Agent Methods for Network Intrusion Detection and Response

Martin Rehák, Michal Pěchouček, David Medvigy, Magda Prokopová,
Jan Tožička, and Lukáš Foltýn

Department of Cybernetics, Czech Technical University in Prague
Technická 2, Prague 6, 166 27 Czech Republic
{mrehak,pechouc,prokopova,tozicka,lfoltyn}@labe.felk.cvut.cz

Abstract. While the need to build the Intrusion Detection Systems (IDS) based on on a distributed and cooperative (P2P) paradigm is being generally acknowledged, the field has been disconnected from the recent advances in the multi-agent research, most notably the field of trust modeling. Our contribution reviews recent implementations of IDS systems and presents them from an agent research perspective. We also identify the opportunities where the agent approaches can be successfully used. Agent techniques can make the IDS more adaptive, scalable and reliable while increasing their autonomy and reducing the maintenance requirements. Besides trust modeling, we propose that the distributed decision-making and planning techniques can be used to shorten the detection-response loop, making the system more robust while facing worm attacks.

1 Introduction

The use of the agent paradigm to develop and deploy an efficient distributed Intrusion Detection and Protection System on the computer network seems natural and the idea has appeared relatively early [1]. However, since then, the agent technology and IDS field have largely ignored each other, possibly at the detriment of the research quality. This contribution explains how to use the concepts from the agent technology to make the IDS systems able to efficiently counter the new types of threats as presented in the suite. While introducing the topic of network security to agent research community, this paper is also relevant for network security researchers, as it presents a wide choice of useful techniques applicable to their field.

1.1 The Threat

In the past most of the attacks targeted a single system, in an attempt to disable it or to gain an unauthorized access to the data in the system. Today, the most dangerous threats no longer attack hosts, but rather the network infrastructure: worms and Distributed Denial of Service (DDOS) attacks launched from bot nets are currently a significant threat, and the protection options remain limited. These attacks are currently used to extort the money from service providers for not launching DOS attacks against their sites [2].

V. Marik, V. Vyatkin, A.W. Colombo (Eds.): HoloMAS 2007, LNAI 4659, pp. 149–160, 2007.
© Springer-Verlag Berlin Heidelberg 2007

The main danger of worm attacks is similar to that of DDOS, but with a significant difference of not being aimed towards a specific system. Instead, they are designed to spread across the Internet, infect as many vulnerable hosts as possible and use these hosts for further propagation. Typically, they use well known vulnerabilities to infect hosts [3]. As many of the systems connected to the Internet were vulnerable (unpatched), the *CodeRed I* worm infected more than 250 000 systems during the 9 hours of July 19, 2001 [4]. *CodeRed I* and *II* worms exploit the same vulnerability of the Windows IIS server. On the other hand, these (distinct) worms differ by their spreading strategy – while the *CodeRed I* (and *SQL-Snake* that targets the Microsoft MS-SQL servers) uses a random IP generator, the *CodeRed II* worm uses a more local approach, with an increased probability of local spreading. A similar spreading strategy is also used by the *Nimda* worm.

The *Slammer* worm, that has attacked in January 2003 exploited a vulnerability (discovered in June 2002) in the MS-SQL server and MSDE, its desktop database engine. It's spread was very fast, resulting in an infection of 90% of vulnerable hosts in 10 minutes, following the classical exponential growth curve limited only by the size of the vulnerable population [5]. The main cause of disruption was not the benign payload of the worm, but its propagation. In contrast to previous, TCP based worms, the horizontal UDP scan aimed at port 1434 was faster and allowed a better concurrency model. The worm was also able to attack wider range of hosts, due to its ability to use multiple exploits for infection. Therefore, the spread of the worm was limited only by the available bandwidth, and its propagation disrupted the connection of the infected database servers.

