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
E-mail: {paulo, westphal, carla}@lrg.ufsc.br

Marcos Dias de Assunção
Grid Computing and Distributed Systems Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Victoria, Australia
E-mail: marcosd@csse.unimelb.edu.au

***Abstract*—Intrusion Detection Systems (IDSs) have been substantially improved in recent past. However, network attacks have become more sophisticated and increasingly complex: many of current attacks are coordinated and originated in multiple networks. To detect these attacks, IDSs need 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 misuses 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 existing IDSs.**

***Index Terms*—Distributed Intrusion Detection Systems, Grid Services, Globus, IDS Composition.**

I. Introduction

Distributed Intrusion Detection Systems (DIDSs) started to emerge in early 90s due to the need to correlate information from multiple network domains in order to detect distributed attacks [1]. Since then, research on DIDSs has received much interest, mainly because centralised and monolithic Intrusion Detection Systems (IDSs) are not able to provide enough information to prevent attacks that are coordinated and originated in multiple networks. The research community and industry have proposed varying solutions for integrating heterogeneous IDSs [2]–[5]. The Intrusion Detection Working Group IDWG highlights several reasons and benefits of integrating IDSs [6].

Although a DIDS can allow the detection of distributed attacks, it requires a high degree of coordination among its components and can be complex and difficult to maintain. Moreover, the use of multiple tools for intrusion detection or the integration of existing IDSs are neither straightforward nor easy tasks; they demand the design and implementation of protocols for communication, data transfer, among others. In this case, Grid computing is appealing as it enables the development of distributed applications and coordination in a distributed environment [7].

Grid computing aims to enable coordinated resource sharing within dynamic groups of individuals and/or physical organisations. Grid middleware enables for secure access, management and allocation of remote resources; and provides resource information services and protocols and mechanisms for data transfer [8]. Grid systems, driven by Service Oriented Architectures (SOAs), have been structured as networks of interoperating services that communicate with one another via standard interfaces [9]. In this scenario, resources and existing applications can be encapsulated and provided as services to end users. We envision that IDSs can also be encapsulated and delivered as services to users.

In our previous work, we proposed an architecture based on Grid computing technology for the composition of a DIDS, through the encapsulation of existing IDSs as Grid services [10]. We have demonstrated the usefulness of this integration through simulation using GridSim Toolkit [11]. Here, we describe the implementation of our architecture using Globus Toolkit 4.0.1 [12], complying with the WSRF specifications [9]. The proposed system, termed Distributed Intrusion Detection System on Grid (DIDSoG) enables heterogeneous IDSs to work together in a cooperative way. We design a common interface to integrate IDSs to DIDSoG. Each IDS is viewed as a resource accessed through WSRF interfaces. DIDSoG uses WSRF compliant services offered by Globus including: communication (i.e. XML and SOAP), Grid Security Infrastructure (GSI) and Monitoring and Discovery Service (WS-MDS).

The rest of this paper is organised as follows. Section II presents related work. Section III describes the proposed system. The development of DIDSoG is discussed in Section IV. We present and discuss experimental results in Section V. Section VI concludes the paper and presents the future work.

II. Related Work

Sterne *et al*. present a hierarchical architecture for a DIDS in which information is collected, aggregated, correlated and analysed as it is sent up in the hierarchy [13]. Components in the same level of the hierarchy cooperate with one another. Similarly, the integration proposed by DIDSoG follows a hierarchical architecture. An IDS integrated to DIDSoG offers functionalities at a given level of the hierarchy and requests functions from IDSs at other levels. However, DIDSoG differs from the work by Sterne *et al*. [13] by enabling the integration of heterogeneous IDSs.

Leu *et al*. [2] propose the use of Globus Toolkit for intrusion detection focusing on Denial of Service (DoS) and Distributed

Denial of Service (DDoS) attacks. Leu *et al*. [3] introduce 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 number of network packets.

Leu *et al*. [2], [3] point out that IDSs developed upon Grid platforms are less vulnerable to attacks because of the distribution provided for by such platforms. They have demonstrated through experimental results the advantages on performance and dependability of applying computational Grids to IDSs. Our work also 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.

