This is the author-generated version of an article published in *Dependable Computing Systems: Paradigms, Performance Issues and Applications*, edited by Hassan B. Diab and Albert Y. Zomaya. New Jersey: Wiley, 2005.

Safeguarding Critical Infrastructures

David Gamez, Simin Nadjm-Tehrani, John Bigham, Claudio Balducelli, Kalle Burbeck, Tobias Chyssler

1. Introduction

Large complex critical infrastructures play a key role in modern life, providing services such as telecommunications, electricity, gas and water within and between countries. In recent years, far reaching changes to these infrastructures has made them increasingly vulnerable to attacks, failures and accidents. The most significant transformation has been the deregulation of these industries. This may have increased their efficiency, but it has also increased the mobility of their workforces and made them less familiar with their working environment and arguably more likely to take revenge if cost-cutting leads to redundancies. Furthermore, new regulations, such the requirement to give precedence to sellers of renewable energy in the power generation and distribution industry (such as wind power with its spiky generation behavior) make control of the infrastructure and the co-ordination of the monitoring and diagnostic functions more difficult.

Adding to these problems is the increased interconnectivity and interdependence between critical infrastructures. Cascading failures can now propagate internationally, as happened when a failure in a Swiss transmission line caused a widespread blackout in Italy [1]. There is also a rising risk of cascading effects between infrastructures of

different types. Although the interdependency between telecommunications and electricity has been around for some time, now that electricity suppliers provide communication services, it is possible that the failure of a transmission line could impact telecommunications directly. The increased interconnectivity has also made malicious attacks much easier. In the past a telecommunications switching station or electricity plant could only be sabotaged through direct physical contact; now terrorists can attack critical infrastructures from anywhere the world.

Safeguard [2] is a system that aims to improve the dependability and survivability of large complex critical infrastructures by monitoring and protecting them with autonomous agents. At present the availability and integrity of critical infrastructures are usually monitored and maintained by human operators. Safeguard uses agent technology to improve the capabilities of the automatic control functions while also helping the human operators to make the right decisions at the right time.

This chapter will introduce some of the threats to critical infrastructures and the agent-based solution that we have developed to tackle these threats. Although the Safeguard project has focused on electricity and telecommunications networks as its practical examples, the aim is to produce a generic solution that can be adapted to other forms of large complex critical infrastructure.

2. Attacks, Failures and Accidents

In general, critical infrastructures can be divided into the following three layers:

1. **The physical layer**. This is the network of pipes, lines, cables, etc. that delivers the essential services. In the telecommunications domain the physical layer consists of

the routers, switches and copper and fiber-optic lines that carry the telecommunications data. In the electricity domain the physical layer is the network of transmission lines, transformers, breakers and generators that create and transport the electrical energy.

2. **The cyber layer**. This is the computers, networks and data gathering sensors that are used to monitor and control the physical layer. In the telecommunications domain, the cyber infrastructure is used to monitor and control the many routers and switches within the system. In electricity, the cyber infrastructure gathers information about power flows and breaker states and transmits the operators' control signals to the breakers and transformers. Although the cyber layer may share communications channels with the physical layer – in telecommunications, for example – the data that is transmitted within the cyber layer has a very different function from that within the physical layer.

3. **Human operations layer**. All the information gathered by the cyber layer is passed on to the human operators, who use it to manage the physical and cyber layers. The organizational processes that are in place for management, security enforcement, and recovery from failures are part of this layer.

In the past it has generally been the physical and human operations layers that have been the most vulnerable to attacks, failures and accidents. Accidents and failures in the physical layer have always been part of the daily running of the network and this layer has occasionally been subject to physical attacks as well. Within the human operations layer, operators inevitably make mistakes and they also have the specialized

tools and knowledge that are needed to carry out malicious actions. It has also always been possible for attackers to gain physical access to the control room or to manipulate operators by social engineering. None of these problems have gone away in recent years, but a number of factors have contributed to a dramatic rise in the vulnerability of the cyber-layer, which has been the main area of focus for the Safeguard project.

2.1 Telecommunications Vulnerabilities

In most European countries traditional circuit-switched telecommunication networks are being replaced by packet-based IP networks. It is therefore the manifestation of attacks, failures and accidents in IP networks that constitutes the most significant threat to telecommunications infrastructures. In recent years the dependability of IP infrastructures has decreased, due to the proliferation of worms, denial of service attacks, buffer overflow vulnerabilities and viruses. One study [3] estimates that the number of intrusion attempts over the entire Internet is in the order of 25 billion each day and increasing. Furthermore, many of these types of attack are getting more sophisticated and more automated, which means that the skills needed to launch them are reduced (see McHugh's discussion of this [4]). There are also the inevitable problems of hardware failures and software crashes. All of these problems are being exacerbated by the rapid deployment of hardware and software that has been designed primarily for other contexts (e.g. personal computing) within the patchwork that constitutes modern complex systems. This spread of commercial off-the-shelf hardware and software can also lead to a monoculture within which it is easy for malicious processes to spread.

To ensure that the system is delivering its essential services,[1] network administrators rely on large numbers of diagnosis and detection programs distributed across the cyber infrastructure. Some of these programs are part of intrusion detection systems (IDS), others monitor the general state of processes, processors, file systems, and so on. These devices are collectively referred to as INFOSEC devices. It might be thought that INFOSEC devices, such as intrusion detection systems and virus checkers, would be enough to protect modern systems, but the problem with these "protective" devices is the extremely large number of alarms that they generate. An initial study in a test network set up by Safeguard showed that even in the presence of few attacks and very little normal activity on the machines, there were over two million alarms generated in a 50-host network over a two week period.

Another problem encountered by the security managers of IP networks is the false alarm rate of some INFOSEC devices, such as intrusion detection systems. To see why even an apparently low false alarm rate is a significant problem consider a hypothetical situation where an INFOSEC device has a 1% false alarm rate and a 100% detection rate. During one day, 1 million benign events and 20 malicious events are presented to this system. The INFOSEC device will detect all 20 attacks, but also classify 10,000 of the benign events as attacks [6]. The operator has to investigate 10,020 alarms, of which only about 0.2% of them are true.

A third problem faced by telecommunications operators is that the systems they are monitoring are in a constant state of change. This makes it very hard to identify the normal behavior of the system and detect when it is going wrong. If one could fix the topology of the network and define the traffic patterns that will appear in it over time,

[1] See [5] for a definition of essential services.

there would be some hope of detecting deviations from this behavior. Unfortunately in the evolving world of telecommunication services this is an unrealistic assumption, and solutions for enhancing the survivability of such networks need to be able to adapt to a changing normality.

To reduce the vulnerabilities in a telecommunications management network, we therefore need to have defense mechanisms that operate in a meaningful way in the presence of component failures (processors and software crashes), overloads (data overloads, alarm overloads), and attacks. These defense mechanisms must also interact in an effective way with the security administrator and adapt to legitimate changes in the underlying network operations.