Many systems are considered safe from worm attacks as their configuration is not typical and the random spread of the worm is unlikely to find a vulnerable host. When we examine an attack of the *Witty* worm in 2004 [6], we can see a somewhat different image. This worm begun to spread on March 19, 2004, targeting a buffer overflow flaw in RealSecure ISS family of security products. This vulnerability was a very recent one, disclosed only a single day before the worm has appeared. Besides proving that even a less widespread system is prone to attack, the *Witty* worm case have shown that even a very rapid reaction by human standards is insufficient to protect the vulnerable networks, stressing the need to make the protection process automatic (see Section 4).

Using the selected cases mentioned above, we can realize that the security paradigm has changed significantly between the year 1999, when the DARPA data set [7] didn't include any worm-like attacks and the situation today, when self-propagating code (e.g. worms) is considered a major threat. When analyzing the worm's behavior, we shall clearly distinguish between the two components of any worm: its *propagation mechanism* (or strategy) and its *payload*. The propagation mechanism is essentially a code that exploits a particular vulnerability of the targeted group of hosts, combined with the IP scanning strategy. Worm propagation strategies were theoretically modelled [8] and experimentally verified [5]. Therefore, besides the specific vulnerability exploit, most worms differ by their payload. The payload is the code that affects the infected machine, and its effects can range from doing nothing (Slammer) through opening backdoors (CodeRed II) and exposing the sensitive information (Nimda) to malicious acts like data corruption (Witty) or DoS attacks (Code Red I). We argue that the payload

evolutions will be the biggest threats in the future, making the worms a vector of spread and obfuscation of the well targeted attacks. Introduction of backdoors for future exploitation, recruiting the systems for zombie networks or launching distributed DOS attacks are only some of the options for the attacker. A mere launch of the worm with several distinct payloads or with mutation capabilities can allow the attacker to hide a specific, well-directed attack in the mass, making its detection and tracking more difficult. The scanning strategies of worms evolve as well - some recent worms perform pre-scanning to identify vulnerable population, spread slowly to a large group of seed hosts and only then launch an Internet wide attack, making the containment of such widespread threat more difficult.

Bot (*zombie*) [9] networks are a significant threat today: they are composed of infected hosts with a running malicious process (bot) that are able to receive orders through a command infrastructure, and perform an attack, ranging from various types of DDOS to spam sending. Zombie networks are typically controlled through IRC channels, and their detection and suppression is extremely difficult due to their number, distributed nature, slow propagation and the fact that most internet-connected computers lack adequate protection.

We argue that the current IDS systems (Section 2) lack a capability to efficiently detect and counter the worm attack without relying on the human to make a decision. Given the characteristics of the attacks, we perceive this as a weakness and we propose a survey of methods from the multi-agent research (Section 3) to address this vulnerability. Instead of relying on human operator to take a critical decision in real-time, we propose to let the intrusion detection/prevention system to take this decision autonomously, albeit within the strict policy-specified bounds defined by the operator [10]. The automated decision process outlined in Section 4 can (arguably) take into account more factors than human and automatically select the best solution from a wide scope of techniques, making the networks more robust and hands-off human supervision more efficient.

2 Brief Overview of Intrusion Detection Systems

Currently, intrusion detection systems can be roughly split into two distinct categories. Most commercial and existing open-source products fall into the *signature-detection* category: these systems match the packets in the network with the predefined set of rules [11]. While this system can be valuable in detecting known attack patterns, it provides no protection against novel threats, or even new permutations of known attack, that are not represented in its knowledge base. Furthermore, as the rules are typically sparse to achieve efficiency, the signature-based IDS have a non-negligible false positives ratio. Another major disadvantage of the existing systems is a high computational cost of reasonably complete set of rules – currently, complete SNORT database is not typically entirely deployed due to the computational power limitations.

The *anomaly detection* approaches in NIDS (Network IDS) are typically based on classification/aggregation of sessions or flows (unidirectional components of sessions) into classes and deciding whether a particular class contains malicious or legitimate flows. While these approaches are able to detect novel attacks, they suffer from

comparably high false-positive (or false negative depending on the model tuning) rate. Most research in the NIDS area currently delivers a combination of signature and anomaly detection and aims to improve the effectiveness and efficiency of intrusion detection. In the following paragraphs we are going to present a brief overview of selected (N)IDS systems [12].

