Securing Critical Infrastructures, Grenoble, October 2004

SAFEGUARDING ELECTRICITY CYBER-INFRASTRUCTURES AGAINST THE WORM THREAT

J. Bigham (*), D. A. O Gamez (+), Xuan Jin (*), Julian Rodaway (*), Chris Phillips (*) and Leonid Titkov (*)

(*) Department of Electronic Engineering, Queen Mary, University of London, London, UK

(+) Department of Computer Science, University of Essex, Colchester, UK.

1. Introduction

Large Complex Critical Infrastructures (LCCIs) 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 has been made on these infrastructures such as increased dependence on IP networks, increased use of commercial off the shelf software such as Windows, increased import and export of electrical power and increased use of renewable energy supplies. These infrastructures are now highly interconnected and interdependent. This has made them more vulnerable to attacks, failures and accidents. Now a major security incident also has the potential to affect the management network of critical infrastructures, potentially disabling them so that the operators lose control of the network.

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 centre network. The dangers of this standardization and interconnection became apparent in the recent Ohio nuclear incident 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. The Welchia (or Nachi) worm even infected check-in system of Air Canada and the US Navy and Marine Corps computers. [1]

Safeguard is a system that aims to enhance the dependability and survivability of LCCIs by protecting it against threats such as worms within the control system [2]. At present the availability and integrity of critical infrastructures are usually mainly monitored and maintained by human operators. Intrusion detection software has already been deployed in many LCCIs to help human operators monitor the system. However currently these software generate too many false positive and false negative alerts. Human operators are often overwhelmed when bursts of alerts arrive or misled by the wrong alert reports. More seriously, cascading alerts and failures can be aggravated when the operator cannot make decisions and act promptly. Safeguard uses agents to monitor and protect LCCIs by improving the capabilities of the automatic control functions and also helping human operators to make the right decisions and the right time. There are different kinds of agents. Some are used to detect anomalies within the system; others used to wrap existing software so that they can interact with the other agents; others to correlate the information from many agents into a diagnosis (or identification of the state of the system) and initiate automatic responses through action agents and actuators. Correlation is an effective solution that combines distributed detection and response with integration of critical information from many sources. In Safeguard, the objective of the correlation agent is to make sense of diverse pieces of information and perform timely action. It correlates alerts in real-time from multiple heterogeneous detection systems.

This paper will describe the ability of Safeguard to detect and respond to the impact of worms in the electricity control centre LAN. It will start with a brief review of the electricity cyber infrastructure. Then a review of the main agents in the Safeguard system is given. Then an outline of the worm emulation that we built to test worm propagation under controlled conditions is given. Finally the paper concludes with some experimental results. Although this paper focuses on the electricity control centre the same methodology applies to protecting the control networks of other critical infrastructures.

2. Electricity Cyber Infrastructure

The electricity cyber layer contains a number of control centres running workstations, energy management software (EMS) and databases over a local area network. These control centres 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). The RTUs are connected with the control centre via a wide area network.

Worms can affect both but this paper will look at the impact of worms on the electricity data in the electricity control centre.

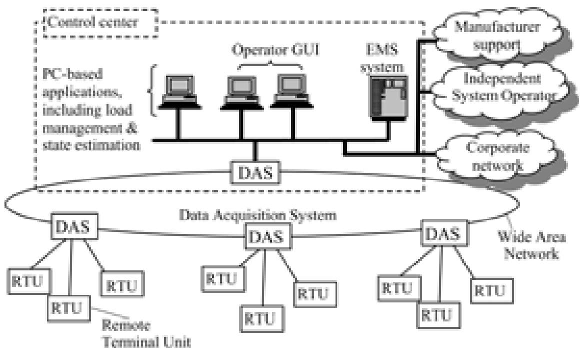

Figure 1. *Electricity cyber-infrastructure*

3. Safeguard Agent System

3.1 Overview

The Safeguard agent system is implemented as a hierarchically layered agent system [10]. It combines distributed detection and distributed response with integration of critical information from many sources. The Safeguard agent system has hybrid detector agents that monitor the network, operators, system components (machines etc) and data passed around the system and system malfunction detectors within an infrastructure in order to assess the state of the system and determine if it contains erroneous data or is under attack. Problems within the system, such as anomalous data or file integrity violations, will be identified. This information can be either passed to the operator or automatically acted upon, in order to prevent or limit inappropriate behaviour. The most important agents will now be covered in more detail.