2.2 Electricity Vulnerabilities

The electricity cyber layer contains a number of control centers running workstations, energy management software (EMS) and databases over a local area network. These control centers interact with the Supervisory Control and Data Acquisition (SCADA) system that consists of a software interface and specialized hardware units (RTUs), which monitor sensors and interface with breakers and transformers (see Figure

1).

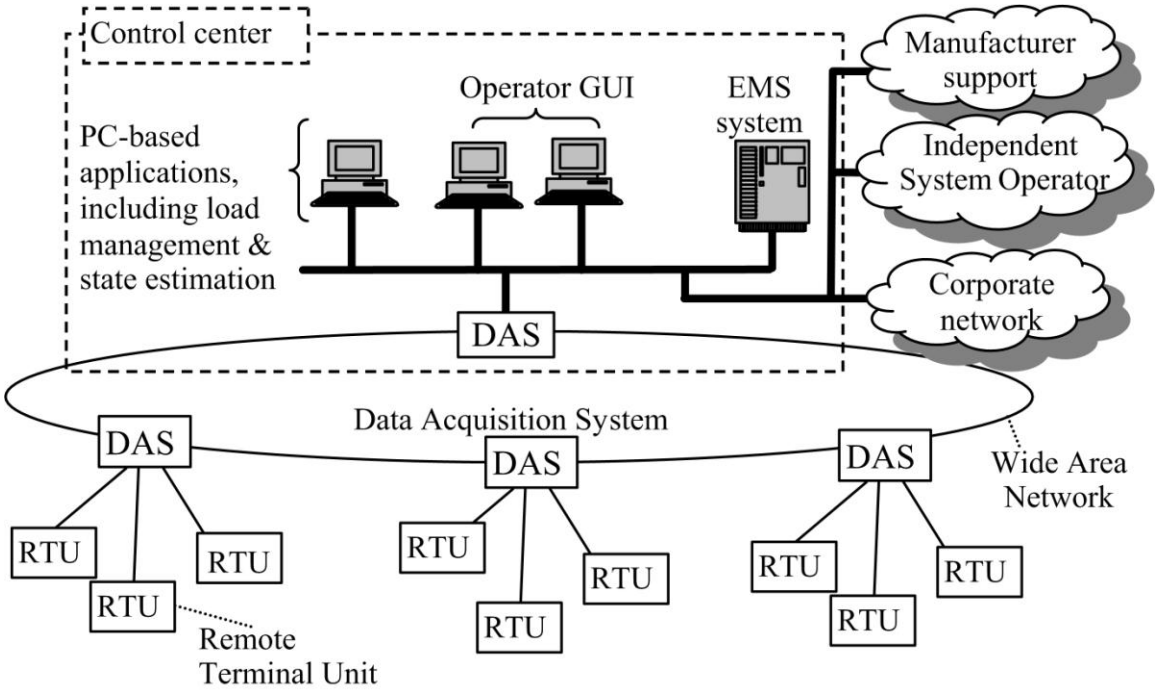

Figure 1. Electricity cyber infrastructure

Traditionally this cyber infrastructure was protected by its relative isolation and the non standard protocols that were used to communicate over it. However, with the deregulation of the electricity industry it no longer makes sense for each company to develop its own hardware and design proprietary protocols, and companies are increasingly looking to commercial products to solve their communication needs. A second consequence of this market orientation is the increased interconnectivity between the electricity management networks and other networks, most problematically between the corporate network and the control center network. The dangers of this standardization and interconnection became apparent in the recent Ohio nuclear incident [7] when the Slammer worm copied itself across from the corporate network into the plant network and disabled a safety monitoring system for nearly five hours, forcing the operators to

switch to an analogue backup. In a separate incident the propagation of Slammer blocked SCADA traffic and impeded the ability of operators to monitor and control the electricity system [8].

In addition to these problems that are linked to the increasing use of TCP/ IP in electricity management, SCADA systems are also subject to specific vulnerabilities of their own. One possibility is that an attacker could directly connect to intelligent electronic devices over a telephone line and open breakers or disrupt safety systems [9]. Another possibility is that a failure or attack could alter the operators' picture of the electricity network. This could lead them to take incorrect actions with potentially damaging consequences. This was one of the contributing factors to the recent U.S. blackout, when a bug in the state estimation software led to First Energy operators using outdated information about the status of the electricity network for over an hour [10].[2]

As these examples show, the key problem in the management of electricity networks today is the impact of attacks, failures and accidents on the monitoring and control of the electricity grid. What is needed is a system that can evaluate the reliability of the electricity data and carry out a rapid automatic response when necessary.

3. Solutions

Within the Safeguard project we have aimed to produce a solution that can tackle some of these challenges, using an open architecture that can be extended to cope with emerging threats. At the center of this Safeguard solution is an agent system, which facilitates the distributed gathering and filtering of information from a number of different sources and

[2] The state estimator is a program that runs in the electricity control center and uses the known electrical properties of the network to calculate a best-fit hypothesis about its current state.

the execution of a rapid response to attacks, failures and accidents. In this section we will describe the motivation behind the Safeguard approach. Section 4 will then cover the Safeguard architecture in more detail.

3.1 Agents

Agents are semi-autonomous software entities that perceive their local environment, process this information in some way and carry out actions. By communicating with one another they can develop collective solutions to problems. Agents run on agent middleware that separates their generic services (such as communication capability, service discovery, heart beat emissions) from the specific "business" logic used for detection, correlation or action. This paradigm offers a number of advantages over a single centralized intelligence:

Scalability. With a centralized system, communication bottlenecks can build up around the central controlling node as the network increases in size. Within a decentralized system the communication overhead depends upon the degree of coupling between the distributed components. However, since this overhead is distributed over the whole of the network, even quite tightly coupled distributed systems can generally be scaled more easily. For example, in the management of IP networks it has been found that although there may be a higher growth in signal traffic if flooding is employed to inform all other nodes of state change, this represents only a small percentage of link bandwidth [11]. The processing of information is also distributed across many nodes in agent systems.

Local detection and response. The remoteness of centralized intelligence systems makes them useless in the event of a partial or complete network failure. On the other hand, a distributed agent system can carry out an autonomous rapid response in isolated parts of the network, which can perform definitive repair actions or simply "buy time" before a more lasting remedy is found. This is a substantial advantage in critical infrastructures, which are generally managed using a central control room (with backup control available at different locations).[3]

Robustness. Agent systems ideally do not have a single point of failure and can continue to operate even if a number of them are attacked or fail. This is a useful feature in dependability applications, where graceful degradation rather than rapid collapse is the optimal behavior.

Emergent behavior. A lot of work has been done on the way in which certain forms of intelligence can emerge through large numbers of local interactions. A number of relatively simple agents can self-organize to perform tasks that would very difficult to create and coordinate from a central point.

