A Cooperative AIS Framework for Intrusion Detection

Katja Luther and Rainer Bye
DAI-Labor
Technische Universität Berlin
Franklinstr. 28, 10587 Germany
katja.luther, rainer.bye@dai-labor.de

Tansu Alpcan and Achim Müller
Deutsche Telekom Laboratories
Technische Universität Berlin
Ernst-Reuter Platz 7, 10587 Germany
tansu.alpcan, achim.mueller01@telekom.de

Şahin Albayrak
DAI-Labor
Technische Universität Berlin
Franklinstr. 28, 10587 Germany
sahin.albayrak@dai-labor.de

***Abstract*— We present a cooperative intrusion detection approach inspired by biological immune system principles and P2P communication techniques to develop a distributed anomaly detection scheme. We utilize dynamic collaboration between individual artificial immune system (AIS) agents to address the well-known false positive problem in anomaly detection . The AIS agents use a set of detectors obtained through negative selection during a training phase and exchange status information and detectors on a periodical and event-driven basis, respectively. This cooperation scheme follows peer-to-peer communication principles in order to avoid a single point of failure and increase the robustness of the system. We illustrate our approach by means of two specific example scenarios in a novel network security simulator.**

I. Introduction

There are known limitations to signature-based intrusion detection systems (IDSs) such as the need to manually update the signature database and inability to detect new threats. As an alternative, anomaly detection having the promise of detecting new attack types, is gaining popularity. However, it also has inherent problems such as false positives, i.e. classifying a normal event as an attack. In this paper we introduce a collaborative intrusion detection (ID) approach based on an artificial immune system (AIS) which is in turn inspired by biological immune system (BIS). We present an agent-based AIS as well as a collaborative ID framework where hosts share information about possible attacks to address the false positives problem. We also simulate the algorithms proposed in specific scenarios and verify our approach via realistic simulations. The simulations are run in a novel network security simulator (NeSSi) which will be introduced.

Our goals in this study are to investigate the applicability of basic AIS-based methods to network intrusion detection, introduce novel collaborative schemes for ID utilizing peer-to-peer (P2P) architectures, and conduct an initial evaluation of the framework developed in the NeSSi for two different scenarios. The rest of the paper is organized as follows: In the next section we provide a general overview on AIS and collaboration approaches. In Section III we propose a cooperative AIS framework where we describe the cooperation between the host-based AIS agents. Section IV introduces the NeSSi environment used in the simulations which are described in Section V and followed by a discussion of the results in Section VI. The paper concludes with the remarks of Section VII.

II. Overview of Cooperative AIS

In this section, we provide an overview of collaborative frameworks and AIS with a focus on intrusion detection.

A. Collaboration Frameworks

We are interested in the cooperative approaches between different detection units for the purpose of intrusion detection. We define a *detection unit* as component for detecting intrusions in a networked environment, which could be either signature- or anomaly based. In this context, we define *collaborative detection* as the process of detecting intrusions within a group of detection devices which share information with each other.

In order to develop such cooperative detection schemes, we need to answer questions such as how do detection units know and understand each other and what information needs to be exchanged to improve intrusion detection performance. Groups of detection units can be configured manually in a static way but as contemporary networked systems are more and more dynamic, this leads to a lot of administrative overhead. P2P systems offer a good alternative in a distributed detection environment for the purpose of maintenance and look-up of collaborative detection groups as they have shown to be efficient in other application areas such as file sharing, e.g. Gnutella[1]. On the other hand, there are several approaches that allow the exchange of sensor data even between different detection units. One example is IDMEF, the Intrusion Data Message Exchange Format [1]. It defines a common language based on XML allowing different systems to exchange sensor data or detection results. An example how exchanged information between detection units can be taken into account is given in [2], where security settings of a browser or an e-mail client are adjusted according to attack events sent by detection units. In this paper, we introduce on the one hand a basic P2P infrastructure which handles the tasks of look-up, maintenance, and communication between the detection units. On the other hand, we present a collaborative approach, where the so called detection status of each participant is taken into

[1] http://www.gnutella.com/

account by all the other members and detection results are shared. Moreover, the detection status of a unit is computed using a BIS/AIS-based method which will be described in the following section.