MIDAS [13] is an early intrusion detection system which uses heuristic intrusion detection employing both anomaly and misuse detection. MIDAS builds on P-BEST expert system which is written in Lisp and its improved version was used also in other IDS systems - IDES [14] and NIDES [15]. The rule-based expert system uses alarm events generated by the system (such as attempts to run special `suid` commands or number of bad logins) which are matched against primary rules describing some predefined action corresponding to the intrusion detection. The secondary rules determine system reaction to alert, typically an alert to the operator. MIDAS is considered to be the first system employing misuse detection.

EMERALD [16] project was developed as for large enterprise networks and combines signature analysis with statistical profiling. EMERALD provides a real-time protection of the network services and infrastructure against external attackers. The analysis scheme of EMERALD is hierarchically layered and extensible by new monitoring modules. Using well-defined interfaces, EMERALD enables incorporation of third-party IDS engines. The analysis scheme is used for correlation of local analysis of results obtained at multiple lower-level monitoring modules by a more abstract upper layer.

IDA system [17] uses the concept of mobile agents to track intruders among various hosts involved in an intrusion attack. IDA uses sensors (e.g. log search) which are present at each target system, in combination with event-specific mobile agents that migrate across the network and gather the information about the attack from local sensors.

MINDS system [18] is an IDS system incorporating signature and anomaly detection suitable for host and network protection. As an input for traffic analysis it uses flow information which is aggregated in 10 minutes intervals. It extracts time-window and connection-window based features, from the traffic, to detect both fast and stealth attacks. The signature detection is used for protection against known threats. Anomaly detection uses a local outlier factor by which an unusual traffic is discovered. The human operator then decides which anomalous traffic is actually malicious and can use the captured information do define the signatures for subsequent traffic filtering.

SABER [19] combines both detection (anomaly and signature) and protection mechanisms. It incorporates a DoS-resistant network topology that shields valuable resources from the malicious traffic, and supports a business process migration to the safe location upon the attack. Real-time host-based IDS monitors program's use of Windows Registry and detects the actions of malicious software. The analogous approach has been used for file-based anomaly detection in Unix environment, when the IDS monitors a set of information about each file access. SABER is completed by autonomic software patching mechanism [20] which tries to automatically patch vulnerable software by identifying and correcting buffer overflows in the code, using the information from specialized honeypots operated alongside production machines.

The Achilles' heel of most intrusion detection and prevention systems is their efficiency and effectively. High number of false positives/negatives requires labor-intensive incident analysis and discourages the administrators from system deployment or use. It also makes the deployment of any automatic protection mechanism highly problematic, as it might cause more harm than good by filtering-out the legitimate traffic, or creating a false sense of security if tuned too liberally. In the next section, we present a selection of techniques from the multi-agent field that can help to alleviate the problem.

3 Relevant Multi-agent Techniques

The biggest threat of the computer network attacks is given by the fact that the attackers operate a collective of loosely coupled, highly autonomous, often collaborating, migrating (and reproducing) computational processes. The success of such collective attacks is given by the fact that they attack uniform computational processes and hardware elements, with identical widespread vulnerabilities. In order to protect the computational resources, information and network as such, we propose to exploit the techniques, algorithms and theoretical results available in the field of multi-agent system, a branch of computer science investigating the collective behaviors of collaborative as well as self-interested, autonomous, intelligent, computational processes.

3.1 Distributed Task Allocation

Balanced allocation of the monitoring process within the network as well as an efficient placement of the intrusion response processes need to be decided and reconfigured locally, in a peer-to-peer interaction among the network components. That is valid because there is a desire to limit centralized decision making processes and centralized collection of data in the network.