Modularization. Whilst the object-oriented paradigm does facilitate a reasonable amount of modularization, agent systems take this even further since each agent operates as an independent functioning system. Different agents can be designed by different people to handle specific tasks and only the messages between them need to be agreed. This makes it is easy to set up communications between agent systems made by different companies for diverse critical infrastructures.

[3] This true even in mobile networks, where regional control centers manage traffic from many base stations.

These advantages of agent-based systems make them the best choice for the monitoring and protection of large critical infrastructures. However, it is the intelligence inside the agents that enables them to identify problems and react to them appropriately. The Safeguard approach to this will now be covered in the next two sections.

3.2 Detection

We have already outlined some of the INFOSEC devices that are used to gather data about the critical infrastructure that is being protected. For fault detection there are many expert-based diagnostic systems that classify symptom sets to identify causes. For security diagnosis virus checkers and intrusion detection systems apply knowledge-based techniques (typically signature-based misuse detection), or behavior-based techniques to identify malicious attacks on the system. There are also general tools that simply act as a probe and facilitate the subsequent use of an intrusion detection system, either by data mining log results, or by collecting alerts for further examination by a human operator [12].

Although all of this data is extremely useful for reaching conclusions about the system, the central problem is that most of these devices rely on signatures of the problems that are looking for and are unable to detect novel kinds of attack or failure. The only way in which they can manage this limitation is through frequent updates, which often arrive after a worm or virus has already spread. To address this limitation of signature-based detection there has been considerable research over the last five years into anomaly detectors that learn a model of the normal behavior of users or the protected system. This model of the normal behavior of the system is then compared with its actual

behavior to identify anomalies. Since no knowledge of attacks is needed to train the normality model, anomaly detection has the ability to detect previously unknown attacks. Typical examples of anomaly detection are the statistical profile-based anomaly detection in EMERALD [13] and Forrest's models of system calls inspired by the immune system [14]. The disadvantage of anomaly detection is that it can suffer from high rates of false alarms and it can be difficult to adapt a normal model to gradual changes in the system over time.

In Safeguard we combine knowledge-based and behavior-based detection into a hybrid detection model that leverages existing knowledge and also looks for significant deviations from normality in the system. This hybrid detection is carried out by agents tuned to the data types of the telecommunications and electricity domains. For example, one agent monitors IP packets, another looks at timing differences in SCADA signals and another examines the power readings from the electricity network. The modularity of the agent architecture also enables us to integrate new kinds of anomaly detector as the need arises.

The detection that has been described so far takes place at the lower levels of the system. The second phase in the identification of problems is the correlation of this information to bring it together into a single picture that can form the basis of a coordinated response. In conventional systems, very little correlation takes place and the operator has to sort through and integrate the large number of alarms. A key part of the Safeguard strategy is to take advantage of the distributed agent architecture to do this correlation for the operator.

3.3 Response

Distributed agent architectures are good at gathering and processing information. They are also an effective way to deliver a rapid local response to problems within the system. When worms like Slammer can infect most vulnerable machines within ten minutes of their release [15], an automatic response is essential because there is no time for operators to sift through thousands of alarms and halt the spread of the problem. The danger with autonomous action is that agents can create more problems than they solve. This risk can be reduced if the agents' autonomous actions are only implemented when it is judged to be more risky for them to delay than to act on their limited information. If the response is not time-critical, or if there is a danger that they could damage essential functions, it is safer to make the agents wait for approval from the human operator.

4. The Safeguard Architecture

Now that we have covered the motivation and general outline of the Safeguard approach we will describe the way in which the Safeguard agent system has been implemented within the telecommunications and electricity domains. The key to a generic solution that is applicable in many infrastructures is the definition of a number of different types of agent, each of which carries out a different role. These are common to the defense of many different infrastructures, but in some cases their functionality is optimized for a particular domain. The agent types and the communications between them are depicted in Figure 2.

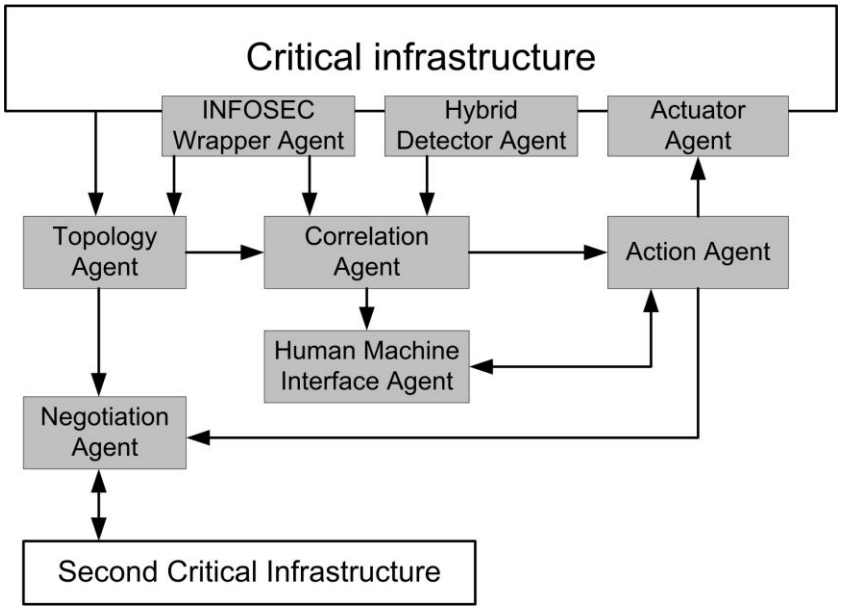

Figure 2. The Safeguard agent architecture

Wrapper agents. Interface with other applications running on the system, for example the intrusion detection system, firewall, virus checker and diagnostic software. Their task is to filter and normalize information from these applications and then pass it on to the correlation agents.

Topology agents. Gather dynamic network topology information, such as host type, operating system, services provided and known vulnerabilities.

Hybrid detector agents. Identify potential problems in the cyber infrastructure. These agents use domain specific knowledge about the behavior of the system and compare the system's current behavior with a learnt normal profile.

Correlation agents. Identify problems by bringing together information from the wrapper, topology and hybrid detector agents. The tasks of these agents include

ordering and filtering alarms to provide a focus for the operators and diagnosing specific problems in the network.

Action agents. Automatically or semi-automatically respond to problems in the network that have been identified by the correlation agents.

Negotiation agents. Communicate with agents in other critical infrastructures to request services and pass on information about major security alerts.

Human-Machine Interface agents. Provide an appropriate interface for one or many system operators. This allows the human operators to monitor the agent system, view the cyber infrastructure and approve the agents' actions.

Actuator agents. Wrapper that enables the action agents to interact with the lower layer of software and hardware (the firewall, for example).

