A Grid-aware Intrusion Detection System

Michał Witold Jarmołkowicz

Kongens Lyngby 2007
IMM-THESIS-2007-109

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Abstract

Existing Intrusion Detection Systems (IDS) are not designed to deal with any specific types of systems. The purpose of this work is to investigate the possibility of Grid-focused IDS. The main stress is put on feature selection and performance of the system.
An existing framework, IDSNet, is used as a basis for considerations and development. An algorithm based on Self-Organizing Map has been selected for pattern discovery in traffic analysis.
No Grid environment was available for testing, therefore no real-life experiments could have been performed, and main focus was shifted to system performance and away from feature selection.
It is shown that the performance of the system greatly depends on the efficiency of the underlying framework. A number of optimizations are shown that improve the algorithm's performance by 2 orders of magnitude.

Preface

This M.Sc thesis was carried out at the Informatics and Mathematical Modeling department at the Technical University of Denmark. The project was completed under the supervision of Professor Robin Sharp and corresponds to 40 ECTS points.

I would like to thank Robin Sharp for long discussions throughout the project. I would also like to thank my manager Leo Moesgaard for allowing very flexible hours. I appreciate the support I received from all. Special appreciation goes to Jakub Jelonek for proofreading this document.

Lyngby, December 2007

Michał Witold Jarmołkowicz

Contents

Chapter 1

Introduction

1.1 Motivation

The author claims that no system can be assumed to be fully secure. The more complicated a system is, the more possible is that it will have security flaws that render it susceptible to intrusions, penetrations, and other forms of abuse. Even a theoretically flawless system would not be safe, if an insider decided to misuse his privileges. In real-life, systems with known flaws are not easily replaced by systems that are more secure. They may have attractive features that are missing in the more secure systems or the reason may be strictly economic. In all cases finding and fixing all deficiencies is not feasible [8].

Attacks take place on IT systems. An attacker will use any combination of attack strategies looking for the weakest link that allows to gain total or partial control of the system. Existing security measures and control components of an IT system may be bypassed, if an attacker succeeds at exploiting the system's deficiencies.

Network Intrusion Detection Systems aim at detecting attacks, that would cause loss of integrity, confidentiality, denial of resources, or unauthorized use of resources. Such systems have the potential to detect that an attempt is being made before the attack succeeds.

Above paragraphs highlight the importance of developing Intrusion Detection Systems. Such systems are considered as a second line of defense, intended to complement existing security measures such as authentication and access control. Along with the evolution of Grid technologies, the need for Grid-specific Intrusion Detection Systems, that would help to protect its resources is becoming more and more important.

1.2 Background Information

Application environment is a collection of protocols in which the application uses for communication (to either interact with other applications or for internal purposes).

Grid computing is an effort to exploit underutilized resources among a vast number of hosts. The vital differentiator of Grid computing from others types of computation is the middleware that is used. This is designed so that the applications can be run on clusters of computers and on distributed computing. The aim of the middleware is to provide all the support functions to ensure that the applications need.

Malicious Activity in general, is any form of activity that brakes the established security policies.

Neural Networks in general have the ability of representing non-linear relationships between input and output. A neural network would be capable of analyzing data, even if the data were incomplete or distorted.

Features is in general information extracted form or calculated based on data that characterizes some of its properties.

Intrusion Detection System is software that automates the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies.

Unsupervised Learning is a method of machine learning where the system learns itself by analyzing previously created data sets.

1.3 Problem Statement

The ultimate goal of any Intrusion Detection System (IDS) is to obtain as high detection rate as possible and as low false alarm rate as possible. In this project the aim is to investigate the possibility of focussing the system on application environments which are used in Grid computing.

The work shall focus on methods that use neural networks trained by unsupervised learning.

The main questions that need to be answered are:

- What are possible attacks on Grid systems?
- What are good features for Grid-aware IDS?
- What are good algorithms for Grid-aware IDS?

The anticipated outcome that is intended is a simple, but functional Grid-aware Intrusion Detection System, which can provide the user with useful information about attack patterns observed in real-life Grid computing systems.

1.4 Structure of this Thesis

This thesis can be divided into 3 main sections.
The first section, comprising chapters 2 and 3, introduces background information. In chapter 2, Grid Systems are presented, along with the current state of standardization in that area, followed by a brief study of attacks on Grids in section 2.2. Chapter 3 contains a section about theory of Intrusion Detection Systems (IDS) followed by a presentation and study of different approaches for designing IDS suitable for Grid environments.
The next section, composed of chapters 4–6, discusses the solution contained in this thesis. Chapter 4 contains a description of specification and development of an IDS aimed at Grid computing. Chapters 5 and 6 discuss the results obtained with the tool developed for this thesis.

The last section, chapters 7 and 8, present a consideration of possible future work and conclusions of the thesis.

Chapter 2

Grid Computing

2.1 Protocols in Grid

2.1.1 Introduction

In this section the recognized standards that form the base for Grid systems shall be identified. The goal is to provide a list of network protocols used in Grid Computing for the purpose of further study. The scope shall be limited to recognized standards.

2.1.2 Grid Middleware

The goal of Grid computing is to "create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources" [19]. To fulfill this goal many distributed systems were created, but not all can be classified as Grid systems. Grid systems are considered to be the network middleware connecting applications and network resources [26]. The middleware that is based on recognized standards, so as to facilitate cooperation between different

institutions and their implementations of Grid. The intention is to enable Grid computing over Enterprise Intranet, as well as over general Internet [26]. The middleware that forms the base of any grid, "provides all the support functions (...) that the applications need".[26]

A body named Open Grid Forum, formerly known as the Global Grid Forum (GGF), is attempting to standardize and document the processes and protocols that are, or will be, required to construct grids. It is a community initiative that includes participants of over 200 organizations in more than 30 countries.

A middleware that has implementations of the OGF-defined protocols and is currently most heavily used is the Globus Consortium's Globus Toolkit [26]. It is selected as an example of Grid middleware for this study and all further considerations shall be based on it.

2.1.3 Globus Toolkit

The toolkit is being developed mainly in the Argonne National Laboratory (ANL) and provides one of the most popular systems for furnishing the libraries and services for Grid computing [26]. It is based on the Open Grid Services Infrastructure (OGSI) standards.

The High Level Services implemented by the toolkit are independent of the network layer and shall not be considered in this study. What is relevant and shall be further analyzed are Grid Core Services (GCS).

2.1.4 Grid Core Services

Though it is possible to create mixed IPv4 and IPv6 environments for the Globus Toolkit and such implementations are proven to work, it is expected that IPv4 environments will persist for a long time [26]. Based on this prediction, IPv6 is moved outside the scope of this study.

