al [3]. This is key, once it has to process millions of events a day in real time. Its extension regards the introduction of the new fields *ext_class*, *ext_src_node_addr_type*, *ext_tgt_node_addr_type*, *ext_priority* and *ext_taxonomy*. This scheme is similar to the one presented by Lan et al [28].

Field *ext_class* partitions the alert space into five groups: denial of service (DoS), probing (Probe), remote acess (R2L), superuser access (U2R) and data theft (Data). On the other hand, fields *ext_src_node_addr_type* and *ext_tgt_node_addr_type* classify, respectively, source and target nodes among: external, internal or pertaining to the Demilitarized Zone (DMZ).

Each alert is also given a priority in the *ext_priority* field. This attribute takes values from one (1) to three (3), where one is the highest priority, and three, the lowest; this scheme is analogous to the one employed by Snort [29]. Sensors usually fill this attribute with the information defined by its source. That is, whenever an alert source contains an attribute named severity or priority, their value is mapped onto the *ext_priority* field. There are, however, cases where the event source does not present its own view about the priority of the alert. In these cases, the sensor will define the priority based on the type of the given alert. This implies that the sensor developer needs to have a good understanding of security alerts in order to consistently define this attribute. Also notice that priorities are pre-defined by sensors or alert sources, being, therefore, immutable regarding its operating environment.

Arguably the most important extension to the model is the introduction of the *ext_taxonomy* field. Taxonomy is the subject of research for many years, dating back to the work of Debar et al [27] . These works have resulted in the genesis of the IDMEF model. Despite its importance, the IDMEF model lacks a semantic description of the types of its events. Al-Fedaghi and Mahdi [30] have also observed the need for categorizing security log entries [30].

We need a simple way of identifying alerts that provides support to the fusion and classification processes. We want to identify each alert with a single value that depicts what that alert means. The naming space of *ext_taxonomy* needs to be finite, so that it can be further applied in some of the algorithms of the classification layer.

To achieve these two requirements (expressiveness and finitude) we derived a scheme based upon the observation of real-world alerts and its expression in natural language. This provides for a method of describing event types that is meaningful, yet well-structured.

By observing alerts coming from different sources (IDSs, system calls, web server access logs, Windows security events and firewall events), we have divided them into three main groups:

1. An action against an object (<*object*>:<*action*>).
2. A condition of an object (<*object*>:<*condition*>).
3. A suspicion of the state of an object (<*object*>:<*state*>).

Every group is, therefore, characterized by an *object* plus an *action*, a *condition* or a *state*. *Objects* are represented in a multilevel hierarchy. The most top level element of an object is a system, a network element, a protocol, a user, a file or security. Each of these elements is then subdivided into more granular concepts. Figure 2 describes an example of a hierarchy of objects as used in our experiments. This hierarchy may be extended to suite other managed elements and objects as necessary.

An *action* resembles a method to which an *object* is associated. Therefore, an alert about the shutdown of the operating system would be represented as *system.os:shutdown*. As with objects, actions are also represented in a multilevel hierarchy. The lower levels of the hierarchy qualify the upper ones. Qualifications may be adverbs associated with the action, the result of the

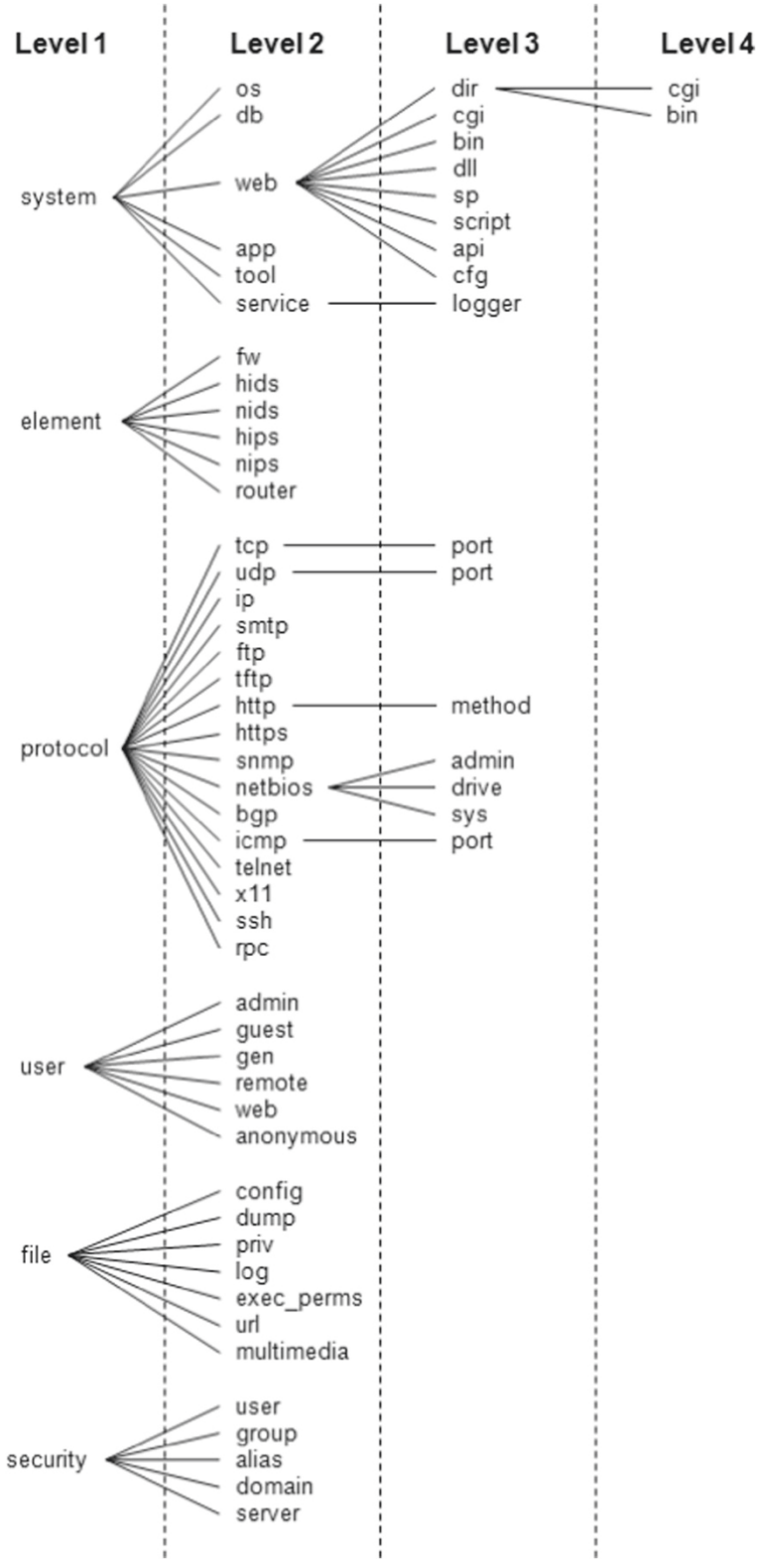


Figure 2 Hierarchy of objects.

action, or something similar. For example, a failure in the attempt to login as the administrator of a system — due to a wrong password — would be represented as *user.admin:login.fail.pwd*.