All of these agents run on a platform, which provides generic services such as discovery and messaging. This is an implementation, developed by Aplicaciones en Informática Avanzada (AIA),[4] of the Java Agent Services (JAS) specification. This is an industry standard for the development of Java agent and service architectures.[5]

Within this architecture the functionality of agents that operate at a high level of abstraction from the critical infrastructure is generic to the electricity and telecommunications domains. Agents working closer to the cyber infrastructure have been designed to take into account the specific requirements of the domain that they are

[4] This is one of the Safeguard project partners.
[5] See [16] for a comparison between the Safeguard agent platform, JADE and Tryllian.

operating in. In the management systems of telecommunications networks there are upwards of 50 machines and very large quantities of alarms. One type of correlation agent in the telecommunications domain performs static and adaptive filtering to reduce the alarm rates. In the electricity domain there are problems with the corruption and loss of data. A hybrid detector agent has been built to detect anomalies in the electricity data and improve the operators' view of the network.

This section starts with a brief description of some of the agents that are common to both domains. It then covers the agents that have been developed to address specific problems within the telecommunications and electricity cyber infrastructures. Since the majority of the research effort within Safeguard has been dedicated to the development of the domain-specific agents, these will be covered in more detail.

4.1 Generic Agents

The functionalities of the agents in this section are generic to all critical infrastructures.

4.1.1 Topology Agent

The topology agent stores information about the cyber infrastructure and makes it available for the other agents running on the system. This information is gathered from databases and human operators and includes the hardware and software components, their interconnections, importance and vulnerabilities, the services they are running and the people responsible for them. This agent also provides a measure of health for the components that it is monitoring – a computer with a high CPU load and high temperature is rated as less healthy than one that has a low load and is cool running.

4.1.2 Human-Machine Interface Agent

The Safeguard agents' autonomous actions are carried out under the supervision of a human operator. In some situations the agents will need to ask human operators for permission to carry out actions. In other situations the agents will be unable to take action and alarms or requests for help need to be sent to the human operators. The human-machine interface agent manages all these forms of communication between Safeguard and the human operator. It displays the ranking of the filtered alarms that were processed by the correlation agent and enables the human operator to monitor and control the autonomous actions of the agents. The human-machine interface agent also displays the information gathered by the topology agent about the cyber infrastructure.

4.1.3 Negotiation Agent

The negotiation agent handles the interactions with other critical infrastructures. As was mentioned in Section 2, one of the emerging problems with critical infrastructures is their increasing interdependency and the negotiation agent's main task is to minimize the risk of cascading failures by keeping the operators and agents in other critical infrastructures informed about attacks, failures and accidents. The negotiation agent also exchanges information that can help in the day-to-day running of the networks.

In the event of a significant failure, the negotiation agent can also request services from other critical infrastructures of a similar or different type. For example, if a telecommunications network experiences a major failure, the negotiation agent can arrange for its calls to be switched through the most suitable alternative network at an appropriate price. If the data communications in an electricity network fail, more services can be requested from a telecommunications network.

4.2 Telecommunication Agents

This section describes some of the agents that were specifically designed to operate in the telecommunications domain.

4.2.1 Network Hybrid Anomaly Detection Agent

As explained in section 3.2, the Safeguard hybrid detector agents use a combination of knowledge-based and behavior-based techniques to identify problems in the cyber infrastructure. In this agent, the anomaly detecting part uses an algorithm called ADWICE (Anomaly Detection With fast Incremental Clustering), a new adaptive anomaly detection scheme built on BIRCH clustering [17]. This behavior-based data mining approach is combined with a knowledge-based white-list engine, resulting in a hybrid anomaly detector. Although this agent is designed primarily for detecting anomalies among network packets, its algorithms are general enough to be applied to any data provided that a small class is implemented to access the data source and normalize the data.

The white-list engine implements simple specification-based intrusion detection [18], where data known to be normal is described by manually constructed signatures. In our case, hosts and services that are known to produce abnormal behaviour (e.g. DNS server port 53) are filtered away, but rules for arbitrary features can be used. Data that is classified as normal by the white-list engine is not fed into ADWICE, which reduces the size of the normality model without decreasing detection coverage. Normality is represented by a large number of small possibly overlapping clusters, where each cluster is represented by a compact *cluster feature* (CF). A *cluster feature* is a triple (n, $\vec{S}$, SS),

where n is the number of data points in the cluster, $\vec{S}$ the linear sum of the n data points and SS the square sum of all data points. Given the cluster features for two clusters, the centroid v_0 and radius R may be computed. ADWICE incrementally constructs a cluster feature tree which facilitates fast searching of the normality model.

Once the cluster feature tree has been constructed it is used to identify anomalies in data from the network At detection time, the Euclidean distance between a new data point v_i and the centroid of the cluster CF_j closest to v_i is calculated. If this is greater than a threshold, then it is unlikely that the data point belongs to any of the learnt clusters and it is classified as anomalous. This anomaly model can be incrementally expanded and contracted to accommodate the dynamic nature of telecom/IP networks [17]. Furthermore, the agent can adapt to new network circumstances by dynamically changing the threshold that is used for detection purposes.

4.2.2 Wrapper Agent for Alarm Database

The main objective of this agent is to encode the rules that are currently used by expert operators to reduce the volume of reported alarms. These rules are applied in three stages of static filtering, dynamic filtering and normalization (Figure 3). Expert operators generally do this kind of filtering on information from several INFOSEC devices and so this agent was constructed to process information from one network-based intrusion detection system, Snort (a rule-based intrusion detector that inspects network packets), and two host-based systems, Samhain (a file integrity checker) and Syslog (system log messages).

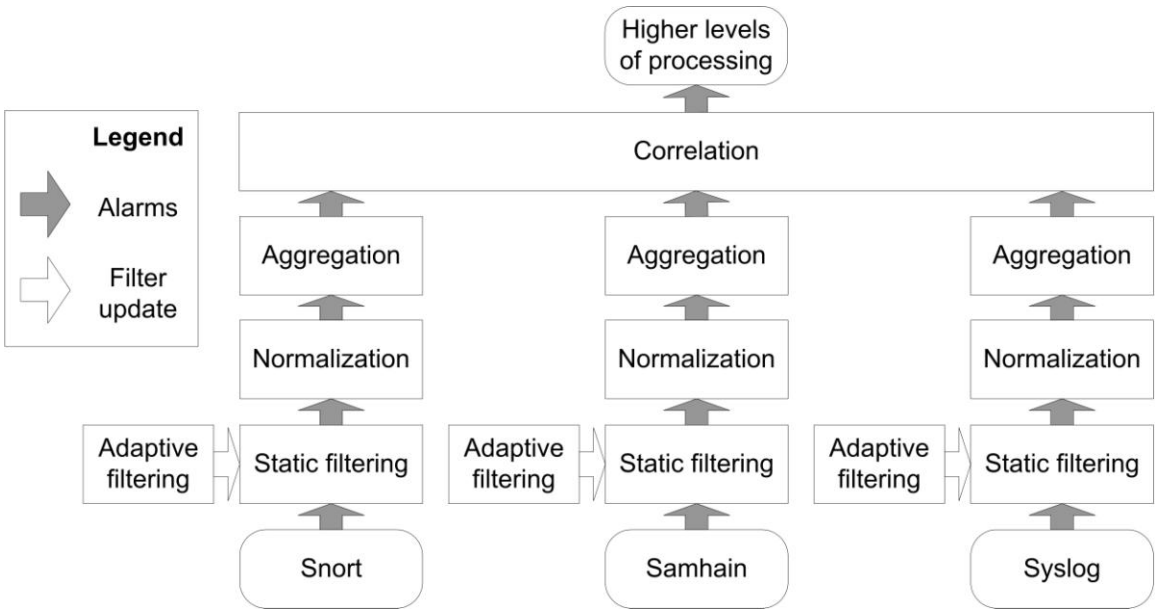

Figure 3. Functions in the bottom three rows are implemented in the wrapper agent for alarm database

The filtering and normalization that this agent applies to the alarm data will now be covered in more detail.