B. Biological and Artificial Immune Systems

The BIS can be considered as a successful classification system which distinguishes between *self* and *nonself* as well as "good" self/nonself and "not good" self/nonself. In this context *self* refers to the organism and *nonself* to the intruders, respectively. Its advantages include robustness, the ability to detect new attacks and to learn these attacks in order to respond in a faster second immune reaction. The immune system is a very complex system with different defense layers such as the skin as a physical border for pathogens but we limit our description to the adaptive immune system as our AIS algorithms are based on it. The adaptive immune system consists of two kinds of cells, the t- and b-cells (there are more differentiations which can be ignored for the purposes of our model). The immune cells have receptors on their surface with specific shapes and charges. They can bind to patterns of contrary shape and charge resulting in a pattern matching mechanism. The t-cells mature in the thymus and this is the place, where they learn to differ between *self* and *nonself*. The maturation process is called *negative selection*. Self proteins and cells are presented to the immature t-cells. The cell dies if the receptor of a cell binds to a self-pattern with high affinity. Only cells with receptors not matching presented patterns survive and become mature t-cells. The mature cells migrate into the peripheral lymphatic organs and detect nonself if a receptor of a cell has a high affinity to a pattern at the surface of another cell or molecule. The t-cells not only need to recognize foreign patterns but also to receive co-stimulation from b-cells for a full immune response. This mechanism minimizes the possibility of false positives and an auto-immune-response. The immune system has no central control, and no single point of failure because of the high numbers of single cells building up the immune system. Therefore, the scalability and robustness of the system is very good. For more details we refer to [3].

The idea to adapt BIS algorithms for anomaly detection is not new. In the context of network security there exist systems with immune analogies based on different algorithms such as negative selection, immune-network or danger theory. One of the first approaches are the negative selection algorithm (see II-B.1) and the Lysis system of Stephanie Forrest et al. [4], [5]. Most of these early systems are based on network traffic, especially at the packet level. Other systems have tried to find anomalies in Unix process calls [5]. All these systems have utilized the negative selection algorithm described in [4]. More recently, Luther et al. have developed an agent based AIS for a single host similar to the "lysis" system of Forrest and Hofmeyr [6] (see Figure 1) and a concept for a distributed AIS which constitutes the basis of this paper [7].

1) Negative Selection Algorithm: The main idea of the negative selection algorithm is to produce detectors randomly and compare them to the normal patterns obtained during training. Every detector that matches to these normal patterns is eliminated, and hence, the remaining detectors recognize only abnormal patterns. A pseudocode for the negative selection algorithm is shown in Table I.

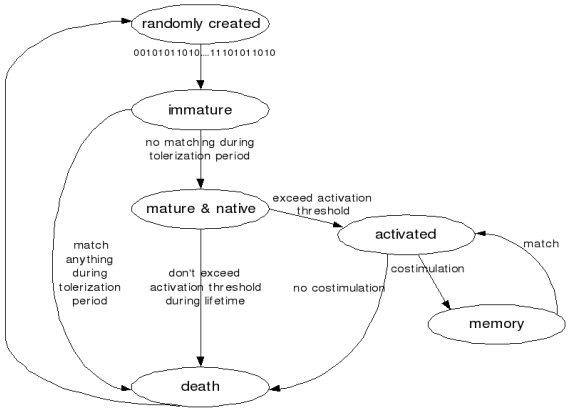

Fig. 1. Negative Selection and Anomaly Detection based on Forrest et al. [6]

TABLE I
PSEUDOCODE OF THE NEGATIVE SELECTION ALGORITHM.

```
SET DetectorSet as the set of Detectors
SET NormalSet from the training data
SET NumberDetectors as the number of Detectors

WHILE DetectorSet size smaller than NumberDetectors
    CREATE a Detector
    FOR each featureVector in the NormalSet
        COMPARE the Detector with FeatureVector
        IF they are similar
            DELETE Detector
        ELSE
            PUT Detector in DetectorSet
```

In Figure 1 the Negative Selection algorithm is presented in the context of the lysis system of Hofmeyr et al. [6]. The first two steps are the negative selection and the following circle describes the clonal selection (see Section II-B.2). In the literature, the AIS is initially used as a packet-based Intrusion Detection System and the detectors are comparable to binary strings composed from the data path triple (IP addresses and used protocol and port). For these systems, the matching decision is based on a binary r-contiguous comparison or a modified version the r-chunk comparison [8]. In later studies, more complex feature vectors and the Euclidean distance have been used for the comparison [9].

2) Clonal Selection: The second important algorithm in artificial immune systems is the clonal selection. This algorithm describes the selection of the detectors that recognize the abnormal patterns. The detectors are continuosly compared to the occurring patterns, and if one is similar to such a pattern, it clones itself. The similarity between a detector and a feature

vector is called *affinity* and the detectors with the highest affinity are selected and cloned. Alternatively, the detectors mutate for a better cover of the nonself space. Mutation means that they do not only produce copies of themselves but also copies with little differences. In the CLONALG algorithm the mutation rate is inversely proportional to the affinity of the detector [10].