Grid-specific Host-based IDS (GHIDS), a specific IDS for Grids, is presented by Feng *et al*. [4]. GHIDS verifies the kernel of the operating system and generates reports relating information of the host with information of the Grid. The experiments on GHIDS were based on Globus 4.0. DIDSoG, on the other hand, 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 eventually be used to identify attacks in a Grid environment through the integration of Grid IDSs.

Multi-Agent Approach to Intrusion Detection for Grid (MAIDG) is presented by Zhu *et al*. [5]. MAIDG uses Globus to dynamically integrate intrusion detection resources to the Grid. The detection resources publish, locate and transfer data using Globus tools. MAIDG emphasises the benefits of Globus in the integration of intrusion detection resources. The benefits are on improvement and facility on communication, security and resource discovery.

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 (i.e. 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 integrated intrusion detection resources.

III. The DIDSoG Architecture

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

Figure 1 presents how DIDSoG collects data from hosts, networks or applications. Initially, a Native Sensor collects the data and stores it in a database. The Sensor Gateway, specifically developed to communicate with the database of the Native Sensor, carries out the first access to the DIDSoG.

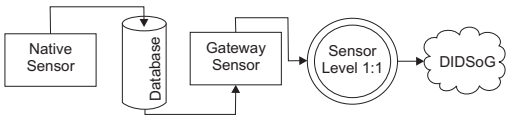

Fig. 1. DIDSoG data gathering.

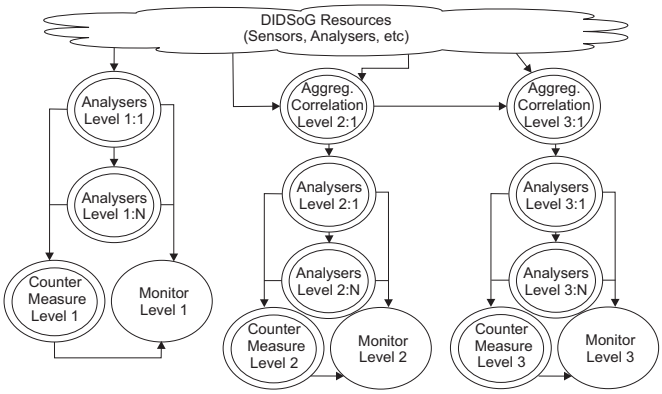

Fig. 2. DIDSoG data flow.

Sensor Gateway sends the collected data to a Sensor resource. The Sensor resource sends the data to other DIDSoG resources according to its configuration. The Sensor resource is the first DIDSoG resource in which data passes through, thus it is the entry point for the data in DIDSoG.

Once collected, the data flows according to the way resources were configured in DIDSoG. Figure 2 presents a DIDSoG composed of different intrusion detection services provided by different IDSs; these services can include: data aggregation, data correlation, analysis, intrusion response and management services. The figure also illustrates the information flow and the relationship between the levels of scope and complexity.

Figure 2 shows that an Analyser that acts on data 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 from other resources that have already processed data, such as Analysers and other Aggregators. The Aggregation resource in level 3:1 receives and aggregates data from several other resources that have already processed the data.

The 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 Counter-Measure and Monitoring services registered to its scope. An Analyser can invoke a Counter-Measure service to handle a detected attack, or inform a Monitoring service about the ongoing attack, so 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 Analysers 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

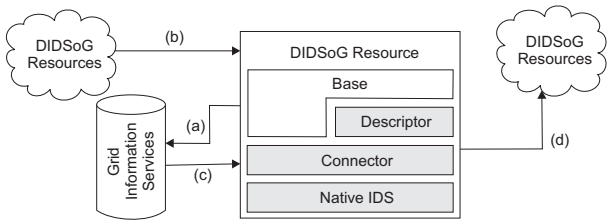

Fig. 3. The architecture of a DIDSoG resource.

first level of complexity in the next scope (level 3:1).