Distributed task allocation is a typical problem that is solved in its different variations in the multi-agent research communities for years now (e.g. [21]). The distributed task allocation algorithms are based on different auctioning approaches (e.g. English, Vickery, Dutch, Seal-bid, All-pay [22]), each having different properties in different environments. The most widely used approaches to distributed task allocation are based on the CNP (*contract-net-protocol*) [23] (based on single round Seal-bid auction), iterated CNP, its $OSCM$-CNP optimality improvements [24] and other combinatorial auctions. These techniques have been successfully used in a number of network oriented applications. In more complex task allocation and distributed planning scenarios, more elaborate negotiation techniques and protocols need to be used. Another CNP extension is the ECNP [25,26] protocol which allows partial bid, provisional accepts and refusals in order to decompose the task into subtasks that can be allocated to the agents in the system. Therefore, it merges planning and task allocation phases.

3.2 Adaptation Methods

The field of multi-agent system provides also very specialized approaches towards achieving run-time revisions of the collective of autonomous computational processes: agent migration and multi-agent reflection.

Agent Migration. Changing the location of the agents code and state (data), without any change of its identity is denoted as agent migration [27]. We consider two types of migration: (*i*) weak mobility – an agent migrates from one host machine to another by its own intention and (*ii*) strong mobility, when an agent is forced to migrate from one host machine to another. Both weak and strong agent mobility are suitable mechanisms for network intrusion observation and response as they enable physical reallocation of critical data and computation.

Multi-agent Reflection. Inspired by classical works in computational reflection [28], an ability of agent to understand its running program and to reflect its operation by runtime program modification is understood as a multi-agent reflection. We distinguish between three different types of reflection in multi-agent systems: (*i*) *Individual Reflection* takes into account only the strictly local, internal information: agent's knowledge, resources, and computational capacity. (*ii*) *Mutual Reflection* extends the individual reflection by **unilaterally** taking into account agent's knowledge about other agents, such as trust and reputation, resources, deduced or communicated long-term motivations and intentions. This knowledge is referred to as *social knowledge* [29]. (*iii*) *Collective Reflection* - a revision of agents' collective behavior is not an individual process, but an interaction between multiple agent's reflective layers. The collective reflection can be achieved either by: *(a)* introduction of a single reflective agent (e.g. a meta-agent) that monitors the community behavior and suggests individual agents to alter their behavior or *(b)* emergently by the collective of agents, each carrying out its specific cognitive/reflective reasoning. Unlike in the case of the mutual reflection, the agents update not only their own knowledge and algorithms, but they also influence the changes inside other agents.

Key use cases of the multi-agent reflection deployment are adaptation to changes, system resilience, code sharing, automated programme code assembly, runtime code exchange, run-time code alteration or various performance improvements [30].

3.3 Trust and Reputation Modeling

Trust modeling is the primary approach to the high-level security in the field of the multi-agent systems. Trust models use in the scope of the IDS is two-fold: to model the trustfulness of the individual traffic sources (e.g. applications), and to protect the IDS system against infiltration and misuse.

There are many definitions of trust – the one we will use was proposed by Marsh in [31]: Agent x trusts the agent y if "*Agent x expects that y will behave according to x best interests, and will not attempt to harm x*". In the open multi-agent systems with self-interested agents, it is necessary to (i) maintain knowledge regarding the trustfulness of one's collaborators, in order to avoid the untrustful agents and to (ii) motivate the agents to behave in a trustful manner.

Trust models [32,33,34,35] are efficient and specialized knowledge structures, that help the agents to accomplish the above goals by efficiently organizing the past experience of the agent and other relevant information [36]. In the current body of work, most systems are based on the following principles (or their stronger modifications): (*i*) The trustfulness of the partner agent is always associated with an *identity* of a particular agent. (*ii*) The trustfulness value associated with the partner depends on the past behavior of the agent in question, as well as the past experience of the other trusting

agents (i.e. reputation) and past experience with similar agents. A notable exception in this context is a model developed by Andreas Birk [37], where the trustfulness values are associated with a physical feature of the trusted agent (robot), rather than with its identity. (*iii*) The trustfulness of the partner may be wither general, aggregated for all roles and situations, or situational, depending on task and environment.