During grid operation jobs are submitted to individual machines that form the grid. Job submission usually consists of three parts [19]:

1. A job is sent to a machine.

2. The job is executed on the machine

3. The results are sent back to the submitter.

There are four main components of interest to this study that are utilized during grid operations. They include components responsible for security, data management, information services and execution management. Brief description for selected parts of the components follows in the order stated:

Grid Security Infrastructure (GSI) provides security functions including single/mutual authentication, confidential communication, authorization, and delegation [1].

The infrastructure is based on the SSL protocol (Secure Socket Layer), public key encryption, and x.509 certificates.

All of the below components are built on top of the underlying Grid Security Infrastructure (GSI).

GridFTP is a key protocol intended to be used in all data transfers on the grid. It extends the standard FTP protocol. Globus Toolkit implementation supports most of the features defined in the GridFTP protocol [19].

There are two supported types of file transfer:

- Standard — Client sends a local file to the remote machine.
- Third-party — A file in one remote storage is sent to another remote server.

The data management component provides support to transfer files among machines in the grid and for the management of these transfers.

Monitoring and Discovery Service (MDS) provides support for collecting information (static and dynamic) in the grid and for querying this information.

It is based on the Lightweight Directory Access Protocol (LDAP).

It contains the following components [19]:

- Grid Resource Information Service (GRIS) — Repository of local resource information that is being registered with GIIS.
- Grid Index Information Service (GIIS) — Can be seen as a grid wide information server. Contains indexes of resource information registered by the GRIS and other GIISs.
- Information Providers — Have to be created to translate the properties and statuses of resources to GRIS.
- MDS client — Performs search for resource information based on the LDAP client command, ldapsearch.

Grid Resource Allocation Manager (GRAM) is the module that provides remote execution and status management of the execution.

It is based on RSL/HTTP1.1 protocol.

Consists of the following elements [19]:

- The globusrun command — GRAM client. Submits and manages remote jobs.
- Resource Specification Language (RSL) — the language used by clients for submit jobs (including the executable file and condition on which it must be executed).
- The gatekeeper daemon — Builds the secure communication between clients and servers. After authentication, gatekeeper forks and creates a job manager delegating the authority to communicate with clients.
- The job manager — Parses RSL scripts, allocates job requests to the local resource managers (job schedulers), sends callbacks to clients (if necessary), receives the status and cancel requests from clients, and sends output results to clients using GASS (if requested).
- Global Access to Secondary Storage (GASS) — Provides the mechanism to transfer the output file from servers to clients. Used by the globusrun command, gatekeeper, and job manager.
- Dynamically-Updated Request Online Coallocator (DUROC) — Using the DUROC syntax in RSL scripts, users are able to submit jobs to different job managers at different hosts or to different job managers at the same host.

2.1.5 Conclusions

The Globus Toolkit is the standard for building Grid solutions. It was selected as the base of this study. The protocols identified are:

- SSL (Secure Socket Layer)
- GridFTP
- LDAP (Lightweight Directory Access Protocol)
- RSL/HTTP1.1

2.2 Attacks on Grids

2.2.1 Introduction

In the previous section a number of protocols used in Grid Computing was selected. In this section the goal is to further define the particular application environments used in Grid and present examples of malicious behaviors in Grid.

2.2.2 Grid Application Environment

The hosts that build up the Grid may have other functions serving and/or using it. In most organizations, most desktop machines are busy less than 5 percent of the time. Server machines can as well be relatively idle. Grid computing provides a framework for exploiting those underutilized resources [19]. In this study, it is assumed that not all hosts are fully dedicated to the Grid. Hence, there are protocols other than Grid protocols operating across the network and the participating hosts may not be the same as the ones forming the Grid.

Most of the Globus Toolkit is created in Java. The system can span across many platforms running various operating systems.

The list of protocols includes fundamental network services.

Some other protocols may be present. This may be caused by the Globus system utilizing external applications or some other local or remote services that are independent of Globus.

The application environment in which a Grid may exist in extensive. To cover them would be outside the scope of this study. The focus shall be on the protocols that are specific to Grids.

2.2.3 Malicious Behavior in Grids

In general, the possible threats that every system faces include: attempted break-in, masquerading or successful break-in, penetration by legitimate user, leakage by legitimate user, inference by legitimate user, trojan horse, virus, Denial-of-Service [8].

The heterogeneous characteristic of Grid systems and the system being possibly spread over organizational and geographical boundaries lead to potential security issues [26].

Grid software has complex interactions and adds to the complexity of the system overall. A compromise at a single site can have very far reaching effects. The more complex the system, the harder it is to understand what is going on. [28]

An attacker will use any combination of attack strategies looking for the weakest link that allows to gain total or partial control of the system. Usually the final step of an attack to gain administrative access of a host is some form of local exploit. Once a system is compromised, backdoors are created to allow remote access and wholesale account harvesting may be employed [28].

Collaborative environments are especially vulnerable to identity theft. If reasonable network security measures are in place, identity theft is the most likely attack. Grid is very different environment than common authentication systems. The authentication in Grid is based on x509 certificates.

What the attacker needs to impersonate a user, is his private key. Practice shows that acquiring this information may not always be hard. Another set of issues comes from the problematic certificate revocation mechanism that happens to be not used or is poorly managed. Grid identity theft is a huge potential problem and has been poorly addressed in the current environments. [28]

The classic form of attack on Grid is exploiting a vulnerable service over the network. It is similar to existing remote exploit problems and is most common in environments where network security is inadequate. As the Grid services are exposed on the network, they are potential targets. The issue that is specific to Grid is arbitrary code that may be distributed as part of job request and run on the remote host. [28] The first attack on Globus was noted in late September

2001! Another significant one occurred in 2003, when GridFTP was discovered to be vulnerable to a buffer overflow attack that allowed administrative access to the host. [2] One has to remember that the client applications can be exploited in the same way, if a vulnerability is found.

Another, but not so common form of attack, may happen when a user who has limited access, finds vulnerability and uses it to gain administrative access. The problem is very similar to existing local exploit issues. This attack is either an "inside job", or was preceded by identity theft. Typically, it is harder to protect against this form of attack than against remote exploits. [28]

Previous paragraphs described various ways of gaining access to a Grid system. Malicious behavior can also be categorized according to the threat that it creates. The first category is often of most concern and is defined as attempts to read or modify confidential data. The second category stands for Denial of Service attacks, that prevent genuine users from accessing services. The last category defines a threat of using local network resources to launch attacks elsewhere. [2]

Port probing and other network mapping attacks are also considered malicious behaviors.