The Analyser in the second scope acts similarly to the Analyser in the first scope by directing the information to a more complex Analyser and putting the Counter-Measure and Monitoring resources in action in case attacks are detected. 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 Analysers in the first level of complexity in the third scope (level 3:1). The information could also be sent to the Aggregation resource in the next scope in case there resources registered to that level. The Analysers in the third scope act similarly to the Analysers in the first and second scopes, except that in the third scope they act on information from multiple domains. The functionalities of the registered resources in each of the scope and complexity levels can vary from one environment to another, but DIDSoG allows the development of "N" levels of scope and complexity.

Figure 3 presents the architecture of a resource participating in the DIDSoG. 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 (e.g. 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. The Base component is responsible for the communication of a resource with other resources of DIDSoG and with the Grid Information Service (GIS). This component registers the resource and queries other resources in the GIS. 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 performs the necessary changes in the data, so it can be interpreted by Native IDS, and sends 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.

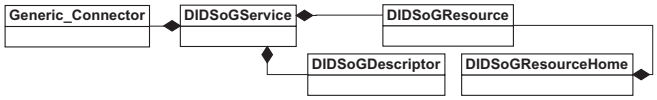

Fig. 4. Main classes of the DIDSoG services.

When a resource is initialised, it (a) registers itself to a GIS thus other participating resources can query the services provided. After the registration, the resource is able to receive data from other resources. A given resource of DIDSoG interacts with other resources by (b) receiving and processing data; (c) querying the GIS about other resources available; and (d) sending the results to other resources, therefore forming a Grid of intrusion detection resources.

IV. Implementation

The proposed system has been implemented using Globus Toolkit 4.0.1 [12]. A DIDSoG resource is made available to DIDSoG as a Grid Service implemented on a WSRF platform [9]. 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. The *DIDSoGService* class is related to *DIDSoGDescriptor*, which implements the Descriptor component. The Descriptor is read by the *DIDSoGDescriptor* class from a XML document that contains the Descriptor specification. *DIDSoGService* also relates with the abstract class *Generic_Connector*.

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

V. Experimental Results

In order to evaluate DIDSoG, we create resources for data gathering, analysis, aggregation/correlation and response. For each resource, we simulate the processing tasks of a Native IDS. For each Native IDS we develop 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 deploy them in the Web Services container provided by Globus Toolkit. The Descriptors are defined in XML format for each DIDSoG resource. The DIDSoG resources are distributed in different hosts and the container is executed in each host.

With the DIDSoG resources available as Grid Services, a publication of each Grid Service in the Web Service Monitoring and Discovery Service (WS-MDS) is carried out. From

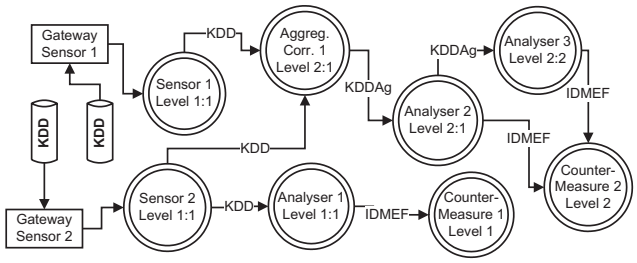

Fig. 5. Execution flow for the experimental scenario.

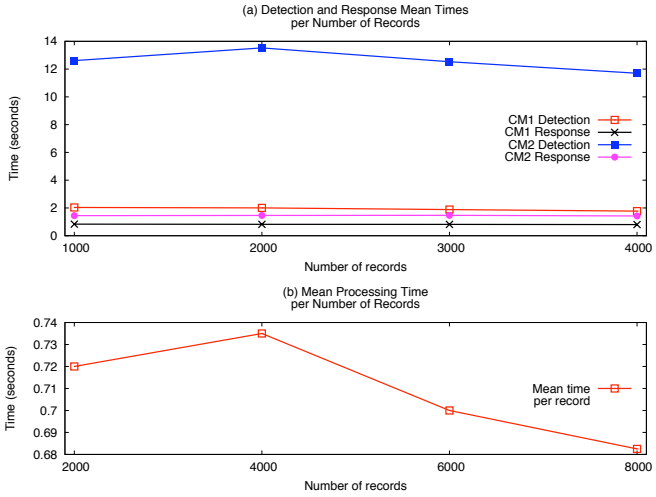

Fig. 6. Mean times per number of records.

that moment onwards, all the DIDSoG resources are available for communication and execution.