The architecture of the agent system monitoring the system of figure 1 is given in figure 2. Different hybrid detector agents are positioned in the system based on the type of the activity they are monitoring. Information from these is passed on to the correlation agent, which makes an assessment of the state of the system components. If there is a problem, the correlation agent may use the actuators to take action, if there is time with the authorisation of the administrator operating through the man machine interface agent, otherwise if a time-critical response is required as in the case of a worm attack, then correlation agent may proceed automatically.

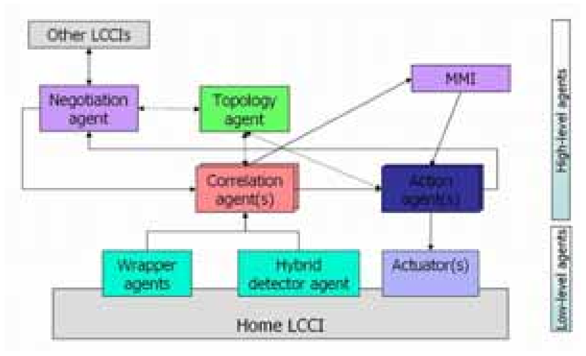

Figure 2. *The agent architecture*

There is not space to cover full Safeguard architecture in detail. This can be found in [13]. This paper will only give a brief description of the agents that are involved in the protection of the electricity control centre and were used to protect against worm attacks.

3.2 Hybrid Detector Agents

The Hybrid detector agents (HDAs) are effectively sensors that are used to gather information about diverse aspects of the system. Typically, their role does not exceed passive monitoring, although some may perform certain actions on the managed system, but only if explicitly permitted by the action agent. HDAs combine known information with a dynamic model of the systems normal behaviour. Because many of the HDAs in Safeguard are based on constructing different models of normality or defining invariants or approximate invariants in the system, they are capable of detecting anomalies that have not occurred before.

A large number of different types of dedicated agents are placed in the system to monitor many aspects of system activity, such as electricity data, network traffic and so on. An important one from point of view of this paper is the network traffic anomaly detector, which monitors the average level of network traffic entering and exiting the machine.

3.3 Wrapper Agents

The Wrapper Agents are attached to the existing intrusion detection systems that gather information about the system and possible attacks on the system. Wrapper agents simply allow information from existing diagnostic and IDS components to be integrated within the Safeguard system. Information from wrapper agents is sent to the correlation agents. An example is file integrity checker wrapper agent, which monitors integrity violations on critical system files. To detect known worms we wrap Prelude, an open source IDS which uses the SNORT signatures to detect known attacks on the system [14]. To detect unknown worms it is assumed that worms are likely to make changes to system files in order to ensure that they start up. So also wrapped is AFICK, a file integrity checker [15].

3.4 Workflow Correlation Agents

Workflow correlation agents are used to integrate information from the IDS wrappers and the HDAs into a diagnosis about what is happening on the system. The Workflow Correlation agents (WCAs) contain an embedded workflow management system named Bossa [8]. Predefined workflow models for the managed network are loaded to the workflow management system. Transitions of these workflow models are associated with predefined Bayesian network models. Workflow correlation agents are responsible for integrating information from the different HDAs or wrapper agents and reasoning about the state of the network system that is being protected and the behaviour of operators. Some of these transitions are used to model actions or to communicate with a separate action actuator agent. In this way the correlation and action actuator agents work together to provide a quick response that rectifies problems as they arise. An example of available responses includes killing worm processes when a worm is reported by the WCAs to stop the propagation of the worm in the network. A fuller description of the technology inside this agent is given in Section 4.

3.5 Actuators

Actuators are used to carry out actions on the network. In the context of the worm example, if the worm was known then the process name that it created was killed. Otherwise the most recently created processes were killed. This is not an optimal solution, but it is arguably better to have some incorrect processes occasionally killed accidentally, rather than have the whole control network paralysed by a worm.

3.6 Man Machine Interface Agent

The Man Machine Interface (MMI) agent is used to manage the Safeguard agents and define the scope of their legitimate activity.

4. Bayesian Workflow Correlation

4.1 Overview

Correlation uses workflows that are augmented by Bayesian networks. These combine probabilistic reasoning in order to make decisions at points in time with the management of the temporal evolution of events and resource checking of workflows.