When studying GridFTP protocol alone, the following threats have been identified [7]:

- Denial of Service Attacks (DoS) — Deliberate attacks by malicious clients that subvert the overall performance by deliberately tying up resources of the GridFtp server.
- Flash Crowds — Sudden increase in the number of (legitimate) clients.
- Compromised client machines

2.2.4 Conclusion

The application environment specific to Grid systems have been further defined. The complexity of such systems has been outlined. The threats, attacks on Grid in general and on GridFTP in particular, as well as other forms of malicious behavior have been described.

Chapter 3

Intrusion Detection Systems

3.1 Introduction

This section is a result of study of the literature on the topic of network intrusion detection. The goal is to present different approaches to the problem and select those that fit the stated task best.

3.2 Basic Theory

An intrusion is characterized as "any use of the given network that compromises its stability and/or security of information stored across the network." [29]

Intrusion Detection Systems (IDS) can be classified as:

- signature detection systems
- anomaly detection systems

The first type of systems are rule-based and search for any kind of known abnormal behavior. Their strength is detection of known patterns. The second category characterizes systems that search for anomalies in the way the system is used. Their strength is in detecting unknown attacks and comes at the cost of false positives. An anomaly in not necessarily an indication of malicious behavior. It can occur as a result of legitimate system use that hasn't been observed before.

For an IDS to be able recognize any attack, it has to be trained. Based on how this is achieved, two types of systems can be distinguished:

- supervised
- unsupervised

As the name suggests, supervised systems require supervision of the learning process. As the task is complicated, it requires experts to be involved. A lot of data has to be manually processed, hence the procedure is error-prone. Unsupervised systems learn by themselves by analyzing data. Such data should be free of attacks. As this may be sometimes hard to guarantee, some systems allow statistically small (possibly insignificant) portion of the data to represent malicious behavior.

Misuse IDSs are trained in supervised manner. Anomaly IDSs use unsupervised learning but profit from manual fine-tuning.

Next classification characterizes how IDSs operate:

- off-line
- on-line

Off-line IDSs benefit from the fact that all data that is to be processed is available. They may be more accurate in their findings. The drawback is that the detection is made post factum. They are best suited for forensic applications or host-based IDSs.
On-line systems, also called real-time, have less data. The data may include only part (e.g. only the beginning) of an event at the time of analysis. Such systems are computationally very expensive, as they require continuous monitoring. Though not as accurate as off-line IDSs, those systems have the big advantage of detecting intrusions at they are happening.

Network IDSs can be classified according to the semantic level of the data that is analyzed:

- low-level
- high-level

Low-level systems make their analysis based on information that can be obtained without reconstructing network packets. This approach has the advantage of being fast, but the system may fail to recognize attacks that take place on the higher level of the connection.
High-level systems reconstruct the network packets and extract additional information about the interactions between hosts. There are examples of systems that implement only one of the approaches as well as both.

There are other classifications possible, but only the above are relevant to this work. This thesis focuses on high-level on-line anomaly IDS trained by unsupervised learning.

It is often desirable to filter the data before it is presented for analysis. Eliminating data that is not useful to the system may have different advantages depending on the system. For off-line IDSs the required amount of storage is

reduced. For on-line systems it improves processing times. In all cases the detection rate may be improved. If filtering is applied, it should be done with caution, as useful data may be discarded. [5]

3.3 Feature Selection

3.3.1 Introduction

It is assumed that malicious behavior involves abnormal use of the system [8]. To discriminate between malicious and non-malicious behavior some characteristics of the system use have to be extracted from the passing network traffic. Those characteristics are called features. "Feature construction is an important step in intrusion detection and involves tremendous efforts." [29]

The goal of this section is to develop features that are suitable for use in an Intrusion Detection System aimed at Grid Application Environment.

3.3.2 A feature

One could represent every observed value that passes over the network, but that may provide no data that is useful to an Intrusion Detection System. The goal is to provide some meaning to the data. Hence, feature information is extracted based on a priori knowledge of communication protocols in use (the construction of the data). Feature information can be extracted from different layers of the protocol stack. The higher the level, the more reconstructive work has to be done.

A feature may be representing the value of some field in a given protocol. It may as well be a combination of a number of fields. The value may be continuous or categorical in nature. A feature may as well be a statistic that is calculated based on available data. In general, it may be any function.

Features may be constructed based on data that is treated just like a stream of bytes. Meaning that is has no assumed format, set of keywords, expected tokens or limited range of values, unlike the reconstructed information.

Algorithms that are particularly useful for feature construction [13]:

- Classification — Maps a data item into one of several pre-defined categories.
- Link analysis — Determines relations between fields.
- Sequence analysis — Models sequential patterns. These algorithms can

discover what time-based sequence of events are frequently occurring together.

3.3.3 How to select good features?

A good feature is one that has good discriminating power for given data. The better the information gain the better the feature. Information gain is defined as "the reduction in entropy, which characterizes the impurity of a dataset." [13]

Feature ranking and selection is an important issue [18]. Some features are truly useful, some are less significant and some are useless. Selecting the right set of features for Intrusion Detection Systems (IDS) is critical. Performance of an IDS system in terms of both accuracy and speed, depends directly on the choice of features [13][23]. The speed is especially important in real-time IDS systems [5].

The problem is similar in nature to other engineering problems [23]:

- Having a large number of inputs variables of varying degrees of importance: Some are essential, some are less important, some of them may not be mutually independent, and some may be useless or noise.
- Lacking an analytical model or mathematical formula that precisely describes the input-output relationship.
- Having available a finite set of experimental data, based on which a model can be built for simulation and prediction purposes.

The author could find little research in the topic of efficient feature selection for Intrusion Detection Systems. "There does not exist any model or function that captures the relationship between different features or between the different attacks and features. If such a model did exist, the intrusion detection process would be simple and straightforward." [5] A complete analysis that would identify such relationships is both infeasible and not infallible [23].

An interesting approach to the problem is presented in [13]. The impression of the author is that the described process of feature selection is unquestionably accepted by the research community. The paper instructs how to, in an iterative manner, apply data mining tools to data representing connection records, to compute "frequent patterns", that are used to construct new good features. The author recognizes that the process is flawed by the inherent problem of selecting a representative dataset.

No other approaches to the problem could be identified. The art of constructing good features is mostly considered to be a guess work. Many papers present algorithms for downsizing a preexisting set of features [23][5][22][6][12]. Better performance of an IDS can be achieved when those are applied — improved computation speed with no statistically significant change to accuracy. Examples include use of evolutionary procedures, genetic algorithms and data mining tools. One algorithm is here selected because of its simplicity and speed. The iterative process is defined in the following way [23]:

1. Delete one input feature from the data
2. Use the resultant data set for training and testing the IDS
3. Analyze the results of the IDS — accuracy of classification, training time and testing time
4. Rank the importance of the feature — improved performance, no change or worsened performance
5. Repeat steps 1 to 4 for each of the input features.

