A framework for implementing a Distributed Intrusion Detection System (DIDS) with interoperabilty and information analysis

Pablo Davicino*, Javier Echaiz, and Jorge Ardenghi

Laboratorio de Investigación de Sistemas Distribuidos (LISiDi),
Departamento de Ciencias e Ingeniería de la Computación
Teléfono: +54 291 4595135, Fax: +54 291 4595136
Universidad Nacional del Sur, Bahía Blanca (8000), Argentina
{pmd,je,jra}@cs.uns.edu.ar,
WWW home page: http://lisidi.cs.uns.edu.ar

Abstract. Computer Intrusion Detection Systems (IDS) are primarily designed to protect availability, condentiality and integrity of critical information infrastructures. A Distributed IDS (DIDS) consists of several IDS over a large network(s), all of which communicate with each other, with a central server or with a cluster of servers that facilitates advanced network monitoring. In a distributed environment, DIDS are implemented using cooperative intelligent sensors distributed across the network(s). A significant challenge remains for IDS designers to combine data and information from numerous heterogeneous distributed agents into a coherent process which can be used to evaluate the security of the system. Multisensor data sensing, or distributed sensing, is a discipline used to combine data from multiple and diverse sensors and sources in order to make inferences about events, activities and situations. Today, common environments consists in large networks of high bandwidth. In these scenarios the amount of data produced by the sensors is extremely large so the efficient processing becomes a critical factor.
In this article we propose a framework that aims to achieve the interoperability of the diverse heterogeneous agents that compose the typical infrastructure of a DIDS. Also, we address the alert aggregation and correlation problem proposing an alert processing software pipeline.

Keywords: distributed intrusion detection, alert correlation, alert aggregation, security.

1 Introduction

An Intrusion Detection System (IDS) aims to solve the problem of identifying unauthorized use, misuse or abuse of computer system resources by both system insiders and external penetrators. Today the huge proliferation of heterogeneous

* Partially supported by Comisión de Investigaciones Científicas (CIC) de la Provincia de Buenos Aires.

computer networks provides an additional complexity for the intrusion detection problem. The high availability of connectivity and the lower cost of such communications, gives greater access to outsiders and makes it easier for intruders to avoid detection.

IDS's attempt to identify intruders based of the concept that their behavior is noticeably different from that of a legitimate user. Early in the research, two major principles known as *anomaly detection* [6, 9] and *signature detection* [8, 13] had been explored. The former tries to flag all behaviour that is abnormal for an entity, by building models of normal data and detecting deviations from the normal model in observed data. The latter works by flagging behaviour that is close to some previously defined pattern signature of a known intrusion.

The problem with the first approach relies in the fact that is complex to design a model that can recognize fairly the boundaries between the normal and abnormal behaviour. As a result, the system does not necessarily detect undesirable behaviour, and the false alarm rates can be high. The flaw with the second approach is its inability to detect intrusions that have not yet been made known by the intrusion detection system. Also, the efficiently of the system depends on continuous updates of the signature database.

In a Distributed Intrusion Detection System (DIDS) [14] conventional intrusion detection techniques are embedded inside inteligent agents that are deployed over a large network. In a distributed enviroment IDS agents (sensors) communicate with each other in a cooperative scheme. Distributed monitoring allows early detection of planned and coordinated attacks, thereby allowing Security Administrators (or the system itself) to take preventive and proactive measures. Different sensors can measure distinct features of network traffic or system activity, providing fundamentally different information wich can be used to improve intrusion detection capabilities. Clearly there is a need for tools and techniques for aggregate and combine the outputs of multiple IDSs, filter out incorrect alerts and provide a high level view of the security state of the system being protected.

To address this issue researchers have proposed *alert correlation* [16], an analysis process that takes the alerts generated by IDS agents and produces compact reports on the security status of the components under surveillance. In the real world, each sensor generates a lot of information. In this scenarios actual alerts could be mixed with false alerts and the amount of information becomes rapidly unmanageable. Although a number of correlation approaches have been suggested, there is no consensus on what this process is or how it should be implemented and evaluated. In particular, existing correlation schemes operate on only a few aspects of the correlation process. As a result, it is not clear if and how different parts of the correlation process contribute to the overall goals of correlation.

In this paper we propose a simple framework that aims to achive the interoperability of the heterogenous agents that typically compose a DIDS. We describe the inherent problematics of such distributed environment, such as the high rate of generation of alerts and false positives. In this context we propose a processing pipeline for alert aggregation, correlation and analysis.