4.2 Workflows

Workflows are defined by the Workflow Management Coalition as follows: The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [3]. They have been used in business for a number of years to model the flow of information within an organisation and the operations carried out on that information. These applications are oriented towards managing a complex sequence of activities. W.M.P. van der Aalst [4] has described Petri Nets as a tool for modelling workflows and describes modelling with sequential activities, parallel activities, AND splits, OR splits (implicit and explicit), different joins and iteration. van der Aalst’s paper also describes standard ways that transitions in the workflow are triggered: Automatic, User, Message, and Time. In our research, we follow this terminology and notation.

4.3 Workflow Management System

Workflow management systems are used to define, manage and execute workflows using software whose order of execution is driven by a computer representation of the workflow logic [5]. There are many commercial and non-commercial workflow management tools available such as Cosa [6] and OpenWFE [7]. Each application provides different functionalities and serves different users. In our research we chose the Bossa Workflow System as our workflow management system because it has the following advantages:

1) Bossa uses augmented Petri Nets to provide an intuitive way of modelling workflows and a way to verify workflow correctness. Extended Petri Nets even allow users to model time and include a hierarchy of workflow models.

2) Bossa is designed to be embedded and it is easy to define and dynamically load workflows in Bossa.

3) Bossa is written in Java, which can be platform independent. Also it is relatively easy to integrate Bossa with other Java code linked to workflow functions

4) Bossa is lightweight and fast. One reason is because tracking of position in a workflow is implemented without using separate threads for each workflow. This allows the agent to deal with a large number of different workflows relating to different event sequences in the monitored system.

4.4 Constructing Workflows

Basic Petri Net modelled workflows for Bossa workflow management system have the following elements: Places, Transitions, and weighted Arcs. Workflows in our system can only be started at one point. This point should be marked by placing a single token at that point. Four different types of transition can be used to construct a workflow. The transition type is set using the first few letters of its name.

1) Bayesian controlled transitions. These transitions are fired when the probability of the node in the Bayesian network identified as corresponding to the transition exceeds the prescribed threshold.

2) Message sending transitions. When a workflow reaches a message sending transition a message is sent to another agent. For example, a message sending transition can send a message to the MMI agent with the subject *WorkflowStarted*. The attributes of the workflow will also be sent to the receiving agent when the transition fires. It is also possible for the agent to send a message to itself. This feature can be

used to start a workflow or set an observable in another workflow.

3) Timer transitions. A timer transition either sets a timer or checks to see if the timer has expired. Timer transition can be used to execute periodic tasks.

4) Ordinary transitions. An ordinary transition is used to route the workflow based on static or dynamic attributes. These transitions are always fireable when they are reached by the workflow.

A simple Petri Net modelled workflow using our interface to the Bossa workflow engine is shown as figure 3. Transition T1 has an explicit OR-split, meaning that only one of the paths is traversed. Variable a is global to the particular workflow but invisible to other workflows. When transition T1 is fired, according the evaluation results of the post-conditions of T1, the token in place P0 will be transferred to P1 or P2.

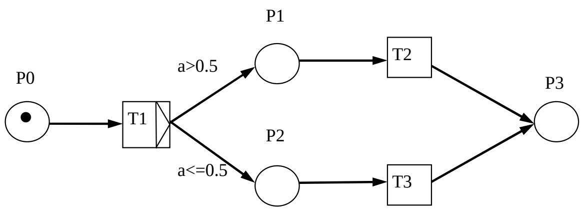

Figure 3. *Simple Petri Net modelled workflow*

4.5 Constructing Bayesian Networks

Bayesian networks are constructed in an XML format using the graphical capabilities of the JavaBayes software [9]. The EBayes software [11] is embedded in the correlation agent and this is used when updating the beliefs of nodes in the Bayesian networks that are linked to transitions in the workflow. A Bayesian network provides a link between incoming messages from anomaly detectors and a transition in the workflow. Each Bayesian network contains one or more observable nodes which corresponding to incoming messages. Observables are linked by the Bayesian network to nodes associated with workflow transitions. The belief in the Bayesian network node linked with this associated Petri net transition node is queried to see if the transition should fire at runtime. Deterministic relationships between observables and transitions are handled as a degenerate case where the probabilities are 0 and 1. Observable nodes do not depend upon any other Bayesian network nodes as they are set directly by other agents of Safeguard system.