Actions are typically used to represent sentences of the following constructions:

- Subject + Intransitive Verb (+ Adverbial Clause);
- Transitive Verb + Direct Object (+ Adverbial 4 Clause).

The differentiation of both cases is dependent on the transitiveness of the verb. Nonetheless, this ambiguity does not present a practical impact in the use of the taxonomy. Solving this ambiguity would invariably make our notation more complex, without real gains to our approach.

Some alerts do not represent actions against objects, but states (or *conditions*) in which those objects are. These *conditions* are tipically represented by constructions like: Subject + Copulative Verb + Complement. For example, if one wants to represent that a TCP port is unaccessible, the taxonomy would render: *protocol.tcp.port: unreachable*.

Finally there is a category of alerts that represent suspicions towards a given activity or evidence. It differs from the previous cases in that it represents not a fact but deductions associated with facts. Given the inherent uncertainty, our taxonomy set *suspicions* apart from *actions* and *conditions*. An example would be the suspicion that there is an attempt to use a backdoor in a system. This would be described as: *system.os:malware.backdoor.attempt*.

Note that *objects*, *actions*, *conditions* and *suspicions* are all represented in multilevel hierachies. This allows correlations to operate on the level of granularity that is more convenient in each scenario. For example, if it is irrelevant to check if a login failure occurred due to an invalid user or an invalid password, than, the following taxonomies would be equivalent: *user.admin:login.fail*, *user.admin:login.fail.pwd*, and *user.admin:login.fail.user*. The use of hierarchical levels addresses this issue in an elegant manner, by making it possible to completely ignore the third level of the action hierarchy, reducing all the previous options simply to: *user.admin:login.fail*.

The result of this scheme is the ability to assign a single finite description to each security alert. This description is meaningful (is it was derived from real alerts), well-structured (use of hierarchy and natural language concepts) and finite.

Another advantage is that it establishes a new dictionary to be used when classifying alerts. Therefore, it does not matter whether the original source uses the term login, logon or begin_user_session; it does not matter if the original system classified the login as failed or denied; it also doesn't matter if the administration user is called root, superuser or administrator; in our taxonomy (and system), the fact that an administrator user tried to access a system and failed will always be described as *user.admin:login.fail*.

Therefore, our taxonomy provides the final tool to normalize events coming from different sources while still maintaining a proper, meaningful, description of the alert type.

Since our proposed taxonomy conforms to the structure we have outlined above (specially derived from natural language), developers can more easily associate an alert to an existing taxonomy entry or, better, create new entries to address new alert types that take place in his/her specific environment.

3.3 Fusion or clustering

This layer takes alerts provided by the collection and normalization layer and groups them into meta-alerts. This clustering process is an extension of the concepts introduced by Valdes and Skinner [31]. Its design fulfills the following requirements:

1. Efficience: operate under the shortest subset of alert attributes and limit searches to meta-alerts in time.
2. Intelligibility: the resulting meta-alerts must make sense to human beings.
3. Coherence: clustering mechanism must adhere to the types of attacks supported by the model.

We decided on using a deterministic, straightforward, algorithm to fusion. It enables the clustering of the alerts in real time as they are made available by the undelying layer. The resulting meta-alerts are simple and intelligible to human beings, as shown in Table 1.

Arguably, the most important field in the meta-alert structure is the *alert_taxonomy_set*. This is a bit array that represents each of the supported alert types of the taxonomy. If one (or more) alert(s) of a given type is (are) present in the meta-alert, its corresponding bit is set. Otherwise, it remains clear.

Think of this bit array as the set of all the possible clues in a crime scene. Potential attacks (or crimes), depicted by meta-alerts, present a subset of these clues. The more a set of clues resembles a previous known scenario, the more likely it is to be a real attack.

The use of an array bit provides support to internal product techniques that verify the similarity between two scenarios in the classification layer.

Filling in the meta-alert record is a natural result of the clustering algorithm below:

Require: $j\_cache > 0, j\_dos > 0, j\_probe > 0, j\_r2l > 0, j\_u2r > 0$
Require: $a \in Alertas$, M set of Meta-Alerts
Ensure: M updated with alert a

```
1: f ← ∅
2: for all m in M do
3:    if a.ext_class ≠ m.ext_class then
4:       nop
5:    else if a.ext_class = DoS then
6:       if (a.src_node_addr ∈ m.src_network_addr) and
         (a.tgt_node_addr ∈ m.tgt_node_addr_list) and
         (a.create_time ≤ (m.end_time + j_dos)) and
         (a.create_time ≥ (m.init_time − j_dos)) then
7:          f ← m
8:       end if
9:    else if a.ext_class = Probe then
10:      if (a.src_node_addr ∈ m.src_node_addr_list) and
         (a.create_time ≤ (m.end_time + j_probe)) and
         (a.create_time ≥ (m.init_time − j_probe)) then
11:         f ← m
12:      end if
13:   else if a.ext_class = R2L then
14:      if (a.src_node_addr ∈ m.src_node_addr_list) and
         (a.tgt_node_addr ∈ m.tgt_node_addr_list) and
         (a.create_time ≤ (m.end_time + j_r2l)) and
         (a.create_time ≥ (m.init_time − j_r2l)) then
15:         f ← m
16:      end if
17:   else if a.ext_class = U2R then
18:      if (a.src_node_addr ∈ m.src_node_addr_list) and
         (a.tgt_node_addr ∈ m.tgt_node_addr_list) and
         (a.create_time ≤ (m.end_time + j_u2r)) and
         (a.create_time ≥ (m.init_time − j_u2r)) then
19:         f ← m
20:      end if
21:   end if
22: end for
23: if f = ∅ then
24:   n = new Meta-Alert
25:   init n with data from a
26:   a.meta_alert = n
27:   M = M ∪ {n}
28: else
29:   a.meta_alert = f
30:   Update f with data from a
31: end if
```

Alerts are grouped according to their timestamps and the time windows defined at the requirement lines. This defines sliding windows within which correlated alerts are fused.

Once an alert is received, the algorithm iterates over a cache of meta-alerts looking for potential candidates for fusion. Alerts are only fused together if they share the same alert class. Further requirements stand for each individual alert class. In the case of denial of services (DoS) attacks, the target address of the alerts must match in order for their alerts to be fused together. If it is a probing attack, then it is the source address that must remain unchanged.