3.3.4 Feature Choice for Grid-Aware IDS

Depending on what assumptions are made about the traffic inside the Grid, different set of features should be considered.

One of the properties of Grid Security Infrastructure (GSI) is confidentiality of the data transfered over the network. If it is employed, some of the benefits of a network IDS would be lost. The system could not see the data payload portion of the packet because of encryption. Analysis would be based only on the low-level information, that can be extracted from the packet header. [19] The source and destination host address is not a good feature in an Grid-Aware IDS. The author's believe in this statement is based on the fact of very dynamic nature of Grid systems. The author does not claim that the information should be discarded. It could be used in some temporal features that describe the amount of traffic to the same host in last n seconds.

If the data transferred over the network is not encrypted, the IDS could additionally benefit form high-level features extracted form the data. In this case, the following would be good features in the author's opinion:

- For LDAP protocol, a categorical feature expressing the operation that is requested (e.g. search, add, modify, delete)

- For RSL/HTTP1.1 a feature that would model the '`<job>...</job>`' part of the request. For example in the way the PAYL [27] system models payload, creating different models for different lengths of the modeled parameter.
- For SSL protocol, a feature that would measure the entropy (should be high when SSL is used), or one that expresses the distribution of data (should be close to uniform when SSL is used).
- For GridFTP protocol, a categorial feature expressing what kind of file transfer is used — standard or third-party.

3.3.5 Conclusion

Only one algorithm that constructs good features was identified [13]. It should be applied if possible, as data mining tools applied would most likely uncover good relationships that a human would fail to recognize. The very least that should be done for an IDS aimed at Grids is feature reduction. An iterative algorithm that serves this goal was identified [23].

A drawback that appeared while conducting this research project is that no access to real Grid system was given to the author. A number of attempts to secure such access were made well in advance. One was terminated at IBM Nordic Security Group, based on company policy that strictly forbids sniffing in company networks. Other attempts failed for the same as well as other reasons. Having no real Grid data to base this research on, forced the author to change the focus of this study. All algorithms identified in the previous section require such data. Having no means to verify his claims about good features, the author ends his analysis of feature selection offering only a few claims about good features for Grid-Aware IDS. The focus from now on shall be on efficient algorithms for Intrusion Detection Systems with the goal of implementing a system that can handle high amount of traffic. It is an important goal for an IDS aimed at Grid, as high traffic is an inherent characteristic of Grid computing.

3.4 Feature Preprocessing

The data extracted from network traffic has to be preprocessed before it is analyzed. The feature set may contain categorical and numerical features of different sources and scales. An essential step for handling such data is metric embedding which transforms the data into a metric space [11]. A good approach is presented in [11] where all features are scaled with respect to each feature's mean and standard deviation.

3.5 Intrusion Detection Systems

3.5.1 Supervised Signature Detection

Supervised Signature IDSs are based on classification of data instances. They require labeled data to be trained. Labels are difficult or sometimes impossible to obtain. The task of analyzing network traffic logs is very time-consuming. Usually only small portion of available data can be labeled. Sometimes it may be impossible to unambiguously assign a label to a data instance. One can never be sure that a set of available labeled examples covers all possible attacks. New attacks may not have been seen in training data [11]. Detection of intrusions is often ineffective in dealing with dynamic changes in intrusion patterns and characteristics [29].

They perform extremely well at detecting known intrusion attempts and exploits. When data contains no unknown attacks, they outperform unsupervised methods. The performance of all supervised methods drops significantly when unknown attacks are present. They fail to recognize new attacks and carefully crafted variants of old exploits. It is always one step behind the attackers [21][11][27].

3.5.2 Unsupervised Anomaly Detection

Unsupervised Anomaly IDSs are based on the assumption that exploitation of a system's vulnerabilities involves abnormal use of the system. What they are designed to detect are abnormal patterns of system usage [8]. Unlike supervised methods, they do not require a laborious labeling process [14]. What they require is a set of purely normal traffic data. The data is used to create models

of normal patterns [29]. The task of every anomaly IDS is to determine whether activity is unusual enough to suspect an intrusion." [8][27]

Their advantage is the ability to find new attacks not seen before [29]. The performance is similar to the performance of the supervised learning when unknown attack are present. Compared to supervised classification methods, it is relatively easy to accommodate for changes in behavior and characteristics of network attacks, when unsupervised anomaly IDS is used [29].

Their disadvantage is that well-known attacks may not be detected, particularly if they fit the learned profile of normal system use. When an attack is detected it is often difficult to characterize the nature of the attack for forensic purposes [5].

Anomaly IDSs based on unsupervised learning should be the method of choice, if unknown attacks can be expected [11][29].

3.5.3 Selected Algorithms

This section lists a selection of solutions that performed best in tests and were considered as good candidates for later implementation. While the focus is on Self Organizing Maps, that have been chosen for later implementation, the author wishes to also name different approaches to the problem.

Standard k-means algorithm is widely used. The popularity is largely due to its simplicity, low time complexity, and fast convergence [29].

PHAD and ALAD. PHAD stands for **P**acket **H**eader **A**nomaly **D**etection. It uses attributes stored in the fields of packet headers. ALAD stands for **A**pplication **L**ayer **A**nomaly **D**etection. The attributes are the application protocol keywords, opening and closing TCP flags, source address, and destination address and port number. Both components are anomaly detectors. The IDS described by Mahoney and Chan [20] consists of both PHAD and ALAD running at the same time. It is selected as it performed almost as well as system that combined both signature and anomaly detection.

Data Mining approaches are new methods in Intrusion Detection Systems. They have been used before for anomaly detection. Data mining attempts to extract knowledge in the form of models from data, which may not be seen

easily with the naked eye [5]. The solution presented by Chebrolu et al.[5] is an interesting approach. It has the advantage of discovering useful knowledge that describes a user's or program's behavior from large audit data sets.

Support Vector Machines have proven to be good candidates for intrusion detection because of its speed and scalability. These are learning machines that plot training vectors in high-dimensional feature space, labeling each vector by its class [5].

Self Organizing Map algorithm is computationally efficient. It performs extremely well at very low false positive rates [29]. It is one of the most popular neural network models. Two main advantages of using SOMs are their efficient update scheme and the ability to compress high dimensional information into a 1- or 2-dimensional space, while preserving the most important topological and/or metric relationships of the input data [10][14]. Typical connection characteristics will be emphasized and could be observed by the user as densely populated regions of the map. Atypical activities will appear in sparse regions of the topology [18]. The fact that the topological map is defined by the user allows for good visualization [29]. This features of SOM make it very attractive for this work.