Filtering

With static filtering simple rules are set up to remove uninteresting or irrelevant messages. However, the problem with applying static filters to misconfiguration messages is that it is not possible to foresee all future misconfigurations and so adaptive filtering algorithms are needed to update some of the filters. For the Syslog alerts an automatic filter detector based on Naïve Bayesian learning [19] was designed. This trains a Naïve Bayesian text classifier to classify messages by looking at the words that appear in them. The Naïve Bayesian classifier is first trained by presenting a number of messages labeled as interesting or uninteresting. During training, a knowledge base is built up by counting the occurrences of the tokens (words) in the message. After training,

messages are classified by computing the probability that they belong to a class based on how common the values of their attributes (words) are in that class within the training data.

A more detailed workflow for the training of the adaptive filters is as follows: On a periodic basis, or on demand, the algorithm for adaptive filtering is launched. The human-machine interface agent then presents the algorithm's classification to a human expert. In order not to overload the expert, only the top scoring messages are selected (e.g. messages with a high count, which are the ones with the most influence). The human expert then classifies the messages as interesting or uninteresting. An interesting message is one that is a useful indicator of an attack. For example, a message saying "telnetd[1]: Successful login as user root" or "File changed" is classified as interesting, but messages like "Syslog-ng: 54 Objects alive. Garbage collecting while idle" or "Could not resolve host name" will be classified as uninteresting (the last message is an example of a misconfiguration that should be detected and reported separately from the intrusion detection process). The human expert's reply is then used to optimize the training corpus (thus achieving on-line retraining). To our knowledge this is the first work in which text classification is used for adaptive filtering in intrusion detection systems.

Normalization

Snort, Samhain and Syslog produce information and alarms in very different formats and to effectively correlate the information from these sources it is necessary to normalize the data in a standard way.

More detail on the above techniques and the evaluation of this agent on the telecommunications test network are presented in [20].

4.2.3 Alarm Reduction Correlation Agent

After filtering and normalization the output from Snort, Syslog and Samhain need to be correlated in order to identify the real attacks on the system. But before doing this we can reduce the number of alarms by correlating the output from each sensor within a time window. The next two subsections describe the combination of knowledge-based and behavioral-based techniques that we have employed to do this correlation.

Aggregation

Aggregation is used to integrate information that is of the same type or from the same source. For example, port scans generate a lot of messages in Snort and are not of primary interest. It reduces the information overload if each scan can be reduced to a single alarm (if possible, for example when the scan was performed from the same source). The source of a scan is also useful as an additional parameter when looking for other attacks, since it is known that network scans are often the first sign of an attack. Other alarms can also be aggregated if they are similar and occur within a certain time of each other. This kind of aggregation can be applied to all intrusion detection systems, but the definition of which alarms should be treated as similar is specific to each intrusion detection system. For example, Snort alarms with the same source IP, destination IP and signature are aggregated if they fall within a given time window.

Multi-sensor correlation

Next we deal with correlation of alarms from several sources. At present this correlation is carried out by a security administrator who analyzes the output of these data sources to find the real attacks. In Safeguard we wanted to automate this process, but the difficulty is that the way in which a security expert analyses the information is complicated and cannot be written down as a simple collection of rules. A more sophisticated machine learning algorithm was needed to separate the true and false positives. For the alarm reduction correlation agent, three different correlation methods were explored and compared to each other.

Before the alarms are correlated, they are integrated by taking all the alarms relating to one host during a period of time (i.e. the correlation parameters are time and IP address) and then plotting the sum of their severities against each other in three dimensions. Some typical alarms that are brought together in this way are illustrated in Figure 4.

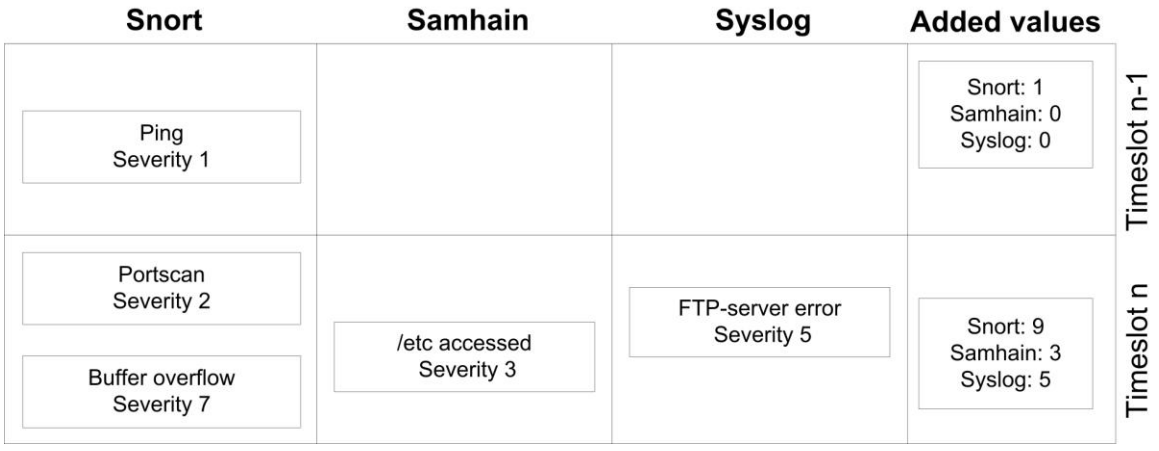

Figure 4. Alarms appearing in two time slots

The resulting three-dimensional vectors are then used as input to an additive correlator, a neural network and a K-nearest neighbor algorithm. The additive correlator

is the most simple of these and it simply adds the three severities and generates an alarm if the sum is larger than the threshold. By varying the threshold different detection and false positive rates can be achieved. This provides a simple means by which adaptiveness can be build into the detection and analysis mechanism. When the operator is extremely overloaded the threshold can be automatically adjusted to deliver fewer alarms (trading off accuracy with fewer false positives and buying time). The other two correlation algorithms are first trained with some of the time slots and then used to classify new time slots. More details about the other two correlation algorithms can be found in the documents [20] and [6], which also contain a comparative evaluation of these techniques. To summarize the findings, the additive correlator was found to have the simplest deployment (no training) and was more adaptable than the other two. It had a reasonable detection rate (78%) with a high (5%) false positive rate in the chosen configuration. The neural network correlator had the best detection rate (95%) combined with a low false positive rate (1.5%), whereas the K-NN correlator had a low detection rate (44%) but almost no false positives. The latter two were of course less adaptive to normality changes due to the need for retraining.