Current trust models use a variety of techniques to hold the trustfulness data: some models use a *probability estimate* based on the time-weighted past experience [37]. The REGRET [38] and other models [34] also provide a formal framework used to integrate the observations of the agent with the opinions of the others. However, a simple probabilistic approach seems to be insufficient as it fails to capture the uncertainty of the trust estimate.[1] Therefore, Yu and Singh [39] use a Dempster-Shafer theory to capture the uncertainty in the reputation opinions and Josang et al. [40] use subjective logic for the same goal. The concept can be further extended by use of linguistic categories [32] or by representing the trustfulness as a fuzzy number [35] or a probability distribution. On the other hand, the issues of memory and computational requirements can not be ignored and the efficiency of the inference process is one of the main criteria for the model selection.

Regarding the deployment of the model, two basic options are possible. Many, mainly older architectures [33] rely on a centralized model, where all agents in the system share a single trust model. While this architecture offers some advantages, like fast learning (crucial in large domains, e.g. eBay) and memory efficiency, it requires the agents to share their private knowledge with the reputation manager, requires communication both at the observation and query time and doesn't allow the specialization of the model for individual agent's needs. Therefore, recent architectures rely on the model where each agent maintains its own model, and can combine its own observations with the data obtained from the others by means of reputation sharing.

Both the centralized approach and reputation models can suffer from the disinformation, aimed to maliciously improve or harm a reputation of one or more of the evaluated agents. While such attacks can never be ruled out entirely, the trust model itself can be used to evaluate the trustfulness of other agents in the recommender role [39], making the repetitive false opinions easy to eliminate. This feature is especially important when the system is deployed in a potentially insecure environment, e.g. network under attack.

Most current trust models are well suited for high-level security, with clearly defined and authenticated actors. In the next section, we will discuss the necessary modifications of the trust models in the context of broader network security architecture.

4 Reference Architecture

This Section defines reference architecture that combines the properties of the current generation of the Intrusion Detection systems as presented in Section 2 and appropriate multi-agent techniques briefly outlined in Section 3. The goal of the proposed architecture is to provide a near-real time autonomous attack discovery and response, replacing the direct human decision-making by well defined policies regulating autonomous runtime system decisions. Such rapid response is necessary to counter the threats outlined

[1] Many of the cited models do provide the uncertainty as a separate value.

in Section 1.1. The architecture is distributed, based on a peer-to-peer cooperation of autonomous agents:

• **Detection and Reaction Agents (DRA)** are the intelligent core of the system. They are located on appropriate hosts or network elements in the protected network and are responsible for collaborative traffic analysis, attack detection and planning of the reaction. They incorporate the trust model (or similar technology) used for attack detection and classification. Their decision-making in the reaction phase is governed by user-defined policies.

• **Network Sensors (NS)** are specialized agents that observe the traffic on the network, perform the low level analysis and feature extraction and inform the DRA's about their observations. They may also directly detect known malicious traffic (by means of signature detection) and raise an alarm. If appropriate, they may be collocated with DRA's.

• **Host Sensors (HS)** are the agents that reside on the host and are able to raise an alarm when they suspect an intrusion attempt. This alarm is then used for evaluation of the past traffic towards the host by the DRA's.

• **Reconfigurable Network Elements (NE)** are used by DRA agents to implement the protection mechanism when a new kind of attack is detected.

The agents in the community operate in three conceptual phases. In the **first phase**, the NS agents observe the traffic on the network and report the features of the relevant observed network flows to the DRA. The DRA agents then use their trust model to classify unknown traffic by matching it with a reaction from the Host Sensors.