The remainder of this paper is structured as follows: Section 2 shows the background for our research, summarizing the related work in the field of DIDS and alert processing. Section 3 presents a prototype to implement a DIDS composed of heterogeneous IDS agents, addressing the problems of alert aggregation and correlation by proposing a technique for their processing. Section 4 describes the general architecture of the system and the testing results. Finally the paper is ended with the conlusion and future work in Section 5.

2 Related Work

In the last decade security in computing, and in particular intrusion detection, have become an important topic of widespread interest in academic research area and industry. Several approaches to intrusion detection and alert correlation have been proposed. Our work is related to work by Cuppens, Ortalo and Lambda [3], Cuppens [2], Debar and Wespi [5], Morin, Mé, Debar and Ducassé [11], Valdes and Skinner [16].

All the schemes performs some sort of analysis on the data. Cuppens [2] uses a central database for alert aggregation and analysis, using a similarity-based technique. Valdes and Skinner also develops a similarity-based technique, using a probabilistic approach to perform correlation of output data from mutiple agents, relying in the concept of *threads* to mantain links between alerts. Debar and Wespi uses features in the Tivoli Enterprise Console to perform correlation, and focus on the abilities of a management system to reduce the amount of data presented to a Security Administrator. The overall system relies in a pre-defined attack scenario. This technique utilizes the fact that intrusions often require several actions to take place in order to succeed. Low-level alerts are compared against pre-defined attack scenario before the alerts can be correlated.

Our work, by comparison focuses on a simple framework which aims to produce practical and efficient alert correlation across heterogeneous intrusion detecion sensors. In this context, interoperability is critical in order to provide a proper base for information sharing among sensors and possibily other system security components . In particular, correlation provides us with the potential ability to see beyond the actual alerts themselves, and determine trends, recognize patterns, and infer relationships between alerts in a distributed enviroment.

3 Prototype for Heterogeneous IDSs Interoperability

As organizations deploy multiple intrusion detection sensors at key points in their networks, the effectiveness of the correlation process becomes critical. The main objective of the correlation process is to produce a reduced and precise overview of the security related activity on the network. Correlation is especially advantageous when heterogeneous sensors are employed because of the potential to aggregate different views of the same incident. To facilitate interoperability of diverse sensors, the Internet Engineering Task Force (IETF) has

developed a standard format, called Intrusion Detection Message Exchange Format (IDMEF), allowing sensors to generate messages in an XML syntax [4].

3.1 Alert Normalization

The first step in alert correlation is the normalization process. The goal of the alert normalization component is to translate all attributes in each sensor alert in a common format. In this way, information from heterogeneous agents can be aggregated and correlated. Due the advantages and widespread utilization of XML format, in our approach we choose the IDMEF as the tool for formalizing the output data generated by each sensor. Figure 1 shows an example of an alert IDMEF message.

In this way we model the alert correlation mechanism as a multicomponent process, that receives as input a stream of alerts from multiple heterogeneos intrusion detection systems. Today, mostly all IDSs systems academic or commercial like Snort [13] or Prelude [17, 1] generates output alert information in IDMEF format. For this reason we assume that there is no need to map the IDS output to IDMEF. In our test we particulary deploy multiple Snort sensors over networks under our domain. Each sensor has the Snort IDMEF plugin which generates output data in IDEMF XML format.

```
<?  l   r  on='1 0' encoding='UT -8'?>
<! OCTYPE      - essage P   LIC
  '-//  T // T  R   XXXX       v1 0//  '
  'idmef-message dtd'>
<      - essage>
  <Alert ident='some_  '>
    <Anal zer model='some_  S_Anal zer'/>
    < reateTime nt stam ='0xc28859cf 0x0'>
        2011-06-05-T13:23:12Z
    </ reateTime>
    <Target>
      < ode>
        <name>some_node</name>
      </ ode>
    </Target>
    <Additional ata meaning='some_data' t  e='string'>
        value
    </Additional ata>
  </Alert>
</     - essage>
```

Fig. 1. Example IDMEF alert.

3.2 Alert Aggregation and eduction

The goal of the alert aggregation process is to group together alerts using similarity metrics. We use as metrics simple factors such as timestamps, IDS type,

IP matches and alert classification. After determining which alerts can be group together, we remove redundant alerts. An alert is considered redundant if it not match with a predefined criteria. A criteria could be quite simple, such as ingoring certain kind of message from a particular sensor, or could have a complex logic that analizes the actual state of the components involved.