Further work on the correlation of alarms from different sources and the integration of the alarm and topology data is in progress and will be evaluated on our test network.

4.3 Electricity Agents

In this section we describe some of the agents that were specifically designed to operate in the electricity domain.

4.3.1 Electricity Data Hybrid Detector Agent

This agent monitors the data from the electricity network and looks for corruptions in it using a mixture of knowledge-based and anomaly-detecting methods. This agent also has the ability to improve the quality of the data by adjusting the weights on different electricity readings or, in an emergency, by suggesting values for some of the readings. A number of different methodologies are used:

Linear Equation Induction. This technique builds up a normal model of the data by looking for relationships between different data readings. In the data from electricity networks this approach is particularly effective since most of the data is interrelated in a systematic manner. For example, in the networks that we have experimented on, the relationship between the power flow readings at either end of a line are, to a high degree of accuracy, of the form P1 = kP2 + C, where k and C are constants.[6] During the training phase the data is examined for linear relationships by fitting an approximate model to the training data and computing its residual (e.g. least squares). During the monitoring phase, the data is checked to see if the equations still hold. If they do not, there could be data corruption or the manipulation of data by a malicious attacker.

Range checking. This builds up a model of the normal data by learning the normal range for each electricity reading. If the data shows a significant deviation from this range during the detection phase, there could be a problem with the data.

[6] The true relationship is not linear, but the approximation is good enough to detect anomalies.

Bus zero sum. It is a known property of electricity networks that the sum of real and reactive powers entering a bus equals the sum of the real and reactive powers leaving a bus. If this does not hold, there is likely to be a problem with the power readings on the bus.

Switch / power correlation. When a switch is open, the power readings on the line that the switch is linked to should be zero. If this is not the case, the information about the switch state could be wrong or the power reading could be incorrect.

All of this information about the electricity data is integrated within a dynamically generated Bayesian network constructed using the topology of the electricity network. This Bayesian network works out the probability that each reading is correct and the probability that the switches are open or closed. Figure 5 shows an extract from this Bayesian network showing the nodes related to a single line reading. The final network within this agent is an interconnected combination of a large number of these nodes.

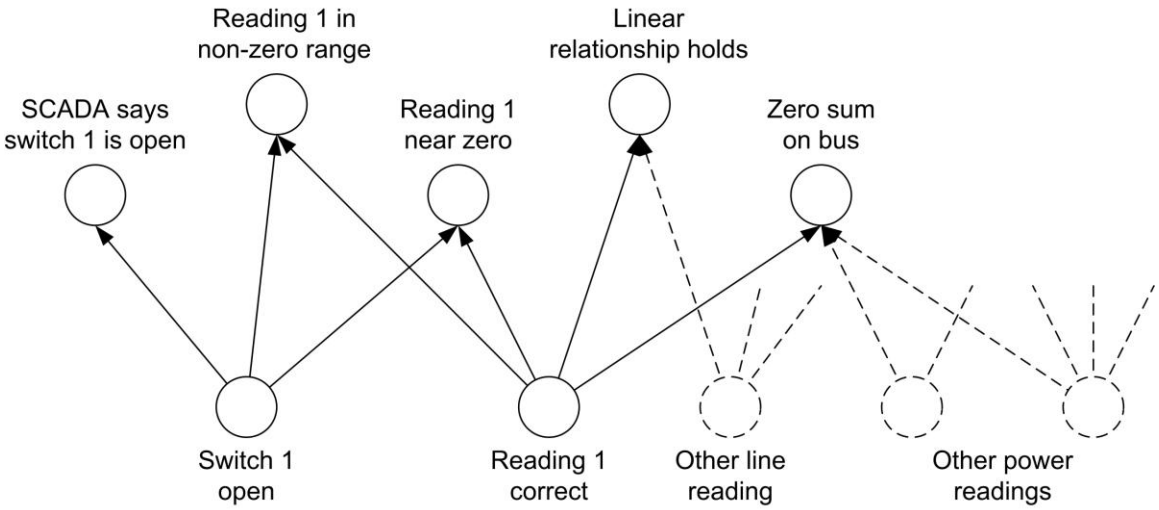

Figure 5. Bayesian network used to integrate information from different sources

More information about this agent and some preliminary results can be found in a recent paper [21].

4.3.2 Event Course Hybrid Detector Agent

This agent uses a case-based reasoning technique to recognize the most typical sequences of events coming from software modules inside the SCADA system. Initially it performs offline learning of the time constraints of event sequences. It then monitors the sequences online and associates them with an anomaly level. The sequences of events that are to be recognized are defined and edited inside the agent's case base by an expert on SCADA systems [22].

To apply this methodology within a specific knowledge domain, it is necessary to define the following:

- The *structure* of the individual cases. For each domain the case structures and working features must be specified.
- The *metrics* establishing the degree of similarity between cases.
- The *attributes* on which the learning mechanisms must to be applied to modify or include different cases in the case base.

These definitions will now be covered in detail for the application of case-based reasoning within SCADA systems.

Structure of the Generic Case

The generic case structure corresponds to a course of events captured inside the SCADA system. An example of a SCADA course of events is shown in Table 1.

Case Name: X	E_S	E_1	E_2	E_3	E_4	E_5	E_6	Total Time
Time Label	0	t_1,d_{t1}	t_2,d_{t2}	t_3,d_{t3}	t_4,d_{t4}	t_5,d_{t5}	t_6,d_{t6}	T

Table 1. Representation of a generic course of events case

In this case T is the total time for which the course has to be considered and E_S represents the starting event of the sequence, occurring at time 0. For every event following E_S in the event course there is a corresponding record of the time elapsed since the starting event (t_1, t_2,…t_6). However, in a real case each event's time is only an average value that can vary with the system's memory consumption and workload. For this reason, the maximum expected time deviations (d_{t1}, d_{t2},…,d_{t6}) are also given for every event.

Metrics Definitions and Similarity Concept

During the case-based reasoning process, the above case, stored inside the case base, must be compared with a *current case*, X, acquired from software instrumentation installed in the SCADA devices. This case has the structure shown in Table 2.