Artificial neural networks in general have the advantage of easier representation of nonlinear relationships between input and output.
A neural network would be capable of analyzing the data from a network, even if the data were incomplete or distorted. Another advantage of neural networks is its inherent computational speed [5].

The drawback of SOMs is that the performance of clustering depends on the number of neurons, which is predefined at the start. To obtain better performance, various numbers of neurons may need to be tested [14].

3.5.4 Combining Anomaly and Signature Detectors

As different systems use different features and different algorithms, correlating their results may significantly improve results [27][20][29]. A common approach is to combine the use of anomaly and misuse detection [11].

3.5.5 Handling Enormous Traffic

Leu, Lin et al. [17] present a solution to enormous network traffic. Grid architecture is used to gain plentiful computing resources to cooperatively detect high volume network packets, especially when DDoS attack happens. The authors claim that their system, unlike a single-node IDS, will not suffer from high volume attacks.

The approach to utilize Grid resources to improve performance of Intrusion Detection Systems is presented in [17][15][16].

3.5.6 IDS Training

The common problem for all anomaly detection IDS is the one of selecting good training data. If there is not enough training samples, the empirical distribution may lay far from the true distribution. This will lead to a faulty detector [27].

A high false positive rate may be a result of a narrowly trained detection algorithm. A high false negative rate may result for a broadly trained system [5].

3.5.7 IDS-Aware Attacks

Mimicry attacks are a possible threat. An attacker may attempt to replicate normal behavior to "hide" his attack. If the algorithm presented by Wang and Stolfo [27] is used, the attacker would need to sniff for a long period of time and analyze the traffic in the same fashion as the detector, and would also then need to figure out a way to insert his poisoned payload to mimic the normal model. The problem is not solved, but the task is a harder one for the attacker.

If an attacker knows that he is being profiled by an anomaly IDS, he can change the profile slowly over time to essentially train the IDS to learn his malicious behavior as normal [5].

3.6 IDS Testing

There are usually two scenarios under which an IDS system is tested. Under the first scenario, both training and test data come from the same distribution.

Under the second scenario, attacks unseen in training data are present in test data. This is a typical scheme to test the ability of an IDS to cope with unknown attacks [11].

It is very hard, if possible at all, to generate data that is a good sampling of actual intrusions. One problem is to keep the data distribution realistic. Another one, that no matter how much data one collects, there may still not be enough for some extremely infrequent case. On the other hand, is is possible to generate a realistic sample of unusual behavior [25].

A common benchmark for evaluating IDSs is the 1999 DARPA data set collected at MIT Lincoln Labs to evaluate Intrusion Detection Systems. What is relevant to this work is the data recorded in tcpdump format. All data is publicly available. It includes the entire payload of each packet that was recorded. This data set has been used in many research efforts [11][18][29][3] and the results obtained are reported. As the quality of data suffers due to its artificial nature, practically every research paper presents a unique way to normalize the data. Researchers also tend to select different portions of the data for their tests [27][20]. Based on that the author doubts if it may be used to effectively compare reported results. Nevertheless it remains a useful indicator in comparing various techniques [27]. More importantly it allows researchers to verify results claimed by others. The attacks contained in this data set are now out of date, but it may still be used to evaluate anomaly based systems, that have no knowledge of the attacks. It is not the case for signature detectors [21]. One of the problems identified in this data is the unnatural attack rate. Another one is small fixed range of many attributes that would have large and growing range in real traffic [21]. Main problems are addressed if real traffic is injected into the data.[21].

The performance of an IDS is evaluated based on false positive rate and attack detection rate. The false positive rate is the percentage of normal instances that are labeled as attacks. The attack detection rate represents the percentage of all attack instances that are detected. Some researchers [29][11] use ROC (Receiver's Operating Characteristics) to show the tradeoff between the false positive rate and the detection rate.

3.7 Conclusion

An overview of different approaches to ingredient tasks and issues in building an Intrusion Detection System was presented. The findings made in this section will be used to develop an Intrusion Detection System aimed at Grid Computing.

Chapter 4

Solution

4.1 Introduction

The goal is to implement a simple Intrusion Detection System aimed at Grid computing. Previous chapters aimed at selecting methods, based on a study of literature, that facilitate this goal. This section describes the specification and development of the system that exploits the chosen methods.

4.2 Solution proposal

The focus is on performance in terms of computational load. The requirement shall be that the system can handle high amount of traffic, what is characteristic for Grid computing.

The pattern discovery component shall employ an unsupervised learning system based on Self Organizing Map (SOM) [9] to detect and visualize the characteristics of a common connection.

Though the author has no means, due to lack of Grid data-sets, to employ the algorithm selected for feature reduction, the system shall be build in a way that

facilitates the needs of the algorithm.

The IDSNet system that was developed and used in previous research [4][24] shall be studied with the goal of assessing its value as a framework for this work. The system shall be reused if possible.

The system shall provide the user with useful information about attack patterns.

4.3 Self-Organizing Map Algorithm

Before the algorithm is started the synaptic weights of all l neurons in the network have to be initialized. It is done by picking small values from a random number generator (preferably from the domain of the input samples [10])

The algorithm is iterative and has three steps [9]:

1. Competition: For each neuron $j \in (1, 2, ..., l)$, Euclidean distance is calculated between the current input feature vector $x = [x_1, x_2, ..., x_m]$ and the synaptic weights of the neuron $w_j = [w_{j1}, w_{j2}, ..., w_{jm}]$. This forms the basis for competition among the neurons. The neuron with the lowest value is the winner of the competition.

2. Cooperation: Neurons in the neighborhood of the winner are selected. Winner defines the center of neighborhood. The width of the neighborhood $o(n) = o_0 * exp(\frac{-n}{t_1})$ has to decrease with the number of iterations n. Gaussian function $h_{j,i(x)} = exp(\frac{-distance^2(i,j)}{2*o^2(n)})$ is used as the topological neighborhood. The width measures the degree to which neurons in the vicinity of the winner participate in the adaptation process. As the number of iterations n increases, the width decreases at an exponential rate. This provides the basis for cooperation.

3. Synaptic Adaptation: Allows the network to adapt. The synaptic weights of the winner and the selected neighboring neurons are updated. The formula $w_j(n+1) = w_j(n) + r(n) * h_{j,i(x)}(n) * (x - w_j(n))$ is applied to all neurons inside topological neighborhood of winning neuron i. The learning parameter $r(n) = r_0 * exp(\frac{-n}{t_2})$ should decrease gradually with time.