Table 1 Meta-Alert record

Attribute	Type	Description
meta_alert_id	Integer	Unique identification of a meta-alert
analyzer_id_list	String	List of sensors that generated the alerts of this meta-alert
analyzer_count	Integer	Number of sensors that detected the alerts
init_time	Date	Timestamp of the oldest event in this meta-alert
end_time	Date	Timestamp of the most recent event in this meta-alert
time_window_len	Integer	Number of seconds between end_time e init_time
src_network_addr	String	Base address of the network that originatd the alerts
src_node_addr_list	LongText	List of the addresses that originated the alerts
src_node_count	Integer	Number of different addresses that originated the alerts
src_user_id_list	String	List of user identifications that originated the alerts
src_user_count	Integer	Number of users that originated the alerts
src_proc_id_list	String	List of process identifications that originated the alerts
src_proc_count	Integer	Number of processes that originated the alerts
tgt_node_addr_list	LongText	List of target addresses
tgt_node_count	Integer	Number of different target addresses
tgt_port_list	LongText	List of target ports
tgt_port_count	Integer	Number of different target ports
tgt_user_id_list	String	List of target user ids
tgt_user_count	Integer	Number of different target user ids
tgt_proc_id_list	String	List of target process ids
tgt_proc_count	Integer	Number of different target process ids
tgt_file_name_list	LongText	List of target file names
tgt_file_count	Integer	Number of different target file names
ext_class	String	Attack class of the meta-alert
ext_max_priority	Integer	Highest priority amongst the alerts in the meta-alert
alert_count	Integer	Number of alerts in the meta-alert
alert_taxonomy_set	BitArray	One bit set per alert type present in meta-alert

If a matching meta-alert is found, the alert is fused into it. Otherwise, a new meta-alert is synthetized with the information from this single alert.

This simple algorithm has presented excellent results in the Data to Information Ratio (DIR), and provides a good support to the classification layer, as we will show later.

There is a price to be paid for the simplicity, though. More refined attributes for fusion, such as sessions, users, process and files are left behind. The algorithm does not support the fusion of alerts pertaining to hybrid or multistage attacks, and we left these scenarios as future work.

3.4 Classification

Separating meta-alerts that represent attacks from false alarms is an ideal task for machine learning techniques. Here, we verify how modern techniques, such as SVMs and Bayesian Networks, behave within the classification layer.

Choosing the subset of the attributes of a meta-alert to be used by these techniques is key. One must avoid choosing too many attributes in order to avoid overfitting. From the list of attributes presented in Table 1, we have selected the following: *alert_count*, *ext_max_priority*, *tgt_node_count*, *tgt_port_count*, *analyzer_id_list*, *ext_class*, and *alert_type_set*.

Some attributes play important roles in the segregation between real attacks and false alarms. The number of alerts, nodes and port help detect DoS and Probe attacks. The list of sensors (*analyzer_id_list*) helps improve our situational awareness; the more sensors have detected traces of an attack, the more likely it is to be real.

However, it is the bit array representing entries of our taxonomy that plays the most central role in the

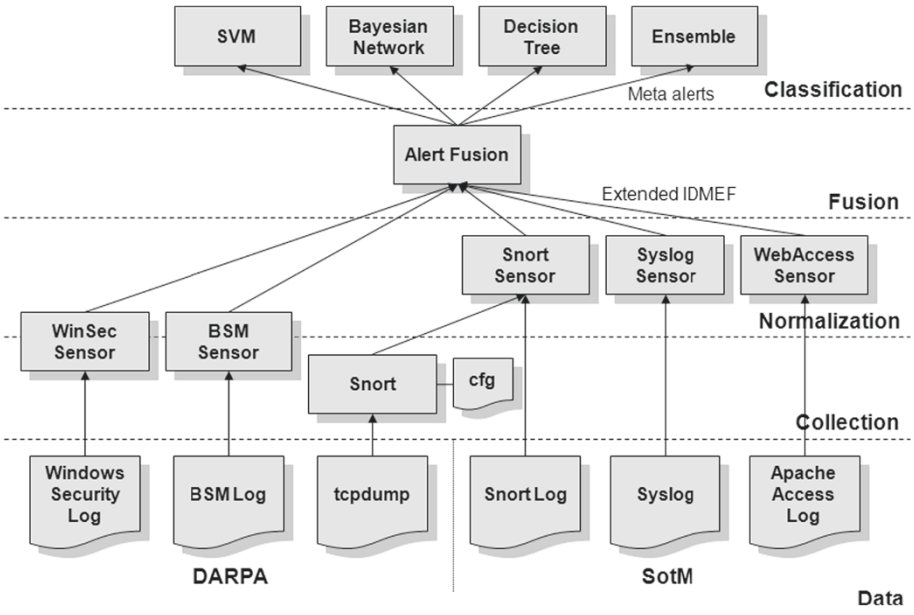


Figure 3 Model instantiation.

classification. This bit array is analogous to the *bag of words* technique used for comparing text documents. The internal product of two arrays represents the degree of similarity between two documents, or, in our case, two meta-alerts. This concept is very natural to SVMs, but has proven very succesful with Bayesian Networks and Decision Trees as well.

This approach was proven very flexible and resilient in the detection of new attacks. Naturally, the result is dependent upon the granularity of the taxonomy. If it is too specific, the learning algorithm might end up overfitting. If it is too generic, the classification is jeopardized by the lack of information in the decision making process.

4 Implementation and experiments

In order to test our approach, we had it implemented and tested against two major, publicly available, data sources: (a) the DARPA challenge [12], the *de facto* benchmark for IDS testing; and (b) the *Scan of the Month* (SotM) from the Honeynet Project [13]. These two sources have different and complemental characteristics.

The DARPA challenge was conceived by DARPA (Defense Advanced Research Project Agency) in association with MIT Lincoln Labs. It simulates an American Air Force base local network that has been attacked in very specific ways. Information about the traffic on the LAN and about the attacks is provided in the form of tcpdump files, BSM files, audit logs, Windows alerts and other

Table 2 DARPA event sources

		Analyzer			
		Int. Snort	**Ext. Snort**	**BSM**	**Windows**
Week 1	Records	142,674	143,098	2,063,809	581,192
	Alerts	142,674	143,098	846	2,953
Week 2	Records	47,405	47,826	2,151,011	3,650,045
	Alerts	47,405	47,826	728	405
Week 3	Records	18,742	21,687	2,147,384	3,574,791
	Alerts	18,742	21,687	10,752	419
Week 4	Records	17,169	23,032	1,841,269	2,292,926
	Alerts	17,169	23,032	701	643
Week 5	Records	34,652	53,612	2,949,363	2,476,508
	Alerts	34,652	53,612	912	852

sorts of audits. This set encompasses five weeks of activities, and has been used by many IDS research groups, such as: Ohta et al [11], Mahoney and Chan [32], Bowen et al [15], Mukkamala et al [33], Faraoun and Boukelif [34,35], Tandon and Chan [36], Lee et al [17], Mukkamala and Sung [37], Valdes and Skinner [31], and Sabata and Ornes [25].