Actually, our implementation only support static criteria specified as a configuration parameter of the aggregation component. To specify the criteria we choose to deploy a syntax similar to SQL or LINQ [10]. For example, is common to private LAN Network IDS (NIDS) capture a lot of ARP or ICMP traffic that not represent an attack (false positives). In this context, if we want to ignore the alerts of pings captured by LAN NIDS sensor *snort_lan* the aggregation component could be parametrized as as indicated in the following pseudocode:

$p\ r\ m \leftarrow$ FROM sensornode='snort_lan' WHERE class LIKE = 'Ping*'
insert_criteria($p\ r\ m$)

Additional examples could be alerts not appropriate for the environment, such as Internet Information Services attacks on an Apache web server, or duplicate alerts. In either case, the goal is to remove information that is definitely not relevant.

3.3 Alert Correlation

With alerts reduced to a more relevant and manageable subset, the next step is determine wich alerts may be correlated. Intrusion correlation refers to the interpretation, combination and analysis of information from all available sources about the target system activity for the purpose of intrusion detection and response. The goal of the correlation process is to combine alerts that represent the independent detection of the same attack ocurrence by different intrusion detection agents. In fact, in our model the data can be collected from various sources, like firewalls, web server logs, NIDS, Host IDS (HIDS) and so on. The correlation of alerts produced by heterogeneous resources can provide a number of potential advantages. The only requisite is that the data are expressed in IDMEF format.

The decision to fuse two alerts is based on time, source IP, target IP, ports, name and classification. In principle could be virtually any field deemed relevant in determining an attack. Following the idea presented in [7] we develop a sliding-window mechanism to fuse alerts. The alerts within the time window are stored in a queue, which is maintained in order in a time basis using the timestamp associated to each alert. Like the aggregation criteria, the window size is represented as a parameter passed to the correlation component:

$p\ r\ m \leftarrow$ SET INTERVAL $t\ me\ t\ mp_1$ TO $t\ me\ t\ mp_2$
insert_wsize($p\ r\ m$)

Where each timestamp conforms the time attributes specified by IDMEF. When a new alert arrives, it is compared to the alerts in the queue starting

with the alert with the earliest timestamp. A match is found if the overlaping attributes are equal. Upon finding a match, a new alert is generated as a result of the merging of the original alerts. We call this new alert *hyper-alert*. The merged alerts are assumed to be related to the same attack, so the timestamp of the hyper-alert is assigned the earlier of both start-times and end-times. The attributes fields of this new alert are set to the union of the values of the respective alerts attributes. Later a new alert can be merged with the fused alerts, generating in this way a new hyper-alert.

When the time difference between the oldest and the newest alert in the queue exceeds the specified window size, the older alerts are removed from the queue and passed to the data analysis component. In our experiments, at this point we realized the importance of the window size. A value that is too low cause related alerts to escape the merge process, while a value that is too high caused unrelated alerts to be fused. Following the approach, different time windows and queues could be used to correlate different types of attacks. In particular, we found interesting the use a narrow time window to detect duplicate alerts. In our experiments we choose an interval between 1 and to 2 seconds to detect duplicate alerts. Table 1 shows the result of the fusion of two alerts in a hyper-alert as a result of a port scanning attack. In this case the sliding window is set to 1.5 seconds. The columns named "Start" and "End" represent the timestamps of the alerts. For simplicity we represent them as decimal numbers but they are internally stored in the IDMEF format (eg 2011-06-07T18:23:00-02:00). The pseudocode of the merging process is as follows:

```
function MERGE( ert1, ert2):alert
    hyper_alert ← new alert()
    hyper_alert.id ← get_alert_id()                    ▷ gets a globlal unique id
    hyper_alert.start_time ← min(alert1.start_time,    ▷ minimum time value
                                 alert2.start_time)
    hyper_alert.end_time ← min(alert1.end_time,
                               alert2.end_time)

    for  = 1 to alert.attributeCount do                ▷ generates the attributes
        if (attr.name = start_time)                          ▷ of the hyper-alert
            and (attr.name = end_time) then
                hyper_alert.attr ← alert1.attr ∪ alert2.attr
        end if
    end for
    return hyper_alert
end function
```

3.4 Data Analysis

At this point the system must extrapolate from the obtained set of data either to determine events that may have occurred , or would be likely to occur. Clearly

Table 1. Example of alert fusion in a simple port scanning attack.

AlertID	Classification	Agent	Start	End	Source	Target
15	Simple Portscan	snort_sen1	17.5	19.1	190.11.12.1	192.168.15.33
17	Simple Portscan	snort_sen2	17.4	19.2	190.11.12.1	192.168.15.33
23	**Hyper-Alert**	{15, 17}	17.4	19.1	190.11.12.1	192.168.15.33

this process uses more complex logic than does the correlation component, but the same general principles apply. The principal difference is that an induction process is used to extrapolate or interpolate information from the dataset obtained, rather than simply determine relationships. Examples of this could be predicting the next step of an attack, deducing missing components, or creating attack scenarios as suggested in [12, 18].