4.6 Bayesian Workflow Correlation

In our research we are applying workflows to model and monitor both normal and abnormal flow of activities within an organisation. It is understood that many of the workflows are not already in place. Messages from the different anomaly detectors are pushed to the correlation agent. Each message has a value of true or false, which will be used to set observable nodes in the Bayesian networks in the appropriate workflows. (This could be generalized so that probabilities could be passed, but it is not so in the current implementation.) When repeated messages are sent the latest incoming message will update the corresponding observable node to it new value. These messages are stored by the correlation agent so that it can retrieve other attributes of the message when a transition fires. By doing so we are able to correlate different reports and alerts from different intrusion detection agents at different times and stages of evolution of an attack.

There can be hundreds of predefined workflow types existing in a single correlation agent and for each workflow type there could be many instances of this workflow running simultaneously. The Bossa workflow management system handles the concurrency and parallel execution.

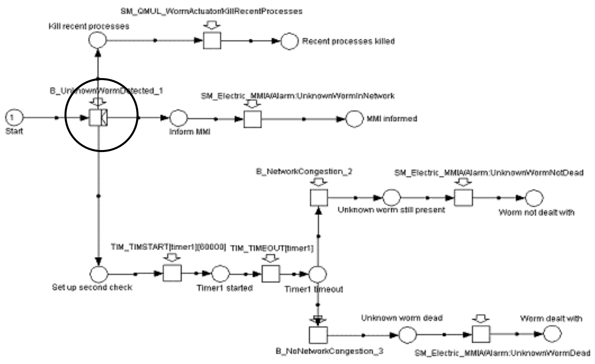

Figure 4. *Workflow for detecting and responding to worms. The Unknown Worm Detected transition is circled.*

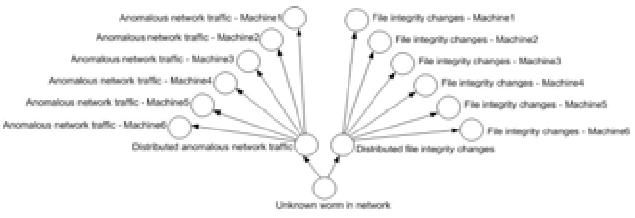
Anomalous network traffic - Machine1
Anomalous network traffic - Machine2
Anomalous network traffic - Machine3
Anomalous network traffic - Machine4
Anomalous network traffic - Machine5
Anomalous network traffic - Machine6
Distributed anomalous network traffic
File integrity changes - Machine1
File integrity changes - Machine2
File integrity changes - Machine3
File integrity changes - Machine4
File integrity changes - Machine5
File integrity changes - Machine6
Distributed file integrity changes
Unknown worm in network

Figure 5*. Bayesian network that controls the transition UnknownWormDetected*

This modular reasoning process will be made clearer through an example. Suppose that a worm enters the electricity control centre’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, here shown in figure 5. The structure of such a network is in general a directed acyclic graph, though the one shown is simply a tree.

Figure 4 shows a 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. If the Bayesian reasoning in figure 5 concludes that there is a good probability of that *Unknown Worm Detected* is true, then the correlation agent activates this workflow and fires the circled transition in figure 4. This workflow then kills some recent processes, alerts a human operator about the problem and also sets up a time delay so that it can wait before checking if the worm is still propagating on the system using another Bayesian network.

5 Experiments

5.1 Overview

In this scenario we wanted to test the ability of Safeguard to handle a scenario in which a worm enters the electricity control centres local area network and starts to scan and copy itself onto other machines. The Safeguard hybrid detector agents and wrapper agents detect some of the consequences, such as network congestion, scanning, and file modifications. This information is passed to the correlation agent. We wanted to test both known worms where there is an IDS signature and unknown worms.

5.2 Worm emulation

One way of testing the worm scenario would have been to capture a number of worms, such as Code Red, Slammer and so on using a honey pot, and then release them within our test network. However, there are a number of problems with this way of testing:

Real worms exploit very specific vulnerabilities (patches, versions etc.). This makes running a number of different worms on the same machines potentially very complicated.

Real worms can do considerable damage to the machines that they are copying themselves onto. They can also be difficult to clean up after each test, especially if they target the anti virus software.

Depending on their scanning algorithm, real worms can propagate very slowly on their local network, even if they propagate rapidly on a global scale. This makes studying their impact within a network difficult.

The properties of real worms (packet size, number of threads, port, and so on) cannot be changed, which makes it hard to come up with general conclusions about worm behaviour.

