Design and Evaluation of a Grid Computing Based Architecture for Integrating Heterogeneous IDSs

Paulo F. Silva, Carlos B. Westphall, Carla M. Westphall
Network and Management Laboratory
Post-Graduate Program in Computer Science
Federal University of Santa Catarina, Florianópolis, Brazil
[paulo, westphal, carla]@lrg.ufsc.br

Marcos Dias de Assunção
Grid Computing and Distributed Systems Laboratory
Department of Computer Science and Software Engineering
University of Melbourne, Victoria, Australia
assuncao@acm.org

Abstract - Intrusion Detection Systems (IDSs) have substantially improved in recent past. However, the sophistication of network attacks has also improved; many attacks are coordinated and originated in multiple networks. The detection of such attacks requires IDSs to obtain information on network events from multiple networks or administrative domains. This work demonstrates that a Distributed IDS (DIDS) can be composed of existing IDSs, improving the detection of misuse in a multiple network environment. We use a Grid middleware for creating a service-based intrusion detection Grid. We demonstrate through experimental results that the proposed DIDS allows the integration of heterogeneous existing IDSs and improves the detection of attacks by exploring the synergy between these IDSs.

Keywords - Distributed Intrusion Detection Systems, Grids Services, Globus, Composition;

I. INTRODUCTION

The research community and industry have proposed varying solutions for integrating heterogeneous IDSs (Intrusion Detection Systems) [4][5][6][10]. The IDWG (Intrusion Detection Working Group) describes several reasons for integrating IDSs [9].

Due to the need for correlating information from multiple network domains to detect distributed attacks, DIDSs (Distributed Intrusion Detection Systems) started to emerge in early 90s [7]. Research on DIDS has received much interest since then, mainly because centralised and monolithic IDSs are not able to provide enough information to prevent such kinds of attacks. However, a DIDS requires a high degree of coordination among its components and can be difficult to maintain. On the other hand, the use of multiple tools or the integration of existing IDSs are not straightforward or easy tasks, demanding the design and implementation of protocols for communication, data transfer, among others. Grid computing technology is appealing in this case as it enables the development of distributed applications and coordination in a distributed environment.

Grid computing aims to enable coordinate resource sharing in dynamic groups of individuals and/or organizations. Moreover, Grid middleware provides means for secure access, management and allocation of remote resources; resource information services; and protocols and mechanisms for data transfer [1]. Grid systems have increasingly been structured as networks of interoperating services that communicate with one another via standard interfaces. WSRF (Web Services Resource Framework) [2] provides a set of Web Services interfaces to realise this service orientation. In this scenario, resources, existing or new applications can be encapsulated and provided as services to end users. In a similar fashion, IDSs can be encapsulated and delivered as services.

In our previous work [12], we proposed an architecture based on Grid computing technology for the composition of a DIDS, through the encapsulation of existing IDSs as Grid services. We demonstrated the usefulness of such integration through simulation using a Grid simulator. In this work, we describe the implementation of our architecture using Globus Toolkit 4.0.1 [3] and complying with the WSRF specifications [2]. The proposed system, termed DIDSsG (Distributed Intrusion Detection System on Grid) enables heterogeneous IDSs to work together in a cooperative way. We design a common interface for IDSs integrated to our DIDSsG. Each IDS is viewed as a resource accessed through WSRF interfaces. DIDSsG uses WSRF compliant services offered by Globus, including: communication (XML, SOAP), Grid Security Infrastructure (GSI) and Monitoring and Discovery Service (WS-MDS).

The rest of this paper is organised as follows. Section 2 discusses on related work. The proposed system is presented in Section 3. Section 4 describes the development. Experimental results and their discussion are presented in Section 5. We conclude the paper and present future work in Section 6.