There has been some controvery regarding the use of the DARPA dataset. Sommer and Paxson [14] state: "The two publicly available datasets that have provided something of a standardized setting in the past - DARPA ... KDD ... - are now a decade old...". On the other hand, Perdisci et al [4], states: "Even though the DARPA 1999 dataset has been largely critizicized, it is the reference dataset in the evaluation of IDS performance".

Our approach is less sensible to the age of the dataset, as it relies on alerts derived from the network traffic (by sensors such as Snort), instead of the network traffic itself. Therefore, the changes that have taken place in the characteristics of network traffic are masqueraded by our sensors and have a lesser impact on the effectiveness of our algorithms.

In order to further compensate for the limitations of the DARPA dataset, we decided to add a second dataset to our tests. The SotM data is provided by the *The Honeynet Project* [13] and provides information about real attacks that have been perpetrated against a honeynet controled by the project. This data set complements the first one, as it was originated in a real environment, and as it contains newer types of attacks. Forms of data include webserver logs, syslog, Snort logs and ipTables logs. Since they correspond to real attacks against honey pots, the noise ratio (data not related to attacks) is lower in these logs.

This is a newer dataset and a more real one (not synthesized). On the other hand, it is certainly a smaller one. As we will show later, our implementation has achieved even better results with this dataset.

4.1 Implementation

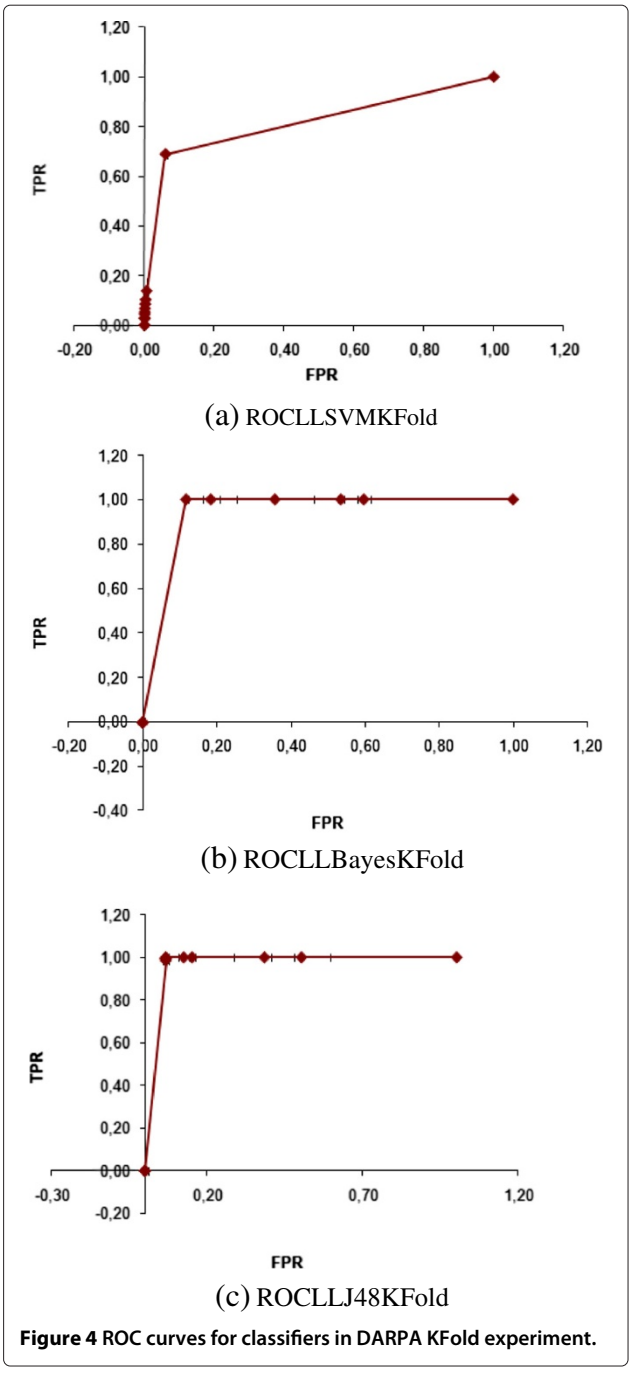
We implemented our model using Perl and three different machine learning techniques. The implementation handles the data sources of our test environments. Figure 3 shows a simplified view of the implementation.

Table 3 Event reduction in DARPA experiment

Entity	Quantity
Records	24,278,195
Alerts	569,108
Meta-alerts	19,550
Indicated attacks	268
Data to information ratio	2,124

Basically, for each data source there is one module for collection and normalization, namely: Windows Security Log, BSM Log, tcpdump and Snort Log, Syslog and Apache Access Log. These modules handle the specifics of the data source and generate alerts in extended IDMEF format.

The AlertFusion module clusters the normalized data using the algorithm described in Section 3.3, generating meta-alerts. The classification layer takes the resulting



(a) ROCLLSVMKFold

(b) ROCLLBayesKFold

(c) ROCLLJ48KFold

Figure 4 ROC curves for classifiers in DARPA KFold experiment.

meta-alerts and processes them using three different machine learning algorithms: SVM, Bayesian Network and a Decision Tree.

For the SVM classifier we used the SVM implementation from libSVM [38]. We discretized attributes *alert_count*, *tgt_node_count* and *tgt_port_count* into four different values in order to reduce the sensitiveness of the algorithm to the learning data. Basically we do not want the algorithm to overlook a DoS attack composed of 200 alerts, just because it was previously trained with a scenario that contained 500 alerts. We also linearized attributes *ext_class* and *alert_type_set*; we mapped each possible value of the first and bit of the latter onto a different dimension (*dummy variables*). We optimized the resulting SVM using the guidelines proposed by Hsu et al [39]. Finally, we induced the learning process using different weights for false positives and false negatives; basically, a false negative had a higher cost than a false positive.

We implemented the Bayesian Network classifier and the Decision Tree using Weka [40]. We coded the first using *weka.classifiers.bayes.BayesNet*, while in the latter we used *weka.classifiers.trees.J48*, which implements algorithm C4.5. We also implemented the assimetry of costs described for the SVM classifier using *weka. classifers.meta.CostSensitiveClassifier*.

4.2 Experiments

We have performed three experiments:

1. DARPA kFold: using the DARPA database, we tested the classifiers using a *5-fold cross validation*.
2. DARPA 3x2: using the DARPA database, we trained our classifiers with the three weeks of tagged attacks, and tested their performance against the remaining two weeks.
3. SotM kFold: using the SotM database, we tested the classifiers using a *5-fold cross validation*.