To avoid these problems it was decided to create a worm emulation to test this scenario. This was created in Java to enable it to run on any machine. This emulation consisted of a malicious process that scanned machines or sent a replication packet or both and also a vulnerable process that listened on a specific port for the infection packet and started up a new malicious process when the replication packet was received.

Since the exact properties of each worm are different, the worm emulation software was not written to fully emulate any of the known worms. Instead it was decided to create general worm-like processes that can imitate the properties of a particular worm family. The distinction between worm families was drawn based on type of protocols used, replication strategies, level of network congestion, and CPU time consumed. The two worm families that were chosen for this test scenario were Code Red and Slammer. Code Red propagates using ICMP to detect machines and then TCP to propagate between machines. This method of propagation necessitates a large number of threads. Slammer, on the other hand, simply generates a single UDP packet and sends this to a randomly chosen host. Dependant on the protocol used the malicious code can be either embedded in the UDP datagram or TCP message. The properties of these two worm families are summarised in the table below.

Worm Family	**Protocol**	**Number of threads**	**Impact**
Code Red	ICMP (machine detection) TCP (propagation)	Large (99)	CPU & Network
Slammer	UDP	Small (1)	Network

Table 1 *Properties of the two worm families chosen*

The Worm Emulation Software behaves as an infected machine and for the Code Red family emulation it uses the ICMP protocol to scan hosts on the local network as well on the global network for valid IP addresses, i.e. ones that actually exists. If it does not receive a reply to the ICMP message the address does not exist. The worm then tries to replicate itself on every valid IP address.

More specifically, the worm emulation software has between 1 and 99 threads running, which are specifiable as a command line configurable argument, each one sending ICMP type one requests (PING) to a randomly generated IP address and listening for a reply. The more threads the more virulent the worm. A successful reply means the machine exists, whereas an ICMP type three message (destination is unreachable) means that this address is unavailable. The global network scan uses randomly generated IP addresses, whereas the local network scan takes into consideration the address of the host machine and the subnet mask. In case of a valid IP address, the Code Red emulation tries to replicate on the other host using the TCP protocol. Since only one out of eight ICMP requests corresponds to a valid IP address (this is due to the probability of generating an IP address correctly in the procedure used) a machine infected with this type of malicious code can be detected by the number on ICMP type three messages sent to it.

To replicate the worm needs to find vulnerabilities, e.g a service with known vulnerability running on an open port of the potential victim machine. After exploiting the vulnerability the emulation establishes a connection between the worm process and the vulnerable service process. It will then try to replicate itself by sending a replication message which is a sequence of one packet messages. One of these packets contains malicious code which initializes the replication procedure on the recipient machine. Embedded malicious code has a particular signature which is used by the intrusion detection system to identify the malicious activity on the network and take the appropriate actions.

For the Slammer family emulation, the Worm Emulation Software uses UDP for its propagation and sends a single replication packet to a randomly chosen IP address. To control the propagation rate, the range from which this random IP address is set using a command line argument.

The impact of the worm emulation upon the electricity network and the Safeguard response were measured by using Network Probe to monitor the total network traffic. The worm emulation was initially run without Safeguard and then with Safeguard so that the effectiveness of the Safeguard response could be evaluated. In the Safeguard tests, a String was included inside the worm and a signature written for Prelude to test the ability of Safeguard to respond to a known worm. Safeguard was also tested without Prelude to evaluate Safeguard's ability to detect and respond to an unknown worm on the network. Two families of worm were emulated for test purposes, namely Code Red and Slammer.

A set of workflows correspond to mechanisms to detect an unknown worm were constructed. The workflow shown in figure 4 is one of them. Linked workflows relate to monitoring patterns of behaviour of a suspected attacking host and another that is initialized for each suspected victim machine. These are initialized on the first indications and then they build up evidence of attacker or victim. All the workflows are within the correlation agent and run simultaneously.

Experiment	**Worm family**	**Number of threads**	**Packet size (Kb)**
1	Code Red	26 & 20	65 & 30
2	Code Red	20	65 & 30
3	Code Red	20	65 & 45
4	Slammer	50	65
5	Unknown Code Red	10 & 2	30
6	Unknown Slammer	50	65

Table 2. *Parameters for worm experiments*

6 Results

The graphs below show the results of the tests of the worm scenario using the worm emulation software. The X axis represents time, with each cell corresponding to 30 seconds. The Y axis displays network throughput in Mbit/sec. In the graphs the normal electrical activity on the network appears as a regular jagged line just below the 2.0 Mbit/s level. Irregular activity above this level indicates the impact of the worm upon the network. The parameters of the worm emulation for each experiment are included below the graphs.