II. RELATED WORK

Sterne et al. present a hierarchical architecture for a DIDS; information is collected, aggregated, correlated and analysed as it is sent up in the hierarchy [8]. Components in the same level of the hierarchy cooperate with one another. The integration proposed by DIDSsG also follows a hierarchical architecture. An IDS integrated to the DIDSsG offers functionalities at a given level of the hierarchy and requests functionalities from IDSs from other levels. The hierarchy presented by Sterne et al. [8] integrates homogeneous IDSs whereas the hierarchical architecture of DIDSsG integrates heterogeneous IDSs.

Leu et al. [5] propose the use of Globus Toolkit for intrusion detection, especially for DoS (Denial of Service) and DDoS (Distributed Denial of Service) attacks. Leu et al. [4] propose Grid-based IDS named Fault-tolerant Grid Intrusion Detection System (FGIDS) which explores a Grid’s dynamic and abundant computing resources to detect malicious behaviours from a massive amount of network packets.
Leu et al. [4], [5] point out that IDSs developed upon Grid platforms are less vulnerable to attacks because of the distribution provided for such platforms. Leu et al. [5] have demonstrated through experimental results the advantages on performance and dependability of applying computational Grids to IDSs. Our work proposes the development of a DIDS upon a Grid platform. However, the resulting DIDS integrates heterogeneous IDSs whereas the DIDSs presented by Leu et al. [4],[5] do not consider the integration of heterogeneous IDSs.

GHIDS (Grid-specific Host-based IDS), a specific IDS for Grids, is presented by Feng et al. [6]. GHIDS verifies the kernel of the operating system and generates reports relating information of the host with information of the Grid. The experiments related to the GHIDS were based on Globus 4.0. DIDSoG does not aim at detecting intrusions in a Grid environment. In contrast, DIDSoG uses the Grid to compose a DIDS by integrating specific IDSs; the resulting DIDS could however be used to identify attacks in a Grid environment, through the integration of Grid IDSs.

MAIDG (Multi-Agent Approach to Intrusion Detection for Grid) is presented by Zhu et al. [10]. MAIDG uses resources from the Globus to dynamically integrate intrusion detection resources to the Grid. The detection resources publish, locate and transfer data using resources of the Globus. MAIDG emphasizes the benefits of the Globus in the integration of intrusion detection resources. The benefits are on improvement and facility on communication, security and localization.

Similarly to MAIDG, DIDSoG dynamically integrates intrusion detection resources using a Grid computing middleware. The difference between the two systems relies on the relationship between the resources and the target of the intrusion detection. DIDSoG uses a service-oriented approach (WSRF and Web Services), while MAIDG uses a multi-agent system approach. While the target of intrusion detection of MAIDG is specifically the Grid, DIDSoG has any kind of environment as a target, including Grids. The target of intrusion detection of DIDSoG is defined by the characteristics of the intrusion detection resources integrated to it.

III. THE DIDSOG ARCHITECTURE

DIDSoG presents a hierarchy of intrusion detection services; this hierarchy is organized in a two-dimensional vector defined by “Scope:Complexity”. The IDSs composing DIDSoG can be organized in different levels of scope and complexity, depending on its functionalities, the topology of the target environment and expected results.

Figure 1 presents a data gathering scenario (host, network or application) of the DIDSoG. The data that feeds DIDSoG is initially collected by a Native Sensor, which stores it in a database, according to the database specification. A Sensor Gateway, specifically developed to communicate with the database of the Native Sensor, carries out the first access to the DIDSoG. The Sensor Gateway sends the collected data to a Sensor resource. The Sensor resource sends the data to other resources of DIDSoG, according to its configuration. The Sensor resource is the first DIDSoG resource in which data passes through, thus it is the entry point of the data in DIDSoG.

Figure 2 shows that an Analyser that acts on the information from a single host (level 1:1) receives information from sensor resources. An Aggregation resource in level 2:1 can receive information from sensor resources or other resources that have already processed data, such as Analysers and other Aggregators. An Aggregation resource in level 3:1 receives and aggregates data from several other resources that have already processed the data.