4.2.1 *Experiment 1: DARPA kFold*

Table 2 shows the result of the collection and normalization processes on the DARPA data.

The Snort Sensor is responsible for most of the collected alerts. In this sensor, every log entry corresponds to an alert. This is a passive sensor, with an 1x1 cardinality. For the BSM and Windows Sensors, we verify the results of the applied heuristics, where only a subset of the records is promoted to alerts. These sensors act as filters for the upper fusion layer.

As a consequence, from a domain of over 24 million records, sensors produce less than 570 thousand alerts.

Table 3 shows the performance of the Fusion layer applied to the DARPA database. As you can see, the implementation has reached a *Data to Information Ratio* of 2,124, a value higher than the ones obtained by Valdes and Skinner [21], and by Sabata and Ornes [25].

The classifiers were first tuned based upon cost assimetry, using different *weights* for false positives and false negatives. In average, we have found that an assimetry of 1:60 was the best solution for the classifers. These so-tuned classifiers returned the results summarized in the ROC curves depicted in Figure 4.

One can see that both the Bayesian and the Decision Tree classifiers have presented good results, giving rise to very steep inflexions in the graph. This translates into achieving high values in TPR for low values in FPR. That means more true positives and true negatives, for fewer false positives and false negatives.

This can also be observed in Figure 5. There one can see how the Bayesian Network classifier performs in detecting specific types of attacks as our sensitiveness variable is changed. High levels of detection have been achieved in all categories, with a special remark for DoS attacks. The detection of R2L attacks is the one that is more impacted by the change in the sensitiveness of the classifier.

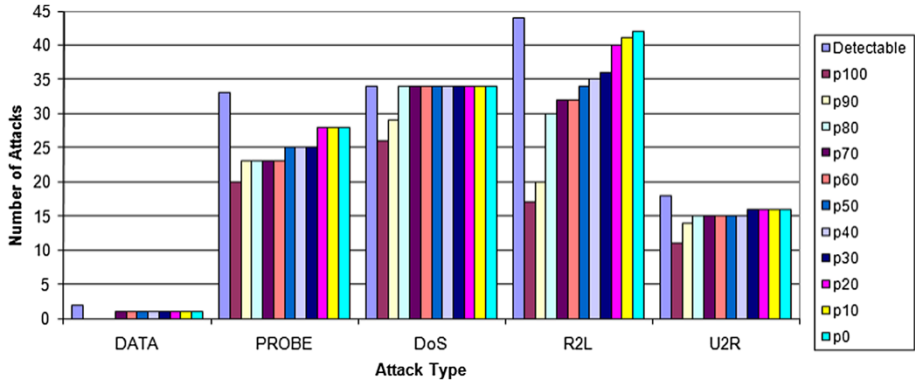


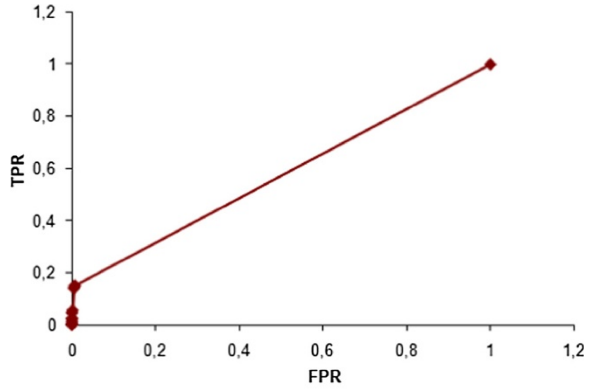
Figure 5 Detection by attack type using Bayesian network in DARPA KFold experiment.

4.2.2 Experiment 2: DARPA W3xW2

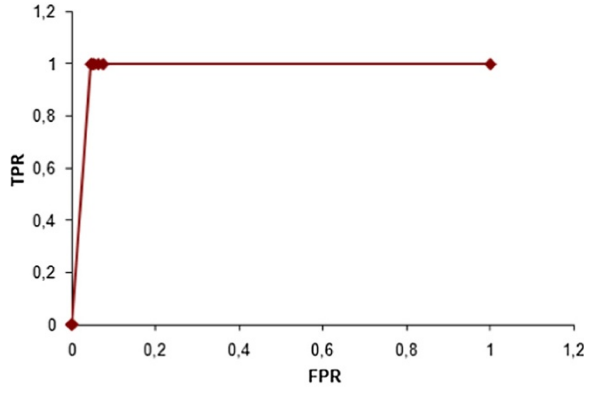
Performing the experiment against the two remaining weeks, we have achieved the results in Figure 6.

The conformance of the curves summarizes the good performance of the classifiers.

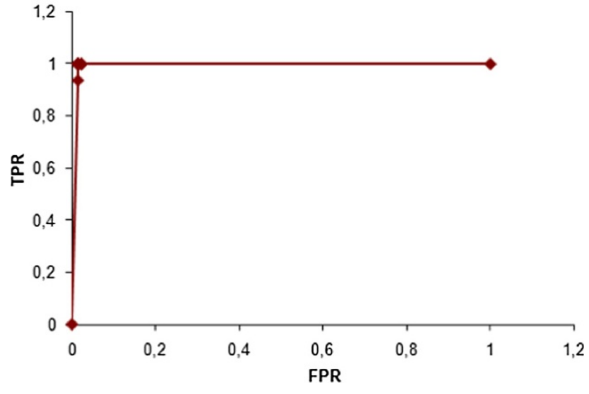
The combination of a generic taxonomy and the use of these classifiers have demonstrated the possibility of detecting new attacks, for which there had been no samples in the training data. This can also be seen in Figure 7.



(a) ROCLLSVMW3xW2



(b) ROCLLBayesW3xW2



(c) ROCLLJ48W3xW2

Figure 6 ROC Curves for classifiers in DARPA W3xW2 experiment.

The implementation managed to identify most of the detectable attacks in the data set provided by DARPA. We consider an attack detectable if some evidence of its existence could be tracked in the alerts provided to the Snort and other sensors used in our experiments.

Our implementation managed to detect attacks that were not detected by any of the contenders of the initial DARPA challenge, like *ipsweep*, *queso*, *snmpget* and *ntfsdos*. We also achieved a higher identification rate for *lsdomain*, *portsweep*, *ncftp* and *netbus*. Table 4 shows the results obtained for these attacks when compared to the results of the best system of the DARPA Challenge [12].

The results show that our approach improves the detection of new types of attacks, specially probe and R2L ones. We also improved the detection of Windows-related attacks.

4.2.3 Experiment 3: SotM kFold

To complement our study, we tested our implementation against a second data base: the *Scan of the Month 34* [13].