To perform the match in the **second phase** of the detection, the existing trust models must be considerably extended. The match with the current state of the art is much less obvious, as we seriously relax the trust model assumptions from Section 3.3. Instead of a well-defined agent identity, we must define the evaluated connection by its features (e.g. [18]). These features will be then used to model the identity of the observed connection in the same manner as the context of the trusting decision is modeled in [41], by using metric spaces for identity and context representation. Trustfulness values are therefore no longer relative to the real identities or individual connections, but are attached to the centroids of the clusters created in the metric space of the features. When a new connection appears, we use a clustering algorithm to either attach it to an existing cluster, or to create a new cluster altogether. To update the trustfulness of the individual clusters, we rely on a feedback of the HS agents. Once aggregated, the aggregated feedback from related HS is used as a new observation of connection trustfulness for all open and recent relevant connections. One of the important assumptions we make is that the protected system contains many heterogenous hosts with a variety of Host Sensors. Therefore, given the random propagation strategy of typical worm attacks, we shall be able to identify the attack on the hosts that are not vulnerable and are appropriately protected.

Each DRA maintains its own instance of the trust model, and can extend the basic shared feature space with unique features of its own. To communicate these values between the DRA's, the agents use a reputation mechanism in both the subscribe-inform and query mode. Such mechanism will ensure that once the attack is suspected by a

single DRA, its classification will spread through the system and allow the other DRA to react faster. On the other hand, they are still autonomous to react differently, reducing the risk of false positives.

In the **third phase**, occurring after the attack detection, the DRA's need to create its description from the model, typically using generalizing machine learning methods. The extracted classifier shall be simple enough to be re-inserted into the reconfigurable Network Elements.

Once the DRA's have generated the description, they need to react swiftly in order to protect the network. By its nature, the proposed mechanism detects most of the new attacks when a part of the network has already been compromised. Therefore, we need to ensure that all connections between any two hosts on the network (including the internet gateway) will be inspected using the filter with the generated description. If such goal can't be achieved for a part of the network, such part must be handled as a single system (and potentially considered unsafe). Taking the distributed decision where to deploy the filters and how to shape the network traffic to inspect most of the connections is a distributed task allocation problem, solvable by the methods from Section 3.1.

Once we have a patch for the exploited vulnerability (possibly generated automatically [20]), we can protect & clean the vulnerable hosts, while removing their protection at network level. This approach requires careful replanning, in order to be able to return to the nominal network configuration as fast as possible, without compromising the network security. Once the protection is no longer necessary, we only keep the generated filter rule on few specialized nodes for the future network-based detection of the external attacks of the same type.

Mechanism Protection. Trust modeling method proposed for attack detection also fulfills the requirement to protect the cooperation between the distributed nodes of the IDS against disinformation, false observations and system misuse in general – the compromised or faked node can be detected and eliminated from the community. Currently, some IDS concepts [42] include this functionality, but with a relatively low level of sophistication. The IDS infrastructure presents a very valuable target for future attacks [6] due to the high level of access rights and the fact that it can not be efficiently shielded from the suspicious traffic. Therefore, an efficient auto-protection mechanism is a necessary prerequisite of the deployment of the autonomous systems that we need to counter the worm-type attacks.

5 Conclusions and Future Directions

This contribution presents a focused overview of the current and relevant IDS systems and proposes the integration of several selected families of agent technologies to address the threat from the Worm-type attacks. We argue that any efficient response mechanism against worm epidemics must be automatic, without including the human reaction into the loop. Direct human supervision can be replaced by policies regulating the system operations [10].

The distributed system, as presented in our reference architecture (see Section 4), shall act in three phases: (i) **Observation**, when the network and host sensors gather the information. This phase relies on a web of heterogenous hosts and sensors. (ii) **Detection**, when the Detection/Reaction agents use the data as an input for their trust models and detect the attacks in the traffic. (iii) **Reaction**, when the DRA's use their trust models to extract a filtering rule for the attack and modify network devices in order to enforce the rule on the bulk of the network communication, preventing the further worm spread.

In order to implement the abstract architecture outlined above, we propose to integrate a wide range of agent techniques to address the inherently distributed nature of all phases of the problem. We also argue that these techniques will require additional development to match the challenging performance requirements of the network security field, and our text specifies several relevant areas and targets, most notably in the field of trust modeling.

In our current and future work, we are experimentally evaluating presented agent techniques in a simulated computer network [43], in order to validate the reference architecture and its implementation before the integration with the real network components.