Current Case	E_S	*	*	E_1	E_2	*	*	*	E_3	*	E_4	E_5	*	E_6	*	*
Time Label	0	*	*	Tc_1	tc_2	*	*	*	tc_3	*	tc_4	tc_5	*	tc_6	*	*

Table 2. Current case X acquired from SCADA instrumentation

E_S-E_6 are the first occurrences of the events of the generic case, the "*" character represents other types of events or successive occurrences of the same types of events, and tc_1-tc_6 are the times of the events E_S-E_6. Now is possible to evaluate the *distance* or *similarity* between case X and the generic case using Equation 1.

$$S = \frac{\sum_{i=1}^{n} \frac{|t_i - tc_i|}{d_{ti}}}{n}$$

Equation 1. The distance or similarity between the current and generic case

In Equation 1, n is the number of events belonging to the case and S represents the *degree of similarity* normalized to 1. This means that S=0 is a situation in which two cases are completely similar, whereas S=1 is a situation in which there is a complete lack of similarity.

Case Base Training Phase

A *training phase* is also necessary to update the times of the generic cases. Although generic cases are initially defined by experts using an appropriate case editor, more specific values of the case attributes are added through training. During this training phase, the agent monitors the different event sequences for a time period which must be sufficient to obtain enough adequate data sets to calculate average time values for each event. One of the advantages of case-based systems is their ability to handle *incomplete sets* of cases. In other words, the case base does not have to cover all possible situations from the very beginning. The case base reasoning system can work with an incomplete case set, covering only a limited set of the most important events. With additional training this case base can be extended with new cases covering new situations and events.

4.3.3 Electricity Correlation Agent

The task of the correlation agent is to gather information from the other agents, come up with hypotheses about the state of the system and suggest an appropriate response. In the telecommunications domain most of the information is in the form of a large number of alarms relating to attacks on the IP infrastructure and the main challenge is to sort through these alarms and identify those which are most likely to indicate an attack. In the electricity domain, there is a lower volume of alarms and the behavior of the SCADA system, the output of the state estimator and the quality of the electricity data all have to be taken into account by the correlation process. The challenge with this kind of correlation in the electricity domain is the complexity of the system and the vast number of possible scenarios. It also happens that several independent events contribute to a single observed phenomena and there is the further problem that many situations are not visible all at once, but unfold gradually over time.

It is not feasible to try to address these challenges with a single expert system that can recognize any problem within the domain. Such systems are hard to create and difficult to test and maintain. Instead, we have developed a very modular probabilistic reasoning system that breaks the reasoning down into a number of ‘streams of thought’ or workflows, which model sequences of events within the cyber infrastructure. As the correlation agent runs it monitors messages from the other agents and uses Bayesian reasoning to determine whether one of these sequences of events is taking place. When this is the case, it activates the matching workflow, which can then be used to carry out further diagnosis, request actions, trigger an alarm, or send messages to other agents.

Inside the correlation agent the modeling and tracking of workflows is handled using Petri nets. The transitions in these Petri nets are associated with nodes in a Bayesian network that carry out the probabilistic reasoning about whether an event is taking place. The observables in the Bayesian network are set using information from the other agents' messages and the Petri net transitions become fireable when the Bayesian node associated with them has a probability over a certain threshold. When this happens, the workflow is activated and its temporal reasoning is brought into operation. The next transition or set of transitions in the workflow are anticipated by the agent and the transitions prior to the firing transition set the temporal context for the event represented by the firing transition.

This modular reasoning process will be made clearer through an example. Suppose that a worm enters the electricity control center's local area network and starts to scan and copy itself onto other machines. The Safeguard agents will detect some of the consequences – network congestion, scanning, file modifications and so on - and pass this information on to the correlation agent. When the correlation agent receives this information it will use it to set the appropriate nodes in the Bayesian networks B1 and B3, shown in Figure 6.

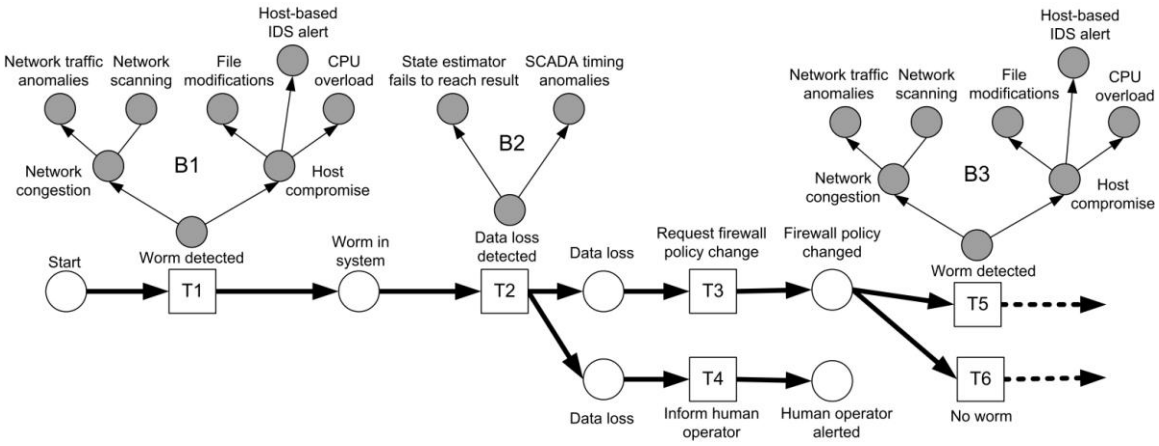

Figure 6. Simplified extract from a Bayesian workflow that is used to identify and respond to worms on the system. The clear squares are the Petri net transitions, the clear circles are Petri net places and the hatched circles are Bayesian network nodes.

If the Bayesian reasoning in network B1 concludes that there is a high probability of a worm on the system, then the correlation agent activates this workflow and fires transition T1. This workflow predicts that the worm attack will lead to the loss of SCADA data. When this hypothesis has been confirmed by Bayesian network B2, transition T2 fires and the workflow attempts to take action to deal with the worm. One action is to send a message to the action agent requesting it to block the port that the worm is propagating on (transition T3). The workflow also alerts a human operator about the problem (transition T4). The workflow then checks to see if the worm is still propagating on the system using Bayesian network B3, which in this case is the same as B1. If the worm is still present, transition T5 will fire and some other actions may be carried out. Otherwise transition T6 will fire and the agent may perform some clean up operations before closing the workflow.

A large number of these Bayesian workflows can be active at the same time. They are very lightweight because they are implemented using the workflow engine, Bossa [23], rather than separate threads. This allows the agent to deal with an indefinite number of different workflows relating to different event sequences in the monitored system. It is also very easy to extend the reasoning of the correlation agent and adapt it to new situations by adding new Bayesian workflows.

5. Future Work

The Safeguard agent system has been built and is currently in the process of being tested. In the telecommunications domain it is being tested on a realistic IP network of around 50 hosts with various hardware, switches, operating systems and levels of vulnerability. In the electricity domain, we are using a simulator developed by Aplicaciones en Informática Avanzada (AIA) to generate electricity data. A software emulation of a SCADA system is then used to transport this data across a real network to a number of machines connected together at the control centre. This arrangement allows us to explore electricity specific scenarios as well as the effects of an IP attack (a worm, for example) on the transportation of SCADA data.