In our current prototype the induction component has not been implemented yet. Instead of it, the information gathered as output from the correlation process is presented to the Security Administration in a format similar to Table 1. This information could help the Security Administrator to infer the intention or presence of an attack.

4 Architecture and Testing

As pointed out in Section 3, the schema for reach an interoperability between heterogeneous intrusion detection agents, contains several components structured as a software pipeline as shown in Figure 2. In a similar way the next subsection describe the general architecture of the system and the components used to testing it.

4.1 Architecture

The architecture of the prototype consists in three major components: the IDS Sensors, the Processing Unit and and the Administrative Tool. Figure 3 shows the interaction between the components. We choose a simple architecture with the aim to meet the goals of simplicity and efectiveness in the integration process between applications.

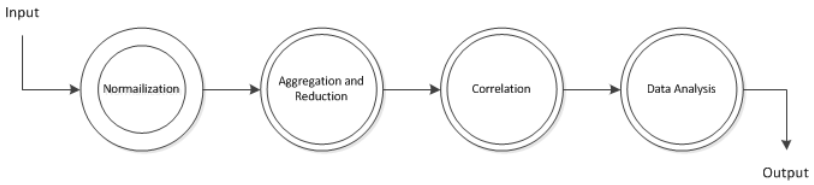

Fig. 2. Alert processing pipeline.

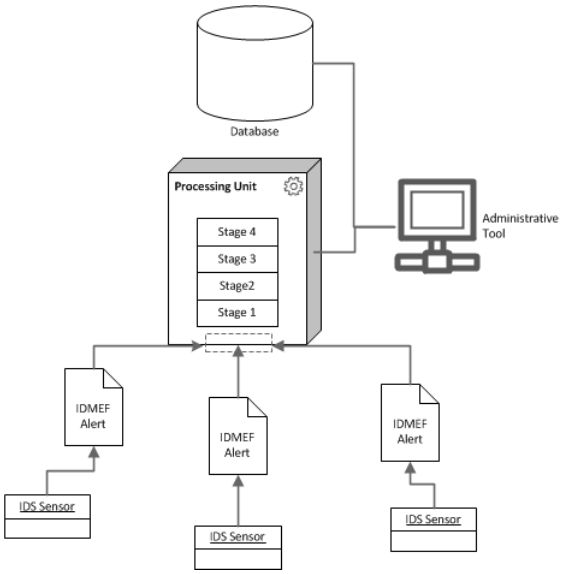

Fig. 3. System Architecture.

Alert ensor. The alert sensor (agent) captures information about the resources that is monitoring. In the case of NIDS, the information is about the network activity in some specific segment. On the other hand, HIDS gather data about local activity in a host. As we pointed out in Section 3.1, we assume that each component generates output data in normalized IDMEF format. In this way, further monitoring components could be easily added to the system.

Processing Unit. The processing unit is the core of our prototype. It contains the software pipeline showed in Figure 2. As the data left the IDS agents, it enters in the processing unit which forward them to the first stage of the pipe. At the same time, the processing unit is responsible of the alert storage. As alerts left the queues described in Section 3.3, they are buffered for storage in a relational database. Some IDS such Snort has special plugins that can interact directly with a database like MySQL. Snort defines a database schema with a set of tables to store the information generated by each sensor. Datamining techniques like [15] can be used with the stored information to perform further data analysis.

Administrative Tool. This component is represented by an application that allows the Security Administrator to gain access to the output data generated by the processing pipeline, and the information stored in the relational database. The application also allows to monitor the real-time state of each sensor and to adjust the parameters presented in Sections 3.2 and 3.3. Finally the application represents the interface between the administra-

tor and the database, allowing to perform tasks like delete records, merge alerts, change alert data and add aditional information.

4.2 Testing

We developed a testing procedure using two Snort sensors deployed over a internal network inside our laboratory. Each sensor runs over an Ubuntu Server 10.04 LTS x64 in a virtualized enviroment provided by a Citrix Xen Server. Each server also runs a MySQL database for storage information about both alert IDMEF and unified output data generated by Snort. In the latter case, we use the Snort Barnyard plugin.

At this point our work is in this initial phase, so our tests focus only in the deployment and debugging of our architecture. Besides the normal traffic produced by our own machines in all day working, there is no additional traffic in the internal network. With that dynamic and simple port scan attacks, we saw and efficient processing and correlation of alerts detecting effectively the threat. Also by parametrizing the system, we observate the reduction of the false positives, mainly produced by ICMP and ARP traffic.