An Analyser in the first scope and complexity level sends the information to more complex Analysers in the next levels of complexity (level 1:N). When an Analyser detects an intrusion, it communicates with Countermeasure and Monitoring services registered to its scope. An Analyser can invoke a Countermeasure service that replies to a detected attack, or informs a Monitoring service about the ongoing attack, thus the administrator can act accordingly.

Aggregation and correlation resources in the second scope receive information from Sensors from different sources. These resources process the received information and send it to the analysis resources registered to the first level of complexity in the second scope (level 2:1). The information is also sent to the aggregation and correlation resources registered to the first level of complexity in the next scope (level 3:1).

An Analysis resource in the second scope acts like the an Analysis resource in the first scope, directing the information to a more complex Analysis resource and putting the Countermeasure and Monitoring resources in action in case of detected attacks. Aggregation and correlation resources in the third scope receive information from different sources. These resources then carry out the aggregation and correlation of the information from different domains and send the resulting information to the analysis resources in the first level of...
complexity in the third scope (level 3:1). The information could also be sent to the Aggregation service in the next scope in case of any resources registered to such level. The Analysis resources in the third scope act similarly to the Analysis resources in the first and second scopes, except that the Analysis resources in the third scope act on information from multiple domains.

The DTD file that validates the received XML. Binary. Class XML contains the DTDFile attribute to specify represents the data format that the resource accepts to receive.

A DIDSoG resource is composed of four components: Native IDS, Descriptor, Base and Connector. A Native IDS corresponds to the IDS being integrated to DIDSoG. This component processes the received data and generates new data to be sent to other DIDSoG resources. The Native IDS component can be any tool that processes information related to intrusion detection, for example, analysis, data gathering, data aggregation, data correlation, intrusion response or management tools.

The Descriptor is responsible for the information that identifies a resource and its respective destination resources in DIDSoG. This component processes the received data and generates new data that will be sent to other DIDSoG resources. The Native IDS component can be any tool that processes information related to intrusion detection, for example, analysis, data gathering, data aggregation, data correlation, intrusion response or management tools.

The class DIDSoGService, DIDSoGResource and DIDSoGResourceHome implement the necessary requirements for the execution of a WSRF compliant Grid Service. The DIDSoGService class implements the Base component. DIDSoGService class relates to DIDSoGDescriptor, which implements the Descriptor component. The Descriptor is read by the DIDSoGDescriptor class from a XML document that specifies the Descriptor component. DIDSoGService also relates with the abstract class Generic_Connector.

The Connector component must be developed for each DIDSoG resource. The Connector component is implemented by creating a class that implements Generic_Connector. The implemented class will have methods to receive and send data to the Base component (DIDSoGService). The Connector component must be implemented in a way that interacts with the Native IDS that will be integrated to the DIDSoG.

The TargetResources class describes the features of the destination resources for a given resource. This class aggregates the Resource class. The Resource class identifies the characteristics of a Destination Resource. This identification is made through the featureType attribute and the Level and DataType classes. A given resource analyses the information from Descriptors from other resources, and compares this information with the information specified in TargetResources to know to which resources to send the results of its processing tasks.

The Base component is responsible for the communication of a resource with other resources of DIDSoG and with the Grid Information Service. This component registers the resource and queries other resources in the Grid Information Service.

The Connector component is the link between Base and Native IDS. The information that Base receives from source resources is passed to the Connector component. The Connector component performs the necessary changes in the data so a given resource is able to send this data to Native IDS for processing. Connector has also the responsibility for collecting the information processed by Native IDS and for making the necessary changes, so that the information can pass through the DIDSoG again. After these changes, Connector sends the information to Base, which in turn sends it to the destination resources in accordance with the specifications of the Descriptor component.

IV. IMPLEMENTATION

The proposed system was developed using Globus Toolkit 4.0.1 [3]. A DIDSoG resource is made available to DIDSoG as a Grid Service implemented on a WSRF platform [2]. DIDSoG system enables the implementation of DIDSoG resources. The main classes of the system are presented in Figure 4.