This database reflects real attacks performed by crackers against a honey net. Table 5 shows the sources used in the experiment and the results from the Collection and Normalization Layers.

Using the SotM database, we have achieved a lower, although good, Data to Information Ratio, as shown in Table 6. This reflects the fact that this database was collected at a honeynet, that typically presents less noise in the alerts.

The performance of the classifiers with this real-world attacks has been remarkable, as presented in Figure 8. Once again, high levels of true positives and true negatives have been achieved with low levels of false positives and false negatives.

These results derive from the good behaviour of the classifiers, specially the SVM and Decision Tree, in a more homogeneous attack environment. The DARPA data have been synthesized to emulate more complex attacks than the ones found on a daily basis in the real world. With the SotM, we have a data sample that is closer to the behaviour of an average cracker.

This scenario makes it easier for our machine learning techniques to learn patterns and apply them to the new attacks. Therefore, our results are even better in this more realistic scenario.

As shown in Figure 9, the Bayesian Network classifier was able to detect every single attack, with the exception of the *Probe* attack class.

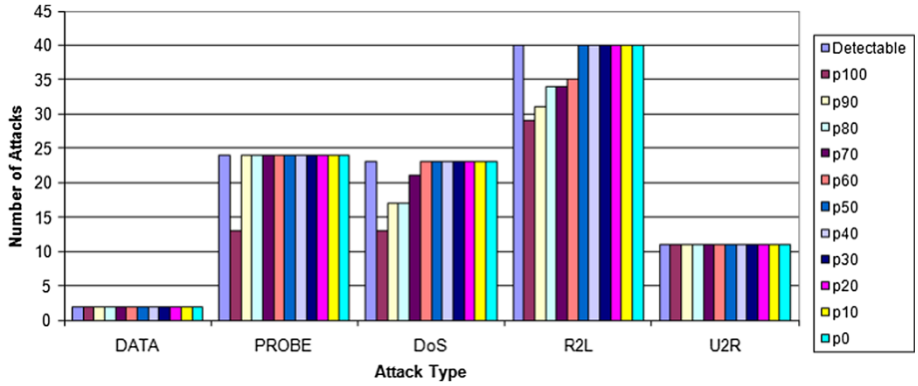


Figure 7 Detection by attack type using Bayesian network in DARPA W3xW2 experiment.

4.2.4 A comparative analysis

This work encomprises one of the largest testbeds in intrusion detection systems. We used two datasets (DARPA and SotM), several event sources and three different classification techniques.

Most of the published works in intrusion detection use only a subset of the event sources available in the DARPA challenge [12]. Others process only a portion of the data volume available: Faraoun and Boukelif use 10% of the DARPA dataset [34,35]; Mukkamala et al use 20% of the same dataset [33]. Our experiments made use of the five weeks of data from the DARPA challenge, spanning from the network data to the specific Solaris and Windows data sources.

Our results indicate higher levels of detection for new and stealthy attacks — *ipsweep*, *queso*, *snmpget*, *ntfsdos*, *lsdomain*, *portsweep*, *ncftp*, *netbus* — when compared to the contenders of the DARPA challenge, according to Lippmann et al [12], as depicted in Table 4. This is a consequence of the combined use of good classifiers (specially the Bayesian Network) and an effective taxonomy.

Detection rates were also better for the Probe attacks against Linux boxes, and for R2L attacks against Solaris servers. Our implementation also detected more R2L and U2R attacks against Windows systems than the best implementation in the DARPA challenge [12]. This can verified in Table 7.

As noted by Kayacik and Zincir-Heywood, the number of false positives generated by a Snort IDS, using state of the art rules, is very high [41]. In all our experiments, this module was the one responsible for most of our alerts. Without the fusion and classification techniques we applied, it would be impossible for a security operations group to process the events generated by a Snort system in this busy network. This would also be the case in any real-world, enterprise-class, network.

Given the attacks for which there were evidences (detectable attacks), our approach achieved higher detection rates than the ones presented by Lippmann et al [12], even when the testing attacks did not match the ones used for training. Detection rates in this scenario approached 100% with a limited amount of false positives (Figure 6).

5 Conclusions

Information security remains an unsolved challenge for organizations. Old approaches, like perimeter defense, are no longer effective in a new scenario of ever growing threats and rapidly changing attack patterns.

Table 4 Detection of new and stealthy attacks

Attack name	Category	Details	Total instances	DARPA challenge	Bayesian classifier
ipsweep	Probe	Stealthy	4	0	4
lsdomain	Probe	Stealthy	2	1	2
portsweep	Probe	Stealthy	11	3	8
queso	Probe	New	4	0	4
ncftp	R2L	New	5	0	1
netbus	R2L	New-windows	3	1	3
snmpget	R2L	Old	4	0	4
ntfsdos	U2R	New-windows	3	1	3

Table 5 SotM event sources

Analyzer	Records	Alerts
Snort	69.039	69.039
Web logs	3.554	3.414
Syslog	1.158	953
Total	73.751	73.406

Facing this reality requires a more flexible and dynamic approach, that provides quick responses to incidents. Detecting incidents in real time is a challenge itself, once there are millions of events to be handled, and many of them correspond to false positives, that drain the energy and time of a security team.

This work holds the following contributions to this research area: (a) suggesting the use of a taxonomy to better classify security alerts - and providing one for the scope of our datasets; (b) applying a clustering mechanism to security alerts; (c) experimenting different machine learning techniques on top of these meta-alerts – the ones provided by the clustering mechanism–, specializing the classification module suggested by Perdisci et al [4].

Our work differs from most of the applied machine learning techniques on IDSs, as we do not analyze network traffic. Instead, we rely on alerts provided by sensors (IDSs such as Snort, logs, etc). Therefore, we work on a different level of abstraction, that carries more semantics to our machine learning approach. We believe this addresses part of the "semantic gap" raised by Sommer and Paxson [14]. It also seems to render us more independent on the "diversity of network traffic" (also criticized by Sommer and Paxson [14]) as it is no longer our subject of analysis. We also don't rely on "anomaly-detection", which is the technique that has been heavily criticized by Sommer and Paxson [14]. We chose to follow the classification path of machine learning, which seemed more natural and more semantically relevant. Finally, the fact that we provide meta-alerts composed of the underlying alerts also addresses the "high cost of errors" brought up by Sommer and Paxson [14], as we drastically reduce cardinality (1000x less meta-alerts than alerts to classify), properly classify them (low level of false positives) and provide the underlying alerts of a meta-alert for faster analysis (more actionable piece of information).