Acknowledgment

We gratefully acknowledge the support of the presented research by Army Research Laboratory projects N62558-05-C-0028, N62558-07-C-0001 and N62558-07-C-0007.

References

1. Jansen, W., Mell, P., Karygiannis, T., Marks, D.: Mobile agents in intrusion detection and response. In: 12th Annual Canadian Information Technology Security Symposium, Ottawa, Canada (2000)
2. Pappalardo, D., Messmer, E.: Extortion via ddos on the rise (2005)
3. Yegneswaran, V., Barford, P., Ullrich, J.: Internet intrusions: global characteristics and prevalence. In: SIGMETRICS, pp. 138–147 (2003)
4. CERT: Overview of attack trends. Technical report (2002)
5. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: Inside the slammer worm. IEEE Security and Privacy 01, 33–39 (2003)
6. Shannon, C., Moore, D.: The Spread of the Witty Worm. Technical report, CAIDA - Cooperative Association for Internet Data Analysis (2004)
7. McHugh, J.: Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans. Inf. Syst. Secur. 3, 262–294 (2000)
8. Moore, D., Shannon, C., Voelker, G.M., Savage, S.: Internet quarantine: Requirements for containing self-propagating code. In: INFOCOM (2003)
9. Cooke, E., Jahanian, F., Mcpherson, D.: The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. In: Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI), pp. 39–44 (2005)

10. Sierhuis, M., Bradshaw, J., Acquisiti, A., van Hoof, R., Jeffers, R., Uszok, A.: Human-agent teamworks and adjustable autonomy in practice. In: Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS - NARA, Japan (2003)
11. : SNORT intrusion prevention system (Accessed in January 2007) (2007), http://www.snort.org/
12. Axelsson, S.: Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Chalmers Univ (2000)
13. Sebring, M.M., Shellhouse, E., Hanna, M.E., Whitehurst, R.A.: Expert systems in intrusion detection: A case study. In: Proceedings of the 11th National Computer Security Conference, Baltimore, Maryland, NIST, pp. 74–81 (1988)
14. Lunt, T.F., Tamaru, A., Gilham, F., Jagannathan, R., Jalali, C., Neumann, P.G., Javitz, H.S., Valdes, A., Garvey, T.: A real-time intrusion-detection expert system (ides). Technical report, SRI International (1992)
15. Anderson, D., Lunt, T.F., Javitz, H., Tamaru, A., Valdes, A.: Detecting unusual program behavior using the statistical component of the next-generation intrusion detection expert system (NIDES). Technical Report SRI-CSL-95-06, Computer Science Laboratory, SRI International, Menlo Park, CA (1995)
16. Porras, P.A., Neumann, P.G.: EMERALD: Event monitoring enabling responses to anomalous live disturbances. In: Proc. 20th NIST-NCSC National Information Systems Security Conference, pp. 353–365 (1997)
17. Asaka, M., Okazawa, S., Taguchi, A., Goto, S.: A method of tracing intruders by use of mobile agents. In: INET'99 (1999)
18. Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P.N., Kumar, V., Srivastava, J., Dokas, P.: Minds - minnesota intrusion detection system. In: Next Generation Data Mining, MIT Press, Cambridge (2004)
19. Keromytis, A.D., Parekh, J., Gross, P.N., Kaiser, G., Misra, V., Nieh, J., Rubenstein, D., Stolfo, S.: A holistic approach to service survivability. In: Proceedings of the 2003 ACM Workshop on Survivable and Self-Regenerative Systems (SSRS), pp. 11–22 (2003)
20. Sidiroglou, S., Keromytis, A.D.: Countering network worms through automatic patch generation. IEEE Security & Privacy 3, 41–49 (2005)
21. Walsh, W.E., Wellman, M.P.: A market protocol for distributed task allocation. In: In Third International Conference on Multiagent Systems, Paris (1998)
22. Sandholm, T.: Distributed Rational Decision Making. In: Weiss, G. (ed.) Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, pp. 201–258. MIT Press, Cambridge, MA (1999)
23. Smith, R.G.: The contract net protocol: High level communication and control in a distributed problem solver. IEEE Transactions on Computers C-29, 1104–1113 (1980)
24. Sandholm, T.W., Lesser, V.R.: Coalitions among computationally bounded agents. Artificial Intelligence 94, 99–137 (1997)
25. Perugini, D., Lambert, D., Sterling, L., Pearce, A.: Agent-based global transportation scheduling in military logistics. In: AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1278–1279. IEEE Computer Society, Washington, DC (2004)
26. Rehak, M., Pechoucek, M., Volf, P.: Distributed planning algorithm for coalition logistics in semi-trusted environment. In: DIS '06: Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06), pp. 265–272. IEEE Computer Society, Washington, DC (2006)
27. Suri, N., Carvalho, M.M., Bradshaw, J.M., Breedy, M.R., Cowin, T.B., Groth, P.T., Saavedra, R., Uszok, A.: Enforcement of communications policies in software agent systems through mobile code. In: POLICY, pp. 247–250 (2003)