The classes DIDSoGService, DIDSoGResource and DIDSoGResourceHome implement the necessary requirements for the execution of a WSRF compliant Grid Service. The DIDSoGService class implements the Base component. DIDSoGService class relates to DIDSoGDescriptor, which implements the Descriptor component. The Descriptor is read by the DIDSoGDescriptor class from a XML document that specifies the Descriptor component. DIDSoGService also relates with the abstract class Generic_Connector.

Figure 4. Class diagram of the Descriptor component.
V. EXPERIMENTAL RESULTS

In order to evaluate DIDSoG, we created resources for data gathering, analysis, aggregation/correlation and responses. For each resource, we simulated the processing tasks of a Native IDS. For each Native IDS it was developed a class that implements Generic_Connector. This class corresponds to the Connector component of the Native IDS and aims at integrating it to the DIDSoG.

After the development of the DIDSoG resources, we deployed them in the Web Services container provided by Globus Toolkit. The Descriptors were defined in XML format for each DIDSoG resource. The DIDSoG resources were distributed in different hosts, and the container was executed in one of these hosts.

With the DIDSoG resources available as Grid Services, a publication of each Grid Service was performed in WS-MDS (Web Service Monitoring and Discovery Service). From that moment onwards, all the DIDSoG resources are available for communication and execution.

![DIDSoG execution flow.](image)

For the data gathering purposes, we used a KDD database [11], obtained from a simulation of a large amount of intrusions to a military network environment. Two Gateway sensors have been developed, capable of reading information in the KDD format. We attributed one KDD database to each Gateway sensor.

Two experiments were performed to demonstrate the behaviour of the DIDSoG. The first experiment analyses the behaviour of the DIDSoG with different amounts of data. The second experiment analyses the behaviour of the DIDSoG with different scenarios.

Figure 6 presents the scenario for the first experiment, according to specification of the descriptors of each resource. In this scenario, two Gateway Sensors collect KDD data from a database and send it to other.

![Figure 6. DIDSoG execution flow.](image)

The experiment was performed with 1000, 2000, 3000 and 4000 records for each Gateway Sensor. Each Gateway Sensor was implemented to pick up a random record from database and send it to the next DIDSoG component.

According to logs on Counter-Measure_1 (CM1) and Counter-Measure_2 (CM2), Figure 7(A) presents both the detection and response average times for each counter-measure component. We can observe that detection and response average times are little impacted by the different amounts of collected data.

The response time on CM1 and CM2 presents a small reduction when the amount of data increases. The detection time on CM1 and CM2 also presents a small reduction with 3000 and 4000 records. We can observe that the detection time on CM2 is bigger than all other times. It occurs because this time is influenced by the aggregation component. This component performs the aggregation from received records and this task increases the detection time on CM2.

The average processing time for a record, with different amount of records, is presented in Figure 7(B). We can observe a decrease in the time to process a record when the amount of records is increased. With 8000 records, the DIDSoG registers the smallest average time to process a record.

![Figure 7. Graphs by amount of records.](image)

For the second experiment we consider three scenarios. Figure 8 presents these scenarios, according to specification of the descriptors of each resource. These experiments aim at evaluating the impact of the hierarchical structure in several parts of DIDSoG on different scenarios.

According to Figure 8, Case1 is a scenario with a low hierarchy where the data is sent directly to target component, without intermediate components. On Case2 we consider a hierarchy structure between the Aggregator and the Analysers. In this scenario, the Aggregator sends data only to Analyser_1. Analyser_1 forwards the data to Analyser_2 and Analyser_2 in turn forwards data to Analyser_3.

In Case3 there is also a hierarchical structure between the Counter-Measure components. In this scenario, all Analysers send data only to Counter-Measure_1, which forwards data to Counter-Measure_2.