There are different clustering algorithms used for anomaly detection in network security such as self organizing maps (SOM) (for example [11]). However, they all characterize normal behavior by computing the difference between the feature vectors and the normal space: if it is significantly different then it is classified as an anomaly. The AIS approach, on the contrary, measures the distance between detectors representing the abnormal. This approach is better suited for a normal space that is not arranged in a small amount of big clusters but in a lot of small clusters. Gonzalez et al. [12] have compared a SOM based and an immune based approach and found that the AIS approach is less sensitive to changes on the threshold. However, in this comparison the time series data created by the Mackey-Glass equation has been used. In this paper, we evaluate the AIS approach using realistic network data we obtain from NeSSi (see section IV).

3) Agent-based Approaches: Its distributed nature is one of the main advantages of the BIS. However, in the case of an AIS it is not necessary to implement all the "small cells" with only one receptor. On the other hand, distribution of tasks and coordination between the AIS hosts seem to be useful properties worth focusing on. In the literature, mostly hierarchical approaches has been proposed (see Boudaout et al. [13] for an agent-based IDS without anomaly detection and Dasgupta et al. [14]) due to the simplicity of management of an agent-system communicating in an hierarchical manner. Unfortunately such systems have a single point of failure. There also exist different approaches that combine the immune system metaphor with other biology inspired algorithms such as evolutionary algorithms or swarm theory. Kim and Bentley have developed a system that creates the detectors by using an evolutionary algorithm [15], [16]. An approach with mobile agents that migrate similar to ants towards a pheromone concentration has been presented by Fenet [17], where the pheromone concentration decodes the distance to a host that detected an anomaly.

III. A Cooperative AIS Framework

A. AIS Architecture

We choose the negative selection algorithm described in Section II-B.1 as the basic algorithm for the selection of the detectors for given training data. The feature vectors are composed of the statistics of the network-traffic on one client. The detailed description of the feature vectors is provided in Section V. After the learning period the new feature vectors are presented to the remaining detectors and the distance between the feature vectors and the detectors is computed. If the distance is smaller than the threshold of the detector, a counter is incremented. When the counter reaches a pre-defined threshold, an alarm is raised and in the next step clonal selection should take place. Clonal selection means that the detector is copied but with small differences in the values of the feature vector (mutations).

In our implementation, we chose a simple anomaly detection scheme using only the positive selection to detect anomalies. If a detector does not reach its threshold during a pre-defined life period it is eliminated from the detector set and a new detector is created. The data for the feature vectors is collected by a monitoring component of the agent. The agent collects the packet data using *jpcap*[2] and creates feature vectors continuously. During the training period, each match between a detector and presented patterns increments a detector-specific counter. If a certain threshold is reached then the detector is eliminated(negative selection). The distance between the detectors and the feature vectors is computed by the hamming distance.

B. Cooperation Structure

Several hosts in a network running instances of the presented AIS architecture can presumably achieve better detection performances if they cooperate by sharing their detection results and own status. We present a P2P infrastructure allowing different AIS agents to collaborate for this purpose.

P2P systems in general can be classified into three categories: Purely decentralized, partially centralized, and hybrid decentralized [18]. The P2P infrastructure we use in this paper is based on a hybrid decentralized architecture. The nodes are equal in their abilities but one node acts always as the "super node" for the purpose of the initial look-up of other peers containing AIS-systems. Apart from this, the clients communicate with all other peers autonomously. We next describe the course of action for an AIS client that wants to share information with the other peers. The central entity is denoted as the server and the other AIS agents are simply called clients or peer nodes. The `PEER` data structure contains the IP address of the peer, a counter for maintenance purposes and data applicable to the system using the P2P infrastructure.

- **Initialization**
  A node that wants to connect to the P2P overlay network contacts the server with a *register* message.
- **Send Update Message**
  Informs all other client nodes about interesting updates such as the status of the detection units. We refer to the next section for details regarding the content of messages exchanged between AIS agents.
- **Receive Message**
  Four types of messages are handled:
  1) In the case of a *peer list from the server* message the node is added to the client nodes list.
  2) Upon receipt of an *update message* the state or information about a node in the peer list is actualized. For all the other nodes, an age counter