The iterations of the algorithm are divided into two phases [9]:

1. Ordering: It may take as many as 1000 iterations and possibly more. The learning parameter $r(n)$ should begin with a value close to 0.1, then decrease gradually, but not bellow 0.01. This condition is satisfied if the constant $r_0 = 0.1$ and the constant $t_2 = 1000$. $h_{j,i(x)}$ should initially include almost all neurons in the network around winner and shrink slowly. Satisfied, if the constant o_0 is equal to the radius of the network and the constant $t_1 = \frac{1000}{log(o_0)}$. The neighborhood may become the winner only.
2. Convergence: The number of iterations must be at least 500 times the number of neurons l for the network to be able to provide accurate statistical quantification of the input space. In this phase the learning parameter r_0 should be maintained at a small value on the order of 0.01 and can never decrease to zero. The neighborhood function should include only nearest neighbors or the winner only.

The results obtained by Zhong et al. [29] suggest that the network size should be set to 200 neurons. In that configuration the algorithm performed extremely well at very low false positive rates. This was not the case when the network size was set to 100 neurons.

4.4 IDSNet

The first task was to make the IDSNet executable in its initial state. In previous research [4][24], the operating system on which the system was run consisted of earlier version on libraries required for the system operation. The author's operating system is a custom build Linux compiled from the newest stable releases of Gentoo distribution. The differences between the earlier utilized libraries and the ones present in the authors system resulted in failed compilation of the IDSNet. The task was additionally complicated by the fact that the sources contained earlier approaches that were abandoned. Hence, it was initially not obvious what components of the system are relevant. The task required a lot of effort.

4.5 Architectural Overview

This section presents a brief overview of the architecture of the IDSNet.

The author identified the following main components that are relevant to this work:

- FormMonitor class — main GUI; uses Monitor class
- Monitor class — control flow component
- Device class — I/O component
- Source class — input component; extends Device class
- Destination class — output component; extends Device class
- Ethernet class — component providing input from ethernet devices; extends Source class
- FormSomsensor class — GUI for Somsensor
- Somsensor class — component providing output to SOM algorithm; extends Destination class
- CPacket class — represents single IP packet that can be taken from Source component or put to Destination component
- CConnection class — represents a connection and provides connection features
- CTrafic class — represents known CConection components and provide traffic features
- Somnn class — represents a neural network
- Somnet class — an interface between the Somsensor component and the Somnn component
- FeatureVector class — represents features extracted from CConnection and CTraffic components

4.6 Data flow

This section describes the flow of data in the IDSNet in a scenario relevant to this work.

1. An active Source component obtains an IP packet and creates a CPacket object.
2. The Monitor components collects the CPacket from the active Source component.

3. The Monitor provides each active Destination component with the collected CPacket.
4. An active Somsensor component receives the CPacket.
5. The Somsensor uses CTraffic and CPacket to obtain a CConnection object
6. FeatureVector component is used to extract features from the CConnection object
7. The Somsensor provides feature values to a Somnet component.
8. The Somnet component passes the feature values to a Somnn component.
9. Somnn analyses feature values and returns result to the Somnet component.
10. Somnet passes the result to Somsensor.
11. Upon user request, FromSomsensor retrieves results from Somsensor.

4.7 Identified Deficiencies

This section describes identified deficiencies of the IDSNet, that have to be addressed if the stated goals are to be met.

- Feature selection is hard-coded.
- SOM input vector size is hard-coded.
- Not possible to manually switch between SOM training and detection.
- Not possible to additionally train SOM after training finishes.
- No anomaly detection, as cluster labeling is not implemented.
- Issues in performance of SOM algorithm, that may not be addressed by the compiler.
- SOM algorithm code quality raises questions of its correctness.
- Usability suffers, as the initial system configuration is hard-coded.
- FormSomsensor GUI not properly displayed.
- Handling of tcpdump format not implemented.

4.8 Functional Decomposition

This section describes the (abstract) functional components of the intended IDS system. These components will later be mapped to an actual design consistent with the IDSNet framework.

Configuration Provider component reads a configuration file and provide the obtained information to other components.

Data Acquisition component acquires data that has to be analyzed.

Data Filtering component filters acquired data using simple rules.

Feature Extraction component extracts feature information from data.

Feature Preprocessing component allows scaling of feature information.

Analysis component that is trained and used to detect anomalies.

Presentation component presents results of the analysis.

User Interface allows user interaction.

Control Flow integrates the system.

4.9 High Level Design

This section describes the high-level design for IDSNet implementing the required functional components. Only classes relevant to this work are presented. During later low-level design more details must be added.

Configuration Provider component shall be implemented by a new ConfigurationProvider class. It shall be a static class initialized at system start. The class shall read and parse java-like property files. It shall support commented line (those that start with '#') and empty lines. A configuration entry shall be of the format "key=value".

Data Acquisition component is implemented in IDSNet by classes derived from Source class like Ethernet class. The ethernet class uses pcap library. The implementation shall be extended to allow offline use of pcap — reading of tcpdump files.

Data Filtering component is implemented in IDSNet by PacketFilter class. The class is used by the Data Acquisition component to filter out unwanted data using simple rules based on IP number, port number and protocol type.

Feature Extraction component is implemented in IDSNet by CPacket, CConnection, CTraffic and FeatureVector classes. FeatureVector class has to be changed to allow for selection of features. The configuration shall be taken from the Configuration Provider component.

Feature Preprocessing component is implemented in IDSNet in a very efficient way by many classes derived from Function class. The functions may take not only the feature value as argument, but also other functions. Each feature type is assigned a hard-coded function. The preprocessing is automatically applied when feature values are requested from FeatureVector class. The design shall not be changed. Configuration Provider component can be used in future to allow for configurability of feature preprocessing.

Analysis component in implemented by Somsensor, Somnet and Somnn classes. The implemented algorithm has to be verified. The implementation has to be studied for possible performance optimizations, as it is responsible for the computational complexity of the whole solution. The implementation has to be changed to allow for configured input vector size. Design has to be changed to facilitate manual switching between training and detection. Simple cluster labeling mode needs to be implemented.

Presentation component is implemented in IDSNet by FormMonitor, FormSomsensor and FormPacketFilter classes. Slight modifications to FormSomsensor class are required to facilitate new functionality of anomaly detection. Clusters that represent detected attacks have to be distinguishable from normal clusters. Somsensor shall be extended to write information about detected attacks to a file. The file format shall be Comma Separated Values to allow for easier visualization and further analysis. The entries that are written to the file shall contain timestamp, destination address, port number, protocol type, number of the cluster and values of all features.