The components Gateway Sensor 1 and 2 were fed with 1000 KDD records each, and configured to select records randomly from the databases. The experiment registers detection and response average times on components Counter-Measure1 (CM1) and Counter-Measure2 (CM2) on each scenario. Figure 9(A) presents the detection and response...
average times registered in each scenario. Each alert on all
diagrams was sent to all Counter-Measure components. Then,
the detect time is the same on Counter-Measure 1 and 2 and the
diagram shows only a line about detection time.

According to Figure 9(A), Case2 presents the best detection
time while Case1 has the worst. Case1 has the worst detection
time because all analysers are connected directly with the
aggregator component, impacting into its performance and
influencing on the detection time of all analysers.

Case1 registers the best response time, while Case3
registers the worst. All response times were greater in CM2
than in CM1. Moreover, this difference increases in Case3,
mainly because in this case, CM1 needs to forward alerts to
CM2, and only later CM2 generates the responses.

Case2 presents an average time between the best and the
worst time, but it has also the best detection time as shown in
Figure 9(A). Case2 presents the best final time between collect
time and response time.

According to Figure 9(B), Case2 also presents the best average
time needed to process a record. Case3 registers the
greater time needed to process each record. It is also true for
the response time as shown in Figure 10. A hierarchical
structure for Counter-Measure components therefore has not
been very appealing. However, according with results
presented in Figure 9, a hierarchical structure between
aggregation and analysers has advantages.

The experiments demonstrate the behaviour of the
hierarchical structure in several parts of DIDS. The
resources carry out tasks (data collection, aggregation, analysis
and generation of alerts) in an integrated manner.

VI. CONCLUSION

The integration of heterogeneous IDSs is important.
However, the incompatibility and diversity of IDS solutions
make such integration extremely difficult. This work proposes
a middleware for the composition of DIDS by integrating
existing IDSs on a computational Grid platform (DIDS). DIDSs in DIDS are encapsulated as Grid services [2] for
intrusion detection. A computational Grid platform [3] is used
for the integration by providing the basic requirements for
communication, localization, and security mechanisms.

Requirements of security, communication and localization
were provided by Globus Toolkit. The authenticity of the
DIDS resources, the confidentiality and the integrity of the data
are made through digital certificate by GSI. The
communication between the DIDS resources is made in
XML through the protocol SOAP. A DIDS resource use
WS-MDS to publish and locate the other DIDS resources.

Based on the components of the architecture, several
resources were modelled forming a Grid of intrusion detection.
The test demonstrated the usefulness of the proposed system.
Data from the sensor resources has been read and used to feed
other resources of DIDS. Resources providing different
intrusion detection services have been integrated.

Various resources have been modelled following the
architecture components. The components of DIDS have
served as base for the integration of the resources presented in
the tests. During the tests, the different IDSs cooperated with
one another in a distributed manner; however, in a coordinated
way with an integrated view of the events, having, thus, the
capability to detect distributed attacks. This capability
demonstrates that the IDSs integrated have resulted in a DIDS.

DIDS presents new research opportunities that we would
like to pursue, including: use services of the Grid to manage
data of the DIDS (Grid-FTP); enable distribution of task
processing that require a great deal of computing (e.g. analysis
and correlation); allow specification of intrusion detection
policies for different environments, development of incentives
for integration of IDSs (economic problems).

REFERENCES
[1] Foster, I.; et al. From Open Grid Services Infrastructure to WS-
[4] Leu, F.; Li, M.; Lin, J. Intrusion Detection based on Grid,
International Multi-Conference on Computing in the Global
Architecture for MANETs. Proc. 3rd IEEE IWIA05,March 05.
[10] Zhu, P. et al. A New Flexible Multi-Agent Approach to Intru-
Available at kdd.ics.uci.edu/databases/kddcup99/
Composition of a DIDS by Integrating Heterogeneous IDSs on
Grids. Proc. of the 4th International Workshop on MGC, with
Middleware 2006 ACM/IPIP/USENIX 7th International

Figure 9. Graphs by hierarchical scenarios.