[2]http://jpcap.sourceforge.net/

TABLE II

PSEUDOCODE FOR PEER-TO-PEER COMMUNICATION

```
List of peers peerList

INITIALIZATION
    Send to server Register Message

SEND UPDATE MESSAGE
    Send to peers in list Update Message

RECEIVE MESSAGE
    CASE Message Type OF

        Peer list Message from server:
            Add peers to own list

        Update Message:
            FOR all peers in list
                IF sender of Message equals peer
                    actualize peer with Update
                ELSE
                    increase age counter
                    IF age counter > threshhold
                        delete peer from list
                IF sender of Message was not in list
                    add sender to peer list

        Quit Message:
            remove sender from peer list
        Request List Message:
            Send peer list to client

DISCONNECT
    Send to server Quit Message
    Send to peers in list Quit Message
```

is incremented. If the counter reaches a specific threshold indicating that a node has not sent update messages in a certain time interval it is deleted from the list. This is important when a client is not able to disconnect properly. This approach can be used for the maintenance of the list as periodic messages are an essential part of our system (see III-C). If the sender of the update message is not an element of the peer list, it is added automatically to that list.

3) A *quit message* results in the deletion of the sending node from the list.
4) *The request list message* is answered by transmitting the list of nodes to the requestor.

- **Disconnect**
  A message from a node to the central server but also to the clients in the peer list that the agent wants to disconnect.

In this paper we consider scenarios and applications of the detection system to subnetworks owned by a company or a university department. Therefore, there is a valid range of known IP addresses in such subnetworks. If an outside attacker tries to carry out a spoof attack, i.e., manipulates the packets with IP addresses from inside the network, this can be easily detected at the gateway to other networks respectively the Internet. Thus, this mechanism provides some protection from outside attackers that want to be added to the peer list and falsify detection results.

C. Cooperation in the domain of the AIS

The success of a biological immune system results from the cooperation of its various components. An immune system response is initiated only if different events take place concurrently. In our distributed environment we want to achieve a similar effect by using a variety of feature vectors and exploiting the cooperation between the host based AIS agents. We hence associate every AIS agent with a status metric that represents the possibility of infection (anomaly), which is shared through the P2P system described in the previous section.

A second possibility of cooperation appears when a host detects an anomaly. If its own status reaches a pre-defined threshold it sends a message to his neighbors containing its status as well as the actual detectors associated with the alarm. Such messages are called *event driven*. If a host gets an event-driven message, then it adds the new detectors to his own detector set and updates its own status. The new status is computed as a function of the own status after running the detection module and the status obtained from the received messages. However, in this process the agents give a lower weight to the status of the neighbors than their own.

IV. SIMULATION ENVIRONMENT

We focus in our project on the detection of malware within a network. The goal is to investigate how to efficiently detect and eliminate malware packets as they are traversing the network before reaching their designated target. In other words, the ultimate goal is the development of a network-level intrusion detection system. In the scope of this project, a novel network security simulator (NeSSi) has been developed which allows the creation of IP-based networks and corresponding packet streams.

An anomaly detection algorithm like the one developed in this paper (see Section III) needs meaningful and realistic data in the training phase. A lot of research is based on the training data sets provided by the DARPA in the intrusion detection evaluation project[3]. This data is meaningful because it has been recorded in a real but small networked environment. The latest data sets are from the year 2000. Hence, the data is static, inflexible, and dated. A significant advantage of NeSSi is its ability to provide dynamic data for evaluation of intrusion detection schemes such as the one developed in this study.

The NeSSi is built upon the JIAC (Java Intelligent Agent Componentware) agent framework [19], which is utilized in modeling and implementing the network participants such as routers, clients, and servers as software agent entities. The front-end consists of a graphical user interface that allows the creation of arbitrary IP network topologies. The communication between clients, servers, and routers takes place by real IPv4 packet transmission. This is realized using communication services between the agents. Above the network layer the

[3]http://www.ll.mit.edu/IST/ideval/index.html

simulator emulates TCP and UDP protocols on the transport layer. At the application layer we support the HTTP and SMTP protocols faithfully. The NeSSi allows us to capture the packets that traverse over a link and write them to the common TCPDump file format[4]. The TCP/IP stack is emulated in such a way that a tool like *Ethereal*[5] can extract the TCP payload and show the application layer protocols such as HTTP with the appropriate content.