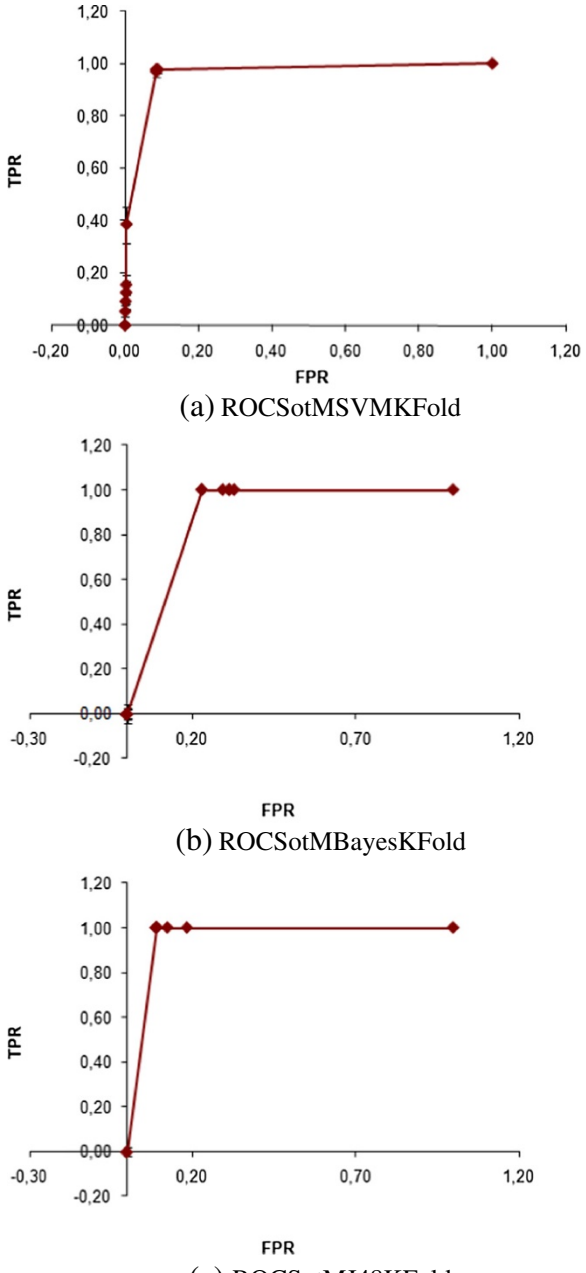


Figure 8 ROC Curves for classifiers in SotM KFold experiment.

Table 6 Event reduction in SotM experiment

Entity	Quantity
Records	73,751
Alerts	73,406
Meta-alerts	8,528
Indicated attacks	1,469
Data to information ratio	50

We have been able to verify that the combination of a taxonomy and the fusing of alerts into meta-alerts provides a solid basis for the correlation of alerts. Using data mining techniques, specially Bayesian Networks, has also demonstrated to be a good approach to detecting new and stealthy attacks.

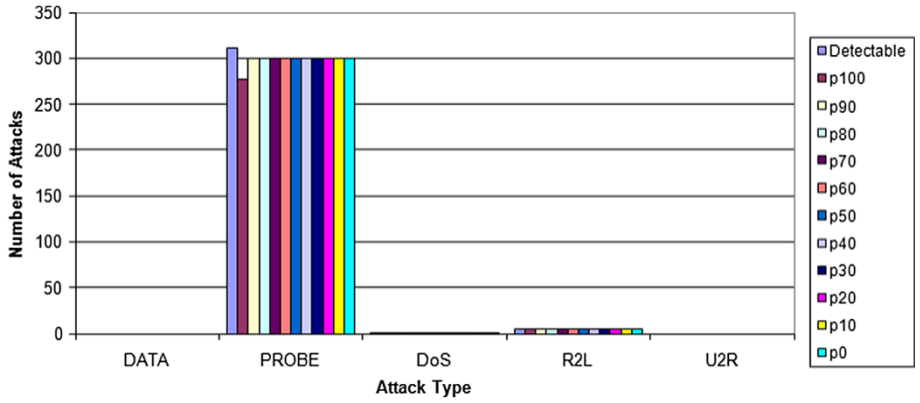


Figure 9 Detection by attack type using Bayesian network in SotM KFold experiment.

Our experiments have shown that it is possible to achieve high levels of Data to Information Ratio, and still be able to detect most of the attacks in our testing data. On top of that, new attacks — some that were not in our training datasets — have been detected by our combination of taxonomy–fusion–machine learning approach.

High detection rates have been achieved with low levels of false positives (lower than the thresholds suggested by the DARPA challenge [12]), making our approach a viable option for use in a scenario of intense, ever changing, attack types that constitute our reality and the near future.

This work can be extended in several dimensions; notably:

- Allow meta-alerts to be clustered into higher level meta-alerts, providing for the fusing of distinct attack types and multi-stage attacks;
- Research the use of *incremental clustering* as an alternative to the alert fusion algorithm;
- Analyze the possibility of using pre-defined topologies in the Bayesian Network classifier, in order to provide more deterministic and easier-to-understand classification criteria;
- Extend the taxonomy for different domains, including physical security;
- Enhance the approach to integrate service level management practices, using CMDBs (*Configuration Management Data Base*) to analyse the impacts of security incidents.

Table 7 Comparison regarding the detection rate in different operating systems and attack types

OS	Attack type	Best in Darpa [12]	Bayesian classifier
Linux	Probe	60%	78%
Solaris	R2L	50%	67%
Windows	R2L	< 40%	78%
Windows	U2R	< 40%	50%

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
KS created and developed the approach. KS, ERMM and SKG participated in the experiments. KS, ERMM and SKG wrote the manuscript. All authors read and approved the final manuscript.

Acknowledgements
The authors would like to thank FAPESP and CNPq for the financial support.

Author details
[1]Icaro Technologies, Campinas, SP, Brazil. [2]IC - Institute of Computing UNICAMP, University of Campinas, Campinas, SP, Brazil.