For the data gathering purposes, we use a KDD database [14] obtained from a simulation of a large number of intrusions to a military network environment. Two Gateway sensors are developed, capable of reading information in the KDD format. We assign one KDD database to each Gateway sensor.

We perform two experiments to demonstrate the behaviour of DIDSoG. The first experiment analyses the behaviour of DIDSoG with different amounts of data. The second experiment analyses DIDSoG in different hierarchical scenarios. Figure 5 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. We perform the experiment with 1000, 2000, 3000 and 4000 records for each Gateway Sensor. Each Gateway Sensor randomly selects records from the database and and sends them to the next DIDSoG component.

Figure 6(a) presents the results of both the detection and response mean times for each counter-measure component (i.e. Conter-Measure 1 (CM1) and Counter-Measure 2 (CM2)). We observe that detection and response mean times are little impacted by the different amounts of data collected. 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

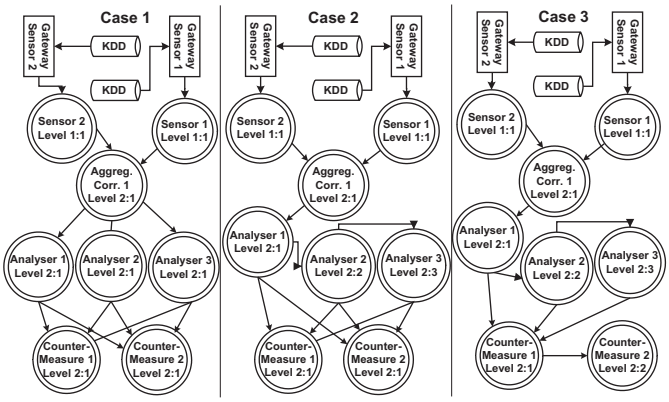

Fig. 7. The three considered hierarchical scenarios.

records. We notice that the detection time on CM2 is larger than all other times. It occurs because this time is influenced by the aggregation component. This component performs the aggregation from received records, which increases the detection time on CM2. The average processing time of a record, with different numbers of records, is presented in Figure 6(b). We observe a decrease in the time to process a record when the number of records increases. With 8000 records, the DIDSoG registers the shortest mean time.

For the second experiment we consider three scenarios. Figure 7 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 in different scenarios. Case 1 is a scenario with a low hierarchy where the data is sent directly to the target component, without intermediate components. In Case 2, we consider a hierarchical 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 Case 3 there is also a hierarchical structure between the Counter-Measure components. In this case, all Analysers send data only to Counter-Measure 1, which forwards data to Counter-Measure 2.

The components Gateway Sensor 1 and 2 are fed with 1000 KDD records each, and configured to select records randomly from the databases. We evaluate the detection and response mean times on components Counter-Measure 1 (CM1) and Counter-Measure 2 (CM2) in each scenario. Figure 8(a) presents the detection and response average times obtained in each scenario. Each alert on all scenarios is sent to all Counter-Measure components. The detect time is the same on Counter-Measure 1 and 2 and the graph shows only a line about detection time. Case 2 presents the best detection time while Case 1 the worst. Case 1 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. Case 1 registers the best response time, while Case 3 registers the worst. All response times were longer in CM2 than in CM1.

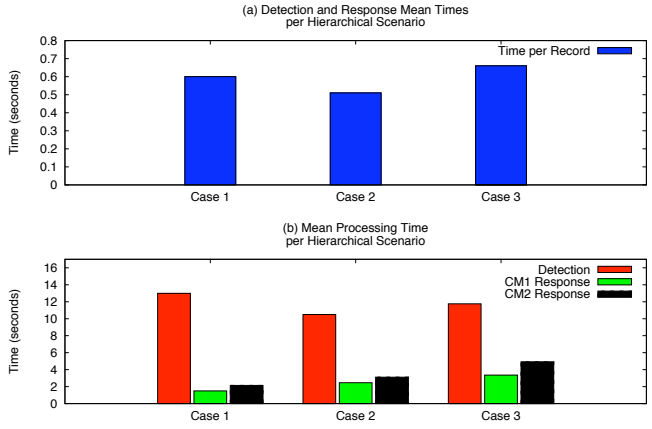

Fig. 8. Mean times of the hierarchical scenarios.