We also introduce the concept of profiles for clients within the simulator. A profile simulates the sequence of characteristic actions a computer performs every day. This includes on the one hand the user (E-Mail, HTTP) and on the other hand the system inherent behavior such as the protocols used by the operating system, e.g. NETBIOS. A timestamp can be assigned to each action and each profile can be executed according to a system-wide timer. The actual time and a

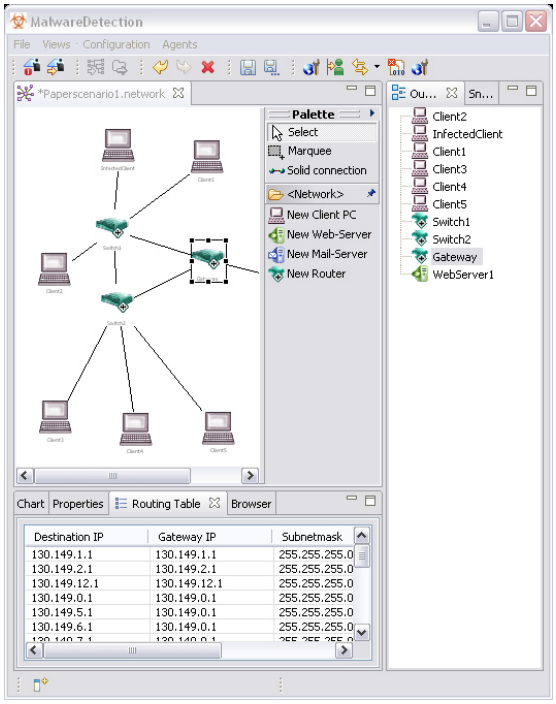

Fig. 2. Graphical user interface of the NeSSi.

multiplicator for the execution speed can be configured in the NeSSi. Figure 2 shows the graphical user interface of the NeSSi tool. Further details on the network simulator can be found in [**?**] and the used protocols and profiles are described in the next section.

V. Simulations

As part of the characterization of "typical" client behavior we have recorded daily TCPDumps of several Windows workstations on the subnetwork of the DAI-Labor. After an analysis of the captured traffic we have modelled the UDP background traffic on an example network consisting of idle clients.

[4]http://www.tcpdump.org/
[5]http://www.ethereal.com/

TABLE III
UDP client background profile

Port Number	Protocol	Packets per hour
88	Kerberos	3
123	NTP	1
137	NETBIOS Name Service	1
138	NETBIOS Datagram Service	2
389	CLDAP	6

The protocols chosen and summarized in Table III include NETBIOS which is used for name and session services, NTP for clock synchronizing, Kerberos as standard authentication protocol for Windows and CLDAP for X.500 directory access. NETBIOS is the only protocol that uses broadcast messages, the others are unicast. The traffic volume in inactive mode consists of 13 packets per hour and host. In addition, the clients produce HTTP traffic to represent user interaction. We have modelled a higher usage of HTTP traffic by the clients at specific times per day referring to beginning of the workday, after the lunch break, and after the working hours.

We use the timer based profiles described in Section IV for modeling the behavior of the client. Here, HTTP is faithfully emulated. For the UDP background traffic we use UDP-packets with correct ports and behavior (unicast, broadcast) but without meaningful content. The content of single UDP-packets does not play an important role because we are considering packet flows, used ports, and IP addresses for our analysis without resorting to a deep inspection approach. Each client has a defined subset of ports where it accepts packets. The packets arriving on these ports are regarded as valid connection attempts. In this paper, we use the following ports for UDP-based services: 123 (NTP), 137 (NETBIOS Name Service), 138 (NETBIOS Datagram Service) and 1500 (P2P communication).

The P2P infrastructure is implemented based on the UDP-protocol in the NeSSi. Hence, the port 1500 is reserved for it. We do not consider valid TCP-Ports here, because connections in the application layer protocol HTTP are initiated on the client side. Therefore, no client needs to listen to TCP connection attempts. In the simulations, each client measures incoming and outgoing connections using a network statistic component which records data such as the used ports, the number of packets, and whether a connection attempt was valid or not. Feature vectors are extracted out of this data serving as an input for the AIS. We next consider two scenarios where each client executes the client-profile according to the global timer. Some of the clients run an AIS agent. Disabling respectively enabling the P2P infrastructure allows the comparison of the detection results between the cooperative and non-cooperative approaches.

The AIS on the hosts has three different phases: monitoring, training, and detecting. In the monitoring phase the measuring component of the clients samples all incoming and outgoing packets. The feature vectors are created and stored in previ-

ously specified time intervals. The feature vectors we use are given in Table IV.