6.1 Code Red Type Impact

Experiment 1 without Safeguard

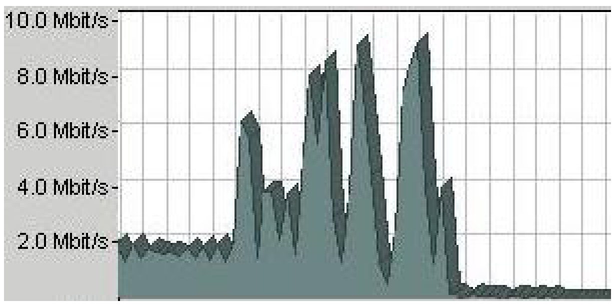

Figure 6 *Network traffic to the control centre*

As indicated in Table 2 the attacker worm with parameters shown starts and spreads to the victim. Here the worm emulation completely disrupts network activity causing the flow of electrical data to the control centre to fall to zero.

Experiment 1 with Safeguard

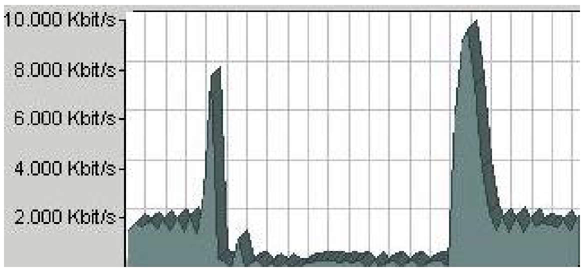

Figure 7 *Network traffic to the control centre: Code Red and Safeguard*

The same dramatic impact on network traffic can be seen as in Experiment 1. However the automatic response of Safeguard manages to kill the worm and the flow of electricity data eventually resumes.

Experiment 2 without Safeguard

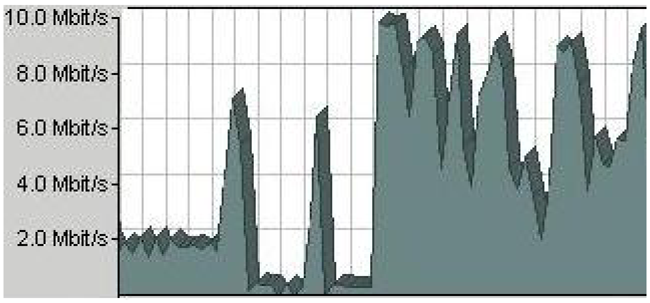

Figure 8 *Network traffic to the control centre: Code Red and No Safeguard*

With these parameter settings the Code Red worm disrupts network activity and then continues to propagate, seriously disrupting network traffic.

Experiment 2 with Safeguard

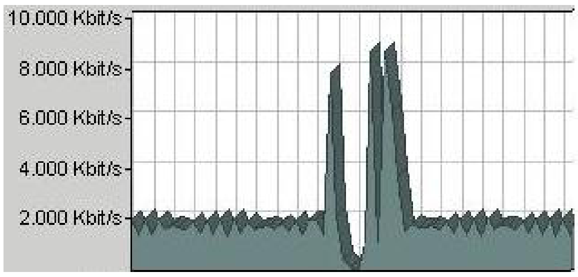

Figure 9 *Network traffic to the control centre: Code Red and Safeguard*

The worm is killed quite rapidly here without a major disruption of network traffic.

Experiment 3 without Safeguard

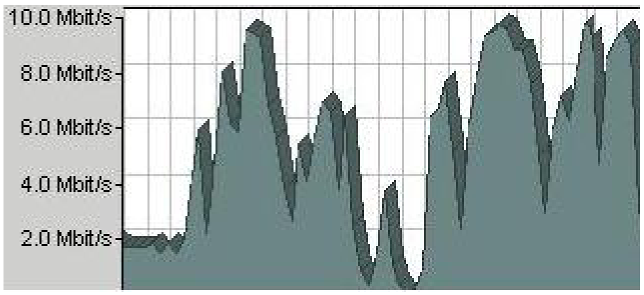

Figure 10 *Network traffic to the control centre: Code Red and No Safeguard*

With these parameter settings the worm disrupts network activity and then continues to propagate, seriously disrupting network traffic.