In a second phase, is our intention to test the system with known data sets of alerts, as Defcon. The final test will be the deployment of the sensors over a public network.

5 Conclusions and Future Work

In this paper we describe a framework for the interoperability of heterogeneous IDS agents in the implementation of a Distributed Intrusion Detection System (DIDS). We highlight the principal issues when dealing with multiple IDS agents depoyed over a distributed enviroment. We address the aggregation, correlation and reduction problems over a software processing pipeline, and we present the typical architecture of the system.

We believe that the simplicity of our approach is the key for the success, due to it facilitates the integration of external components and applications. Also, we believe that our work is in its initial phase with a lot of future work to do. In this context, we contemplate the implementation of an overlay network to distribute the information generated by the multiple IDS agents in the system. Turn we want to develop a several interfaces to the Normalization module. In this way, we relax the limitation that every agent in the systems generates output in IDEMF format. In this way each vendor could produce their own raw data and interface with first stage of the pipeline to make the mapping.

Finally, we believe that will be a key point of our implementation develop the dynamic nature of the sliding window parameter. We contemplate future research of datamining techniques to analize further the information stored in multiple distributed databases inside the system.

References

1. Blanc, M., Oudot, L., Glaume, V.: Global intrusion detection: Prelude hybrid ids. Tech. rep. (2003)
2. Cuppens, F.: Managing alerts in a multi-intrusion detection environment. Computer Security Applications Conference, Annual 0, 0022 (2001)
3. Cuppens, F., Ortalo, R.: Lambda: A language to model a database for detection of attacks. In: Debar, H., M, L., Wu, S. (eds.) Recent Advances in Intrusion Detection, Lecture Notes in Computer Science, vol. 1907, pp. 197–216. Springer Berlin / Heidelberg (2000)
4. Curry, D., Debar, H.: Intrusion detection message exchange format data model and extensible markup language (xml) document type definition. Intrusion Detection Working Group (January 2003)
5. Debar, H., Wespi, A.: Aggregation and correlation of intrusion-detection alerts. In: In Recent Advances in Intrusion Detection, LNCS 2212. pp. 85–103. Springer-Verlag (2001)
6. Denning, D.E.: An intrusion-detection model. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 13(2), 222–232 (1987)
7. Eskin, E.: Modeling system calls for intrusion detection with dynamic window sizes. In: In Proceedings of DARPA Information Survivabilty Conference and Exposition II (DISCEX (2001)
8. ah Kim, H.: Autograph: Toward automated, distributed worm signature detection. In: In Proceedings of the 13th Usenix Security Symposium. pp. 271–286 (2004)
9. Lazarevic, A., Ozgur, A., Ertoz, L., Srivastava, J., Kumar, V.: A comparative study of anomaly detection schemes in network intrusion detection. In: In Proceedings of the Third SIAM International Conference on Data Mining (2003)
10. Meijer, E., Beckman, B., Bierman, G.: Linq: reconciling object, relations and xml in the .net framework. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data. pp. 706–706. SIGMOD '06, ACM, New York, NY, USA (2006)
11. Morin, B., Mé, L., Debar, H., Ducassé, M.: A logic-based model to support alert correlation in intrusion detection. Inf. Fusion 10, 285–299 (October 2009)
12. Ning, P., Cui, Y., Reeves, D.S.: Constructing attack scenarios through correlation of intrusion alerts. In: In Proceedings of the 9th ACM conference on Computer and communications security. pp. 245–254 (2002)
13. Roesch, M.: Snort - lightweight intrusion detection for networks. In: Proceedings of the 13th USENIX conference on System administration. pp. 229–238. LISA '99, USENIX Association, Berkeley, CA, USA (1999)
14. Shetty, P.: DISTRIBUTED INTRUSION: Detection Systems. VDM Verlag, Saarbrucken, Germany, Germany (2010)
15. Vaarandi, R.: Real-time classification of ids alerts with data mining techniques. In: Proceedings of the 28th IEEE conference on Military communications. pp. 1786–1792. MILCOM'09, IEEE Press, Piscataway, NJ, USA (2009)
16. Valdes, A., Skinner, K.: Probabilistic alert correlation. In: Lee, W., M, L., Wespi, A. (eds.) Recent Advances in Intrusion Detection, Lecture Notes in Computer Science, vol. 2212, pp. 54–68. Springer Berlin / Heidelberg (2001)
17. Zaraska, K.: Prelude ids: current state and development perspectives. Tech. rep. (2003)
18. Zhu, B., Ghorbani, A.A.: Alert correlation for extracting attack strategies. International Journal of Network Security 3, 244–258 (2006)