User Interface component is implemented in IDSNet by FormMonitor, FormSomsensor and FormPacketFilter. Slight modifications have to be made to FormSomsensor and/or FormMain, so the GUI is properly displayed.

Control Flow component is implemented in IDSNet by Monitor class. To increase usability, a list of tcpdump files shall be taken form the Configuration Provider component during initialization.

4.10 Low Level Design

The scope of this section is optimization of the SOM algorithms. All implementation details not specified before are left to the authors discretion.

In-depth analysis of the SOM algorithm allowed the author to offer a very significant optimization. As the number of iteration increases, the width of the neighborhood decreases. At the point when it includes only the winner, the complexity of the algorithm may be decreased from the initial $O(lm)$ to $O(m)$, where l is the number of neurons and m is the size of the input vector. As the number of neurons is at least 100, the algorithm will perform 2 orders of magnitude faster. The approach requires that a value of the neighborhood function for two closest clusters is evaluated every n iterations, until it reaches zero. At that point the algorithm should be switched to a mode, where it only updates the weights for the winner, instead of all neurons. Further optimizations are possible. At that point it is no longer necessary to evaluate the neighborhood function, as it will be always equal to 1 for the winner.

4.11 Conclusions

The IDSNet framework was modified and extended according to the design provided in this section. The solution uses an efficient SOM algorithm as the detection engine. It facilitates the needs of the selected feature reduction algorithm that promises increase in performance when applied. The tool provides the user with useful information about attacks in the form of the SOM specific topology map that reflects the relationships in input data. If an anomaly is detected an entry for each packet that was classified as anomalous is created in the detection log.

Chapter 5

Evaluation

This section provides important results obtained with the developed tool.

5.1 SOM Test

The goal is to check if the implementation of the SOM algorithm is not flawed. The test shall be considered successful if the map produced by the system has the intended characteristics — clearly identifiable groups of densely populated areas. The test was carried out using data from the DARPA 1999 IDS evaluation. The system was first trained on an attack free sample of the data. The second step used another attack free sample of the date to label good clusters. The test was successful. Results are shown on figure 5.1.

5.2 Performance Test

As the focus of this research is on performance, such tests have been carried out and the results are reported here. The tests were carried out on a 1.6GHz

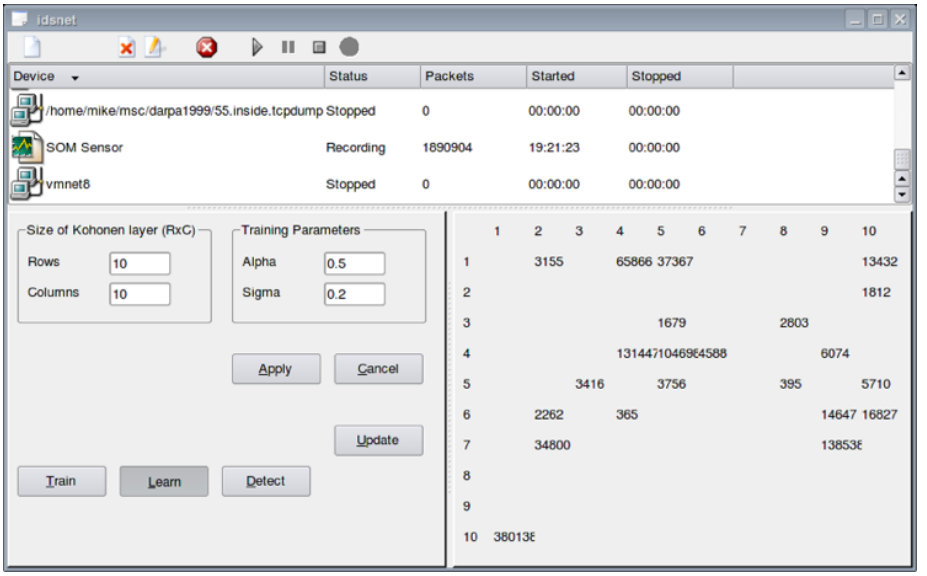

Figure 5.1: Test results

Pentium-M laptop. The neural network size was set to 100. A very simple strategy was selected. One of the DARPA 1999 IDS evaluation tcpdump files was selected. The time of processing the data was measured for various configurations of the system. The results are provided in the form of throughput.

Playing the file (no feature extraction): 11856 kBps.

Playing the file and extracting 1 feature: 777 kBps.

Playing the file, extracting 1 feature and detection: 711 kBps.

Playing the file and extracting 18 features: 658 kBps.

Playing the file, extracting 18 features and detection: 499 kBps.

A rough calculation shows that the implemented SOM algorithm constitutes ca. 8.5% of the total calculation time for the 1 feature test and 24% of the calculation time for the test with 18 features.

Chapter 6

Discussion

The focus of this work is on providing the best algorithms and implementing them in an optimized way. The results obtained through the system evaluation are far from anticipated. It was assumed that the detection engine will consume most of the computational resources, while the data showed otherwise. The parts contributed by this work to the IDSNet perform well and the changes made to the design of the framework allow for optimization of the solution. The anticipated outcome was a simple, yet functional IDS. The framework requires further development to be feasible for real-time Intrusion Detection System deployed in Grid environment.

Chapter 7

Future work

The next step would be to test the methods selected in this thesis in a real Grid environment. The problem of performance of the underlying IDSNet framework would have to be addressed.

An interesting task would be to use a framework like the IDSNet to combine different detection engines. Correlation of detection results may greatly improve performance.

A field that requires more research is automated feature construction.

Chapter 8

Conclusions

The aim of this study was to investigate the possibility of focussing an Intrusion Detection System (IDS) on application environments which are used in Grid computing. The author characterized Grid systems and identified various forms of malicious behavior in Grid. The work continues with introducing the concept of features and presenting algorithms that aid the task of feature selection. The author offers a list of features, that he expects to be good for Grid-aware IDS. The claims need to be verified, but the task is removed from the scope due to unavailability of Grid data-sets on which they could be tested. An overview of different approaches to Intrusion Detection is presented. Self-Organizing Map is selected, as it fits the needs of this project. It is efficient and has compelling data visualization properties. As the offered features cannot be tested, the design of the system focuses on computational performance. Optimizations of the SOM algorithm are presented that increase the performance of training by two orders of magnitude. When tested, the overall performance of the developed system is unexpectedly low. The problem is identified to lay in the underlying framework that was chosen as the base for the developed system.

Bibliography

[1] Globus toolkit 4.0 release manuals. Website, 2007. http://www.globus.org/toolkit/docs/4.0/.