The Safeguard project ends in May 2004 and in the longer term we are looking to develop our prototype into a commercial system through collaboration with industrial partners. We will also be looking at the application of Safeguard within other domains, such as the critical infrastructures that are used to transport water and gas.

6. Conclusion

This chapter has outlined the emerging challenges to the large complex infrastructures that are used to provide critical services, such as gas, water, electricity and telecommunications. Whilst the physical and human layers of these infrastructures have always been vulnerable to some extent, the deregulation and interconnection of these industries, combined with the move towards commercial off-the-shelf software, has increased the vulnerability of their cyber infrastructures and significantly raised the probability of uncontrollable cascading failures.

Safeguard is an agent system that is designed to address these threats to the cyber infrastructure. In its first implementation it has focused on problems within the telecommunications and electricity domains. The Safeguard architecture is composed of a number of different types of agent, each with their own particular role. There are agents that detect anomalies in the cyber infrastructure, agents that filter and merge alarms, agents that diagnose problems and agents that carry out a healing response. Some of the functionality within the agents is identical across different domains, whilst other functionalities have had to be adapted to match the particular requirements of each domain.

Many of the individual agents have already undergone a substantial amount of testing. References to papers have been given where this is the case. At the time of writing the Safeguard system is undergoing testing as a whole. The next phase of the Safeguard project will be commercial exploitation and the application of its methodologies within other domains.

Acknowledgements

We would like to thank the other members of the Safeguard project for their fruitful cooperation. These include Wes Carter (QMUL), Xuan Jin (QMUL), Julian Rodaway (QMUL), Stefan Burschka (Swisscom), Michael Semling (Swisscom), Tomas Dagonnier (Swisscom), Giordano Vicoli (ENEA), Sandro Bologna (ENEA), Luisa Lavalle (ENEA), David Moyano (AIA), Carlos López Ullod (AIA) and Oleg Morajko (AIA). We would also like to thank the European IST Programme (IST Project Number: IST-2001-32685) for their support for this project.

References

[1] GRTN report on the 28th September 2003 Italian blackout available at: http://www.grtn.it/eng/documentinewsstatiche/BLACKOUT28SET03.PDF.

[2] Safeguard website: www.ist-safeguard.org.

[3] V. Yegneswaran, P. Barford and J. Ullrich, "Internet Intrusions: Global Characteristics and Prevalence". *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, June 2003, pp. 138-147.

[4] J. McHugh, "Intrusion and Intrusion Detection", *International Journal of Information Security*, Vol. **1**, No 1, Springer Verlag, August 2001, pp. 14 – 35.

[5] K. Burbeck, S. G. Andres, S. Nadjm-Tehrani, M. Semling and T. Dagonnier, "Time as a Metric for Defense in Survivable Networks", *Proceedings of the Work in Progress session of 24th IEEE Real-Time Systems Symposium (RTSS 2003)*, December 2003, available at: http://www.ida.liu.se/~rtslab/publications/2003/Burbeck03swarmWIP.pdf.

[6] T. Chyssler, *Reducing False Alarm Rates in Intrusion Detection Systems*, Masters Thesis, LITH-IDA-EX--03/067—SE, Linköping University, November 2003.

[7] K. Poulsen, "Slammer worm crashed Ohio nuke plant network", SecurityFocus News: http://www.securityfocus.com/news/6767.

[8] North American Electricity Reliability Council, “SQL Slammer Worm: Lessons Learned for Consideration by the Electricity Sector Slammer”, available at: http://www.esisac.com/publicdocs/SQL_Slammer_2003.pdf.

[9] P. Oman, E. Schweitzer and J. Roberts, “Safeguarding IEDs, Substations, and SCADA Systems Against Electronic Intrusions”, available at: http://tesla.selinc.com/techpprs.htm.

[10] Kevin Poulsen, “Software Bug Contributed to Blackout”, SecurityFocus News: http://www.securityfocus.com/news/8016.

[11] G. Astrolopoulos, R. Guerin and S. Kamat, “Implementation and Performance Measurements of QoS Routing Extensions to OSPF”, *Proceedings INFOCOMM’99*, New York 1999.

[12] S. Managanaris, M. Christensen, D. Zerkle and K. Hermiz, “A Data Mining Analysis of RTID Alarms”, *Computer Networks*, **34**(4), Elsevier publishers, October 2000.

[13] P. A. Porras and P. G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances", National Information Systems Security Conference, October, 1997, available at: http://www.sdl.sri.com/papers/e/m/emerald-niss97/emerald-niss97.html.

[14] S. Forrest, S. Hofmeyr, A. Somayaji and T. Longstaff, ‘A sense of self for UNIX processes’, *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*, IEEE Press, 1996.

[15] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford and Nicholas Weaver, “The Spread of the Sapphire/Slammer Worm”, available at: http://www.caida.org/outreach/papers/2003/sapphire/sapphire.htm.

[16] K. Burbeck, D. Garpe and S. Nadjm-Tehrani, "Scale-up and Performance Studies of Three Agent Platforms", *Proceedings of International Performance, Communication and Computing Conference, IPCCC 2004, Middleware Performance Workshop, April 2004*, available at: www.ist-safeguard.org.

[17] K. Burbeck and S. Nadjm-Tehrani, “ADWICE – Anomaly Detection with Fast Incremental Clustering”, Technical Report, February 2004, available at: http://www.ist-safeguard.org.

[18] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang and S. Zhou, "Specification Based Anomaly Detection: A New Approach for Detecting Network Intrusions", presented at ACM Conference on Computer and Communications Security, 2002.

[19] D. Heckerman, “A Tutorial on Learning With Bayesian Networks”, Technical Report MSR-TR-95-06, Microsoft Research. March 1995 (Revised November 1996).

[20] T. Chyssler, S. Nadjm-Tehrani, S. Burschka and K. Burbeck, “Alarm Reduction and Correlation in Defense of IP Networks”, Technical Report, December 2003, available at: http://www.ist-safeguard.org.

[21] J. Bigham, D. A. O. Gamez and N. Lu, “Safeguarding SCADA Systems with Anomaly Detection", *Proceedings of the Second International Workshop on*

Mathematical Methods, Models and Architectures for Computer Network Security MMM-ACNS 2003, Lecture Notes in Computer Science, Vol. **2776**, Springer Verlag, 2003, pp.171-182.

[22] C. Balducelli, L. Lavalle, G. Vicoli, "Novelty detection and management to safeguard information intensive Critical Infrastructures", *11th Annual Conference of The International Emergency Management Society*, Melbourne, Australia, May 18-21, 2004.

[23] Bossa workflow system: http://www.bigbross.com/bossa/.