Experiment 3 with Safeguard

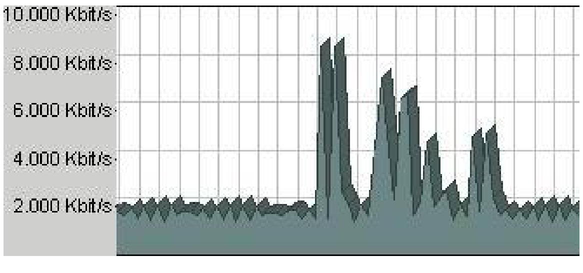

Figure 10 *Network traffic to the control centre: Code Red and Safeguard*

The worm is killed quite rapidly without a major disruption of network traffic.

6.2 Slammer Type Impact

Experiment 4 without Safeguard

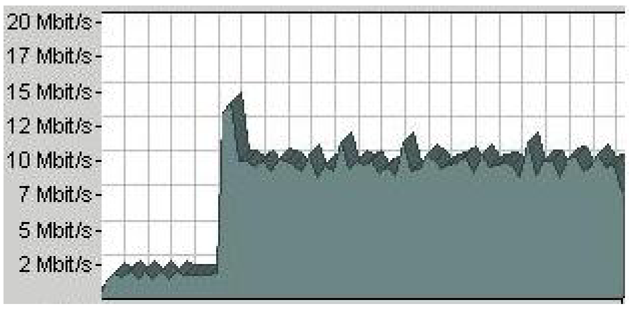

Figure 11 *Network traffic to the control centre: Slammer and No Safeguard*

The impact of the Slammer worm with these parameters is serious, but does not completely eliminate the electricity traffic whose zigzag signature can be seen continuing on top of the Slammer traffic. Part of the reason for this is that Slammer only copied itself onto two machines in this experiment, which reduced the amount of network disruption.

Experiment 4 with Safeguard

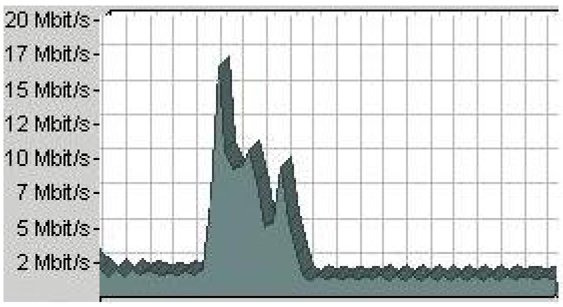

Figure 12 *Network traffic to the control centre: Slammer and Safeguard*

The rapid response of Safeguard prevents any serious disruption of network traffic.

6.3 Unknown worm impact

An unknown worm is in our case is simply a worm that cannot be detected by Prelude and so has to be detected only by the anomaly detectors. Such detection is less specific, only indicating a deviation from normality.

Experiment 5 with Safeguard

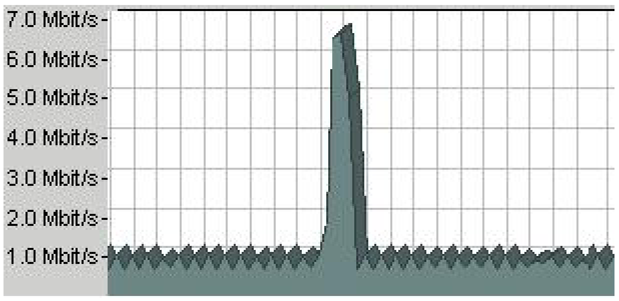

Figure 13 *Network traffic to the control centre: unknown Code Red worm and Safeguard*

The surge of network activity and file integrity modifications led to the unknown Code Red family worm being rapidly detected and eliminated by Safeguard.

Experiment 6 with Safeguard

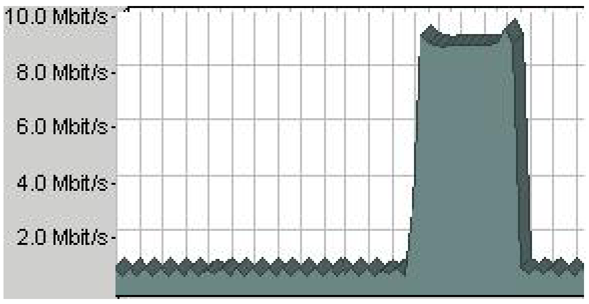

Figure 14 *Network traffic to the control centre: unknown Slammer worm with Safeguard*

The Slammer traffic led to the rapid detection of the worm, but it took some time for the killing messages to get through so that the recent worm processes could be eliminated.