TABLE IV
FEATURE VECTORS FOR THE AIS

Feature vector 1	Feature vector 2
daytime	daytime
main IP addresses TCP	number of TCP connections
main IP addresses UDP	number of UDP packets
main ports	number of used ports
	number of port scans
	number of TCP packets

During the training period a set of detectors is created and compared to the training data collected in the monitoring phase. As described in Section II-B.1, all detectors matching a feature vector of the training set are deleted and new detectors are created. The detectors are not created in a completely randomized fashion because the space to be covered would be huge. In our simulation environment we have a limited range of IP addresses, so we can reduce the space of possible addresses. The same argument can also be made for the ports. Hence, we use only port numbers between 1 and 450 because we only use ports in this interval. We choose an interval of 20 seconds with a multiplicator of 50 (in realtime that would be 16 minutes) The detectors are compared during the detection period with the incoming feature vectors created by the same statistic component as the one in the monitoring period. If one or more detectors match a feature vector, the status of the AIS is incremented proportional to the affinity and the number of matching detectors. If the status reaches a threshold, an alarm is issued. The feature vectors that did not raise an alarm are stored and used for future training, so that the system is adaptive to variations in the environment.

The cooperation of the AIS in the network is based on the shared status. After every status calculation each AIS component sends its status and a counter to the other peers through the P2P system and updates his own status according to the incoming status of the adjacent peers. Every client uses only the most recent status during the update, i.e., they are ignored if the incoming status values are not up to date. We next present two different attack scenarios in a typical subnetwork such as a department of a university or a small company.

A. Scenario I: Compromised host in a subnetwork

In this scenario, one of the client computers has been compromised by an attacker. This means a vulnerability of the client has been exploited and the attacker has complete access to the client's resources. According to the five Ps [18] Probe, Penetrate, Persist, Propagate and Paralyze, the attacker tries to persist on the compromised machine, for example by installing a backdoor. The next step is the "Propagate" stage where the subnetwork is scanned for other further vulnerabilities. In this scenario, we model a port scanning profile that extends the previously defined client profile. This profile includes a scanning of several ports on all clients in the whole subnetwork over a variable time period.

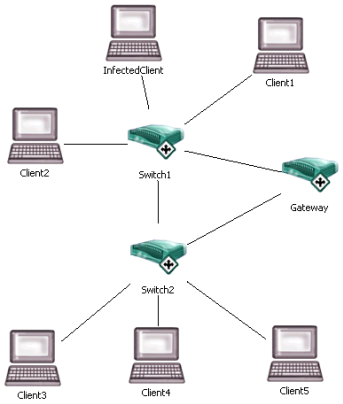

Fig. 3. Scenario I: Compromised host in a subnetwork

B. Scenario II: Detecting a worm attack

In this scenario one host is infected by a worm that spreads via only one UDP packet like the infamous Slammer worm did [20]. The worm uses only a single port (port 137). However, if it reaches a host that provide the port the host is infected with a probability of 0.7. If a client is infected by the worm, it starts to send the same UDP packet to randomly created IP addresses and tries to infect other clients.

VI. SIMULATION RESULTS

We first give an overview of the simulation results. In Table V the detection rate and false positive rates are depicted for a single AIS and the collaborative approach in both scenarios described in the previous section.

TABLE V
FALSE POSITIVES FOR THE DIFFERENT SCENARIOS

Scenario	false positives	true positives
1, single	25%	79%
1, cooperative	18%	100%
2, single	16%	54%
2, cooperative	10%	57%

As to be expected, the false positive rate for both scenarios is quite high for the single client AIS. Yet, the collaboration between the AIS clients decreases the false positive rate to 18 and 10% respectively, while the detection rate increase. In Table VI the status of an AIS agent before and after cooperation is presented. The rows printed in bold show when the cooperation helps eliminating false postives. We also observe that in one case the cooperation creates a false negative. We next discuss the results for the two scenarios separately.

TABLE VI

STATUS OF THE CLIENTS BEFORE AND AFTER COOPERATION

Status client 1	after cooperation	Status client 9	after cooperation
...	...	...	...
0.01	0.20	0.20	0.04
0.04	0.07	0.04	0.00
1.00	**0.80**	**0.00**	**0.20**
0.16	0.13	0.00	0.00
0.00	0.00	0.00	0.00
1.00	**0.80**	**0.00**	**0.00**
2.00	1.60	1.00	1.20
0.30	0.40	2.00	2.00
1.00	1.20	1.20	1.36
1.01	0.80	0.00	0.00
...	...	...	...
0.00	**0.20**	**1.00**	**0.80**
0.00	0.00	1.00	0.80
0.00	0.00	0.16	0.12

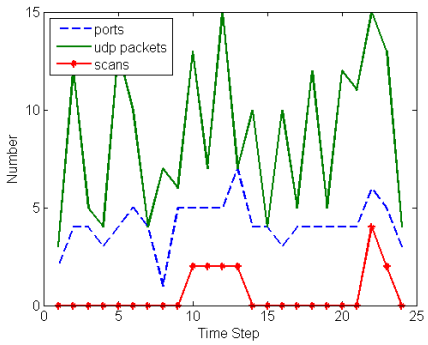

Fig. 4. Network statistics for Scenario 1.