28. Maes, P.: Computational reflection. Technical report 87-2, Free University of Brussels, AI Lab (1987)
29. Pěchouček, M., Mařík, V., Bárta, J.: Role of acquaintance models in agent's private and semi-knowledge disclosure. Knowledge-Based Systems, 259–271 (2006)
30. Foltýn, L., Tožička, J., Rollo, M., Pěchouček, M., Jisl, P.: Reflective-cognitive architecture: From abstract concept to self-adapting agent. In: DIS '06: Proceedings of the Workshop on Distributed Intelligent Systems, IEEE Comp. Soc. Los Alamitos (2006)
31. Marsh, S.: Formalising trust as a computational concept (1994)
32. Ramchurn, S., Huynh, D., Jennings, N.R.: Trust in multiagent systems. The Knowledge Engineering Review 19 (2004)
33. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artif. Intell. Rev. 24, 33–60 (2005)
34. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: An integrated trust and reputation model for open multi-agent systems. Journal of Autonomous Agents and Multi-Agent Systems 13, 119–154 (2006)
35. Rehák, M., Foltýn, L., Pěchouček, M., Benda, P.: Trust model for open ubiquitous agent systems. In: Intelligent Agent Technology, 2005 IEEE/WIC/ACM International Conference. Number PR2416, IEEE, Los Alamitos (2005)
36. Castelfranchi, C., Falcone, R., Pezzulo, G.: Integrating trustfulness and decision using fuzzy cognitive maps. In: Nixon, P., Terzis, S. (eds.) iTrust 2003. LNCS, vol. 2692, pp. 195–210. Springer, Heidelberg (2003)
37. Birk, A.: Boosting cooperation by evolving trust. Applied Artificial Intelligence 14, 769–784 (2000)
38. Sabater, J., Sierra, C.: Regret: reputation in gregarious societies. In: AGENTS '01: Proceedings of the fifth international conference on Autonomous agents, pp. 194–195. ACM Press, New York (2001)
39. Yu, B., Singh, M.P.: Detecting deception in reputation management. In: AAMAS '03, pp. 73–80. ACM Press, New York (2003)
40. Josang, A., Gray, E., Kinateder, M.: Simplification and analysis of transitive trust networks. Web Intelligence and Agent Systems 4, 139–162 (2006)
41. Rehak, M., Gregor, M., Pechoucek, M., Bradshaw, J.M.: Representing context for multiagent trust modeling. In: IEEE/WIC/ACM Intl. Conf. on Intelligent Agent Technology (IAT'06), pp. 737–746. IEEE Computer Society, USA (2006)
42. Janakiraman, R., Waldvogel, M., Zhang, Q.: Indra: A peer-to-peer approach to network intrusion detection and prevention. In: Proceedings of IEEE WETICE 2003 (2003)
43. Rehák, M., Pěchouček, M., Prokopová, M., Foltýn, L., Tožička, J.: Autonomous protection mechanism for joint networks in coalition operations. In: Knowledge Systems for Coalition Operations 2007, Proceedings of KIMAS'07 (2007)