7. Future Work

The results presented are seen as early indicators of the capabilities of the system and we accept that there is a need for more comprehensive testing over a wider range of scenarios and parameters and investigation of the effects on agent traffic in virulent cases. Techniques to improve the performance of the system the case of unknown worms are being investigated. An obvious, but still important problem is that the response of Safeguard was often hampered by the network congestion caused by the worm. There is a need to have a dedicated network for Safeguard or some kind of dedicated bandwidth allocation for Safeguard traffic, e.g. routing with DiffServe or Zebra [16]. Again it is planned to look at this.

During the project a considerable amount of effort was placed on devising anomaly detectors for electricity and telecommunication data. For example, sophisticated anomaly detection for network data building on the BIRCH algorithm [12] was tested in real network traffic, and a special technique based on invariants in electrical data was developed. We plan to integrate these into the electricity test bed and evaluate the Safeguard system on large sets of data extracted from electricity network simulations.

8. Conclusions

This paper has concentrated on an aspect of our approach to correlation. By correlation is meant the synthesis of information from diverse kinds of anomaly detector and IDS and making sense of this information. Information is synthesized for patterns of attack that can extend over a period of time. The time events occur can matter and one of the strengths of the approach chosen is that this can be accommodated in the correlation process. For example, a scan in itself may not be significant, but in the context of a suspected kind of attack it can give supporting or contradictory evidence.

The experiments reported here demonstrate that the worm emulation software has a significant impact on the system performance and if no appropriate measures are taken the network and individual machines can come to standstill. With Safeguard in place the worm activities can be detected and eliminated successfully. In all of our test cases the network was restored to its normal state. These results suggest that the Safeguard method can be an effective way of dealing with known and unknown worms in the network. However more tests will need to be performed to establish the parameters for this response in more detail.

Whilst the experiments reported related to worms, workflows can be used to model a wide range of behaviour, ranging from monitoring the actions of repair staff (and so recognising and reacting to attacks from within) to detecting and managing load shedding.

Acknowledgements

This work was partially funded by the CEC as part of the Safeguard project, and help from our partners is much appreciated: Aplicaciones en Informtica Avanzada (AIA), Spain; ENEA, Italy; Linköping University. Sweden; Swisscom Corporate Technology. Switzerland. Also Frank J. Wesslowski developed the graphical interface to the workflow software while he was an MSc student the Electronic Engineering Department.

We are also grateful for the support from the EPSRC grant number GR/R97733/01 “Certificate Based Distributed Firewalls for Secure E-commerce Transactions”

References

[1] Thomas M. Chen. “Intrusion Detection for Viruses and Worms” http://engr.smu.edu/~tchen/papers/iec2004.pdf
[2] Safeguard website. http://www.ist-safeguard.org/
[3] Layna Fischer (ed). “The Workflow Handbook 2003”. Published in association with the Workflow Management Coalition (WfMC).

[4] W.M.P. van der Aalst. “The Application of Petri Nets to Workflow Management”. Journal of Circuits, Systems, and Computers, Pages 21-66, 1998.
[5] Workflow Management Consortium website. http://www.wfmc.org
[6] Cosa website. http://www.transflow.com/english
[7] OpenWFE website. http://www.openwfe.org
[8] Bossa website. http://www.bigbross.com/bossa
[9] JavaBayes website. http://www-2.cs.cmu.edu/~javabayes/Home/index.html
[10] J. P. Muller, M. Pischel, M. Thiel. “Modelling reactive behaviour in vertically layered agent architectures”. Intelligent Agents. pages 261276. Springer Verlag, LNAI 890, 1995.
[11] Ebayes website. http://www-2.cs.cmu.edu/~javabayes/EBayes/index.html/
[12] K. Burbeck, S. Nadjm-Tehrani. "ADWICE - Anomaly Detection With Fast Incremental Clustering”. submitted for publication, Draft version available from [2].
[13] D. Gamez, S. Nadjm-Tehrani, J. Bigham, C. Balducelli, K. Burbeck, T. Chyssler. Chapter 19: Safeguarding Critical Infrastructures, “Dependable Computing Systems: Paradigms, Performance Issues, and Applications”. Wiley [Imprint] Inc. 2005.
[14] Prelude website. http://www.prelude-ids.org/
[15] Afick website. http://afick.sourceforge.net/
[16] Zebra website. http://www.zebra.org/