Scenario I

In Figure 4 we observe at a specific client that during the scanning the number of ports as well as UDP packets increase due to packets arriving to a port that is not normally used between time steps 9 and 11 and between 22 and 23. Here, the term *scan* denotes a packet that is received on a port normally not used by the client.

In Figure 5 we observe a high number of packets arriving to the same client from the IP address ending with 18.2 in the same time intervals as when the high number of UDP packets and scans are observed. For a single client AIS it is difficult to differ between "normal" events such as contacting a new webserver and a port scan. Such confusions easily result in false positives. By using the cooperative approach, several clients recognize an anomaly and communicate it to others. This minimizes the possibility of false positives as shown in Table V.

Scenario II

In Figure 6 the high variance in usage of port 137 (shown with a bold line) is identifiable. We also observe a larger number of different IP addresses contacted by the client or contacting to the client, could also be interpreted as a sign of anomaly, too. In scenario II we have observed some false

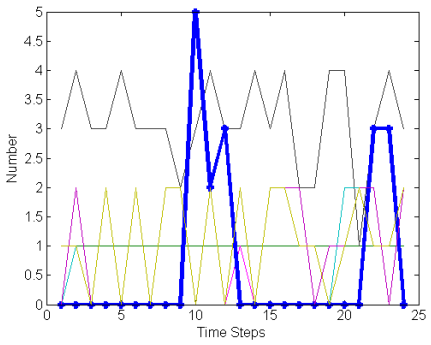

Fig. 5. Contacted IP addresses in Scenario I.

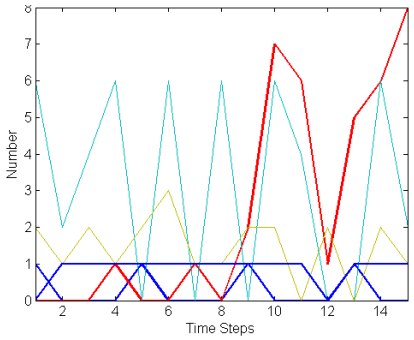

Fig. 6. Number of packets/port for client 9 in scenario II

negatives because it is harder to detect a single abnormal packet. It could be detected only if the AIS client is already infected by sending a lot of packets over port 137.

As seen in Table V, the detection rate for the first scenario is very high. Although the false positive rate is high for the AIS on a single client, the cooperative approach has improved the results. Comparing the different feature vector types, we can see that for the first scenario the second feature vector (in Table IV) is detecting more anomalies than the feature vector with the IP addresses and port numbers which in turn cause less false positives.

In Scenario II the feature vectors of the second category detects the most anomalies because of the unusual port usage pattern (see Figure 6) and the increasing number of different IP addresses. We see more false negatives because the anomaly we are looking for is harder to detect. The defined detectors are based on IP addresses and ports. They can only detect anomalies if there is a significant deviation in usage patterns. The statistical detectors are not powerful enough for this scenario because the appearance of abnormal traffic has to be significantly different from the normal. In this scenario the challenge of the huge state space also plays an important role as the system has to detect an increase in the usage frequency of a port that is otherwise very rarely used. Therefore, we have reduced the space for ports to between 80 and 450 after observing that the simulator only uses ports in this interval. In a real world scenario such domain knowledge has to be used for the creation of the detectors in order to reduce the

possibility of gaps in the coverage.

VII. Conclusion and Future Work

In the simulations we have observed that the detection rate is good and the false positive rate can be decreased through cooperation of the AIS agents. Taking the simplicity of our implementation into account this is an indicator of the fact that our approach works well for anomalies appearing on different hosts in a subnetwork.

The Cooperative AIS Framework for Intrusion Detection comprises several sub components. Hence, the consideration of future work is stated on the one hand for the constituent parts AIS and P2P system, but on the other hand for the whole detection framework in general.

The P2P approach we present in this paper has proven its feasibility during the simulations due to his robustness and simplicity. For usage in large scale deployments the system needs additional techniques to prevent being compromised by attackers. Therefore, trust models and in special adaptive ones due to the dynamic nature of P2P systems have to be analyzed. A second interesting area is providing context or meta information about the participating units such as whether a detector resides on a "normal client", database server, or a separate machine the detection unit is installed on. This leads to a new interesting research topic in the security domain: formation of detection groups based on interests.