Received: 6 February 2013 Accepted: 6 February 2013
Published: 11 March 2013

References

1. Joosen W, Lagaisse B, Truyen E, Handekyn K (2012) Towards application driven security dashboards in future middleware. J Internet Serv Appl 3: 107–115. 10.1007/s13174-011-0047-6
2. Hale J, Brusil P (2007) Secur(e/ity) management: A continuing uphill climb. J Netw Syst Manage 15(4): 525–553
3. Ganame AK, Bourgeois J, Bidou R, Spies F (2008) A global security architecture for intrusion detection on computer networks. Elsevier Comput Secur 27: 30–47
4. Giacinto G, Perdisci R, Roli F (2005) Alarm clustering for intrusion detection systems in computer networks. In: Perner P, Imiya A (eds), vol 19. pp 429–438
5. Ning P, Cui Y, Reeves DS, Xu D (2004) Techniques and tools for analyzing intrusion alerts. ACM Trans Inf Syst Secur (TISSEC) 7: 274–318
6. Boyer S, Dain O, Cunningham R (2005) Stellar: A fusion system for scenario construction and security risk assessment. In: Proceedings of the Third IEEE International Workshop on Information Assurance. IEEE Computer Society, pp 105–116
7. Julisch K (2003) Clustering intrusion detection alarms to support root cause analysis. ACM Trans Inf Syst Security 6: 443–471
8. Liu P, Zang W, Yu M (2005) Incentive-based modeling and inference of attacker intent, objectives, and strategies. ACM Trans Inf Syst Secur (TISSEC) 8: 78–118
9. Sabata B (2005) Evidence aggregation in hierarchical evidential reasoning. In: UAI Applications Workshop, Uncertainty in AI 2005. Edinburgh, Scotland

10. Chyssler T, Burschka S, Semling M, Lingvall T, Burbeck K (2004) Alarm reduction and correlation in intrusion detection systems. In: Detection of Intrusions and Malware & Vulnerability Assessment workshop (DIMVA). Dortmund, Deutschland, pp 9–24
11. Ohta S, Kurebayashi R, Kobayashi K (2008) Minimizing false positives of a decision tree classifier for intrusion detection on the internet. J Netw Syst Manage 16: 399–419
12. Haines JW, Lippmann RP, Fried DJ, Tran E, Boswell S, Zissman MA (2000) The 1999 darpa off-line intrusion detection evaluation. Comput Netw. Int J Comput Telecommunications Netw 34: 579–595
13. Project TH (2004) Know Your Enemy : Learning about Security Threats (2nd Edition). Addison-Wesley Professional
14. Sommer R, Paxson V (2010) Outside the closed world: On using machine learning for network intrusion detection. In: Proceedings of the IEEE Symposium on Security and Privacy
15. Bowen T, Chee D, Segal M, Sekar R, Shanbhag T, Uppuluri P (2000) Building survivable systems: An integrated approach based on intrustion detection and damage containment. In: DARPA Information Survivability Conference (DISCEX)
16. Vigna G, Eckmann ST, Kemmerer RA (2000) The stat tool suite. In: Proceedings of DISCEX 2000. Hilton Head, IEEE Computer Society Press
17. Lee W, Stolfo SJ, Chan PK, Eskin E, Fan W, Miller M, Hershkop S, Zhang J (2001) Real time data mining-based intrusion detection. In: Proc. Second DARPA Information Survivability Conference and Exposition. Anaheim, USA, pp 85–100
18. Neumann PG, Porras PA (2005) Experience with EMERALD to date. In: Proceedings 1st USENIX Workshop on Intrusion Detection and Network Monitoring. Santa Clara, CA, USA, pp 73–80
19. Grimaila M, Myers J, Mills R, Peterson G (2011) Design and analysis of a dynamically configured log-based distributed security event detection methodology. J Defense Model Simul: Appl Methodolgy Tech: 1–23
20. Rieke R, Stoynova Z (2010) Predictive security analysis for eventdriven processes. In: MMM-ACNS'10 Proceedings of the 5th international conference on Mathematical methods, models and architectures for computer network security
21. Valdes A, Skinner K (2001) Probabilistic alert correlation. In: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001). Davis, CA, USA, pp 54–68
22. Asif-Iqbal H, Udzir NI, Mahmod R, Ghani AAA (2011) Filtering events using clustering in heterogeneous security logs. Inf Technol J 10: 798–806
23. Corona I, Giacinto G, Mazzariello C, Roli F, Sansone C (2011) Information fusion for computer security: State of the art and open issues. Inf Fusion 10: 274–284
24. Burroughs DJ, Wilson LF, Cybenko GV (2002) Analysis of distributed intrusion detection systems using bayesian methods. In: Proceedings of IEEE International Performance Computing and Communication Conference. Phoenix, AZ, USA, pp 329–334
25. Sabata B, Ornes C (2006) Multisource evidence fusion for cyber-situation assessment. In: Proc. SPIE Vol. 6242, 624201 (Apr. 18, 2006). Orlando, FL, USA
26. Endsley MR (1995) Toward a theory of situation awareness in dynamic systems. Human Factors: J Human Factor Ergon Soc 37: 32–64
27. Debar H, Curry D, Feinstein B (2007) The intrusion detection message exchange format (idmef). Internet experimental RFC 4765. Available at http://tools.ietf.org/html/rfc4765
28. Lan F, Chunlei W, Guoqing M (2010) A framework for network security situation awareness based on knowledge discovery. In: Computer Engineering and Technology (ICCET)
29. Cox K, Gerg C (2004) Managing security with snort and IDS tools. O'Reilly Media, Sebastopol
30. AlFedaghi S, Mahdi F (2010) Events classification in log audit. Int J Netw Secur Appl (IJNSA) 2: 58–73
31. Valdes A, Skinner K, International S (2000) Adaptive, model-based monitoring for cyber attack detection. In: Recent Advances in Intrusion Detection (RAID 2000). Springer-Verlag, pp 80–92
32. Mahoney MV, Chan PK (2002) Learning nonstationary models of normal network traffic for detecting novel attacks. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 376–385
33. Mukkamala S, Sung AH, Abraham A (2003) Intrusion detection using ensemble of soft computing. In: Paradigms, Advances in Soft Computing. Springer Verlag, pp 239–248
34. Faraoun KM, Boukelif A (2006) Securing network traffic using genetically evolved transformations. Malays J Comput Sci 19(1): 9–28. (ISSN 0127-9084)
35. Faraoun KM, Boukelif A (2006) Neural networks learning improvement using the k-means clustering algorithm to detect network intrusions. Int J Comput Intell Appl 6(1): 77–99
36. Tandon G, Chan P (2003) Learning rules from system call arguments and sequences for anomaly detection. In: ICDM Workshop on Data Mining for Computer Security (DMSEC). Melbourne, FL, USA, pp 20–29
37. Mukkamala S, Sung AH (2002) Feature ranking and selection for intrusion detection systems using support vector machines. In: Proceedings of the Second Digital Forensic Research Workshop
38. Chang CC, Lin CJ (2001) LIBSVM: a library for support vector machines. Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
39. wei Hsu C, chung Chang C, jen Lin C (2007) A practical guide to support vector classification," tech. rep., Department of Computer Science, National Taiwan University. Available at http://www.csie.ntu.edu.tw/~cjlin
40. Witten IH, Frank E (2000) Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann
41. Kayacik HG, Zincir-Heywood AN (2003) Using intrusion detection systems with a firewall: Evaluation on darpa 99 dataset. Tech. rep., NIMS Technical Report 062003

doi:10.1186/1869-0238-4-7
Cite this article as: Stroeh *et al.*: **An approach to the correlation of security events based on machine learning techniques.** *Journal of Internet Services and Applications* 2013 **4**:7.