Moreover, this difference increases in Case 3, mainly because in this case, CM1 needs to forward alerts to CM2, and only later CM2 generates the responses. Case 2 presents an average time between the best and the worst time, but it has also the best detection time. Case 2 presents the best final time between collect time and response time.

According with Figure 8(b), Case 2 also presents the best average time needed to process a record. Case 3 registers the longest time required to process a record. It is also true for the response time. A hierarchical structure for Counter-Measure components therefore has not been very appealing. However, according with results presented in Figure 8, a hierarchical structure between aggregation and analysers has advantages. The experiments demonstrate the behaviour of the hierarchical structure in several parts of DIDSoG. The resources carry out tasks (data collection, aggregation, analysis and generation of alerts) in an integrated manner.

VI. Conclusions

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 (DIDSoG). IDSs in DIDSoG are encapsulated as Grid services for intrusion detection. A computational Grid platform is used for the integration by providing the basic requirements for communication, resource discovery, and security mechanisms.

Security, communication and resource discovery features are provided by Globus Toolkit. The authenticity of the DIDSoG resources, the confidentiality and the integrity of data rely on the Public Key Infrastructure provided by the Globus GSI. DIDSoG resources use WS-MDS to publish and locate the other DIDSoG resources.

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

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

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

Acknowledgments

We thank Kyong Hoon Kim from the University of Melbourne for sharing his thoughts on the topic. Marcos' PhD research is partially supported by National ICT Australia (NICTA).

References

[1] S. R. Snapp *et al.*, "DIDS: Motivation, architecture and an early prototype," in *Proceedings of the 15 IEEE NCSC*, Baltimore, October 1992.

[2] F. Leu *et al.*, "Integrating Grid with intrusion detection," in *Proceedings of the 19th IEEE AINA05*, March 2005.

[3] F. Leu, M. Li, and J. Lin, "Intrusion detection based on Grid," in *Proceedings of the Int. Multi-Conference on Computing in the Global Information Technology (ICCGI 2006)*, 2006.

[4] G. Feng *et al.*, "GHIDS: Defending computational grids against misusing of shared resources," in *Proceedings of IEEE APSCC '06*, China, 2006.

[5] P. Zhu *et al.*, "A new flexible multi-agent approach to intrusion detection for Grid," in *Proceedings of the 15th ICMLC2006*, August 2006.

[6] M. Wood, "Intrusion detection message exchange requirements," October 2002. [Online]. Available: www.ietf.org/internet-drafts

[7] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[8] I. Foster *et al.*, "Open Grid services infrastructure to WS-Resource Framework," ogsi_to_wsrf_1.0.pdf, May 2004. [Online]. Available: www. globus.org/wsrf/specs/

[9] ——, "The WS-Resource Framework," ws-wsrf.pdf, May 2004. [Online]. Available: www.globus.org/wsrf/specs/

[10] P. Silva *et al.*, "Composition of a DIDS by integrating heterogeneous IDSs on Grids," in *Proceedings of the 4th Int. Workshop on Middleware for Grid Computing (MGC 2006)*, Melbourne, Australia, November 2006.

[11] R. Buyya and M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience (CPE)*, vol. 14, no. 13-15, p. 11751220, November-December 2002. [Online]. Available: http://www.gridbus.org/papers/gridsim.pdf

[12] I. Foster, "A globus toolkit primer," August 2005. [Online]. Available: Available at www.globus.org/toolkit/docs/4.0/key

[13] D. Sterne *et al.*, "A general cooperative intrusion detection architecture for MANETs," in *Proceedings of the 3rd IEEE IWIA05*, March 2005.

[14] "The 15th international conference on KDD, 1999." [Online]. Available: kdd.ics.uci.edu/databases/kddcup99/kddcup99.html