For the AIS or other detection units there exist different directions for future research. It would be interesting to implement and test more pattern matching and machine learning algorithms in the simulator. To find the best solution for the problem of gaps in the set of detectors it seems to be useful to create the detectors with better methods than uniform random generation. One possibility could be Quasi-Monte Carlo methods for a better coverage of the detection space. The cooperation presented in this paper is only between hosts with the same type of detection unit. In the future we will also investigate different types of detection units and cooperation schemes.

Regarding the detection framework in general, the Cooperative AIS Framework for Intrusion Detection has to prove its feasibility in a real networked environment. Hence, a test laboratory composed of routers and end devices is planned for the evaluation of the system. Additionally we aim to apply the well known DARPA data set (see III-C), although being static and out dated, to our simulations.

Acknowledgments: This work is part of a project funded by Deutsche Telekom. The authors would like to thank all the people making this paper possible, especially Joël Chinnow, Thorsten Rimkus and Sebastian Linkiewicz for their work at the network simulator.

References

[1] "Intrusion detection message exchnage format," 2006, http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-16.txt.

[2] V. Vlachos, S. Androutsellis-Theotokis, and D. Spinellis, "Security applications of peer-to-peer networks," *Computer Networks*, vol. 45, no. 2, pp. 195–205, June 2004.

[3] C. A. J. Jr., P. Travers, M. Walport, and M. J. Shlomchik, *Immunobiology*. New York: Garland Publisher, 2001.

[4] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proceedings of the IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 1994, pp. 202–212.

[5] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A Sense of Self for Unix Processes," in *Proceedings of the 1996 IEEE Symposium on Security and Privacy*. Los Alaitos, CA: IEEE Computer Society Press, 1996, pp. 120–128.

[6] S. Hofmeyr and S. Forrest, "Architecture for an Artificial Immune System," *Evolutionary Computation Journal*, vol. 8, no. 4, pp. 443–473, 2000.

[7] K. Luther, "Entwurf eines Künstlichen Immunsystems zur Netzwerküberwachung auf Basis eines Multi-Agenten-Systems," Master's thesis, Technische Universität Berlin, 2006.

[8] J. Balthrop, F. Esponda, S. Forrest, and M. Glickman, "Coverage and Generalization in an Artificial Immune System," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. New York: Morgan Kaufmann Publishers, 2002.

[9] F. A. Gonzalez and D. Dasgupta, "Anomaly detection using real-valued negative selection," in *Proceedings of Genetic Programming and Evolvable Machines 2003*. Kluwer Academic Publisher, 2003.

[10] L. N. de Castro and F. j. Von Zuben, "Learning and Optimization Using the Clonal Selection Principle," in *IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems*, 2002.

[11] S. Albayrak, C. Scheel, D. Milosevic, and A. Muller, "Combining self-organizing map algorithms for robust and scalable intrusion detection," in *CIMCA '05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 123–130.

[12] F. A. Gonzalez and D. Dasgupta, "Neuro-Immune and Self-Organising Map Approaches to Anomaly Detection: A Comparison," in *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*. University of Kent at Canterbury: University of Kent at Canterbury Printing Unit, September 2002, pp. 203–211.

[13] K. Boudaoud, H. Labiod, R. Boutaba, and Z. Guessoum, "Network security management with intelligent agents," in *Proceedings of 2000 IEEE/IFIP Network Operations and Management Symposium NOMS 2000*, 2000.

[14] D. Dasgupta and H. Brian, "Mobile Security Agents for Network Traffic Analysis," in *Proceedings of DARPA Information Survivability Conference and Exposition II*, vol. 2. IEEE Computer Society Press, 2001.

[15] J. Kim and P. Bentley, "An Artificial Immune Model for Network Intrusion Detection," in *7th European Conference on Intelligent Techniques and Soft Computing (EUFIT'99)*, 1999.

[16] ——, "A Model of Gene Library Evolution in the Dynamic Clonal Selection Algorithm," in *Proceedings of the First International Conference on Artificial Immune Systems (ICARIS) Canterbury*, Sept. 2002.

[17] S. Fenet and S. Hassas, "A distributed Intrusion Detection and Response System based on mobile autonomous agents using social insects communication paradigm." *Electr. Notes Theor. Comput. Sci.*, vol. 63, 2001.

[18] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.

[19] S. Fricke, K. Bsufka, J. Keiser, T. Schmidt, R. Sesseler, and S. Albayrak, "Agent-based telematic services and telecom applications," *Communications of the ACM*, vol. 44, no. 4, pp. 43–48, Apr. 2001. [Online]. Available: http://www.acm.org/pubs/citations/journals/cacm/2001-44-4/p43-fricke/

[20] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," *IEEE Security & Privacy*, vol. 1, no. 4, pp. 33–39, 2003.