[2] J.V Hajnal D. Rueckert D.L.G Hill A.L. Rowland, M. Burns. Using grid services from behind a firewall, 2005. http://www.allhands.org.uk/2005/proceedings/papers/402.pdf.

[3] Damiano Bolzoni, Sandro Etalle, Pieter H. Hartel, and Emmanuele Zambon. Poseidon: a 2-tier anomaly-based network intrusion detection system. In *IWIA*, pages 144–156. IEEE Computer Society, 2006. http://doi.ieeecomputersociety.org/10.1109/IWIA.2006.18.

[4] M. Bornhøft and M. Carvalho. Hierarkisk distribueret IDS ved anvendelse af neural net. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2004. Supervised by Prof. Robin Sharp.

[5] S. Chebrolu, A. Abraham, and J. Thomas. Feature deduction and ensemble design of intrusion detection systems. *Computers and Security*, 24(4):295–307, 2005.

[6] Yuehui Chen, Ajith Abraham, and Ju Yang. Feature selection and intrusion detection using hybrid flexible neural tree. In Jun Wang, Xiaofeng Liao, and Zhang Yi, editors, *ISNN (3)*, volume 3498 of *Lecture Notes in Computer Science*, pages 439–444. Springer, 2005.

[7] Onur Demir, Michael R. Head, Kanad Ghose, and Madhusudhan Govindaraju. Securing grid data transfer services with active network portals. In *IPDPS*, pages 1–8. IEEE, 2007.

[8] Dorothy E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2):222–232, 1987.

[9] Simon Haykin. *Neural Networks: A Comprehensive Foundation, International Edition*, chapter Self-Organizing Maps. Prentice Hall, second edition, 1999.

[10] T. Kohonen. Self-organizing maps, volume 30 of springer series in information sciences. springer, 1995. (second extended edition 1997)., 1995. http://books.google.com/books?hl=en&lr=&id=e4igHzyfO78C&oi=fnd&pg=PA1&dq=kohonen+self+organizing+maps+springer+series+information+&ots=tgTLnCTl9H&sig=E59pBDRCaFbzBsQ_60RsN9lYjT4#PPA107,M1.

[11] Pavel Laskov, Patrick Düssel, Christin Schäfer, and Konrad Rieck. Learning intrusion detection: Supervised or unsupervised?. In Fabio Roli and Sergio Vitulano, editors, *ICIAP*, volume 3617 of *Lecture Notes in Computer Science*, pages 50–57. Springer, 2005. http://dx.doi.org/10.1007/11553595_6.

[12] Chi Hoon Lee, Jin Wook Chung, and Sung Woo Shin. Network intrusion detection through genetic feature selection. In *SNPD-SAWN '06: Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06)*, pages 109–114, Washington, DC, USA, 2006. IEEE Computer Society.

[13] Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *Information and System Security*, 3(4):227–261, 2000. http://citeseer.ist.psu.edu/article/lee00framework.html.

[14] John Zhong Lei and Ali A. Ghorbani. Network intrusion detection using an improved competitive learning neural network. In *CNSR*, pages 190–197. IEEE Computer Society, 2004. http://csdl.computer.org/comp/proceedings/cnsr/2004/2096/00/20960190abs.htm.

[15] Fang-Yie Leu, Ming-Chang Li, and Jia-Chun Lin. Intrusion detection based on grid. In *ICCGI '06: Proceedings of the International Multi-Conference on Computing in the Global Information Technology*, page 62, Washington, DC, USA, 2006. IEEE Computer Society.

[16] Fang-Yie Leu, Jia-Chun Lin, Ming-Chang Li, and Chao-Tung Yang. A performance-based grid intrusion detection system. In *COMPSAC '05: Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05) Volume 1*, pages 525–530, Washington, DC, USA, 2005. IEEE Computer Society.

[17] Fang-Yie Leu, Jia-Chun Lin, Ming-Chang Li, Chao-Tung Yang, and Po-Chi Shih. Integrating grid with intrusion detection. In *AINA '05: Proceedings of the 19th International Conference on Advanced Information Networking and Applications*, pages 304–309, Washington, DC, USA, 2005. IEEE Computer Society.

[18] P. Lichodzijewski, A. Zincir-Heywood, and M. Heywood. Dynamic intrusion detection using self organizing maps. In *the proceedings of the 14th Annual Canadian Information Technology Security Symposium*, 2002.

[19] Jonathan Armstrong Mike Kendzierski Andreas Neukoetter Masanobu Takagi Richard Bing-Wo Adeeb Amir Ryo Murakawa Olegario Hernandez James Magowan Norbert Bieberstein Luis Ferreira, Viktors Bersits. Introduction to grid computing with globus, 2003. http://www.redbooks.ibm.com/abstracts/sg246895.html?Open.

[20] M. Mahoney and P. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks, 2002. http://citeseer.ist.psu.edu/mahoney02learning.html.

[21] Matthew V. Mahoney and Philip K. Chan. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection, 1999. http://citeseer.ist.psu.edu/mahoney03analysis.html.

[22] Srinivas Mukkamala and Andrew H. Sung. Feature ranking and selection for intrusion detection systems using support vector machines. http://citeseer.ist.psu.edu/583136.html.

[23] Srinivas Mukkamala and Andrew H. Sung. Performance based feature identification for intrusion detection using support vector machines. In Ajith Abraham, Javier Ruiz del Solar, and Mario Köppen, editors, *HIS*, volume 87 of *Frontiers in Artificial Intelligence and Applications*, pages 351–364. IOS Press, 2002.

[24] A. Oksuz. Unsupervised intrusion detection system. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2007. Supervised by Prof. Robin Sharp, IMM, DTU.

[25] Jake Ryan, Meng-Jang Lin, and Risto Miikkulainen. Intrusion detection with neural networks. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998. http://citeseer.ist.psu.edu/ryan98intrusion.html.

[26] Peter Kirstein UCL Sheng Jiang, Piers O'Hanlon. The combination of ipv6 and grid systems, 2002. http://www.gridtoday.com/05/0117/104472.html.

[27] K. Wang and S. Stolfo. Anomalous payload-based network intrusion detection, 2004. http://citeseer.ist.psu.edu/wang04anomalous.html.

[28] Steve Chan William Kramer. Risks of being on the grid, 2004. http://grid.ncsa.uiuc.edu/ggf12-sec-wkshp/panel3/kramer-GridRisks.ppt.

[29] Shi Zhong, Taghi M. Khoshgoftaar, and Naeem Seliya. Clustering-based network intrusion detection. *International Journal of Reliability, Quality and Safety Engineering*, 14(2):169–187, 2007.